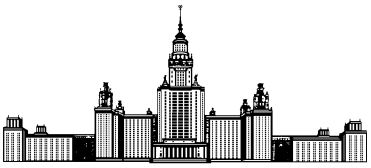
Московский государственный университет имени М. В. Ломоносова



Факультет Вычислительной Математики и Кибернетики
Кафедра Математических Методов Прогнозирования

Выпускная квалификационная работа

ТЕМАТИКО-СТИЛИСТИЧЕСКИЕ ВЕКТОРНЫЕ ПРЕДСТАВЛЕНИЯ ТЕКСТОВЫХ ПОЛЬЗОВАТЕЛЬСКИХ ЗАПРОСОВ

Выполнил:
студент 417 группы
Скачков Николай Андреевич

Научный руководитель:
д.ф-м.н., доцент
Воронцов Константин Вячеславович

Москва, 2019

Содержание

1 Вступление **2**

2 Постановка задачи **3**

3 Обзор подходов **4**

3.1 Тематическое моделирование 4

3.2 Авторегрессионные модели 5

3.3 Нейросети с вниманием 7

3.4 Современные подходы 10

4 Описание метода **12**

4.1 Аппроксимация глобального софтмакса 12

4.2 Архитектуры моделей 14

5 Эксперименты **15**

5.1 Данные 15

5.2 Настройка тематической модели 16

5.3 Обучение нейросетевых моделей 17

6 Анализ результатов **19**

6.1 Сравнение подходов 19

6.2 Кластеризация пользовательских запросов 21

6.3 Построение классификатора научных запросов 22

7 Заключение **24**

1 Вступление

Построение семантических векторных представлений для текстов — это получившая большое распространение в последнее время задача перевода текстов естественного языка в численные представления определённой, фиксированной длины.

Эти вектора чисел предполагается использовать в качестве признаковых описаний текста: с их помощью предполагается определять тематику текста, определять семантическую близость текстов, определять тональность текстов и решать другие известные задачи обработки текстов естественного языка.

Изначально для решения этой задачи использовались исключительно тематические модели [1], [2], которые объединяют слова в темы на основе совстречаемости их в одних документах. Тематические модели для каждого текста строят векторное представление — вероятностное распределение текста по темам. Однако тематические не используют информацию о порядке слов в тексте, что затрудняет синтаксический и стилистический анализ текстов, а также известна проблема тематических моделей при работе с текстами небольшой длины [3].

В последнее время активно стал развиваться подход моделирования языка с помощью нейросетевых архитектур [4]. Было показано, что предобученные архитектуры улучшают качество решения задач обработки текстов. Сложные нейросетевые архитектуры способны учитывать семантические, стилистические и синтаксические особенности текстов. Однако языковые модели учитывают связи слов внутри предложения, но не текстов между собой.

Чтобы решить эту проблему стали использоваться подходы, изначально предназначенные для обучения векторных представлений слов на основе совстречаемости слов в одном окне [5]. Для обучения моделей предложений вместо окна рассматривались ближайшие слева и справа предложения [6]. Модели, основанные на информации из контекста предложения стали успешно применяться для обучения векторных представлений текстов для задач определения семантического сходства предложений и фраз.

В данной работе показано, с какими проблемами можно столкнуться при обучении контекстных векторных представлений предложений стандартными способами. Также предложен способ борьбы с этими проблемами в виде специальной функции потерь, показана эффективность ее использования на реальных данных.

Также в работе продемонстрирована эффективность использования обученных векторных представлений в задачах кластерного анализа данных и классификации текстов на основе человеческой разметки.

2 Постановка задачи

Перейдём к формальной постановке решаемой задачи.

Дана выборка пользовательских запросов в систему автоматического перевода на английском языке. Когда пользователь заходит в систему, он начинает пользовательскую сессию, внутри которой им формируются запросы.

Таким образом, для каждого текстового запроса пользователя известен идентификационный номер сессии, в которой она была задана.

Необходимо построить векторные представления пользовательских запросов, которые бы передавали стилистические и тематические особенности запросов и были бы схожими для запросов одного «типа».

Полученные векторные представления планируется использовать для следующих целей:

1. анализ пользовательской аудитории, кластеризация запросов;
2. обучение классификаторов на определенные типы запросов с помощью человеческой разметки.

Как несложно заметить, понятие типа не определено, что усложняет постановку задачи, так как неизвестно на каком уровне детализации нужно эти типы определять и сколько их.

Однако можно выделить важное свойство этих типов, которое поможет нам сформулировать способ и критерий решения задачи: у запросов одной пользовательской сессии должны быть схожие стилистические и тематические свойства. Такое требование можно выдвинуть, исходя из того, что пользователи внутри одной сессии чаще всего пользуются системой перевода с одной общей целью или берут предложения для перевода из одного источника.

В таком случае задача построения стилистико-тематических векторных представлений сводится к задаче классификации принадлежности пар пользовательских запросов одному контексту со следующим решающим правилом:

$$f(q_1, q_2) = \mathbb{1}\big(\rho(h(q_1), h(q_2)) < \varepsilon\big),$$

где $h(\cdot)$ — функция проекции текстов в пространство векторных представлений $\mathcal{H}$, а q_1, q_2 — пара запросов, $\rho(\cdot)$ — некоторая функция близости векторов в пространстве $\mathcal{H}$, $\mathbb{1}(\cdot)$ — функция-индикатор.

Решающее правило выбирается именно таким образом, чтобы заставить обучаемую нами функцию проекции $h(\cdot)$ для запросов из одного контекста выучивать близкие векторные представления.

Проекцию текстового запроса q в векторное представление $h(q)$ впоследствии будем обозначать h_q.

Сформулируем теперь критерий качества решения нашей задачи. Так как сама задача классификации принадлежности пар запросов одному контексту не является нашей основной целью, будем использовать относительную метрику качества.

Для запросов u, v рассмотрим множество $\{q_i\}_{i=1}^K$ из K случайных пользовательских запросов и введём функцию $\text{rank}(u, v)$ — номер позиции расстояния $\rho(h_u, h_v)$ среди отсортированных по убыванию расстояний $\{\rho(h_u, h_{q_i})\}_{i=1}^K$. Данная функция показывает, насколько векторные представления пары (u, v) близки относительно случайных. Для случайной пары запросов, значение этой функции для разных выборок будет колебаться около $K/2$.

Тогда определим качество векторных представлений в задаче предсказания принадлежности пар запросов одной сессии как следующую величину:

$$\mathcal{L}\big(h(\cdot)\big) = \mathbb{E}_{(q_1,q_2)\in\mathcal{B}}\big[\text{rank}(q_1, q_2)\big], \tag{1}$$

где $\mathcal{B}$ — множество пар запросов из одной пользовательской сессии.

Данная величина показывает, насколько вектор запроса из той же сессии в среднем ближе чем вектор случайного запроса. Отметим, что для случайного преобразования $h(\cdot)$, значении метрики качества $\mathcal{L}\big(h(\cdot)\big) = K/2$.

