

## 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 ранжирования, поэтому введем математическую формулировку

## ✓ Задача ранжирования(Learning to Rank)

- $X$  - множество объектов
- $X^l = \{x_1, x_2, \dots, x_l\}$  - 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](https://zenodo.org/record/1199620/files/SO_vectors_200.bin?download=1)

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

```
➡ --2025-03-03 17:19:32-- https://zenodo.org/record/1199620/files/SO_vectors_
Resolving zenodo.org (zenodo.org)... 188.185.48.194, 188.185.43.25, 188.185.
Connecting to zenodo.org (zenodo.org)|188.185.48.194|:443... connected.
HTTP request sent, awaiting response... 301 MOVED PERMANENTLY
Location: /records/1199620/files/SO_vectors_200.bin [following]
--2025-03-03 17:19:33-- https://zenodo.org/records/1199620/files/SO_vectors_
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 13.9MB/s in 1m 47
2025-03-03 17:21:20 (12.9 MB/s) - 'SO_vectors_200.bin?download=1' saved [145
```

```
from gensim.models.keyedvectors import KeyedVectors
wv_embeddings = KeyedVectors.load_word2vec_format("SO_vectors_200.bin?download=1
```

## ✓ Как пользоваться этими векторами?

Посмотрим 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:

```
wv_embeddings.similar_by_word('dog', topn=5)
```

```
⇒ [('animal', 0.8564180135726929),  
   ('dogs', 0.7880866527557373),  
   ('mammal', 0.7623804211616516),  
   ('cats', 0.7621253728866577),  
   ('animals', 0.760793924331665)]
```

### ✓ **Вопрос 1:**

- Входит ли слово cat в топ-5 близких слов к слову dog? Какое место оно занимает?

```
# method most_similar  
word = 'cat'  
if word in wv_embeddings.similar_by_word('dog', topn=5):  
    print('Yes')  
else:  
    print('No')
```

```
⇒ No
```

**Ваш ответ:** 'cat' не входит в топ-5 ближайших слов к 'dog'.

### ✓ 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  
# 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()
```

```

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
    """

    '''your code'''

    words = tokenizer.tokenize(question)
    # words = tokenizer.tokenize(question.lower())

    word_vectors = []

    for word in words:
        if word in embeddings:
            word_vectors.append(embeddings[word])

    if word_vectors:
        return np.mean(word_vectors, 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)?

```

# Предложение
question = "I love neural networks"

question_to_vec(question, wv_embeddings, tokenizer)[2].round(2)

```

 -1.29

Ответ: -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 посмотреть на позицию дубликата. Мы хотим, чтобы дубликат был первым в ранжированном списке.

## Hits@K

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

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

- $[x < 0] \equiv \begin{cases} 1, & x < 0 \\ 0, & x \geq 0 \end{cases}$  - индикаторная функция
- $q_i$  -  $i$ -ый вопрос
- $q'_i$  - его дубликат
- $\text{rank}_{q'_i}$  - 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.

## DCG@K

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

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

С такой метрикой модель штрафует 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**.



## ✓ 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 с++?"
2. "Что такое язык python?"
3. "Хочу учить Java"
4. "Не понимаю Tensorflow"

$$\Rightarrow \text{rank}_{q'_i} = 2$$

Вычислим метрику  $\text{Hits}@K$  для  $K = 1, 4$ :

- $[K = 1] \text{Hits}@1 = [\text{rank}_{q'_i} \leq 1]$

Проверяем условие  $\text{rank}_{q'_i} \leq 1$ : **условие неверно.**

Следовательно,  $[\text{rank}_{q'_i} \leq 1] = 0$ .

- $[K = 4] \text{Hits}@4 = [\text{rank}_{q'_i} \leq 4] = 1$

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

Вычислим метрику  $\text{DCG}@K$  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)

$$DCG@10 = \frac{1}{\log_2(1+9)} \cdot [9 \leq 10] = \frac{1}{\log_2 10}$$

```
import math
round(1/math.log2(10),1)
```

⇒ 0.3

Ответ: 0.3

### ✓ 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 ( $rank_{q'_i}$ ):
  - $rank_{q'_1} = 2$ ,
  - $rank_{q'_2} = 5$ ,
  - $rank_{q'_3} = 1$ .

