Лабораторная работа №5. Предобработка и классификация текста.

Студент: Кривцов Н.А.
Группа: ИУ5-22М

Часть 1

In [1]:

```
import spacy
from spacy.lang.en import English
```

In [2]:

```
nlp = spacy.load("en_core_web_sm")
```

In [3]:

```
text1 = """
Gloria, you're always on the run now.
Running after somebody, you've got to get him somehow.
I think you've got to slow down before you start to blow it.
I think you're headed for a breakdown, so be careful not to show it.
"""

text2 = """
Greta Garbo, and Monroe.
Dietrich and DiMaggio.
Marlon Brando, Jimmy Dean
On the cover of a magazine.
Grace Kelly, Harlow, Jean,
Picture of a beauty queen,
Gene Kelly, Fred Astaire,
Ginger Rogers dance on air.
They had style, they had grace,
Rita Hayworth gave good face.
Lauren, Katharine, Lana too.
Bette Davis, we love you!
Ladies with an attitude,
Fellas that were in the mood.
Don't just stand there, let's get to it,
Strike a pose, there's nothing to it,
Vogue!
"""
```

In [4]:

```
spacy_text1 = nlp(text1)
spacy_text2 = nlp(text2)
```

In [5]:

```
# Токенизация
for t in spacy_text1:
  print(t)
```

```
Gloria
,
you
're
always
```

```
on
the
run
now
.


Running
after
somebody
,
you
've
got
to
get
him
somehow
.


I
think
you
've
got
to
slow
down
before
you
start
to
blow
it
.


I
think
you
're
headed
for
a
breakdown
,
so
be
careful
not
to
show
it
.
```

In [6]:

```
# Разбор по частям речи
for t in spacy_text1:
    print(f'{t.text} - {t.pos_} - {t.dep_}')
```

```
 - SPACE -
Gloria - PROPN - npadvmod
, - PUNCT - punct
you - PRON - nsubj
're - AUX - ROOT
always - ADV - advmod
on - ADP - prep
the - DET - det
run - NOUN - pobj
now - ADV - advmod
. - PUNCT - punct
```

```
 - SPACE -
Running - VERB - advcl
after - ADP - prep
somebody - PRON - pobj
, - PUNCT - punct
you - PRON - nsubj
've - AUX - aux
got - VERB - ROOT
to - PART - aux
get - AUX - xcomp
him - PRON - dobj
somehow - ADV - advmod
. - PUNCT - punct

 - SPACE -
I - PRON - nsubj
think - VERB - ROOT
you - PRON - nsubj
've - AUX - aux
got - VERB - ccomp
to - PART - aux
slow - VERB - xcomp
down - ADP - prt
before - ADP - mark
you - PRON - nsubj
start - VERB - advcl
to - PART - aux
blow - VERB - xcomp
it - PRON - dobj
. - PUNCT - punct

 - SPACE -
I - PRON - nsubj
think - VERB - ROOT
you - PRON - nsubjpass
're - AUX - auxpass
headed - VERB - ccomp
for - ADP - prep
a - DET - det
breakdown - NOUN - pobj
, - PUNCT - punct
so - ADV - advmod
be - AUX - conj
careful - ADJ - acomp
not - PART - neg
to - PART - aux
show - VERB - xcomp
it - PRON - dobj
. - PUNCT - punct

 - SPACE -
```