Данную функцию качества (1) в наших экспериментах будем называть ранговой близостью.

3 Обзор подходов

В данном разделе будут описаны основные подходы к решению задачи классификации принадлежности одному контексту пар текстовых запросов.

Кроме описания подходов, также здесь будут представлены лучшие на данный момент модели, решающие данную задачу.

3.1 Тематическое моделирование

Тематическое моделирование — это метод выявления семантической структуры в корпусе текстов на основе совстречаемости слов в документах. Рассмотрим постановку задачи тематического моделирования.

Дана коллекция документов D, и словарь W всевозможных слов. Каждый документ d представляется в виде «мешка слов», то есть порядок слов внутри документа не имеет значения, а считается лишь количество вхождения каждого слова в документ n_{dw}.

Введём параметры модели: вероятность принадлежности слова w теме t:

$$\varphi_{wt} = p(w|t)$$

и вероятность принадлежности темы t документу d:

$$\theta_{td} = p(t|d)$$

Тогда вероятность встречи слова w в документе d можно моделировать следующим образом:

$$p(w|d) = \sum_{t \in T} p(w|t)\, p(t|d),$$

где T — множество всех тем в модели. Данное равенство выполняется внутри предположения, что $p(w|t) = p(w|t, d)$.

Тогда, решая задачу максимального правдоподобия, получаем:

$$\mathcal{L}(\varphi, \theta) = \sum_{d \in D} \sum_{w \in W} n_{dw} \log \sum_{t \in T} \varphi_{wt} \theta_{td} \longrightarrow \max_{\varphi, \theta}, \tag{2}$$

при ограничениях:

$$\sum_{w \in W} \varphi_{wt} = 1; \quad \sum_{t \in T} \theta_{td} = 1.$$

Данная постановка задачи известна как PLSA [1]. Решение системы (2) не единственно, поэтому в [7] был предложен подход с добавлением регуляризаторов, улучшающих качество построения тематических моделей.

Заметим, что в нашей постановке задачи сессии пользователей можно рассматривать как документы, а на момент предсказания принадлежности двух запросов одной сессии, использовать вектора распределений θ_{tq_1}, θ_{tq_2} для проверки близости текстов. Вероятность принадлежности запросов одному контексту моделируется в следующем виде:

$$p\big((q_1, q_2) \in \mathcal{B}\big) = \sum_{t \in T} \theta_{tq_1}\, \theta_{tq_2} = \theta_{q_1}^T\, \theta_{q_2}$$

3.2 Авторегрессионные модели

Гипотеза «мешка слов» хорошо себя показывает на длинных документах, но при моделировании коротких фраз порядок слов имеет большое значение. Для учитывания порядка

слов в тексте используются нейросетевые модели, приспособленные для работы с последовательностями. Эти архитектуры используются в современных подходах к решению задачи классификации пар текстов, поэтому опишем сразу их основную суть.

Долгое время лучшими в задачах моделирования последовательностей считались рекуррентные нейронные сети, в частности, LSTM [8]. Последние получили распространение из-за возможности использования зависимостей между далёкими в тексте словами за счёт механизма долгой памяти.

Также данные архитектуры хорошо подходят для задачи языкового моделирования [4], которая заключается в предсказании слова по словам, находящимся слева от него в тексте. В статье [9] было показано, что предобученные нейросетевые языковые модели улучшают качество решения различных задач обработки текстов естественного языка.

Языковое моделирование основывается на том, что текст на естественном языке читается слева направо. Модели реализующие это свойство называются авторегрессионными.

Авторегрессионность определяется как условная зависимость слова от предыдущих слов текста, которую можно записать следующим образом:

$$p(s) = \prod_{i=1}^{N} p(w_i | w_{<i}),$$

где s — текст длины N слов, w_i — i-ое его слово, а $w_{<i}$ — множество слов, предшествующих слева i-ому слову.

В модели SkipThought [6] был предложен подход использования авторегрессионных моделей для моделирования вероятности совстречаемости соседних предложений. Вероятность предложения при условии контекста s_c моделировалась следующим образом:

$$p(s|s_c) = \prod_{i=1}^{|s|} p(w_i | w_{<i}, s_c),$$

где $|s|$ — это длина предложения s. В данном подходе, к обычной условной зависимости слова от предыдущих слов, предполагаемой в авторегрессионных моделях, добавляется зависимость от контекстного предложения.

При обучении этой модели максимизировался логарифм правдоподобия, складывающийся из произведения вероятностей всех предложений при условии своих левых и правых соседей:

$$\mathcal{L}(\theta) = \sum_i \log p_\theta(s_{i+1}|s_i) + \sum_i \log p_\theta(s_i|s_{i+1}) \longrightarrow \max_\theta$$

Вероятность $p_\theta(s|s_c)$ моделировалась с помощью многослойной LSTM [8] как показано на рисунке 1. Сначала рекуррентно строится векторное представление контекстного

предложения, затем рекуррентно этот вектор используется для предсказания слова и модифицируется для предсказания следующих слов предложения.

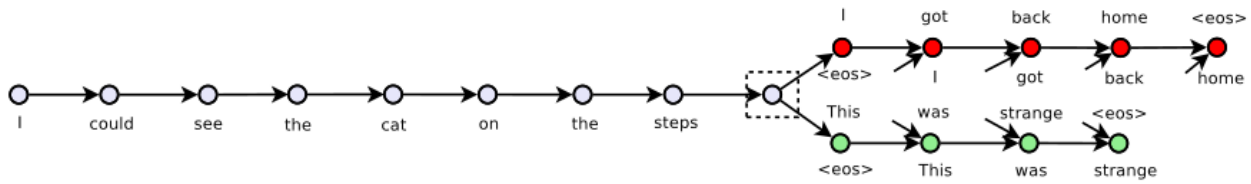


Рис. 1: Авторегрессионное моделирование предложения при условии контекста в модели Skip Thought.

Данную модель можно использовать в задаче классификации принадлежности пар предложений одному контексту, так как у близких предложений значение $p(s|s_c)$ будет выше. В статье [10] было показано, что данная модель даёт в этой задаче высокое качество.

3.3 Нейросети с вниманием

Как было сказано, LSTM получили своё распространение за счёт возможности моделирования связей между далёкими в тексте словами за счёт механизма долгой памяти.

Однако в статье [11] было показано что качество моделирования этих связей можно значительно улучшить, используя механизм *внимания*. Суть этого метода заключается в использовании контекстного вектора c_i, являющегося взвешенной суммой векторов слов контекста h_j:

$$c_i = \sum_{j<i} \alpha_{ij} h_j,$$

где веса a_{ij} вычислялись с помощью нейросети, которая предсказывала насколько значимым является слово w_j в контексте слова w_i. Данная архитектура показала значительно лучшие результаты в задаче языкового моделирования.

