Министерство науки и высшего образования 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, искусственный интеллект и системы управления ___

КАФЕДРА _____Системы обработки информации и управления_____

ОТЧЁТ
К ЛАБОРАТОРНОЙ РАБОТЕ №5

НА ТЕМУ:
Предобработка и классификация текста

Студент: Громоздов Д.Р.
Группа: ИУ5-23М

Преподаватель: Гапанюк Ю.Е.

2022 г.

Цель лабораторной работы: изучение методов предобработки и классификации текстовых данных.

Задание:

1. Для произвольного предложения или текста решите следующие задачи:
   - Токенизация.
   - Частеречная разметка.
   - Лемматизация.
   - Выделение (распознавание) именованных сущностей.
   - Разбор предложения.
2. Для произвольного набора данных, предназначенного для классификации текстов, решите задачу классификации текста двумя способами:
   - Способ 1. На основе CountVectorizer или TfidfVectorizer.
   - Способ 2. На основе моделей word2vec или Glove или fastText.
   - Сравните качество полученных моделей.

In [1]:
```
import pandas as pd
from nltk import tokenize
```

Загрузим датасет с классификацией записей в сети Твиттер и предполагемой тональностью их содержимого:

In [2]:
```
df_class = pd.read_csv('data/tweet_emotions.csv', sep=",")
df_class.head()
```

Out[2]:

	tweet_id	sentiment	content
0	1956967341	empty	@tiffanylue i know i was listenin to bad habi...
1	1956967666	sadness	Layin n bed with a headache ughhhh...waitin o...
2	1956967696	sadness	Funeral ceremony...gloomy friday...
3	1956967789	enthusiasm	wants to hang out with friends SOON!
4	1956968416	neutral	@dannycastillo We want to trade with someone w...

In [3]:
```
# выделим тестовое сообщение, с которым затем будем выполнять задачи предобработки текста
test_val = 100
texts = df_class['content']
test_text = texts.iloc[test_val]
test_text
```

Out[3]:'First ever dropped call on my mobile. On a call to @Telstra no less! (being charged for data even though I have a data pack)'

Предобработка текста

Токенизация

In [4]:
```
import nltk
nltk.download('punkt')
```

```
[nltk_data] Downloading package punkt to
[nltk_data]     C:\Users\Lenovo\AppData\Roaming\nltk_data...
[nltk_data]   Package punkt is already up-to-date!
```

Out[4]:True

Токенизация по предложениям:

In [7]:
```
nltk_tk_sents = nltk.tokenize.sent_tokenize(test_text)
print(len(nltk_tk_sents))
nltk_tk_sents
```

```
3
```

Out[7]:
```
['First ever dropped call on my mobile.',
 'On a call to @Telstra no less!',
 '( being charged for data even though I have a data pack )']
```

Токенизация по словам:

In [8]:
```
nltk_tk_1 = nltk.WordPunctTokenizer()
nltk_tk_1.tokenize(test_text)
```

Out[8]:
```
['First',
 'ever',
 'dropped',
 'call',
 'on',
 'my',
 'mobile',
 '.',
 'On',
 'a',
 'call',
 'to',
 '@',
 'Telstra',
 'no',
 'less',
 '!',
 '(',
 'being',
 'charged',
 'for',
 'data',
 'even',
 'though',
 'I',
 'have',
 'a',
 'data',
 'pack',
 ')']
```

Частеречная разметка

In [5]:
```
from spacy.lang.en import English
import spacy
nlp = spacy.load('en_core_web_sm')
spacy_test = nlp(test_text)
```

Просмотрим какие части речи присутсвуют в тестовом твите:

In [14]:
```
for token in spacy_test:
    print('{} - {} - {}'.format(token.text, token.pos_, token.dep_))
```

```
First - ADV - advmod
ever - ADV - advmod
dropped - VERB - ROOT
call - NOUN - dobj
on - ADP - prep
my - PRON - poss
mobile - NOUN - pobj
. - PUNCT - punct
On - ADP - prep
a - DET - det
call - NOUN - pobj
to - ADP - prep
@Telstra - PROPN - pobj
no - ADV - neg
less - ADJ - ROOT
! - PUNCT - punct
( - PUNCT - punct
being - AUX - auxpass
charged - VERB - ROOT
for - ADP - prep
data - NOUN - pobj
even - ADV - advmod
though - SCONJ - mark
I - PRON - nsubj
have - VERB - advcl
a - DET - det
data - NOUN - compound
pack - NOUN - dobj
  - SPACE - dep
) - PUNCT - punct
```