In [7]:

```
# Лемматизация
for t in spacy_text1:
      print(t, t.lemma, t.lemma_)
```

```
 962983613142996970

Gloria 11337752517245580020 Gloria
, 2593208677638477497 ,
you 5612281913124630 89 -PRON-
're 10382539506755952630 be
always 17471638809377599778 always
on 5640369432778651323 on
the 7425985699627899538 the
run 12767647472892411841 run
now 17157488710739566268 now
. 12646065887601541794 .

 962983613142996970

Running 12767647472892411841 run
```

```
Running 12707647472692411641 fun
after 13428508259213873547 after
somebody 10432936886633602915 somebody
, 2593208677638477497 ,
you 561228191312463089 -PRON-
've 14692702688101715474 have
got 2013399242189103424 get
to 3791531372978436496 to
get 2013399242189103424 get
him 561228191312463089 -PRON-
somehow 6325893992711880729 somehow
. 12646065887601541794 .

 962983613142996970

I 561228191312463089 -PRON-
think 16875814820671380748 think
you 561228191312463089 -PRON-
've 14692702688101715474 have
got 2013399242189103424 get
to 3791531372978436496 to
slow 6002275471848253686 slow
down 6421409113692203669 down
before 11320251846592927908 before
you 561228191312463089 -PRON-
start 6480458294193393462 start
to 3791531372978436496 to
blow 17041142777417009832 blow
it 561228191312463089 -PRON-
. 12646065887601541794 .

 962983613142996970

I 561228191312463089 -PRON-
think 16875814820671380748 think
you 561228191312463089 -PRON-
're 10382539506755952630 be
headed 8419699711262724776 head
for 16037325823156266367 for
a 11901859001352538922 a
breakdown 17007358339637182149 breakdown
, 2593208677638477497 ,
so 9781598966686434415 so
be 10382539506755952630 be
careful 13557793935247935909 careful
not 447765159362469301 not
to 3791531372978436496 to
show 1916734850589852068 show
it 561228191312463089 -PRON-
. 12646065887601541794 .

 962983613142996970
```

In [8]:

```
# Поиск именованных сущностей
for ent in spacy_text2.ents:
  print(ent.text, ent.label_)
```

```
Greta Garbo ORG
Monroe GPE
Dietrich PERSON
DiMaggio PERSON
Marlon Brando PERSON
Jimmy Dean PERSON
Grace Kelly PERSON
Harlow GPE
Jean GPE
Gene Kelly PERSON
Fred Astaire PERSON
Ginger Rogers PERSON
Rita Hayworth PERSON
Lauren PERSON
Katharine PERSON
```

```
Lana PERSON
Bette Davis PERSON
```

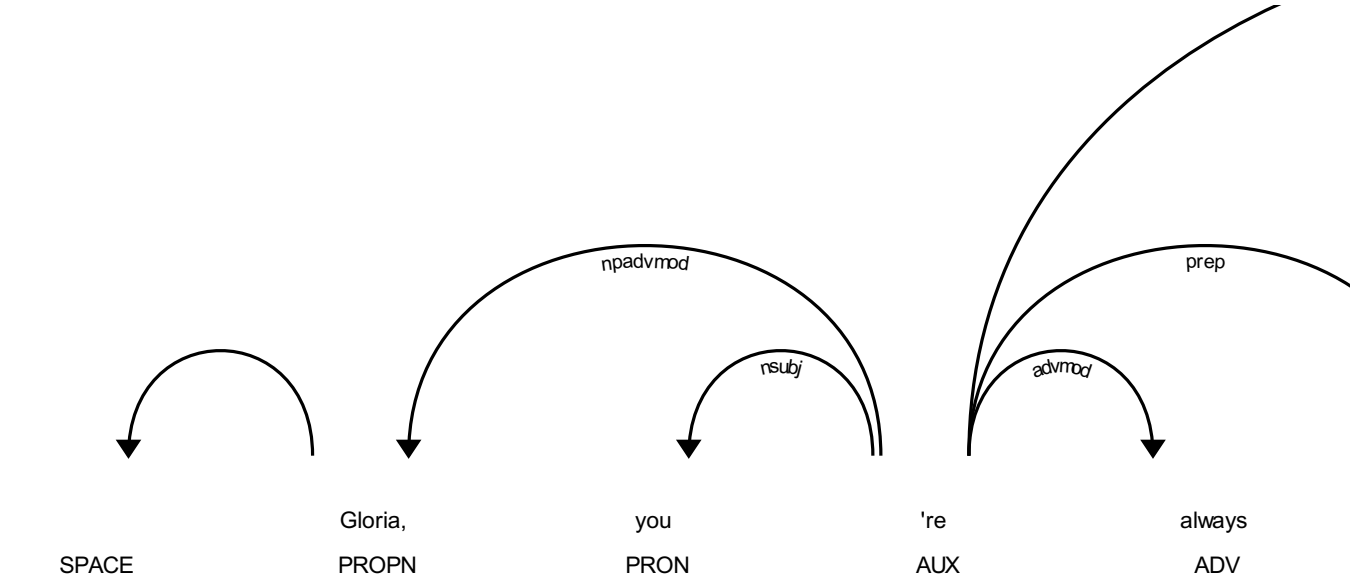
In [9]:

```
from spacy import displacy
displacy.render(spacy_text2, style='ent', jupyter=True)
```

