Московский государственный технический университет им. Н.Э. Баумана
Кафедра «Системы обработки информации и управления»

Лабораторная работа №5
по дисциплине
«Методы машинного обучения»
на тему

«Предобработка и классификация текстовых данных.»

Выполнил:
студент группы ИУ5-21М
Хуан Яовэнь

Москва — 2022 г.

1. Цель лабораторной работы:¶

изучение методов предобработки и классификации текстовых данных.

2. 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.

3. Ход выполнения работы

3.1 Для произвольного предложения или текста решите следующие задачи

В этой части выбрана библиотека **spacy** (на сегодняшний день одна из наиболее развитых библиотек для обработки естественного языка, в том числе ориентирована на русский язык.).

3.1.1 Токенизация

Токенизация, также известная как лексический анализ, представляет собой процесс преобразования символов в токены (строки со связанными идентифицирующими значениями).

In [1]:

```
text1='Я из Китая и в настоящее время учусь на магистра на
text2='我来自中国，现在在莫斯科国立鲍曼技术大学信息处理专业攻读硕士学位。如果有可能，之后我想继续攻
```

In [2]:

```
from spacy.lang.ru import Russian
from spacy.lang.zh import Chinese
import spacy
nlp_ru = spacy.load('ru_core_news_sm')
nlp_zh = spacy.load('zh_core_web_sm')
spacy_text1 = nlp_ru(text1)
spacy_text1
```

Out[2]:

```
Я из Китая и в настоящее время учусь на магистра на факультете ИУ5 в МГТУ имени Баумана. Если возможно, я хотел бы продолжить учиться на кандидата.
```

In [3]:

```
spacy_text2=nlp_zh(text2)
spacy_text2
```

Out[3]:

```
我来自中国，现在在莫斯科国立鲍曼技术大学信息处理专业攻读硕士学位。如果有可能，之后我想继续攻读副博士学位。
```

In [4]:

```
for t1 in spacy_text1:
    print(t1)
```

```
Я
и з
К и т а я
и
в
н а с т о я щ е е
в р е м я
у ч у с ь
н а
м а г и с т р а
н а
ф а к у л ь т е т е
И У 5
в
М Г Т У
и м е н и
Б а у м а н а
.
Е с л и
в о з м о ж н о
,
я
х о т е л
б ы
п р о д о л ж и т ь
у ч и т ь с я
н а
к а н д и д а т а
.
```

In [5]:

```
for t2 in spacy_text2:
    print(t2)
```

```
我
来自
中国
，
现在
在
莫斯科
国立
鲍曼
技术
大学
信息
处理
专业
攻读
硕士
学位
。
如果
有
可能
，
之后
我
想
继续
攻读
副博士
学位
。
```

3.1.2 Частеречная разметка

В библиотеке **spacy** вначале выполняется частеречная разметка, а далее на ее основе выполняется лемматизация.

In [6]:

```
for token in spacy_text1:
    print('{} - {} - {}'.format(token.text, token.pos_, token.dep_))
```

```
Я - PRON - nsubj
и з - ADP - case
К и т а я - PROPN - nmod
и - CCONJ - cc
в - ADV - advmod
н а с т о я щ е е - ADJ - fixed
в р е м я - NOUN - fixed
у ч у с ь - VERB - ROOT
н а - ADP - case
м а г и с т р а - NOUN - obl
н а - ADP - case
ф а к у л ь т е т е - NOUN - obl
И У 5 - PROPN - appos
в - ADP - case
М Г Т У - PROPN - nmod
и м е н и - NOUN - nmod
Б а у м а н а - PROPN - nmod
. - PUNCT - punct
Е с л и - SCONJ - mark
в о з м о ж н о - ADV - parataxis
, - PUNCT - punct
я - PRON - nsubj
х о т е л - VERB - ROOT
б ы - AUX - aux
п р о д о л ж и т ь - VERB - xcomp
у ч и т ь с я - VERB - xcomp
н а - ADP - case
к а н д и д а т а - NOUN - obl
. - PUNCT - punct
```

