Deep Learning School

Физтех-Школа Прикладной математики и информатики (ФПМИ) МФТИ

Прочитайте семинар, пожалуйста, для успешного выполнения домашнего задания

Задача поиска схожих по смыслу предложений

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

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

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 $a : X \to R$ такую, что**

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

Данные

`test.tsv` - **тестовая выборка. В каждой строке через табуляцию записаны: *<вопрос>, <похожий вопрос>, <отрицательный пример 1>, <отрицательный пример 2>, ...***

Будем использовать предобученные векторные представления слов [GoogleNews-vectors-negative300 (https://drive.google.com/file/d/0B7XkCwpI5KDYNlNUTTlSS21pQmM/edit)](https://drive.google.com/file/d/0B7XkCwpI5KDYNlNUTTlSS21pQmM/edit) (скачаем их), которые были обучены с помощью стандартной модели word2vec на данных Google News (100 миллиардов слов). Модель содержит 300-мерные вектора для 3 миллионов слов и фраз

Загрузим их после скачивания с помощью функции [KeyedVectors.load_word2vec_format (https://radimrehurek.com/gensim/models/keyedvectors.html)](https://radimrehurek.com/gensim/models/keyedvectors.html) библиотеки Gensim, с которой вы познакомились на семинаре. Загрузим только часть векторов, указав параметр *limit* = 500000.

In [0]:

```
# download and unpack tsnecuda from anaconda.org

!wget https://anaconda.org/CannyLab/tsnecuda/2.1.0/download/linux-64/tsnecuda-2.1.0-cud
a100.tar.bz2
!tar xvjf tsnecuda-2.1.0-cuda100.tar.bz2
!cp -r site-packages/* /usr/local/lib/python3.6/dist-packages/
```

In [0]:

```
# create a symbolic link between the downloaded libfaiss.so file and the location pytho
n's looking at

!echo $LD_LIBRARY_PATH
# this is probably /usr/lib64-nvidia

!ln -s /content/lib/libfaiss.so $LD_LIBRARY_PATH/libfaiss.so
```

In [0]:

```
import gensim
import tsnecuda
tsnecuda.test()

!wget -c "https://s3.amazonaws.com/dl4j-distribution/GoogleNews-vectors-negative300.bi
n.gz"
wv_embeddings = gensim.models.KeyedVectors.load_word2vec_format(\
       fname='GoogleNews-vectors-negative300.bin.gz', limit=500000, binary=True)
```

Как пользоваться этими векторами?

Посмотрим на примере одного слова, что из себя представляет embedding

In [4]:

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

```
float32 (300,)
```

Еще раз напомню, что семинар нужно прочитать

Найдем наиболее близкие слова к слову dog :

In [5]:

```
wv_embeddings.most_similar('dog')
```

```
/usr/local/lib/python3.6/dist-packages/gensim/matutils.py:737: FutureWarni
ng: Conversion of the second argument of issubdtype from `int` to `np.sign
edinteger` is deprecated. In future, it will be treated as `np.int64 == n
p.dtype(int).type`.
  if np.issubdtype(vec.dtype, np.int):
```

Out[5]:

```
[('dogs', 0.8680489659309387),
 ('puppy', 0.8106428384780884),
 ('pit_bull', 0.780396044254303),
 ('pooch', 0.7627377510070801),
 ('cat', 0.7609456777572632),
 ('golden_retriever', 0.7500902414321899),
 ('German_shepherd', 0.7465174198150635),
 ('Rottweiler', 0.7437614798545837),
 ('beagle', 0.7418621778488159),
 ('pup', 0.740691065788269)]
```

Вопрос 1:

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

In [6]:

```
wv_embeddings.most_similar('dog') #your code
```

```
/usr/local/lib/python3.6/dist-packages/gensim/matutils.py:737: FutureWarni
ng: Conversion of the second argument of issubdtype from `int` to `np.sign
edinteger` is deprecated. In future, it will be treated as `np.int64 == n
p.dtype(int).type`.
  if np.issubdtype(vec.dtype, np.int):
```

