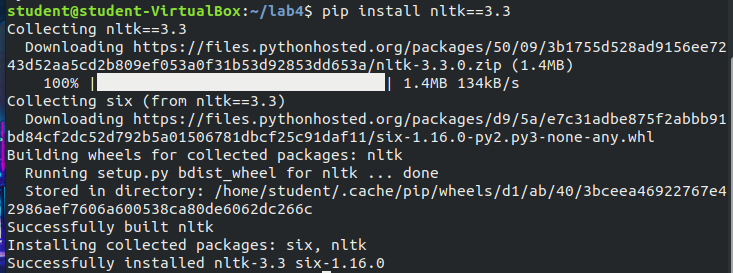
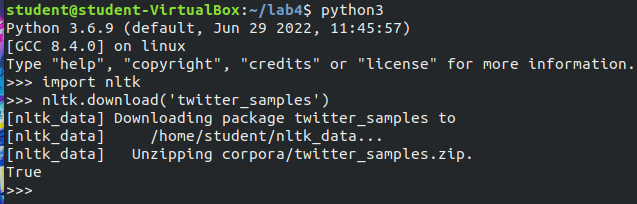
|  |  |  |
| --- | --- | --- |
| Лабораторная работа № 4  «Корпусная лингвистика. Машинное обучение»  Компьютерная лингвистика | Ф.И.О. | Самороков Н. Н. |
| Группа | ИВТ-363 |
| Преподаватель | Руженников А. А. |
| Дата сдачи |  |

**Ход работы:**

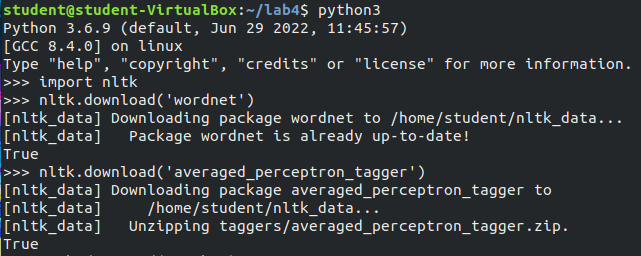
Установка пакета NLTK

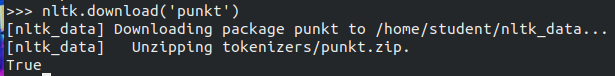


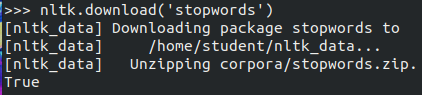
Импортируем модуль nltk и с помощью его загрузит примеры «твитов»



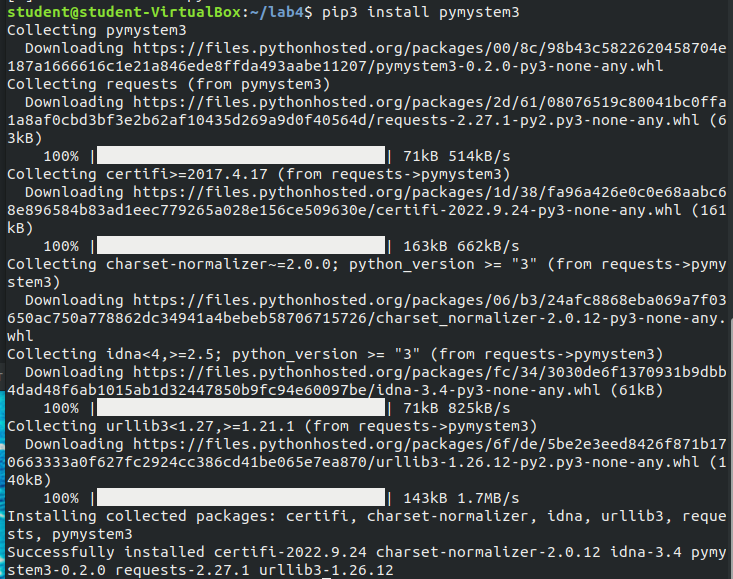
Прежде, чем приступить к анализу структуры слова и его контекста, загрузим необходимые ресурсы:







Для русского языка установим pymystem3



Создадим файл nlp\_test.py в котором будем писать код



**Отредактируем файл nlp\_test.py (**получим код для тестового примера**)**

from nltk.stem.wordnet import WordNetLemmatizer

from nltk.corpus import twitter\_samples, stopwords

from nltk.tag import pos\_tag

from nltk.tokenize import word\_tokenize

from nltk import FreqDist, classify, NaiveBayesClassifier

import re, string, random

def remove\_noise(tweet\_tokens, stop\_words = ()):

cleaned\_tokens = []

for token, tag in pos\_tag(tweet\_tokens):

token = re.sub('http[s]?://(?:[a-zA-Z]|[0-9]|[$-\_@.&+#]|[!\*\(\),]|'\

'(?:%[0-9a-fA-F][0-9a-fA-F]))+','', token)

token = re.sub("(@[A-Za-z0-9\_]+)","", token)

if tag.startswith("NN"):

pos = 'n'

elif tag.startswith('VB'):

pos = 'v'

else:

pos = 'a'

lemmatizer = WordNetLemmatizer()

token = lemmatizer.lemmatize(token, pos)

if len(token) > 0 and token not in string.punctuation and token.lower() not in stop\_words:

cleaned\_tokens.append(token.lower())

return cleaned\_tokens

def get\_all\_words(cleaned\_tokens\_list):

for tokens in cleaned\_tokens\_list:

for token in tokens:

yield token

def get\_tweets\_for\_model(cleaned\_tokens\_list):

for tweet\_tokens in cleaned\_tokens\_list:

yield dict([token, True] for token in tweet\_tokens)

if \_\_name\_\_ == "\_\_main\_\_":

positive\_tweets = twitter\_samples.strings('positive\_tweets.json')

negative\_tweets = twitter\_samples.strings('negative\_tweets.json')

text = twitter\_samples.strings('tweets.20150430-223406.json')

tweet\_tokens = twitter\_samples.tokenized('positive\_tweets.json')[0]

stop\_words = stopwords.words('english')

positive\_tweet\_tokens = twitter\_samples.tokenized('positive\_tweets.json')

negative\_tweet\_tokens = twitter\_samples.tokenized('negative\_tweets.json')

positive\_cleaned\_tokens\_list = []

negative\_cleaned\_tokens\_list = []

for tokens in positive\_tweet\_tokens:

positive\_cleaned\_tokens\_list.append(remove\_noise(tokens, stop\_words))

for tokens in negative\_tweet\_tokens:

negative\_cleaned\_tokens\_list.append(remove\_noise(tokens, stop\_words))

all\_pos\_words = get\_all\_words(positive\_cleaned\_tokens\_list)

freq\_dist\_pos = FreqDist(all\_pos\_words)

print(freq\_dist\_pos.most\_common(10))

positive\_tokens\_for\_model = get\_tweets\_for\_model(positive\_cleaned\_tokens\_list)

negative\_tokens\_for\_model = get\_tweets\_for\_model(negative\_cleaned\_tokens\_list)

positive\_dataset = [(tweet\_dict, "Positive")

for tweet\_dict in positive\_tokens\_for\_model]

negative\_dataset = [(tweet\_dict, "Negative")

for tweet\_dict in negative\_tokens\_for\_model]

dataset = positive\_dataset + negative\_dataset

random.shuffle(dataset)

train\_data = dataset[:7000]

test\_data = dataset[7000:]

classifier = NaiveBayesClassifier.train(train\_data)

print("Accuracy is:", classify.accuracy(classifier, test\_data))

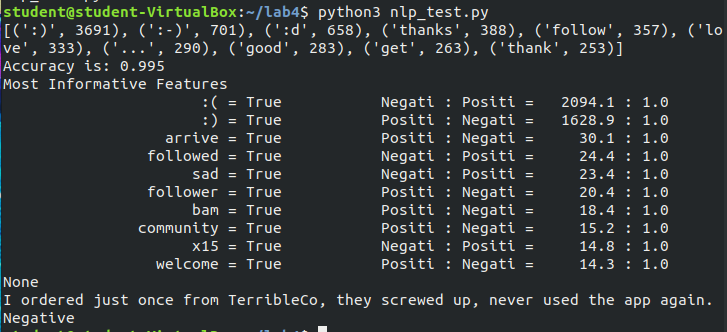
print(classifier.show\_most\_informative\_features(10))

custom\_tweet = "I ordered just once from TerribleCo, they screwed up, never used the app again."

custom\_tokens = remove\_noise(word\_tokenize(custom\_tweet))

print(custom\_tweet, classifier.classify(dict([token, True] for token in custom\_tokens)))

Запустим полученный файл с кодом



**Код для анализа 10-ти предложений и определения тональности текста:**

from nltk.stem.wordnet import WordNetLemmatizer

from nltk.corpus import twitter\_samples, stopwords

from nltk.tag import pos\_tag

from nltk.tokenize import word\_tokenize

from nltk import FreqDist, classify, NaiveBayesClassifier

import re, string, random

def remove\_noise(tweet\_tokens, stop\_words = ()):

cleaned\_tokens = []

for token, tag in pos\_tag(tweet\_tokens):

token = re.sub('http[s]?://(?:[a-zA-Z]|[0-9]|[$-\_@.&+#]|[!\*\(\),]|'\

'(?:%[0-9a-fA-F][0-9a-fA-F]))+','', token)

token = re.sub("(@[A-Za-z0-9\_]+)","", token)

if tag.startswith("NN"):

pos = 'n'

elif tag.startswith('VB'):

pos = 'v'

else:

pos = 'a'

lemmatizer = WordNetLemmatizer()

token = lemmatizer.lemmatize(token, pos)

if len(token) > 0 and token not in string.punctuation and token.lower() not in stop\_words:

cleaned\_tokens.append(token.lower())

return cleaned\_tokens

def get\_all\_words(cleaned\_tokens\_list):

for tokens in cleaned\_tokens\_list:

for token in tokens:

yield token

def get\_tweets\_for\_model(cleaned\_tokens\_list):

for tweet\_tokens in cleaned\_tokens\_list:

yield dict([token, True] for token in tweet\_tokens)

if \_\_name\_\_ == "\_\_main\_\_":

positive\_tweets = twitter\_samples.strings('positive\_tweets.json')

negative\_tweets = twitter\_samples.strings('negative\_tweets.json')

text = twitter\_samples.strings('tweets.20150430-223406.json')

tweet\_tokens = twitter\_samples.tokenized('positive\_tweets.json')[0]

stop\_words = stopwords.words('english')

positive\_tweet\_tokens = twitter\_samples.tokenized('positive\_tweets.json')

negative\_tweet\_tokens = twitter\_samples.tokenized('negative\_tweets.json')

positive\_cleaned\_tokens\_list = []

negative\_cleaned\_tokens\_list = []

for tokens in positive\_tweet\_tokens:

positive\_cleaned\_tokens\_list.append(remove\_noise(tokens, stop\_words))

for tokens in negative\_tweet\_tokens:

negative\_cleaned\_tokens\_list.append(remove\_noise(tokens, stop\_words))

all\_pos\_words = get\_all\_words(positive\_cleaned\_tokens\_list)

freq\_dist\_pos = FreqDist(all\_pos\_words)

#print(freq\_dist\_pos.most\_common(10))

positive\_tokens\_for\_model = get\_tweets\_for\_model(positive\_cleaned\_tokens\_list)

negative\_tokens\_for\_model = get\_tweets\_for\_model(negative\_cleaned\_tokens\_list)

positive\_dataset = [(tweet\_dict, "Positive")

for tweet\_dict in positive\_tokens\_for\_model]

negative\_dataset = [(tweet\_dict, "Negative")

for tweet\_dict in negative\_tokens\_for\_model]

dataset = positive\_dataset + negative\_dataset

random.shuffle(dataset)

train\_data = dataset[:7000]

test\_data = dataset[7000:]

classifier = NaiveBayesClassifier.train(train\_data)

print("Accuracy is: ", classify.accuracy(classifier, test\_data))

print("\n")

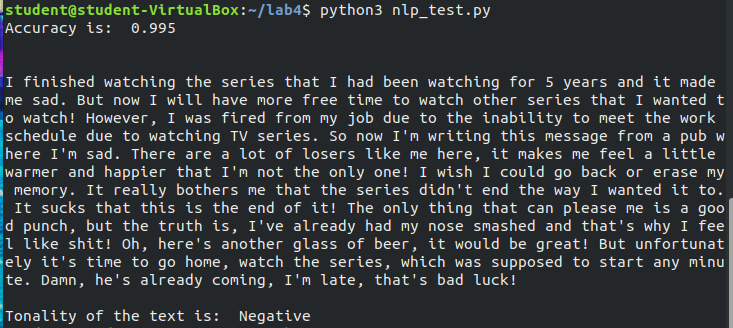
#print(classifier.show\_most\_informative\_features(10))

custom\_tweet = "I finished watching the series that I had been watching for 5 years and it made me sad. But now I will have more free time to watch other series that I wanted to watch! However, I was fired from my job due to the inability to meet the work schedule due to watching TV series. So now I'm writing this message from a pub where I'm sad. There are a lot of losers like me here, it makes me feel a little warmer and happier that I'm not the only one! I wish I could go back or erase my memory. It really bothers me that the series didn't end the way I wanted it to. It sucks that this is the end of it! The only thing that can please me is a good punch, but the truth is, I've already had my nose smashed and that's why I feel like shit! Oh, here's another glass of beer, it would be great! But unfortunately it's time to go home, watch the series, which was supposed to start any minute. Damn, he's already coming, I'm late, that's bad luck!"

custom\_tokens = remove\_noise(word\_tokenize(custom\_tweet))

print(custom\_tweet + "\n\n" + "Tonality of the text is: ", classifier.classify(dict([token, True] for token in custom\_tokens)))

Результат работы кода:



# Вывод:

Эта работа была посвящена базовой модели анализа настроений с использованием библиотеки nltk в Python 3. Сначала вы выполнили предварительную обработку твитов, используя токены, нормализуя слова и удаляя шумы. Затем вы визуализировали часто встречающиеся элементы в данных. Наконец, вы создали модель, чтобы связать твиты с определенным настроением. Контролируемая модель обучения хороша только как данные обучения. Чтобы еще больше усилить модель, вы не ограничены возможностью рассмотреть и добавить большее количество категорий, таких как, например, волнение и гнев.