Можно заметить, что механизм внимания сам по себе позволяет моделировать взаимосвязи между словами, поэтому необходимость в требовании авторегрессионности модели отпадает.

Эта идея была реализована в модели Transformer [12], которая показала наилучший результат в задаче машинного перевода, а также в других задачах обработки текстов естественного языка [9], [15].

Каждый слой в архитектуре Transformer состоит из слоя внимания и двухслойной полносвязанной нейросети. При этом изначально входные слова проецируются в пространство

эмбеддингов и к ним добавляется вектор, кодирующий позицию слова в тексте. Общая архитектура сети представлена на Рис. 2.

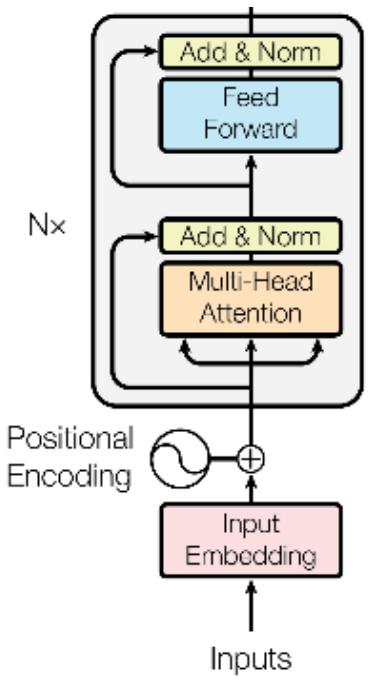


Рис. 2: Общая архитектура кодировщика нейросети Transformer.

Рассмотрим теперь подробнее как устроен механизм внимания в архитектуре Transformer.

Пусть h_i — это векторное представление i-го слова на каком-то слое нейросети. Пусть размерность векторных представлений равна D. Возьмём линейные преобразования вектора h_i: q_i, k_i и v_i. Будем называть их запросом, ключом и значением соответственно:

$$q_i = W_Q\, h_i; \quad k_i = W_K\, h_i; \quad v_i = W_V\, h_i,$$

где матрицы W_Q, W_K, W_V — размера $P \times D$, где число P — какая-то константа.

Тогда результатом *головы внимания* назовём следующую величину:

$$\text{head}_i = \sum_j \alpha_{ij}\, v_j,$$

где j принимает все значения от 1 до длины предложения, а α_{ij} — это веса внимания, вычисляемые по следующим формулам:

$$\alpha_{ij} = \frac{\exp(k_j^T\, q_i)}{\sum_l \exp(k_l^T\, q_i)}.$$

По смыслу это означает следующее. Матрица W_Q при умножении на вектор даёт нам запрос на поиск информации в других словах, а матрица W_K даёт ключ поиска, по которому информация находится в других словах. Вектора h_i, h_j, под воздействием этих матриц дают веса внимания. Далее вектора всех слов предложения под действием матрицы W_V, которая при умножении на вектор достаёт из него нужную информацию, суммируются в соответствии с найденными весами. Подробно данная процедура представлена на Рис. 3 слева.

На каждом слое внимания, к вектору h_i независимо применяются несколько голов внимания, а результаты их работы конкатенируются. Чтобы сохранить размерность пространства после слоя внимания, размерность головы внимания P выбирается так, чтобы после конкатенации всех голов размерность была снова равна D. Подробно данная процедура представлена на Рис. 3 справа.

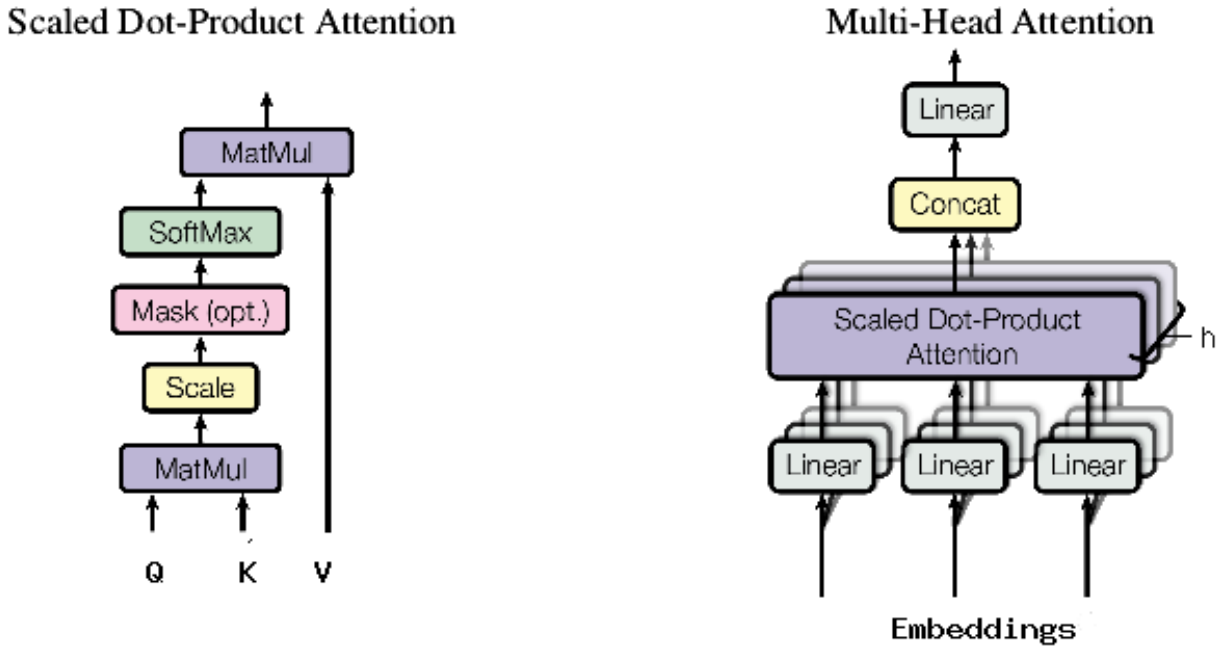


Рис. 3: Механизм внимания в модели Transformer.

Использование нескольких голов предпочтительнее, чем использование одной, так как это приём существенно понижает количество параметров в слое: с $O(D^2)$ до $O(PD)$.

По смыслу данная архитектура позволяет на каждом слое контексту влиять на каждое векторное представление каждого слова, обогащая его смыслом. На выходе сети Transformer выдается по одному контекстному вектору на каждое слово входа.

Обучаются современные нейросетевые архитектуры методом стохастического градиентного спуска [13] и его надстройками, например, Adam [14]. Суть этих методов заключается в том, чтобы вычислить градиент функции потерь по параметрам нейросети и изменить эти параметры в соответствии с направлением вычисленного антиградиента для того, чтобы уменьшить значение функции потерь.

3.4 Современные подходы

Как было сказано, требование рекуррентности в современных моделях скорее излишнее, поэтому большинство современных моделей в архитектурном плане представляют из себя Transformer.