Out[6]:

```
[('dogs', 0.8680489659309387),
 ('puppy', 0.8106428384780884),
 ('pit_bull', 0.780396044254303),
 ('pooch', 0.7627377510070801),
 ('cat', 0.7609456777572632),
 ('golden_retriever', 0.7500902414321899),
 ('German_shepherd', 0.7465174198150635),
 ('Rottweiler', 0.7437614798545837),
 ('beagle', 0.7418621778488159),
 ('pup', 0.740691065788269)]
```

Ответ: да, слово `cat` находится на 5-ом месте топ-5 близких слов к слову `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 [0]:

```
import numpy as np
from nltk.tokenize import WordPunctTokenizer
tokenizer = WordPunctTokenizer()
```

In [0]:

```
def question_to_vec(question, embeddings, dim=300):
    """
        question: строка
        embeddings: наше векторное представление
        dim: размер любого вектора в нашем представлении

        return: векторное представление для вопроса
    """
    words = question.split(' ') #your code
    # убрать знак вопроса, если он есть
    n_known = 0
    result = np.array([0] * dim, dtype=float)

    for word in words:
        if word in embeddings:
            result += embeddings[word] #your code
            n_known += 1

    if n_known != 0:
        return result / n_known #your code
    else:
        return result
```

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

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:

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

- 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 низкую позицию корректного ответа

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

HITS_COUNT и DCG_SCORE

Каждая функция имеет два аргумента: dup_ranks **и** k**.** dup_ranks **является списком, который содержит рейтинги дубликатов** $rank\_q_i^{'}$ **(их позиции в ранжированном списке). Например,** $dup\_ranks = [2]$ **для примера, описанного выше.**

In [0]:

```
def hits_count(dup_ranks, k):
    """
        result: Вернуть  Hits@k
    """

    N = len(dup_ranks)
    hits_value = np.mean((np.array(dup_ranks) <=k)) #your code
    return hits_value
```

In [0]:

```
def dcg_score(dup_ranks, k):
    """
        result: вернуть DCG@k
    """
    """

    :param dup_ranks:
    :param k:
    :return:

    """
    new_dup_ranks = np.array(dup_ranks)
    N = len(dup_ranks)
    #your code
    dcg_value =np.mean( 1 / np.log2(1+new_dup_ranks) * (new_dup_ranks <=k))
    #dcg_value = hits_count(dup_ranks, k)/ np.log2(1+new_dup_ranks)
    return dcg_value
```

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

In [11]:

```
import pandas as pd
copy_answers = ["How does the catch keyword determine the type of exception that was th
rown"]

# наги кандидаты
candidates_ranking = [["How Can I Make These Links Rotate in PHP",
                       "How does the catch keyword determine the type of exception that
was 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 = [candidates_ranking[i].index(copy_answers[i]) + 1 for i in range(len(copy_a
nswers))]
# вычисляем метрику для разных k
print('Ваш ответ HIT:', [hits_count(dup_ranks, k) for k in range(1, 5)])
print('Ваш ответ DCG:', [np.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 [12]:

```
# 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[12]:

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

Ранжирование вопросов StackOverflow

- ***тестовая*** **выборка (test.tsv) содержит в каждой строке:** ***вопрос, похожий вопрос, отрицательный пример 1, отрицательный пример 2, ...*** **TEST!!!**

Считаем тестовую выборку для оценки качества текущего решения.

In [0]:

```
!gdown https://drive.google.com/uc?id=12VAWjs-kvpgapurkuURus7E53F711uFV&export=download
```

In [0]:

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

In [0]:

```
test = read_corpus('test.tsv')
```

Кол-во строк

In [16]:

```
len(test)
```

Out[16]:

```
3760
```

Размер нескольких первых строк

In [17]:

```
for i in range(5):
    print(i + 1, len(test[0]))
```

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

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 [0]:

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

In [0]:

```
def rank_candidates(question, candidates, embeddings, dim=300):
    """
        question: строка
        candidates: массив строк(кандидатов) [a, b, c]
        result: пары (начальная позиция, кандидат) [(2, c), (0, a), (1, b)]
    """
    vec_question = question_to_vec(question, embeddings, dim)
    vec_candidates = np.array([vec_question for i in range(len(candidates))])
    rank_candidates = np.array([(i, candidates[i]) for i in range(len(candidates))])
    # ранжирование
    dist_s = cosine_similarity(np.array([question_to_vec(i, embeddings, dim) for i in c
andidates]), np.array([vec_question]))[:, 0]
    return deepcopy(rank_candidates[dist_s.argsort()[::-1]])
```

