Информационные ресурсы в финансовом мониторинге

НИЯУ МИФИ, КАФЕДРА ФИНАНСОВОГО МОНИТОРИНГА

КУРС ЛЕКЦИЙ

В.Ю. РАДЫГИН. ЛЕКЦИЯ 6

Решение некоторых задач

Введение

В данной лекции мы разберём решения некоторых задач, похожих на задачи курсовой работы и некоторых лабораторных работ.

Задача 1

В данной лекции мы разберём решения некоторых задач, похожих на задачи курсовой работы и некоторых лабораторных работ.

Напишите программу:

1) скачивающую с сайта bugs.python.org книгу по языку Python (https://bugs.python.org/file47781/Tutorial_EDIT.pdf) ;
2) выделяющую из файла содержательную текстовую часть книги;
3) выделяющую из текстовой части отдельные слова;
4) рассчитывающую дискретную функцию зависимости числа неповторяющихся слов (словаря) от общего числа слов текста;
5) строящую график данной зависимости.

Шаг 1. Загрузка файла

lection6_runner.py - C:\Users\Victor\Desktop\Students\InfRes\2024\Lection6\lection6_runner.py (3.11.2)

File Edit Format Run Options Window Help

```python
from urllib import request
from urllib.request import Request, urlopen

url = 'https://bugs.python.org/file47781/Tutorial_EDIT.pdf'

req = Request(url, headers={"User-Agent": "Mozilla/5.0"})
file = urlopen(req) # , context = ctx)
data = file.read()
file2 = open('py.pdf', 'wb+')
file2.write(data)
file2.close()

file.close()
```

Ln: 14 Col: 0

Шаг 1. Загрузка файла (текстом)

```
from urllib import request
from urllib.request import Request, urlopen

url = 'https://bugs.python.org/file47781/Tutorial_EDIT.pdf'

req = Request(url, headers={"User-Agent": "Mozilla/5.0"})
file = urlopen(req) # , context = ctx)
data = file.read()
file2 = open('py.pdf', 'wb+')
file2.write(data)
file2.close()

file.close()
```

Шаг 2. Выделение текста

lection6_runner.py - C:\Users\Victor\Desktop\Students\InfRes\2024\Lection6\lection6_runner.py (3.11.2)

File Edit Format Run Options Window Help

```python
from PyPDF2 import PdfReader

pdf_document = "py.pdf"
with open(pdf_document, "rb") as filehandle:
    pdf = PdfReader(filehandle)
    info = pdf.metadata
    pages = pdf.pages
    full_text = ''
    for i in range(8, 150):
        page = pdf.pages[i]
        text = page.extract_text()
        full_text += "\n" + text

print(full_text[0:5000])
```

Ln: 27 Col: 0

Шаг 2. Выделение текста (текстом)

```
from PyPDF2 import PdfReader


pdf_document = "py.pdf"
with open(pdf_document, "rb") as filehandle:
  pdf = PdfReader(filehandle)
  info = pdf.metadata
  pages = pdf.pages
  full_text = ''
  for i in range(8, 150):
     page = pdf.pages[i]
     text = page.extract_text()
     full_text += "\n" + text


print(full_text[0:5000])
```

Результат

IDLE Shell 3.11.2

File Edit Shell Debug Options Window Help

```
You could write a Unix shell script or Windows batch files for some of these tasks, but shell scripts are be
st
at moving around files and changing text data, not well-suited for GUI applications or games. You could
write a C/C++/Java program, but it can take a lot of development time to get even a first-draft program.
Python is simpler to use, available on Windows, Mac OS X, and Unix operating systems, and will help you
get the job done more quickly.
Python is simple to use, but it is a real programming language, offering much more structure and support
for large programs than shell scripts or batch files can offer. On the other hand, Python also offers much
more error checking than C, and, being a very-high-level language , it has high-level data types built in,
such
as flexible arrays and dictionaries. Because of its more general data types Python is applicable to a much
larger problem domain than Awk or even Perl, yet many things are at least as easy in Python as in those
languages.
Python allows you to split your program into modules that can be reused in other Python programs. It
comes with a large collection of standard modules that you can use as the basis of your programs — or as
examples to start learning to program in Python. Some of these modules provide things like file I/O, system
calls, sockets, and even interfaces to graphical user interface toolkits like Tk.
Python is an interpreted language, which can save you considerable time during program development
because no compilation and linking is necessary. The interpreter can be used interactively, which makes it
easy to experiment with features of the language, to write throw-away programs, or to test functions during
bottom-up program development. It is also a handy desk calculator.
Python enables programs to be written compactly and readably. Programs written in Python are typically
much shorter than equivalent C, C++, or Java programs, for several reasons:
•the high-level data types allow you to express complex operations in a single statement;
•statement grouping is done by indentation instead of beginning and ending brackets;
•no variable or argument declarations are necessary.
Python is extensible : if you know how to program in C it is easy to add a new built-in function or module
to the interpreter, either to perform critical operations at maximum speed, or to link Python programs to
libraries that may only be available in binary form (such as a vendor-specific graphics library). Once you
3
```

Ln: 39 Col: 65

Результат

Результат приемлемый, но некоторые моменты не очень красивые. Это проблема библиотеки **PyPDF2**. Для нашей задачи такой результат может и подойдёт, но рассмотрим альтернативу.

