Содержание

1 Введение

Целью данной практики является закрепление и развитие профессиональных компетенций научно-исследовательской и проектной деятельности. Для достижения поставленной цели потребовалось решить следующие задачи практики (в соответствии с программой практики):

- Закрепление и расширение теоретических и практических знаний, полученных студентом в процессе обучения;
- Получение навыков самостоятельной работы, а также работы в составе научно-исследовательских коллективов.
- Работа над исследованием по анализу аффинных преобразований в семантическом пространстве BERT.
- Обработка полученных материалов и оформление отчета о прохождении практики.

Исследование проводится на языке python в среде Jupyter Notebok при использовании Google Colab. Jupyter Notebok является наиболее удобной платформой для проведения исследований на python. Google Colab является бесплатной и мощной платформой для запуска кода. При этом дается 12Гб оперативной памяти, доступ к Google диску для доступа к данным, а также есть возможность запускать код с использованием GPU.

2 Содержательная часть

2.1 Описание профессиональных задач студента

Для проведения исследования по анализу аффинных преобразований в семантическом пространстве BERT необходимо решить следующие задачи:

- Исследование моделей векторного представления слов;
- Исследование методов оценки аффинных преобразований;
- Разработка метода оценки точности параллельного переноса для контекстуализированных моделей;
- Подготовка экспериментальных данных;
- Проведение экспериментов;
- Оценка полученных результатов.

2.2 Исследование моделей векторного представления слов

Векторное представление слов — метод обработки естественного языка, в основе которого лежит идея представить каждое слово или токен в виде вектора определенной размерности.

2.2.1 One-hot encoding

Самой простой реализацией модели векторного представления слов является one-hot encoding. Идея этой модели заключается в том, что в наборе из K слов каждому слову соответствует вектор длиной K со всеми нулями и единицей с позицией i, где i - это номер слова во всем наборе. Недостатком этого метода является то, что по данным векторным представлениям нельзя судить о схожести слов. Также для больших наборов слов размер векторных представлений будет очень большим, из-за чего их неэффективно хранить в памяти.

2.2.2 Word2vec

Word2vec [1] одна из первых моделей, использующих нейронные сети для создания векторных представлений слов. Идея создания векторов в word2vec основана на предположении о контекстной близости, а именно на том, что слова встречающиеся в одинаковых контекстах скорее всего имеют схожее значение. Предлагается проверять схожесть слов при помощи косинусного сходства их векторных представлений (1).

$$similarity(A,B) = cos(A,B) = cos(\theta) = \frac{A \cdot B}{\|A\|\|B\|} \tag{1}$$

где

A - векторное представление первого слова,

B - векторное представление второго слова,

θ - угол между векторами A и B.

Существует 2 метода обучения word2vec: kip-gram и CBOW (Continuous Bag of Words). В Skip-gram по слову прогнозируется слова из его контекста, в CBOW, наоборот, по контексту предсказывается слово. В качестве выходного слоя в моделях применяется функция softmax в различных вариациях для того.

2.2.3 FastText

FastText является продолжением развития модели word2vec. При этом в fastText отличается от word2vec тем, что у новой модели используются N-граммы символов. Например, для слова молоко 3-граммами являются мол, оло, лок, око. Векторные представления строятся именно для N-грамм, векторные представления слов - это сумма векторных представлений всех его N-грамм. При этом решается проблема того, что словарь модели word2vec был ограничен. Использование N-грамм позволяет получать векторные представления для редких слов.

2.2.4 ELMO

ELMO [3] была одной из первых моделей обработки естественного языка, которая учитывала контекст слова. Для моделей word2vec или fastText при вычислении векторного представления не учитывается контекст слова, для омонимов и омографов представления будут одинковыми. В ELMO решается эта проблема, в основе

этой модели лежит многослойная двунаправленная рекуррентная нейронная сеть с LSTM (Long short-term memory).

2.2.5 BERT

Модель BERT [4] или Bidirectional Encoder Representations from Transformers была опубликована командой Google AI в 2018 году. На момент появления модель BERT показала лучшее качество на тесте SQuAD 1.1 [5].

BERT состоит из 12 следующих друг за другом энкодеров. Каждый энкодер состоит из компонентов, первый компонент - это слой внутреннего внимания (self-attention), второй - нейронная сеть прямого распространения (feed-forward neural network).

