RU EN ПРОГРАММНЫЕ СИСТЕМЫ: ТЕОРИЯ И ПРИЛОЖЕНИЯ ISSN 2079-3316
научная статья искусственный интеллект, интеллектуальные системы, нейронные сети

УДК 81'322
10.25209/2079-3316-2023-14-1-31-54

О декомпозиции метода построения энкодера языковой модели

Игорь Владимирович **Трофимов**✉
Институт программных систем им. А. К. Айламазяна РАН, Веськово, Россия
✉*itrofimov@gmail.com*

Аннотация. Энкодер в составе языковой модели является механизмом преобразования текстовой информации в *эффективное* числовое представление, пригодное для решения широкого круга задач обработки текста при помощи нейросетевых методов. В данной статье предложен способ декомпозиции процесса обучения языкового энкодера. Рассматриваются вопросы целесообразности такой декомпозиции с точки зрения снижения вычислительных затрат, контроля качества на промежуточных стадиях обучения, обеспечения интерпретируемости результатов каждой стадии. Приводятся оценки качества энкодера.

Ключевые слова и фразы: обработка естественного языка, нейронные сети, языковая модель, энкодер, контекстно-зависимые представления, разрешение лексической неоднозначности

Для цитирования: Трофимов И. В. *О декомпозиции метода построения энкодера языковой модели* // Программные системы: теория и приложения. 2023. **Т. 14**. № 1(56). С. 31–54. `https://psta.psiras.ru/read/psta2023_1_31-54.pdf`

Введение

Современные языковые модели развиваются преимущественно в экстенсивной манере – рост качественных показателей сопровождается ростом размера нейронной сети. Так, уже BERT [1] имел 340 млн. оптимизируемых параметров. Более поздние модели T5 [2] вышли на уровень 11 млрд., GPT-3 [3] – 175 млрд., Switch Transformer [4] – более триллиона.

Эксперименты с обучением и даже использованием таких сетей вычислительно затратны и требуют применения довольно мощных аппаратных средств. Это обстоятельство влечет увеличение стоимости и сроков исследований, проблематичной стала верификация SOTA-подходов (а также сравнение со state-of-the-art на нестандартных задачах), исследования фрагментируются, т.к. многогранное исследование становится слишком затратным.

Выполнение исследований с соответствующими моделями меньших размерностей не всегда оправданно, т.к. полученный в таком эксперименте результат будет (с большой долей вероятности) ниже, чем в эксперименте с полноценной моделью, что обесценивает саму идею такого эксперимента. В этой связи видятся актуальными исследования в направлении интенсификации – поиска новых решений в архитектуре и новых малозатратных подходов к обучению языковых моделей.

Гигантский размер языковой модели является проблемой и для ее практического применения, где существенным показателем становится время предсказания, а также имеются ограничения на объем оперативной и долговременной памяти. Исследователями был предложен ряд решений, таких как дистилляция, квантизация и усечение. Общая идея дистилляции сводится к обучению небольшой модели подражать поведению крупноразмерной, квантизация заключается в снижении точности представления числовой информации, а усечение осуществляется за счет исключения из большой модели части весов с низкой практической значимостью.

Замечательный современный обзор этих методов представлен в [5], который можно дополнить разве что исследованиями по квантизации (например, [6]) и еще одним оригинальным подходом к дистилляции [7]. Однако эти подходы не решают проблему *исследования* больших SOTA-моделей и методов на их основе.

С вычислительной точки зрения существенным недостатком в методологии обучения языковых моделей является то, что модели учатся целиком. В данной работе предлагается подход, предполагающий разбиение процесса обучения языковой модели на отдельные этапы, на каждом из которых

выполняется обучение только части весов. Ожидается, что такой метод позволит значительно сократить вычислительные затраты на обучение.

Статья состоит из двух разделов. В первом описан предлагаемый метод построения энкодера – рассматриваются общие архитектурные идеи, описаны методы создания статической модели и модели контекстуализации. Во втором разделе изложены результаты эксперимента по обучению энкодера и его оценке на задаче разрешения лексической неоднозначности.

1. Поэтапный метод построения энкодера

1.1. Общие замечания об архитектуре

Типовой энкодер на основе механизма трансформеров включает в себя следующие элементы:

- токенизатор, разбивающий строку текста на фрагменты (наибольшее распространение получили статистические подходы, такие как `SentencePiece` [8], `BPE` [9], `WordPiece` [10]);
- матрицу эмбеддингов для хранения «базовых» представлений каждого субтокена;
- механизм моделирования позиции субтокена в тексте;
- слои трансформеров для получения контекстуализованных (контекстно-зависимых) представлений субтокенов, кодирующих внутриязыковые и внутритекстовые связи между субтокенами.

Для обучения таких энкодеров распространен `MLM`-подход (от `masked language modeling`) [1], иногда в тандеме с декодером-генератором, обучаемым на различных языковых задачах.

Отталкиваясь от описанной типовой схемы, предложим ряд альтернативных идей, на основе которых может быть построена новая архитектура и подход к обучению (см. таблицу 1).

Архитектура предлагаемого энкодера и результаты его работы схематически изображены на рисунке 1.

В данной работе не рассматриваются верхние уровни, хотя необходимость их очевидна. Сосредоточимся на устройстве и обучении нижних уровней энкодера, результатом работы которых должно стать качественное представление *лексических значений слов* и *морфологической информации* с учетом разрешения неоднозначностей. Верхние уровни должны дополнить это представление позиционной информацией и закодировать внутритекстовые связи между токенами (синтактико-семантические, кореферентные и др.).

Таблица 1. Ключевые особенности предлагаемого подхода к построению энкодера языковой модели

