Deep Learning School

0.1 Физтех-Школа Прикладной математики и информатики (ФПМИ) МФТИ

Some parts of the notebook are almost the copy of [mmta-team course (https://github.com/mmta-team/mmta_fall_2020)](). Special thanks to mmta-team for making them publicly available. [Original notebook (https://github.com/mmta-team/mmta_fall_2020/blob/master/tasks/01_word_embeddings/task_word_embeddings.ipynb)]().

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

1 Задача поиска схожих по смыслу предложений

Мы будем ранжировать вопросы StackOverflow (https://stackoverflow.com) на основе семантического векторного представления

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

2 Задача ранжирования(Learning to Rank)

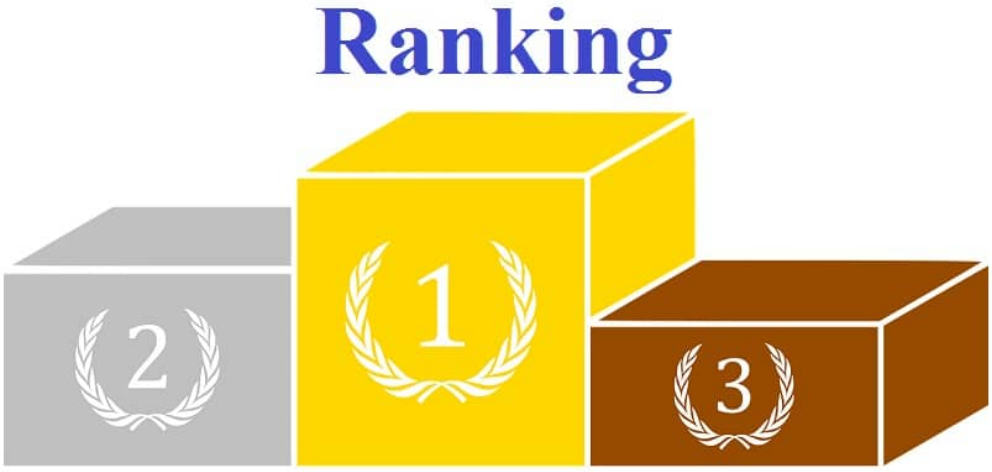
- X - множество объектов

- $X^l = \{x_1, x_2, \ldots, x_l\}$ - 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

2.1 Задача:

построить ранжирующую функцию $a : X \to R$ такую, что

$$i \prec j \Rightarrow a(x_i) < a(x_j)$$



2.2 Embeddings

Будем использовать предобученные векторные представления слов на постах Stack Overflow.
A word2vec model trained on Stack Overflow posts (https://github.com/vefstathiou/SO_word2vec)

```
!wget https://zenodo.org/record/1199620/files/SO_vectors_200.bin?download=1
```

Ввод [1]:

```
from gensim.models.keyedvectors import KeyedVectors
wv_embeddings = KeyedVectors.load_word2vec_format("SO_vectors_200.bin", binary=True)
```

executed in 14.3s, finished 16:22:39 2021-10-01

2.2.1 Как пользоваться этими векторами?

Посмотрим на примере одного слова, что из себя представляет embedding

Ввод [2]:

```
word = 'dog'
if word in wv_embeddings:
    print(wv_embeddings[word].dtype, wv_embeddings[word].shape) # вектор размерности 200
```

executed in 7ms, finished 16:23:03 2021-10-01

```
float32 (200,)
```

Ввод [3]:

```
print(f"Num of words: {len(wv_embeddings.index2word)}")
```

executed in 15ms, finished 16:23:04 2021-10-01

```
Num of words: 1787145
```

Найдем наиболее близкие слова к слову dog :

2.2.2 Вопрос 1:

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

Ввод [4]:

```
import numpy as np
```

executed in 11ms, finished 16:23:05 2021-10-01

Ввод [5]:

```
def cosine_sim(a,b):
    return a@b/np.linalg.norm(a)/np.linalg.norm(b)
```

executed in 6ms, finished 16:23:08 2021-10-01

Ввод [6]:

```
cs = []
for word in wv_embeddings.vocab:
    cs.append((word,cosine_sim(wv_embeddings['dog'],wv_embeddings[word])))
```

executed in 32.6s, finished 16:23:41 2021-10-01

Ввод [7]:

```
cs.sort(key = lambda x: x[1],reverse=True)
cs[:5]
```

executed in 1.64s, finished 16:23:43 2021-10-01

Out[7]:

```
[('dog', 1.0),
 ('animal', 0.856418),
 ('dogs', 0.78808665),
 ('mammal', 0.7623805),
 ('cats', 0.76212525)]
```

Ввод [8]:

```
cs.index(('cat', cosine_sim(wv_embeddings['dog'],wv_embeddings['cat'])))
```

executed in 8ms, finished 16:23:44 2021-10-01

Out[8]:

```
26
```

> Ответ: Нет, не входит; находится на 27 месте.

2.3 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.

Ввод [9]:

```
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()
```

executed in 9ms, finished 16:23:49 2021-10-01

Ввод [10]:

```
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
    """
    tokens = tokenizer.tokenize(question)
    emb = np.zeros(dim)
    for token in tokens:
        if token in embeddings.vocab:
            emb += embeddings[token]
    return emb / len(tokens)
```

executed in 20ms, finished 16:23:50 2021-10-01

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

2.3.1 Вопрос 2:

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

Ввод [11]:

```
round(question_to_vec('I love neural networks', wv_embeddings, tokenizer)[2],2)
```

executed in 16ms, finished 16:23:52 2021-10-01

Out[11]:

```
-0.96
```

> Ответ: -0.96

2.4 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.

2.4.1 Hits@K

Первой простой метрикой будет количество корректных попаданий для какого-то K:

$$\text{Hits@K} = \frac{1}{N}\sum_{i=1}^{N}[rank\_q_i^{'} \leq K],$$

- $[x < 0] \equiv \begin{cases} 1, & x < 0 \\ 0, & x \geq 0 \end{cases}$ - 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.

Суть данной метрики в контроле позиции дубликата с помощью K (если K = 5, то к метрике для всего что выше K прибавляется 1/N, для всего что ниже +0) (если rank меньше K, то хорошо)

2.4.2 DCG@K

Второй метрикой будет упрощенная DCG метрика, учитывающая порядок элементов в списке путем домножения релевантности элемента на вес равный обратному логарифму номера позиции::

$$\text{DCG@K} = \frac{1}{N}\sum_{i=1}^{N}\frac{1}{\log_2(1+rank\_q_i^{'})}\cdot[rank\_q_i^{'} \leq K],$$

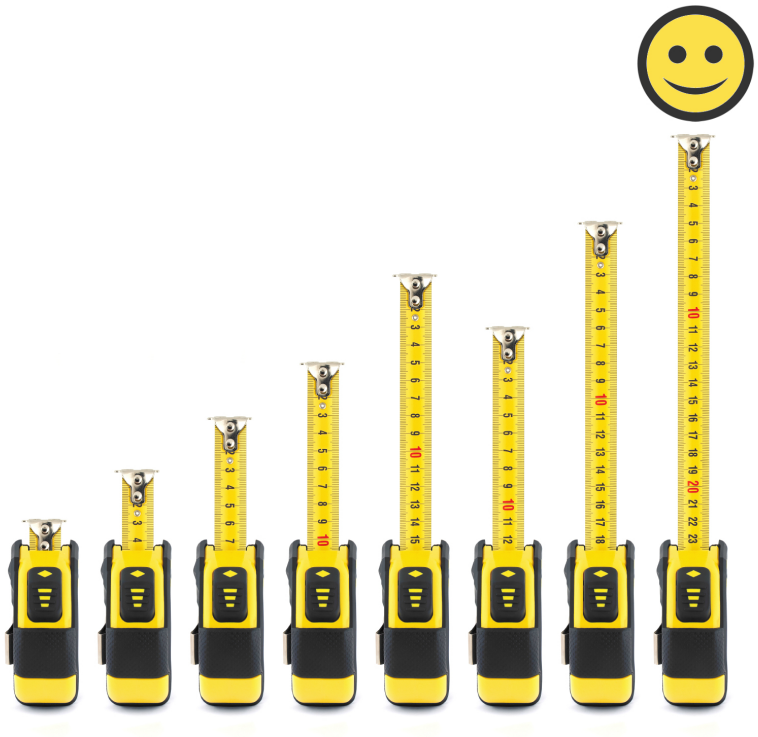
С такой метрикой модель штрафуется за большой ранк корректного ответа

Суть DCG такая же, как у Hits но идёт штраф за большой rank

!!! Обе метрики в идеале должны быть равны 1 !!!

2.4.3 Вопрос 3:

- Максимум `Hits@47 - DCG@1` ?



> Ответ: 1, т.к max Hits@K это 1, а min DCG@K это 0, т.е 1 - 0

2.4.4 Пример оценок

Вычислим описанные выше метрики для игрушечного примера. Пусть

- $N = 1$ -- пример для 1 вопроса, $R = 3$ -- 3 случайных отрицательных примера
- "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"

$\Rightarrow rank\_q_i^{'} = 2$ -- позиция дубликата

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

- [K = 1] $\text{Hits@1} = [rank\_q_i^{'} \leq 1)] = 0$ -- если в скобках выполняется, то +1 (но rank_q_i = 2)
- [K = 4] $\text{Hits@4} = [rank\_q_i^{'} \leq 4] = 1$

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

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

2.4.5 Вопрос 4:

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

Ввод [12]:

```
round(1 / np.log2(1+9),1)
```

executed in 20ms, finished 16:23:59 2021-10-01

Out[12]:

```
0.3
```

> Ответ: 1/log2(1+9) = 0.3

2.5 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.

Ввод [13]:

```
def hits_count(dup_ranks, k):
    """
        dup_ranks: list индексов дубликатов
        result: вернуть  Hits@k
    """
    len_dup_ranks = len(dup_ranks)
    hits_sum = 0
    for rank in dup_ranks: hits_sum += 1 if rank <= k else 0
    return hits_sum / len_dup_ranks
```

executed in 13ms, finished 16:24:02 2021-10-01

Ввод [14]:

```
def dcg_score(dup_ranks, k):
    """
        dup_ranks: list индексов дубликатов
        result: вернуть DCG@k
    """
    len_dup_ranks = len(dup_ranks)
    dcg_sum = 0
    for rank in dup_ranks: dcg_sum += 1/np.log2(1+rank) if rank <= k else 0
    return dcg_sum / len(dup_ranks)
```

executed in 8ms, finished 16:24:03 2021-10-01

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

Ввод [15]:

```
import pandas as pd
```

executed in 1.93s, finished 16:24:06 2021-10-01

Ввод [16]:

```
copy_answers = ["How does the catch keyword determine the type of exception that was thrown

# наги кандидаты
candidates_ranking = [["How Can I Make These Links Rotate in PHP",
                       "How does the catch keyword determine the type of exception that was
                       "NSLog array description not memory address",
                       "PECL_HTTP not recognised php ubuntu"],]
# dup_ranks — позиции наших копий, так как эксперимент один, то этот массив длины 1
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(1, 5)])
```

executed in 14ms, finished 16:24:06 2021-10-01

```
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

Ввод [17]:

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

executed in 21ms, finished 16:24:08 2021-10-01

Out[17]:

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

2.6 Данные

arxiv link (https://drive.google.com/file/d/1QqT4D0EoqJTy7v9VrNCYD-m964XZFR7_/edit)

`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>, ...**

```
!unzip stackoverflow_similar_questions.zip
```

Считайте данные.

Ввод [18]:

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

executed in 7ms, finished 16:24:12 2021-10-01

Нам понадобиться только файл validation.

Ввод [19]:

```
validation_data = read_corpus('./data/validation.tsv')
```

executed in 1.15s, finished 16:24:15 2021-10-01

Кол-во строк

Ввод [20]:

```
len(validation_data)
```

executed in 17ms, finished 16:24:20 2021-10-01

Out[20]:

```
3760
```

Размер нескольких первых строк

Ввод [21]:

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

executed in 16ms, finished 16:24:22 2021-10-01

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

2.7 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)]**.

Ввод [22]:

```
from sklearn.metrics.pairwise import cosine_similarity
from copy import deepcopy
```

executed in 578ms, finished 16:24:26 2021-10-01

Ввод [23]:

```
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)]
    """
    pair = []
    cos_sim = []
    ques_vec = question_to_vec(question, embeddings, tokenizer, dim)
    for cand in candidates:
        pair.append((candidates.index(cand),cand))
        cos_sim.append(cosine_similarity(ques_vec.reshape(1,-1),question_to_vec(cand,embedd
    pair = [x[0] for x in sorted(zip(pair,cos_sim),key = lambda x: x[1],reverse=True)]
    return pair
```

executed in 21ms, finished 16:24:27 2021-10-01

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

Ввод [24]:

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

candidates = [['Convert Google results object (pure js) to Python object', # первый экспери
               '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',      # второй эксперим
               'WPF- How to update the changes in list item of a list',
               'select2 not displaying search results']]
```

executed in 19ms, finished 16:24:29 2021-10-01

Ввод [25]:

```
for question, q_candidates in zip(questions, candidates):
        ranks = rank_candidates(question, q_candidates, wv_embeddings, tokenizer)
        print(ranks)
        print()
```

executed in 26ms, finished 16:24:30 2021-10-01

```
[(1, 'C# create cookie from string and send it'), (0, 'Convert Google result
s object (pure js) to Python object'), (2, 'How to use jQuery AJAX for an ou
tside domain?')]

[(1, 'WPF- How to update the changes in list item of a list'), (0, 'Getting
all list items of an unordered list in PHP'), (2, 'select2 not displaying se
arch results')]
```

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

Ввод [26]:

```
# 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?')],
           [(1, 'Getting all list items of an unordered list in PHP'), #был скрыт
            (0, 'select2 not displaying search results'), #был скрыт
            (2, 'WPF- How to update the changes in list item of a list')]] #был скрыт
```

executed in 16ms, finished 16:24:39 2021-10-01

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

2.7.1 Вопрос 5:

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

> Ответ: 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.

Ввод [27]:

```
from tqdm.notebook import tqdm
```

executed in 166ms, finished 16:24:43 2021-10-01

Ввод [28]:

```
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)
```

executed in 18m 37s, finished 16:44:04 2021-10-01

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

Ввод [29]:

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

executed in 72ms, finished 16:44:24 2021-10-01

```
  0%|          | 0/6 [00:00<?, ?it/s]

DCG@   1: 0.287 | Hits@   1: 0.287
DCG@   5: 0.348 | Hits@   5: 0.402
DCG@  10: 0.368 | Hits@  10: 0.465
DCG@ 100: 0.413 | Hits@ 100: 0.691
DCG@ 500: 0.437 | Hits@ 500: 0.876
DCG@1000: 0.450 | Hits@1000: 1.000
```

2.8 Эмбеддинги, обученные на корпусе похожих вопросов

Ввод [30]:

```
train_data = read_corpus('./data/train.tsv')
```

executed in 2.00s, finished 16:44:34 2021-10-01

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

Так как предложения являются небольшими ~10 слов в среднем, то можно не брать большие значения, однако и маленькое значение даст плохую работу с контекстом, поэтому показалось, что 3 будет оптимальным размером окна.

Ввод [31]:

```
window = 3
words = [tokenizer.tokenize(' '.join(x)) for x in train_data]
```

executed in 10.4s, finished 16:45:01 2021-10-01

Ввод [32]:

```
from gensim.models import Word2Vec
embeddings_trained = Word2Vec(words,
                 size=200,
                 min_count=5,
                 window=window).wv
```

executed in 1m 33.8s, finished 16:47:23 2021-10-01

Ввод [33]:

```
wv_ranking = []
#max_validation_examples = 100
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)
```

executed in 21m 15s, finished 17:09:28 2021-10-01

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

Ввод [34]:

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

executed in 62ms, finished 17:12:31 2021-10-01

```
  0%|          | 0/6 [00:00<?, ?it/s]

DCG@   1: 0.239 | Hits@   1: 0.239
DCG@   5: 0.308 | Hits@   5: 0.370
DCG@  10: 0.329 | Hits@  10: 0.436
DCG@ 100: 0.380 | Hits@ 100: 0.686
DCG@ 500: 0.409 | Hits@ 500: 0.910
DCG@1000: 0.419 | Hits@1000: 1.000
```

2.9 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.

3 Вывод:

1) Лемматизация, так как не у всех слов можно просто отбрасывать некоторую часть (что и делает стемминг), но с другой стороны для её применения нам необходимо иметь заранее составленный набор нормальных форм слов.

2) Конечно, так как при её отсутвии в словарь могут попасть различные формы слова, но при этом имещющие один и тот же смысл, из-за чего его размер может стать слишком большим.

3) На малых объёмах данных обычно лучше справляются предобученные эмбеддинги, оссобенно схожие с темой имеющегося набора текста.

4) Много схожих вопросов, а также общая тема и много специфических слов.

5) Можно использовать tf-idf, за счёт его способности выделять уникальные слова, то есть учитывать и частоту слова в документе и общую встречаемость слова)

Ввод [ ]:

```
```