Физтех-Школа Прикладной математики и информатики (ФПМИ) МФТИ

Some parts of the notebook are almost the copy of mmta-team course. Special thanks to mmta-team for making them publicly available. Original notebook.

Прочитайте семинар, пожалуйста, для успешного выполнения домашнего задания. В конце ноутка напишите свой вывод. Работа без вывода оценивается ниже.

Задача поиска схожих по смыслу предложений

Мы будем ранжировать вопросы StackOverflow на основе семантического векторного представления

До этого в курсе не было речи про задачу ранжировния, поэтому введем математическую формулировку

Задача ранжирования(Learning to Rank)

- X - множество объектов
- $X^l = \{x_1, x_2, ..., x_l\}$ - обучающая выборка
  На обучающей выборке задан порядок между некоторыми элементами, то

есть нам известно, что некий объект выборки более релевантный для нас, чем другой:

- $i \prec j$ - порядок пары индексов объектов на выборке X^l с индексами i и j

Задача:

построить ранжирующую функцию a : $X \rightarrow R$ такую, что $$i \prec j \Rightarrow a(x_i) < a(x_j)$$

Embeddings

Будем использовать предобученные векторные представления слов на постах Stack Overflow.
A word2vec model trained on Stack Overflow posts

In [2]:
```
!wget https://zenodo.org/record/1199620/files/SO_vectors_200.bin?download
```

```
zsh:1: no matches found: https://zenodo.org/record/1199620/files/SO_vector
s_200.bin?download=1
```

In [1]:
```
from gensim.models.keyedvectors import KeyedVectors
wv_embeddings = KeyedVectors.load_word2vec_format("SO_vectors_200.bin", b
```

Как пользоваться этими векторами?

Посмотрим на примере одного слова, что из себя представляет embedding

In [2]:
```
word = 'dog'
if word in wv_embeddings:
    print(wv_embeddings[word].dtype, wv_embeddings[word].shape)
```

```
float32 (200,)
```

In [3]:
```
print(f"Num of words: {len(wv_embeddings.index_to_key)}")
```

```
Num of words: 1787145
```

Найдем наиболее близкие слова к слову `dog`:

Вопрос 1:

- Входит ли слов `cat` топ-5 близких слов к слову `dog`? Какое место?

In [4]:
```
# method most_simmilar
nearest_words = wv_embeddings.most_similar(positive=['dog'], topn=5)
for word, similarity in nearest_words:
    print(f"{word} - {similarity}")
```

```
animal - 0.8564179539680481
dogs - 0.7880867123603821
mammal - 0.7623804211616516
cats - 0.7621253728866577
animals - 0.7607938647270203
```

Ответ на Вопрос 1:

Именно слово cat, не входит, однако на 4ой позиции находится его множественное число - cats.

Векторные представления текста

Перейдем от векторных представлений отдельных слов к векторным представлениям вопросов, как к **среднему** векторов всех слов в вопросе. Если для какого-то слова нет предобученного вектора, то его нужно пропустить. Если вопрос не содержит ни одного известного слова, то нужно вернуть нулевой вектор.

In [5]:
```
import numpy as np
import re
# you can use your tokenizer
# for example, from nltk.tokenize import WordPunctTokenizer
class MyTokenizer:
    def __init__(self):
        pass
    def tokenize(self, text):
        return re.findall('\w+', text)
tokenizer = MyTokenizer()
```

In [6]:
```
def question_to_vec(question, embeddings, tokenizer, dim=200):
    """
        question: строка
        embeddings: наше векторное представление
        dim: размер любого вектора в нашем представлении

        return: векторное представление для вопроса
    """

    default_vector = []
    for word in tokenizer.tokenize(question):
      if word in embeddings:
```

```
        default_vector.append(embeddings[word])

    if default_vector:
      return np.mean(default_vector, axis=0)
    else:

      return np.zeros(dim)
```

Теперь у нас есть метод для создания векторного представления любого предложения.

Вопрос 2:

- Какая третья(с индексом 2) компонента вектора предложения `I love neural networks` (округлите до 2 знаков после запятой)?