Настоящее положение дел	**Предлагаемая альтернатива**
В современных подходах проблема неполноты словаря (`oov`-проблема) решается за счет субтокенизации вплоть до отдельных литер с дальнейшим восстановлением «значения» слова путем комбинирования «значений» субтокенов и контекстуализации. Таким образом, ситуация `out-of-vocabulary` возникает крайне редко.	Предлагается крупногранулярная токенизация, чтобы лексическое «значение» слова содержалось в токене в готовом виде и не требовало восстановления из фрагментов. Восстановление «значения» в нередких теперь `oov`-случаях предлагается осуществлять за счет контекстуализации. Так как у модели нет необходимости обучаться комбинированию субтокенов, слои контекстуализации могут быть компактнее.
Так как «значения» субтокенов тесно связаны с механизмом их комбинирования, затруднительно разделить процессы их обучения. Матрица эмбеддингов и слои трансформеров в языковых моделях учатся совместно.	Предлагается независимо обучать подмодель представления токенов быстрыми методами, такими как `word2vec` [11]. Затем обучать модель контекстуализации с «замороженной» подмоделью для токенов.
Обучение в соответствии с `MLM`-подходом опирается на предсказание замаскированных словарных единиц (субтокенов). Это ставит в прямую зависимость вычислительную эффективность и размер словаря. В языковых моделях обычно словарь делают умеренных размеров (например, в оригинальной `BERT` словарь включал 30 тыс. единиц).	При обучении контекстуализации предлагается в замаскированных позициях предсказывать не словарную единицу, а ее векторное представление (т. к. оно «заморожено»). То есть решать задачу регрессии. Размер словаря в этом случае почти не влияет на вычислительную эффективность. Это позволяет языковой модели иметь в готовом виде достаточно большой «словарный запас».
Векторные представления (токенов или субтокенов) не имеют какой-либо внутренней структуры и интерпретации, хотя и замечено, что механизм «внимания» в отдельных `head` трансформеров иногда отражает ту или иную теоретико-лингвистическую идею (синтаксическую связь, общность лексических значений и т.п.). Заранее неизвестно, какой `head` какой идее (а чаще смеси идей) научится. Это затрудняет осмысление выполняемых сетью операций.	Предлагается в явном виде моделировать разные виды лексико-семантической общности и закрепить их позиции в векторном представлении. А именно, предлагается моделировать близость значений в категориальном, ассоциативном и грамматическом аспектах.

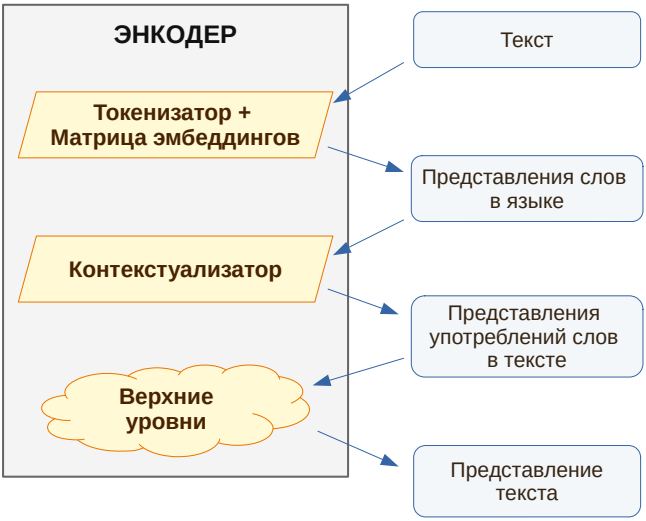

Рисунок 1. Общая схема энкодера

Первым шагом в создании и обучении такого энкодера будет построение статических[1] представлений слов в языке (матрицы эмбеддингов).

1.2. Представление слов в языке

Основанные на идеях дистрибутивной семантики методы получения статических представлений слов зарекомендовали себя как довольно эффективное средство при решении задач классификации текстов, информационного поиска, машинного перевода и др. Появление быстрых алгоритмов построения статических эмбеддингов (`skip-gram`), а также появление множества доступных готовых моделей позволило статическим представлениям стать ключевым компонентом в большом числе приложений, осуществляющих содержательную обработку текстовой информации (хоть и на непродолжительный период).

Очевидным недостатком статических подходов было отсутствие возможности эффективно моделировать «значение» многозначных слов.

[1]Будем пользоваться терминами *статическое* и *динамическое* представление для отсылки, соответственно, к контекстно-независимым представлениям (или представлениям слов в языке) и контекстно-зависимым представлениям (или представлениям употреблений слов в тексте). В литературе такое именование не получило широкого распространения, но, будучи компактным, оно более удобно для изложения. В англоязычной литературе для упоминания контекстно-зависимых представлений принято пользоваться термином contextualized representation (embedding).

Со временем для устранения этой проблемы была предложена идея динамических моделей – моделей, которые позволяли бы получать контекстно-зависимые представления «значений» для отдельных употреблений слов в тексте.

Общий принцип моделирования «значения» слова в статических моделях следующий. Всякое слово представляет собой точку в многомерном пространстве. Чем больше общность значений пары слов, тем ближе они должны быть размещены в упомянутом многомерном пространстве в соответствии с выбранной мерой близости (обычно угловой). Замечено, что основания для определения общности значений могут быть различны. В данной работе предлагается оперировать тремя видами общности на основе выделения категориального, ассоциативного и грамматического компонентов.

Подробный анализ и обзор по теме категориальной и ассоциативной близости значений содержится в [**12**]. Кратко перескажем основные положения (в терминах стоящих за словами понятий):

- Категориальное сходство отражает общность неотъемлемых (физических или функциональных) свойств сущностей. Например, значительную степень категориального сходства можно усмотреть в парах *автомобиль – велосипед, лошадь – корова.*
- Ассоциация отражает связь сущностей через отношение, ситуацию, тему. Ассоциативно связанные сущности объединяет принадлежность одной предметной области, одной типовой жизненной ситуации и т. п. (*автомобиль – водитель, автомобиль – бензин, лошадь – седло, лошадь – аллюр*).
    - К ассоциациям принято относить и «языковые» связи, обусловленные, например, идиоматичностью (*страница – истории*) или терминологичностью (*коробка – передач*).

Грамматическая общность основана на сходстве морфологических признаков слов. В отличие от категориальной и ассоциативной общности эта составляющая векторного представления не лежит в плоскости лексической семантики. Ее задача кодировать информацию о синтаксических свойствах слов.

Дистрибутивный подход позволяет строить качественные модели для категориальной и ассоциативной компоненты представления. Известно, что для моделирования категориальной близости синтаксический контекст предпочтительней линейно-оконного [**13**]. Ассоциации лучше обнаруживаются широким линейным окном. Конкатенацию категориальной и ассоциативной компоненты в дальнейшем будем называть лексико-семантическим представлением.

В данном исследовании для получения статических представлений слов использовался инструментарий `conll2vec`[2], способный порождать модель, включающую все три вышеперечисленные компоненты. В целом `conll2vec` можно охарактеризовать как функционально-насыщенную версию `word2vec`. Тезисно сформулируем ключевые особенности.

