|  |  |
| --- | --- |
| Gerb-BMSTU_01 | **Министерство образования и науки Российской Федерации**  **Федеральное государственное бюджетное образовательное учреждение**  **высшего образования**  **«Московский государственный технический университет**  **имени Н.Э. Баумана**  **(национальный исследовательский университет)»**  **(МГТУ им. Н.Э. Баумана)** |

ФАКУЛЬТЕТ: Информатика и системы управления (ИУ)

КАФЕДРА: Системы обработки информации и управления (ИУ5)

Отчет по курсовой работе

“Предобработка текста и построение семантических эмбеддингов”

По курсу “Методы машинного обучения”

Выполнил студент:

Аршанов Глеб

Группа:

ИУ5-32М

*2021 г.*

Постановка задачи

### Формирование семантически близких кластеров для анализа новостной ленты с использованием FastText. Сделать тегирование. Провести пайплайн работы с текстом.

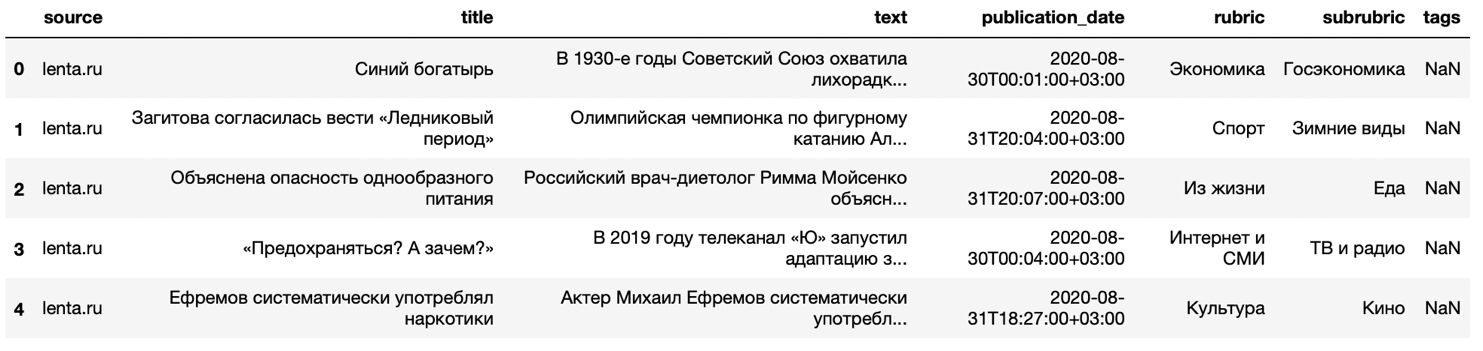
### Пайплайн

1. Предобработка
2. обучение модели,
3. создание эмбеддингов с помощью FastText,
4. определение оптимального количетва кластеров методом локтя,
5. кластеризация,
6. выделение топ слов в каждом кластере,
7. тестрование точности определения ключевых слов.

import pandas as pd  
import numpy as np  
import pymorphy2  
import re  
  
from sklearn.metrics import pairwise\_distances\_argmin\_min  
from sklearn.decomposition import PCA  
from sklearn.pipeline import Pipeline  
from sklearn.cluster import KMeans  
from sklearn import preprocessing  
  
from nltk.probability import FreqDist  
  
from gensim.models import FastText  
from matplotlib import pyplot  
from itertools import chain  
from tqdm import tqdm

## загрузка стопслов

with open('stopwords-ru.txt', encoding='utf-8') as f:  
 stopwords = f.read().splitlines()  
  
stopwords.append('%')  
stopwords.append('num')  
stopwords.append('d\_')  
stopwords.append('x')

df = pd.read\_csv('news.csv', sep=',', encoding='utf-8')  
df = df[~df.text.isna()]  
df.head()

def clean\_text(text):  
 text = text.lower()  
 regular = r'[^a-zа-яё]+'  
 regular\_url = r'(http\S+)|(www\S+)|([\w\d]+www\S+)|([\w\d]+http\S+)'  
 text = re.sub(regular, ' ', text)  
 text = re.sub(regular\_url, r'URL', text)  
 text = re.sub(r'(\d+\s\d+)|(\d+)',' num ', text)  
 text = re.sub(r'\s+', ' ', text)  
 return text  
  
def delete\_stopwords(string):  
 tokens = string.split()  
 tokens\_res = []  
 for i in tokens:  
 if i not in stopwords:  
 tokens\_res.append(i)  
 return ' '.join(tokens\_res)  
  
def check\_list(list\_):  
 if list\_:  
 return True  
 else: return False

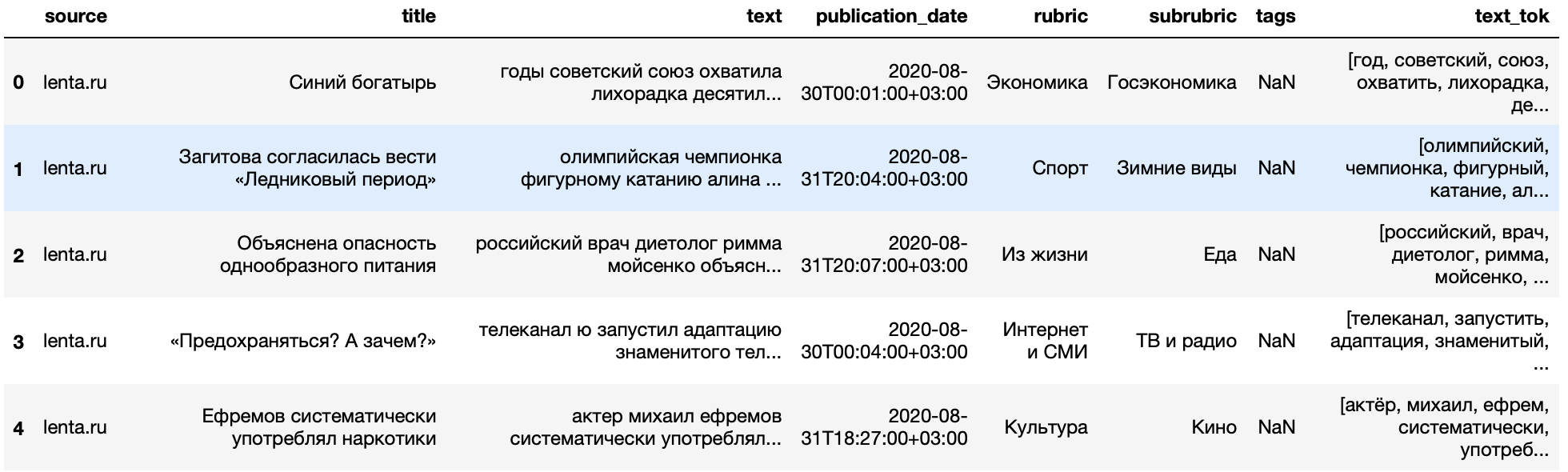
## предобработка

df['text'] = df['text'].apply(lambda x: delete\_stopwords(clean\_text(x)))

morph = pymorphy2.MorphAnalyzer()  
  
sentences = df['text'].to\_list()

%%time  
sentences\_tok = []  
  
for sent in sentences:  
 tokens\_sent = sent.lower().split()  
# tokens\_sent = [short\_dict.get(x,x) for x in tokens\_sent]  
 tokens\_sent = delete\_stopwords(clean\_text(' '.join(tokens\_sent))).split()  
 tokens\_sent = [word for word in tokens\_sent if len(word) > 3]  
 tokens = list([morph.parse(\_)[0].normal\_form for \_ in tokens\_sent])  
 sentences\_tok.append(tokens)  
   
df['text\_tok'] = sentences\_tok  
df = df[df.apply(lambda x: check\_list(x['text\_tok']), axis=1)]

Wall time: 11min 44s

df.head()

def check\_len(tokens):  
 tokens\_cleaned = []  
 for i in tokens:  
 if len(i) >= 3:  
 tokens\_cleaned.append(i)  
 return tokens\_cleaned

df.text\_tok = df.text\_tok.apply(lambda x: check\_len(x))  
df = df.drop\_duplicates(subset='text')

## обучение FastText

df = df[df.text\_tok.apply(lambda x: (len(x) > 2))]

df\_orign = df.copy()  
  
df = df.head(5000)  
df.shape

(5000, 8)

model = FastText(vector\_size=300, window=5, min\_count=2)  
model.build\_vocab(corpus\_iterable=df.text\_tok.to\_list())  
model.train(corpus\_iterable=df.text\_tok.to\_list(), total\_examples=len(df.text\_tok.to\_list()), epochs=10)

(6075228, 6301690)

model.wv.most\_similar('продук')

[('продукт', 0.9649566411972046),  
 ('продажа', 0.9047145247459412),  
 ('продавец', 0.8981866240501404),  
 ('пробежка', 0.8883215188980103),  
 ('прокачка', 0.8853464722633362),  
 ('продавщица', 0.882567822933197),  
 ('приморнефтепродукт', 0.8743192553520203),  
 ('просрочка', 0.8706531524658203),  
 ('продуктовый', 0.8704662919044495),  
 ('произвол', 0.8660559058189392)]

model.save('news.model')

## загрузка обученной модели

model = FastText.load('news.model')

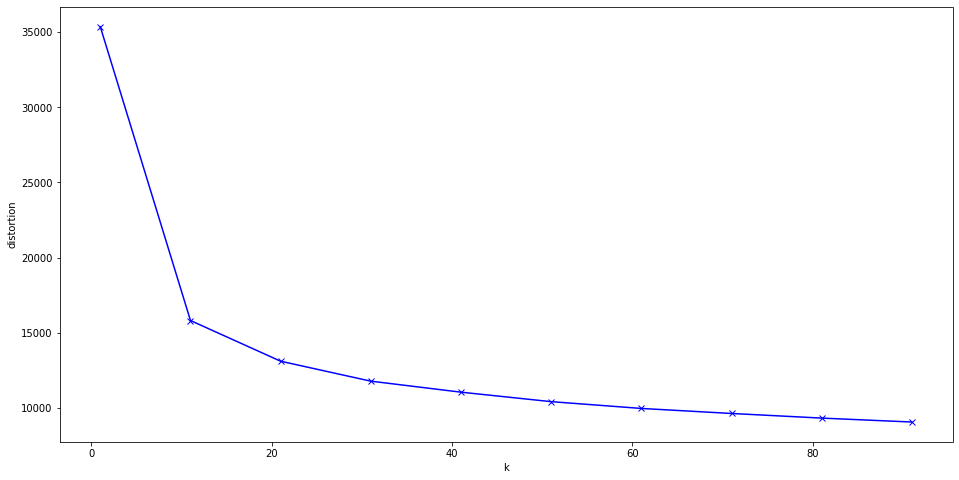
def word\_embadding\_fasttext(word):  
 res = model.wv.get\_vector(word)  
 if res is None: return np.zeros(300)  
 else: return res  
  
def sentence\_embadding\_fasttext(sent):  
 embedding = [word\_embadding\_fasttext(x) for x in sent]   
 embedding = np.mean(embedding, axis=0)  
 return embedding

sent\_emb = [sentence\_embadding\_fasttext(sent) for sent in df.text\_tok]

### определение оптимального количества кластеров

K\_n = range(1, 100, 10)  
  
distortions = []  
  
for k in tqdm(K\_n):  
 kmeans = KMeans(n\_clusters=k).fit(sent\_emb)  
 distortions.append(kmeans.inertia\_)  
  
pyplot.figure(figsize=(16,8))  
pyplot.plot(K\_n, distortions, 'bx-')  
pyplot.xlabel('k')  
pyplot.ylabel('distortion')  
pyplot.show()

100%|██████████| 10/10 [00:58<00:00, 5.84s/it]

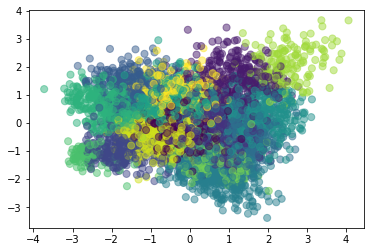


### кластеризация

kmeans = KMeans(n\_clusters=15).fit(sent\_emb)

pca = PCA(n\_components=2)  
result = pca.fit\_transform(sent\_emb)  
  
x = [x[0] for x in result]   
y = [y[1] for y in result]   
  
df['x'] = x  
df['y'] = y

pyplot.scatter(df['x'], df['y'], c=kmeans.labels\_.astype(float), s=50, alpha=0.5)  
pyplot.show()



df['label'] = kmeans.labels\_

pd.DataFrame(list(chain.from\_iterable(df[df.label == 7].text\_tok.to\_list()))).value\_counts().head()

коронавирус 886  
россия 508  
страна 454  
человек 379  
москва 327  
dtype: int64

# test

test\_orign = 'военные на границе с россией выставили вооружения'

# предобработка текста  
  
test = delete\_stopwords(clean\_text(test\_orign))  
test

'военные границе россией выставили вооружения'

# получение эмбединга текста  
  
test\_emb = sentence\_embadding\_fasttext([test])  
test\_emb[:4]

array([ 0.00619834, 0.08600322, -0.2625694 , 0.10302448], dtype=float32)

# определение кластера  
  
claster = kmeans.predict([test\_emb])[0]  
claster

11

# вывод топ 5 наиболее частых слов в кластере  
  
fdist = FreqDist(list(chain.from\_iterable(df[df.label == claster].text\_tok.to\_list())))  
words\_uniq\_claster = [i[0] for i in fdist.most\_common(5)]  
words\_uniq\_claster

['военный', 'россия', 'страна', 'российский', 'заявить']