Используем другое средство – **PDFMiner**. Эта библиотека несколько сложнее, чем PyPDF2. Самым простым решением будет найти готовый пример её использования. Хорошим вариантом будет [1]. С небольшими изменениями данный пример даёт очень хороший результат.

На некоторых PDF PyPDF2 может плохо обрабатывать переносы…

Python 3.7.2 Shell

File Edit Shell Debug Options Window Help

```
THE FATF RECOMMENDATIONS
 INTERNATIONAL STANDARDS ON COMBATING
 MONEY LAUNDERING AND THE FINANCING OF TERRORISM & PROLIFERATION
 6   2012-2019
 INTRODUCTION
 The Financial Action Task Force (FATF) is an inter
-governmental body established in 1989 by the
Ministers of its Member jurisdictions. The mandate of the FATF is to set standards and to promote
effective implementation of legal, regulatory and operational
measures for combating money
laundering, terrorist financing and the financing of proliferation, and other related threats to the

integrity of the international financial system. In collaboration with other international
stakeholders, the FATF also works t
o identify national
 level vulnerabilities with the aim of
protecting the international financial system from misuse.
 The FATF Recommendations set out a comprehensive and consistent framework of measures which
countries should implement in order to combat
money laundering and terrorist financing, as well as
the financing of proliferation of weapons of mass destruction. Countries have diverse legal,
administrative and operational frameworks and different financial systems, and so cannot all take
identical me
asures to counter these threats. The FATF Recommendations, therefore, set an
international standard, which countries should implement through measures adapted to their

particular circumstances. The FATF Recommendations set out the essential measures that c
ountries
```

Ln: 6 Col: 4

Шаг 2. Выделение текста

lection6_runner.py - C:\Users\Victor\Desktop\Students\InfRes\2024\Lection6\lection6_runner.py (3.11.2)

File Edit Format Run Options Window Help

```python
from io import StringIO
from pdfminer.pdfinterp import PDFResourceManager, PDFPageInterpreter
from pdfminer.converter import TextConverter
from pdfminer.layout import LAParams
from pdfminer.pdfpage import PDFPage
import os
import sys, getopt

def convert(fname, pages=None):
    if not pages:
        pagenums = set()
    else:
        pagenums = set(pages)

    output = StringIO()
    manager = PDFResourceManager()
    converter = TextConverter(manager, output, laparams=LAParams())
    interpreter = PDFPageInterpreter(manager, converter)

    infile = open(fname, 'rb')
    for page in PDFPage.get_pages(infile, pagenums):
        interpreter.process_page(page)
    infile.close()
    converter.close()
    text = output.getvalue()
    output.close
    return text

text = convert("py.pdf", pages = range(8, 150))
print(text[0:5000])
```

Ln: 60 Col: 0

Шаг 2. Выделение текста (текстом)

```
from io import StringIO
from pdfminer.pdfinterp import PDFResourceManager, PDFPageInterpreter
from pdfminer.converter import TextConverter
from pdfminer.layout import LAParams
from pdfminer.pdfpage import PDFPage
import os
import sys, getopt


def convert(fname, pages=None):
    if not pages:
        pagenums = set()
    else:
        pagenums = set(pages)
    output = StringIO()
    manager = PDFResourceManager()
    converter = TextConverter(manager, output, laparams=LAParams())
    interpreter = PDFPageInterpreter(manager, converter)
    infile = open(fname, 'rb')
    for page in PDFPage.get_pages(infile, pagenums):
        interpreter.process_page(page)
    infile.close()
    converter.close()
    text = output.getvalue()
    output.close
    return text


text = convert("py.pdf", pages = range(6, 150))
print(text)
```

Результат

IDLE Shell 3.11.2

File Edit Shell Debug Options Window Help

```
at moving around files and changing text data, not well-suited for GUI applications or games. You could
write a C/C++/Java program, but it can take a lot of development time to get even a first-draft program.
Python is simpler to use, available on Windows, Mac OS X, and Unix operating systems, and will help you
get the job done more quickly.
Python is simple to use, but it is a real programming language, offering much more structure and support
for large programs than shell scripts or batch files can offer. On the other hand, Python also offers much
more error checking than C, and, being a very-high-level language, it has high-level data types built in, s
uch
as flexible arrays and dictionaries. Because of its more general data types Python is applicable to a much
larger problem domain than Awk or even Perl, yet many things are at least as easy in Python as in those
languages.
Python allows you to split your program into modules that can be reused in other Python programs. It
comes with a large collection of standard modules that you can use as the basis of your programs — or as
examples to start learning to program in Python. Some of these modules provide things like file I/O, system
calls, sockets, and even interfaces to graphical user interface toolkits like Tk.
Python is an interpreted language, which can save you considerable time during program development
because no compilation and linking is necessary. The interpreter can be used interactively, which makes it
easy to experiment with features of the language, to write throw-away programs, or to test functions during
bottom-up program development. It is also a handy desk calculator.
Python enables programs to be written compactly and readably. Programs written in Python are typically
much shorter than equivalent C, C++, or Java programs, for several reasons:

• the high-level data types allow you to express complex operations in a single statement;
• statement grouping is done by indentation instead of beginning and ending brackets;
• no variable or argument declarations are necessary.

Python is extensible: if you know how to program in C it is easy to add a new built-in function or module
to the interpreter, either to perform critical operations at maximum speed, or to link Python programs to
libraries that may only be available in binary form (such as a vendor-specific graphics library). Once you
```

Ln: 79 Col: 4

Шаг 3. Выделение слов

Для выделения слов для начала попробуем воспользоваться регулярными выражениями (модуль re).