- Для построения векторной модели `conll2vec` опирается на размеченные текстовые данные (разметка включает морфологическую и синтаксическую информацию, а также результат лемматизации текста).
- Векторное представление состоит из трех частей. Категориальная компонента строится алгоритмом `skip-gram` на основе синтаксических контекстов; ассоциативная – модификацией `skip-gram` с одной матрицей (контекстом считаются все слова предложения); грамматическая – классификатором на основе морфологической разметки (подобно `skip-gram`, но вместо матрицы контекстов используется матрица представлений для отдельных граммем – логистическая регрессия по каждой граммеме).
- Векторная модель строится в два этапа: сначала представления лемм, затем представления словоформ.
- Имеется возможность использовать дополнительные ресурсы с информацией о близости значений, в том числе, векторные модели большей размерности.

Перечисленный функционал позволяет построить словарь `V` и соответствующую матрицу эмбеддингов `E`. Кроме того, инструментарий `conll2vec` способен:

- строить грамматическую компоненту представления для распространенных суффиксов слов (суффикс понимается как цепочка букв, на которую заканчивается слово: *-щемся, -емому, -лась* и т.п);
- частично группировать словоформы, относящиеся к одной «морфологической парадигме» (в кавычках, т.к. имеет место аналогия, а не строгое соответствие), – применяется эвристический алгоритм.

Этот дополнительный функционал позволяет реализовать идею ограниченной субтокенизации на основе следующих соображений.

Словарь векторной модели `V` разделяется на три части:

- словарь псевдооснов (для форм, которые удалось собрать в «морфологические парадигмы»);
- словарь суффиксов;

[2] https://github.com/parus-proj/conll2vec

- словарь словоформ (для форм, не вошедших в словарь псевдооснов; сюда попадают омоформы и формы не отвечающие критериям эвристического алгоритма).

Матрица эмбеддингов E преобразуется в две матрицы:

- матрицу лексико-семантических представлений;
- матрицу грамматических представлений.

Их соотношение со словарями отражено на рисунке 2. Эмбеддинги, относящиеся к словарю словоформ, образуются путем разделения исходного эмбеддинга на лексико-семантическую и грамматическую части. За эмбеддинг псевдоосновы принимается лексико-семантическое представление *леммы*.

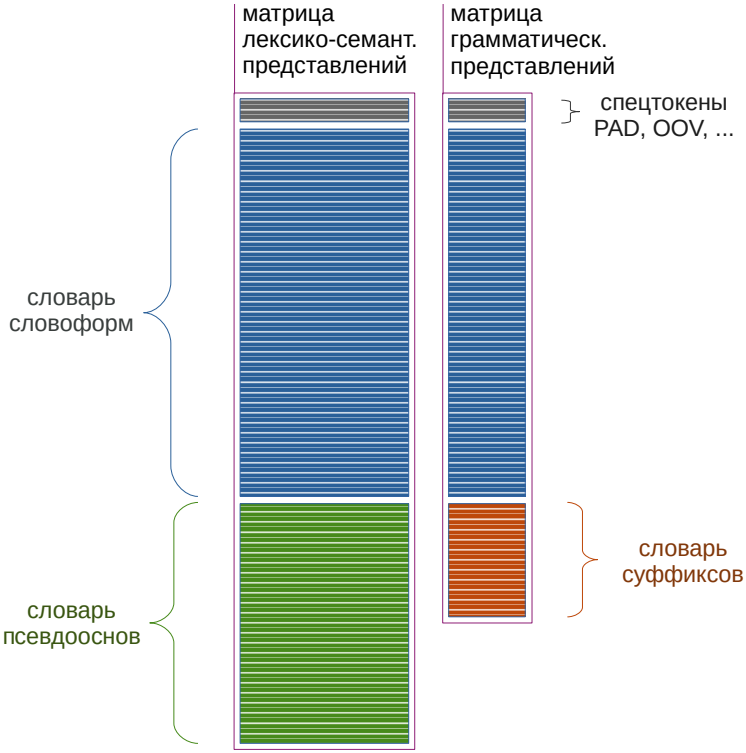

Рисунок 2. Матрицы эмбеддингов в предлагаемой модели энкодера

Токенизация реализуется следующим образом. Сначала текст делится на слова по пробельным символам. Затем для каждого слова токенизатор возвращает пару индексов $\langle \mathrm{i}_{\mathrm{LEX}}, \mathrm{i}_{\mathrm{GR}} \rangle$ (по одному для каждой матрицы

представлений) в соответствии со следующим алгоритмом.

- Если слово содержится в словаре словоформ, то индексы будут одинаковы и соответствовать лексико-семантической и грамматической частям представления для данного слова.
- Иначе ищем способы разделить слово на две части так, чтобы левой соответствовала словарная псевдооснова, а правой – словарный суффикс (или нулевой). Если это удается, то i_{LEX} будет индексом наиболее длинной псевдоосновы (среди обнаруженных вариантов), а i_{GR} будет найден как наиболее длинный словарный суффикс, присутствующий в данной словоформе. Например, слово *абонентская* может быть разделено на псевдооснову *абонентск-* и суффикс *-ая*, но если есть представление для более длинного суффикса, например, *-ская*, то предпочтение будет отдано этому (более информативному) представлению.
- Если не удастся найти псевдооснову, то i_{LEX} берется для специального представления `out-of-vocabulary` (нулевой вектор), а i_{GR} соответствует наиболее длинному из найденных словарных суффиксов. Если и суффикс не найден, то для грамматической части также берется `oov`-представление.

В сравнении с непосредственным использованием единственной матрицы эмбеддингов для словоформ предложенная субтокенизация имеет ряд преимуществ.

- Меньший объем памяти, необходимой для хранения матриц.
- Словарь суффиксов позволяет получить информативное грамматическое представление для большого числа несловарных слов.
- Словарь псевдооснов позволяет получить лексико-семантические представления для несловарных слов, относящихся к данной «морфологической парадигме».

Теперь перейдем к устройству контекстуализации.

1.3. Модель контекстуализации и ее обучение

Предлагаемый подход к контекстуализации опирается на следующие предположения:

- для разрешения лексической неоднозначности достаточно учесть тематическую информацию и ближайший контекст многозначного слова;
- для разрешения морфологической неоднозначности достаточно ближайшего контекста.

В основе модели контекстуализации лежит гибридная архитектура, сочетающая рекуррентные сети (для анализа ближайшего контекста) и

механизм внимания (для представления тематической информации). Компоненты представления (категор., ассоц. и грамм.) обрабатываются независимо, поэтому способы их контекстуализации могут быть различными. Рассмотрим общий подход к контекстуализации на примере категориальной части представления (после чего опишем особенности обработки других частей).

При обучении модели применяется `MLM`-подход. Для каждой текстовой позиции (за исключением `out-of-vocabulary`) модель учится предсказывать целевую компоненту исходного статического векторного представления за счет контекста. Будем иллюстрировать работу модели на примере следующего предложения:

(пр.-1) *Дети поели и пошли гулять.*

Пусть маскируемым (предсказываемым) словом будет союз 'и'[3] . Тогда прямой проход обучения и вычисление ошибки схематически можно изобразить так – рисунок 3.