Так, например, модель Universal Sentence Embeddings (USE) [15] представляет из себя обученную на наборе различных задач архитектуру, переводящую предложение в векторное представление фиксированного размера.

Способ обучения и функции потерь данной архитектуры не были описаны в работе, но был представлен список задач и текстовых корпусов, на которых она была обучена. Среди них можно выделить следующие задачи:

- анализ тональности отзывов;
- предсказание принадлежности пары предложений одному контексту;
- предсказание смысловой близости предложений;
- классификация запросов в вопросно-ответной системе.

В [10] было показано, что USE является лучшей на данный момент моделью в задаче предсказания близости пар предложений, а также предобученная модель находится в открытом доступе. Поэтому именно эту модель мы будем использовать в для основного сравнения.

Рассмотрим теперь современные подходы к обучению векторных представлений предложений. Основываясь на идее использования слов из контекста для выучивания векторных представлений слов, предложенной в [5], в [17] был предложен подход, позволяющий выучивать векторные представления на основе контекстной информации уже для любых объектов. Единственным требованием является определения контекста в обучающих данных.

Функция потерь в общем виде записывается следующим образом:

$$\log \mathcal{L}_\theta = \sum_{(a,b)\in\mathcal{B}} \sum_{(a,b_i^-)\notin\mathcal{B}} \log \mathcal{L}\Big(\rho\big(h_\theta(a), h_\theta(b)\big), \big\{\rho\big(h_\theta(a), h_\theta(b_i^-)\big)\big\}_i\Big), \tag{3}$$

где $\mathcal{B}$ — множество пар объектов из одного контекста; $h_\theta(\cdot)$ — некоторый кодировщик; $\rho(\cdot)$ — функция близости, $\mathcal{L}(\cdot)$ — функция потерь на одном объекте с одним объектом релевантным данному и несколькими нерелевантными.

Пусть функция близости задаётся как вероятность встретить тексты в одном контексте и моделируется следующим образом:

$$\rho(h_{s_1}, h_{s_2}) = p((s_1, s_2) \in \mathcal{B}) = p_{s_1 s_2} = \sigma(h_{s_1}^T h_{s_2}),$$

где $\sigma(\cdot)$ — сигмоидная функция. Тогда логарифм функции потерь для одного объекта можно записать в следующем виде:

$$\log \mathcal{L}_\theta(s_1, s_2) = \mathbb{1}\big((s_1, s_2) \in \mathcal{B}\big) \log p_{s_1 s_2} + \mathbb{1}\big((s_1, s_2) \notin \mathcal{B}\big) \log\big(1 - p_{s_1 s_2}\big). \tag{4}$$

Данная функция называется бинарная перекрёстная энтропия (БПЭ). Она поощряет уменьшение вероятности совстречаемости в случае, если предложения не из одного контекста и, наоборот, увеличение, если из одного. Данный выбор функции потерь является стандартным для задачи бинарной классификации пар объектов на принадлежность одному контексту [5], [9].

Если подставить в качестве функции потерь для одного объекта бинарную перекрёстную энтропию, тоё общую функцию потерь(3) можно записать в следующем виде:

$$\log \mathcal{L}_\theta = \sum_{(s, s_+) \in \mathcal{B}} \log p_{s s_+} + \sum_{i=1}^{K} \mathbb{E}_{(s, s_-) \sim \overline{\mathcal{B}}} \log(1 - p_{s s_-}). \tag{5}$$

Первая сумма поощряет увеличение близости между объектом s и объектом из его контекста, а вторая сумма поощряет увеличение дальности от того же объекта s до случайного. Вычислительно, математическое ожидание во второй сумме оценивается сэмплированием одного случайного объекта. Этот приём называется негативным сэмплированием [5], а число K определяет количество негативных сэмплов в модели.

Функция потерь, записанная в виде (5) практически совпадает с функцией потерь модели Word2Vec [5], если вместо совстречаемости слов внутри окна рассматривать совстречаемость предложений в одном рядом друг с другом в одном контексте.

Ещё одной популярной функцией потерь является функция потерь на триплетах. Триплет функция потерь на тройке объект a, положительный пример p, отрицательный пример n задаётся следующей формулой:

$$\mathcal{L}(a, p, n) = \max\big(0, \rho(h_a, h_p) - \rho(h_a, h_n) + 1\big), \tag{6}$$

где в качестве функции дальности ρ, часто берется евклидово расстояние. Работу данной функции потерь мы будем рассматривать далее.

4 Описание метода

Опишем теперь подробнее подход, который будем применять для решения задачи классификации пар запросов на принадлежность одной пользовательской сессии.

4.1 Аппроксимация глобального софтмакса

Вернемся еще раз к функции потерь (5), которая, как мы показали, напоминает функцию потерь модели Word2Vec [5].

Изначально в модели Word2Vec оптимизировался следующий функционал:

$$\log \mathcal{L}_\theta = -\sum_{(s,s_+)\in\mathcal{B}} \log \frac{\exp(h_s^T\, h_{s_+})}{\sum_{s_-\in W} \exp(h_s^T\, h_{s_-})}, \tag{7}$$

где сумма в знаменателе идёт по всевозможным словам s_- из словаря W, а s, s_+ — слова из одного контекстного окна.

Заметим, что записанная формула является логарифмом софтмакс-функции по всему словарю, где софтмакс-функция определяется покомпонентно для вектора значение z_i как распределение следующего вида:

$$\operatorname{softmax}\big(\{z_i\}_i\big)_j = \frac{\exp(z_j)}{\sum_i \exp(z_i)}$$

Необходимость вести оптимизацию по всему словарю вычислительно неэффективна и эта проблема решалась аппроксимацией (5) формулой с сэмплингом негативных примеров.

В нашей задаче в формуле (7) s, s_+ — это текстовые запросы из одной пользовательской сессии, а W — это множество всех запросов, которое, вообще говоря, может быть сколь угодно большим.

Функцию потерь вида (7) будем называть глобальным софтмаксом. Рост глобального софтмакса осуществляется за счёт одновременно роста близости с положительными примерами и увеличения дальности до всех негативных, что полностью соответствует цели нашей задачи. Однако оптимизация глобального сорфтмакса невозможна, так как количество вычислений пропорционально размеру обучающей выборки, размер которой не ограничен.

Аппроксимация глобального софтмакса (5) заставляет для каждого объекта независимо сэмплировать несколько негативных примеров. В процедуре обучения методом стохастического градиентного спуска, размер подвыборки, на которой считается градиент ограничен размерами оперативной памяти вычислительного устройства. Если для каждого положительного примера генерируется 5 негативных примеров, то количество положительных

объектов в этой подвыборке составляет 20%. Так как все негативные примеры выбираются случайно, получается, что информативность такой подвыборки падает, и обучение модели становится менее эффективным.

В случае больших моделей, размер подвыборки, подающейся на вход кодировщику на каждой итерации обучения, достаточно небольшой. Если доля положительных примеров в нём достаточно мала, то процедура может стать крайне неэффективной.

Попробуем вывести аппроксимацию глобального софтмакса (7), максимально эффективно использующую вычислительные ресурсы.

Для этого заменим сумму потерь на отдельных парах математическим ожиданием потерь на случайных подвыборках пар объектов $B \in \mathcal{B}$ размера K. Все переходы выделены цветом:

$$\log \mathcal{L}_\theta = -\sum_{(s,s_+)\in\mathcal{B}} \log \frac{\exp(h_s^T h_{s_+})}{\sum_{s_-\in W} \exp(h_s^T h_{s_-})} = -\frac{|\mathcal{B}|}{K}\mathbb{E}_B \left[\sum_{(s,s_+)\in B} \log \frac{\exp(h_s^T h_{s_+})}{\sum_{s_-\in W} \exp(h_s^T h_{s_-})} \right]$$

В нашей модели ошибка для пары объектов несимметрична. Симметричность функции потерь модели относительно пар объектов достигается за счёт того, что для любых двух текстов:

$$(s, s') \in \mathcal{B} \Longrightarrow (s', s) \in \mathcal{B}.$$

Следовательно пары текстов в B упорядочены. Будем для удобства называть вторые тексты в парах положительными примерами для первых, а множество всех положительных примеров в подвыборке B обозначим B_+.

Заметим теперь, что:

$$\sum_{s_-\in W} \exp(h_s^T h_{s_-}) \approx \frac{|W|}{|B|} \sum_{s_-\in B_+} \exp(h_s^T h_{s_-})$$

Данное приблизительное равенство становится строгим при больших размерах выборки B. Мы пренебрежем неточностью, так как данная замена существенно сократит нам число вычислений, и сделаем замену:

$$\begin{aligned}
\log \mathcal{L}_\theta &= -\frac{|\mathcal{B}|}{K}\mathbb{E}_B \left[\log \frac{\exp(h_s^T h_{s_+})}{\sum_{s_-\in W} \exp(h_s^T h_{s_-})} \right] \\
&= -\frac{|\mathcal{B}|}{K}\mathbb{E}_B \left[\log \frac{\exp(h_s^T h_{s_+})}{\frac{|W|}{|B|}\sum_{s_-\in B_+} \exp(h_s^T h_{s_-})} \right] \\
&= -\frac{|\mathcal{B}|}{K}\mathbb{E}_B \left[\sum_{(s,s_+)\in\mathcal{B}} \log \frac{\exp(h_s^T h_{s_+})}{\sum_{s_-\in B_+} \exp(h_s^T h_{s_-})} \right] + C(\theta).
\end{aligned}$$

Итак, мы получили функцию потерь в следующем виде:

$$\log \mathcal{L}_\theta = -\frac{|\mathcal{B}|}{K}\mathbb{E}_B\left[\sum_{(s,s_+)\in B} \log \frac{\exp(h_s^T\, h_{s_+})}{\sum_{s_-\in B_+}\exp(h_s^T\, h_{s_-})}\right] \longrightarrow \max_\theta . \tag{8}$$

В данном выражении нет излишнего сэмплирования отрицательных примеров, так как в качестве отрицательных примеров берутся положительные примеры из других пар.

Выражение (8) будем называть *аппроксимацией глобального софтмакса на случайных подвыборках* (АГСНП). Основной смысл полученной формулы можно сформулировать так: данная аппроксимация совпадает с глобальным софтмаксом в случае, если бы тексты приходили к нам онлайн, небольшими группами.

Также можно заметить, что данная аппроксимация является частным случаем общей функции потерь, описанной формулой (3).

4.2 Архитектуры моделей

Как было уже сказано, в качестве кодировщика $h(\cdot)$ может использоваться любая функция, переводящая текст в векторное представление фиксированной длины.

В качестве основной модели мы будем учить архитектуру, которая показала себя лучше всего в таких задачах, а именно Transformer. Устройство архитектуры уже было описано ранее и в проведённых экспериментах не было добавлено никаких модификаций.

Однако один важный вопрос пока не был затронут: как из последовательности векторных представлений, которые даёт Transformer получить один вектор — представление запроса.

В архитектуре USE [15] эту проблему решили следующим образом: векторные представления слов на выходе сети суммировали и делили на корень из длины предложения. Авторы мотивировали это тем, что деление на корень сохраняет зависимость вектора от длины текста, но при этом делает эту зависимость не слишком ярко выраженной.

В наших экспериментах будет сделано для сравнения также, но кроме этого будет использован более разумный способ агрегации векторных представлений слов — это *агрегация с вниманием*.

Под агрегацией с вниманием будет подразумеваться полноценный слой внимания, являющийся частью архитектуры Transformer, в котором в качестве запроса используется фиксированный единичный вектор. Таким образом, выходом этого слоя является единственный вектор, являющийся конкатенацией результатов голов внимания, каждая из которых, в свою очередь, выбирает из каких векторных представлений слов что нужно брать с помощью весов.

Также для того, чтобы проверить влияние сложности модели на качество решения задачи, будем учить более простые архитектуры, которые не моделируют связи между словами в предложении. В качестве такой упрощённой архитектуры возьмем следующую сеть. Сначала слова w_i предложения проецируются в векторное представление a_i^0, а затем эти векторные представления усредняются в вектор h^0. Затем полученный вектор несколько раз прогоняется через полносвязанные слои (W^j, b^j) со сквозными связями.

Записать это можно следующим образом:

$$\begin{aligned} a_i &= W^0_{w_i}; \\ h^0 &= \frac{1}{N} a_i; \\ h^j &= f(W^j\, h_{j-1} + b^j) + h_{j-1}, \end{aligned}$$

где $f(\cdot)$ — какая-то нелинейная функция активации.

Данная архитектура похожа на глубинные усредняющие сети [16] и считается простой но гибкой архитектурой. Простота архитектуры заключается в том, что данная архитектура никак не учитывает порядок слов в предложении, а имеет информацию только об их наличии. Данная особенность значительно ускоряет время работы сети, но может сказаться на качестве решения задачи.

За счёт глубины данная архитектура позволяет словам давать нелинейный вклад в векторное представление текста, что при всей простоте делает данную архитектуру достаточно гибкой.

В наших экспериментах все описанные архитектуры будут обучаться с одними и теми же функциями потерь, будут сравнены друг с другом и с предобученными моделями.

5 Эксперименты

Опишем теперь подробно как проводились эксперименты: как устроены данные, как настраивались и задавались параметры и к какие основные результаты были получены.

5.1 Данные

Данные представляют несколько миллионов пользовательских запросов в систему автоматического перевода. Записи запросов идут в том же порядке, в каком они поступали в систему. Также присутствуют пометки номеров пользовательских сессий, из которых они пришли.

Распределение текстовых запросов по длинам представлено на Рис. 4. Из данного распределения видно, что около половины запросов имеют длину один. Это объясняется тем, что пользователи часто используют система автоматического перевода в качестве словаря. Двухсловные запросы также часто могут являться устойчивыми выражениями и словосочетаниями, встречающимися в словарях.

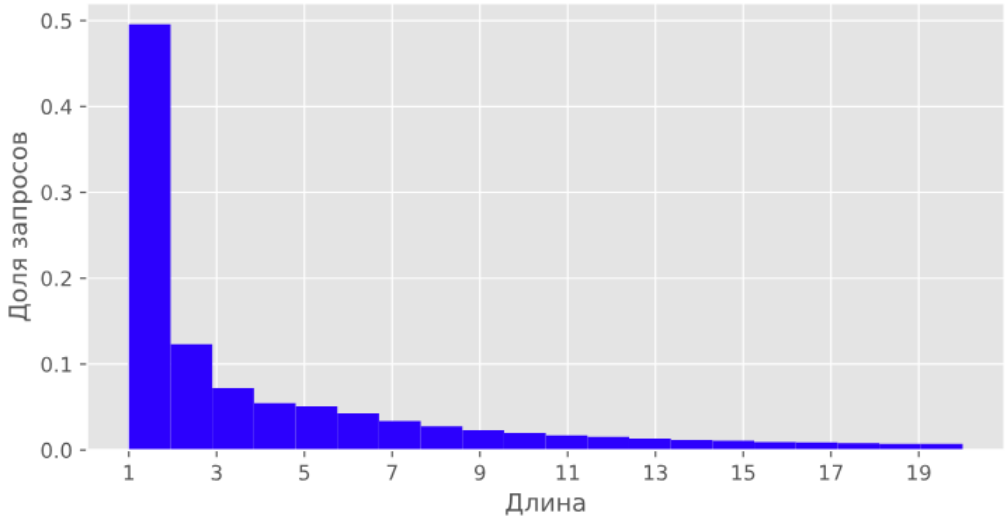


Рис. 4: Распределение длин текстов пользовательских запросов в логарифмической шкале. По оси абсцисс представлена длина запроса в словах, по оси ординат — доля объектов такой длины в выборке.

Слишком короткие запросы не несут в себе стилистических или тематических особенностей, могут быть совершенно не связаны друг с другом внутри сессий, поэтому для обучения всех моделей, запросы длины меньше трёх использоваться не будут.

Очищенные от коротких запросов обучающие данные разделены на обучение и валидацию. Размер валидации составляет 50 тысяч запросов, на ней будут считаться функции потерь для отсечения переобучения, а также на ней будет считаться основная метрика качества (1).

5.2 Настройка тематической модели

Для построения тематической модели, как уже было сказано, текстовые запросы внутри сессий объединяются в один «документ», для того чтобы тематическая модель могла выявить семантическую связь между запросами.

Для построения тематической модели использовалась библиотека BigARTM, дающая

возможность учить модели с аддитивными дополнительными функциями потерь предложенными в [7].

Сначала подберём основной гиперпараметр тематической модели — число тем. Выбор числа тем описан в таблице 1.

Количество тем	Значение качества
50	125.3
100	115.3
150	120.1

Таблица 1: Подбор числа тем в тематической модели. Для оценки качества используется основная метрика качества (1).

Для построения модели использовался регуляризатор сглаживания фоновой темы, который добавляет некоторую константу к параметрам φ_{wt_0} выделенной фоновой темы t_0 для того, чтобы слова, не относящиеся ни к одной другой теме, становились фоновыми. Настройка гиперпараметра сглаживания не сильно влияла на качество, однако с ним качество было лучше, чем без него. Точное значение можно увидеть в таблице 2.

Также использовался регуляризатор декоррелирования тем, который увеличивает различность тем в тематической модели. Настройка гиперпараметра этого регуляризатора проводилась перебором по логарифмической сетке. Лучшее значение описано в таблице 2.

Параметры ТМ	Значение качества
ТМ без регуляризаторов	115.3
ТМ + декоррелирование	112.6
ТМ + сглаж. фона + декор.	105.7

Таблица 2: Настройка гиперпараметров тематической модели.

Все описанные модели обучались до сходимости функции потерь.

В случае наличия фоновой темы, во время вычисления близости векторов θ фоновая тема выкидывалась.

Средняя доля фона в векторных представлений составила 15%.

5.3 Обучение нейросетевых моделей

Нейросетевые архитектуры обладают огромным количеством гиперпараметров, среди которых есть число слоёв, количество нейронов на каждом слое, порядок слоёв и так далее.

Честная настройка всех гиперпараметров невозможна, в силу их количества а также длительности обучения каждой конфигурации нейросети.

Для модели Transformer в исходной статье [12] уже предложены два набора параметров, с которыми, по словам авторов, модель работает лучше всего.

Первая конфигурация называется Transformer-base и подразумевает 6 слоёв, размерность векторных представлений в сети равную 512, 8 голов внимания и процедурой дропаута с коэффициентом дропаута, равным 0.15.

Во второй конфигурации Transformer-big размерность равна 1024 и 16 голов внимания. Остальные параметры те же.

В наших экспериментах мы будем учить исключительно base версию архитектуры. Связано это с тем, что существенным является время применения модели на больших объемах данных, а у big версии время применения значительно больше при тех же ресурсах.

Кроме параметров архитектуры важны также параметры обучения: размер подвыборки в методе оптимизации, скорость градиентного спуска, метод оптимизации. Все эти параметры мы взяли из оригинальной статьи: используем метод оптимизации Adam [14], скорость градиентного спуска спуска адаптивная, увидеть подробнее значений скорости обучения для каждой итерации можно на графике 5.

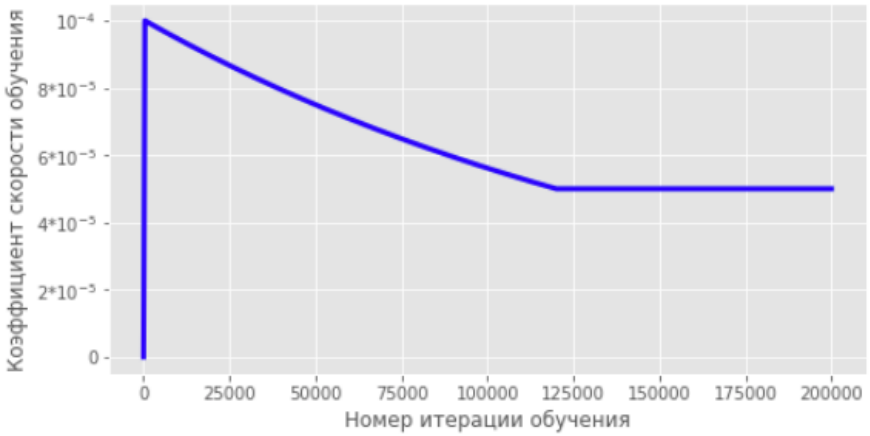


Рис. 5: Значение коэффициента скорости обучения для разных итераций.

Размер батча выбирался максимально возможным, исходя из ограничений памяти графических процессоров, равным 300.

Теперь рассмотрим способ получения векторного представления запроса из векторных представлений слов на выходе модели Transformer.

В таблице 3 представлены результаты функции потерь на валидационной выборке, при обучении сетей с конфигурациями, отличающимися только способом агрегации векторных

представлений слов в представление запроса. Как видно с процедурой агрегации с вниманием сеть обучилась несколько лучше, что связано с тем, что данные способ является более гибким. Оба результаты представлены после сходимости сетей. Обучались сети с функцией потерь аппроксимации глобального софтмакса на подвыборках (8).

Способ агрегации	Функция потерь на валидации
среднее $\times \sqrt{\text{длина}}$	3.34
Агрегация с вниманием	3.29

Таблица 3: Сравнение способов агрегации векторов слов в вектор запроса.

Теперь рассмотрим конфигурацию глубинных усредняющих сетей. Размер векторного представления выберем равным 512, как в Transformer. Количество полносвязных слоёв выберем равным 5. Подбирать эти гиперпараметры мы не будем, так как цель экспериментов с этими сетями — понять насколько данная задача хорошо решается простыми архитектурами.

Данные архитектуры обучаются с той же функцией потерь, что и Transformer.

6 Анализ результатов

6.1 Сравнение подходов

Рассмотрим теперь результаты сравнения предложенных подходов к решению задачи классификации принадлежности пар запросов одной пользовательской сессии. Результаты сравнения можно увидеть в таблице 4. Метрика ранговой близости (1) считалась по случайным подвыборкам размера $K = 300$. Таким образом, у модели, выдающей вектор случайных чисел в качестве ответа, значение метрики качества равнялось бы 150.

Рассмотрим подробнее модели, описанные в таблице. Модель USE —это описанная ранее предобученная модель, считающаяся лучшей на данный момент в задаче определения семантической близости текстов. Архитектура данной сети представляет из себя Transformer Big, то есть в данной модели на порядок больше параметров, чем в обученной нами. Как видно, данная модель имеет высокое качество, которое только одной модели удалось превзойти.

Функции потерь Триплеты, БПЭ и АГСНП — это триплет функция потерь, бинарная перекрёстная энтропия и аппроксимация глобального софтмакса на подвыборках, описанные (6), (4) и (8) соответственно.

Модель	Метрика ранговой близости
Случайный вектор	150
USE (baseline)	**28.0**
Word2Vec	125.6
Тематическая модель	105.7
ГУС + АГСНП	58.6
Transformer + БПЭ	141.4
Transformer + Триплеты	33.4
Transformer + АГСНП	**19.5**

Таблица 4: Сравнение различных подходов к решению задачи классификации принадлежности пар запросов одной пользовательской сессии.

Как можно увидеть из экспериментов, отличающихся только функцией потерь (последние три строки), с бинарной перекрёстной энтропией модель практически не отличается от случайных предсказаний, а лучше всего себя показывает описанная нами аппроксимация глобального софтмакса (8). Объяснить это можно эффективностью использования негативных примеров. В относительно небольшом батче, соотношение положительных и отрицательных примеров БПЭ один к одному, что сильно усложняет сходимость сети. Также стоит отметить, что АГСНП — это единственная функция потерь, с которой удалось достичь качества, выше предобученной USE.

Модель ГУС — это глубинная усредняющая сеть. Как видно, сложность сети имеет большое значение для качества решения задачи. Несмотря на то что сеть училась с той же функцией потерь АГСНП, качество получилось заметно хуже.

Модель Word2Vec [5] — предобученная, не видела данных из рассматриваемого домена. Векторное представление запроса получалось путём усреднения представлений слов. Как видно, качество решения задачи данной моделью довольно низкое. Это объясняется тем, что модель простая и одновременно не училась на данных из нашей обучающей выборки.

Тематическая модель училась на данных из рассматриваемого домена, однако функция потерь, которую она оптимизирует, не связана с задачей контекстной близости пар запросов. Поэтому, как видно, качество решения хуже чем у моделей, обучавшихся на специальные функции потерь, но лучше чем у предобученной модели Word2Vec.

6.2 Кластеризация пользовательских запросов

Рассмотрим теперь кластеризацию векторных представлений, полученных нашей лучшей моделью. Для построения кластеризации будем использовать агломеративную кластеризацию снизу-вверх. Сначала запускается алгоритм k-средних, для получения мелкой кластеризации, а ее результаты объединяются иерархически.

Такой алгоритм кластеризации выбран для того, чтобы при анализе пользовательской аудитории можно было проводить исследования на разных уровнях детальности.

Выберем сначала срез в 30 кластеров. В таблице 5 представлены топ tf-idf слов для каждого из кластеров.

Гаджеты	**Экология**	**Медицина**	**Школьная тема**
shipping	water	body	never
color	animals	disease	school
package	environmental	patients	home
phone	planet	therapy	going
items	natural	cells	friend
light	air	blood	friends
case	species	clinical	english
leather	pollution	treatment	play
product	plants	study	got
pictures	climate	health	live
usb	earth	drug	let
condition	soil	cell	read
screen	world	risk	write
iphone	energy	effects	say
buy	nature	during	house

Таблица 5: Топ-слова по tf-idf для кластеров, построенных на основе полученных векторных представлений. Слева представлены кластера-темы, справа, кластера более сложной природы.

Среди полученных кластеров можно выделить кластера-темы, представленные в таблице слева, которые объединены общей темой и характерной лексикой, такие кластера находят все модели.

Справа же представлен кластер более сложной структуры — запросы школьников, кото-

рый не находят никакие модели, кроме той, что училась на контекстную близость. Запросы этого кластера не имеют общей темы и посвящены истории, географии, политике, грамматическим заданиям и так далее. Примеры запросов из этого кластера можно увидеть в таблице 6.

Oscar had an elder brother and a younger sister
Having breakfast in the eighth quarter.
What kind of animals are elks, bears, foxes and hares?
The nevalyashka has a big and two small handles.

Таблица 6: Примеры запросов из школьного кластера.

На самом мелком уровне кластеризации можно увидеть, из чего состоит данный кластер. В нем можно выделить следующие подкластера:

- тексты домашнего чтения;
- задания для переводов на русский язык;
- тексты для отработки грамматики;
- задания с выбором пунктов и пропусками;
- запросы на переводы формулировок заданий.

Как видно, анализируемые данные имеют довольно сложную структуру, которую не всем моделям удаётся уловить.

6.3 Построение классификатора научных запросов

Попробуем применить теперь полученные векторные представления для обучения классификаторов запросов из данного домена. Цель данных классификаторов может заключаться в выявлении каких-либо пользовательских подгрупп. В качестве такой подгруппы рассмотрим научную аудиторию, которая присылает в перевод запросы научного содержания.

Для обучения классификатора возьмём подвыборку размером в 2000 объектов, сбалансированную по кластерам и размеченную людьми на факт принадлежности научной тематике. Балансировка производилась из-за того, что изначальная доля научных запросов слишком мала.

После этого строим линейный классификатор первого уровня, с помощью которого отбираем более сложные объекты для разметки.

На этих данных сравним работу различных классификаторов:

- логистическая регрессия над tf-idf векторами запросов для того, чтобы понять насколько хорошо задача решается простым классификатором;
- логистическая регрессия над векторами предобученной модели Transformer + АГСНП;
- логистическая регрессия над векторами предобученной модели Transformer + АГСНП с доучиванием модели;
- логистическая регрессия над векторами обучаемой с нуля модели,

где АГСНП — это предложенная нами функуция потерь (8).

Результаты сравнения приведены в таблице 7. Как можно увидеть, модель с tf-idf имеет довольно неплохое качество. Это объясняется тем, что научным текстам свойственна специфичная лексика, которую как раз улавливают tf-idf вектора.

Можно заметить, что использование в качестве признаков векторов из предобученной на контекстную близость модели значительно улучшает результат. Также можно заметить, что доучиванием модели не получается сильно улучшить этот результат, что говорит о том, что в изначальных векторах уже есть достаточно необходимой информации.

Также из таблицы видно, что та же архитектура, обученная с нуля, имеет худшее качество. Это показывает обогащенность наших векторов информацией, которой нет изначально в размеченной выборке, но есть в данных, на которых моделях предобучалась.

Модель	ROC AUC на валидации
Лог. регр. tf-idf	0.80
Лог. регр. предобуч.	0.89
Лог. регр. предобуч. + доуч	0.92
Обучение Transformer с нуля	0.83

Таблица 7: Результаты обучения классификатора научных запросов на предобученных векторных представлениях.

7 Заключение

Таким образом, в нашей работе удалось формализовать общую постановку задачи построения тематико-стилистических векторных представления, свести её к построению контекстно зависимых векторных представлений.

Был сформулирован критерий качества решения данной задачи, рассмотрены принятые в мировом сообществе решения и лучшие на данный момент модели, решающие данную задачу.

В общих подходах был выявлен минус, связанный с избыточностью вычислений при подсчёте функции потерь во время обучения тяжелых нейросетевых архитектур и показана низкая с практической точки зрения эффективность стандартных методов.

Для решения данной проблемы в работе предложен новый функционал качества, оптимизация которого решает изначально ту же задачу, но приводит к сравнительно лучшему решению.

Далее полученные векторные представления успешно применены для кластерного анализа данных. При этом выявлены особенности структуры обучающих данных, которые не видят другие модели.

Также показана эффективность использования обученных векторных представлений в задаче построения доменных классификаторов. Показано, что полученного качества не удаётся достичь простыми моделями, а также не удаёся значительно увеличить доучиванием модели, что говорит о высоком качестве обученных векторных представлений.

Предложенный метод рекомендуется к применению в задаче построения семантических контекстных векторных представлений текстов, при условиях наличия достаточного количества данных для обучения, а также наличия контекстной структуры в данных.

Список литературы

[1] *Thomas Hoffmann* Unsupervised Learning by Probabilistic Latent Semantic Analysis. Machine Learning, Vol. 42, Issue 1-2, pp. 177-196, 2001.

[2] *David M. Blei, Andrew Y. Ng, and Michael I. Jordan.* 2003. Latent dirichlet allocation. J. Mach. Learn. Res. 3 (March 2003), 993-1022.

[3] *Zhao W.X. et al.* (2011) Comparing Twitter and Traditional Media Using Topic Models. In: Clough P. et al. (eds) Advances in Information Retrieval. ECIR 2011. Lecture Notes in Computer Science, vol 6611. Springer, Berlin, Heidelberg

[4] *Mikolov, T., et al.* (2010). Recurrent neural network based language model. In T. Kobayashi, K. Hirose S. Nakamura (eds.), INTERSPEECH (p./pp. 1045-1048), : ISCA.

[5] *Mikolov, Tomas Corrado, G.s Chen, Kai Dean, Jeffrey.* (2013). Efficient Estimation of Word Representations in Vector Space. 1-12. Proceedings of the International Conference on Learning Representations

[6] *Ryan Kiros, Yukun Zhu, Ruslan Salakhutdinov, Richard S. Zemel, Antonio Torralba, Raquel Urtasun, and Sanja Fidler.* 2015. Skip-thought vectors. In Proceedings of the 28th International Conference on Neural Information Processing Systems - Volume 2 (NIPS'15), C. Cortes, D. D. Lee, M. Sugiyama, and R. Garnett (Eds.), Vol. 2. MIT Press, Cambridge, MA, USA, 3294-3302.

[7] *Vorontsov K., Potapenko A.* Additive regularization of topic models // Machine Learning. — 2015. — Vol. 101, no. 1-3. — Pp. 303–323

[8] *Sepp Hochreiter and Jürgen Schmidhuber.* 1997. Long Short-Term Memory. Neural Comput. 9, 8 (November 1997), 1735-1780.

[9] *Jacob Devlin et al.* (2018) BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding, CoRR 1810.04805

[10] *Perone, Christian S. et al.* "Evaluation of sentence embeddings in downstream and linguistic probing tasks." CoRR abs/1806.06259 (2018): n. pag.

[11] *Bahdanau, Dzmitry et al.* "Neural Machine Translation by Jointly Learning to Align and Translate." CoRR abs/1409.0473 (2015): n. pag.

[12] *Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, L., Polosukhin, I.* (2017). Attention Is All You Need. NIPS.

[13] *Robbins, Herbert; Monro, Sutton.* A Stochastic Approximation Method. Ann. Math. Statist. 22 (1951), no. 3, 400–407. doi:10.1214/aoms/1177729586. https://projecteuclid.org/euclid.aoms/1177729586

[14] *Kingma, Diederik P. and Jimmy Ba.* "Adam: A Method for Stochastic Optimization." CoRR abs/1412.6980 (2015): n. pag.

[15] *Cer, D., Yang, Y., Kong, S., Hua, N., Limtiaco, N., John, R.S., Constant, N., Guajardo-Cespedes, M., Yuan, S., Tar, C., Sung, Y., Strope, B., Kurzweil, R.* (2018). Universal Sentence Encoder. CoRR, abs/1803.11175.

[16] *Iyyer, Mohit Manjunatha, Varun Boyd-Graber, Jordan Daumé III, Hal.* (2015). Deep Unordered Composition Rivals Syntactic Methods for Text Classification. 1681-1691. 10.3115/v1/P15-1162.

[17] *Wu, L.Y., Fisch, A., Chopra, S., Adams, K., Bordes, A., Weston, J.* (2018). StarSpace: Embed All The Things! AAAI.