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

#### Для $K = 1$ :

Подставим значения:  $\text{Hits@1} = \frac{1}{3} \cdot \left( [rank_{q'_1} \leq 1] + [rank_{q'_2} \leq 1] + [rank_{q'_3} \leq 1] \right)$ .

Проверяем условие  $rank_{q'_i} \leq 1$  для каждого вопроса:

- $rank_{q'_1} = 2 \rightarrow 2 \not\leq 1 \rightarrow 0$ ,
- $rank_{q'_2} = 5 \rightarrow 5 \not\leq 1 \rightarrow 0$ ,
- $rank_{q'_3} = 1 \rightarrow 1 \leq 1 \rightarrow 1$ .

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



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


---

Для  $K = 5$ :

Подставим значения:  $\text{Hits}@5 = \frac{1}{3} \cdot (\text{rank}_{q'_1} \leq 5 + \text{rank}_{q'_2} \leq 5 + \text{rank}_{q'_3} \leq 5)$ .  $\square$

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

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

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:  $\text{DCG}@1 = \frac{1}{3} \cdot \left( \frac{1}{\log_2(1 + \text{rank}_{q'_1})} \cdot [\text{rank}_{q'_1} \leq 1] + \frac{1}{\log_2(1 + \text{rank}_{q'_2})} \cdot [\text{rank}_{q'_2} \leq 1] + \frac{1}{\log_2(1 + \text{rank}_{q'_3})} \cdot [\text{rank}_{q'_3} \leq 1] \right)$ .  $\text{DCG}@1 = \frac{1}{3} \cdot \left( \frac{1}{\log_2(1 + \text{rank}_{q'_1})} \cdot [\text{rank}_{q'_1} \leq 1] + \frac{1}{\log_2(1 + \text{rank}_{q'_2})} \cdot [\text{rank}_{q'_2} \leq 1] + \frac{1}{\log_2(1 + \text{rank}_{q'_3})} \cdot [\text{rank}_{q'_3} \leq 1] \right)$ .  $\square$

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

- $\text{rank}_{q'_1} = 2 \rightarrow 2 \not\leq 1 \rightarrow 0$ ,
- $\text{rank}_{q'_2} = 5 \rightarrow 5 \not\leq 1 \rightarrow 0$ ,

- $\text{rank}_{q_3} = 1 \implies \text{rank}_{q_3} = 1 \implies 1 \leq 1 \implies 1 \leq 1 \implies 1 \leq 1$ .

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

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

**Для  $K = 5$ :** Подставим значения:  $\text{DCG}@5 = \frac{1}{3} \cdot \left( \frac{1}{\log_2(1 + \text{rank}_{q_1})} \cdot [\text{rank}_{q_1} \leq 5] + \frac{1}{\log_2(1 + \text{rank}_{q_2})} \cdot [\text{rank}_{q_2} \leq 5] + \frac{1}{\log_2(1 + \text{rank}_{q_3})} \cdot [\text{rank}_{q_3} \leq 5] \right)$ .  
 $\text{DCG}@5 = \frac{1}{3} \cdot \left( \frac{1}{\log_2(1 + \text{rank}_{q_1})} \cdot [\text{rank}_{q_1} \leq 5] + \frac{1}{\log_2(1 + \text{rank}_{q_2})} \cdot [\text{rank}_{q_2} \leq 5] + \frac{1}{\log_2(1 + \text{rank}_{q_3})} \cdot [\text{rank}_{q_3} \leq 5] \right)$ .

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

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

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

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

#### ✓ **Вопрос 4:**

- Найдите максимум Hits@47 - DCG@1 ?

Ответ: 1

Причем это верно при выполнении такого условия:  $2 \leq \text{rank}_{q_i} \leq 47$ .

## ✓ 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]`.

```
import numpy as np
```

```
def hits_count(dup_ranks, k):
    """
        dup_ranks: list индексов дубликатов
        k: пороговое значение для ранга
        result: вернуть Hits@k
    """
    # Подсчитываем количество дубликатов, чей ранг <= k
    hits_value = 0
    for rank in dup_ranks:
        if rank <= k:
            hits_value += 1

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

```
dup_ranks = [2]
```

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

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

```
⇒ Hits@1 = 0.0
   Hits@4 = 1.0
```

```
def dcg_score(dup_ranks, k):
    """
        dup_ranks: list индексов дубликатов
        k: пороговое значение для ранга
        result: вернуть DCG@k
    """
```

```
# Вычисляем сумму для всех релевантных дубликатов
dcg_value = 0
for rank in dup_ranks:
    if rank <= k:
        dcg_value += 1/np.log2(1+rank)

# Делим на общее количество вопросов
dcg_value /= len(dup_ranks)
return dcg_value
```

```
# Пример списка позиций дубликатов
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@10 = {dcg_value:.3f}")
```

```
➡ DCG@1 = 0.000
   DCG@10 = 0.631
```

Протестируем функции. Пусть  $N = 1$ , то есть 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 that
```

```
# наши кандидаты
candidates_ranking = [
    ["How Can I Make These Links Rotate in PHP",
     "How does the catch keyword determine the type of exception",
     "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
dup_ranks = [cands.index(i) + 1 if i in cands else None for i, cands in zip(copy,
```

```
# 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

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



	1	2	3	4
<b>HITS</b>	0	1.00000	1.00000	1.00000
<b>DCG</b>	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
```



```
Archive:  stackoverflow_similar_questions.zip
  creating: data/
  inflating: data/.DS_Store
   creating: __MACOSX/
   creating: __MACOSX/data/
  inflating: __MACOSX/data/._.DS_Store
  inflating: data/train.tsv
  inflating: data/validation.tsv
```

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 понадобится только файл validation.

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

Кол-во строк

```
len(validation_data)
```

```
↩ 3760
```

Размер нескольких первых строк

```
for i in range(25):  
    print(i + 1, len(validation_data[i]))
```

```
↩ 1 1001  
2 1001  
3 1001  
4 1001  
5 1001  
6 1001  
7 1001  
8 1001  
9 1001  
10 1001  
11 1001  
12 1001  
13 1001  
14 1001  
15 1001  
16 1001  
17 1001  
18 1001  
19 1001  
20 1001  
21 1001  
22 1001  
23 1001  
24 1001  
25 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

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)]
    """
    '''your code'''

    question_vec = question_to_vec(question, embeddings, tokenizer, dim).reshape(1, -1)
    candidate_vecs = []
    for candidate in candidates:
        candidate_vec = question_to_vec(candidate, embeddings, tokenizer, dim).reshape(1, -1)
        candidate_vecs.append(candidate_vec)

    similarities = []
    for i, candidate_vec in enumerate(candidate_vecs):
        similarity = cosine_similarity(question_vec, candidate_vec)[0][0]
        similarities.append((i, similarity))

    similarities.sort(key=lambda x: -x[1])
    result = [(idx, candidates[idx]) for idx, _ in similarities]

    return result
```

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', # первый
               '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
               'WPF- How to update the changes in list item of a list',
               'select2 not displaying search results']]
```

Это результат, если не приводить к нижнему регистру.

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

⇒ [(1, 'C# create cookie from string and send it'), (0, 'Convert Google result  
[(1, 'WPF- How to update the changes in list item of a list'), (0, 'Getting

Это результат, если приводить к нижнему регистру.

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

⇒ [(1, 'C# create cookie from string and send it'), (0, 'Convert Google result  
[(0, 'Getting all list items of an unordered list in PHP'), (2, 'select2 not

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')]] #скрыт
```

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: 102 ОДНАКО, ЕСЛИ в функции `def question_to_vec(question, embeddings, tokenizer, dim=200)`

ВМЕСТО `words = tokenizer.tokenize(question)` НАПИСАТЬ `words = tokenizer.tokenize(question.lower())`, ТО ЕСТЬ привести к нижнему регистру, то ОТВЕТ: 021

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



27%

1000/3760 [09:39<26:02, 1.77it/s]

```
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)
```



27%

1000/3760 [09:27<26:01, 1.77it/s]

Это результат, если приводить к нижнему регистру.

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



100%

6/6 [00:00<00:00, 150.89it/s]

```
DCG@ 1: 0.415 | Hits@ 1: 0.415  
DCG@ 5: 0.502 | Hits@ 5: 0.582  
DCG@ 10: 0.525 | Hits@ 10: 0.651  
DCG@ 100: 0.570 | Hits@ 100: 0.874  
DCG@ 500: 0.583 | Hits@ 500: 0.973  
DCG@1000: 0.586 | Hits@1000: 1.000
```

А это результат, если не приводить к нижнему регистру.

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



100%

6/6 [00:00<00:00, 237.12it/s]

```
DCG@ 1: 0.285 | Hits@ 1: 0.285  
DCG@ 5: 0.342 | Hits@ 5: 0.393  
DCG@ 10: 0.360 | Hits@ 10: 0.449  
DCG@ 100: 0.406 | Hits@ 100: 0.679  
DCG@ 500: 0.431 | Hits@ 500: 0.879  
DCG@1000: 0.444 | Hits@1000: 1.000
```

Из формул выше можно понять, что

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

✓ Эмбединги, обученные на корпусе похожих вопросов

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

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']

train\_data[111258]



['Determine if the device is a smartphone or tablet?',  
'Change imageView params in all cards together']

```
# Создаем общий корпус текстов
```

```
corpus = []
```

```
for pair in train_data:
```

```
    question = pair[0]
```

```
    candidate = pair[1]
```

```
    combined_text = f"{question} {candidate}"
```

```
    tokens = tokenizer.tokenize(combined_text)
```

```
    corpus.append(tokens)
```

```
print(corpus[:2])
```

```
➡ [['converting', 'string', 'to', 'list', 'Convert', 'Google', 'results', 'obj
```

```
from gensim.models import Word2Vec
```

```
embeddings_trained = Word2Vec(
```

```
    sentences=corpus,          # Корпус токенизированных текстов
```

```
    vector_size=200,          # Размерность векторов
```

```
    window= 5,                # Размер окна контекста
```

```
    min_count=2,              # Минимальная частота слов
```

```
    workers=4                 # Количество потоков
```

```
).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, tokenizer)
```

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



27%

1000/3760 [09:47<23:46, 1.93it/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, hi
```



100%

6/6 [00:00&lt;00:00, 267.24it/s]

DCG@ 1:	0.259		Hits@ 1:	0.259
DCG@ 5:	0.326		Hits@ 5:	0.386
DCG@ 10:	0.351		Hits@ 10:	0.460
DCG@ 100:	0.404		Hits@ 100:	0.721
DCG@ 500:	0.429		Hits@ 500:	0.916

### Замечание:

Решить эту задачу с помощью предобученной 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 эмбединги лучше справляются с задачей и почему?
- Почему получилось плохое качество решения задачи?
- Предложите свой подход к решению задачи.

### ✓ Вывод:

1) Лучший принцип токенизации: токенизация с использованием библиотек, например nltk, которая поддерживают удаление стоп-слов, лемматизацию и обработку пунктуации. Потому что удаление стоп-слов уменьшает шум в данных, а лемматизация приводит слова к их базовой форме, что помогает модели лучше обобщать.

2) Да, нормализация слов (приведение к нижнему регистру, лемматизация, стемминг) помогает. Мы это увидели на конкретном примере: при приведении слов к нижнему регистру качество решения задачи повышается.

3) Предобученные. Так как они обучены на огромных корпусах текстов.

4) Качество решения задачи не самое лучшее из-за самого метода представления предложения. А качество на предобученных эмбедингах лучше видимо также из-за разницы в объеме тренировочных данных.

5) Шаги:

-Предобработка текста: удаление стоп-слов, лемматизация или стемминг, обработка пунктуации и специальных символов.