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 ноутка напишите свой вывод. Работа без вывода оценивается ниже.

Задача поиска схожих по смыслу предложений

Мы будем ранжировать вопросы [StackOverflow]() на основе семантического векторного представления

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

Задача ранжирования**(Learning to Rank)**

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

$$\begin{aligned} & i \prec j \\ \Rightarrow\ & a(x_i) \\ <\ & a(x_j) \end{aligned}$$

Imports and NLTK downloads

In [1]:

```
try:
    print("Trying installing prerequisites with conda...")
    %conda install -c conda-forge gdown gensim halo numpy nltk pandas scikit-learn tqdm
wget --yes --quiet --satisfied-skip-solve
except ValueError:
    print("Installation via conda failed, trying pip...")
    %pip install gdown gensim halo numpy nltk pandas scikit-learn tqdm --quiet --exists-
action i
finally:
    print("Prerequisites installed successfully!")
```

```
Trying installing prerequisites with conda...

# All requested packages already installed.


Note: you may need to restart the kernel to use updated packages.
Prerequisites installed successfully!
```

In [142]:

```
import gc
import pathlib
import os
import random

import gdown
import gensim.models
import gensim.models.keyedvectors
import halo
import numpy as np
import pandas as pd
import nltk
nltk.download('perluniprops')
nltk.download('punkt')
nltk.download('wordnet')
import nltk.tokenize
import nltk.tokenize.nist
import nltk.tokenize.stanford
import nltk.stem
import sklearn.metrics.pairwise
import tqdm.notebook as tqdm
```

```
[nltk_data] Downloading package perluniprops to
[nltk_data]     /home/shaorrran/nltk_data...
[nltk_data]   Package perluniprops is already up-to-date!
[nltk_data] Downloading package punkt to /home/shaorrran/nltk_data...
[nltk_data]   Package punkt is already up-to-date!
```

```
[nltk_data] Downloading package wordnet to
[nltk_data]     /home/shaorrran/nltk_data...
```

Reproducibility settings

In [3]:

```
RANDOM_STATE = 42
random.seed(RANDOM_STATE)
os.environ["PYTHONHASHSEED"] = str(RANDOM_STATE)
np.random.seed(RANDOM_STATE)
```

Embeddings

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

In [4]:

```
if not pathlib.Path("data/SO_vectors_200.bin").is_file():
    !wget https://zenodo.org/record/1199620/files/SO_vectors_200.bin?download=1 -O data/SO_vectors_200.bin
```

In [5]:

```
with halo.HaloNotebook(text="Loading embeddings from file...", spinner="line", placement="right"):
    wv_embeddings = gensim.models.keyedvectors.KeyedVectors.load_word2vec_format("data/SO_vectors_200.bin", binary=True)
```

Как пользоваться этими векторами?

Посмотрим на примере одного слова, что из себя представляет embedding

In [6]:

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

```
float32 (200,)
```

In [7]:

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

```
Num of words: 1787145
```

Найдем наиболее близкие слова к слову `dog`:

Вопрос 1:

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

In [8]:

```
# method most_simmilar
wv_embeddings.most_similar(positive=["dog"], topn=5)
```

Out[8]:

```
[('animal', 0.8564180135726929),
 ('dogs', 0.7880866527557373),
```

```
 ('dogs', 0.7860660529327393),
 ('mammal', 0.7623804211616516),
 ('cats', 0.7621253728866577),
 ('animals', 0.760793924331665)]
```

Слово "cat" в единственном числе не входит в топ-5 близких к слову "dog" (однако, его множественное число "cats" является 4-ым по близости 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.

In [9]:

```
# you can use your tokenizer
# for example, from nltk.tokenize import WordPunctTokenizer
tokenizer = nltk.tokenize.WordPunctTokenizer() # you said it, not me
```

In [10]:

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

        return: векторное представление для вопроса
    """
    if dim != embeddings.vector_size:
        raise ValueError("Dimension not equal to embeddings vector size, will not cut or
pad vectors.")
    return np.mean([embeddings[i]
                    if i in embeddings else np.zeros(shape=(dim,))
                    for i in tokenizer.tokenize(question)],
                   axis=0)
```

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

Вопрос 2:

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

In [11]:

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

Out[11]:

```
-0.96
```

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:

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

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

DCG@K

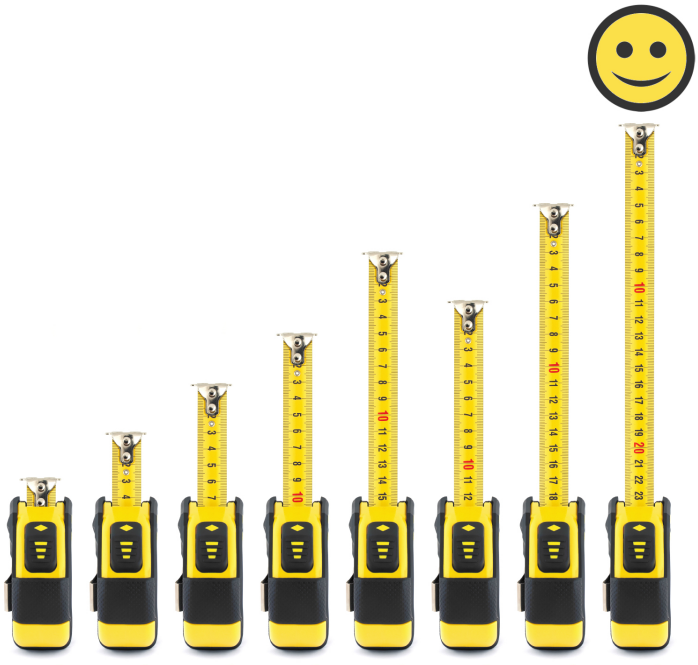
Второй метрикой будет упрощенная **DCG** метрика, учитывающая порядок элементов в списке путем домножения релевантности элемента на вес равный обратному логарифму номера позиции::

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

С такой метрикой модель штрафуется за большой ранк корректного ответа

Вопрос **3:**

- Максимум `Hits@47 - DCG@1`**?**



Пример оценок

Пример оценок

Вычислим описанные выше метрики для игрушечного примера. Пусть

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

Пусть модель выдала следующий ранжированный список кандидатов:

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

$$\Rightarrow rank\_q_i'' = 2$$

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

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

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

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

Вопрос **4:**

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

HITS_COUNT и DCG_SCORE

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

In [12]:

```
def hits_count(dup_ranks, k):
    """
        dup_ranks: list индексов дубликатов
        result: вернуть  Hits@k
    """
    return np.mean((dup_ranks <= k).astype(int))
```

In [13]:

```
def dcg_score(dup_ranks, k):
    """
        dup_ranks: list индексов дубликатов
        result: вернуть DCG@k
    """
    return np.mean((dup_ranks <= k).astype(int) / np.log2(1 + dup_ranks))
```

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

In [14]:

```
copy_answers = ["How does the catch keyword determine the type of exception that was thro
wn",]

# 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"],]
# dup_ranks — позиции наших копий, так как эксперимент один, то этот массив длины 1
_, _, dup_ranks = np.intersect1d(copy_answers, candidates_ranking, return_indices=True)
dup_ranks += 1 # offset for non-programmer counting

# 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

In [15]:

```
# 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
```

Out[15]:

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

Данные

arxiv link

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

`validation.tsv` **- тестовая выборка.**

В каждой строке через табуляцию записаны: <вопрос>, <похожий вопрос>, <отрицательный пример 1>, <отрицательный пример 2>, ...

In [16]:

```
if not (pathlib.Path("data/stackoverflow_similar_questions.zip").is_file()
        or (pathlib.Path("data/train.tsv").is_file() and pathlib.Path("data/validation.t
sv").is_file())):
    gdown.download("https://drive.google.com/uc?id=1QqT4D0EoqJTy7v9VrNCYD-m964XZFR7_", "d
ata/stackoverflow_similar_questions.zip", quiet=False)
```

In [17]:

```
if not (pathlib.Path("data/train.tsv").is_file() and pathlib.Path("data/validation.tsv")
.is_file()):
    !unzip stackoverflow_similar_questions.zip
```

Считайте данные.