Текст у нас английский поэтому мы будем искать все конструкции из букв, цифр и дефиса, содержащие хотя бы одну букву. Для простоты все буквы переведём в нижний регистр.

Результат будет большой, поэтому запишем его в файл.

Шаг 3. Выделение слов

lection6_runner.py - E:\Works\Victor\Students\infres\2020\Lection6\lection6_runner.py (3.7.2)

File Edit Format Run Options Window Help

```
text = text.lower()

import re

words = re.findall(r'\b(?:\w|-)*[qwertyuiopasdfghjklzxcvbnm](?:\w|-)*\b', text,
used_words = []
func = []
func.append(0)
for word in words:
    if word not in used_words:
        used_words.append(word)
    func.append(len(used_words))

file = open('text_re.txt', 'wb+')
file.write(' '.join(words).encode('utf-8'))
file.close()
```

Ln: 73 Col: 10

Шаг 3. Выделение слов (текстом)

```
text = text.lower()
import re
words = re.findall(r'\b(?:\w|-)*[qwertyuiopasdfghjklzxcvbnm](?:\w|-)*\b', text, flags=re.MULTILINE)
used_words = []
func = []
func.append(0)
for word in words:
    if word not in used_words:
        used_words.append(word)
    func.append(len(used_words))
file = open('text_re.txt', 'wb+')
file.write(' '.join(words).encode('utf-8'))
file.close()
```

Результат

text_re.txt – Блокнот

Файл Правка Формат Вид Справка

```
chapter one whetting your appetite if you do much work on computers eventually you find that there s some task you d like to automate for example you may wish to perform a sear
gui applications or games you could write a c c java program but it can take a lot of development time to get even a first-draft program python is simpler to use available on w
to program in python some of these modules provide things like file i o system calls sockets and even interfaces to graphical user interface toolkits like tk python is an inter
odule to the interpreter either to perform critical operations at maximum speed or to link python programs to libraries that may only be available in binary form such as a ven
atures of the python language and system through examples beginning with simple expressions statements and data types through functions and modules and finally touching upon ad
mmand prompt in a dos box set path path c program files python37 typing an end-of-file character control-d on unix control-z on windows at the primary prompt causes the interpr
ively when called with a file name argument or with a file as standard input it reads and executes a script from that file a second way of starting the interpreter is python c co
d in using-on-general argument passing when known to the interpreter the script name and additional arguments thereafter are turned into a list of strings and assigned to the
prints a welcome message stating its version number and a copyright notice before printing the first prompt python3 python default sep gcc on linux type help copyright credits
other than the default one a special comment line should be added as the first line of the file the syntax is as follows coding encoding where encoding is one of the valid codec
secondary prompt on a line by itself in an example means you must type a blank line this is used to end a multi-line command many of the examples in this manual even those ent
s straightforward the operators and work just like in most other languages for example pascal or c parentheses can be used for grouping for example division always returns a f
try to access an undefined variable traceback most recent call last file stdin line in module nameerror name n is not defined there is full support for floating point operators
s the j or j suffix to indicate the imaginary part e g 5j strings besides numbers python can also manipulate strings which can be expressed in several ways they can be enclosed
int isn t they said isn t they said s first line nsecond line n means newline s without print n is included in the output first line nsecond line print s with print n produces
sage message hostname to connect to produces the following output note that the initial newline is not included usage thingy options h h hostname display this usage message ho
ng index there is no separate character type a character is simply a string of size one word python word p word n character in position character in position indices may also
n excluded characters from position included to the end characters from the second-last included to the end one way to remember how slices work is to think of the indices as p
fully when used for slicing word on word python strings cannot be changed they are immutable therefore assigning to an indexed position in the string results in an error word
ether other values the most versatile is the list which can be written as a list of comma-separated values items between square brackets lists might contain items of different
ods later cubes append add the cube of cubes append and the cube of cubes assignment to slices is also possible and this can even change the size of the list or clear it entir
the variables a and b simultaneously get the new values and on the last line this is used again demonstrating that the expressions on the right-hand side are all evaluated firs
all decent text editors have an auto-indent facility when a compound statement is entered interactively it must be followed by a blank line to indicate completion since the pa
python chapter four more control flow tools besides the while statement just introduced python knows the usual control flow statements known from other languages with some twis
a string in the order that they appear in the sequence for example no pun intended measure some strings words cat window defenestrate for w in words cat window defenestrate pr
of length it is possible to let the range start at another number or to specify a different increment even negative sometimes this is called the step range range range to itera
uch an iterator the function list is another it creates lists from iterables list range later we will see more functions that return iterables and take iterables as argument b
lse clause of a try statement than it does that of if statements a try statement s else clause runs when no exception occurs and a loop s else clause runs when no break occurs
lace-holder for a function or conditional body when you are working on new code allowing you to keep thinking at a more abstract level the pass is silently ignored def initlog
tically produce online or printed documentation or to let the user interactively browse through code it s good practice to include docstrings in code that you write so make a
l symbol table is created for that call a function definition introduces the function name in the current symbol table the value of the function name has a type that is recogni
return fibonacci series up to n f100 fib2 call it f100 write the result see below result append a a b b a b return result this example as usual demonstrates some new python fe
see classes the method append shown in the example is defined for list objects it adds a new element at the end of the list in this example it is equivalent to result result a
es or no this example also introduces the in keyword this tests whether or not a sequence contains a certain value the default values are evaluated at the point of function de
uldn t action end print if you put voltage volts through it print lovely plumage the type print it s state accepts one required argument voltage and three optional arguments s
rtant this also includes non-optional arguments e g parrot voltage is valid too no argument may receive a value more than once here s an example that fails due to this restric
d like this cheeseshop limburger it s very runny sir it s really very very runny sir shopkeeper michael palin client john cleese sketch cheese shop sketch and of course it wou
```