Greta Garbo ORG , and Monroe GPE .
Dietrich PERSON and DiMaggio PERSON .
Marlon Brando PERSON , Jimmy Dean PERSON
On the cover of a magazine.
Grace Kelly PERSON , Harlow GPE , Jean GPE ,
Picture of a beauty queen,
Gene Kelly PERSON , Fred Astaire PERSON ,
Ginger Rogers PERSON dance on air.
They had style, they had grace,
Rita Hayworth PERSON gave good face.
Lauren PERSON , Katharine PERSON , Lana PERSON too.
Bette Davis PERSON , we love you! Ladies with an attitude, Fellas that were in the mood. Don't just stand there, let's get to it,
Strike a pose, there's nothing to it, Vogue!

In [10]:

```
# Разбор предложения
displacy.render(spacy_text1, style='dep', jupyter=True)
```



Часть 2

In [46]:

```
import pandas as pd
import numpy as np
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score, balanced_accuracy_score, f1_score
from sklearn.neighbors import KNeighborsClassifier
import gensim
from gensim.models import word2vec
```

In [37]:

```
data = pd.read_csv('/content/drive/MyDrive/MMO/wiki_movie_plots_deduped.csv')
df = data[['Genre', 'Plot']]
df = df[(df.Genre == 'drama') | (df.Genre =='comedy')]
df.head()
```

Out[37]:

	Genre	Plot
7	comedy	The film is about a family who move to the sub...
14	comedy	Before heading out to a baseball game at a nea...
15	comedy	The plot is that of a black woman going to the...
16	drama	On a beautiful summer day a father and mother ...
17	drama	A thug accosts a girl as she leaves her workpl...

In [38]:

```
df.Genre.value_counts(normalize=True)
```

Out[38]:

```
drama     0.576622
comedy    0.423378
Name: Genre, dtype: float64
```

In [39]:

```
Y_all = df['Genre']
X_all = df['Plot']
```

In [40]:

```
X_train, X_test, Y_train, Y_test = train_test_split(X_all, Y_all, test_size=0.2, random_state=1102)
```

In [66]:

```
from sklearn.pipeline import Pipeline
def genre(v, c):
    model = Pipeline(
        [("vectorizer", v),
         ("classifier", c)])
    assert len(X_train) == len(Y_train)
    model.fit(X_train, Y_train)
    Y_pred = model.predict(X_test)
    print_class_metrics(Y_test, Y_pred)
```

In [70]:

```
def class_metrics(y_true, y_pred):
    # 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
        temp_f1 = f1_score(
            y_true, y_pred, pos_label=c
        )
        res[c] = (temp_acc, temp_f1)
    return res

def print_class_metrics(y_true, y_pred):
    accs = class_metrics(y_true, y_pred)
    print(pd.DataFrame.from_dict(accs, orient='index', columns=['Accuracy', 'F1']))
```

In [77]:

```
from tqdm import tqdm

corpus = []
for line in tqdm(X_all):
  line1 = line.lower()
  tokens = nlp(line1)
  new_sentence = [t.text for t in tokens if not t.is_stop and not t.is_punct]
  corpus.append(new_sentence)
```

```
100%|██████████| 10343/10343 [11:38<00:00, 14.80it/s]
```

