Школа глубокого обучения ФПМИ МФТИ

Домашнее задание. Эмбеддинги.

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 ноутбука напишите свой вывод. Работа без вывода оценивается ниже.

Настройка окружения

> python 3.11

numpy==1.26.4

scipy==1.10.1

scikit-learn==1.3.0

pandas==2.0.3

gensim==4.3.2

nltk==3.8.1

jupyter

```
import numpy as np
import pandas as pd
import sklearn
import gensim
import nltk

print(f"NumPy: {np.__version__}")
print(f"Pandas: {pd.__version__}")
print(f"Scikit-learn: {sklearn.__version__}")
print(f"Gensim: {gensim.__version__}")
print(f"Gensim: {nltk.__version__}")
```

```
NumPy: 1.26.4
Pandas: 2.0.3
Scikit-learn: 1.3.0
Gensim: 4.3.2
Gensim: 3.8.1
```

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 ранжирования, поэтому введем её математическую формулировку.

Задача ранжирования(Learning to Rank)

Основная идея

Пользователь на Stack Overflow задает вопрос, а система должна найти уже существующие вопросы, которые означают то же самое, даже если сформулированы другими словами. Это важно! Система ищет не ответы, а вопросы похожие на заданный вопрос. Термин *Дубликат* означает максимально похожий вопрос на исходный. "Дубликат" дублирует не текст, а смысл (намерение, семантику) исходного вопроса. Система ищет в своей базе смысловые дубликаты - вопросы, которые хоть и сформулированы по-другому, но по сути задают одно и то же.

a* 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

Задача:

построить ранжирующую функцию $a : X \to R$ 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]

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

```
--2025-02-15 10:45:40--  https://zenodo.org/record/1199620/files/SO_vectors_200.
Resolving zenodo.org (zenodo.org)... 188.185.43.25, 188.185.48.194, 188.185.45.9
Connecting to zenodo.org (zenodo.org)|188.185.43.25|:443... connected.
HTTP request sent, awaiting response... 301 MOVED PERMANENTLY
Location: /records/1199620/files/SO_vectors_200.bin [following]
--2025-02-15 10:45:41--  https://zenodo.org/records/1199620/files/SO_vectors_200
```

```
Reusing existing connection to zenodo.org:443.
HTTP request sent, awaiting response... 200 OK
Length: 1453905423 (1.4G) [application/octet-stream]
Saving to: ‘SO_vectors_200.bin?download=1’

SO_vectors_200.bin? 100%[===================>]   1.35G  13.1MB/s    in 1m 55s

2025-02-15 10:47:36 (12.1 MB/s) - ‘SO_vectors_200.bin?download=1’ saved [1453905
```

```
from gensim.models.keyedvectors import KeyedVectors
```

```
wv_embeddings = KeyedVectors.load_word2vec_format("D:\Data\StackOverflow\SO_vec
```

Как пользоваться этими векторами?

Рассмотрим на примере одного слова, что из себя представляет embedding.

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

```
float32 (200,)
```

```
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 слово `cat` в топ-5 близких слов к слову `dog`?
- Какое место занимает слово `cat` в наиболее близких словах к слову `dog`?

```
# method most_simmilar
similar_words = wv_embeddings.most_similar("dog", topn=5)

for word, similarity in similar_words:
    print(f"{word}: {similarity:.4f}")
```

```
animal: 0.8564
dogs: 0.7881
mammal: 0.7624
cats: 0.7621
animals: 0.7608
```

```
similar_words = wv_embeddings.most_similar("dog", topn=1000)

words_only = [word for word, similarity in similar_words]

if "cat" in words_only:
    cat_rank = words_only.index("cat") + 1
    cat_similarity = similar_words[cat_rank - 1][1]  # получаем схожесть

    print(f"Слово 'cat' занимает {cat_rank}-е  к 'dog'")
    print(f"Косинусная расстояние: {cat_similarity:.4f}")
```

```
Слово 'cat' занимает 26-е  к 'dog'
Косинусная расстояние: 0.6852
```

Ваш ответ:

> слово 'cat' не входит в в топ-5 близких слов к слову `dog`

> слово 'cat' занимает 26-место в наиболее близких словах к слову `dog`

Ряд показательных примеров для операций с векторами

```
wv_embeddings.most_similar('javascript', topn=100)
```

```
[('javscript', 0.7809842824935913),
 ('javacript', 0.7594384551048279),
 ('javascipt', 0.7432674765586853),
 ('java-script', 0.7240772843360901),
 ('jquery', 0.7194275856018066),
 ('js', 0.704460620880127),
 ('javascript/jquery', 0.6996371746063232),
 ('javasript', 0.6899282336235046),
 ('javascrip', 0.6709383130073547),
 ('jquery/javascript', 0.6435250043869019),
 ('javascripts', 0.6310728788375854),
 ('javacsript', 0.6218552589416504),
 ('javasscript', 0.619502067565918),
 ('js/jquery', 0.6146624088287354),
 ('javascirpt', 0.6112344861030579),
 ('javasctipt', 0.607463002204895),
 ('ajax', 0.6062032580375671),
 ('html', 0.6023529171943665),
 ('client-side', 0.5959587097167969),
 ('dom', 0.5954562425613403),
 ('html/javascript', 0.588080108165741),
 ('javascript-code', 0.5816146731376648),
 ('dhtml', 0.577498733997345),
 ('javescript', 0.5723744630813599),
 ('jscript', 0.5716807246208191),
 ('javascrpt', 0.5711583495140076),
 ('jquery/js', 0.568124532699585),
 ('server-side', 0.5672400593757629),
 ('javascript/ajax', 0.5670605301856995),
 ('mootools', 0.5584954619407654),
```

```
 ('javascrit', 0.5563507676124573),
 ('xmlhttprequest', 0.5551258325576782),
 ('iframe', 0.5517319440841675),
 ('js-code', 0.5484967231750488),
 ('swfobject', 0.5398107767105103),
 ('serverside', 0.5389335751533508),
 ('clientside', 0.5377546548843384),
 ('css', 537625253200531),
 ('js/ajax', 0.5316885709762573),
 ('html5', 0.5311249494552612),
 ('javsacript', 0.5302574634552002),
 ('jquery/', 0.5289410948753357),
 ('xhr', 0.5275776982307434),
 ('paperscript', 0.5226115584373474),
 ('javsscript', 0.5212085843086243),
 ('iframes', 0.5211840271949768),
 ('cleditor', 0.5200235843658447),
 ('javascript/html', 0.519462525844574),
 ('javascritp', 0.5188539624214172),
 ('html/js', 0.5167841911315918),
 ('noscript', 0.5114295482635498),
 ('content-script', 0.5102307796478271),
 ('jsni', 0.5088101625442505),
 ('non-script', 0.5061529278755188),
 ('browser-side', 0.5060920119285583),
 ('markup', 0.5037639737129211),
 ('onload', 0.5031086206436157),
```

```
wv_embeddings.similarity('pandas', 'numpy')
```

```
0.67640203
```

```
wv_embeddings.similarity('pandas', 'laravel')
```

```
0.079919524
```

> Найти что-то (?), что относится к mongodb так же, как python относится к javascript.

```
wv_embeddings.most_similar(positive=['python', 'mongodb'], negative=['javascrip
```

```
[('pymongo', 0.6239628195762634),
 ('mongo', 0.6117824912071228),
 ('cassandra', 0.5893003940582275),
 ('postgresql', 0.5870639085769653),
 ('rethinkdb', 0.5810126662254333),
 ('postgres', 0.5799827575683594),
 ('aerospike', 0.5758125185966492),
 ('mongo-connector', 0.5724862813949585),
 ('arangodb', 0.5705552697181702),
 ('python3', 0.5391701459884644)]
```

> Найти что-то (?), что относится к javascript так же, как django относится к python (фреймворк).

```
wv_embeddings.most_similar(positive=['django', 'javascript'], negative=['pythor
```

```
[('jquery', 0.6296848654747009),
 ('angularjs', 0.6073837280273438),
 ('ajax', 0.5964499115943909),
 ('js', 0.5861358642578125),
 ('backbone', 0.5699291825294495),
 ('javascripts', 0.562743604183197),
 ('jqm', 0.562324583530426),
 ('angular', 0.5492519736289978),
 ('client-side', 0.5395373106002808),
 ('javscript', 0.5311560034751892)]
```

> Найти что-то (?), что относится к nosql так же, как mysql относится к sql (конкретная реализация).