Стр 1, стлб 1 | 100% | Windows (CRLF) | UTF-8

Шаги 4-5. График

lection6_runner.py - E:\Works\Victor\Students\infres\2020\Lection6\lection6_runner.py (3.7.2)

File Edit Format Run Options Window Help

```
print(len(func))
print(func[-1])
points_x = []
points_y = []
for i in range(0, len(func), 1):
    points_x.append(i)
    points_y.append(func[i])

import matplotlib.pyplot as plt
plt.get_current_fig_manager().window.wm_geometry('1400x750+50+50')
plt.scatter(points_x, points_y)
plt.show()
```

Ln: 73 Col: 10

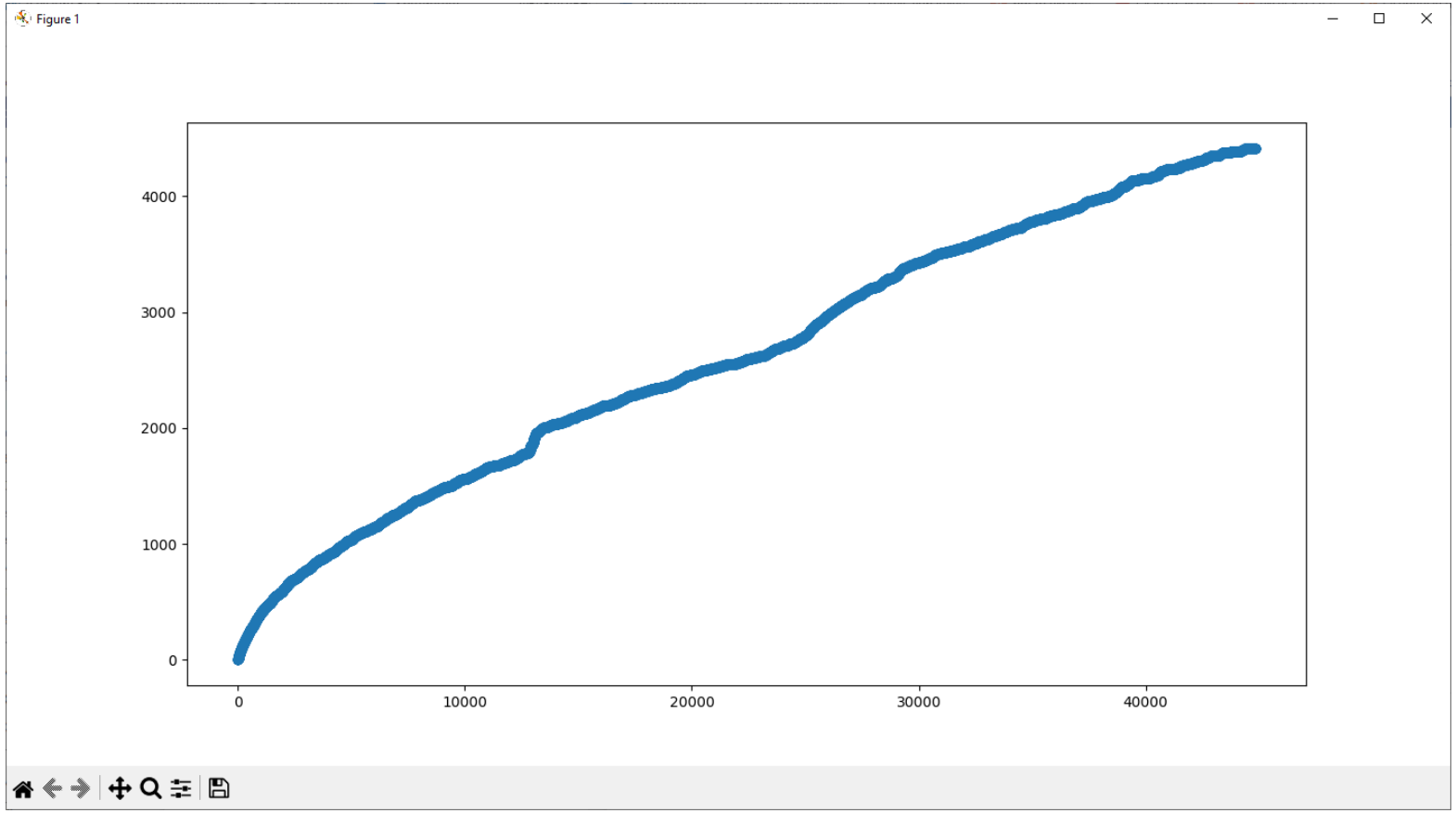
Шаги 4-5. График (текстом)

```
print(len(func))
print(func[-1])
points_x = []
points_y = []
for i in range(0, len(func), 1):
    points_x.append(i)
    points_y.append(func[i])

import matplotlib.pyplot as plt
plt.get_current_fig_manager().window.wm_geometry('1400x750+50+50')
plt.scatter(points_x, points_y)
plt.show()
```

Результат

На данном графике хорошо видна одна из ступенек роста числа слов по закону Хипса.

Тем не менее, результат нельзя считать качественным. Мы не удалили стоп слова. Мы не преобразовали слова в их нормальные формы. Мы учитывали в статистике имена, названия и т.д.



Используем NLTK

Решение на основе модуля re не является оптимальным. Для устранения недостатков данного решения мы будем использовать библиотеку **NLTK**. Для удобства, шаг выделения слов мы заменим «похожим» шагом, встроенным в NLTK – разбиением на токены.

Пакеты NLTK

NLTK содержит собственную пакетную систему. Если Вы установили NLTK – это ещё не значит, что все её функции будут работать. Может потребоваться дополнительная установка модулей. Данная операция в NLTK выполняется при помощи метода nltk.download.

Для выполнения токенизации (разбиения на токены) установим модуль пунктуации:

nltk.download('punkt')

nltk.download('punkt_tab')

Токенизация с помощью NLTK

lection6_runner.py - C:\Users\Victor\Desktop\Students\InfRes\2024\Lection6\lection6_runner.py (3.11.2)

File Edit Format Run Options Window Help

```
text = text.lower()

import nltk

# nltk.download('punkt')
# nltk.download('punkt_tab')

from nltk.tokenize import sent_tokenize, word_tokenize

words = word_tokenize(text)
```

Ln: 70 Col: 27

Токенизация с помощью NLTK (текстом)

```
text = text.lower()

import nltk

# nltk.download('punkt')
# nltk.download('punkt_tab')

from nltk.tokenize import sent_tokenize, word_tokenize

words = word_tokenize(text)
```

Удаление стопслов с помощью NLTK

К стопсловам в английском языке относятся различные частицы, местоимения и т.д. Например, can, my, his и т.д.

Для удаления стопслов установим соответствующий модуль:

nltk.download('stopwords')

Удаление стопслов с помощью NLTK

lection6_runner.py - C:\Users\Victor\Desktop\Students\InfRes\2024\Lection6\lection6_runner.py (3.11.2)

File Edit Format Run Options Window Help

```
# nltk.download('stopwords')

stop_words = set(stopwords.words('english'))

for w in words:
    if w not in stop_words:
        words_filtered.append(w)

file = open('text_nltk1.txt', 'wb+')
file.write(' '.join(words_filtered[0:1000]).encode('utf-8'))
file.close()
```

Ln: 75 Col: 2

Удаление стопслов (текстом)

```
stop_words = set(stopwords.words('english'))


for w in words:
    if w not in stop_words:
        words_filtered.append(w)


file = open('text_nltk1.txt', 'wb+')
file.write(' '.join(words_filtered).encode('utf-8'))
file.close()
```

Результат

text_nltk1.txt – Блокнот

Файл Правка Формат Вид Справка

```
chapter one whetting appetite much work computers , eventually find ’ task ’ like automate . example , may wish perform search-and-replace large number text files , rename rearr
nguage , offering much structure support large programs shell scripts batch files offer . hand , python also offers much error checking c , , very-high-level language , high-level
ivalent c , c++ , java programs , several reasons : • high-level data types allow express complex operations single statement ; • statement grouping done indentation instead b
. rest tutorial introduces various features python language system examples , beginning simple expressions , statements data types , functions modules , finally touching upon a
es interpreter exit zero exit status . ’ work , exit interpreter typing following command : quit ( ) . interpreter ’ line-editing features include interactive editing , histor
ingle quotes . python modules also useful scripts . invoked using python -m module [ arg ] ... , executes source file module spelled full name command line . script file used ,
v command module handle . 2.1.2 interactive mode commands read tty , interpreter said interactive mode . mode prompts next command primary prompt , usually three greater-than
```

Стр 1, стл6 6174 | 100% | Windows (CRLF) | UTF-8

Лишние токены

К сожалению, остались лишние токены – не слова, а числа или знаки пунктуации. Уберём их старым способом!

Удаление лишних токенов

lection6_runner.py - E:\Works\Victor\Students\infres\2020\Lection6\lection6_runner.py (3.7.2)

File Edit Format Run Options Window Help

```
words_filtered2 = []
for word in words_filtered:
    if re.match(r'^(?:\w|-)*[qwertyuiopasdfghjklzxcvbnm](?:\w|-)*$', word):
        words_filtered2.append(word)

file = open('text_nltk2.txt', 'wb+')
file.write(' '.join(words_filtered2[0:1000]).encode('utf-8'))
file.close()
```

Ln: 74 Col: 12

Удаление лишних токенов (текстом)

```
words_filtered2 = []
for word in words_filtered:
    if re.match(r'^(?:\w|-)*[qwertyuiopasdfghjklzxcvbnm](?:\w|-)*$', word):
        words_filtered2.append(word)


file = open('text_nltk2.txt', 'wb+')
file.write(' '.join(words_filtered2[0:1000]).encode('utf-8'))
file.close()
```

Результат

text_nltk2.txt – Блокнот

Файл Правка Формат Вид Справка

```
chapter one whetting appetite much work computers eventually find task like automate example may wish perform search-and-replace large number text files rename rearrange bunch p
checking c very-high-level language high-level data types built flexible arrays dictionaries general data types python applicable much larger problem domain awk even perl yet m
g brackets variable argument declarations necessary python extensible know program c easy add new built-in function module interpreter either perform critical operations maxim
two using python interpreter invoking interpreter python interpreter usually installed machines available putting unix shell search path makes possible start typing command si
and line editing available able use backspace remove characters current line interpreter operates somewhat like unix shell called standard input connected tty device reads exe
s assigned argv variable sys module access list executing import sys length list least one script arguments given empty string script name given meaning standard input set -c
d utf-8 languages world used simultaneously string literals identifiers comments although standard library uses ascii characters identifiers convention portable code follow disp
eans must type blank line used end multi-line command many examples manual even entered interactive prompt include comments comments python start hash character extend end phy
```

Стр 1, стлб 1 | 100% | Windows (CRLF) | UTF-8

Лемматизация

Лемматиза́ция — процесс приведения словоформы к **лемме** — её нормальной (словарной) форме (Википедия). Данный процесс позволит, например, считать слова has, have, had и т.д. одним и тем же словом.

К сожалению, одно и то же по написанию слово может быть в английском языке в разных ипостасях, например:

- для глагола stripes нормальная форма strip;
- для существительного stripes нормальная форма stripe.

Определить правильно назначение слова можно только из контекста. В данном лекции мы не будем учитывать контекст. Подробнее об этом и о лемматизации можно прочесть в [2].

Для удаления лемматизации без указания части речи слова установим модуль:

nltk.download('wordnet')

Тестируем лемматизацию

lection6_runner.py - E:\Works\Victor\Students\infres\2020\Lection6\lection6_runner.py (3.7.2)

File Edit Format Run Options Window Help

```
# nltk.download('wordnet')

from nltk.stem import WordNetLemmatizer

lemmatizer = WordNetLemmatizer()

print(lemmatizer.lemmatize("feet"))
print(lemmatizer.lemmatize("are"))
```

Ln: 82 Col: 0

Тестируем лемматизацию (текстом)

```
# nltk.download('wordnet')


from nltk.stem import WordNetLemmatizer

lemmatizer = WordNetLemmatizer()

print(lemmatizer.lemmatize("feet"))

print(lemmatizer.lemmatize("are"))
```

Результат

```
*Python 3.7.2 Shell*
File  Edit  Shell  Debug  Options  Window  Help
Python 3.7.2 (tags/v3.7.2:9a3ffc0492, Dec 23 2018, 23:09:28) [MSC v.1916 64 bit
(AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
==== RESTART: E:\Works\Victor\Students\infres\2020\Lection6\lection6en.py ====
foot
are
Ln: 11  Col: 0
```

Без указания части речи лемматизатор не справился с глаголом are.

Лемматизация с частями речи

Для правильной обработки речи перед началом лемматизации к токенам обавляют служебную информацию – тэги. В нашем случае, нас будет интересовать добавление информации о том, к какой части речи относится слово.

Установим тэггер:

nltk.download('averaged_perceptron_tagger’)

nltk.download('averaged_perceptron_tagger_eng')

Функцию для обработки тэгов возьмём из [2].

Тестируем лемматизацию

lection6_runner.py - C:\Users\Victor\Desktop\Students\InfRes\2024\Lection6\lection6_runner.py (3.11.2)

File Edit Format Run Options Window Help

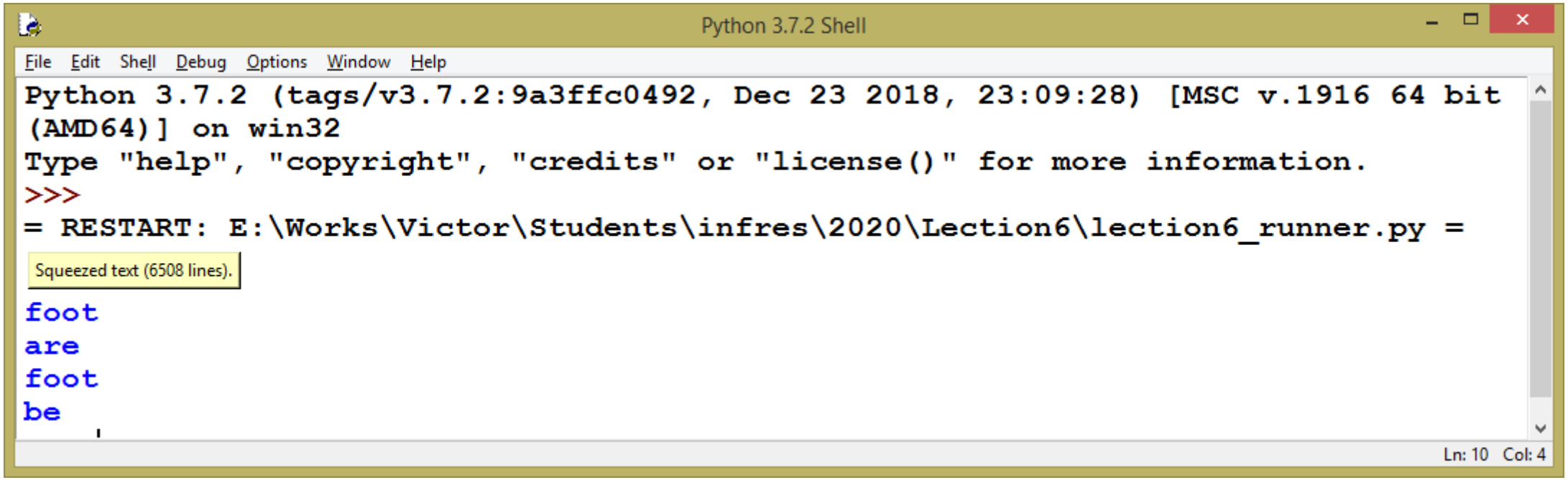
```
# nltk.download('averaged_perceptron_tagger')
# nltk.download('averaged_perceptron_tagger_eng')

import nltk
from nltk.corpus import wordnet
def get_wordnet_pos(word):
    tag = nltk.pos_tag([word])[0][1][0].upper()
    tag_dict = {"J": wordnet.ADJ,
                "N": wordnet.NOUN,
                "V": wordnet.VERB,
                "R": wordnet.ADV}

    return tag_dict.get(tag, wordnet.NOUN)

lemmatizer = WordNetLemmatizer()

print(lemmatizer.lemmatize("feet", get_wordnet_pos("feet")))
print(lemmatizer.lemmatize("are", get_wordnet_pos("are")))
```

Ln: 125 Col: 0

Тестируем лемматизацию (текстом)

```
import nltk
from nltk.corpus import wordnet
def get_wordnet_pos(word):
    tag = nltk.pos_tag([word])[0][1][0].upper()
    tag_dict = {"J": wordnet.ADJ,
                "N": wordnet.NOUN,
                "V": wordnet.VERB,
                "R": wordnet.ADV}
    return tag_dict.get(tag, wordnet.NOUN)
lemmatizer = WordNetLemmatizer()
print(lemmatizer.lemmatize("feet", get_wordnet_pos("feet")))
print(lemmatizer.lemmatize("are", get_wordnet_pos("are")))
```

Результат



Результат гораздо лучше!

Нормализация слов с помощью NLTK

lection6_runner.py - E:\Works\Victor\Students\infres\2020\Lection6\lection6_runner.py (3.7.2)

File Edit Format Run Options Window Help

```
base_words = []
for word in words_filtered2:
    bword = lemmatizer.lemmatize(word, get_wordnet_pos(word))
    base_words.append(bword)

file = open('text_nltk3.txt', 'wb+')
file.write(' '.join(base_words).encode('utf-8'))
file.close()
```

Ln: 112 Col: 12

Нормализация слов (текстом)

```
base_words = []
for word in words_filtered2:
    bword = lemmatizer.lemmatize(word, get_wordnet_pos(word))
    base_words.append(bword)


file = open('text_nltk3.txt', 'wb+')
file.write(' '.join(base_words).encode('utf-8'))
file.close()
```

Результат

text_nltk3.txt – Блокнот

Файл Правка Формат Вид Справка

```
chapter one whet appetite much work computer eventually find task like automate example may wish perform search-and-replace large number text files rename rearrange bunch photo
h-level data type built flexible array dictionary general data type python applicable much large problem domain awk even perl yet many thing least easy python language python a
w built-in function module interpreter either perform critical operation maximum speed link python program library may available binary form vendor-specific graphic library pyt
e command since choice directory interpreter life installation option place possible check local python guru system administrator popular alternative location window machine p
rgument file standard input read executes script file second way start interpreter python -c command arg executes statement command analogous shell -c option since python statem
module consume python interpreter option processing left command module handle interactive mode command read tty interpreter say interactive mode mode prompt next command prim
ial comment line add first line file syntax follow cod encode encode one valid codecs support python chapter use python interpreter python tutorial release example declare windo
```

Стр 1, стлб 1 | 100% | Windows (CRLF) | UTF-8

Шаги 4-5. График (заново)

lection6_runner.py - E:\Works\Victor\Students\infres\2020\Lection6\lection6_runner.py (3.7.2)

File Edit Format Run Options Window Help

```
used_words = []
func = []
func.append(0)
for word in base_words:
    if word not in used_words:
        used_words.append(word)
    func.append(len(used_words))

print(len(func))
print(func[-1])
points_x = []
points_y = []
for i in range(0, len(func), 1):
    points_x.append(i)
    points_y.append(func[i])

import matplotlib.pyplot as plt
plt.get_current_fig_manager().window.wm_geometry('1400x750+50+50')
plt.scatter(points_x, points_y)
plt.show()
```