In [0]:

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

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

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

In [21]:

```
for question, q_candidates in zip(questions, candidates):
        ranks = rank_candidates(question, q_candidates, wv_embeddings, 300)
        print(ranks)
        print()
```

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

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

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

In [0]:

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

Последовательность начальных индексов вы должны получить для эксперемінта 1 1, 0, 2. Для второго экперимента вы знаете один индекс уже.

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).

In [0]:

```
wv_ranking = []
for line in test:
    q, *ex = line
    ranks = rank_candidates(q, ex, wv_embeddings)
    wv_ranking.append([r[0] for r in ranks].index('0') + 1)
```

In [25]:

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

```
DCG@   1: 0.210 | Hits@   1: 0.210
DCG@   5: 0.262 | Hits@   5: 0.308
DCG@  10: 0.279 | Hits@  10: 0.362
DCG@ 100: 0.319 | Hits@ 100: 0.560
DCG@ 500: 0.352 | Hits@ 500: 0.817
DCG@1000: 0.371 | Hits@1000: 1.000
```

Если вы проделали все шаги правильно, то вы должны немного разочароваться полученными результатами. Давайте попробуем понять, почему качество модели такое низкое. Когда вы работаете с какими-либо данными, очень полезно первым делом посмотреть на них глазами. Выведим несколько вопросов из наших данных:

In [26]:

```
for line in test[:3]:
    q, *examples = line
    print(q, *examples[:3])
    print()
```

```
How to print a binary heap tree without recursion? How do you best convert
a recursive function to an iterative one? How can i use ng-model with dire
ctive in angular js flash: drawing and erasing

How to start PhoneStateListener programmatically? PhoneStateListener and s
ervice Java cast object[] to model WCF and What does this mean?

jQuery: Show a div2 when mousenter over div1 is over when hover on div1 de
penting on if it is on div2 or not it should act differently How to run se
lenium in google app engine/cloud? Python Comparing two lists of strings f
or similarities
```

Как вы можете заметить, мы имеем дело с сырыми данными. Это означает, что там присутствует много опечаток, спецсимволов и заглавных букв. В нашем случае это все может привести к ситуации, когда для данных токенов нет предобученных векторов. Поэтому необходима предобработка.

Реализуйем функцию предобработки текстов. Вам требуется:

- **Перевести символы в нижний регистр;**
- **Заменить символы пунктуации на пробелы;**
- **Удалить "плохие" символы;**
- **Удалить стопслова.**

In [23]:

```
import re
import nltk
import string
nltk.download('stopwords')
from nltk.corpus import stopwords
```

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

Кол-во стоп стопслов

In [24]:

```
stopWords = set(stopwords.words('english'))
len(stopWords)
```

Out[24]:

```
179
```

In [0]:

```
def text_prepare(text):
    """
        text: a string

        return: modified string
    """
    # Перевести символы в нижний регистр
    text = text.lower() #your code

    # Заменить символы пунктуации на пробелы
    text = re.sub(r'[{}]'.format(string.punctuation), ' ', text)

    # Удалить "плохие" символы
    text = re.sub('[^A-Za-z0-9 ]', '', text)

    # Удалить стопслова.
    stopWords = set(stopwords.words('english'))
    for stopWord in stopWords:
        text = re.sub(r'\b{}\b'.format(stopWord), '', text)
    return text
```

In [30]:

```
from tqdm import tqdm_notebook
from copy import deepcopy
new_test = deepcopy(test)
for i in tqdm_notebook(range(len(test))):
    for j in range(len(test[i])):
        new_test[i][j] = text_prepare(test[i][j])
```

```
/usr/local/lib/python3.6/dist-packages/ipykernel_launcher.py:4: TqdmDeprec
ationWarning: This function will be removed in tqdm==5.0.0
Please use `tqdm.notebook.tqdm` instead of `tqdm.tqdm_notebook`
  after removing the cwd from sys.path.
```

Теперь преобразуйте все вопросы из тестовой выборки. Оцените, как изменилось качество. Сделайте выводы. Для изменения текста понадобиться около 30 минут.

In [0]:

```
wv_ranking = []
for line in new_test:
    q, *ex = line
    ranks = rank_candidates(q, ex, wv_embeddings)
    wv_ranking.append([r[0] for r in ranks].index('0') + 1)
```

In [32]:

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

```
DCG@   1: 0.340 | Hits@   1: 0.340
DCG@   5: 0.412 | Hits@   5: 0.477
DCG@  10: 0.427 | Hits@  10: 0.523
DCG@ 100: 0.463 | Hits@ 100: 0.702
DCG@ 500: 0.482 | Hits@ 500: 0.855
DCG@1000: 0.498 | Hits@1000: 1.000
```

Finally!.. Visualization! (Again..)

Раз уж мы научились получать эмбеддинги предложений, а не только слов, давайте попробуем визуализировать эмбеддинги предложений!

Функция получения эмбеддинга по предложению у нас уже есть (question_to_vec в начале ноутбука). Нам осталось выбрать, какой датасет мы будем использовать (quora.txt с семинара или stackoverflow из этого дз), и далее:

1. **Проделать предобработку вопросов (text_prepare, как выше)**
2. **Для всех вопросов получить эмбединнги (question_to_vec)**
3. **Применить к массиву эмбеддингов TSNE (как на семинаре)**
4. **Не забыть нормализовать векторы, полученные из TSNE**
5. **Запустить функцию draw из семинара!**

In [0]:

```
!wget -O quora.txt https://www.dropbox.com/s/obaitrix9jyu84r/quora.txt?dl=1
```

In [0]:

```
quora = read_corpus('quora.txt')
```

In [0]:

```
from tqdm import tqdm_notebook
from copy import deepcopy
```

1.Text Preparing

In [29]:

```
new_test_quora = deepcopy(quora)
for i in tqdm_notebook(range(len(quora))):
    for j in range(len(quora[i])):
        new_test_quora[i][j] = text_prepare(quora[i][j])
```

```
/usr/local/lib/python3.6/dist-packages/ipykernel_launcher.py:2: TqdmDeprec
ationWarning: This function will be removed in tqdm==5.0.0
Please use `tqdm.notebook.tqdm` instead of `tqdm.tqdm_notebook`
```

2. Embeddings

In [0]:

```
quora_vectors_emb = []
for num in new_test_quora:
    q, *ex = num
    quora_vectors_emb.append(question_to_vec(q, wv_embeddings))
```

In [0]:

```
quora_vectors_emb = np.array(quora_vectors_emb)
```

3. TSNE_CUDA

In [0]:

```
from tsnecuda import TSNE as TSNE_CUDA
tsne = TSNE_CUDA(n_components=2, verbose=50)
quora_vectors_tsne = tsne.fit_transform(quora_vectors_emb)
```

4. StandardScaler

In [0]:

```
from sklearn.preprocessing import StandardScaler

ss = StandardScaler().fit(quora_vectors_tsne)
quora_vectors_tsne = ss.transform(quora_vectors_tsne)
```

4. Draw

In [0]:

```
from nltk.tokenize import WordPunctTokenizer
import bokeh.models as bm, bokeh.plotting as pl
from bokeh.io import output_notebook
output_notebook()
```

Каждое предложение

In [0]:

```
quora_tokenized = [line[0] for line in quora]
```

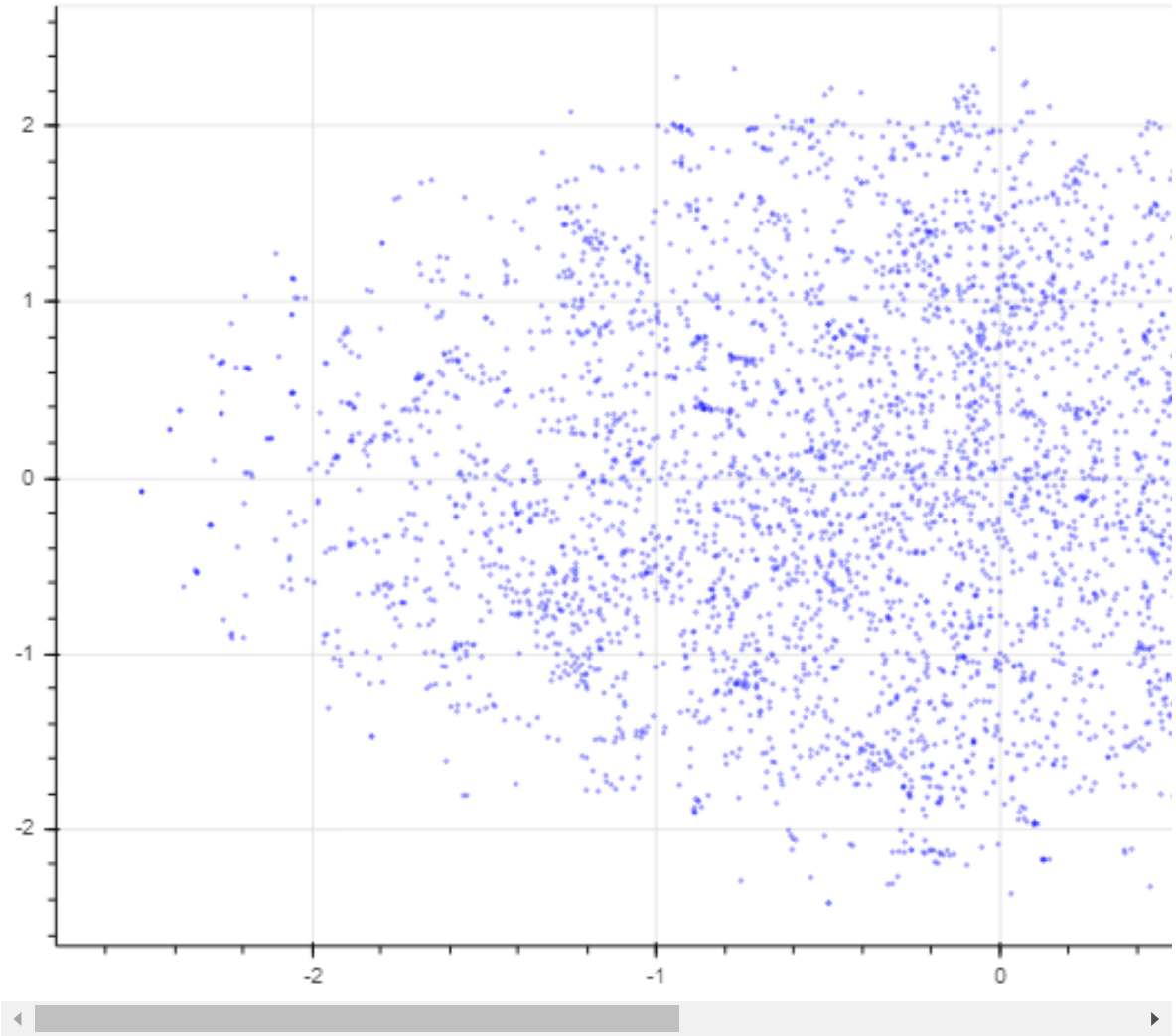
In [0]:

```
import bokeh.models as bm, bokeh.plotting as pl
from bokeh.io import output_notebook
output_notebook()

def draw_vectors(x, y, radius=2, alpha=0.25, color='blue',
                 width=1000, height=500, show=True, **kwargs):
    if isinstance(color, str): color = [color] * len(x)
    data_source = bm.ColumnDataSource({ 'x' : x, 'y' : y, 'color': color, **kwargs })

    fig = pl.figure(active_scroll='wheel_zoom', width=width, height=height)
    fig.scatter('x', 'y', size=radius, color='color', alpha=alpha, source=data_source)

    fig.add_tools(bm.HoverTool(tooltips=[(key, "@" + key) for key in kwargs.keys()]))
    if show: pl.show(fig)
    return fig
```

