ПРАВИТЕЛЬСТВО РОССИЙСКОЙ ФЕДЕРАЦИИ
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ
«ВЫСШАЯ ШКОЛА ЭКОНОМИКИ»

**Московский институт электроники и математики**
**Им. А.Н.Тихонова НИУ ВШЭ**

**Департамент компьютерной инженерии**

Москва 2021 г.

# Содержание

**1 Введение** . . . . . **3**

**2 Содержательная часть** . . . . . **4**

2.1 Описание профессиональных задач студента . . . . . 4

2.2 Исследование моделей векторного представления слов . . . . . 4

2.2.1 One-hot encoding . . . . . 4

2.2.2 Word2vec . . . . . 4

2.2.3 FastText . . . . . 5

2.2.4 ELMO . . . . . 5

2.2.5 BERT . . . . . 5

2.3 Исследование методов оценки аффинных преобразований . . . . . 6

2.4 Постановка задачи пропорциональной аналогии в терминах аффинного преобразования . . . . . 6

2.5 Разработка метода оценки точности параллельного переноса для контекстуализированных моделей . . . . . 6

2.6 Подготовка экспериментальных данных . . . . . 6

2.7 Проведение экспериментов . . . . . 7

2.8 Оценка полученных результатов . . . . . 8

**3 Заключение** . . . . . **9**

**4 Приложения** . . . . . **10**

# 1 Введение

Целью данной работы является исследование и измерение качества аффинных преобразований для модели BERT.

Для достижения поставленной цели необходимо решить следующие задачи:

- Исследование моделей векторного представления слов;
- Исследование методов оценки аффинных преобразований;
- Постановка задачи пропорциональной аналогии в терминах аффинного преобразования;
- Разработка метода оценки точности параллельного переноса для контекстуализированных моделей;
- Подготовка экспериментальных данных;
- Проведение экспериментов;
- Оценка полученных результатов.

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

Исследование моделей векторного представления слов;

Исследование методов оценки афинных преобразований;

Постановка задачи пропорциональной аналогии в терминах аффинного преобразования;

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 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).

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

где

$A$ - векторное представление первого слова

$B$ - векторное представление второго слова

$\theta$ - 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 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 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.

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

2

## 2.4 Постановка задачи пропорциональной аналогии в терминах аффинного преобразования

3

## 2.5 Разработка метода оценки точности параллельного переноса для контекстуализированных моделей

4

## 2.6 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.

## 2.7 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 $Google\_analogy\_test\_set$.

## 2.8 Оценка полученных результатов

На рисунках 2 и 3 представлены результаты

# 3 Заключение

# 4 Приложения

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

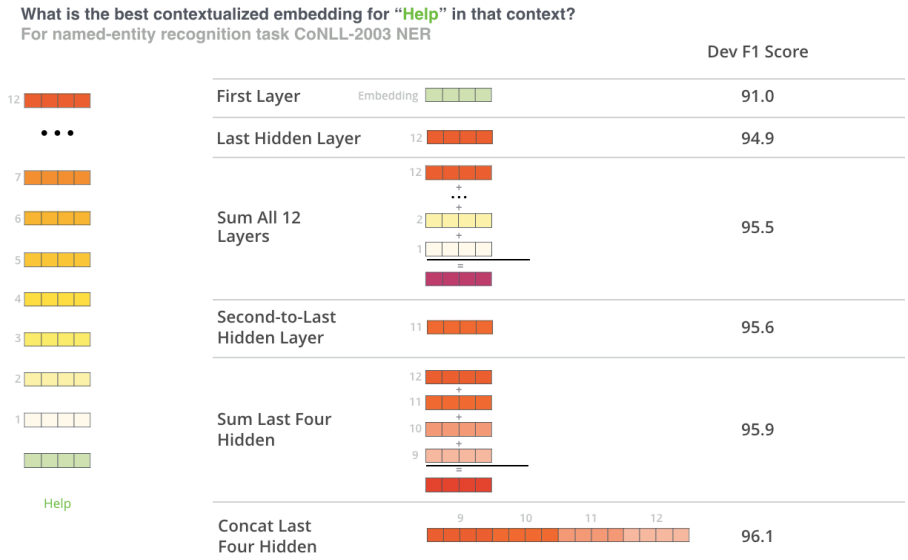


Рис. 1: Возможные варианты получения векторного представления

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

Рис. 2: Результаты тестирования для текста с литературой

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

Рис. 3: Результаты тестирования для текста с политикой