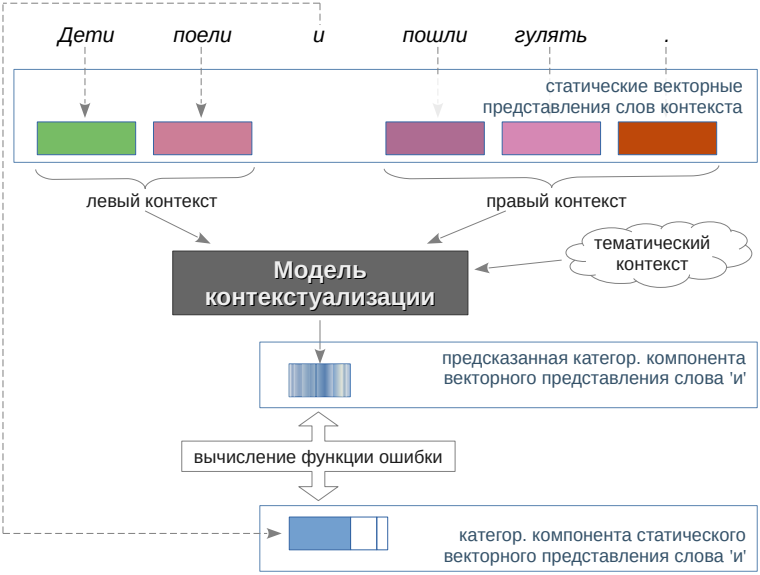

Рисунок 3. Обучение контекстуализации (предсказание векторного представления замаскированного слова)

[3]Заметим, что словоформа 'и' является категориально многозначной. Кроме употребления в роли союза 'и' может входить в состав сокращений (*и.о.*), быть инициалом (*В. И. Ленин*) или даже собственным именем (*Ван И* – министр иностранных дел КНР).

На вход контекстуализатора подаются левый и правый контексты замаскированного слова, а также представление тематического контекста (подробнее ниже). На выходе контекстуализатор стремится построить целевую компоненту векторного представления замаскированного слова. Величина ошибки вычисляется по формуле:

$$loss = CosLoss + \gamma \cdot MseLoss, \tag{1}$$

где $CosLoss$[4] – угловая ошибка, $MseLoss$ – среднеквадратичная ошибка, γ – коэффициент, подбираемый так, чтобы вклад среднеквадратичной ошибки в $loss$ был небольшим в сравнении с более важной угловой составляющей. Первое слагаемое отвечает за корректное угловое размещение предсказываемого вектора в пространстве статической модели, второе – побуждает не пренебрегать длиной вектора.

Внутреннее устройство части контекстуализатора, отвечающей за анализ ближнего контекста, показано на рисунке 4.

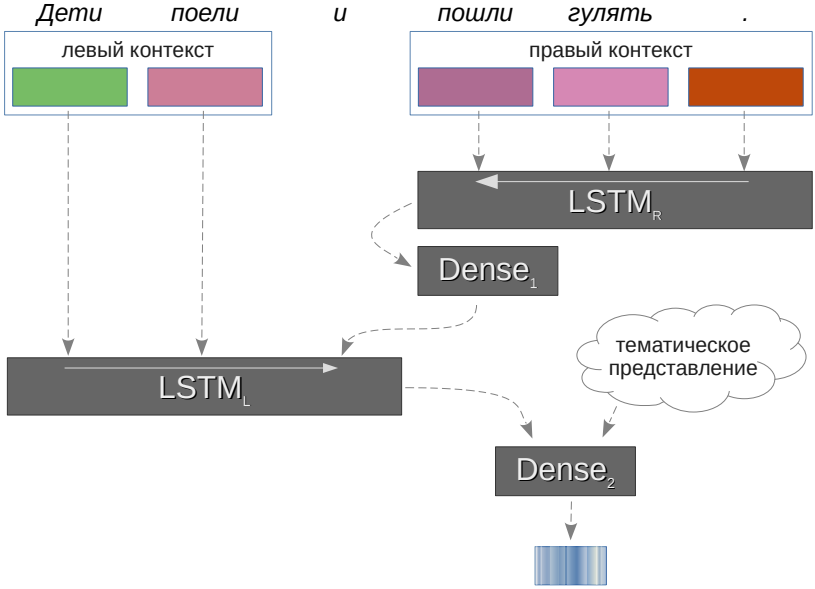

Рисунок 4. Схема обработки левого и правого контекстов в контекстуализаторе

[4] $CosLoss = -\cos(\vec{v}_1, \vec{v}_2)$, где $\vec{v}_1$, $\vec{v}_2$ – сравниваемые векторные представления.

Обработка[5] начинается с анализа правого контекста однослойной однонаправленной LSTM-сетью по направлению к замаскированному слову. На выходе LSTM_R порождается *представление правого контекста*, которое затем масштабируется при помощи однослойной FF-сети (Dense_1) до размерности входных векторов. После этого выполняется анализ левого контекста, за последний элемент которого принимается *представление правого контекста*. Для анализа вновь используется LSTM (LSTM_L). Полученное в результате представление для всего локального контекста (левого и правого) путем конкатенации объединяется с тематическим представлением (о котором чуть ниже) и подается на вход еще одного FF-слоя (Dense_2). На его выходе формируется представление замаскированного слова, выведенное на основе контекста.

Присоединение *представления правого контекста* ко входу LSTM_L позволяет сократить пространство гипотез при предсказании. Так, при рассмотрении левого контекста независимо от правого LSTM_L должна допускать очень разнородные продолжения текста:

- *Дети поели супа* ...
- *Дети поели горячего* (супа) ...
- *Дети поели наспех* (приготовленного супа) ...
- *Дети поели за* (кухонным столом) ...
- и т.д.

Информация из правого контекста может помочь LSTM_L блокировать часть подобных гипотез[6].

Тематическое представление является общим для всех слов одного предложения и формируется при помощи `attention`-механизма на базе ассоциативных компонент статических представлений. Коэффициенты внимания вычисляются не для токена целиком, а для каждого измерения в ассоциативной компоненте токена. За тематический вектор предложения принимается среднее ассоциативных компонент всех слов предложения,

[5]Для краткости здесь и далее опускаем вопросы регуляризации и соответствующие компоненты и гиперпараметры сети. Интересующиеся деталями реализации могут изучить исходные коды модели и скриптов ее обучения по адресу https://github.com/parus-proj/rue.

[6]Возможна реализация подобного информирования LSTM_R о левом контексте.

взвешенных согласно коэффициентам внимания:

$$t_s = \frac{1}{N}\sum_{i=1}^{N} \vec{a}_i \odot \vec{v}_i, \tag{2}$$

где t_s – тематическое представление для предложения s, $\vec{a}_i$ – вектор коэффициентов внимания для i-ого слова, $\vec{v}_i$ – вектор ассоциативной компоненты статического представления i-ого слова, N – количество слов в предложении.

Архитектура сети, вычисляющей коэффициенты внимания $\vec{a}_i$, изображена на рисунке 5 и представляет собой каскад из трех FF-слоев.

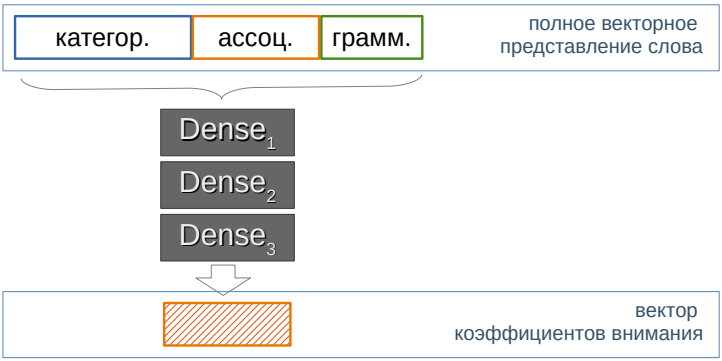

Рисунок 5. Вычисление коэффициентов внимания

Размерности ассоциативной компоненты на входе и вектора коэффициентов внимания на выходе одинаковы. Обучение этой сети выполняется только совместно с контекстуализатором ассоциативных частей[7].

Таким образом, для каждого слова в предложении вычисляется не просто его значимость для определения темы[8], а выполняется декомпозиция слова на признаки – каждый признак (измерение) в ассоциативной компоненте может вносить разный вклад в тематический вектор.

Выше была описана контекстуализация категориальной компоненты представления. Возвращаясь к вопросу специфики контекстуализации остальных частей, отметим следующее.

[7]При обучении контекстуализаторов для категориальной и грамматической компонент обратное распространение ошибки по этой ветви сети блокируется.

[8]Такой подход тоже приемлем. Очевидно, что, например, служебные слова практически не вносят вклада в формирование темы предложения. Поэтому даже простой индикатор, вроде IDF, мог бы здесь работать.

- Тематический вектор обучается только в рамках контекстуализатора для ассоциативной компоненты.
- В грамматическом контекстуализаторе вместо тематического вектора используется категориальная компонента предсказываемого слова с наложенной на него маской (`dropout` вектора с очень высоким коэффициентом зануления). Тема предложения практически бесполезна для предсказания морфологических признаков в какой-то конкретной текстовой позиции, а наличие категориальной информации делает контекстуализатор более точным.

После того как сформированы представления всех компонент слова, выполняется их конкатенация в том же порядке, в каком они размещены в статическом представлении.

Очевидно, что получить качественное (в значительное степени специфичное) представление только на основе контекста не всегда возможно. В этом нетрудно убедиться, рассмотрев следующее предложение с замаскированным словом X:

(пр.-2) *В цеху установили X станок.*

Здесь контекст допускает довольно разнообразные по значению подстановки: *деревообрабатывающий, новый, дорогостоящий, двухосный, малошумящий, безоператорный, 200-тонный, неоткалиброванный* и т.д. Выведенным на основе такого контекста представлением, вероятно, окажется малоинформативная смесь всех этих значений.

Вообще о самой идее контекстуализации можно заметить следующее:

- применительно к многозначному[9] слову контекстуализация уточняет представление;
- применительно к однозначному слову – размывает «значение».

Хороший контекстуализатор должен стремиться минимизировать размытие.

В предлагаемой здесь модели, чтобы ослабить эффект размытия, выполняется смешивание исходного статического представления и представления, выведенного сугубо из контекста. Для этого используется формула экспоненциального скользящего среднего:

$$R_{dynamic} = \alpha \cdot R_{ctx} + (1 - \alpha) R_{static}, \tag{3}$$

[9] `Out-of-vocabulary` считается частным случаем многозначного слова.

где $R_{dynamic}$ – окончательное контекстуализованное представление слова, R_{ctx} – представление, выведенное только на основе контекста, R_{static} – статическое представление слова в языке, α – коэффициент сглаживания ($\alpha \in [0, 1]$). Важно отметить, что в ходе обучения контекстуализатора на MLM-задаче смешивание отключено ($R_{dynamic} = R_{ctx}$). Оно включается лишь после того, как контекстуализатор будет способен порождать R_{ctx}.

Каждая словарная единица имеет собственный набор коэффициентов сглаживания для каждой компоненты представления (категор., ассоц. и грамм.). Иными словами:

- каждому лексико-семантическому представлению (строке одноименной матрицы на рисунке 2) соответствует два коэффициента сглаживания (категор. и ассоц.);
- каждому грамматическому представлению – один коэффициент.

Для получения коэффициентов сглаживания организуется особый этап обучения модели (коэффициенты будут обучаемыми весами в этой модели). Архитектура нейросети временно расширяется до схемы энкодер–декодер (рисунок 6). Сеть обучается предсказанию текста исходного предложения в

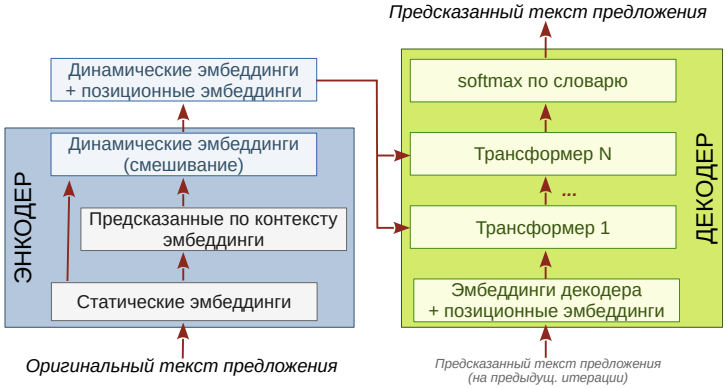

Рисунок 6. Архитектура для обучения коэффициентов сглаживания

условиях, когда словари энкодера и декодера различны. `SentencePiece`-словарь декодера строится на основе тех же текстовых данных, которые ранее использовались для `MLM`-обучения. Сам декодер реализуется на базе трансформерной архитектуры. Его особенностью является лишь то, что механизм внимания, обращенный в сторону энкодера, имеет только одну голову (хотя традиционно он `multi-head`). Это связано с тем,

что выходные вектора энкодера имеют предопределенную структуру, разбивать которую произвольным образом лишено смысла. В качестве функции ошибки используется перекрестная энтропия (`categorical crossentropy loss`).

Во время обучения коэффициентов сглаживания в энкодере замораживаются не только статические эмбеддинги, но и веса, отвечающие за формирование представления на основе только лишь контекста (R_{ctx}). Неиспользовавшаяся ранее логика смешивания статических и выведенных из контекста представлений включается в вычислительный процесс. Порождаемое энкодером представление дополняется генерируемой позиционной информацией (путем конкатенации).

В результате обучения происходит «настройка» коэффициентов сглаживания так, чтобы динамические эмбеддинги стали наиболее информативны для декодера. При этом декодер следует делать не слишком емким, для того чтобы ему затруднительно было восстанавливать текст предложения без качественных эмбеддингов от энкодера (т.е. за счет собственных весов).

2. Эксперименты

2.1. Построение модели

Экспериментальная модель была обучена на данных корпуса русского литературного языка `PaRuS` [14]. Корпус содержит около 2.5 млрд. токенов. Текстовые данные в нем сегментированы, выполнены морфологическое и синтаксическое аннотирование, а также лемматизация.

Статические представления строились инструментарием `conll2vec`. Для эксперимента мы ограничились небольшими размерностями: 60 измерений для категориальной компоненты, 40 для ассоциативной и 20 для грамматической. Исследование статической модели на тестсетах для оценки семантической близости (`RUSSE-15` [15] и `RuSim1000` [12]) показало отсутствие *существенного* роста качества лексико-семантических представлений при дальнейшем увеличении размерности.

Построение статической модели включало следующие шаги: создание словарей, обучение лексико-семантических представлений для лемм (10 эпох), преобразование модели лемм в модель словоформ, обучение грамматических представлений (4 эпохи), преобразование словарей

и матриц к окончательному виду (группировка в «морфологические парадигмы»).

Наличие шумов в данных потребовало установить довольно высокие частотные пороги при формировании словарей. В результате были получены:

- словарь словоформ объемом 250 тыс. элементов;
- словарь псевдооснов – 40 тыс.;
- словарь суффиксов – 3 тыс. элементов.

Гиперпараметры модели контекстуализации были следующими:

- размер контекстного окна – по 5 токенов в каждом направлении;
- размерности LSTM – 768, 256 и 256 (для категориальной, ассоциативной и грамматической компонент, соответственно);
- размерности FF-слоев для порождения тематического представления – 256, 128 и 40.

Для обучения коэффициентов смешивания использовались следующие гиперпараметры:

- для кодирования позиционной информации использовалось 30 измерений (способ кодирования аналогичен `BERT`);
- в декодере использовалось 2 слоя трансформеров (размерность представлений 128, голов 8, размерность FF-слоев 256);
- словарь декодера на 10 тыс. входов построен алгоритмом `SentencePiece`.

В общей сложности обучение статической модели продолжалось около 10 часов. При этом необходимо принять во внимание, что для обучения использовался сервер с двумя `CPU Intel Xeon E5540` – пиковая производительность данной установки равна 162 `GFLOPS`[10]. Для сравнения современные `CPU` high-end-сегмента имеют пиковую производительность 2–5 `TFLOPS` (например, десктопный `Intel Xeon W-3175X`). Таким образом, при использовании более современного оборудования построение статической модели заняло бы на порядок меньше времени на `CPU` и, вероятно, еще меньше при адаптации кодов `conll2vec` к `GPU`[11].

[10]Всего 8 ядер с частотой 2.53 ГГц и 8 FLOP одинарной точности за такт.

[11]Пиковая производительность `GPU Nvidia Tesla A100` составляет 19.5 `TFLOPS` (операций одинарной точности), `Tesla V100` $>$ 15 `TFLOPS`, `Tesla P100` $>$ 9 `TFLOPS`.

Модель контекстуализации обучалась на 6 Nvidia Tesla A100. Собственно контекстуализация (без коэффициентов смешивания статических и динамических представлений) обучалась в течение 3 часов, коэффициенты смешивания – в течение 2 часов.

В общей сложности обучение модели заняло 15 часов (из них 10 часов на несовременном оборудовании). Для сравнения $\text{BERT}_{\text{LARGE}}$ ее авторы обучали на 64 TPU в течение 4 дней.

2.2. Оценка модели

В качестве тестовой задачи для оценки построенного энкодера была выбрана задача разрешения лексической неоднозначности (WSD), а именно, разрешения неоднозначности для существительных, способных выражать событийное и несобытийное значение [16, 17]. Как и в указанных работах оценка производилась на шести тестовых словах. Для каждого слова в тестовом множестве представлено 100 примеров событийного значения и столько же несобытийного (в сумме 1200 примеров). Таким образом, random baseline по точности на данном тестсете составляет 50%.

Эксперимент повторял исследование с BERT, описанное в [17], за тем исключением, что вместо RuBERT [18] использовался предлагаемый энкодер. Таким образом, на «косвенных» данных тренировался классификатор, состоящий из энкодера (с замороженными весами), 128-мерного FF-слоя и softmax-слоя с двумя выходами. Для обучения использовался меньший объем данных, чем в эксперименте с RuBERT (тренировочное множество состояло из 368 тыс. примеров, валидационное – из 12 тыс.). Результат оценки классификатора и сравнение с аналогичным RuBERT-решением приведены в таблице 2 (для исследуемого энкодера приводится среднее в пяти экспериментах и СКО).

Из результатов оценки можно сделать следующие выводы.

- Существенное превосходство макроусредненной точности над random baseline говорит о работоспособности контекстуализации. Применение статических представлений в таком классификаторе дало бы ровно 50% точности.
- Энкодер уступает RuBERT в точности на этой задаче. Однако следует учитывать, что количество параметров в энкодере более чем в три раза меньше, чем в RuBERT.

Таблица 2. Результат оценки построенного энкодера на задаче разрешения лексической неоднозначности, точность в % (accuracy)

Тестовое слово	**Исследуемый энкодер**	**RuBERT из [17]**
организация	80.1 ± 1.2	**89.5**
отделение	$\mathbf{86.8 \pm 0.9}$	85.4
отопление	$\mathbf{58.9 \pm 3.5}$	51.4
публикация	50.5 ± 0	**77.3**
стройка	70.8 ± 2.1	**73.9**
управление	78.1 ± 2.4	**90.9**
macro-average	70.8	**78.1**

Таким образом, эксперимент показал лишь работоспособность подхода в целом. Для достижения более высоких показателей требуется дальнейшее совершенствование методологии и более тщательная настройка гиперпараметров энкодера.

3. Заключение

Несмотря на то, что исследованная модель энкодера пока не демонстрирует впечатляющих результатов на практических задачах, показана принципиальная возможность такого пути построения модели и работоспособность подхода в целом. Подход привлекателен тем, что методология может независимо совершенствоваться на разных уровнях: от методов построения статической модели до высокоуровневых надстроек над контекстуализацией. Преимуществами нового метода являются вычислительная эффективность за счет декомпозиции процесса обучения и естественная интерпретируемость промежуточных результатов. Последнее обеспечивается за счет явного выделения видов близости значений лексических единиц, а также за счет явного различения представлений слов в языке и в тексте.

Автор выражает искреннюю благодарность ЦКП «Центр данных ДВО РАН» [19] *за предоставление вычислительных ресурсов для выполнения экспериментов.*

Список литературы

[1] Devlin J., Chang M-W., Lee K., Toutanova K. *BERT: Pre-training of Deep Bidirectional Transformers for language understanding.*– 2018.– 16 pp. arXiv 1810.04805

[2] Raffel C., Shazeer N., Roberts A., Lee K., Narang Sh., Matena M., Zhou Ya., Li W., Liu P. J. *Exploring the limits of transfer learning with a unified text-to-text transformer.*– 2020.– 67 pp. arXiv 1910.10683

[3] Brown 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 Ch., Chen M., Sigler E., Litwin M., Gray S., Chess B., Clark J., Berner Ch., McCandlish S., Radford A., Sutskever I., Amodei D. *Language models are few-shot learners.*– 2020.– 75 pp. arXiv 2005.14165

[4] Fedus W., Zoph B., Shazeer N. *Switch transformers: scaling to trillion parameter models with simple and efficient sparsity.*– 2021.– 40 pp. arXiv 2101.03961

[5] Kolesnikova A., Kuratov Y., Konovalov V., Burtsev M. *Knowledge distillation of Russian language models with reduction of vocabulary*, Proceedings of the International Conference «Dialogue 2022» (Moscow, June 15–18, 2022), Computational Linguistics and Intellectual Technologies.– vol. **21**.– 2022.– ISBN 978-5-7281-3205-9.– Pp. 295–310 . arXiv 2205.02340

[6] Zafrir O., Boudoukh G., Izsak P., Wasserblat M. *Q8bert: Quantized 8bit bert*, 2019 Fifth Workshop on Energy Efficient Machine Learning and Cognitive Computing – NeurIPS Edition (EMC2-NIPS) (13 December 2019, Vancouver, BC, Canada).– 2019.– Pp. 36–39. arXiv 1910.06188

[7] Clark K., Luong M.-T., Le Q. V., Manning Ch. D. *Electra: Pre-training text encoders as discriminators rather than generators.*– 2020.– 18 pp. arXiv 2003.10555

[8] Kudo T., Richardson J. *SentencePiece: A simple and language independent subword tokenizer and detokenizer for Neural Text Processing.*– 2018.– 6 pp. arXiv 1808.06226

[9] Sennrich R., Haddow B., Birch A. *Neural machine translation of rare words with subword units.*– 2016.– 11 pp. arXiv 1508.07909

[10] Schuster M., Nakajima K. *Japanese and Korean voice search*, 2012 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP) (25–30 March 2012, Kyoto, Japan).– 2012.– Pp. 5149–5152.

[11] Mikolov T., Chen K., Corrado G., Dean J. *Efficient estimation of word representations in vector space.*– 2013.– 12 pp. arXiv 1301.3781

[12] Trofimov I. V., Suleymanova E. A. *A syntax-based distributional model for discriminating between semantic similarity and association*, Proceedings of the International Conference «Dialogue 2017».– V. 1, Computational Linguistics and Intellectual Technologies.– vol. **16**.– 2017.– Pp. 349–359.

[13] Трофимов И. В., Сулейманова Е. А. *Дистрибутивно-семантическая модель для выявления категориального сходства* // Программные системы: теория и приложения.– 2018.– Т. **9**.– № 4(39).– С. 443–460.

[14] Власова Н. А., Трофимов И. В., Сердюк Ю. П., Сулейманова Е. А., Воздвиженский И. Н. *PaRuS — синтаксически аннотированный корпус русского языка* // Программные системы: теория и приложения.– 2019.– Т. **10**.– № 4(43).– С. 181–199.

[15] Panchenko A., Lukashevich N. V., Ustalov D., Paperno D., Meyer K. M., Konstantinova N. *RUSSE: The first workshop on Russian semantic similarity*, Proceedings of the International Conference «Dialogue 2015».– V. 2, Computational Linguistics and Intellectual Technologies.– vol. **14**.– RGGU.– 2015.– Pp. 89–105.

[16] Трофимов И. В., Сулейманова Е. А., Власова Н. А., Подобряев А. В. *Разрешение событийно-несобытийной неоднозначности существительных* // Программные системы: теория и приложения.– 2018.– Т. **9**.– № 4(39).– С. 3–33.

[17] Трофимов И. В., Сердюк Ю. П., Сулейманова Е. А., Власова Н. А. *Разрешение событийно-несобытийной неоднозначности существительных: нейросетевой подход* // Программные системы: теория и приложения.– 2020.– Т. **11**.– № 4(47).– С. 31–53.

[18] Kuratov Yu., Arkhipov M. *Adaptation of deep bidirectional multilingual transformers for Russian language.*– 2019.– 8 pp. arXiv:1905.07213

[19] Сорокин А. А., Макогонов С. В., Королев С. П. *Информационная инфраструктура для коллективной работы ученых Дальнего Востока России* // Научно-техническая информация. Серия 1: Организация и методика информационной работы.– 2017.– № 12.– С. 14—16.

Поступила в редакцию 13.11.2022;
одобрена после рецензирования 17.01.2023;
принята к публикации 09.02.2023;
опубликована онлайн 19.02.2023.

Рекомендовал к публикации *д.ф.-м.н. Н. В. Лукашевич*

Информация об авторе:

Игорь Владимирович Трофимов

старший научный сотрудник Исследовательского центра искусственного интеллекта ИПС им. А. К. Айламазяна, специалист по технологиям автоматической обработки естественного языка, извлечения информации, автоматического планирования.

0000-0002-6903-4730

e-mail: *itrofimov@gmail.com*

Автор заявляет об отсутствии конфликта интересов.

EN RU PROGRAM SYSTEMS: THEORY AND APPLICATIONS ISSN 2079-3316
Research Article ARTIFICIAL INTELLIGENCE, INTELLIGENCE SYSTEMS, NEURAL NETWORKS

UDC 81'322
10.25209/2079-3316-2023-14-1-31-54

Decomposition of construction method for a language encoder

Igor Vladimirovich **Trofimov**
Ailamazyan Program Systems Institute of RAS, Ves'kovo, Russia
✉*itrofimov@gmail.com*

Abstract. An encoder as part of a language model is a mechanism for converting text information into an *effective* numerical representation which is suitable for solving a wide range of text processing tasks by means of neural network methods. This paper suggests a way of decomposing of the learning process for a language encoder. The author considers the issues of expediency of such decomposition taking into account reduction of computational costs, quality control at intermediate training stages, provision of the interpretability of the results on each stage. The quality evaluation of the encoder is given. (*In Russian*).

Key words and phrases: natural language processing, neural networks, language model, encoder, context-sensitive representations, lexical ambiguity resolution

2020 *Mathematics Subject Classification*: 68T07; 68T50

For citation: Igor V. Trofimov. *Decomposition of construction method for a language encoder*. Program Systems: Theory and Applications, 2023, **14**:1(56), pp. 31–54. (*In Russ.*). https://psta.psiras.ru/read/psta2023_1_31-54.pdf

References

[1] J. Devlin, M-W. Chang, K. Lee, K. Toutanova. *BERT: Pre-training of Deep Bidirectional Transformers for language understanding*, 2018, 16 pp. arXiv 1810.04805

[2] C. Raffel, N. Shazeer, A. Roberts, K. Lee, Sh. Narang, M. Matena, Ya. Zhou, W. Li, P. J. Liu. *Exploring the limits of transfer learning with a unified text-to-text transformer*, 2020, 67 pp. arXiv 1910.10683

[3] T. B. Brown, B. Mann, N. Ryder, M. Subbiah, J. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, S. Agarwal, A. Herbert-Voss, G. Krueger, T. Henighan, R. Child, A. Ramesh, D. M. Ziegler, J. Wu, C. Winter, Ch. Hesse, M. Chen, E. Sigler, M. Litwin, S. Gray, B. Chess, J. Clark, Ch. Berner, S. McCandlish, A. Radford, I. Sutskever, D. Amodei. *Language models are few-shot learners*, 2020, 75 pp. arXiv 2005.14165

[4] W. Fedus, B. Zoph, N. Shazeer. *Switch transformers: scaling to trillion parameter models with simple and efficient sparsity*, 2021, 40 pp. arXiv 2101.03961

[5] A. Kolesnikova, Y. Kuratov, V. Konovalov, M. Burtsev. "Knowledge distillation of Russian language models with reduction of vocabulary", Proceedings of the International Conference "Dialogue 2022" (Moscow, June 15–18, 2022), Computational Linguistics and Intellectual Technologies, vol. **21**, 2022, ISBN 978-5-7281-3205-9, pp. 295–310 . arXiv 2205.02340

[6] O. Zafrir, G. Boudoukh, P. Izsak, M. Wasserblat. "Q8bert: Quantized 8bit bert", 2019 Fifth Workshop on Energy Efficient Machine Learning and Cognitive Computing – NeurIPS Edition (EMC2-NIPS) (13 December 2019, Vancouver, BC, Canada), 2019, pp. 36–39. arXiv 1910.06188

[7] K. Clark, M.-T. Luong, Q. V. Le, Ch. D. Manning. *Electra: Pre-training text encoders as discriminators rather than generators*, 2020, 18 pp. arXiv 2003.10555

[8] T. Kudo, J. Richardson. *SentencePiece: A simple and language independent subword tokenizer and detokenizer for Neural Text Processing*, 2018, 6 pp. arXiv 1808.06226

[9] R. Sennrich, B. Haddow, A. Birch. *Neural machine translation of rare words with subword units*, 2016, 11 pp. arXiv 1508.07909

[10] M. Schuster, K. Nakajima. "Japanese and Korean voice search", 2012 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP) (25–30 March 2012, Kyoto, Japan), 2012, pp. 5149–5152.

[11] T. Mikolov, K. Chen, G. Corrado, J. Dean. *Efficient estimation of word representations in vector space*, 2013, 12 pp. arXiv 1301.3781

[12] I. V. Trofimov, E. A. Suleymanova. "A syntax-based distributional model for discriminating between semantic similarity and association", Proceedings of the International Conference "Dialogue 2017". V. 1, Computational Linguistics and Intellectual Technologies, vol. **16**, 2017, pp. 349–359.

[13] I. V. Trofimov, E. A. Sulejmanova. "A dependency-based distributional semantic model for identifying taxonomic similarity", *Program Systems: Theory and Applications*, **9**:4(39) (2018), pp. 443–460 (in Russian).

[14] N. A. Vlasova, I. V. Trofimov, Yu. P. Serdyuk, E. A. Sulejmanova, I. N. Vozdvizhenskij. "PaRuS — syntax annotated Russian corpus", *Program Systems: Theory and Applications*, **10**:4(43) (2019), pp. 181–199 (in Russian).

[15] A. Panchenko, N. V. Lukashevich, D. Ustalov, D. Paperno, K. M. Meyer, N. Konstantinova. "RUSSE: The first workshop on Russian semantic similarity", Proceedings of the International Conference "Dialogue 2015". V. 2, Computational Linguistics and Intellectual Technologies, vol. **14**, RGGU, 2015, pp. 89–105.

[16] I. V. Trofimov, E. A. Sulejmanova, N. A. Vlasova, A. V. Podobryaev. "Disambiguation between eventive and non-eventive meaning of nouns", *Program Systems: Theory and Applications*, **9**:4(39) (2018), pp. 3–33 (in Russian).

[17] I. V. Trofimov, Yu. P. Serdyuk, E. A. Sulejmanova, N. A. Vlasova. "Eventive vs. non-eventive sense of nouns: disambiguation using neural network approach", *Program Systems: Theory and Applications*, **11**:4(47) (2020), pp. 31–53 (in Russian).

[18] Yu. Kuratov, M. Arkhipov. *Adaptation of deep bidirectional multilingual transformers for Russian language*, 2019, 8 pp. arXiv:1905.07213

[19] A. A. Sorokin, S. V. Makogonov, S. P. Korolev. "The information infrastructure for collective scientific work in the Far East of Russia", *Scientific and Technical Information Processing*, **44**:4 (2017), pp. 302–304.