Построим для 5000 точек вместе с `token` , потому что для размерности в 500000 компьютер взрывается, поэтому приклеплю лишь картинку дял полной размерности

In [37]:

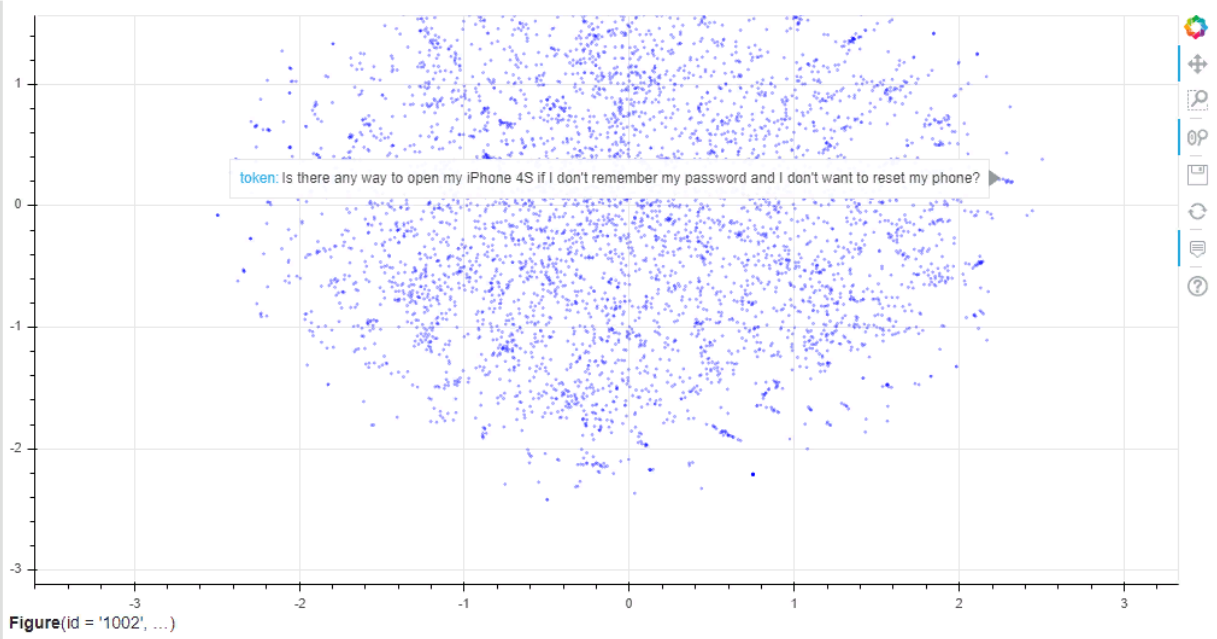
```
draw_vectors(quora_vectors_tsne[:, 0][0:5000], quora_vectors_tsne[:, 1][0:5000],token=q
uora[0:5000])
```



Out[37]:

Figure(id = '1002', …)