In [7]:

```
for token in spacy_text2:
    print('{} - {} - {}'.format(token.text, token.pos_, token.dep_))
```

```
我 - PRON - nsubj
来自 - VERB - ROOT
中国 - PROPN - dobj
， - PUNCT - punct
现在 - NOUN - nmod:tmod
在 - ADP - case
莫斯科 - PROPN - nmod
国立 - ADJ - amod
鲍曼 - NOUN - compound:nn
技术 - NOUN - compound:nn
大学 - NOUN - compound:nn
信息 - NOUN - nsubj
处理 - VERB - nmod:prep
专业 - NOUN - dobj
攻读 - VERB - conj
硕士 - NOUN - compound:nn
学位 - NOUN - dobj
。 - PUNCT - punct
如果 - SCONJ - advmod
有 - VERB - dep
可能 - NOUN - dobj
， - PUNCT - punct
之后 - ADV - advmod
我 - PRON - nsubj
想 - VERB - ROOT
继续 - VERB - xcomp
攻读 - VERB - ccomp
副博士 - NOUN - compound:nn
学位 - NOUN - dobj
。 - PUNCT - punct
```

3.1.3 Лемматизация

Для китайской обработки нет лемматизации.Сложность обработки китайского естественного языка заключается в том, как сегментировать слова.

In [8]:

```
for token in spacy_text1:
      print(token, token.lemma, token.lemma_)
```

```
Я 12329462409828574123 я
из 12183146372738139588 и з
Китая 3745723740408540450 китай
и 15015917632809974589 и
в 15939375860797385675 в
настоящее 6875340309399359805 настоящий
время 14199711609533390218 время
учусь 1047322276092066949 учусь
на 16191904166009283104 на
магистра 6634360558365403035 магистр
на 16191904166009283104 на
факультете 2457577836754492043 факультет
ИУ5 13613433528103112282 иу5
в 15939375860797385675 в
МГТУ 18293469896216851415 мгту
имени 12140660392279389762 имя
Баумана 13062735427812331448 бауман
. 12646065887601541794 .
Если 9050364550658876467 если
возможно 5895594829886689258 возможно
, 2593208677638477497 ,
я 12329462409828574123 я
хотел 14604981939338786408 хотеть
бы 5478479325712016621 бы
продолжить 71995555555813095695 продолжить
учиться 11502212003802204076 учиться
на 16191904166009283104 на
кандидата 6678879842673263593 кандидат
. 12646065887601541794 .
```

3.1.4 Выделение (распознавание) именованных сущностей

In [9]:

```
for ent in spacy_text1.ents:
    print(ent.text, ent.label_)
```

```
Китая LOC
ИУ5 ORG
МГТУ имени Баумана ORG
```

In [10]:

```
from spacy import displacy
displacy.render(spacy_text1, style='ent', jupyter=True)
```

Я из Китая **LOC** и в настоящее время учусь на магистра на факультете ИУ5 **ORG** в МГТУ имени Баумана **ORG** . Если возможно, я хотел бы продолжить учиться на кандидата.

In [11]:

```
for ent in spacy_text2.ents:
    print(ent.text, ent.label_)
```

```
中国 GPE
莫斯科 GPE
鲍曼技术大学信息 ORG
```

In [12]:

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

我来自 中国 **GPE** ，现在在 莫斯科 **GPE** 国立 鲍曼技术大学信息 **ORG** 处理专业攻读硕士学位。如果有可能，之后我想继续攻读副博士学位。

Здесь проблема с распознаванием китайсого языка. Правильный ORG - 莫斯科国立鲍曼技术大学(МГТУ имени Баумана). Здесть идентифицируется как Москва и университет, есть две части.

In [13]:

```
print(spacy.explain("LOC"))
```

```
Non-GPE locations, mountain ranges, bodies of water
```

In [14]:

```
print(spacy.explain("GPE"))
```

```
Countries, cities, states
```

In [15]:

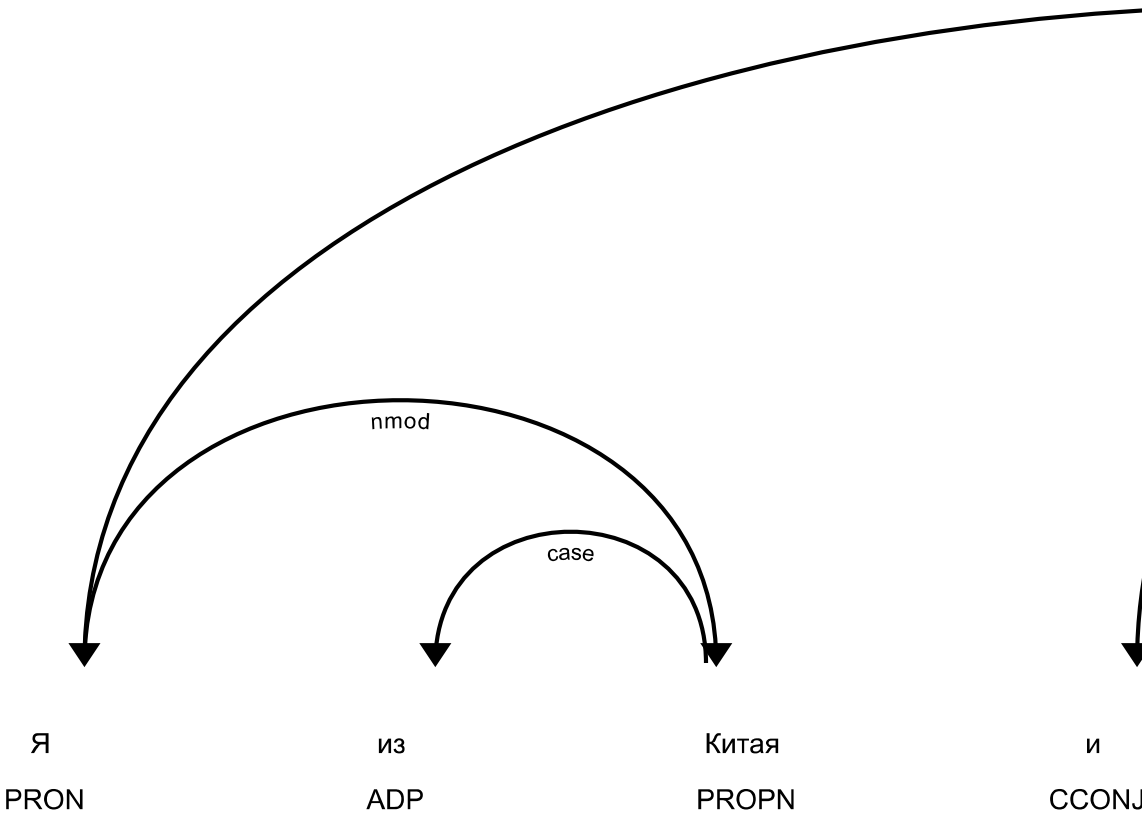
```
print(spacy.explain("ORG"))
```

```
Companies, agencies, institutions, etc.
```

