

# Введение в обработку естественного языка

## Урок 1. Предобработка текста

### Практическое задание

#### Домашнее задание к уроку 1

Осуществим предобработку данных с Твиттера, чтобы отчищенный данные в дальнейшем использовать для задачи классификации. Данный датасет содержит негативные (label = 1) и нейтральные (label = 0) высказывания. Для работы объединим train\_df и test\_df.

Задания:

1) Удалим @user из всех твитов с помощью паттерна "@[\w]\*". Для этого создадим функцию:

для того, чтобы найти все вхождения паттерна в тексте, необходимо использовать re.findall(pattern, input\_txt) для замены @user на пробел, необходимо использовать re.sub()

2) Изменим регистр твитов на нижний с помощью .lower().  
3) Заменяем сокращения с апострофами (пример: ain't, can't) на пробел, используя apostrophe\_dict. Для этого необходимо сделать функцию: для каждого слова в тексте проверить (for word in text.split()), если слово есть в словаре apostrophe\_dict в качестве ключа (сокращенного слова), то заменить ключ на значение (полную версию слова).

4) Заменяем сокращения на их полные формы, используя short\_word\_dict. Для этого воспользуемся функцией, используемой в предыдущем пункте.

5) Заменяем эмодзи (пример: ":)") на пробелы, используя emoticon\_dict. Для этого воспользуемся функцией, используемой в предыдущем пункте.

6) Заменяем пунктуацию на пробелы, используя re.sub() и паттерн r'[\w\s]'

7) Заменяем спец. символы на пробелы, используя re.sub() и паттерн r'[\^a-zA-Z0-9]'

8) Заменяем числа на пробелы, используя re.sub() и паттерн r'[\^a-zA-Z]'

9) Удалим из текста слова длиной в 1 символ, используя ' '.join([w for w in x.split() if len(w)>1]).

10) Поделим твиты на токены с помощью nltk.tokenize.word\_tokenize, создав новый столбец 'tweet\_token'.

11) Удалим стоп-слова из токенов, используя nltk.corpus.stopwords. Создадим столбец 'tweet\_token\_filtered' без стоп-слов.

12) Применим стемминг к токенам с помощью nltk.stem.PorterStemmer. Создадим столбец 'tweet\_stemmed' после применения стемминга.

13) Применим лемматизацию к токенам с помощью nltk.stem.wordnet.WordNetLemmatizer. Создадим столбец 'tweet\_lemmatized' после применения лемматизации.

14) Сохраним результат предобработки в pickle-файл.

Выполнил **Соковнин ИЛ**

```
B [1]: apostrophe_dict = {
    "ain't": "am not / are not",
    "aren't": "are not / am not",
    "can't": "cannot",
    "can't've": "cannot have",
    "'cause": "because",
    "could've": "could have",
    "couldn't": "could not",
    "couldn't've": "could not have",
    "didn't": "did not",
    "doesn't": "does not",
    "don't": "do not",
    "hadn't": "had not",
    "hadn't've": "had not have",
    "hasn't": "has not",
    "haven't": "have not",
    "he'd": "he had / he would",
    "he'd've": "he would have",
    "he'll": "he shall / he will",
    "he'll've": "he shall have / he will have",
    "he's": "he has / he is",
    "how'd": "how did",
    "how'd'y": "how do you",
    "how'll": "how will",
    "how's": "how has / how is",
    "i'd": "I had / I would",
    "i'd've": "I would have",
    "i'll": "I shall / I will",
    "i'll've": "I shall have / I will have",
    "i'm": "I am",
    "i've": "I have",
    "isn't": "is not",
    "it'd": "it had / it would",
    "it'd've": "it would have",
    "it'll": "it shall / it will",
    "it'll've": "it shall have / it will have",
    "it's": "it has / it is",
    "let's": "let us",
    "ma'am": "madam",
    "mayn't": "may not",
    "might've": "might have",
    "mightn't": "might not",
    "mightn't've": "might not have",
    "must've": "must have",
    "mustn't": "must not",
    "mustn't've": "must not have",
    "needn't": "need not",
    "needn't've": "need not have",
    "o'clock": "of the clock",
    "oughtn't": "ought not",
    "oughtn't've": "ought not have",
    "shan't": "shall not",
    "sha'n't": "shall not",
    "shan't've": "shall not have",
    "she'd": "she had / she would",
    "she'd've": "she would have",
    "she'll": "she shall / she will",
    "she'll've": "she shall have / she will have",
    "she's": "she has / she is",
    "should've": "should have",
    "shouldn't": "should not",
    "shouldn't've": "should not have",
    "so've": "so have",
    "so's": "so as / so is",
    "that'd": "that would / that had",
    "that'd've": "that would have",
    "that's": "that has / that is",
    "there'd": "there had / there would",
    "there'd've": "there would have",
    "there's": "there has / there is",
    "they'd": "they had / they would",
    "they'd've": "they would have",
    "they'll": "they shall / they will",
    "they'll've": "they shall have / they will have",
    "they're": "they are",
    "they've": "they have",
    "to've": "to have",
    "wasn't": "was not",
    "we'd": "we had / we would",
    "we'd've": "we would have",
    "we'll": "we will",
    "we'll've": "we will have",
    "we're": "we are",
    "we've": "we have",
    "weren't": "were not",
    "what'll": "what shall / what will",
    "what'll've": "what shall have / what will have",
    "what're": "what are",
```

```
"what's": "what has / what is",
"what've": "what have",
"when's": "when has / when is",
"when've": "when have",
"where'd": "where did",
"where's": "where has / where is",
"where've": "where have",
"who'll": "who shall / who will",
"who'll've": "who shall have / who will have",
"who's": "who has / who is",
"who've": "who have",
"why's": "why has / why is",
"why've": "why have",
"will've": "will have",
"won't": "will not",
"won't've": "will not have",
"would've": "would have",
"wouldn't": "would not",
"wouldn't've": "would not have",
"y'all": "you all",
"y'all'd": "you all would",
"y'all'd've": "you all would have",
"y'all're": "you all are",
"y'all've": "you all have",
"you'd": "you had / you would",
"you'd've": "you would have",
"you'll": "you shall / you will",
"you'll've": "you shall have / you will have",
"you're": "you are",
"you've": "you have"
}
```

```
short_word_dict = {
"121": "one to one",
"a/s/l": "age, sex, location",
"adn": "any day now",
"afaik": "as far as I know",
"afk": "away from keyboard",
"aight": "alright",
"alol": "actually laughing out loud",
"b4": "before",
"b4n": "bye for now",
"bak": "back at the keyboard",
"bf": "boyfriend",
"bff": "best friends forever",
"bfm": "bye for now",
"bg": "big grin",
"bta": "but then again",
"btw": "by the way",
"cid": "crying in disgrace",
"cnp": "continued in my next post",
"cp": "chat post",
"cu": "see you",
"cul": "see you later",
"cul8r": "see you later",
"cya": "bye",
"cyo": "see you online",
"dbau": "doing business as usual",
"fud": "fear, uncertainty, and doubt",
"fwlw": "for what it's worth",
"fyi": "for your information",
"g": "grin",
"g2g": "got to go",
"ga": "go ahead",
"gal": "get a life",
"gf": "girlfriend",
"gfn": "gone for now",
"gmbo": "giggling my butt off",
"gmta": "great minds think alike",
"h8": "hate",
"hagn": "have a good night",
"hdop": "help delete online predators",
"hhis": "hanging head in shame",
"iac": "in any case",
"ianal": "I am not a lawyer",
"ic": "I see",
"idk": "I don't know",
"imao": "in my arrogant opinion",
"imnsho": "in my not so humble opinion",
"imo": "in my opinion",
"iow": "in other words",
"ipn": "I'm posting naked",
"irl": "in real life",
"jk": "just kidding",
"l8r": "later",
"ld": "later, dude",
}
```

```

"ldr": "long distance relationship",
"llta": "lots and lots of thunderous applause",
"lmao": "laugh my ass off",
"lmirl": "let's meet in real life",
"lol": "laugh out loud",
"ltr": "longterm relationship",
"lulab": "love you like a brother",
"lulas": "love you like a sister",
"luv": "love",
"m/f": "male or female",
"m8": "mate",
"milf": "mother I would like to fuck",
"oll": "online love",
"omg": "oh my god",
"otoh": "on the other hand",
"pir": "parent in room",
"ppl": "people",
"r": "are",
"rofl": "roll on the floor laughing",
"rpg": "role playing games",
"ru": "are you",
"shid": "slaps head in disgust",
"somy": "sick of me yet",
"sot": "short of time",
"thanx": "thanks",
"thx": "thanks",
"ttyl": "talk to you later",
"u": "you",
"ur": "you are",
"uw": "you're welcome",
"wb": "welcome back",
"wfm": "works for me",
"wibni": "wouldn't it be nice if",
"wtf": "what the fuck",
"wtg": "way to go",
"wtgp": "want to go private",
"ym": "young man",
"gr8": "great"
}

```

```

emoticon_dict = {
":)": "happy", # 1
":-)": "happy", # 2
":-]": "happy",
":-3": "happy",
":->": "happy",
"8-)": "happy", # 3
":-}": "happy",
":o)": "happy", # 4
":c)": "happy", # 5
":^)": "happy", # 6
"=]": "happy",
"=)": "happy", # 7
"<3": "happy",
":-(": "sad", # 8
":(": "sad", # 9
":c": "sad",
":<": "sad",
":[": "sad", # 10
">: [": "sad", # 11
":{": "sad",
">: (": "sad", # 12
":-c": "sad",
":-<": "sad", # 13
":-[": "sad", # 14
":-||": "sad" # 15
}

```

```

B [2]: import pandas as pd
import numpy as np
import re
import matplotlib.pyplot as plt
import seaborn as sns
import nltk
import warnings
warnings.filterwarnings("ignore", category=DeprecationWarning)
import os

from nltk.tokenize import word_tokenize

```

```
B [3]: # Сброс ограничений на количество выводимых рядов
pd.set_option('display.max_rows', None)

# Сброс ограничений на число столбцов
pd.set_option('display.max_columns', None)

# Сброс ограничений на количество символов в записи
pd.set_option('display.max_colwidth', None)
```

```
B [4]: train_df = pd.read_csv('./data/train_tweets.csv')
train_df.head()
```

Out[4]:

	id	label	tweet
0	1	0	@user when a father is dysfunctional and is so selfish he drags his kids into his dysfunction. #run
1	2	0	@user @user thanks for #lyft credit i can't use cause they don't offer wheelchair vans in pdx. #disapointed #getthanked
2	3	0	bihday your majesty
3	4	0	#model i love u take with u all the time in urøÿ±!!! øÿ™øÿøÿ„øÿ...øÿ øÿ øÿ
4	5	0	factsguide: society now #motivation

```
B [5]: test_df = pd.read_csv('./data/test_tweets.csv')
test_df.head()
```

Out[5]:

	id	tweet
0	31963	#studiolife #aislife #requires #passion #dedication #willpower to find #newmaterialsâ
1	31964	@user #white #supremacists want everyone to see the new â` #birdsâ™ #movie â” and hereâ™s why
2	31965	safe ways to heal your #acne!! #altwaystoheal #healthy #healing!!
3	31966	is the hp and the cursed child book up for reservations already? if yes, where? if no, when? øÿ~øÿ~øÿ~ #harrypotter #pottermore #favorite
4	31967	3rd #bihday to my amazing, hilarious #nephew eli ahmir! uncle dave loves you and missesâ

Осуществим предобработку данных с Твиттера, чтобы отчищенный данные в дальнейшем использовать для задачи классификации. Данный датасет содержит негативные (label = 1) и нейтральные (label = 0) высказывания.

Для работы объединим train\_df и test\_df.

```
B [6]: combine_df = train_df.append(test_df, ignore_index = True, sort = False)
combine_df.head()
```

Out[6]:

	id	label	tweet
0	1	0.0	@user when a father is dysfunctional and is so selfish he drags his kids into his dysfunction. #run
1	2	0.0	@user @user thanks for #lyft credit i can't use cause they don't offer wheelchair vans in pdx. #disapointed #getthanked
2	3	0.0	bihday your majesty
3	4	0.0	#model i love u take with u all the time in urøÿ±!!! øÿ™øÿøÿ„øÿ...øÿ øÿ øÿ
4	5	0.0	factsguide: society now #motivation

```
B [7]: print(combine_df.info())
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 49159 entries, 0 to 49158
Data columns (total 3 columns):
#   Column  Non-Null Count  Dtype
---  -
0   id      49159 non-null    int64
1   label   31962 non-null    float64
2   tweet   49159 non-null    object
dtypes: float64(1), int64(1), object(1)
memory usage: 1.1+ MB
None
```

## Задания:

1. Удалим @user из всех твитов с помощью паттерна "@[\w]\*". Для этого создадим функцию.

- для того, чтобы найти все вхождения паттерна в тексте, необходимо использовать re.findall(pattern, input\_txt)

- для замены @user на пробел, необходимо использовать re.sub()

```
B [8]: # help(re)
```

```
B [9]: print(combine_df['tweet'][1])
```

@user @user thanks for #lyft credit i can't use cause they don't offer wheelchair vans in pdx. #disappointed #getthan  
ked

```
B [10]: result = re.findall(r'@[\\w]*', combine_df['tweet'][1])
        print(result)
```

```
['@user', '@user']
```

```
B [11]: # result = re.sub(r'@[\\w]* ', '', combine_df['tweet'][1]) # удаляем с пробелом после @user
        result = re.sub(r'@[\\w]*', '', combine_df['tweet'][1]) # пробелы остаются
        print(result)
```

thanks for #lyft credit i can't use cause they don't offer wheelchair vans in pdx. #disappointed #getthanked

```
B [12]: def clean_text(text):
        ''' Удалим @user из всех твитов с помощью паттерна "@[\w]*" '''

        return re.sub(r'@[\w]*', '', text)
```

```
B [13]: %%time
# N = 5
# combine_df['tweet_1'] = combine_df['tweet'][:N].apply(clean_text)

combine_df['tweet'] = combine_df['tweet'].apply(clean_text)
```

Wall time: 296 ms

```
B [14]: combine_df.head()
```

Out[14]:

	id	label	tweet
0	1	0.0	when a father is dysfunctional and is so selfish he drags his kids into his dysfunction. #run
1	2	0.0	thanks for #lyft credit i can't use cause they don't offer wheelchair vans in pdx. #disappointed #getthanked
2	3	0.0	bihday your majesty
3	4	0.0	#model i love u take with u all the time in urδY~±!!! δY~TMδY~δY~.δY~...δY~ δY~ δY~
4	5	0.0	factsguide: society now #motivation

2. Изменим регистр твитов на нижний с помощью `.lower()`.

```
B [15]: def low_text(text):
        ''' 2. Изменим регистр твитов на нижний с помощью .lower() '''

        return text.lower()
```

```
В [16]: text = 'Удалим @user Из Всех Твитов С Помощью Паттерна "@[\w]*" '
print(text)
print(low_text(text))
```

Удалим @user Из Всех Твитов С Помощью Паттерна "@[\w]\*"  
удалим @user из всех твитов с помощью паттерна "@[\w]\*"

```
combine_df['tweet'] = combine_df['tweet'].apply(clean_text)
combine_df.head()
```

Out[17]:

id	label	tweet
0	1	0.0 when a father is dysfunctional and is so selfish he drags his kids into his dysfunction. #run
1	2	0.0 thanks for #lyft credit i can't use cause they don't offer wheelchair vans in pdx. #disapointed #getthanked
2	3	0.0 bihday your majesty
3	4	0.0 #model i love u take with u all the time in urδY~±!!! δY~TMδY~δY~.,δY~...δY~!δY~!δY~!
4	5	0.0 factsguide: society now #motivation

Для этого необходимо сделать функцию:<br>

```
B [18]: word = "can't"
         apostrophe_dict[word]
```

```
B [19]: def replacement_text(text, dictionary):
        """
        Для каждого слова в тексте проверить (for word in text.split()),
        если слово есть в словаре apostrophe_dict в качестве ключа (сокращенного слова),
        то заменить ключ на значение (полную версию слова).
        """
        for word in text.split():
            if word in dictionary:
                text = re.sub(word, dictionary.get(word), text)

        return text
```

```
B [21]: %%time
```

```
combine_df['tweet'] = combine_df['tweet'].apply(replacement_text, dictionary = apostrophe_dict)
combine_df.head(3)
```

Out[21]:

	id	label	tweet
0	1	0.0	when a father is dysfunctional and is so selfish he drags his kids into his dysfunction. #run
1	2	0.0	thanks for #lyft credit i cannot use cause they do not offer wheelchair vans in pdx. #disappointed #getthanked
2	3	0.0	bihday your majesty

```
B [22]: text = "121 a/s/l lol ltr lulab lulas luv gr8"
# text = "121, a/s/l"
# text = "121 a/s/l adn afaik afk aight alol b4"
print(text, '\n' + replacement_text(text, short_word_dict))
```

121 a/s/l lol ltr lulab lulas luv gr8  
one to one age, sex, location laugh out loud longterm relationship love you like a brother love you like a sister love  
great

B [23]: %%time

```
combine_df['tweet'] = combine_df['tweet'].apply(replacement_text, dictionary = short_word_dict)
combine_df.head(3)
```

Wall time: 621 ms

Out[23]:

	id	label	tweet
0	1	0.0	when a father is dysfunctional and is so selfish he drags his kids into his dysfunction. #run
1	2	0.0	thanks for #lyft credit i cannot use cause they do not offer wheelchair vans in pdx. #disappointed #getthanked
2	3	0.0	bihday your majesty

5) Заменяем эмодзи (пример: ":)" = "happy") на пробелы, используя `emoticon_dict`. Для этого воспользуемся функцией, используемой в предыдущем пункте.

```
B [24]: def replacement_emoticons(text, dictionary, flag=0):
        """ Заменяем эмодзи (пример: ":)" = "happy") на пробелы. """

        # print(text.split())
        # print()
        for emoticon in text.split():
            word = emoticon
            if emoticon in dictionary:
                if emoticon == ":)": # 1
                    word = r":\)"
                elif emoticon == ":-)": # 2
                    word = r":-\)"
                elif emoticon == "8-)": # 3
                    word = r"8-\)"
                elif emoticon == ":o)": # 4
                    word = r":o\)"
                elif emoticon == ":c)": # 5
                    word = r":c\)"
                elif emoticon == ":^)": # 6
                    word = r":^\)"
                elif emoticon == "=)": # 7
                    word = r"=\)"
                elif emoticon == ":-(": # 8
                    word = r":-\("
                elif emoticon == ":(": # 9
                    word = r":\("
                elif emoticon == ":[": # 10
                    word = r":\[("

                elif emoticon == ">:(": # 11
                    word = r">:\("
                elif emoticon == ">:(": # 12
                    word = r">:\("

                elif emoticon == ":-<": # 13
                    word = r":-\<"
                elif emoticon == ":-[": # 14
                    word = r":-\[("
                elif emoticon == ":-||": # 15
                    word = r":-\||\)"
            if flag == 0 :
                text = re.sub(word, dictionary.get(emoticon), text) # замена на "happy" or "sad"
            else:
                text = re.sub(word, " ", text) # замена на пробел

        return text
```



```
B [25]: emoticon = ":) :-) :-3 :-> 8-) :-} :o) :c) :( :c :< :[ >:[ >:( :-c :{ "

emoticon = ":) :-) :-] :-3 :-> 8-) :-} :o) :c) :^)=]=) <3 :- ( (: :c :< :[ >:[ :{ >:( :-c :-< :-[ :-||"

# emoticon = " >:[ \n >:( "
# emoticon = ":)" # 1
# emoticon = ":-)" # 2
# emoticon = "8-)" # 3
# emoticon = ":o)" # 4
# emoticon = ":c)" # 5
# emoticon = ":^)" # 6
# emoticon = "=)" # 7
# emoticon = ":-(" # 8
# emoticon = ":((" # 9
# emoticon = ":[(" # 10
# emoticon = " >:[ >:[ >:[ " # 11
# emoticon = ">:(" # 12
# emoticon = ":-<" # 13
# emoticon = ":-[" # 14
# emoticon = ":-||" # 15
# print('"' + emoticon + '": ', '"' + emoticon_dict.get(emoticon) + '"')

print(emoticon.split())
print()
print(replacement_emoticons(emoticon, emoticon_dict, 0))
print('*'*50)
print(replacement_emoticons(emoticon, emoticon_dict, 1))

[':)', ':-)', ':-]', ':-3', ':->', '8-)', ':-}', ':(o)', ':(c)', ':(^)', '=)', '=)', '<3', ':-(', ':((', ':(c)', ':(<', ':[(',
 '>:[(', ':{', '>:(', ':-c', ':-<', ':-[', ':-||']

happy happy happy happy happy happy happy happy happy happy happy happy happy happy sad sad sad sad sad >sad sad >sad sad sa
d sad sad
*****
> >
```

```
B [26]: emoticon_dict.get(':)')
```

Out[26]: 'happy'

```
B [27]: %%time

combine_df['tweet'] = combine_df['tweet'].apply(replacement_emoticons, dictionary = emoticon_dict, flag = 0)
combine_df.head(3)

Wall time: 639 ms
```

Out[27]:

	id	label	tweet
0	1	0.0	when a father is dysfunctional and is so selfish he drags his kids into his dysfunction. #run
1	2	0.0	thanks for #lyft credit i cannot use cause they do not offer wheelchair vans in pdx. #disappointed #getthanked
2	3	0.0	bihday your majesty

6. Заменим пунктуацию на пробелы, используя re.sub() и паттерн r'^\w\s]'.

```
B [28]: text = 'when a father is dysfunctional and is so selfish he drags his kids into his dysfunction. #run'
opt = re.sub(r'^\w\s]', '', text)
print(text)
print(opt)
```

when a father is dysfunctional and is so selfish he drags his kids into his dysfunction. #run  
when a father is dysfunctional and is so selfish he drags his kids into his dysfunction run

```
B [29]: def replacement_punctuation(text):
    """
    Заменим пунктуацию на пробелы.
    """
    text = re.sub(r'^\w\s]', ' ', text)

    return text
```

```
B [30]: %%time

combine_df['tweet'] = combine_df['tweet'].apply(replacement_punctuation)
combine_df.head(3)
```

Wall time: 935 ms

Out[30]:

	id	label	tweet
0	1	0.0	when a father is dysfunctional and is so selfish he drags his kids into his dysfunction run
1	2	0.0	thanks for lyft credit i cannot use cause they do not offer wheelchair vans in pdx disapointed getthanked
2	3	0.0	bihday your majesty

7.Заменим спец. символы на пробелы, используя re.sub() и паттерн r'^a-zA-Z0-9'.

```
B [31]: # def replace_special_characters(text):
#       """
#       Заменить пунктуацию на пробелы.
#       """
#       text = re.sub(r'^a-zA-Z0-9',' ', text)
#
#       return text

# combine_df['tweet_7'] = combine_df['tweet_6'].apply(replace_special_characters)
```

```
B [32]: def replace_patern(text, patern):
        """
        Заменить патерн на пробелы.
        """
        text = re.sub(patern, ' ', text)

        return text
```

```
B [33]: %%time

combine_df['tweet'] = combine_df['tweet'].apply(replace_patern, patern = r'^a-zA-Z0-9')
combine_df.head(3)
```

Wall time: 1.37 s

Out[33]:

	id	label	tweet
0	1	0.0	when a father is dysfunctional and is so selfish he drags his kids into his dysfunction run
1	2	0.0	thanks for lyft credit i cannot use cause they do not offer wheelchair vans in pdx disapointed getthanked
2	3	0.0	bihday your majesty

8. Заменим числа на пробелы, используя re.sub() и паттерн r'^a-zA-Z'.

```
B [34]: text = 'a1b2c3'
patern = r'^a-zA-Z'
replace_patern(text, patern)
```

Out[34]: 'a b c '

```
B [35]: %%time

combine_df['tweet'] = combine_df['tweet'].apply(replacement_text, dictionary = short_word_dict)
# combine_df.head(10)
```

Wall time: 502 ms

```
B [36]: %%time

combine_df['tweet'] = combine_df['tweet'].apply(replace_patern, patern = r'^a-zA-Z')
combine_df.head()
```

Wall time: 1.36 s

Out[36]:

	id	label	tweet
0	1	0.0	when a father is dysfunctional and is so selfish he drags his kids into his dysfunction run
1	2	0.0	thanks for lyft credit i cannot use cause they do not offer wheelchair vans in pdx disapointed getthanked
2	3	0.0	bihday your majesty
3	4	0.0	model i love yoyou take with yoyou all the time in yoyour
4	5	0.0	factsguide society now motivation

9. Удалим из текста слова длиной в 1 символ, используя ''.join([w for w in x.split() if len(w)>1]).

```
B [37]: def slugify(x):
        words = [w for w in x.split() if len(w)>1]

        return ' '.join(words)

slugify("My test 1 2 3 string")
```

Out[37]: 'My test string'

```
B [38]: %%time

combine_df['tweet'] = combine_df['tweet'].apply(slugify)
combine_df.head(3)
```

Wall time: 504 ms

Out[38]:

	id	label	tweet
0	1	0.0	when father is dysfunctional and is so selfish he drags his kids into his dysfunction run
1	2	0.0	thanks for lyft credit cannot use cause they do not offer wheelchair vans in pdx disapointed getthanked
2	3	0.0	bihday your majesty

10. Поделим твиты на токены с помощью nltk.tokenize.word\_tokenize, создав новый столбец 'tweet\_token'.

```
B [39]: from nltk import tokenize as tknz
```

```
B [40]: %%time

combine_df['tweet_token'] = combine_df['tweet'].apply(tknz.word_tokenize)
combine_df.head(3)
```

Wall time: 17.2 s

Out[40]:

	id	label	tweet	tweet_token
0	1	0.0	when father is dysfunctional and is so selfish he drags his kids into his dysfunction run	[when, father, is, dysfunctional, and, is, so, selfish, he, drags, his, kids, into, his, dysfunction, run]
1	2	0.0	thanks for lyft credit cannot use cause they do not offer wheelchair vans in pdx disapointed getthanked	[thanks, for, lyft, credit, can, not, use, cause, they, do, not, offer, wheelchair, vans, in, pdx, disapointed, getthanked]
2	3	0.0	bihday your majesty	[bihday, your, majesty]

```
B [41]: type(combine_df['tweet_token'][0])
```

Out[41]: list

11. Удалим стоп-слова из токенов, используя nltk.corpus.stopwords. Создадим столбец 'tweet\_token\_filtered' без стоп-слов.

```
B [42]: nltk.download('punkt')
from nltk.tokenize import word_tokenize
text = "This is a sentence in English that contains the SampleWord"
text_tokens = word_tokenize(text)
print(type(text_tokens))
print(text_tokens)
remove_sw = [word for word in text_tokens if not word in nltk.corpus.stopwords.words()]

print(remove_sw)

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

<class 'list'>
['This', 'is', 'a', 'sentence', 'in', 'English', 'that', 'contains', 'the', 'SampleWord']
['This', 'sentence', 'English', 'contains', 'SampleWord']
```

```
B [43]: def remove_stop_words(token):
        """
        Удалим стоп-слова из токенов.
        """

        remove_sw = [word for word in token if not word in nltk.corpus.stopwords.words("english")]

        return remove_sw
```

```
B [44]: def remove_stop_words(token, stop_words):
        """
        Удалим стоп-слова из токенов.
        """

        remove_sw = [word for word in token if not word in set(stop_words)]

        return remove_sw
```

```
B [45]: stop_words_en = set(nltk.corpus.stopwords.words("english"))
stop_words_en = nltk.corpus.stopwords.words("english")

type(stop_words_en)
```

Out[45]: list

```
B [46]: %%time

# combine_df['tweet_token_filtered'] = combine_df['tweet_token'].apply(remove_stop_words)
combine_df['tweet_token_filtered'] = combine_df['tweet_token'].apply(remove_stop_words, stop_words=stop_words_en)
# combine_df['tweet_token_filtered'] = combine_df.apply(lambda row: [w for w in row['tweet_token'] if not w in stop_words], axis=1)
combine_df.head(3)
```

Wall time: 7.08 s

Out[46]:

	id	label	tweet	tweet_token	tweet_token_filtered
0	1	0.0	when father is dysfunctional and is so selfish he drags his kids into his dysfunction run	[when, father, is, dysfunctional, and, is, so, selfish, he, drags, his, kids, into, his, dysfunction, run]	[father, dysfunctional, selfish, drags, kids, dysfunction, run]
1	2	0.0	thanks for lyft credit cannot use cause they do not offer wheelchair vans in pdx disapointed getthanked	[thanks, for, lyft, credit, can, not, use, cause, they, do, not, offer, wheelchair, vans, in, pdx, disapointed, getthanked]	[thanks, lyft, credit, use, cause, offer, wheelchair, vans, pdx, disapointed, getthanked]
2	3	0.0	bihday your majesty	[bihday, your, majesty]	[bihday, majesty]

```
B [47]: token = combine_df['tweet_token'][0]
```

```
B [48]: combine_df.to_csv("./combine_df.csv", index=False) # Сохранение без индексации
```

```
B [49]: # combine_df = pd.read_csv("./combine_df.csv", index_col='id')
# combine_df = pd.read_csv("./combine_df.csv")
# combine_df.head(3)
```

```
B [50]: # combine_df.drop(columns = ['Unnamed: 0'], axis = 1, inplace=True)
# combine_df.drop(combine_df.columns[[0]], axis = 1, inplace=True)
```

12. Применим стемминг к токенам с помощью nltk.stem.PorterStemmer. Создадим столбец 'tweet\_stemmed' после применения стемминга.

```
B [51]: from nltk.stem.snowball import SnowballStemmer
```

```
B [52]: snowball = SnowballStemmer('english')
```

```
B [53]: # words = word_tokenize(text_test)
# %%time

# combine_df['tweet_stemmed'] = combine_df['tweet_token_filtered'].apply(snowball.stem)
combine_df['tweet_stemmed'] = combine_df.apply(lambda row: [snowball.stem(w) for w in row['tweet_token_filtered']], axis=1)
combine_df.head(3)
```

Out[53]:

	id	label	tweet	tweet_token	tweet_token_filtered	tweet_stemmed
0	1	0.0	when father is dysfunctional and is so selfish he drags his kids into his dysfunction run	[when, father, is, dysfunctional, and, is, so, selfish, he, drags, his, kids, into, his, dysfunction, run]	[father, dysfunctional, selfish, drags, kids, dysfunction, run]	[father, dysfunct, selfish, drag, kid, dysfunct, run]
1	2	0.0	thanks for lyft credit cannot use cause they do not offer wheelchair vans in pdx disapointed getthanked	[thanks, for, lyft, credit, can, not, use, cause, they, do, not, offer, wheelchair, vans, in, pdx, disapointed, getthanked]	[thanks, lyft, credit, use, cause, offer, wheelchair, vans, pdx, disapointed, getthanked]	[thank, lyft, credit, use, caus, offer, wheelchair, van, pdx, disapoint, getthank]
2	3	0.0	bihday your majesty	[bihday, your, majesty]	[bihday, majesty]	[bihday, majesti]

13. Применим лемматизацию к токенам с помощью `nltk.stem.wordnet.WordNetLemmatizer`. Создадим столбец 'tweet\_lemmatized' после применения лемматизации.

```
B [54]: import nltk
nltk.download('wordnet')
```

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

Out[54]: True

```
B [55]: from nltk.stem import PorterStemmer, WordNetLemmatizer
from nltk.corpus import wordnet

def get_lemmatizer(word, lemmatizer, pos):
    """
    Print the results of stemmind and lemmitization using the passed stemmer, lemmatizer, word and pos (part of speech)
    """

    return lemmatizer.lemmatize(word, pos)

print(get_lemmatizer(word = "seen", lemmatizer = WordNetLemmatizer(), pos = wordnet.VERB))
print(get_lemmatizer(word = "drove", lemmatizer = WordNetLemmatizer(), pos = wordnet.VERB))
```

```
see
drive
```

```
B [56]: def get_lemmatizer(words, lemmatizer, pos):
    """
    Print the results of stemmind and lemmitization using the passed stemmer, lemmatizer, word and pos (part of speech)
    """

    lemmas = []
    for word in words:
        lemmas.append(lemmatizer.lemmatize(word, pos = nltk.corpus.wordnet.VERB) )

    return lemmas

print(get_lemmatizer(words = ["seen"], lemmatizer = WordNetLemmatizer(), pos = wordnet.VERB))
print(get_lemmatizer(words = ["drove"], lemmatizer = WordNetLemmatizer(), pos = wordnet.VERB))
```

```
['see']
['drive']
```

```
B [57]: print(combine_df['tweet_token_filtered'][1])
print(get_lemmatizer(words = combine_df['tweet_token_filtered'][1], lemmatizer = WordNetLemmatizer(), pos = wordnet.VERB)
```

```
['thanks', 'lyft', 'credit', 'use', 'cause', 'offer', 'wheelchair', 'vans', 'pdx', 'disapointed', 'getthanked']
['thank', 'lyft', 'credit', 'use', 'cause', 'offer', 'wheelchair', 'vans', 'pdx', 'disapointed', 'getthanked']
```

```
B [58]: %%time

combine_df['tweet_lemmatized'] = \
    combine_df['tweet_token_filtered'].apply(get_lemmatizer, lemmatizer = WordNetLemmatizer(), pos = wordnet.VERB)

combine_df.head(3)
```

Wall time: 5.18 s

Out[58]:

	id	label	tweet	tweet_token	tweet_token_filtered	tweet_stemmed	tweet_lemmatized
0	1	0.0	when father is dysfunctional and is so selfish he drags his kids into his dysfunction run	[when, father, is, dysfunctional, and, is, so, selfish, he, drags, his, kids, into, his, dysfunction, run]	[father, dysfunctional, selfish, drags, kids, dysfunction, run]	[father, dysfunct, selfish, drag, kid, dysfunct, run]	[father, dysfunctional, selfish, drag, kid, dysfunction, run]
1	2	0.0	thanks for lyft credit cannot use cause they do not offer wheelchair vans in pdx disapointed getthanked	[thanks, for, lyft, credit, can, not, use, cause, they, do, not, offer, wheelchair, vans, in, pdx, disapointed, getthanked]	[thanks, lyft, credit, use, cause, offer, wheelchair, vans, pdx, disapointed, getthanked]	[thank, lyft, credit, use, caus, offer, wheelchair, van, pdx, disapoint, getthank]	[thank, lyft, credit, use, cause, offer, wheelchair, vans, pdx, disapointed, getthanked]
2	3	0.0	bihday your majesty	[bihday, your, majesty]	[bihday, majesty]	[bihday, majesti]	[bihday, majesty]

```
B [59]: combine_df.to_csv("./combine_df_total.csv", index=False) # Сохранение без индексации
```

```
B [60]: # combine_df = pd.read_csv("./combine_df_total.csv")
# combine_df.head(3)
```

```
B [61]: lemmatizer = WordNetLemmatizer()

combine_df['tweet_lemmatized'] = combine_df.apply(lambda row: [lemmatizer.lemmatize(w, pos = nltk.corpus.wordnet.VERB)
                                                             for w in row['tweet_token_filtered']], axis=1)

combine_df.head(5)
```

Out[61]:

	id	label	tweet	tweet_token	tweet_token_filtered	tweet_stemmed	tweet_lemmatized
0	1	0.0	when father is dysfunctional and is so selfish he drags his kids into his dysfunction run	[when, father, is, dysfunctional, and, is, so, selfish, he, drags, his, kids, into, his, dysfunction, run]	[father, dysfunctional, selfish, drags, kids, dysfunction, run]	[father, dysfunct, selfish, drag, kid, dysfunct, run]	[father, dysfunctional, selfish, drag, kid, dysfunction, run]
1	2	0.0	thanks for lyft credit cannot use cause they do not offer wheelchair vans in pdx disapointed getthanked	[thanks, for, lyft, credit, can, not, use, cause, they, do, not, offer, wheelchair, vans, in, pdx, disapointed, getthanked]	[thanks, lyft, credit, use, cause, offer, wheelchair, vans, pdx, disapointed, getthanked]	[thank, lyft, credit, use, caus, offer, wheelchair, van, pdx, disapoint, getthank]	[thank, lyft, credit, use, cause, offer, wheelchair, vans, pdx, disapointed, getthanked]
2	3	0.0	bihday your majesty	[bihday, your, majesty]	[bihday, majesty]	[bihday, majesti]	[bihday, majesty]
3	4	0.0	model love yoyou take with yoyou all the time in yoyour	[model, love, yoyou, take, with, yoyou, all, the, time, in, yoyour]	[model, love, yoyou, take, yoyou, time, yoyour]	[model, love, yoyou, take, yoyou, time, yoyour]	[model, love, yoyou, take, yoyou, time, yoyour]
4	5	0.0	factsguide society now motivation	[factsguide, society, now, motivation]	[factsguide, society, motivation]	[factsguid, societi, motiv]	[factsguide, society, motivation]

14. Сохраним результат предобработки в pickle-файл.

```
B [62]: combine_df.to_pickle("./dummy.pkl")
```

```
B [ ]:
```