Прикладные задачи анализа данных

Семинар 2
NLTK (Natural Language Toolkit)

Национальныи Исследовательский Университет
Высшая Школа Экономики

25 января 2018

Natural language toolkit

Language processing task	NLTK modules	Functionality
Accessing corpora	corpus	standardized interfaces to corpora and lexicons
String processing	tokenize, stem	tokenizers, sentence tokenizers, stemmers
Collocation discovery	collocations	t-test, chi-squared, point-wise mutual information
Part-of-speech tagging	tag	n-gram, backoff, Brill, HMM, TnT
Machine learning	classify, cluster, tbl	decision tree, maximum entropy, naive Bayes, EM, k-means
Chunking	chunk	regular expression, n-gram, named-entity
Parsing	parse, ccg	chart, feature-based, unification, probabilistic, dependency
Semantic interpretation	sem, inference	lambda calculus, first-order logic, model checking
Evaluation metrics	metrics	precision, recall, agreement coefficients
Probability and estimation	probability	frequency distributions, smoothed probability distributions
Applications	app, chat	graphical concordancer, parsers, WordNet browser, chatbots
Linguistic fieldwork	toolbox	manipulate data in SIL Toolbox format

nltk.book

План на сегодня

1. segmentation
2. nltk.FreqDist
3. lemmatization
4. nltk.Text
5. Text Corpora

План на сегодня

1. segmentation
2. nltk.FreqDist
3. lemmatization
4. nltk.Text
5. Text Corpora

Токенизация с помощью регулярных выражений

Задаем шаблоны, описывающие токены.

Регулярные выражения на Python

```
In[1]: import re
In[2]: prog = re.compile('[А-Яа-я\-]+')
In[3]: prog.findall("Слова, больше слов,
что-то.")
Out[1]: ['Слова', 'больше', 'слов', 'что-то']
```

nltk.tokenize.sent_tokenize

```
In[1]: text = “Население г. Москва составляет
11.92 млн человек. Это точно? Нет, не точно!”
In[2]: from nltk.tokenize import sent_tokenize
In[3]: sents = sent_tokenize(text)
In[4]: print(sents)
In[5]: len(sents)
Out[1]: [’Население г. Москва составляет 11.92
млн человек.’, ’Это точно?’, ’Нет, не не точно!’]
Out[2]: 3
```

План на сегодня

1. segmentation
2. nltk.FreqDist
3. lemmatization
4. nltk.Text
5. Text Corpora

nltk.FreqDist Functions toolkit

Example	Description
`fdist = FreqDist(samples)`	create a frequency distribution containing the given samples
`fdist[sample] += 1`	increment the count for this sample
`fdist['monstrous']`	count of the number of times a given sample occurred
`fdist.freq('monstrous')`	frequency of a given sample
`fdist.N()`	total number of samples
`fdist.most_common(n)`	the `n` most common samples and their frequencies
`for sample in fdist:`	iterate over the samples
`fdist.max()`	sample with the greatest count
`fdist.tabulate()`	tabulate the frequency distribution
`fdist.plot()`	graphical plot of the frequency distribution
`fdist.plot(cumulative=True)`	cumulative plot of the frequency distribution
`fdist1 |= fdist2`	update `fdist1` with counts from `fdist2`
`fdist1 < fdist2`	test if samples in `fdist1` occur less frequently than in `fdist2`

Входной текст

вводим строку

```
s = '«Карты, деньги, два ствола» культовый фильм Гая Ричи...'
```

или открываем файл с текстом

```
s = ''
with open('data/winni.txt') as infile:
    for line in infile:
        s += line.strip() + ' '
s=s.replace('– ','') # диалоги с '– '
```

nltk.FreqDist

```
import nltk
import re
prog = re.compile('[А-Яа-я\-]+')
l1 = prog.findall(s.lower())
d1 = nltk.FreqDist(l1)
print (d1)
print (d1.most_common(10))
```

Словоформы, типы (NLTK)

most_common для Гая Ричи

```
Out[1]: <FreqDist with 239 samples and 360
outcomes>
Out[2]: [('и', 22), ('но', 7), ('что', 7),
('фильм', 6), ('ричи', 6), ('не', 6), ('все', 5),
('а', 4), ('на', 4), ('в', 4)]
```

most_common для Винни Пуха

```
Out[1]: <FreqDist with 6793 samples and 39465
outcomes>
Out[2]: [('и', 1703), ('он', 892), ('что', 839),
('не', 798), ('сказал', 771), ('в', 736), ('я',
632), ('а', 616), ('пух', 594), ('на', 458)]
```

Распределение длин слов в тексте

FreqDist для длин

```
In[0]:fdist = nltk.FreqDist(len(w) for w in l1)
In[1]: print(fdist)
In[2]: print (fdist.most_common(5))
In[3]: print (fdist.freq(fdist.max()))
```

Гай Ричи

```
Out[1]:<...14 samples and 362 outcomes>
Out[2]:[(5, 56),(3, 45),(6, 40),(1, 39),(7, 37)]
Out[3]:0.1555555555555556
```

Винни Пух

```
Out[1]:<...23 samples and 39666 outcomes>
Out[2]:[(3, 6183),(5, 5275),(6, 5151),
(2, 4955),(1, 4517)]
Out[3]:0.15667046750285063
```

План на сегодня

1. segmentation
2. nltk.FreqDist
3. lemmatization
4. nltk.Text
5. Text Corpora

Леммы без стоп-слов (pymorphy)

pymorphy2 normal_form

```
In[1]: import pymorphy2
In[2]: morph = pymorphy2.MorphAnalyzer()
In[3]: from nltk.corpus import stopwords
In[4]: l3 = [morph.parse(token)[0].normal_form
for token in l1 if not token in
stopwords.words('russian')]
In[5]: d3 = nltk.FreqDist(l3)
In[6]: print (d3, d3.most_common(10))
```

Леммы без стоп-слов (pymorphy)

Вывод для "Гай Ричать"

```
<FreqDist with 154 samples and 224 outcomes>

[('фильм', 9),
 ('гай', 6),
 ('ричать', 6),
 ('криминальный', 4),
 ('снять', 4),
 ('свой', 4),
 ('карта', 3),
 ('деньга', 3),
 ('ствол', 3),
 ('культовый', 3)]
```

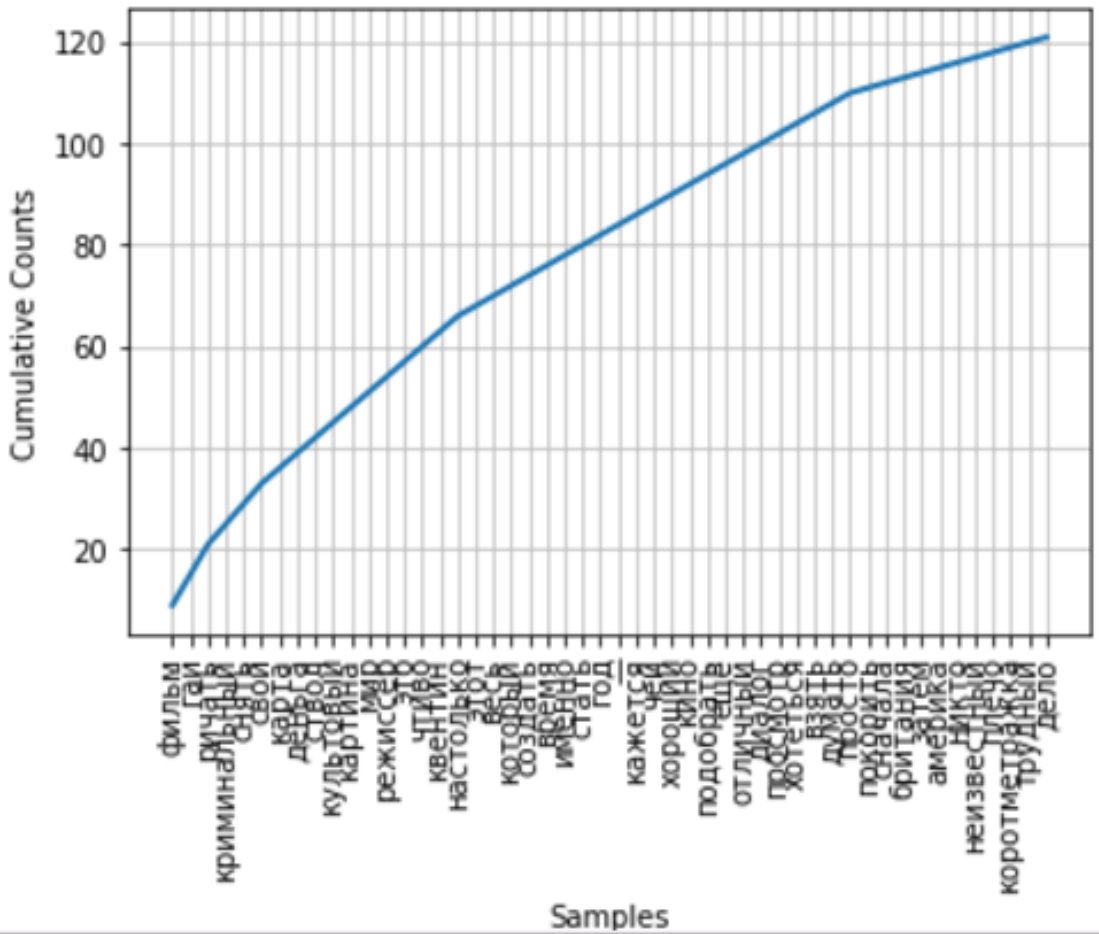
```
print (d3['картина'])
print (d3.freq('картина'))
```

```
3
0.013392857142857142
```

Леммы без стоп-слов (pymorphy)

most_common для "Гай Ричать"

```
d3.plot (50, cumulative = True)
```



Леммы без стоп-слов (pymorphy)

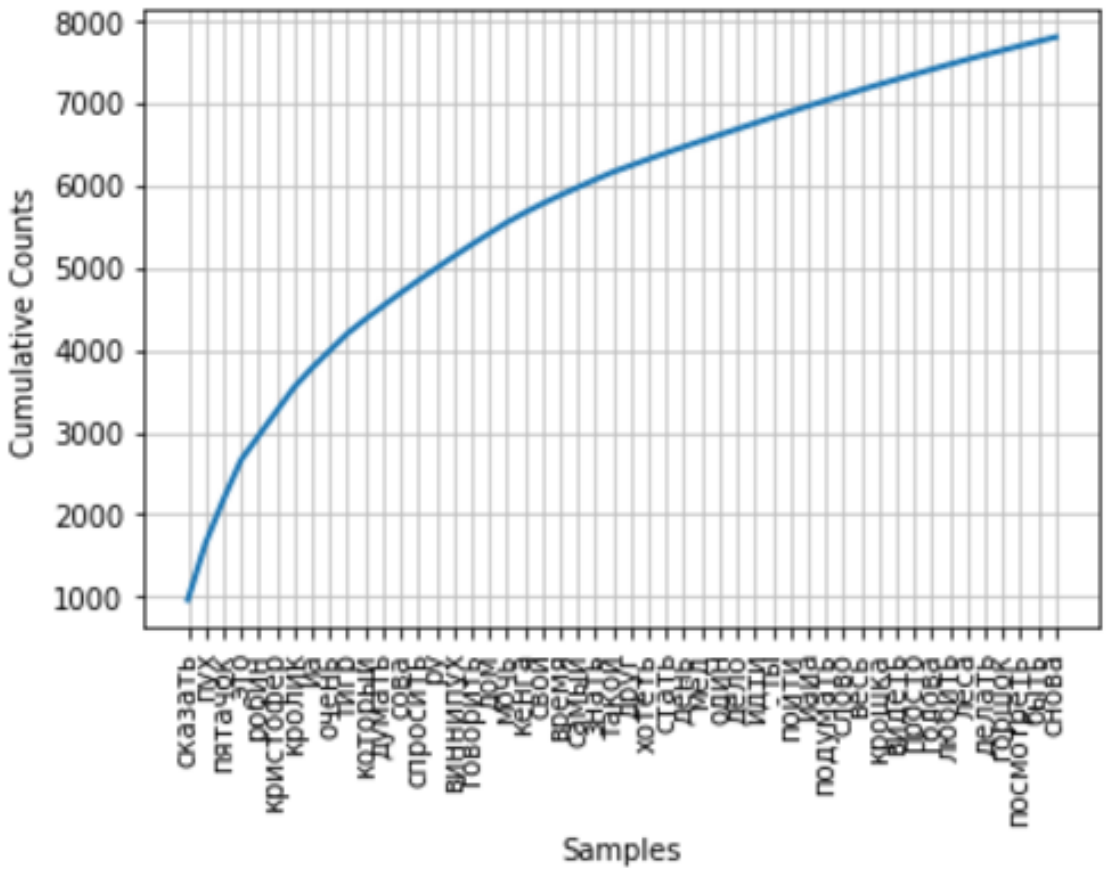
most_common для Винни Пуха

```
<FreqDist with 3916 samples and 23009 outcomes>

[('сказать', 963),
 ('пух', 683),
 ('пятачок', 521),
 ('это', 453),
 ('робин', 297),
 ('кристофер', 295),
 ('кролик', 288),
 ('иа', 229),
 ('очень', 208),
 ('тигр', 205)]
```

График для Винни Пуха

```
d3.plot (50, cumulative = True)
```



Mystem vs pymorphy

```
text = 'Кто знает Гая Ричи'
```

```
from pymorphy2 import MorphAnalyzer
m = MorphAnalyzer()
lemmas = [m.parse(word)[0].normal_form for word in text.split()]
print(' '.join(lemmas))
```

кто знать гай ричать

```
from pymystem3 import Mystem
m = Mystem()
lemmas = m.lemmatize(text)
print(''.join(lemmas))
```

кто знать гай ричи

План на сегодня

1. segmentation
2. nltk.FreqDist
3. lemmatization
4. nltk.Text
5. Text Corpora

re.compile VS nltk.word_tokenize

```
In[1]: t = "Слова? Да, больше слов? Что-то."
In[2]: prog = re.compile('[А-Яа-я\-]+')
In[3]: prog.findall(t)
Out[1]: ['Слова', 'Да', 'больше', 'слов',
'Что-то']
In[4]: nltk.word_tokenize(t)
Out[2]: ['Слова', '?', 'Да', ',', 'больше',
'слов', '?', 'Что-то', '.']
```

Превращаем .txt в NLTK-текст

```
f=open('data/winni.txt')
raw=f.read()
tokens = nltk.word_tokenize(raw)
text = nltk.Text(tokens)
```

```
print(text)
```

```
<Text:  Алан Александр Милн . Винни Пух и...>
```

text.concordance("сова")

```
Displaying 25 of 117 matches:
 именно там , в Дремучем Лесу , жила Сова . `` А если кто-нибудь что-нибудь о
онок про себя , - то это , конечно , Сова . Или я не Винни-Пух , - сказал он.-
ни-Пух.- Значит , все в порядке ! `` Сова жила в великолепном замке `` Каштаны
один во всем Лесу умел писать . Даже Сова , хотя она была очень-очень умная и
 и крикнул очень громким голосом : - Сова ! Открывай ! Пришел Медведь . Дверь
Пришел Медведь . Дверь открылась , и Сова выглянула наружу . - Здравствуй , Пу
как мне его найти ? - Ну , - сказала Сова , - обычная процедура в таких случая
сь сказать . - Я не чихала . - Нет , Сова , ты чихнула . - Прости , пожалуйста
точком меду ... - Ну вот , - сказала Сова , - мы , значит , напишем наше объяв
 стараться слушать то , что говорила Сова . А Сова говорила и говорила какие-т
я слушать то , что говорила Сова . А Сова говорила и говорила какие-то ужасно
`` нет '' на все , что бы ни сказала Сова . И так как в последний раз он говор
 - Как , ты их не видел ? - спросила Сова , явно удивившись.- Пойдем посмотрим
Красивый шнурок , правда ? - сказала Сова . Пух кивнул . - Он мне что-то напом
не нужен , я взяла его домой и ... - Сова , - сказал Пух торжественно , - он к
овы- а все в Лесу были уверены , что Сова прекрасно умеет писать , - он решил
айти к ней в гости . - Доброе утро , Сова ! - сказал Пух . - Доброе утро , Пух
```

```
text.similar('пошел')
```

отправился побежал подошел поплелся повернулся вернулся помчался что но то его так еще был ведь наверно я вышел поднялся лез

```
text.common_contexts(["ты", "я"])
```

а_думал нет_не сказал_не что_придумал разве_не что_говорил что_должен что_думал это_пятачок это_сказал это_не бы_не это_сам а_не что_не конечно_прав а_как а_сегодня что_хотел неужели_правда

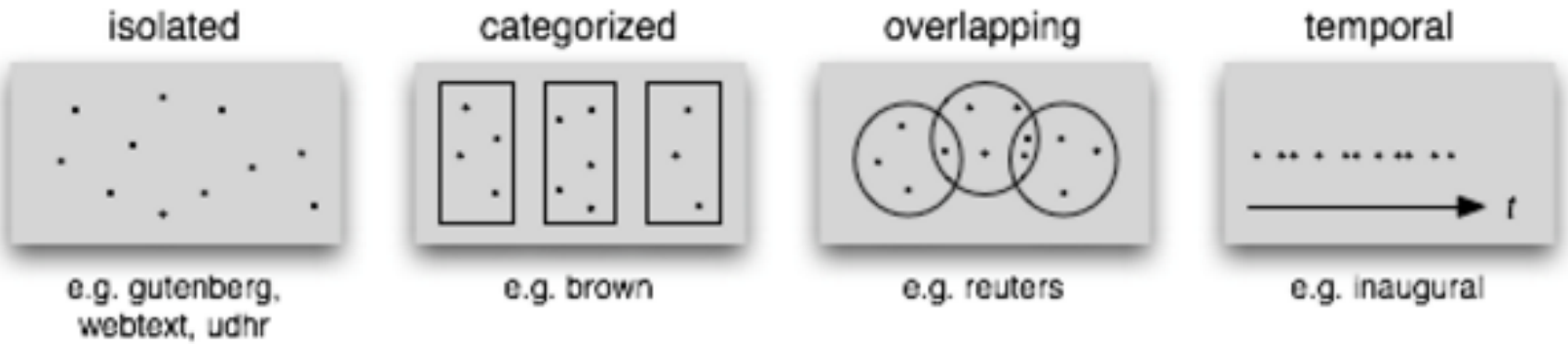
```
text.collocations(30)
```

Кристофер Робин; потому что; сказал Пух; Кристофера Робина; сказал Пух.-; может быть; как раз; сказал Кролик; Кристоферу Робину; про себя; сказала Сова; сказал Кристофер; Доброе утро; сказала Кенга; Кристофером Робином; сказал Пятачок.-; сказал Пятачок; днем рождения; Северный Полюс; сказал Кролик.-; как будто; изо всех; сказал Иа.-; всех сил; два раза; хочу сказать; Может быть; Маленькое Существо; что это; последнее время

План на сегодня

1. segmentation
2. nltk.FreqDist
3. lemmatization
4. nltk.Text
5. Text Corpora

Структура корпуса



Basic Corpus Functionality

Example	Description
`fileids()`	the files of the corpus
`fileids([categories])`	the files of the corpus corresponding to these categories
`categories()`	the categories of the corpus
`categories([fileids])`	the categories of the corpus corresponding to these files
`raw()`	the raw content of the corpus
`raw(fileids=[f1,f2,f3])`	the raw content of the specified files
`raw(categories=[c1,c2])`	the raw content of the specified categories
`words()`	the words of the whole corpus
`words(fileids=[f1,f2,f3])`	the words of the specified fileids
`words(categories=[c1,c2])`	the words of the specified categories
`sents()`	the sentences of the whole corpus
`sents(fileids=[f1,f2,f3])`	the sentences of the specified fileids
`sents(categories=[c1,c2])`	the sentences of the specified categories
`abspath(fileid)`	the location of the given file on disk
`encoding(fileid)`	the encoding of the file (if known)
`open(fileid)`	open a stream for reading the given corpus file
`root`	if the path to the root of locally installed corpus
`readme()`	the contents of the README file of the corpus

Gutenberg Corpus

```
from nltk.corpus import gutenberg
for fileid in gutenberg.fileids():
    num_words = len(gutenberg.words(fileid))
    num_sents = len(gutenberg.sents(fileid))
    print(num_words, num_sents,
          round(num_words/num_sents), fileid)
```

```
192427 7752 25 austen-emma.txt
98171 3747 26 austen-persuasion.txt
141576 4999 28 austen-sense.txt
1010654 30103 34 bible-kjv.txt
8354 438 19 blake-poems.txt
55563 2863 19 bryant-stories.txt
18963 1054 18 burgess-busterbrown.txt
34110 1703 20 carroll-alice.txt
96996 4779 20 chesterton-ball.txt
86063 3806 23 chesterton-brown.txt
69213 3742 18 chesterton-thursday.txt
210663 10230 21 edgeworth-parents.txt
```

Categories in Brown Corpus

```
from nltk.corpus import brown
genre_word = [(genre, word)
              for genre in ['news', 'romance']
              for word in brown.words(categories=genre)]
print (len(genre_word))
print (genre_word[:4])
print (genre_word[-104:-100])
```

Output

```
Out[1]: 170576
Out[2]: [('news', 'The'), ('news', 'Fulton'),
('news', 'County'), ('news', 'Grand')]
Out[3]: [('romance', 'and'), ('romance',
'Freddy'), ('romance', 'in'), ('romance',
'turn')]
```

Conditional FreqDist

```
cfd = nltk.ConditionalFreqDist(genre_word)
print (cfd.conditions())
print(cfd['news'])
print(cfd['romance'])
print (cfd['romance']['president'])
print (cfd['news']['president'])
```

Output

```
Out[1]: ['news', 'romance']
Out[2]: <FreqDist with 14394 samples and 100554
outcomes>
Out[3]: <FreqDist with 8452 samples and 70022
outcomes>
Out[4]: 9
Out[5]: 53
```

Собственный корпус

```
from nltk.corpus import PlaintextCorpusReader
corpus_root = 'data2/'
wordlists = PlaintextCorpusReader(corpus_root, '.*')
print (wordlists.fileids())
```

```
['Airedale.txt', 'American.txt', 'Bedlington.txt',
 'Bull.txt', 'IrishSoft.txt', 'JackRussell.txt', 'Pi
tbull.txt', 'Welsh.txt', 'Yorkshire.txt', 'australia
n.txt', 'fox.txt', 'hoffungs.txt', 'irish.txt', 'sco
ttish.txt', 'white.txt']
```

Работа с собственным корпусом

```
for fileid in wordlists.fileids():
    num_words = len(wordlists.words(fileid))
    num_sents = len(wordlists.sents(fileid))
    print(num_words, num_sents,
          round(num_words/num_sents), fileid)
```

```
1872 111 17 Airedale.txt
1779 117 15 American.txt
505 24 21 Bedlington.txt
877 49 18 Bull.txt
581 32 18 IrishSoft.txt
1460 87 17 JackRussell.txt
849 65 13 Pitbull.txt
538 22 24 Welsh.txt
3745 208 18 Yorkshire.txt
487 24 20 australian.txt
784 44 18 fox.txt
```

Упражнения

Скачайте файлы NLTK-задания.ipynb и alice.txt
и выполните задания.