In [18]:

```
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**.

In [19]:

```
validation_data = pd.DataFrame(read_corpus("data/validation.tsv")) # for some reason pan
das fails on several lines
```

Кол-во строк

In [20]:

```
len(validation_data)
```

Out[20]:

```
3760
```

In [21]:

```
validation_data.head()
```

Out[21]:

	0	1	2	3	4	5	6	7
0	How to print a binary heap tree without recurs...	How do you best convert a recursive function t...	How can i use ng-model with directive in angul...	flash: drawing and erasing	toggle react component using hide show classname	Use a usercontrol from another project to curr...	~ Paths resolved differently after upgrading t...	Materialize datepicker - Rendering when an ico...
1	How to start PhoneStateListener programmatically?	PhoneStateListener and service	Java cast object[] to model	WCF and What does this mean?	How to uncheck checkbox using jQuery Uniform l...	Two projects with same code base	Can't read php file when upload image to serve...	create pandas dataframe from dictionary of dic...
2	jQuery: Show a div2 when mouseenter	when hover on div1 depending on if it is	How to run selenium in	Python Comparing two lists of	Hazelcast creates 3	JSON-LD framing single	Is there a way to print to	Run server-side code on html

	0	1	2	3	4	5	6	7
	over div1 i...	on di...	google app engine/cloud?	strings fo... simi...	nodes/members when configu...	objects arrays	the console in an A...	button press in E...
3	Performing async method in a loop in node.js a...	Asynchronous sequence of events using promises	Django CMS - not able to upload images through...	Sorting an array by alphabetical order before ...	SQL, Microsoft SQl	call a function in place in Objective c	jquery .bind() and/or .ready() not working	How to filter a list of required elements in l...
4	UE4: output game frames to file	Unreal Engine 4: save rendered frame to memory	How to show an default text when an item of th...	simple beginner search program using arrays in...	When typing in editable ComboBox not showing D...	GMap.Net marker initially in incorrect position	What are we doing wrong with git?	LaTeX: remove blank page after a \part or \cha...

5 rows × 1001 columns

Ранжирование без обучения

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

In [22]:

```
def rank_candidates(question, candidates, embeddings, tokenizer, dim=200):
    """
        question: строка
        candidates: массив строк(кандидатов) [a, b, c]
        result: пары (начальная позиция, кандидат) [(2, c), (0, a), (1, b)]
    """
    question_embedding = question_to_vec(question, embeddings, tokenizer, dim)
    cosine_similarities = sklearn.metrics.pairwise.cosine_similarity([question_embedding
],
                                                      [question_to_vec(i, embeddings, tokenizer,
dim) for i in candidates])[0]
    ranks = sorted([(i, x, y) for i, (x, y) in enumerate(list(zip(candidates, cosine_sim
ilarities)))], key=lambda x: x[-1], reverse=True)
    return [i[:-1] for i in ranks]
```

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

In [23]:

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

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

In [24]:

```
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 (pur
e js) to Python object'), (2, 'How to use jQuery AJAX for an outside domain?')]

[(1, 'WPF- How to update the changes in list item of a list'), (0, 'Getting all list item
s of an unordered list in PHP'), (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")]] #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**?**

`102`

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

In [25]:

```
validation_data.head()
```

Out[25]:

	0	1	2	3	4	5	6	7
0	How to print a binary heap tree without recurs...	How do you best convert a recursive function t...	How can i use ng-model with directive in angul...	flash: drawing and erasing	toggle react component using hide show classname	Use a usercontrol from another project to curr...	~ Paths resolved differently after upgrading t...	Materialize datepicker - Rendering when an ico...
1	How to start PhoneStateListener programmatically?	PhoneStateListener and service	Java cast object[] to model	WCF and What does this mean?	How to uncheck checkbox using jQuery Uniform l...	Two projects with same code base	Can't read php file when upload image to serve...	create pandas dataframe from dictionary of dic...
2	jQuery: Show a div2 when mousenter over div1 i...	when hover on div1 depenting on if it is on di...	How to run selenium in google app engine/cloud?	Python Comparing two lists of strings for simi...	Hazelcast creates 3 nodes/members when configu...	JSON-LD framing single object arrays	Is there a way to print to the console in an A...	Run server-side code on html button press in E...

	0	1	2	3	4	5	6	7
3	Performing async method in a loop in node.js a...	Asynchronous sequence of events using promises	Django CMS - not able to upload images through...	Sorting an array by alphabetical order before ...	SQL, Microsoft SQl	call a function in place in Objective c	.bind() and/or .ready() not working	How to filter a list of required elements in l...
4	UE4: output game frames to file	Unreal Engine 4: save rendered frame to memory	How to show an default text when an item of th...	simple beginner search program using arrays in...	When typing in editable ComboBox not showing D...	GMap.Net marker initially in incorrect position	What are we doing wrong with git?	LaTeX: remove blank page after a \part or \cha...

5 rows × 1001 columns

In [26]:

```
def create_dataset_ranking(data, embeddings, tokenizer, max_validation_examples=1000):
    wv_ranking = []
    q = data.iloc[:, 0]
    ex = data.iloc[:, 1:]
    wv_ranking = [rank_candidates(question_i, candidates_i, embeddings, tokenizer)
                  for question_i, candidates_i in tqdm.tqdm(zip(
                      q.values.tolist()[:max_validation_examples],
                      ex.values.tolist()[:max_validation_examples]),
                                                            total=max_validation_examples,
                                                            desc="Ranking candidates",
                                                            unit="question(s)",
                                                            unit_scale=False)]
    wv_ranking = np.asarray([i[0][0] + 1 for i in wv_ranking])
    return wv_ranking
wv_ranking = create_dataset_ranking(validation_data, wv_embeddings, tokenizer)
```

In [27]:

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

In [28]:

```
print_metrics(wv_ranking)
```

```
DCG@   1: 0.276 | Hits@   1: 0.276
DCG@   5: 0.278 | Hits@   5: 0.280
DCG@  10: 0.280 | Hits@  10: 0.285
DCG@ 100: 0.290 | Hits@ 100: 0.339
DCG@ 500: 0.328 | Hits@ 500: 0.648
DCG@1000: 0.365 | Hits@1000: 1.000
```

In [29]:

```
del wv_embeddings
del wv_ranking
gc.collect()
```

Out[29]:

```
1041
```

Эмбеддинги, обученные на корпусе похожих вопросов

In [30]:

```
train_data = pd.DataFrame(read_corpus("./data/train.tsv")).iloc[:, :2]
train_data.head()
```

Out[30]:

	0	1
0	converting string to list	Convert Google results object (pure js) to Pyt...
1	Which HTML 5 Canvas Javascript to use for maki...	Event handling for geometries in Three.js?
2	Sending array via Ajax fails	Getting all list items of an unordered list in...
3	How to insert CookieCollection to CookieContai...	C# create cookie from string and send it
4	Updating one element of a bound Observable col...	WPF- How to update the changes in list item of...

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

In [123]:

```
with halo.HaloNotebook(text="Creating sentences iterable...", spinner="line", placement="
right"):
    words = train_data.apply(lambda x: [tokenizer.tokenize(i) for i in x])
    words = (words[0] + words[1]).values.tolist()
```

Проверим максимальную длину запроса после токенизации:

In [124]:

```
max([len(i) for i in words])
```

Out[124]:

```
105
```

Экспериментальным путём получено значение размера окна в 100 слов (изменения метрик выше этого значения незначительны), что близко к максимальной длине вопроса в датасете. Обычно больший размер окна "запоминает" больше данных о теме запроса, что важно в данной задаче.

(Источник: https://levyomer.files.wordpress.com/2014/04/dependency-based-word-embeddings-acl-2014.pdf)

In [126]:

```
with halo.HaloNotebook(text="Creating Word2Vec model from questions...", spinner="line",
placement="right"):
    embeddings_trained = gensim.models.Word2Vec(words, # data for model to train on
                                                vector_size=200, # embedding vector siz
e
                                                seed=RANDOM_STATE, # set for reproducib
ility
                                                min_count=5, # consider words that occu
red at least 5 times
                                                window=100).wv
```

In [69]:

```
train_ranking = create_dataset_ranking(validation_data, embeddings_trained, tokenizer)
```

In [70]:

```
print_metrics(train_ranking)
```

```
DCG@   1: 0.317 | Hits@   1: 0.317
DCG@   5: 0.318 | Hits@   5: 0.319
DCG@  10: 0.320 | Hits@  10: 0.324
DCG@ 100: 0.331 | Hits@ 100: 0.382
DCG@ 500: 0.364 | Hits@ 500: 0.645
```

```
DCG@1000: 0.401 | Hits@1000: 1.000
```

In [71]:

```
del words
del embeddings_trained
del train_ranking
gc.collect()
```

Out[71]:

```
1041
```

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".