Лемматизация

In [15]:
```
for token in spacy_test:
    print(token, token.lemma, token.lemma_)
```

```
First 11860158879560853892 first
ever 6231102377460051108 ever
dropped 505665066430977685 drop
call 14229572451745258962 call
on 5640369432778651323 on
my 227504873216781231 my
mobile 13895322422246515550 mobile
. 12646065887601541794 .
On 5640369432778651323 on
a 11901859001352538922 a
call 14229572451745258962 call
to 3791531372978436496 to
@Telstra 14311364722520319565 @Telstra
no 13055779130471031426 no
less 589070940943333110 less
! 17494803046312582752 !
( 12638816674900267446 (
being 10382539506755952630 be
charged 16743499924604303818 charge
for 16037325823156266367 for
data 8931270445620108520 datum
even 17339226045912991082 even
though 16680099953739830072 though
I 4690420944186131903 I
have 14692702688101715474 have
a 11901859001352538922 a
data 66455066661261177361 data
pack 11929990034961539164 pack
  8532415787641010193
) 3842344029291005339 )
```

Выделение (распознавание) именованных сущностей

In [16]:
```
for ent in spacy_test.ents:
    print(ent.text, ent.label_)
```

```
First ORDINAL
@Telstra PRODUCT
```

In [17]:
```
print(spacy.explain("ORDINAL"))
```

```
"first", "second", etc.
```

In [18]:
```
print(spacy.explain("PRODUCT"))
```

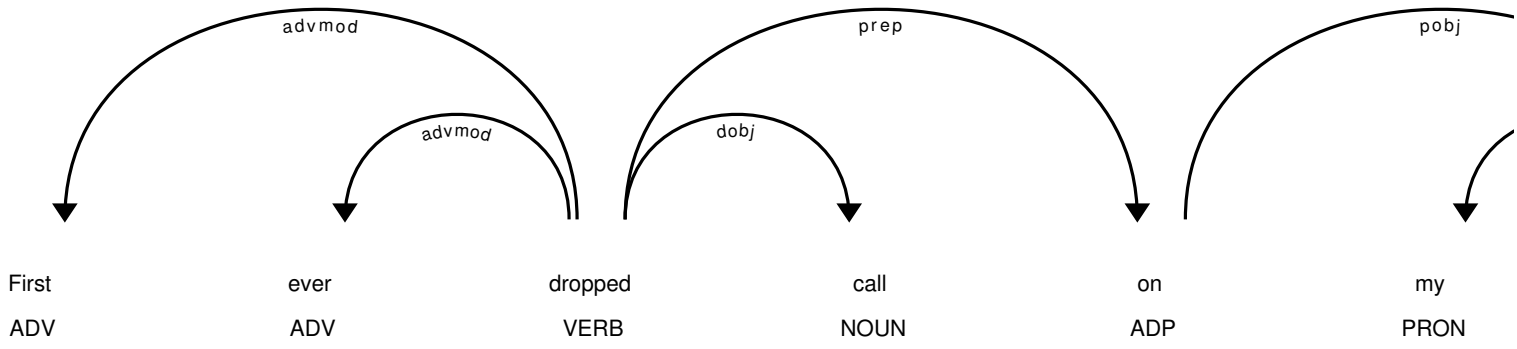
```
Objects, vehicles, foods, etc. (not services)
```

In [7]:
```
from spacy import displacy
displacy.render(spacy_test, style='ent', jupyter=True)
```

First ORDINAL ever dropped call on my mobile. On a call to @Telstra PRODUCT no less! (being charged for data even though I have a data pack)

Разбор предложения

In [8]:
```
displacy.render(spacy_test, style='dep', jupyter=True)
```



Решение задачи классификации текста

In [23]:
```
from sklearn.feature_extraction.text import CountVectorizer, TfidfVectorizer
from sklearn.svm import LinearSVC
from sklearn.naive_bayes import MultinomialNB
from sklearn.model_selection import cross_val_score
```

Зададим целевую переменную -- тональность текста:

In [21]:
```
target = df_class['sentiment'].values
target
```

```
Out[21]:array(['empty', 'sadness', 'sadness', ..., 'love', 'happiness', 'love'],
      dtype=object)
```

Способ 1. CountVectorizer

In [24]:
```
countv = CountVectorizer()
countv_features = countv.fit_transform(df_class["content"])
countv_features
```

```
Out[24]:<40000x48212 sparse matrix of type '<class 'numpy.int64'>'
    with 475946 stored elements in Compressed Sparse Row format>
```

In [26]:
```
%%time
score_count_svc = cross_val_score(LinearSVC(), countv_features, target, scoring='accuracy', cv=3).mean()

print('Модель векторизации - Countvectorizer, \nМодель классификации - LinearSVC, \n3начение accuracy = {}'.format(score_count_svc))
```

```
C:\Users\Lenovo\AppData\Local\Programs\Python\Python39\lib\site-packages\sklearn\svm\_base.py:1225: ConvergenceWarning: Liblinear failed to converge
, increase the number of iterations.
  warnings.warn(
C:\Users\Lenovo\AppData\Local\Programs\Python\Python39\lib\site-packages\sklearn\svm\_base.py:1225: ConvergenceWarning: Liblinear failed to converge
, increase the number of iterations.
  warnings.warn(
Модель векторизации - Countvectorizer,
Модель классификации - LinearSVC,
Значение accuracy = 0.285875415971945
CPU times: total: 1min 20s
Wall time: 1min 22s
```

Получаем достаточно плохой результат. Возможно это следствие особенности текстов в твиттере с ограничением на количество символов, что приводит к сильным сокращениям и искажениям слов. Вообще лексика неформального общения не совсем совпадает со стандартной.

Способ 2. word2vec

In [27]:
```
import gensim
from gensim.models import word2vec
```

In [28]:
```
import re
import pandas as pd
import numpy as np
from typing import Dict, Tuple
from sklearn.metrics import accuracy_score, balanced_accuracy_score
from sklearn.feature_extraction.text import CountVectorizer, TfidfVectorizer
from sklearn.linear_model import LogisticRegression
from sklearn.pipeline import Pipeline
from nltk import WordPunctTokenizer
from nltk.corpus import stopwords
import nltk
nltk.download('stopwords')
```

```
[nltk_data] Downloading package stopwords to
[nltk_data]     C:\Users\Lenovo\AppData\Roaming\nltk_data...
[nltk_data]   Unzipping corpora\stopwords.zip.
```

Out[28]:True

In [29]:
```
# Подготовим корпус
corpus = []
stop_words = stopwords.words('english')
tok = WordPunctTokenizer()
for line in df_class['content'].values:
    line1 = line.strip().lower()
    line1 = re.sub("[^a-zA-Z]"," ", line1)
    text_tok = tok.tokenize(line1)
    text_tok1 = [w for w in text_tok if not w in stop_words]
    corpus.append(text_tok1)
```

In [30]:
```
corpus[:5]
```

Out[30]:
```
[['tiffanylue',
  'know',
  'listenin',
  'bad',
  'habit',
  'earlier',
  'started',
  'freakin',
  'part'],
 ['layin', 'n', 'bed', 'headache', 'ughhhh', 'waitin', 'call'],
 ['funeral', 'ceremony', 'gloomy', 'friday'],
 ['wants', 'hang', 'friends', 'soon'],
 ['dannycastillo', 'want', 'trade', 'someone', 'houston', 'tickets', 'one']]
```

Обучаем модель word2vec на нашем корпусе

In [31]:
```
%time model_dz = word2vec.Word2Vec(corpus, workers=4, min_count=10, window=10, sample=1e-3)
```

```
CPU times: total: 3.72 s
Wall time: 1.89 s
```

In [33]:
```
# Проверим, что модель обучилась
print(model_dz.wv.most_similar(positive=['find'], topn=5))
```

```
[('think', 0.9772072434425354), ('thought', 0.9757692813873291), ('something', 0.9742587804794312), ('mean', 0.9721935987472534), ('anyone', 0.970889
0914916992)]
```

In [37]:
```
def sentiment(v, c):
    model = Pipeline(
        [("vectorizer", v),
         ("classifier", c)])
    model.fit(X_train, y_train)
    y_pred = model.predict(X_test)
    print_accuracy_score_for_classes(y_test, y_pred)
```