Обучение модели BERT основывается на следующих принципах. Первый основывается на том, чтобы заменить 15% слов масками и обучить модель предсказывать эти слова. Этот принцип позволяет модели самой обучаться на полноценных текстах без предварительной разметки, что позволило обучить BERT на огромном массиве данных. Вторая идея состоит в том, чтобы дополнительно научить BERT определять, является ли одно предложение логичным продолжением другого.

2.3 Исследование методов оценки аффинных преобразований

В основе большинства методов оценки аффинных преобразований лежит описанное в формуле (1) косинусное сходство. Для удобства применяется также косинусное расстояние (2).

$$distance(A,B) = 1 - similarity(A,B) \tag{2}$$

Аффинные преобразования над векторными представлениями слов предполагают задачу пропорциональной аналогии. Задача пропорциональной аналогии заключается в том, чтобы установить для группы пар слов определенную логическую связь в парах. Рассмотрим группу пар слов $(a : a^*), (b : b^*)$. Известно, что слова в первой группе относятся друг к другу также, как и слова во второй группе. Представить это можно следующим образом (3).

$$(a : a^*) :: (b : b^*) \tag{3}$$

Если в парах действует одна и та же связь, то можно выразить одно из слов через 3 других слова (4).

$$y = v_b - v_a + v_{a^*} \tag{4}$$

где

y - искомый вектор,

v_x - эмбеддинг для слова x.

Но вектору y может не соответствовать ни одного слова, поэтому в качестве ответа берется ближайшее слово к y. Для корректно работающей модели ответ совпадет с b^*. Результат можно выразить формулой (5).

$$y^* = \underset{v \notin \{v_b, v_a, v_{a^*}\}}{argmax} \; cos(v, v_b - v_a + v_{a^*}) \tag{5}$$

Интуитивно понятным способом оценить качество будет сопоставить все пары слов друг с другом, посчитать одно из слов по формуле (5) [6] и найти среднее значение косинусной близости после трансформаций. Данный способ получил название **3CosAdd**.

Существует пердположение, что 6 эквивалентно (5). Этот метод получил название **3CosMul** [6].

$$y^* = \underset{v \notin \{v_b, v_a, v_{a^*}\}}{argmax} \; cos(v, v_{a^*}) - cos(v, v_a) + cos(v, v_b) \tag{6}$$

Если в группе у пар одна и та же связь, то можно посчитать среднюю разницу между векторами в парах и задача аналогии будет решаться через посчитанную разницу. Эта идея предлагается в метрике **3CosAvg** (7) [7].

$$y^* = \underset{v \notin \{v_b, v_a, v_{a^*}\}}{argmax} \; cos(v, v_b + \frac{\sum_{i=1}^{n} v_{a_i^*}}{n} - \frac{\sum_{i=1}^{n} v_{a_i}}{n}) \tag{7}$$

Существуют и другие методы оценки аффинных преобразований. Метод **Only-b** [8] основан на предположении, что вектора v_b и v_{b^*} находятся максимально близко друг к другу (8).

$$y^* = \underset{v \notin \{v_b, v_a, v_{a^*}\}}{argmax} \; cos(v, v_b) \tag{8}$$

В методе **Ignore-a** [8] предполагается, что результат ближе всего к сумме векто-

ров v_b и v_{a^*} (9).

$$y^* = \underset{v \notin \{v_b, v_a, v_{a^*}\}}{argmax} \; cos(v, v_b + v_{a^*}) \tag{9}$$

В методе **Add-opposite** [8] результатом является вектор, ближайший к выражению $-(v_{a^*} - v_a) + v_b$ (10).

$$y^* = \underset{v \notin \{v_b, v_a, v_{a^*}\}}{argmax} \; cos(v, -(v_{a^*} - v_a) + v_b) \tag{10}$$

Метод **MULTIPLY** [8] имеет следующий вид (11).

$$y^* = \underset{v \notin \{v_b, v_a, v_{a^*}\}}{argmax} \; \frac{cos(v, v_{a^*}) \; cos(v, v_b)}{cos(v, v_a)} \tag{11}$$

2.4 Разработка метода оценки точности параллельного переноса для контекстуализированных моделей

Для контекстуализированных моделей описанные ранее методы в чистом виде не применимы. Для одних и тех же слов в разных контекстах будут получаться разные эмбеддинги. Получить единственный эмбеддинг для токена можно после усреднения всех полученных эмбеддингов для этого токена. В результате каждому токену будет соответствовать только одно векторное представление.