In [7]:
```
question = "I love neural networks"
question_vector = question_to_vec(question, wv_embeddings, tokenizer)

print(question_vector[2])
```

```
-1.2854122
```

Ответ на вопрос 2:

Третья компонента вектора будет равна: -1.28

Оценка близости текстов

Представим, что мы используем идеальные векторные представления слов. Тогда косинусное расстояние между дублирующими предложениями должно быть меньше, чем между случайно взятыми предложениями.

Сгенерируем для каждого из N вопросов R случайных отрицательных примеров и примешаем к ним также настоящие дубликаты. Для каждого вопроса будем ранжировать с помощью нашей модели $R + 1$ примеров и смотреть на позицию дубликата. Мы хотим, чтобы дубликат был первым в ранжированном списке.

Hits@K

Первой простой метрикой будет количество корректных попаданий для какого-то K: $$ \text{Hits@K} = \frac{1}{N}\sum_{i=1}^N \, [rank\_q_i^{'} \le K],$$

- $\begin{equation*} [x < 0] \equiv \begin{cases} 1, &x < 0\\ 0, &x \geq 0 \end{cases} \end{equation*}$ - индикаторная функция
- q_i - i-ый вопрос
- $q_i^{'}$ - его дубликат

- $rank\_q_i^{'}$ - позиция дубликата в ранжированном списке ближайших предложений для вопроса q_i.

DCG@K

Второй метрикой будет упрощенная DCG метрика, учитывающая порядок элементов в списке путем домножения релевантности элемента на вес равный обратному логарифму номера позиции:: $$ \text{DCG@K} = \frac{1}{N} \sum_{i=1}^N\frac{1}{\log_2(1+rank\_q_i^{'})}\cdot[rank\_q_i^{'} \le K],$$ С такой метрикой модель штрафуется за большой ранк корректного ответа

Вопрос 3:

- Максимум `Hits@47 - DCG@1`?

Ответ на вопрос 3:

Максимум данного выражения будет равен 1. Он достигается когда Hits@47 = 1 и при этом DCG@1 = 0. Это может быть если корректный ответ находится на позиции 3 (на самом деле на любой между 2 из 47 включительно позициях).

Пример оценок

Вычислим описанные выше метрики для игрушечного примера. Пусть

- $N = 1$, $R = 3$
- "Что такое python?" - вопрос q_1
- "Что такое язык python?" - его дубликат $q_i^{'}$

Пусть модель выдала следующий ранжированный список кандидатов:

1. "Как изучить c++?"
2. "Что такое язык python?"
3. "Хочу учить Java"
4. "Не понимаю Tensorflow"

$\Rightarrow rank\_q_i^{'} = 2$

Вычислим метрику *Hits@K* для *K = 1, 4*:

- [K = 1] $\text{Hits@1} = [rank\_q_i^{'} \le 1)] = 0$
- [K = 4] $\text{Hits@4} = [rank\_q_i^{'} \le 4] = 1$

Вычислим метрику *DCG@K* для *K = 1, 4*:

- [K = 1] $\text{DCG@1} = \frac{1}{\log_2(1+2)}\cdot[2 \le 1] = 0$
- [K = 4] $\text{DCG@4} = \frac{1}{\log_2(1+2)}\cdot[2 \le 4] = \frac{1}{\log_2{3}}$

Вопрос 4:

- Вычислите `DCG@10`, если $rank\_q_i^{'} = 9$(округлите до одного знака после запятой)

In [8]:
```
import math

1/math.log2(10)
```

Out[8]:
```
0.3010299956639812
```

Ответ на вопрос 4:

DCG@10 = 1/log2(1+9) ~ 0.3

HITS_COUNT и DCG_SCORE

Каждая функция имеет два аргумента: dup_ranks и k. dup_ranks является списком, который содержит рейтинги дубликатов(их позиции в ранжированном списке). Например, $dup\_ranks = [2]$ для примера, описанного выше.

In [9]:
```
def hits_count(dup_ranks, k):
    """
        dup_ranks: list индексов дубликатов
        result: вернуть  Hits@k
    """
    sum = 0
    for idx in range(len(dup_ranks)):
      if dup_ranks[idx] <= k:
        sum += 1
```

```
    hits_value = sum / len(dup_ranks)
    return hits_value
```

In [10]:
```
import math

def dcg_score(dup_ranks, k):
    """
        dup_ranks: list индексов дубликатов
        result: вернуть DCG@k
    """
    sum = 0
    for i in range(len(dup_ranks)):
      if dup_ranks[i] <= k:
        sum += 1 / math.log2(1 + dup_ranks[i])
    dcg_value = sum / len(dup_ranks)
    return dcg_value
```

Протестируем функции. Пусть $N = 1$, то есть один эксперимент. Будем искать копию вопроса и оценивать метрики.

In [11]:
```
import pandas as pd
```

In [12]:
```
copy_answers = ["How does the catch keyword determine the type of excepti

# наги кандидаты
candidates_ranking = [["How Can I Make These Links Rotate in PHP",
                       "How does the catch keyword determine the type of
                       "NSLog array description not memory address",
                       "PECL_HTTP not recognised php ubuntu"],]
# dup_ranks — позиции наших копий, так как эксперимент один, то этот масс
dup_ranks = [2]

# вычисляем метрику для разных k
print('Ваш ответ HIT:', [hits_count(dup_ranks, k) for k in range(1, 5)])
print('Ваш ответ DCG:', [round(dcg_score(dup_ranks, k), 5) for k in range
```

```
Ваш ответ HIT: [0.0, 1.0, 1.0, 1.0]
Ваш ответ DCG: [0.0, 0.63093, 0.63093, 0.63093]
```

У вас должно получиться

In [13]:
```
import numpy as np

# correct_answers — метрика для разных k
correct_answers = pd.DataFrame([[0, 1, 1, 1], [0, 1 / (np.log2(3)), 1 / (
                                index=['HITS', 'DCG'], columns=range(1,5))
correct_answers
```

Out[13]:

	1	2	3	4
HITS	0	1.00000	1.00000	1.00000
DCG	0	0.63093	0.63093	0.63093

Данные

arxiv link

`train.tsv` - выборка для обучения.
В каждой строке через табуляцию записаны: **<вопрос>, <похожий вопрос>**

`validation.tsv` - тестовая выборка.
В каждой строке через табуляцию записаны: **<вопрос>, <похожий вопрос>, <отрицательный пример 1>, <отрицательный пример 2>, ...**

In [15]:
```
# !unzip stackoverflow_similar_questions.zip
```

Считайте данные.

In [16]:
```
def read_corpus(filename):
    data = []
    for line in open(filename, encoding='utf-8'):
        data.append(line.split('\t'))
    return data
```

Нам понадобиться только файл validation.

In [17]:
```
validation_data = read_corpus('./data/validation.tsv')
```

Кол-во строк

In [18]:
```
len(validation_data)
```

Out[18]: 3760

Размер нескольких первых строк

In [19]:
```
for i in range(5):
    print(i + 1, len(validation_data[i]))
```

```
1 1001
2 1001
3 1001
4 1001
5 1001
```

Ранжирование без обучения

Реализуйте функцию ранжирования кандидатов на основе косинусного расстояния. Функция должна по списку кандидатов вернуть отсортированный список пар (позиция в исходном списке кандидатов, кандидат). При этом позиция кандидата в полученном списке является его рейтингом (первый - лучший). Например, если исходный список кандидатов был [a, b, c], и самый похожий на исходный вопрос среди них - c, затем a, и в конце b, то функция должна вернуть список **[(2, c), (0, a), (1, b)]**.

In [20]:
```
from sklearn.metrics.pairwise import cosine_similarity
from copy import deepcopy
```

In [21]:
```
def rank_candidates(question, candidates, embeddings, tokenizer, dim=200)
    """
        question: строка
        candidates: массив строк(кандидатов) [a, b, c]
        result: пары (начальная позиция, кандидат) [(2, c), (0, a), (1, b
    """
    question_vector = [question_to_vec(question, embeddings, tokenizer)]

    distances = []
    for cndt in candidates:
        cndt_vector = [question_to_vec(cndt, embeddings, tokenizer)]
        cos_distance = cosine_similarity(cndt_vector, question_vector)
        distances.append(list(cos_distance)[0])

    output_distances = sorted([(value, index) for index, value in enumera
    final_output = [(idx, candidates[idx]) for v, idx in output_distances
    return final_output
```