```
wv_embeddings.most_similar(positive=['mysql', 'nosql'], negative=['sql'])
```

```
[('no-sql', 0.7577500939369202),
 ('non-relational', 0.7293692827224731),
 ('document-oriented', 0.7177333831787109),
 ('schema-less', 0.6574867367744446),
 ('schema-free', 0.6288107633590698),
 ('schemaless', 0.6252644658088684),
 ('newsql', 0.6230009198188782),
 ('couchdb', 0.6175366044044495),
 ('column-oriented', 0.6124289631843567),
 ('odbms', 0.6096857786178589)]
```

```
wv_embeddings.doesnt_match(['git', 'commit', 'push', 'mysql'])
```

```
'mysql'
```

```
wv_embeddings.doesnt_match(['php', 'python', 'man', 'ai'])
```

```
'ai'
```

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.

```
import numpy as np
import re
class MyTokenizer:
    def __init__(self):
        pass
    def tokenize(self, text):
        return re.findall('\w+', text)
tokenizer = MyTokenizer()
```

```
import numpy as np
from nltk.tokenize import word_tokenize

def question_to_vec(question, embeddings, tokenizer, dim=200):
    """
    Преобразует вопрос в вектор как среднее векторов всех слов в вопросе.

    Args:
        question: строка с вопросом
        embeddings: объект KeyedVectors с векторными представлениями слов
        tokenizer: функция для токенизации текста (например, word_tokenize из r
        dim: размерность векторов в представлении

    Returns:
        numpy array размерности (dim,) - векторное представление вопроса
    """
    words = tokenizer(question.lower())  # приводим к нижнему регистру

    # Собираем векторы для всех слов, которые есть в embeddings
    word_vectors = []

    for word in words:
        # Проверяем, есть ли слово в словаре эмбеддингов
        if word in embeddings.key_to_index:  # для Gensim 4.0.0+
        # if word in embeddings.vocab:  # для старых версий Gensim
            word_vector = embeddings[word]
            word_vectors.append(word_vector)

    # Если нашли хотя бы одно слово с вектором
    if len(word_vectors) > 0:
        # Вычисляем среднее векторов
        question_vector = np.mean(word_vectors, axis=0)
        return question_vector
    else:
        # Если ни одного слова не найдено, возвращаем нулевой вектор
        return np.zeros(dim)
```

```
# Преобразуем предложение в вектор
sentence = "I love neural networks"
sentence_vector = question_to_vec(sentence, wv_embeddings, word_tokenize)

# Получаем третью компоненту (индекс 2)
third_component = sentence_vector[2]
```

```
# Округляем до 2 знаков после запятой
rounded_component = round(third_component, 2)

print(f"Третья компонента вектора: {third_component}")
print(f"Округленное значение: {rounded_component}")
print(f"Размерность всего вектора: {sentence_vector.shape}")
```

```
Третья компонента вектора: -1.2854121923446655
Округленное значение: -1.2899999618530273
Размерность всего вектора: (200,)
```

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

```
word_tokenize(sentence)
```

```
['I', 'love', 'neural', 'networks']
```

```
'networks' in wv_embeddings.key_to_index
```

```
True
```

```
sentence = "I love neural networks"
sentence_vector = question_to_vec(sentence, wv_embeddings, word_tokenize)
third_component = round(sentence_vector[2], 2)
print(f"Третья компонента: {third_component:.2f}")
```

