ЛАБОРАТОРНАЯ РАБОТА. ОБРАБОТКА ТЕХТА В NLTK

Краткие сведения.

Цель настоящей работы – изучить технику работы с файлами и web, т.е. мы не работаем с книгами (books), используемыми nltk.

ПРОВЕРЬТЕ ДОСТУПНОСТЬ СЛЕДУЮЩИХ БИБЛИОТЕК

```
>>> from __future__ import division  # Python 2 users only
>>> import nltk, re, pprint
>>> from nltk import word_tokenize
```

Попытайтесь прочитать текстовый файл в интернете

```
>>> from urllib import request
>>> url = "http://www.gutenberg.org/files/2554/2554.txt"
>>> response = request.urlopen(url)
>>> raw = response.read().decode('utf8')
>>> type(raw)
<class 'str'>
>>> len(raw)
1176893
>>> raw[:75]
'The Project Gutenberg EBook of Crime and Punishment, by Fyodor
Dostoevsky\r\n'
```

Если используйте proxy-сервер попробуйте подключиться к интернет так

```
>>> proxies = {'http': 'http://www.someproxy.com:3128'}
>>> request.ProxyHandler(proxies)
```

Переменная `raw` содержит 1,176,893 символов. Она содержит контент книги.

Выполним токенизацию.

```
>>> tokens = word_tokenize(raw)
>>> type(tokens)
<class 'list'>
>>> len(tokens)
254354
>>> tokens[:10]
['The', 'Project', 'Gutenberg', 'EBook', 'of', 'Crime', 'and',
'Punishment', ',', 'by']
```

Получить текст в конкретном диапазоне индексов:

```
       >>> text = nltk.Text(tokens)
>>> type(text)
<class 'nltk.text.Text'>
>>> text[1024:1062]
['CHAPTER', 'I', 'On', 'an', 'exceptionally', 'hot', 'evening', 'early',
'in',
 'July', 'a', 'young', 'man', 'came', 'out', 'of', 'the', 'garret', 'in',
 'which', 'he', 'lodged', 'in', 'S.', 'Place', 'and', 'walked', 'slowly',
 ',', 'as', 'though', 'in', 'hesitation', ',', 'towards', 'K.', 'bridge',
'.']
>>> text.collocations()
Katerina Ivanovna; Pyotr Petrovitch; Pulcheria Alexandrovna; Avdotya
Romanovna; Rodion Romanovitch; Marfa Petrovna; Sofya Semyonovna; old
woman; Project Gutenberg-tm; Porfiry Petrovitch; Amalia Ivanovna;
great deal; Nikodim Fomitch; young man; Ilya Petrovitch; n't know;
Project Gutenberg; Dmitri Prokofitch; Andrey Semyonovitch; Hay Market
```

Выполните команды поиска:

```
       >>> raw.find("PART I")
5338
>>> raw.rfind("End of Project Gutenberg's Crime")
1157743
>>> raw = raw[5338:1157743]
>>> raw.find("PART I")
0
```

`rfind()` ("reverse find") поиск в обратном направлении.

Работа с HTML

Чтение сайта:

```
       >>> url = "http://news.bbc.co.uk/2/hi/health/2284783.stm"
>>> html = request.urlopen(url).read().decode('utf8')
>>> html[:60]
'<!doctype html public "-//W3C//DTD HTML 4.0 Transitional//EN'
```

Чтобы прочитать текст HTML будем использовать библиотеку *BeautifulSoup*, которую можно скачать с `http://www.crummy.com/software/BeautifulSoup/`:

```
       >>> from bs4 import BeautifulSoup
>>> raw = BeautifulSoup(html).get_text()
>>> tokens = word_tokenize(raw)
>>> tokens
['BBC', 'NEWS', '|', 'Health', '|', 'Blondes', "'to", 'die', 'out', ...]
```

Полезный текст выбираем в определенном диапазоне.

```
       >>> tokens = tokens[110:390]
>>> text = nltk.Text(tokens)
>>> text.concordance('gene')
Displaying 5 of 5 matches:
hey say too few people now carry the gene for blondes to last beyond the
next
blonde hair is caused by a recessive gene . In order for a child to have
blond
have blonde hair , it must have the gene on both sides of the family in
the g
ere is a disadvantage of having that gene or by chance . They do n't
disappear
des would disappear is if having the gene was a disadvantage and I do not
thin
```

Доступ к содержимому блога

Нужна библиотека *Universal Feed Parser*, которую можно скачать с `https://pypi.python.org/pypi/feedparser`, Получаем доступ к блогу таким образом:

```
       >>> import feedparser
>>> llog = feedparser.parse("http://languagelog.ldc.upenn.edu/nll/?
feed=atom")
>>> llog['feed']['title']
'Language Log'
>>> len(llog.entries)
15
>>> post = llog.entries[2]
>>> post.title
"He's My BF"
>>> content = post.content[0].value
>>> content[:70]
'<p>Today I was chatting with three of our visiting graduate students f'
>>> raw = BeautifulSoup(content).get_text()
>>> word_tokenize(raw)
['Today', 'I', 'was', 'chatting', 'with', 'three', 'of', 'our',
'visiting',
'graduate', 'students', 'from', 'the', 'PRC', '.', 'Thinking', 'that',
'I',
'was', 'being', 'au', 'courant', ',', 'I', 'mentioned', 'the',
'expression',
'DUI4XIANG4', '\u5c0d\u8c61', '("', 'boy', '/', 'girl', 'friend',
'"', ...]
```

Чтение локальных файлов.

:

```
       >>> f = open('document.txt')
>>> raw = f.read()
```

Содержимое текущего директория:

```
       >>> import os
>>> os.listdir('.')
```

Метод `read()` считывает все содержимое файла:

```
        >>> f.read()
'Time flies like an arrow.\nFruit flies like a banana.\n'
```

Построчное считывание файла:

```
        >>> f = open('document.txt', 'rU')
>>> for line in f:
...     print(line.strip())
Time flies like an arrow.
Fruit flies like a banana.
```

Метод `strip()` убирает символ перевода каретки.

Метод `nltk.data.find()` получает имена всех файлов в корпусе документов. Затем файл можно прочитать:

```
        >>> path = nltk.data.find('corpora/gutenberg/melville-
moby_dick.txt')
>>> raw = open(path, 'rU').read()
```

```
        >>> raw = open('document.txt').read()
>>> type(raw)
<class 'str'>
```

Токенизация содержимого файла

```
        >>> tokens = word_tokenize(raw)
>>> type(tokens)
<class 'list'>
>>> words = [w.lower() for w in tokens]
>>> type(words)
<class 'list'>
>>> vocab = sorted(set(words))
>>> type(vocab)
<class 'list'>
```

Отыскать позицию подстроки в строке: `find()`:

```
        >>> monty.find('Python')
6
```

Функция `nltk.data.find()` определяет путь к файлу.

```
        >>> path = nltk.data.find('corpora/unicode_samples/polish-
lat2.txt')
```

Использование регулярных выражений:

(выбор слов, заканчивающихся на ed)

```
>>> [w for w in wordlist if re.search('ed$', w)]
['abaissed', 'abandoned', 'abased', 'abashed', 'abatised', 'abed',
'aborted', ...]
```

Сообразите, что делает следующий код:

```
>>> [w for w in wordlist if re.search('^..j..t..$', w)]
['abjectly', 'adjuster', 'dejected', 'dejectly', 'injector',
'majestic', ...]
```

ПОИСК

`re.search('^e-?mail$', w))`.

Использование регулярного выражения при поиске:

```
>>> [w for w in wordlist if re.search('^[ghi][mno][jlk][def]
$', w)]
['gold', 'golf', 'hold', 'hole']
```

Паттерн	Действие
.	Соответствует любому одному символу
^abc	abc Начинает строку
abc$	*abc* заканчивает строку
[abc]	Соответствует любому одному символу
[A-Z0-9]	Диапазон
ed|ing|s	ИЛИ
*	Ноль или более
+	Один или более
?	Ноль или один из предшествующих символов
{n}	Ровно n символов
{n,}	Не менее n символов
{,n}	Не более n символов
{m,n}	В диапазоне
a(b|c)+	Скобки определяют блок символов

Извлечение кусков слова

`re.findall()` находит все куски, удовлетворяющие паттерну:

```
       >>> word = 'supercalifragilisticexpialidocious'
>>> re.findall(r'[aeiou]', word)
['u', 'e', 'a', 'i', 'a', 'i', 'i', 'i', 'e', 'i', 'a', 'i', 'o', 'i',
'o', 'u']
>>> len(re.findall(r'[aeiou]', word))
16
```

Например, найдем последовательность из двух и более подряд идущих гласных:

```
       >>> wsj = sorted(set(nltk.corpus.treebank.words()))
>>> fd = nltk.FreqDist(vs for word in wsj
...                       for vs in re.findall(r'[aeiou]{2,}', word))
>>> fd.most_common(12)
[('io', 549), ('ea', 476), ('ie', 331), ('ou', 329), ('ai', 261), ('ia',
253),
('ee', 217), ('oo', 174), ('ua', 109), ('au', 106), ('ue', 105), ('ui',
95)]
```

.

```
       >>> regexp = r'^[AEIOUaeiou]+|[AEIOUaeiou]+$|[^AEIOUaeiou]'
>>> def compress(word):
...     pieces = re.findall(regexp, word)
...     return ''.join(pieces)
...
>>> english_udhr = nltk.corpus.udhr.words('English-Latin1')
>>> print(nltk.tokenwrap(compress(w) for w in english_udhr[:75]))
Unvrsl Dclrtn of Hmn Rghts Prmble Whrs rcgntn of the inhrnt dgnty and
of the eql and inlnble rghts of all mmbrs of the hmn fmly is the fndtn
of frdm , jstce and pce in the wrld , Whrs dsrgrd and cntmpt fr hmn
rghts hve rsltd in brbrs acts whch hve outrgd the cnscnce of mnknd ,
and the advnt of a wrld in whch hmn bngs shll enjy frdm of spch and
```

Разберитесь со следующим кодом:

```
       >>> rotokas_words = nltk.corpus.toolbox.words('rotokas.dic')
>>> cvs = [cv for w in rotokas_words for cv in re.findall(r'[ptksvr]
[aeiou]', w)]
>>> cfd = nltk.ConditionalFreqDist(cvs)
>>> cfd.tabulate()
    a    e    i    o    u
k  418  148   94  420  173
p   83   31  105   34   51
r  187   63   84   89   79
s    0    0  100    2    1
t   47    8    0  148   37
v   93   27  105   48   49
```

Нахождение основы слова

```
       >>> def stem(word):
...     for suffix in ['ing', 'ly', 'ed', 'ious', 'ies', 'ive', 'es',
's', 'ment']:
...         if word.endswith(suffix):
...             return word[:-len(suffix)]
...     return word
```

.

```
       >>> re.findall(r'^.*(ing|ly|ed|ious|ies|ive|es|s|ment)$',
'processing')
['ing']
```

Отделение суффикса от корня:

```
       >>> re.findall(r'^(.*)(ing|ly|ed|ious|ies|ive|es|s|ment)$',
'processing')
[('process', 'ing')]
```

Проверьте этот код:

```
       >>> re.findall(r'^(.*?)(ing|ly|ed|ious|ies|ive|es|s|ment)?$',
'language')
[('language', '')]
```

Выполните код.

```
       >>> from nltk.corpus import gutenberg, nps_chat
>>> moby = nltk.Text(gutenberg.words('melville-moby_dick.txt'))
>>> moby.findall(r"<a> (<.*>) <man>")
monied; nervous; dangerous; white; white; white; pious; queer; good;
mature; white; Cape; great; wise; wise; butterless; white; fiendish;
pale; furious; better; certain; complete; dismasted; younger; brave;
brave; brave; brave
>>> chat = nltk.Text(nps_chat.words())
>>> chat.findall(r"<.*> <.*> <bro>")
you rule bro; telling you bro; u twizted bro
>>> chat.findall(r"<l.*>{3,}")
lol lol lol; lmao lol lol; lol lol lol; la la la la la; la la la; la
la la; lovely lol lol love; lol lol lol.; la la la; la la la
```

3.7 Регулярные выражения для токенизации текста

```
       >>> raw = """'When I'M a Duchess,' she said to herself, (not in a
very hopeful tone
... though), 'I won't have any pepper in my kitchen AT ALL. Soup does
very
... well without--Maybe it's always pepper that makes people hot-
tempered,'..."""
```

Расщепление текста на отдельные слова:

```
      >>> re.split(r' ', raw)
["'When", "I'M", 'a', "Duchess,'", 'she', 'said', 'to', 'herself,',
'(not', 'in',
'a', 'very', 'hopeful', 'tone\nthough),', "'I", "won't", 'have', 'any',
'pepper',
'in', 'my', 'kitchen', 'AT', 'ALL.', 'Soup', 'does', 'very\nwell',
'without--Maybe',
"it's", 'always', 'pepper', 'that', 'makes', 'people', "hot-
tempered,'..."]
>>> re.split(r'[ \t\n]+', raw)
["'When", "I'M", 'a', "Duchess,'", 'she', 'said', 'to', 'herself,',
'(not', 'in',
'a', 'very', 'hopeful', 'tone', 'though),', "'I", "won't", 'have', 'any',
'pepper',
'in', 'my', 'kitchen', 'AT', 'ALL.', 'Soup', 'does', 'very', 'well',
'without--Maybe',
"it's", 'always', 'pepper', 'that', 'makes', 'people', "hot-
tempered,'..."]
```

Другой пример расщепления:

```
      >>> re.split(r'\W+', raw)
['', 'When', 'I', 'M', 'a', 'Duchess', 'she', 'said', 'to', 'herself',
'not', 'in',
'a', 'very', 'hopeful', 'tone', 'though', 'I', 'won', 't', 'have', 'any',
'pepper',
'in', 'my', 'kitchen', 'AT', 'ALL', 'Soup', 'does', 'very', 'well',
'without',
'Maybe', 'it', 's', 'always', 'pepper', 'that', 'makes', 'people', 'hot',
'tempered',
'']
```

Поиск на основе регулярного выражения.

```
      >>> re.findall(r'\w+|\S\w*', raw)
["'When", 'I', "'M", 'a', 'Duchess', ',', "'", 'she', 'said', 'to',
'herself', ',',
'(not', 'in', 'a', 'very', 'hopeful', 'tone', 'though', ')', ',', "'I",
'won', "'t",
'have', 'any', 'pepper', 'in', 'my', 'kitchen', 'AT', 'ALL', '.', 'Soup',
'does',
'very', 'well', 'without', '-', '-Maybe', 'it', "'s", 'always', 'pepper',
'that',
'makes', 'people', 'hot', '-tempered', ',', "'", '.', '.', '.']
```

.

Символ	Функция
`\b`	Граница слова
`\d`	Десятичная цифра `[0-9]`)
`\D`	Любой символ не-цифра `[^0-9]`)
`\s`	Символ пробела `[ \t\n\r\f\v]`)
`\S`	Любой символ, отличный от пробела `[^ \t\n\r\f\v]`)

Символ	Функция
\w	Алфавитный символ [a-zA-Z0-9_])
\W	Спец-символ[^a-zA-Z0-9_])
\t	Табуляция
\n	Символ новой строки

Токенайзер

```
       >>> text = 'That U.S.A. poster-print costs $12.40...'
>>> pattern = r'''(?x)    # set flag to allow verbose regexps
...     ([A-Z]\.)+        # abbreviations, e.g. U.S.A.
...   | \w+(-\w+)*        # words with optional internal hyphens
...   | \$?\d+(\.\d+)?%?  # currency and percentages, e.g. $12.40, 82%
...   | \.\.\.            # ellipsis
...   | [][.,;"'?():-_`]  # these are separate tokens; includes ], [
... '''
>>> nltk.regexp_tokenize(text, pattern)
['That', 'U.S.A.', 'poster-print', 'costs', '$12.40', '...']
```

ЗАДАНИЕ.

Выполните все примеры

Выполните токенизацию строки и удалите у всех слов гласные

Выполните токенизацию строки и выведите все слова, содержащие ing в конце слова

Продемонстрируйте замену слов с ing в конце слова на ed (ing заменить на ed)