Протестируйте работу функции на примерах ниже. Пусть $N=2$, то есть два эксперимента

In [22]:
```
questions = ['converting string to list', 'Sending array via Ajax fails']

candidates = [['Convert Google results object (pure js) to Python object'
               'C# create cookie from string and send it',
               'How to use jQuery AJAX for an outside domain?'],

              ['Getting all list items of an unordered list in PHP',
               'WPF- How to update the changes in list item of a list',
               'select2 not displaying search results']]
```

In [23]:
```
for question, q_candidates in zip(questions, candidates):
        ranks = rank_candidates(question, q_candidates, wv_embeddings, to
        print(ranks)
        print()
```

```
[(1, 'C# create cookie from string and send it'), (0, 'Convert Google resu
lts object (pure js) to Python object'), (2, 'How to use jQuery AJAX for a
n outside domain?')]

[(1, 'WPF- How to update the changes in list item of a list'), (0, 'Gettin
g all list items of an unordered list in PHP'), (2, 'select2 not displayin
g search results')]
```

Для первого экперимента вы можете полностью сравнить ваши ответы и правильные ответы. Но для второго эксперимента два ответа на кандидаты будут **скрыты**(*)

In [189…
```
# должно вывести
results = [[(1, 'C# create cookie from string and send it'),
            (0, 'Convert Google results object (pure js) to Python object
            (2, 'How to use jQuery AJAX for an outside domain?')],
           [(*, 'Getting all list items of an unordered list in PHP'), #c
            (*, 'select2 not displaying search results'), #скрыт
            (*, 'WPF- How to update the changes in list item of a list')]
```

```
  Cell In[189], line 5
    [(*, 'Getting all list items of an unordered list in PHP'), #скрыт
       ^
SyntaxError: invalid syntax
```

Последовательность начальных индексов вы должны получить `для эксперимента 1` 1, 0, 2.

Вопрос 5:

- Какую последовательность начальных индексов вы получили `для эксперимента 2` (перечисление без запятой и пробелов, например, `102` для первого эксперимента?

Ответ на вопрос 5:

Для эксперимента номер 2 последовательность будет: 102

Теперь мы можем оценить качество нашего метода. Запустите следующие два блока кода для получения результата. Обратите внимание, что вычисление расстояния между векторами занимает некоторое время (примерно 10 минут). Можете взять для validation 1000 примеров.

In [24]:
```
from tqdm.notebook import tqdm

from ipywidgets import IntProgress
from IPython.display import display
```

In [31]:
```
wv_ranking = []
max_validation_examples = 1000
for i, line in enumerate(tqdm(validation_data)):
    if i == max_validation_examples:
        break
    q, *ex = line
    ranks = rank_candidates(q, ex, wv_embeddings, tokenizer)

    wv_ranking.append([r[0] for r in ranks].index(0) + 1)
```

```
  0%|          | 0/3760 [00:00<?, ?it/s]
```

In [32]:
```
for k in tqdm([1, 5, 10, 100, 500, 1000]):
    print("DCG@%4d: %.3f | Hits@%4d: %.3f" % (k, dcg_score(wv_ranking, k)
```

```
  0%|          | 0/6 [00:00<?, ?it/s]
DCG@   1: 0.223 | Hits@   1: 0.223
DCG@   5: 0.282 | Hits@   5: 0.335
DCG@  10: 0.301 | Hits@  10: 0.392
DCG@ 100: 0.347 | Hits@ 100: 0.622
DCG@ 500: 0.372 | Hits@ 500: 0.821
DCG@1000: 0.391 | Hits@1000: 1.000
```

Эмбеддинги, обученные на корпусе похожих вопросов

In [33]:
```
train_data = read_corpus('./data/train.tsv')
```

In [38]:
```
# пример данных
train_data[3][0] + " " + train_data[3][1]
```

Out[38]:
```
'How to insert CookieCollection to CookieContainer? C# create cookie fro
m string and send it\n'
```

Улучшите качество модели.

Склеим вопросы в пары и обучим на них модель Word2Vec из gensim. Выберите размер window. Объясните свой выбор.

In [49]:
```
words = [tokenizer.tokenize(pair[0].strip() + " " + pair[1].strip()) for
```

In [50]:
```
from gensim.models import Word2Vec
embeddings_trained = Word2Vec(words, # data for model to train on
                 vector_size=200,                # embedding vector size
                 min_count=5,            # consider words that occured a
                 window=5).wv
```

In [51]:
```
wv_ranking = []
max_validation_examples = 1000
for i, line in enumerate(tqdm(validation_data)):
    if i == max_validation_examples:
        break
    q, *ex = line
    ranks = rank_candidates(q, ex, embeddings_trained, tokenizer)
    wv_ranking.append([r[0] for r in ranks].index(0) + 1)
```

```
  0%|          | 0/3760 [00:00<?, ?it/s]
```

In [52]:
```
for k in tqdm([1, 5, 10, 100, 500, 1000]):
    print("DCG@%4d: %.3f | Hits@%4d: %.3f" % (k, dcg_score(wv_ranking, k)
```

```
  0%|          | 0/6 [00:00<?, ?it/s]
DCG@   1: 0.257 | Hits@   1: 0.257
DCG@   5: 0.322 | Hits@   5: 0.381
DCG@  10: 0.350 | Hits@  10: 0.466
DCG@ 100: 0.401 | Hits@ 100: 0.722
DCG@ 500: 0.426 | Hits@ 500: 0.922
DCG@1000: 0.435 | Hits@1000: 1.000
```

Блок исследований:

В начале определим ключевые метрики, по которым мы будем сравнивать различные решения:

1. Hits@10
2. DCG@10 Выбор метрик следующий, будем предполагать, что если нужного ответа не будет в первых 10ти строках, то дальше пользователь не будет смотреть, поэтому будем брать топ 10, аналогично будем смотреть и метрику DCG@10 так как нам бы хотелось что бы правильный ответ был бы как можно выше.

Проведем следующее исслодование будем использовать следующие виды токенизации:

1. Базовая (уже реализована в классе MyTokenizer
2. Токенайзер из пакета nltk -> WordPunctTokenizer
3. Токенайзер из пакета -> gensim.utils.tokenize()

После чего выберем лучшую токенизацию и добавим еще нормализацию,для получения финального результата.

In [54]:
```
def get_validation_result(embeddings_trained, tokenizer):
    wv_ranking = []
    max_validation_examples = 1000
    for i, line in enumerate(tqdm(validation_data)):
        if i == max_validation_examples:
            break
        q, *ex = line
        ranks = rank_candidates(q, ex, embeddings_trained, tokenizer)
        wv_ranking.append([r[0] for r in ranks].index(0) + 1)
    k = 10
    print("DCG@%4d: %.3f | Hits@%4d: %.3f" % (k, dcg_score(wv_ranking, k)
```

In [55]:
```
# Эксперимент 1
# Word2Vec, базовая токенизация, без нормализации
words = [tokenizer.tokenize(pair[0].strip() + " " + pair[1].strip()) for
w2v_base_no_normalize = Word2Vec(words, vector_size=200, min_count=5, win

get_validation_result(w2v_base_no_normalize, tokenizer)
```

```
  0%|          | 0/3760 [00:00<?, ?it/s]
DCG@  10: 0.347 | Hits@  10: 0.459
```

In [58]:
```
# Эксперимент 2
# Word2Vec, токенизация из пакета nltk, без нормализации
from nltk.tokenize import WordPunctTokenizer

nltk_tokenizer = WordPunctTokenizer()
words = [nltk_tokenizer.tokenize(pair[0].strip() + " " + pair[1].strip())
w2v_nltk_no_normalize = Word2Vec(words, vector_size=200, min_count=5, win

get_validation_result(w2v_nltk_no_normalize, nltk_tokenizer)
```

```
  0%|          | 0/3760 [00:00<?, ?it/s]
DCG@  10: 0.333 | Hits@  10: 0.435
```

In [78]:
```
# Эксперимент 3
# Word2Vec, токенизация из пакета gensim, без нормализации
from gensim.utils import tokenize

class GensimTokenizer:
    def __init__(self):
        pass
    def tokenize(self, text):
        return list(tokenize(text, lower=False))

gensim_tokenizer = GensimTokenizer()

words = [gensim_tokenizer.tokenize(pair[0].strip() + " " + pair[1].strip(
```

```
w2v_gensim_no_normalize = Word2Vec(words, vector_size=200, min_count=5, w

get_validation_result(w2v_gensim_no_normalize, gensim_tokenizer)
```

```
  0%|          | 0/3760 [00:00<?, ?it/s]
DCG@  10: 0.351 | Hits@  10: 0.468
```

Лучший токенайзер

Как видно по цифрам: DCG@ 10: 0.411 | Hits@ 10: 0.538 Лучшим токенайзером оказался токенайзер из пакета gensim

Ниже будет проведен эксперимент с данным токенайзером + нормальзация слов. Под нормализацией будем предполагать следующее:

1. Удаление стоп слов
2. Удаление знаков пунктуации
3. Удаление чисел
4. Используем Портер стеммер для приведения к основе слова

In [106…

```
import string
from nltk.corpus import stopwords
from nltk.tokenize import word_tokenize
from nltk.stem import PorterStemmer

class GensimTokenizerNormalized:
    def __init__(self):
        pass

    def tokenize(self, text):
        # stop words deleting
        stop_words = set(stopwords.words('english'))
        words = word_tokenize(text)

        no_stop_words = [word for word in words if word.lower() not in st

        #delete numbers
        normalized_text = [re.sub(r'\d+', '', word) for word in no_stop_w

        # delete punctuation
        normalized_text = [word.translate(str.maketrans('', '', string.pu

        # using Porter stemmer
        stemmer = PorterStemmer()
        normalized_text = [stemmer.stem(word) for word in normalized_text

        # delete empty words like ''
        normalized_text = [word for word in normalized_text if word != ''

        return normalized_text


gensim_tokenizer_normalized = GensimTokenizerNormalized()
```

In [109…

```
words = []
for pair in tqdm(train_data):
```

```
    words.append(gensim_tokenizer_normalized.tokenize(pair[0].strip() + "

print(len(words))
print(words[100])
w2v_gensim_normalized = Word2Vec(words, vector_size=200, min_count=5, win

get_validation_result(w2v_gensim_normalized, gensim_tokenizer_normalized)
```

```
  0%|          | 0/1000000 [00:00<?, ?it/s]
1000000
['localdb', 'deploy', 'client', 'pc', 'attach', '2008', 'r2', 'databas',
'2008', 'instanc']
  0%|          | 0/3760 [00:00<?, ?it/s]
DCG@  10: 0.522 | Hits@  10: 0.651
```

Замечание:

Решить эту задачу с помощью обучения полноценной нейронной сети будет вам предложено, как часть задания в одной из домашних работ по теме "Диалоговые системы".

Напишите свой вывод о полученных результатах.

- Какой принцип токенизации даёт качество лучше и почему?
- Помогает ли нормализация слов?
- Какие эмбеддинги лучше справляются с задачей и почему?
- Почему получилось плохое качество решения задачи?
- Предложите свой подход к решению задачи.

Вывод:

Лучшее качественно на нашей метрике, которое удалось достич это DCG@10 = 0.522 и Hits@10 = 0.651, для сравнения но самая базовая модель и эмбеддинги на предобученной модели дали лишь DCG@ = 0.301 и Hits@ = 0.392.

1. Лучше всего сработал токенайзер из библиотеки gensim.
2. Нормализация довольно сильно улучшила итоговое качество.
3. Лучше справляются эмбеддинги из word2vec, причина в том, что данные эмбеддинги были получены на основе обучающей выборке для конкретной задачи, а не из общих данных.
4. Плохое качество получилось ввиду того, что в данной задаче много специфических слов и нужно очень аккуратно подходить к их обработке + возможно сама модель не обладает досаточной обобщающей способностью, что бы выучить оптимальные эмбеддинги для слов.
5. Во-первых, стоит вложиться в улучшение процедеры нормализации слов и их токенизации, как мы видели, но даже простое добавление очищение от стоп слов, удаление символов пунктуации и чесел дало хороший буст. Во-вторых, попробовать поменять размер окна при обучении w2v модели, а также изменить требования по частоте встречаемости, возможно имеет

смысл и более редкие слова обрабатывать. В-третьих, можно поробовать модель Glove, так как она работает на более мелких токенах (не словах).