Эксперименты

Токенайзеры и нормализация

Попробуем другие токенайзеры (поскольку `WordPunctTokenizer` довольно простой). Начнём с `NISTTokenizer`.

In [84]:

```
nist_tokenizer = nltk.tokenize.nist.NISTTokenizer()
```

In [87]:

```
with halo.HaloNotebook(text="Creating sentences iterable...", spinner="line", placement="
right"):
    words = train_data.apply(lambda x: [nist_tokenizer.tokenize(i, lowercase=True) for i
in x])
    words = (words[0] + words[1]).values.tolist()
```

In [88]:

```
with halo.HaloNotebook(text="Creating Word2Vec model from questions...", spinner="line",
placement="right"):
    embeddings_trained_nist = gensim.models.Word2Vec(words, # data for model to train on
                                                     vector_size=200, # embedding vector siz
e
                                                     seed=RANDOM_STATE, # set for reproducib
ility
                                                     min_count=5, # consider words that occu
red at least 5 times
                                                     window=100).wv
```

In [89]:

```
train_ranking_nist = create_dataset_ranking(validation_data, embeddings_trained_nist, tok
enizer)
```

In [90]:

```
print_metrics(train_ranking_nist)
```

```
DCG@   1: 0.206 | Hits@   1: 0.206
DCG@   5: 0.207 | Hits@   5: 0.208
DCG@  10: 0.208 | Hits@  10: 0.213
DCG@ 100: 0.221 | Hits@ 100: 0.282
```

```
DCG@ 500: 0.259 | Hits@ 500: 0.590
DCG@1000: 0.303 | Hits@1000: 1.000
```

In [91]:

```
del words
del embeddings_trained_nist
del train_ranking_nist
gc.collect()
```

Out[91]:

```
1041
```

Качество ухудшилось, попробуем `TweetTokenizer`.

In [95]:

```
tweet_tokenizer = nltk.tokenize.TweetTokenizer()
```

In [99]:

```
with halo.HaloNotebook(text="Creating sentences iterable...", spinner="line", placement="
right"):
    words = train_data.apply(lambda x: [tweet_tokenizer.tokenize(i) for i in x])
    words = (words[0] + words[1]).values.tolist()
```

In [100]:

```
with halo.HaloNotebook(text="Creating Word2Vec model from questions...", spinner="line",
placement="right"):
    embeddings_trained_tweet = gensim.models.Word2Vec(words, # data for model to train on
                                                      vector_size=200, # embedding vector siz
e
                                                      seed=RANDOM_STATE, # set for reproducib
ility
                                                      min_count=5, # consider words that occu
red at least 5 times
                                                      window=100).wv
```

In [101]:

```
train_ranking_tweet = create_dataset_ranking(validation_data, embeddings_trained_tweet, t
okenizer)
```

In [102]:

```
print_metrics(train_ranking_tweet)
```

```
DCG@   1: 0.306 | Hits@   1: 0.306
DCG@   5: 0.307 | Hits@   5: 0.308
DCG@  10: 0.309 | Hits@  10: 0.313
DCG@ 100: 0.318 | Hits@ 100: 0.365
DCG@ 500: 0.353 | Hits@ 500: 0.638
DCG@1000: 0.391 | Hits@1000: 1.000
```

In [103]:

```
del words
del embeddings_trained_tweet
del train_ranking_tweet
gc.collect()
```

Out[103]:

```
1041
```

Незначительное ухудшение с сравнении с `WordPunctTokenizer`. Попрооуем `TreebankWordTokenizer`.

In [104]:

```
tree_tokenizer = nltk.tokenize.TreebankWordTokenizer()
```

In [105]:

```
with halo.HaloNotebook(text="Creating sentences iterable...", spinner="line", placement="
right"):
    words = train_data.apply(lambda x: [tree_tokenizer.tokenize(i) for i in x])
    words = (words[0] + words[1]).values.tolist()
```

In [106]:

```
with halo.HaloNotebook(text="Creating Word2Vec model from questions...", spinner="line",
placement="right"):
    embeddings_trained_tree = gensim.models.Word2Vec(words, # data for model to train on
                                                     vector_size=200, # embedding vector siz
e
                                                     seed=RANDOM_STATE, # set for reproducib
ility
                                                     min_count=5, # consider words that occu
rred at least 5 times
                                                     window=100).wv
```

In [108]:

```
train_ranking_tree = create_dataset_ranking(validation_data, embeddings_trained_tree, tok
enizer)
```

In [109]:

```
print_metrics(train_ranking_tree)
```

```
DCG@   1: 0.295 | Hits@   1: 0.295
DCG@   5: 0.297 | Hits@   5: 0.299
DCG@  10: 0.298 | Hits@  10: 0.303
DCG@ 100: 0.307 | Hits@ 100: 0.351
DCG@ 500: 0.341 | Hits@ 500: 0.623
DCG@1000: 0.381 | Hits@1000: 1.000
```

In [ ]:

```
del words
del embeddings_trained_tree
del train_ranking_tree
gc.collect()
```

Качество хуже `WordPunctTokenizer`. Попробуем добавить к нему `lower()`.

In [111]:

```
with halo.HaloNotebook(text="Creating sentences iterable...", spinner="line", placement="
right"):
    words = train_data.apply(lambda x: [tokenizer.tokenize(i.lower()) for i in x])
    words = (words[0] + words[1]).values.tolist()
```

In [112]:

```
with halo.HaloNotebook(text="Creating Word2Vec model from questions...", spinner="line",
placement="right"):
    embeddings_trained_lower = gensim.models.Word2Vec(words, # data for model to train on
                                                     vector_size=200, # embedding vector siz
e
                                                     seed=RANDOM_STATE, # set for reproducib
```

```
ility
                                                    min_count=5, # consider words that occu
red at least 5 times
                                                    window=100).wv
```

In [113]:

```
train_ranking_lower = create_dataset_ranking(validation_data, embeddings_trained_lower, t
okenizer)
```

In [114]:

```
print_metrics(train_ranking_lower)
```

```
DCG@   1: 0.209 | Hits@   1: 0.209
DCG@   5: 0.210 | Hits@   5: 0.212
DCG@  10: 0.212 | Hits@  10: 0.216
DCG@ 100: 0.223 | Hits@ 100: 0.280
DCG@ 500: 0.261 | Hits@ 500: 0.584
DCG@1000: 0.305 | Hits@1000: 1.000
```

In [115]:

```
del words
del embeddings_trained_lower
del train_ranking_lower
gc.collect()
```

Out[115]:

```
1041
```

`lower()` **не помогает, попробуем** `strip()`**.**

In [117]:

```
with halo.HaloNotebook(text="Creating sentences iterable...", spinner="line", placement="
right"):
    words = train_data.apply(lambda x: [tokenizer.tokenize(i.strip()) for i in x])
    words = (words[0] + words[1]).values.tolist()
```

In [118]:

```
with halo.HaloNotebook(text="Creating Word2Vec model from questions...", spinner="line",
placement="right"):
    embeddings_trained_strip = gensim.models.Word2Vec(words, # data for model to train on
                                                    vector_size=200, # embedding vector siz
e
                                                    seed=RANDOM_STATE, # set for reproducib
ility
                                                    min_count=5, # consider words that occu
red at least 5 times
                                                    window=100).wv
```

In [119]:

```
train_ranking_strip = create_dataset_ranking(validation_data, embeddings_trained_strip, t
okenizer)
```

In [120]:

```
print_metrics(train_ranking_strip)
```

```
DCG@   1: 0.312 | Hits@   1: 0.312
DCG@   5: 0.312 | Hits@   5: 0.313
DCG@  10: 0.314 | Hits@  10: 0.319
```

```
DCG@ 100: 0.324 | Hits@ 100: 0.372
DCG@ 500: 0.358 | Hits@ 500: 0.636
DCG@1000: 0.396 | Hits@1000: 1.000
```

In [121]:

```
del words
del embeddings_trained_strip
del train_ranking_strip
gc.collect()
```

Out[121]:

```
1041
```

Несущественная разница с "чистым" `WordPunctTokenizer`.

Нормализация

Попробуем применить стемминг.

In [130]:

```
stemmer = nltk.stem.PorterStemmer()
```

In [135]:

```
with halo.HaloNotebook(text="Creating sentences iterable...", spinner="line", placement="
right"):
    words = train_data.apply(lambda x: [tokenizer.tokenize(i) for i in x])
    words = (words[0].apply(lambda x: [stemmer.stem(i) for i in x]) + words[1].apply(lam
bda x: [stemmer.stem(i) for i in x])).values.tolist()
```

In [137]:

```
with halo.HaloNotebook(text="Creating Word2Vec model from questions...", spinner="line",
placement="right"):
    embeddings_trained = gensim.models.Word2Vec(words, # data for model to train on
                                                vector_size=200, # embedding vector siz
e
                                                seed=RANDOM_STATE, # set for reproducib
ility
                                                min_count=5, # consider words that occu
red at least 5 times
                                                window=100).wv
```

In [138]:

```
train_ranking = create_dataset_ranking(validation_data, embeddings_trained, tokenizer)
```

In [139]:

```
print_metrics(train_ranking)
```

```
DCG@   1: 0.153 | Hits@   1: 0.153
DCG@   5: 0.154 | Hits@   5: 0.155
DCG@  10: 0.155 | Hits@  10: 0.157
DCG@ 100: 0.170 | Hits@ 100: 0.241
DCG@ 500: 0.211 | Hits@ 500: 0.569
DCG@1000: 0.256 | Hits@1000: 1.000
```

In [140]:

```
del words
del embeddings_trained
```

```
del train_ranking
gc.collect()
```

Out[140]:

```
1041
```

Серьёзное падение качества. Попробуем лемматизацию.

In [143]:

```
lemmatizer = nltk.WordNetLemmatizer()
```

In [144]:

```
with halo.HaloNotebook(text="Creating sentences iterable...", spinner="line", placement="
right"):
    words = train_data.apply(lambda x: [tokenizer.tokenize(i) for i in x])
    words = (words[0].apply(lambda x: [lemmatizer.lemmatize(i) for i in x])
             + words[1].apply(lambda x: [lemmatizer.lemmatize(i) for i in x]))\
    .values.tolist()
```

In [145]:

```
with halo.HaloNotebook(text="Creating Word2Vec model from questions...", spinner="line",
placement="right"):
    embeddings_trained = gensim.models.Word2Vec(words, # data for model to train on
                                                vector_size=200, # embedding vector siz
e
                                                seed=RANDOM_STATE, # set for reproducib
ility
                                                min_count=5, # consider words that occu
red at least 5 times
                                                window=100).wv
```

In [146]:

```
train_ranking = create_dataset_ranking(validation_data, embeddings_trained, tokenizer)
```

In [147]:

```
print_metrics(train_ranking)
```

```
DCG@   1: 0.292 | Hits@   1: 0.292
DCG@   5: 0.292 | Hits@   5: 0.292
DCG@  10: 0.294 | Hits@  10: 0.297
DCG@ 100: 0.305 | Hits@ 100: 0.360
DCG@ 500: 0.340 | Hits@ 500: 0.636
DCG@1000: 0.378 | Hits@1000: 1.000
```

In [148]:

```
del words
del embeddings_trained
del train_ranking
gc.collect()
```

Out[148]:

```
1041
```

Лучше, чем эффект от стемминга, но всё ещё хуже baseline.

Вывод

Напишите свой вывод о полученных результатах.

Напишите свой вывод о полученных результатах.

- Какой принцип токенизации даёт качество лучше и почему? : `WordPunctTokenizer`, видимо, из-за особенностей датасета (небольшие фразы + наличие в словах спецсимволов)
- Помогает ли нормализация слов? : Приведение к нижнему регистру сильно ухудшает качество, удаление пунктуации практически не возымело эффекта. Стемминг и лемматизация приводили к ухудшению качества.
- Какие эмбеддинги лучше справляются с задачей и почему? Обученные с нуля (за счёт соединения двух похожих запросов в одно предложение)
- Почему получилось плохое качество решения задачи? Крайне простой способ получения векторов вопросов (простым усреднением).
- Предложите свой подход к решению задачи. Возможно, стоит заняться **Topic Modelling** (например, попробовать **Latent Dirichlet Allocation**), или использовать **RNN** для получения векторов предложений. Можно попробовать и трансформеры, но не факт, что применения настолько "тяжёлой" модели будет оправдано. (Кажется, наш текущий метод плохо определяет тему предложения, что для ранжирования запросов крайне важно, более продвинутые методы справляются с этой задачей лучше)