3.1.5 Разбор предложения

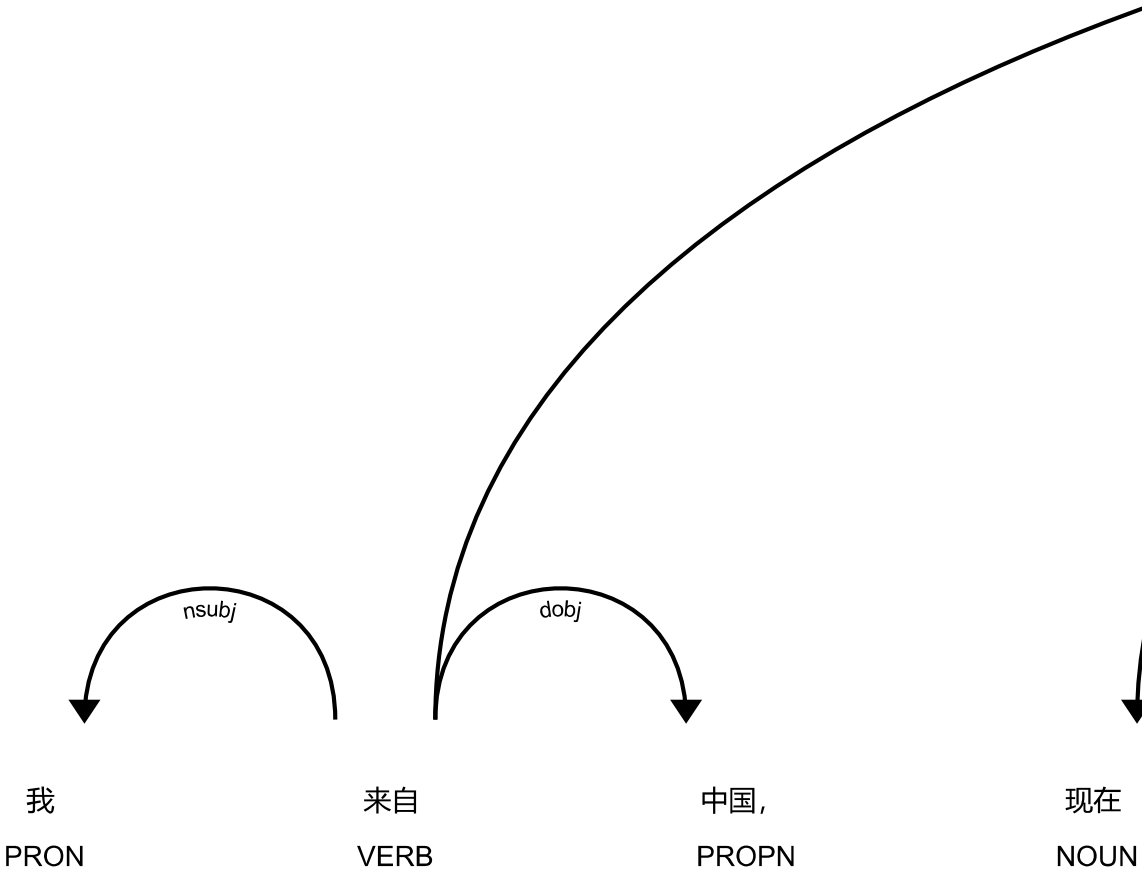
In [16]:

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



In [17]:

```
displacy.render(spacy_text2, style='dep', jupyter=True)
```



3.2 Для произвольного набора данных, предназначенного для классификации текстов, решите задачу классификации текста двумя способами

3.2.1 Способ 1. На основе CountVectorizer или TfidfVectorizer.

Здесь выбран CountVectorizer

In [18]:

```
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.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.naive_bayes import MultinomialNB
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\23882\AppData\Roaming\nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
```

Out[18]:

```
True
```

In [19]:

```
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]))
```

Работа с наборами данных"Amazon Alexa Reviews" из Kaggle.
About the Data

This dataset consists of a nearly 3000 Amazon customer reviews (input text), star ratings, date of review, variant and feedback of various amazon Alexa products like Alexa Echo, Echo dots, Alexa Firesticks etc. for learning how to train Machine for sentiment analysis.

In [20]:

```
amazon_rev=pd.read_csv("amazon_alexa.tsv", sep='\t')
amazon_rev.head()
```

Out[20]:

	rating	date	variation	verified_reviews	feedback
0	5	31-Jul-18	Charcoal Fabric	Love my Echo!	1
1	5	31-Jul-18	Charcoal Fabric	Loved it!	1
2	4	31-Jul-18	Walnut Finish	Sometimes while playing a game, you can answer...	1
3	5	31-Jul-18	Charcoal Fabric	I have had a lot of fun with this thing. My 4 ...	1
4	5	31-Jul-18	Charcoal Fabric	Music	1

In [21]:

```
#Только держать колонки "verified_reviews" и "feedback".
amazon_df = pd.DataFrame(amazon_rev,columns=['verified_reviews','feedback'])
amazon_df.columns = ['text','value']
amazon_df.head()
```

Out[21]:

	text	value
0	Love my Echo!	1
1	Loved it!	1
2	Sometimes while playing a game, you can answer...	1
3	I have had a lot of fun with this thing. My 4 ...	1
4	Music	1

In [22]:

```
amazon_df.shape
```

Out[22]:

```
(3150, 2)
```

In [23]:

```
#Сформируем общий словарь
vocab_list = amazon_df['text'].tolist()
vocab_list[1:10]
```

Out[23]:

```
['Loved it!',
 'Sometimes while playing a game, you can answer a question correctly but Alexa says
you got it wrong and answers the same as you.  I like being able to turn lights on a
nd off while away from home.',
 'I have had a lot of fun with this thing. My 4 yr old learns about dinosaurs, i con
trol the lights and play games like categories. Has nice sound when playing music as
well.',
 'Music',
 'I received the echo as a gift. I needed another Bluetooth or something to play mus
ic easily accessible, and found this smart speaker. Can't wait to see what else it
can do.',
 'Without having a cellphone, I cannot use many of her features. I have an iPad but
do not see that of any use.  It IS a great alarm.  If u r almost deaf, you can hear
her alarm in the bedroom from out in the living room, so that is reason enough to ke
ep her.It is fun to ask random questions to hear her response.  She does not seem to
be very smartbon politics yet.',
 "I think this is the 5th one I've purchased. I'm working on getting one in every ro
om of my house. I really like what features they offer specifily playing music on al
l Echos and controlling the lights throughout my house.",
 'looks great',
 'Love it! I've listened to songs I haven't heard since childhood! I get the news,
weather, information! It's great!']
```

In [24]:

```
vocabVect = CountVectorizer(
    stop_words='english',
    ngram_range=(1, 1),  #ngram_range=(1, 1) is the default
    dtype='double'
)
vocabVect.fit(vocab_list)
corpusVocab = vocabVect.vocabulary_
print('Количество сформированных признаков - {}'.format(len(corpusV
```

Количество сформированных признаков - 3784

In [25]:

```
for i in list(corpusVocab)[1:10]:
    print('{}={}'.format(i, corpusVocab[i]))
```

```
echo=1095
loved=2030
playing=2471
game=1413
answer=236
question=2623
correctly=796
alexa=190
says=2885
```

Подсчитывает количество слов словаря, входящих в данный текст.

In [26]:

```
test_features = vocabVect.transform(vocab_list)
```

In [27]:

```
test_features
```

Out[27]:

```
<3150x3784 sparse matrix of type '<class 'numpy.float64'>'
        with 33005 stored elements in Compressed Sparse Row format>
```

In [28]:

```
test_features.todense()
```

Out[28]:

```
matrix([[0., 0., 0., ..., 0., 0., 0.],
        [0., 0., 0., ..., 0., 0., 0.],
        [0., 0., 0., ..., 0., 0., 0.],
        ...,
        [0., 0., 0., ..., 0., 0., 0.],
        [0., 0., 0., ..., 0., 0., 0.],
        [0., 0., 0., ..., 0., 0., 0.]])
```

In [29]:

```
# Размер нулевой строки
len(test_features.todense()[0].getA1())
```

Out[29]:

```
3784
```

In [30]:

```
# Непустые значения нулевой строки
[i for i in test_features.todense()[0].getA1() if i>0]
```

Out[30]:

```
[1.0, 1.0]
```

In [31]:

```
vocabVect.get_feature_names()[100:120]
```

Out[31]:

```
['account',
 'accounts',
 'accuracy',
 'accurate',
 'accurately',
 'accustom',
 'acknowledge',
 'acoustical',
 'act',
 'acting',
 'action',
 'actions',
 'activate',
 'activated',
 'activates',
 'activating',
 'activation',
 'actively',
 'activities',
 'acts']
```

Разделим выборку на обучающую и тестовую

In [32]:

```
X_train, X_test, y_train, y_test = train_test_split(amazon_df['text'], amazon_df['value'], test_size
```

In [33]:

```
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)
```

Используем классификатор "LogisticRegression"

In [34]:

```
sentiment(CountVectorizer(), LogisticRegression(C=3.0))
```

Метка	Accuracy
0	0.5068493150684932
1	0.9873853211009175

Достаточно хороший результат

3.2.2 Способ 2. На основе моделей word2vec или Glove или fastText.

Здесь выбран word2vec.

In [35]:

```
import gensim
from gensim.models import word2vec
```

In [36]:

```
# Подготовим корпус
corpus = []
stop_words = stopwords.words('english')
tok = WordPunctTokenizer()
for line in amazon_df['text'].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 [37]:

```
corpus[:5]
```

Out[37]:

```
[['love', 'echo'],
 ['loved'],
 ['sometimes',
  'playing',
  'game',
  'answer',
  'question',
  'correctly',
  'alexa',
  'says',
  'got',
  'wrong',
  'answers',
  'like',
  'able',
  'turn',
  'lights',
  'away',
  'home'],
 ['lot',
  'fun',
  'thing',
  'yr',
  'old',
  'learns',
  'dinosaurs',
  'control',
  'lights',
  'play',
  'games',
  'like',
  'categories',
  'nice',
  'sound',
  'playing',
  'music',
  'well'],
 ['music']]
```

In [38]:

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

In [39]:

```
%time model_amazon = word2vec.Word2Vec(corpus, workers=4, min_count=10, window=15, sample=1e-3)
```

```
Wall time: 128 ms
```

In [40]:

```
# Проверим, что модель обучилась
print(model_amazon.wv.most_similar(positive=['terrible'], topn=10))
```

```
[('well', 0.9990426898002625), ('wifi', 0.9990378022193909), ('listening', 0.9990234
971046448), ('hear', 0.9990155696868896), ('using', 0.9990053176879883), ('item', 0.
999001145362854), ('going', 0.9990003705024719), ('love', 0.9990001916885376), ('wit
hout', 0.9989951252937317), ('much', 0.9989670515060425)]
```

In [41]:

```
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 [42]:

```
sentiment(EmbeddingVectorizer(model_amazon.wv), LogisticRegression(C=3.0))
```

```
Метка       Accuracy
0        0.0
1        1.0
```

In [43]:

```
sentiment(EmbeddingVectorizer(model_amazon.wv), LogisticRegression(C=5.0))
```

```
Метка       Accuracy
0        0.0
1        1.0
```

In [44]:

```
from sklearn.neighbors import KNeighborsClassifier
sentiment(EmbeddingVectorizer(model_amazon.wv), KNeighborsClassifier(n_neighbors=5))
```

```
Метка       Accuracy
0        0.0136986301369863
1        0.9988532110091743
```

Очень плохие результаты, независимо от того, как параметры изменены, результаты одинаковы. Классификатор заменили, а результаты особо не изменились. Модель word2vec может считать все в

наборе данных положительным.Такие результаты также связаны с набором данных. В наборе данных очень мало данных со значением = 0. Всего в наборе данных 3150 строк, из которых только 257 имеют значение 0.

In [45]:

```
amazon_df[amazon_df['value']==0]
```

Out[45]:

	text	value
46	It's like Siri, in fact, Siri answers more acc...	0
111	Sound is terrible if u want good music too get...	0
141	Not much features.	0
162	Stopped working after 2 weeks ,didn't follow c...	0
176	Sad joke. Worthless.	0
...	...	...
3047	Echo Dot responds to us when we aren't even ta...	0
3048	NOT CONNECTED TO MY PHONE PLAYLIST :(	0
3067	The only negative we have on this product is t...	0
3091	I didn’t order it	0
3096	The product sounded the same as the emoji spea...	0

257 rows × 2 columns

In [ ]:

```
```