In [35]:
```
class EmbeddingVectorizer(object):
    '''
```

```
    Для текста усредним вектора входящих в него слов
    '''
    def __init__(self, model):
        self.model = model
        self.size = model.vector_size

    def fit(self, X, y):
        return self

    def transform(self, X):
        return np.array([np.mean(
            [self.model[w] for w in words if w in self.model]
            or [np.zeros(self.size)], axis=0)
            for words in X])
def accuracy_score_for_classes(
    y_true: np.ndarray,
    y_pred: np.ndarray) -> Dict[int, float]:
    """
    Вычисление метрики accuracy для каждого класса
    y_true - истинные значения классов
    y_pred - предсказанные значения классов
    Возвращает словарь: ключ - метка класса,
    значение - Accuracy для данного класса
    """
    # Для удобства фильтрации сформируем Pandas DataFrame
    d = {'t': y_true, 'p': y_pred}
    df = pd.DataFrame(data=d)
    # Метки классов
    classes = np.unique(y_true)
    # Результирующий словарь
    res = dict()
    # Перебор меток классов
    for c in classes:
        # отфильтруем данные, которые соответствуют
        # текущей метке класса в истинных значениях
        temp_data_flt = df[df['t']==c]
        # расчет accuracy для заданной метки класса
        temp_acc = accuracy_score(
            temp_data_flt['t'].values,
            temp_data_flt['p'].values)
        # сохранение результата в словарь
        res[c] = temp_acc
    return res

def print_accuracy_score_for_classes(
    y_true: np.ndarray,
    y_pred: np.ndarray):
    """
    Вывод метрики accuracy для каждого класса
    """
    accs = accuracy_score_for_classes(y_true, y_pred)
    if len(accs)>0:
        print('Метка \t Accuracy')
    for i in accs:
        print('{} \t {}'.format(i, accs[i]))
```

In [36]:
```
df_class.shape
```

Out[36]:(40000, 3)

In [41]:
```
df_class.head()
```

Out[41]:

	tweet_id	sentiment	content
0	1956967341	empty	@tiffanylue i know i was listenin to bad habi...
1	1956967666	sadness	Layin n bed with a headache ughhhh...waitin o...
2	1956967696	sadness	Funeral ceremony...gloomy friday...
3	1956967789	enthusiasm	wants to hang out with friends SOON!
4	1956968416	neutral	@dannycastillo We want to trade with someone w...

In [55]:
```
dz_df = pd.concat([df_class["content"], df_class["sentiment"]], axis = 1)
```

In [56]:
```
dz_df.head()
```

Out[56]:

	content	sentiment
0	@tiffanylue i know i was listenin to bad habi...	empty
1	Layin n bed with a headache ughhhh...waitin o...	sadness
2	Funeral ceremony...gloomy friday...	sadness
3	wants to hang out with friends SOON!	enthusiasm
4	@dannycastillo We want to trade with someone w...	neutral

In [59]:
```
# Обучающая и тестовая выборки
boundary = 1000
X_train = corpus[:boundary]
X_test = corpus[boundary:]
y_train = dz_df.sentiment.values[:boundary]
y_test = dz_df.sentiment.values[boundary:]
```

In [61]:
```
%%time
sentiment(EmbeddingVectorizer(model_dz.wv), LogisticRegression(C=5.0))
```

```
C:\Users\Lenovo\AppData\Local\Programs\Python\Python39\lib\site-packages\sklearn\linear_model\_logistic.py:444: ConvergenceWarning: lbfgs failed to con
verge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
```

```
Метка   Accuracy
anger   0.0
boredom   0.0
empty   0.0
enthusiasm   0.0
fun   0.0
happiness   0.02277992277992278
hate   0.0
love   0.0
neutral   0.33809353802213493
relief   0.0
sadness   0.2435454360642407
surprise   0.0009350163627863488
worry   0.5776526740912985
CPU times: total: 2.17 s
Wall time: 1.96 s
```

Результаты, полученные с помощью word2vec тоже не очень хоршие, скорее всего здесь нестандартность лексики ещё больше влияет на работу уже предобученной на более-менее формальных корпусах модели. Короткие неформальные сообщения скорее всего требуют немного других подходов.

In [ ]: