



Deep Learning School

## 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 ноутки 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)

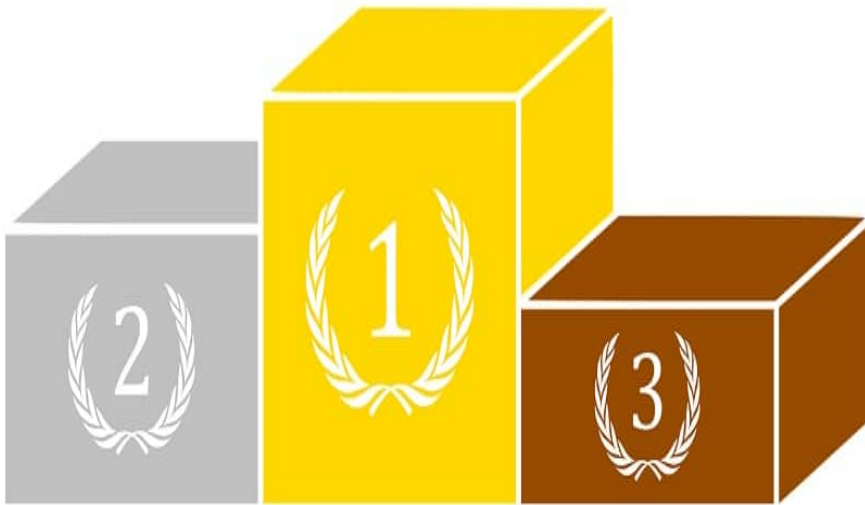
- $X$  - множество объектов
- $X^l = \{x_1, x_2, \dots, x_l\}$  - обучающая выборка  
На обучающей выборке задан порядок между некоторыми элементами, то есть нам известно, что некий объект выборки более релевантный для нас, чем другой:
- $i \prec j$  - порядок пары индексов объектов на выборке  $X^l$  с индексами  $i$  и  $j$

Задача:

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

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

# Ranking



## 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
```

```
--2021-10-11 13:54:05-- https://zenodo.org/record/1199620/files/SO_vectors_200.bin?download=1
Resolving zenodo.org (zenodo.org)... 137.138.76.77
Connecting to zenodo.org (zenodo.org)|137.138.76.77|:443... connected.
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 16.0MB/s in 59s
```

```
2021-10-11 13:55:05 (23.5 MB/s) - 'SO_vectors_200.bin?download=1' saved [1453905423/1453905423]
```

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

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,)
```

```
wv_embeddings["dog"]
```

```
array([ 0.6851772, -1.2778991, -0.41913974,  1.3623164, -3.1675398,
        0.09950767,  0.6402681, -1.1245339, -0.6699619, -0.6998852,
        0.4936771, -0.40500194, -3.0706816, -2.2809966,  0.85798043,
        2.7093108,  0.3492745, -0.03494101, -0.22330493,  1.2290467,
        1.7755157, -3.158358, -0.6515983,  0.7224096,  2.3193083,
       -1.7969862,  0.40903398, -2.744604, -1.7179952, -0.914309,
       -0.75887376, -0.35140672, -0.5182776, -1.9097351, -0.8300773,
        0.02147918,  1.1783471,  0.03169126, -0.3069023,  1.6666299,
        0.6711357, -2.1706133, -0.11800487,  0.22336982, -1.2075394,
       -0.86297905, -0.63865614,  1.1733794,  0.10022762,  0.7017279,
        2.7290728, -0.4640484, -2.1719306, -0.3562852, -1.8449957,
        0.10270727,  1.1125596, -0.8364318,  1.9513408, -0.97937447,
        1.2650859,  0.06809282,  0.6477318, -0.52431005, -0.6103959,
       -2.979829, -0.7889965, -0.11004248,  1.7603841, -1.0547444,
       -0.98998, -1.1834062, -2.5359967, -0.35286787,  0.7733574,
        0.16300043,  0.76991326,  1.9223119, -1.2843752, -1.5023688,
        0.15765315, -2.4150877,  2.4326491, -0.36940518,  2.2511673,
       -1.602164,  1.0818797,  0.59202737, -2.8141215, -0.9547914,
       -1.8816328, -2.251527,  0.208779,  0.24186568,  0.36262512,
       -1.1487632, -0.33167967,  1.4389734,  0.9608894,  0.35947856,
       -0.45259392, -0.45514247, -2.1248987,  1.828458, -1.8585896,
        0.4114255,  1.1428199, -0.50754434, -0.12364098, -3.7919624,
       -1.3027766, -1.0333146,  0.97529024, -0.9289765, -0.9175716,
        0.04012083, -1.7776312, -0.16942771,  2.318477,  0.16809507,
        1.5688989, -1.9980967, -1.0449845,  1.3963503, -0.13359734,
       -1.4952992, -0.60059625, -0.04604611, -1.8134118, -0.50812286,
       -1.7697111, -2.6240456,  0.50520307, -2.0226426, -0.8193453,
       -3.036361,  0.32545397, -0.5707175,  0.93880373,  1.5678531,
       -0.4485128,  1.4617805,  2.6263788,  1.0872725,  0.21335754,
        0.93010294,  3.489708,  0.17392842,  0.911422, -0.61643803,
       -2.1106405,  0.36008754, -0.32839164, -0.2838782, -0.23150575,
        0.73796517, -1.3399363, -2.356109, -2.0610485, -1.2165475,
        0.7498155, -0.19086224, -2.8202996, -0.37605208,  0.1785639,
        0.72514814, -0.8697008,  0.41977307,  0.04011298, -2.8652422,
       -0.18194903, -0.5910473, -2.7400806, -0.6971266,  1.2614748,
       -1.2979602, -3.0204656, -0.743788,  1.6838108,  0.11611916,
        0.84407914, -3.4510055,  0.61605287, -1.2632201, -0.00716789,
       -1.2434675, -0.47464737, -0.7219753, -0.51104206, -3.793404,
       -1.0512625, -1.2473925, -0.67683965, -0.17455211,  2.6480556,
       -2.1741571, -2.2171447,  0.71225303,  4.1232347, -1.4569836 ],
      dtype=float32)
```

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

```
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?

```
# method most_similar
'''your code'''
wv_embeddings.most_similar(positive="dog", topn=10)
```

```
[('animal', 0.8564180135726929),
 ('dogs', 0.7880867123603821),
 ('mammal', 0.7623804807662964),
 ('cats', 0.7621253728866577),
```

```
( 'animals', 0.760793924331665),
( 'feline', 0.7392398118972778),
( 'bird', 0.7315489053726196),
( 'animal1', 0.7219215631484985),
( 'doggy', 0.7213349938392639),
( 'labrador', 0.7209131717681885)]
```

# "cat" не входит в top-5 близких слов, но входит "cats"

```
wv_embeddings.most_similar(positive="dog", topn=30)
```

```
[('animal', 0.8564180135726929),
 ('dogs', 0.7880867123603821),
 ('mammal', 0.7623804807662964),
 ('cats', 0.7621253728866577),
 ('animals', 0.760793924331665),
 ('feline', 0.7392398118972778),
 ('bird', 0.7315489053726196),
 ('animal1', 0.7219215631484985),
 ('doggy', 0.7213349938392639),
 ('labrador', 0.7209131717681885),
 ('canine', 0.7209056615829468),
 ('meow', 0.7185295820236206),
 ('cow', 0.7080444693565369),
 ('dog2', 0.7057910561561584),
 ('woof', 0.7050611972808838),
 ('dog1', 0.7038840055465698),
 ('dog3', 0.701882004737854),
 ('penguin', 0.6970292329788208),
 ('bulldog', 0.6940488815307617),
 ('mammals', 0.6931389570236206),
 ('bark', 0.6913799047470093),
 ('fruit', 0.6892251968383789),
 ('reptile', 0.6891210079193115),
 ('furry', 0.6863498687744141),
 ('carnivore', 0.6862949728965759),
 ('cat', 0.6852341294288635),
 ('horse', 0.6833381056785583),
 ('kitten', 0.6820152997970581),
 ('sheep', 0.6802570223808289),
 ('chihuahua', 0.6791757941246033)]
```

# cat занимает 26-ое место

## 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)
```

```
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'''
    question_embedding = np.zeros(shape=dim)
    num_question_words = 0
    for word in tokenizer.tokenize(text=question.lower()):
        if word in embeddings:
            question_embedding += embeddings[word]
            num_question_words += 1
    if num_question_words:
        return question_embedding / num_question_words
    return question_embedding
```

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)?

```
example_question = "I love neural networks"
```

```
tokenizer = MyTokenizer()
wv_embeddings
```

```
<gensim.models.keyedvectors.Word2VecKeyedVectors at 0x7f6f51a45fd0>
```

```
question_embedding = question_to_vec(example_question, wv_embeddings, tokenizer, dim=200)
question_embedding.shape

(200,)
```

```
round(question_embedding[2], 2)

-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  $K$ :

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

- $[x < 0] \equiv \begin{cases} 1, & x < 0 \\ 0, & x \geq 0 \end{cases}$  - индикаторная функция
- $q_i$  -  $i$ -ый вопрос
- $q'_i$  - его дубликат
- $\text{rank}_{q'_i}$  - позиция дубликата в ранжированном списке ближайших предложений для вопроса  $q_i$ .

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 штрафуются за большой ранг корректного ответа

Вопрос 3:

- Максимум Hits@47 - DCG@1?

# Рассуждения по данному вопросу:

# Чтобы максимизировать разницу, первое слагаемое должно стремиться к максимально возможному своему зна

#  $\max(\text{Hits@47}) = 1$ , если для каждого запроса ранг его дубликата  $\leq 47$ , что вполне реализуемо чисто теор

#  $\min(\text{DCG@1}) = 0$ , если для каждого запроса ранг его дубликата  $> 1$ , что также вполне реализуемо теоретич

# Тогда, при условии ранжирования так, что выполняются условия выше  $\rightarrow \max(\text{Hits@47} - \text{DCG@1}) = 1$



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  $q'_1$

Пусть модель выдала следующий ранжированный список кандидатов:

1. "Как изучить с++?"
2. "Что такое язык python?"

3. "Хочу учить Java"

4. "Не понимаю Tensorflow"

$$\Rightarrow rank_{q'_i} = 2$$

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

- $[K = 1] Hits@1 = [rank_{q'_i} \leq 1] = 0$
- $[K = 4] Hits@4 = [rank_{q'_i} \leq 4] = 1$

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

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

Вопрос 4:

- Вычислите  $DCG@10$ , если  $rank_{q'_i} = 9$  (округлите до одного знака после запятой)

```
import math

# 1/log2(1 + 9) * [9 <= 10]

round(1 / math.log(1 + 9, 2), 1)

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.

```
def hits_count(dup_ranks, k):
    """
        dup_ranks: list индексов дубликатов
        result: вернуть Hits@k
    """
    '''your code'''
    hits_value = 0
    for dup_rank in dup_ranks:
        if dup_rank <= k:
            hits_value += 1
    return hits_value / len(dup_ranks)
```

```
import math
```

```
def dcg_score(dup_ranks, k):
    """
        dup_ranks: list индексов дубликатов
        result: вернуть DCG@k
    """
    '''your code'''
    dcg_value = 0
    for dup_rank in dup_ranks:
```

```

    if dup_rank <= k:
        dcg_value += 1 / math.log(1 + dup_rank, 2)
    return dcg_value / len(dup_ranks)

```

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 was thrown"]
```

```

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

```

```
# вспомогательная ф-я для нахождения ранга дубликата
```

```

def find_dup_ranks(candidates, copy_answer):
    for i, candidate in enumerate(candidates, 1):
        if candidate == copy_answer:
            return i

```

```
# dup_ranks – позиции наших копий, так как эксперимент один, то этот массив длины 1
```

```
dup_ranks = [find_dup_ranks(sentence, copy_answers[i]) for i, sentence in enumerate(candidates_ranking)]
```

```
# 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>, ...**

```
from google.colab import drive
drive.mount('/content/drive')
```

Mounted at /content/drive

```
!unzip /content/drive/MyDrive/stackoverflow_similar_questions.zip
```

```
Archive: /content/drive/MyDrive/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
```

```
ls -la
```

```
total 1419864
drwxr-xr-x 1 root root      4096 Oct 11 14:04 ./
drwxr-xr-x 1 root root      4096 Oct 11 13:52 ../
drwxr-xr-x 4 root root      4096 Sep 30 17:11 .config/
drwxr-xr-x 2 root root      4096 Oct 26  2018 data/
drwx----- 6 root root      4096 Oct 11 14:04 drive/
drwxrwxr-x 3 root root      4096 Sep 22  2019 __MACOSX/
drwxr-xr-x 1 root root      4096 Sep 30 17:12 sample_data/
-rw-r--r-- 1 root root 1453905423 Oct  3 01:07 'SO_vectors_200.bin?download=1'
```

Считайте данные.

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

    return data
```

Нам понадобится только файл validation.

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

```
validation_data[1][-10:] # .strip() позволил избавиться от /n
```

```
['setGraphic() not working correctly on recursively created TreeItems',
 'Strange behaviour of sscanf with string',
 'HTTP method for a WCF Restful Service',
 'SubSonic .Filter() in memory filter',
 'Does ASP.NET MVC use the regular toolbox controls?',
 'How can I render a GSP as a String?',
 'publishing ASP.Net MVC 4 Project',
 'how to hyperlink to a TFS item from visual studio',
 'mutex attribute PTHREAD_PROCESS_SHARED inverts logic?',
 'Unable to call SOAP Webservice using jQuery']
```

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)].

```
import pandas as pd

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_embedding = [question_to_vec(question, embeddings, tokenizer, dim)]
    candidates_embeddings = [question_to_vec(candidate, embeddings, tokenizer, dim) for candidate in ca

    cos_similarity = pd.Series(cosine_similarity(X=question_embedding, Y=candidates_embeddings).squeeze
    #print(cos_similarity)

    return [(i, candidates[i]) for i in cos_similarity.index]
```

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 эксперимент
               '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 эксперимент
               'PHP: How to update the changes in list item of a list']]
```

```
wpf- how to update the changes in list item of a list ,
'select2 not displaying search results']]
```

```
print(tokenizer)
print(wv_embeddings)
```

```
<__main__.MyTokenizer object at 0x7f6f2db96250>
<gensim.models.keyedvectors.Word2VecKeyedVectors object at 0x7f6f51a45fd0>
```

```
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 results object (pure js) to
[(1, 'WPF- How to update the changes in list item of a list'), (0, 'Getting all list items of an u
```

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
```

```
File "<ipython-input-149-858aee104232>", line 5
    [(*, 'Getting all list items of an unordered list in PHP'), #скрыт
     ^
```

SyntaxError: invalid syntax

SEARCH STACK OVERFLOW

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?)

```
# 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.

```
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 [01:08<03:09, 14.57it/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, hits_count(wv_ranking, k)

100% 6/6 [00:00<00:00, 86.33it/s]

DCG@ 1: 0.415 | Hits@ 1: 0.415
DCG@ 5: 0.502 | Hits@ 5: 0.582
DCG@ 10: 0.524 | Hits@ 10: 0.650
DCG@ 100: 0.570 | Hits@ 100: 0.874
DCG@ 500: 0.583 | Hits@ 500: 0.973
DCG@1000: 0.586 | Hits@1000: 1.000

```

# ниже можно увидеть результат, если не использовать .lower()

```

100% 6/6 [00:00<00:00, 102.79it/s]

DCG@ 1: 0.228 | Hits@ 1: 0.228
DCG@ 5: 0.285 | Hits@ 5: 0.336
DCG@ 10: 0.303 | Hits@ 10: 0.392
DCG@ 100: 0.349 | Hits@ 100: 0.622
DCG@ 500: 0.374 | Hits@ 500: 0.821
DCG@1000: 0.393 | Hits@1000: 1.000

```

## Эмбединги, обученные на корпусе похожих вопросов

```

train_data = read_corpus('./data/train.tsv')
train_data[1]

['Which HTML 5 Canvas Javascript to use for making an interactive drawing tool?',
 'Event handling for geometries in Three.js?']

train_data[:5]

[['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']]

len(train_data)

1000000

```

Улучшите качество модели.

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

Объясните свой выбор.

```
tokenizer = MyTokenizer()
```

```
words = [list(tokenizer.tokenize(" ".join(corpus))) for corpus in train_data]
```

```
len(words)
```

```
1000000
```

```
words[0]
```

```
['converting',
 'string',
 'to',
 'list',
 'Convert',
 'Google',
 'results',
 'object',
 'pure',
 'js',
 'to',
 'Python',
 'object']
```

# Посмотрим на пример работы обученного нами Word2Vec

```
from gensim.models import Word2Vec
embeddings_trained = Word2Vec(words,                # data for model to train on
                               size=200,            # embedding vector size
                               min_count=5,          # consider words that occurred at least 5 times
                               window=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, tokenizer)
    wv_ranking.append([r[0] for r in ranks].index(0) + 1)
```

```
27%
```

```
1000/3760 [01:25<03:50, 12.00it/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, hits_count(wv_ranking, k))
```

100%

6/6 [00:00&lt;00:00, 73.76it/s]

# ниже также представлен результат, если не использовать .lower()

```
DCG@ 10: 0.357 | Hits@ 10: 0.465
```

100%

6/6 [00:00&lt;00:00, 97.20it/s]

```
DCG@ 1: 0.259 | Hits@ 1: 0.259
DCG@ 5: 0.327 | Hits@ 5: 0.390
DCG@ 10: 0.354 | Hits@ 10: 0.474
DCG@ 100: 0.404 | Hits@ 100: 0.724
DCG@ 500: 0.430 | Hits@ 500: 0.925
DCG@ 1000: 0.438 | Hits@ 1000: 1.000
```

# Попробуем подобрать более оптимальное значение размера окна

# Посмотрим на распределение кол-ва слов в документах

```
from collections import Counter
```

```
words_count = Counter({i: len(word_i) for i, word_i in enumerate(words)})
count = [val for val in words_count.values()]
```

```
import seaborn as sns
import matplotlib.pyplot as plt
```

```
%matplotlib inline
plt.rcParams["figure.figsize"] = (15, 10)
```

```
sns.histplot(data=count);
```

70000

```

# кол-во слов в предложении в основном около 10 - 30, поэтому размер окна может быть около 20 или даже
# зададим размер окон, которые рассмотрим
window_size = [5, 10, 15, 20, 30]

for window in window_size:

    embeddings_trained = Word2Vec(words,                # data for model to train on
                                  size=200,             # embedding vector size
                                  min_count=5, # consider words that occurred at least 5 times
                                  window=window).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)

    print("window_size: {}".format(window))
    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_ranking, k)))
    print()

```

```

27%                               1000/3760 [01:28<04:01, 11.43it/s]
window_size: 5
100%                               6/6 [00:00<00:00, 9.01it/s]
DCG@ 1: 0.265 | Hits@ 1: 0.265
DCG@ 5: 0.335 | Hits@ 5: 0.398
DCG@ 10: 0.357 | Hits@ 10: 0.467
DCG@ 100: 0.410 | Hits@ 100: 0.732
DCG@ 500: 0.434 | Hits@ 500: 0.919
DCG@1000: 0.443 | Hits@1000: 1.000

```

```

27%                               1000/3760 [01:29<04:00, 11.47it/s]
window_size: 10
100%                               6/6 [00:00<00:00, 72.46it/s]
DCG@ 1: 0.279 | Hits@ 1: 0.279
DCG@ 5: 0.355 | Hits@ 5: 0.422
DCG@ 10: 0.374 | Hits@ 10: 0.488

```

# как можно заметить, лучшим вариантом оказалось окно размером = 20

```
DCG@1000: 0.458 | 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".

```
DCG@ 10: 0.379 | Hits@ 10: 0.488
```

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:

Чтобы ответить на поставленные вопросы, проведём дополнительные действия.

А именно:

- 1) Воспользуемся nltk токенайзером
- 2) Применим лемматизация или стемминг + стоп слова

```
DCG@1000: 0.470 | Hits@1000: 1.000
```

# создаём токенизатор на основе nltk word\_tokenize

```

import nltk
nltk.download("punkt")
from nltk.tokenize import word_tokenize

[nltk_data] Downloading package punkt to /root/nltk_data...
[nltk_data] Unzipping tokenizers/punkt.zip.

class NLTK_Tokenizer():
    def __init__(self, word_tokenize=None, stopWords=False, lemmatizer=False):
        if word_tokenize is None:
            import nltk

```



```

word_tokenize = nltk.tokenize.word_tokenize
self.word_tokenize = word_tokenize

if stopWords:
    import nltk
    nltk.download('stopwords')

    from nltk.corpus import stopwords
    stopWords = set(stopwords.words('english'))
self.stopWords = stopWords

if lemmatizer:
    import nltk
    nltk.download('wordnet')

self.lemmatizer = nltk.WordNetLemmatizer()

def tokenize(self, text):
    if self.stopWords and self.lemmatizer:
        return [self.lemmatizer.lemmatize(word) for word in self.word_tokenize(text.lower()) if wor
    elif self.stopWords:
        return [word for word in self.word_tokenize(text.lower()) if word not in self.stopWords]
    elif self.lemmatizer:
        return [self.lemmatizer.lemmatize(word) for word in self.word_tokenize(text.lower())]
    return self.word_tokenize(text.lower())

nltk_tokenizer = NLTK_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, wv_embeddings, nltk_tokenizer)
    wv_ranking.append([r[0] for r in ranks].index(0) + 1)

27% 1000/3760 [03:32<09:29, 4.85it/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, hits_count(wv_ranking, k)

100% 6/6 [00:00<00:00, 9.68it/s]

DCG@ 1: 0.399 | Hits@ 1: 0.399
DCG@ 5: 0.487 | Hits@ 5: 0.566
DCG@ 10: 0.509 | Hits@ 10: 0.634
DCG@ 100: 0.554 | Hits@ 100: 0.858
DCG@ 500: 0.569 | Hits@ 500: 0.969
DCG@1000: 0.573 | Hits@1000: 1.000

# посмотрим на токенизатор + лемматизатор + стоп слова

nltk_tokenizer_v1 = NLTK_Tokenizer(stopWords=True)
nltk_tokenizer_v2 = NLTK_Tokenizer(lemmatizer=True, stopWords=True)

[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data] Package stopwords is already up-to-date!
[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data] Package stopwords is already up-to-date!

```

```
[nltk_data] Downloading package wordnet to /root/nltk_data...
[nltk_data] Unzipping corpora/wordnet.zip.
```

```
# добавляем стоп слова
```

```
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, nltk_tokenizer_v1)
    wv_ranking.append([r[0] for r in ranks].index(0) + 1)

27% 1000/3760 [04:16<11:29, 4.00it/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, hits_count(wv_ranking, k))
```

```
100% 6/6 [00:00<00:00, 108.43it/s]

DCG@ 1: 0.392 | Hits@ 1: 0.392
DCG@ 5: 0.482 | Hits@ 5: 0.565
DCG@ 10: 0.503 | Hits@ 10: 0.630
DCG@ 100: 0.549 | Hits@ 100: 0.856
DCG@ 500: 0.563 | Hits@ 500: 0.965
DCG@1000: 0.567 | Hits@1000: 1.000
```

```
# стоп-слова + лемматризатор
```

```
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, nltk_tokenizer_v2)
    wv_ranking.append([r[0] for r in ranks].index(0) + 1)

27% 1000/3760 [04:10<11:20, 4.05it/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, hits_count(wv_ranking, k))
```

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


DCG@ 1: 0.392 | Hits@ 1: 0.392
DCG@ 5: 0.482 | Hits@ 5: 0.565
DCG@ 10: 0.503 | Hits@ 10: 0.630
DCG@ 100: 0.549 | Hits@ 100: 0.856
DCG@ 500: 0.563 | Hits@ 500: 0.965
DCG@1000: 0.567 | Hits@1000: 1.000
```

```
# Как можно заметить, качество ранжирования на предобученных эмбедингах при условии нормализации и nlt
# слегка ухудшилось, что наводит на мысль, что словарь предобученных эмбедингов настолько широк, что в
# и в данном случае этого делать не надо
```

```
# С данной задачей ранжирования лучше справились предобученные эмбединги
# Возможно, структура каждого документа, полученная с помощью train.tsv, не является достаточной в том
# А пройдясь по 2м вопросам и подсчитав необходимые метрики, этого может оказаться недостаточным.... в
# этого недостаточно

# Поэтому, для обучения своих эмбедингов, каждый их документов должен содержать большее кол-во информа
# К примеру можно увеличить кол-во ближайших по смыслу вопросов с помощью и на них уже обучать Word2Vec
```

---

 12 мин. 4 сек. выполнено в 20:17

Не удастся связаться с сервисом reCAPTCHA. Проверьте подключение к Интернету и перезагрузите страницу.