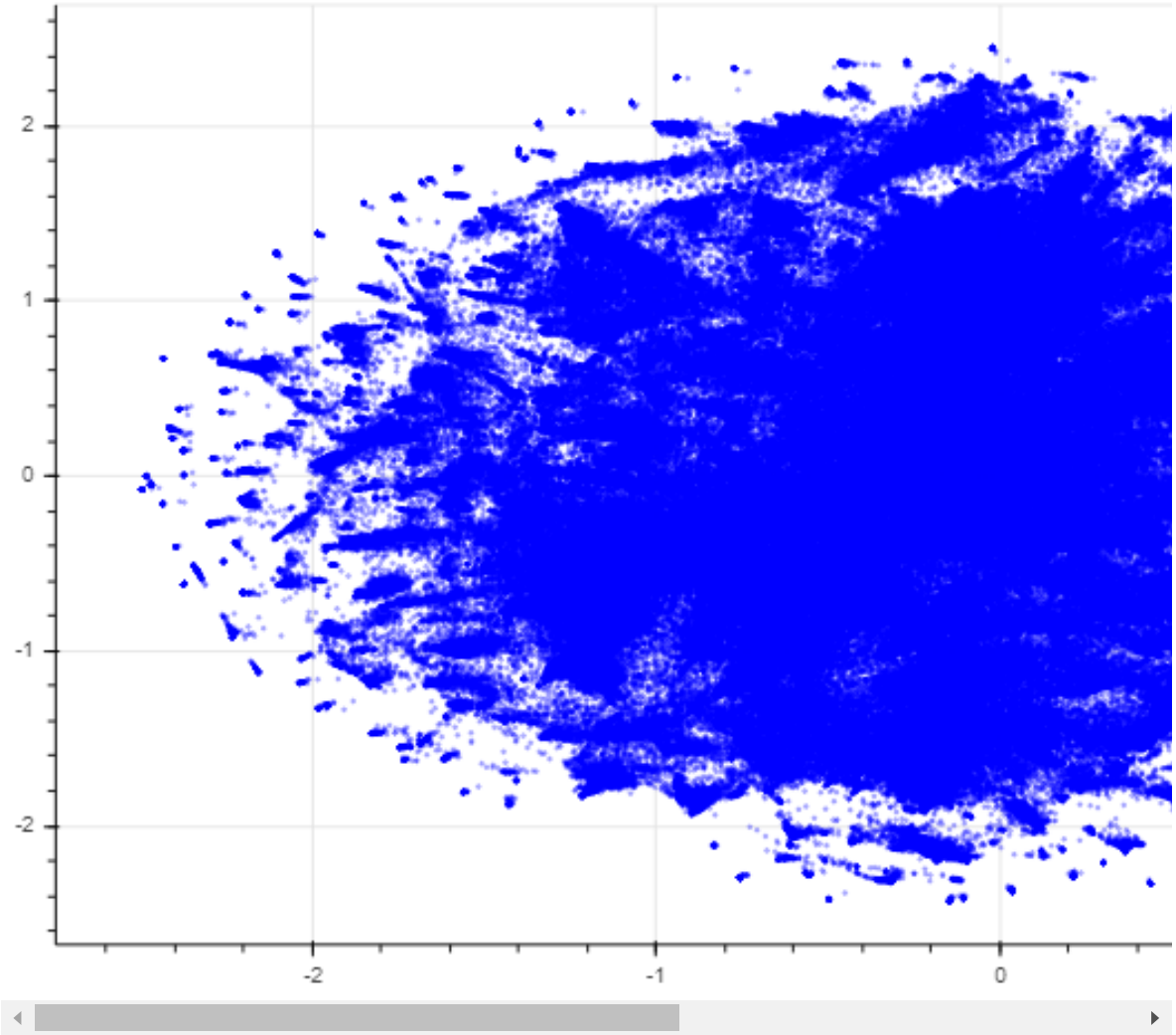
```
draw_vectors(quora_vectors_tsne[:, 0][0:5000], quora_vectors_tsne[:, 1][0:5000],token=quora[0:5000])
```



Figure(id = '1002', …)

In [41]:

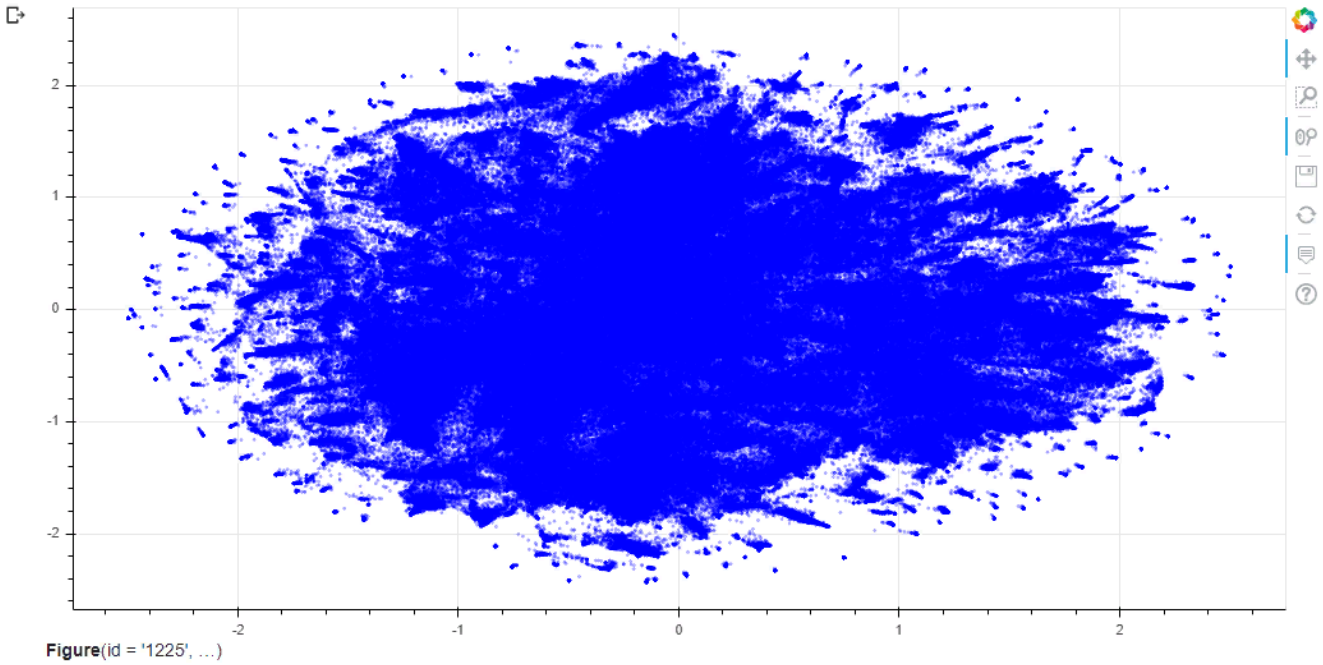
```
draw_vectors(quora_vectors_tsne[:, 0], quora_vectors_tsne[:, 1])
```



Out[41]:

Figure(id = '1225', ...)

```
[41] draw_vectors(quora_vectors_tsne[:, 0], quora_vectors_tsne[:, 1])
```



Figure(id = '1225', ...)

При сохранении данные графики не сохранились, поэтому вставил скрины.

Отлично. Но, к сожалению, визуализацию тестами на канвасе проверить нельзя, поэтому давайте еще напишем функцию, которая будет находить к вопросу ближайшие похожие =)

In [0]:

```
import operator
def find_closest_questions(question, k=5):
    """
    function that finds closest questions from dataset given question
    args:
        question: question, preprocessed using text_prepare
        k: how many nearest questions to find
    """

    vec_question = question_to_vec(question,wv_embeddings).reshape(1,-1)
    dist_s = cosine_similarity(quora_vectors_emb, vec_question)[:,0]
    sort_dist_s = sorted(dist_s)[::-1][:k]
    sorted_questions = deepcopy(np.array(quora_tokenized)[dist_s.argsort()[::-1]])[:k]
    sort_dict = dict(zip(sorted_questions,sort_dist_s))
    sorted_d = sorted(sort_dict.items(), key=operator.itemgetter(1),reverse = True)
    return sorted_d
```

In [40]:

```
find_closest_questions(text_prepare("why am I so stupid"),k=10)
```

Out[40]:

```
[('How do I be stupid?', 1.0000000000000002),
 ('How can I be so stupid?', 1.0000000000000002),
 ('Why am I stupid?', 1.0000000000000002),
 ('How not to be stupid?', 1.0000000000000002),
 ('Stupid is as stupid does?', 1.0000000000000002),
 ('Why is Quora so stupid?', 1.0000000000000002),
 ('Why Sinai is a stupid jewplicate of Ararat?', 1.0000000000000002),
 ('Why am I so stupid?', 1.0000000000000002),
 ('Why is Rahul Gandhi so stupid?', 1.0000000000000002),
 ('What does "stupid is as stupid does" mean?', 0.9466698134856456)]
```

Вопрос 10:

- **Какой самый ближайший вопрос к "Why am I so stupid?"? В канвас напишите слова вопроса с маленькой буквы через пробелы без знаков пунктуации (только латинские буквы)**

how do i be stupid