In [78]:

```
corpus[:5]
```

Out[78]:

```
[['film',
  'family',
  'suburbs',
  'hoping',
  'quiet',
  'life',
  'things',
  'start',
  'wrong',
  'wife',
  'gets',
  'violent',
  'starts',
  'throwing',
  'crockery',
  'leading',
  'arrest'],
 ['heading',
  'baseball',
  'game',
  'nearby',
  'ballpark',
  'sports',
  'fan',
  'mr',
  'brown',
  'drinks',
  'highball',
  'cocktails',
  'arrives',
  'ballpark',
  'watch',
  'game',
  'inebriated',
  'game',
  'appears',
  'reverse',
  'players',
  'running',
  'bases',
  'backwards',
  'baseball',
  'flying',
  'pitcher',
  'hand',
  'game',
  'mr',
  'brown',
  'escorted',
  'home',
  'friends',
  'arrive',
  'brown',
  'house',
  'encounter',
  'wife',
  'furious',
  'friend',
  'proceeds',
  'physically',
  'assault',
  'believing',
  'responsible',
  'husband',
  'severe',
  'intoxication.[1'],
 ['plot',
  'black',
  'woman',
  'going',
  'dentist',
  'toothache',
  'given',
  'laughing',
  'gas',
```

```
 'way',
 'walking',
 'home',
 'situations',
 'stop',
 'laughing',
 'meets',
 'catches',
 'laughter',
 'including',
 'vendor',
 'police',
 'officers'],
['beautiful',
 'summer',
 'day',
 'father',
 'mother',
 'daughter',
 'dollie',
 'outing',
 'river',
 'mother',
 'refuses',
 'buy',
 'gypsy',
 'wares',
 'gypsy',
 'tries',
 'rob',
 'mother',
 'father',
 'drives',
 'gypsy',
 'returns',
 'camp',
 'devises',
 'plan',
 'return',
 'kidnap',
 'dollie',
 'parents',
 'distracted',
 'rescue',
 'crew',
 'organized',
 'gypsy',
 'takes',
 'dollie',
 'camp',
 'gag',
 'dollie',
 'hide',
 'barrel',
 'rescue',
 'party',
 'gets',
 'camp',
 'leave',
 'gypsies',
 'escapes',
 'wagon',
 'wagon',
 'crosses',
 'river',
 'barrel',
 'falls',
 'water',
 'sealed',
 'barrel',
 'dollie',
 'swept',
 'downstream',
 'dangerous',
 'currents',
 'boy',
 'fishing',
```

```
  'river',
  'finds',
  'barrel',
  'dollie',
  'reunited',
  'safely',
  'parents'],
 ['thug',
  'accosts',
  'girl',
  'leaves',
  'workplace',
  'man',
  'rescues',
  'thug',
  'vows',
  'revenge',
  'help',
  'friends',
  'attacks',
  'girl',
  'rescuer',
  'going',
  'walk',
  'time',
  'succeed',
  'kidnapping',
  'rescuer',
  'bound',
  'gagged',
  'taken',
  'away',
  'cart',
  'girl',
  'runs',
  'home',
  'gets',
  'help',
  'neighbors',
  'track',
  'ruffians',
  'cabin',
  'mountains',
  'gang',
  'trapped',
  'victim',
  'set',
  'cabin',
  'fire',
  'thug',
  'rescuer',
  'fight',
  'roof',
  'house']]
```

In [79]:

```
assert len(corpus) == X_all.shape[0]
```

In [81]:

```
%time model_wiki = word2vec.Word2Vec(corpus, workers=4, min_count=10, window=10, sample=1e-3)
```

```
CPU times: user 38.8 s, sys: 227 ms, total: 39 s
Wall time: 22.3 s
```

In [84]:

```
print(model_wiki.wv.most_similar(positive=['daughter'], topn=5))
```

```
[('daughters', 0.9014550447463989), ('rao', 0.8769866228103638), ('sister', 0.869405210018158), ('aunt'
```

, 0.8541515469551086), ('sons', 0.8471763134002686)]

In [85]:

```
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])
```

In [93]:

```
print('Count Vectorizer')
genre(CountVectorizer(analyzer='word', stop_words='english'), KNeighborsClassifier(n_neighbors=7))
```

```
Count Vectorizer
        Accuracy        F1
comedy  0.376190  0.426739
drama   0.735557  0.680467
```

In [94]:

```
print('Word2Vec')
genre(EmbeddingVectorizer(model_wiki.wv), KNeighborsClassifier(n_neighbors=7))
```

```
Word2Vec
        Accuracy        F1
comedy  0.401190  0.426852
drama   0.672905  0.646346
```

`CountVectorizer` показал лучшую точность и F1-метрику в распознавании класса `drama`, а `Word2Vec` -- класса `comedy`.