```
Третья компонента: -1.29
```

Вопрос 2:

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

Ваш ответ:

- Третья компонента: -1.29

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:

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

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

Hits@K измеряет долю вопросов, для которых правильный ответ попал в топ-K позиций среди отранжированных кандидатов.

> Hits@K отвечает на простой вопрос: в каком проценте случаев наш правильный ответ (дубликат) оказался в топ-K результатов поиска?" "Чем выше значение Hits@K, тем лучше наша модель. Обычно K выбирают небольшим: 1, 3, 5, 10, потому что пользователи редко смотрят дальше первой страницы выдачи."

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.

DCG@K измеряет качество ранжирования, учитывая не только факт наличия правильного ответа в топ-K, но и ***его точную позицию***.

> DCG@K — **Discounted** Cumulative Gain на первых K позициях."

> DCG@K это метрика, которая штрафует модель за то, что она ставит правильный ответ не на самое верхнее положение.

> DCG@K является упрощенной потому что не учитывет релевантность документов (запросов).

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 *DCG@K* для *K = 1, 4*:

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

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

- [K = 1] $\mathrm{Hits@1} = [rank\_q_i^{'} \leq 1]$

Проверяем условие $\mathrm{rank}_{q_1'} \leq 1$: ***условие неверно***.

Следовательно, $[\mathrm{rank}_{q_1'} \leq 1] = 0$.

[K = 1] $\mathrm{Hits@1} = 0$

- [K = 4] $\text{Hits@4} = [rank\_q_i^{'} \leq 4]$

Проверяем условие $\text{rank}_{q_1'} \leq 4$: ***условие верно***.

[K = 4] $\text{Hits@4} = 1$

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

Вопрос 3:

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

Ваш ответ: 0,3

Ваше решение: [K = 10]
$\text{DCG@10} = \frac{1}{\log_2(1+9)} \cdot [9 \leq 10] = \frac{1}{\log_2(10)} = \frac{1}{3.3219} = 0,301$

Более сложный пример оценок

Рассмотрим пример с $N > 1$, где $N = 3$ (три вопроса) и для каждого вопроса заданы позиции их дубликатов. Вычислим метрики **Hits@K** для разных значений K.

- $N = 3$: Три вопроса (q_1, q_2, q_3).
- Для каждого вопроса известна позиция его дубликата ($\text{rank}_{q_i'}$):
  - $\text{rank}_{q_1'} = 2$,
  - $\text{rank}_{q_2'} = 5$,
  - $\text{rank}_{q_3'} = 1$.

Мы будем вычислять **Hits@K** для $K = 1, 5$.

Для $K = 1$:

Подставим значения:

$$\text{Hits@1} = \frac{1}{3} \cdot \left([\text{rank}_{q_1'} \leq 1] + [\text{rank}_{q_2'} \leq 1] + [\text{rank}_{q_3'} \leq 1] \right).$$

Проверяем условие $\text{rank}_{q_i'} \leq 1$ для каждого вопроса:

- $\text{rank}_{q_1'} = 2 \rightarrow 2 \not\leq 1 \rightarrow 0$,
- $\text{rank}_{q_2'} = 5 \rightarrow 5 \not\leq 1 \rightarrow 0$,
- $\text{rank}_{q_3'} = 1 \rightarrow 1 \leq 1 \rightarrow 1$.

Сумма:

$$\text{Hits@1} = \frac{1}{3} \cdot (0 + 0 + 1) = \frac{1}{3}.$$

$$\boxed{\text{Hits@1} = \frac{1}{3}}.$$

Для $K = 5$:

Подставим значения:

$$\text{Hits@5} = \frac{1}{3} \cdot \left([\text{rank}_{q'_1} \leq 5] + [\text{rank}_{q'_2} \leq 5] + [\text{rank}_{q'_3} \leq 5] \right).$$

Проверяем условие $\text{rank}_{q'_i} \leq 5$ для каждого вопроса:

- $\text{rank}_{q'_1} = 2 \rightarrow 2 \leq 5 \rightarrow 1$,
- $\text{rank}_{q'_2} = 5 \rightarrow 5 \leq 5 \rightarrow 1$,
- $\text{rank}_{q'_3} = 1 \rightarrow 1 \leq 5 \rightarrow 1$.

Сумма:

$$\text{Hits@5} = \frac{1}{3} \cdot (1 + 1 + 1) = 1.$$

$$\boxed{\text{Hits@5} = 1}.$$

Теперь вычислим метрику **DCG@K** для того же примера, где $N = 3$ (три вопроса), и для каждого вопроса известна позиция его дубликата ($\text{rank}_{q'_i}$):

- $\text{rank}_{q'_1} = 2$,
- $\text{rank}_{q'_2} = 5$,
- $\text{rank}_{q'_3} = 1$.

Мы будем вычислять **DCG@K** для $K = 1, 5$.

Для $K = 1$: Подставим значения:

$$\text{DCG@1} = \frac{1}{3} \cdot \left(\frac{1}{\log_2(1 + \text{rank}_{q'_1})} \cdot [\text{rank}_{q'_1} \leq 1] + \frac{1}{\log_2(1 + \text{rank}_{q'_2})} \cdot [\text{rank}_{q'_2} \leq \right.$$

Проверяем условие $\text{rank}_{q'_i} \leq 1$ для каждого вопроса:

- $\text{rank}_{q'_1} = 2 \rightarrow 2 \nleq 1 \rightarrow 0$,
- $\text{rank}_{q'_2} = 5 \rightarrow 5 \nleq 1 \rightarrow 0$,
- $\text{rank}_{q'_3} = 1 \rightarrow 1 \leq 1 \rightarrow 1$.

Сумма:

$$\text{DCG@1} = \frac{1}{3} \cdot (0 + 0 + 1) = \frac{1}{3}.$$

$$\boxed{\text{DCG@1} = \frac{1}{3}}.$$

Для $K = 5$: Подставим значения:

$$\text{DCG@5} = \frac{1}{3} \cdot \left(\frac{1}{\log_2(1 + \text{rank}_{q_1'})} \cdot [\text{rank}_{q_1'} \leq 5] + \frac{1}{\log_2(1 + \text{rank}_{q_2'})} \cdot [\text{rank}_{q_2'} \leq 5$$

Проверяем условие $\text{rank}_{q_i'} \leq 5$ для каждого вопроса:

- $\text{rank}_{q_1'} = 2 \rightarrow 2 \leq 5 \rightarrow 1$,
- $\text{rank}_{q_2'} = 5 \rightarrow 5 \leq 5 \rightarrow 1$,
- $\text{rank}_{q_3'} = 1 \rightarrow 1 \leq 5 \rightarrow 1$.

Сумма:

$$\text{DCG@5} = \frac{1}{3} \cdot (0.631 + 0.387 + 1) = \frac{1}{3} \cdot 2.018 \approx 0.673.$$

$$\boxed{\text{DCG@5} \approx 0.673}.$$

Вопрос 4:

- Какое максимальное значение может принимать `Hits@47 - DCG@1`?

Ваш ответ: 46/47 ≈ 0.9787

Ваше решение:

- Лучший случай для Hits@47: все 47 документов релевантны, тогда Hits@47 = 1.0
- Худший случай для DCG@1: первый документ нерелевантен, тогда DCG@1 = 0

Чтобы достичь этого:

- Первый документ: нерелевантный
- Остальные 46 документов в топ-47: релевантные
- Все документы после 47-го: любые

Тогда:

Hits@47 = 46/47 ≈ 0.9787

DCG@1 = 0

Hits@47 - DCG@1 ≈ 0.9787

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).

К примеру для "Что такое язык python?" $dup\_ranks = [2]$.

```
def hits_count(dup_ranks, k):
    """
        dup_ranks: список позиций дубликатов в ранжированном списке
        k: пороговое значение для ранга
        result: вернуть Hits@k (доля успешных экспериментов)
    """
    hits = 0
    N = len(dup_ranks)  # количество экспериментов
    for rank in dup_ranks:
        if rank <= k:
            hits += 1
    return hits / N
```

```
# Расчитаем метрику Hits с помощью реализованных функций
dup_ranks = [2]

hits_value = hits_count(dup_ranks, k=1)
print(f"Hits@1 = {hits_value:.3f}")

hits_value = hits_count(dup_ranks, k=4)
print(f"Hits@4 = {hits_value:.3f}")
```

```
Hits@1 = 0.000
Hits@4 = 1.000
```

```
# Расчитаем метрику Hits с помощью реализованных функций из предыдущего примера
dup_ranks = [2, 5, 1]

hits_value = hits_count(dup_ranks, k=1)
print(f"Hits@1 = {hits_value:.3f}")

hits_value = hits_count(dup_ranks, k=5)
print(f"Hits@4 = {hits_value:.3f}")
```

```
Hits@1 = 0.333
Hits@4 = 1.000
```

```
import math

def dcg_score(dup_ranks, k):
    """
        dup_ranks: список позиций дубликатов в ранжированном списке
        k: пороговое значение для ранга
        result: вернуть DCG@k (нормированное значение)
    """
    dcg_value = 0.0
    N = len(dup_ranks)  # количество экспериментов

    for rank in dup_ranks:
        if rank <= k:
            dcg_value += 1.0 / math.log2(rank + 1)
```

```
    # Нормировка на количество экспериментов
    return dcg_value / N
```

```
# Расчитаем метрику DCG с помощью реализованных функций
dup_ranks = [2]

# Вычисляем DCG@1
dcg_value = dcg_score(dup_ranks, k=1)
print(f"DCG@1 = {dcg_value:.3f}")

# Вычисляем DCG@4
dcg_value = dcg_score(dup_ranks, k=4)
print(f"DCG@4 = {dcg_value:.3f}")
```

```
DCG@1 = 0.000
DCG@4 = 0.631
```

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

```
import pandas as pd
```

```
copy_answers = ["How does the catch keyword determine the type of exception tha

# наши кандидаты
candidates_ranking = [["How Can I Make These Links Rotate in PHP",
                       "How does the catch keyword determine the type of except
                       "NSLog array description not memory address",
                       "PECL_HTTP not recognised php ubuntu"],]

# dup_ranks — позиции наших копий, так как эксперимент один, то этот массив дли
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)
```

```
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:

```
import numpy as np
```

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

	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>, ...**

```
# !unzip stackoverflow_similar_questions.zip
```

```
unzip:  cannot find or open stackoverflow_similar_questions.zip, stackoverflow_s
```

Считайте данные.

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

Нам понадобиться только файл validation.

```
validation_data = read_corpus("D:\\Data\\StackOverflow\\validation.tsv")
```

Кол-во строк

```
len(validation_data)
```

```
3760
```

Размер нескольких первых строк

```
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)]**.

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

```
import string
punctuation_set = set(string.punctuation)

def rank_candidates(question, candidates, embeddings, tokenizer, dim=200, clear
    """
        question: строка
        candidates: массив строк(кандидатов) [a, b, c]
        embeddings: предварительно вычисленные эмбеддинги для кандидатов
        tokenizer: токенизатор для обработки текста
        dim: размерность эмбеддингов
        result: пары (начальная позиция, кандидат) [(2, c), (0, a), (1, b)]
    """
    # Получаем эмбеддинг для вопроса
    question_embedding = get_embedding(question, embeddings, tokenizer, dim)

    # Вычисляем косинусное сходство между вопросом и каждым кандидатом
    similarities = []
    for i, candidate in enumerate(candidates):
        candidate_embedding = get_embedding(candidate, embeddings, tokenizer, c
        similarity = cosine_similarity([question_embedding], [candidate_embeddi
        similarities.append((i, similarity))

    # Сортируем по убыванию косинусного сходства
    similarities.sort(key=lambda x: x[1], reverse=True)

    # Возвращаем список пар (начальная позиция, кандидат)
    result = [(idx, candidates[idx]) for idx, _ in similarities]
    return result
```

```
def get_embedding(text, embeddings, tokenizer, dim=200, clear_punctuation=False
    """
    Получает эмбеддинг для текста из предварительно вычисленного словаря
    или вычисляет на лету
    """
    text = text.lower()
    if text in embeddings:
        return embeddings[text]
    else:
        # Если эмбеддинга нет, вычисляем его
        tokens = tokenizer(text)
        if clear_punctuation:
            # Удаляем пунктуацию и не-буквенные токены
            tokens = [
                token for token in tokens
                if token not in punctuation_set and token.isalpha()
            ]
        token_embeddings = []
        for token in tokens:
            if token in embeddings:
                token_embeddings.append(embeddings[token])
            else:
                token_embeddings.append(np.zeros(dim))

        if token_embeddings:
            return np.mean(token_embeddings, axis=0)
        else:
            return np.zeros(dim)
```

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

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

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

```
from nltk.tokenize import word_tokenize
import pprint
for question, q_candidates in zip(questions, candidates):
        ranks = rank_candidates(question, q_candidates, wv_embeddings, word_tok
        pprint.pprint(ranks)
        print()
```

```
[(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?')]
```

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

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

```
# 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'), #скрыт
            (*, 'select2 not displaying search results'), #скрыт
            (*, 'WPF- How to update the changes in list item of a list')]] #скр
```

Вы должны получить `для эксперимента 1` следующую последовательность: `1, 0, 2`.

Вопрос 5:

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

Ваш ответ: 012

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.

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

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

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

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

```
  0%|          | 0/6 [00:00<?, ?it/s]
DCG@   1: 0.399 | Hits@   1: 0.399
DCG@   5: 0.487 | Hits@   5: 0.566
DCG@  10: 0.508 | Hits@  10: 0.633
DCG@ 100: 0.554 | Hits@ 100: 0.858
DCG@ 500: 0.569 | Hits@ 500: 0.969
DCG@1000: 0.573 | Hits@1000: 1.000
```

Из формул выше можно понять, что

- $\mathrm{Hits@K}$ **монотонно неубывающая функция** K, которая стремится к 1 при $K \to \infty$.
- $\mathrm{DCG@K}$ **монотонно неубывающая функция** K, но рост замедляется с увеличением K из-за убывания веса $\frac{1}{\log_2(1+\mathrm{rank}_{q_i'})}$.

Эмбеддинги, обученные на корпусе похожих вопросов

```
train_data = read_corpus("D:\\Data\StackOverflow\\train.tsv")
```

```
train_data[0:100]
```

```
[['converting string to list',
  'Convert Google results object (pure js) to Python object'],
 ['Which HTML 5 Canvas Javascript to use for making an interactive drawing
tool?',
  'Event handling for geometries in Three.js?'],
 ['Sending array via Ajax fails',
  'Getting all list items of an unordered list in PHP'],
 ['How to insert CookieCollection to CookieContainer?',
  'C# create cookie from string and send it'],
 ['Updating one element of a bound Observable collection',
  'WPF- How to update the changes in list item of a list'],
 ['MongoDB error on find()',
  'Retrieve only the queried element in an object array in MongoDB
collection'],
 ['select2 not displaying search results',
  'How to use jQuery AJAX for an outside domain?'],
 ['Using Reduce to merge multiple data frames with passing arguments and
without defining function outside the Reduce (syntax)',
  'R - merge a list of data frames into one data frame with missing values by
row'],
 ['Adding Prototype to JavaScript Object Literal',
  'How does JavaScript .prototype work?',
  'Javascript not setting property of undefined in prototyped object'],
 ["What's the best way to get the directory from which an assembly is
executing",
  'What is the dependency inversion principle and why is it important?',
  'Dependency Inversion with compile time configured Dependency Injection in
an ASP.NET MVC 4 Solution'],
 ['min value sql query', 'row with minimum value of a column'],
```

```
 ['Media Queries: How to target desktop, tablet and mobile?',
  'Media Queries to load resource?'],
 ['opaque element in a transparent in WPF',
  'WPF transparency changing, when elements are bound',
  'How do you override the opacity of a parent control in WPF?'],
 ['SmartThreadPool Blocking UI Thread',
  "What's the difference between Invoke() and BeginInvoke()"],
 ['Adding entity classes dynamically at runtime',
  'Create EntityManagerFactory programatically (without persistence.xml file)
with annotated classes'],
 ['Is it possible to use Font Awesome icon instead to select default drop down
sign',
  'How would I place a font-awesome icon into a select drop-down menu?'],
 ['ios 8 - buttons in horizontal scroll view intercepting pan event - scroll
does not work',
  'Swift UIScrollView not working with buttons'],
 ['an expression of type Null is ineligible for implicit conversion',
  'Scala Option type in function parameter - Best Practices'],
 ['Is it safe to display a image using $_GET for path?',
  'CSRF (Cross-site request forgery) attack example and prevention in PHP'],
 ['How to change scroll bar position with CSS?',
  'Scroll Bar at Top and Bottom in DataTables',
  'horizontal scrollbar on top and bottom of table'],
 ['Is it possible to use `SqlDbType.Structured` to pass Table-Valued
Parameters in NHibernate?',
  'While calling SQL Server stored procedure from NHibernate can I pass list
or custom data type as a parameter'],
 ['private/public qt signals',
  'Why do people use __(double underscore) so much in C++',
```

```
def count_single_queries(data):
    single_count = 0
    paired_count = 0

    for row in data:
        if len(row) == 1:  # 1 вопрос
            single_count += 1
        else:
            paired_count += 1

    total = len(data)

    print(f"Всего записей: {total}")
    print(f"Парных записей (с кандидатами): {paired_count} ({paired_count/total
    print(f"Одиночных запросов: {single_count} ({single_count/total*100:.1f}%)"

    return single_count, paired_count

single_count, paired_count = count_single_queries(train_data)
```

```
Всего записей: 1000000
Парных записей (с кандидатами): 1000000 (100.0%)
Одиночных запросов: 0 (0.0%)
```

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

> **вопрос>**

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

Рассмотрим подробнее данное склеивание.

1. Каждая строка из train_data разбивается на вопрос (question) и список кандидатов.
2. Для каждого кандидата вопрос склеивается с ним в одну строку.
3. Склеенная строка (combined_text) токенизируется, и полученный список токенов добавляется в общий корпус (corpus).

Пример

```
Вопрос: "What is Python?"
Кандидаты: ["Python is a programming language", "Java is another language"]
Склеенные строки:
    "What is Python? Python is a programming language"
    "What is Python? Java is another language"

Токенизированные списки:
    ['what', 'is', 'python', 'python', 'is', 'a', 'programming', 'language']
    ['what', 'is', 'python', 'java', 'is', 'another', 'language']
```

> Склеивание вопросов в пары нужно для того, чтобы модель Word2Vec узнала, что слова из похожих вопросов часто встречаются вместе в одном контексте. Это позволяет модели разместить семантически близкие слова (и, как следствие, целые вопросы) ближе друг к другу в векторном пространстве.

```
from nltk.tokenize import word_tokenize
import string

def build_corpus_with_nltk_cleaning(train_data):
    """
    Построение корпуса с очисткой через NLTK
    """
    corpus = []
    punctuation_set = set(string.punctuation)

    for row in train_data:
        if len(row) < 2:
            continue
```

```
            question = row[0]
            candidates = row[1:]

            for candidate in candidates:
                if candidate.strip():
                    combined_text = question + " " + candidate

                    # Токенизация и очистка
                    tokens = word_tokenize(combined_text.lower())

                    # Удаляем пунктуацию и не-буквенные токены
                    clean_tokens = [
                        token for token in tokens
                        if token not in punctuation_set and token.isalpha()
                    ]

                    if clean_tokens:
                        corpus.append(clean_tokens)

    return corpus
```

```
corpus = build_corpus_with_nltk_cleaning(train_data)
```

```
pprint.pprint(corpus[0:3])
```

```
[['converting',
  'string',
  'to',
  'list',
  'convert',
  'google',
  'results',
  'object',
  'pure',
  'js',
  'to',
  'python',
  'object'],
 ['which',
  'html',
  'canvas',
  'javascript',
  'to',
  'use',
  'for',
  'making',
  'an',
  'interactive',
  'drawing',
  'tool',
  'event',
  'handling',
  'for',
  'geometries',
  'in'],
```

```
 ['sending',
  'array',
  'via',
  'ajax',
  'fails',
  'getting',
  'all',
  'list',
  'items',
  'of',
  'an',
  'unordered',
  'list',
  'in',
  'php']]
```

```
len(corpus)
```

```
1256467
```

```
from gensim.models import Word2Vec
embeddings_trained = Word2Vec(
    corpus,
    vector_size=200,
    min_count=10,
    window=7,
    workers=8,
    sg=0,
    epochs=5
).wv
```

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

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

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

```
  0%|          | 0/6 [00:00<?, ?it/s]
DCG@   1: 0.295 | Hits@   1: 0.295
DCG@   5: 0.371 | Hits@   5: 0.440
DCG@  10: 0.393 | Hits@  10: 0.509
DCG@ 100: 0.446 | Hits@ 100: 0.768
DCG@ 500: 0.468 | Hits@ 500: 0.945
DCG@1000: 0.474 | Hits@1000: 1.000
```

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:

Ваш вывод:

- Какой принцип токенизации даёт качество лучше и почему?

  > Технические термины (camelCase, snake_case) несут смысловую нагрузку, поэтому сохранение целостности идентификаторов переменных и функций улучшает семантические связи

- Помогает ли нормализация слов?

  > Для технических текстов нормализация помогает слабо - в программировании регистр имеет значение. Также акронимы и сокращения теряют смысл.

- Какие эмбеддинги лучше справляются с задачей и почему?

  > Эмбеддинги должны быть построены с большим окном для захвата семантических связей. Также необходимо учесть что существуют редкие термины (по отношению ко всему корпусу), поэтому скипграммы позволяют эту особенность учесть

- Почему получилось плохое качество решения задачи?

  > Предполагаю что на это повлияло следующие факторы:
  > довольно простое преобразование предложения (документа) в вектор, маленькие области для конкретных языковых вопросов,

> слабый учет специфики доменной области: код, акронимы, переменные и тд.

- Предложите свой подход к решению задачи.

> Провести тематическое моделирование, например на основе облака тегов - встроенное в stackoverflow. Использовать методики, которые учитывают редкие термины и механизмы внимания.

```
Напишите программный код или сгенерируйте его с помощью искусственного интеллект
```