После получения усредненных эмбеддингов для всех токенов можно применять описанные ранее методы оценки аффинных преобразований.

2.5 Подготовка экспериментальных данных

Для получения эмбеддингов слов были взяты тексты из электронной библиотеки КиберЛенинка. Тексты двух жанров: литература и политика.

Для оценки качества аффинных преобразований используется датасет $Google\_analog$ Данный датасет был переведен на русский язык с сохранением семантических отношений между словами. Не все слова из данного датасета есть в словаре BERT, поэтому часть отношений пришлось убрать.

2.6 Проведение экспериментов

Для проведения экспериментов используется язык программирования python в среде Jupyter Notebok с использованием Google Colab. Jupyter Notebok является наиболее удобной платформой для проведения исследований на python. Google Colab является бесплатной и мощной платформой для запуска кода. При этом дается 12Гб оперативной памяти, доступ к Google диску для доступа к данным, а также есть возможность запускать код с использованием GPU. В качестве фреймворка для работы с моделью BERT был выбран pytorch, так как это современная и гибкая библиотека для работы с глубинным обучением.

Для проведения экспериментов необходимо подготовить данные для их обработки в модели BERT. Сначала весь текст разбивается на отдельные предложения, далее происходит их токенизация и индексация. На этом этапе обработанные предложения по-одному отправляются в модель BERT. Данным способом обработаны по 1 миллиону предложений для каждого жанра.

Полученные после обработки объекты представляют из себя четырёхразмерные тензоры, где оси отражают следующую информацию (в скобках представлено количество элементов):

1. Номер слоя (13 слоев);
2. Номер батча (1 предложение);
3. Количество слов/токенов в предложении (количество токенов в предложении);
4. Векторное представление (768 свойств).

По оси слоев первый слой - это эмбеддинг, поступающий на вход модели, остальные 12 слоев отображают выходы 12 энкодеров. Номер батча в нашем случае не важен, так как используется только одно предложение. Следующая ось отображает токены в предложении с сохранением порядка. Последняя ось отвечает за векторное представление каждого токена.

Получить итоговое векторное представление для токена можно несколькими способами (рисунок 1). В нашем случае используется способ с суммированием последних четырех слоев, данный способ показывает хорошее качество. Способ с конкатенацией последних четырех не используется, так как он требует в 4 раза больше ресурсов.

Описанным ранее методом обрабатываются все подготовленные предложения. Обработка происходит пачками по 10 тысяч предложений. Векторные представления токенов каждой пачки сохраняются на Google диск. Сделано это из-за ограничений оперативной памяти устройства.

После того как получены векторные представления для всего текста, считаются средние эмбеддинги для всех токенов. Из-за ограничений оперативной памяти нельзя посчитать сразу все векторные представления, поэтому они считаются порциями с сохранением промежуточных результатов.

Далее проверялось семантические отношения полученных эмбеддингов на переведенном датасете $Google\_analogy\_test\_set$. В качестве метрик были взяты 3CosAdd, 3CosAvg и 3CosMul.

2.7 Оценка полученных результатов

На рисунках 2 и 3 представлены результаты аффинных преобразований для литературы и политики соответственно.

Первая колонка type описывает вид связи в группе. Строка metric указывает использованную метрику. Колонка result отражает долю верных ответов. Колонка cosine отображает среднее косинусное расстояние между посчитанным эмбеддингом и правильным эмбеддингом. Последняя колонка описывает количество тестов в группе.

Данные результаты показывают, что аффинные преобразования в модели BERT возможны, но их качество плохое. Это может быть связано с тем, что в эмбеддингах помимо семантического смысла может скрываться какая-то другая информация о словах. Еще одной из возможных причин плохого качества может быть то, что исследование проводилось на русском языке и не все слова из тестового датасета были в словаре модели BERT.

3 Заключение

В ходе практики, были изучены модели векторного представления слов, рассмотрены самые популярные архитектуры моделей преобразования слов в эмбеддиги.

Были приобретены навыки по поиску необходимого датасета, по работе и применению модели BERT.

По окончанию практики была достигнута главная цель - применение теоретических знаний, полученных в процессе обучения, в реальной экспериментальной задаче.

А также приобретены навыки и опыт практической работы с NLP. Данная практика является хорошим практическим опытом для дальнейшей самостоятельной деятельности.

Список использованных источников

1. Mikolov T. et al. Efficient estimation of word representations in vector space //arXiv preprint arXiv:1301.3781. – 2013.
2. Bojanowski P. et al. Enriching word vectors with subword information //Transactions of the Association for Computational Linguistics. – 2017. – Т. 5. – С. 135-146.
3. Peters M. E. et al. Deep contextualized word representations //arXiv preprint arXiv:1802.05365. – 2018.
4. Devlin J. et al. Bert: Pre-training of deep bidirectional transformers for language understanding //arXiv preprint
5. Rajpurkar P. et al. Squad: 100,000+ questions for machine comprehension of text //arXiv preprint arXiv:1606.05250. – 2016.
6. Levy O., Goldberg Y. Linguistic Regularities in Sparse and Explicit Word Representations // Proceedings of the Eighteenth Conference on Computational Natural Language Learning, 2014. P.171-180
7. Drozd A., Gladkova A., Matsuoka S. Word Embeddings, Analogies, and Machine Learning: Beyond King-Man+Woman=Queen // Conference: Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers, 2016. P.3519–3530.
8. Linzen T. Issues in evaluating semantic spaces using word analogies // Proceedings of the 1st Workshop on Evaluating Vector-Space Representations for NLP, 2016. P. 13–18.

4 Приложения

С кодом можно ознакомиться по ссылке:
https://github.com/andrsolo21/hse_Af_Tr_BERT.

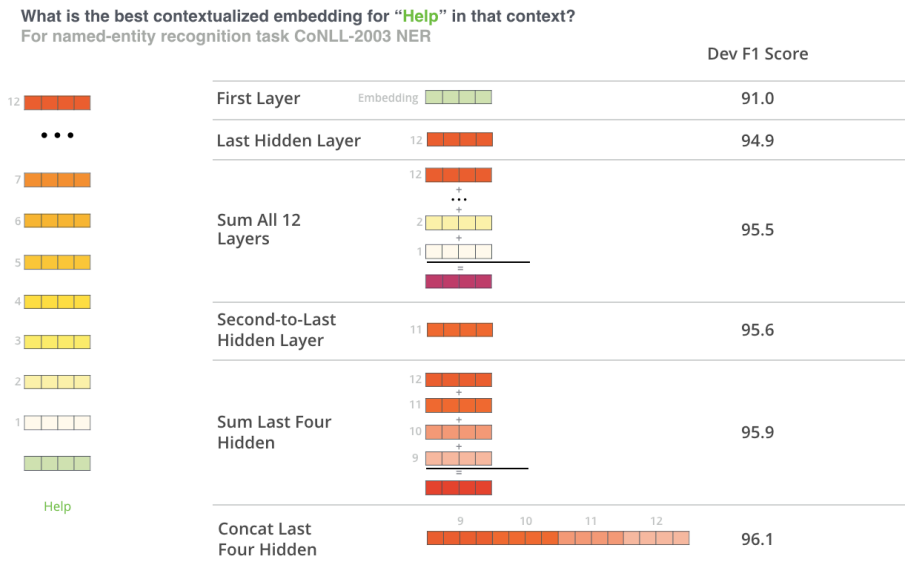

Рисунок 1 – Возможные варианты получения векторного представления

	result		cosine			count
metric	3CosAdd	3CosMul	3CosAdd	3CosMul	3CosAvg	
type						
Adjective adverb	0.222222	0.222222	0.857969	0.930546	1.000000	306
Capital-countries	0.666667	0.666667	0.832437	0.832437	0.921754	3
Comparative	0.206522	0.215580	0.845331	0.907858	1.000000	552
Family	0.659091	0.643939	0.917592	0.937557	1.000000	132
Nationality adjective	0.333333	0.333333	0.899240	0.968291	1.000000	6
Opposite	0.000000	0.000000	0.793130	0.918517	1.000000	20

Рисунок 2 – Результаты тестирования для текста с литературой

	result		cosine			count
metric	3CosAdd	3CosMul	3CosAdd	3CosMul	3CosAvg	
type						
Adjective adverb	0.271930	0.269006	0.856142	0.929608	1.000000	342
Capital-countries	0.000000	0.000000	0.892919	0.892919	0.928382	3
Comparative	0.266082	0.245614	0.841756	0.896573	1.000000	342
Family	0.309524	0.309524	0.888865	0.923934	1.000000	42
Nationality adjective	0.333333	0.333333	0.906793	0.975569	1.000000	12
Opposite	0.000000	0.000000	0.792315	0.894300	1.000000	2

Рисунок 3 – Результаты тестирования для текста с политикой