Ln: 133 Col: 10

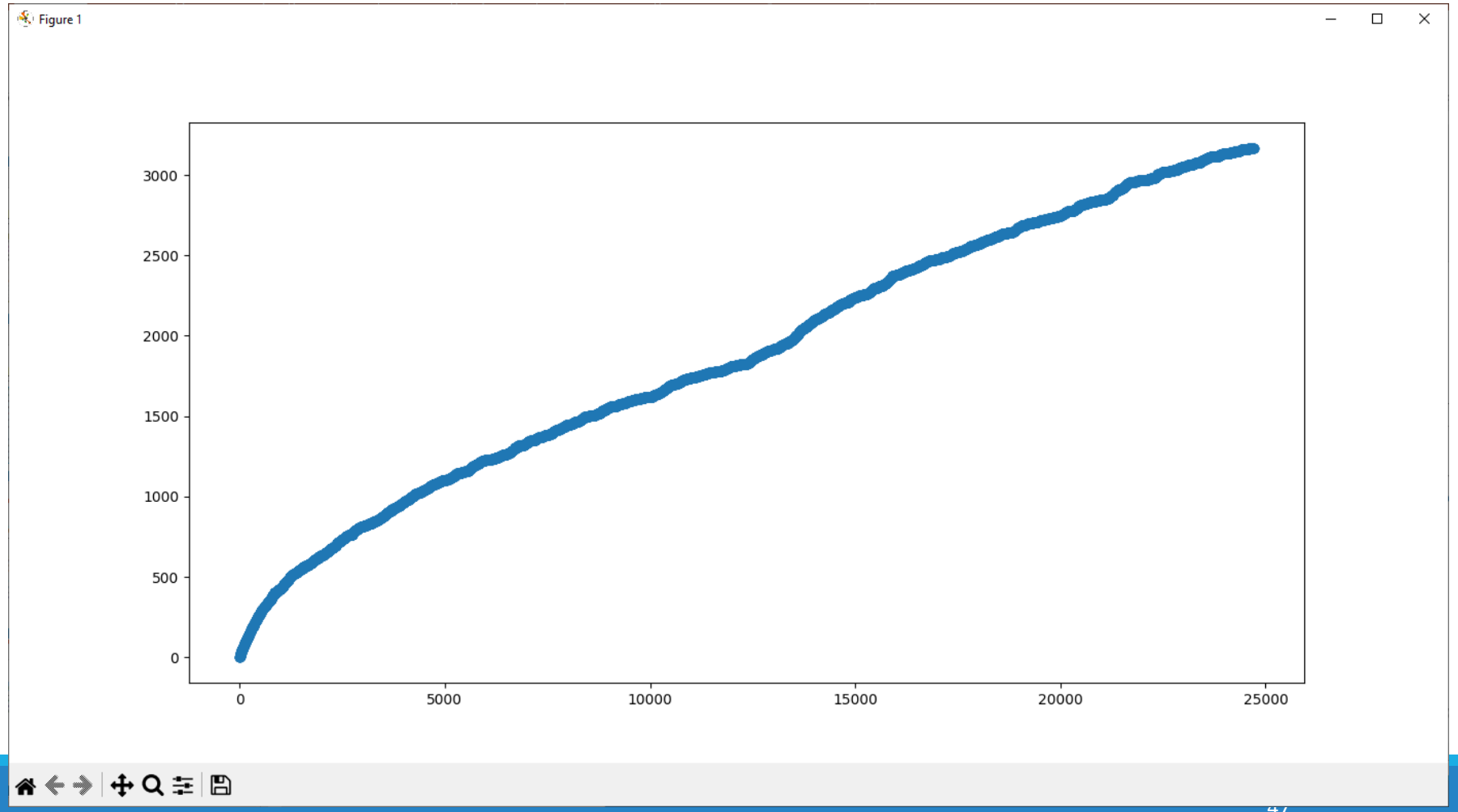
Шаги 4-5. График (текстом)

```
used_words = []
func = []
func.append(0)
for word in base_words:
    if word not in used_words:
        used_words.append(word)
    func.append(len(used_words))
print(len(func))
print(func[-1])
points_x = []
points_y = []
```

```
for i in range(0, len(func), 1):
    points_x.append(i)
    points_y.append(func[i])


import matplotlib.pyplot as plt
plt.get_current_fig_manager().window.wm_geometry('1400x750+50+50')
plt.scatter(points_x, points_y)
plt.show()
```

Результат

Общее число неповторяющихся слов уменьшилось, но поведение осталось старым.



Война и мир

Проверим наш подход на художественном произведении. На сайте

https://www.gutenberg.org/files/2600/2600-h/2600-h.htm

доступно произведение «Война и мир» Л.Н. Толстого на английском языке.

Предположим, что мы скачали содержимое глав данного произведения в файл wap.txt.

Война и мир подготовка

lection6_runner.py - E:\Works\Victor\Students\infres\2020\Lection6\lection6_runner.py (3.7.2)

File Edit Format Run Options Window Help

```
file = open('wap.txt', 'r')
text = file.read()
text = text.lower()
file.close()

words = word_tokenize(text)
words_filtered = []
for w in words:
    if w not in stop_words:
        if re.match(r'^(?:\w|-)*[qwertyuiopasdfghjklzxcvbnm](?:\w|-)*$', w):
            words_filtered.append(lemmatizer.lemmatize(w, get_wordnet_pos(w)))
```

Ln: 163 Col: 78

Война и мир подготовка (текстом)

```
file = open('wap.txt', 'r')
text = file.read()
text = text.lower()
file.close()


words = word_tokenize(text)
words_filtered = []
for w in words:
    if w not in stop_words:
        if re.match(r'^(?:\w|-)*[qwertyuiopasdfghjklzxcvbnm](?:\w|-)*$', w):
            words_filtered.append(lemmatizer.lemmatize(w, get_wordnet_pos(w)))
```

Война и мир анализ

lection6_runner.py - E:\Works\Victor\Students\infres\2020\Lection6\lection6_runner.py (3.7.2)

File Edit Format Run Options Window Help

```
used_words = []
func = []
func.append(0)
for word in words_filtered:
    if word not in used_words:
        used_words.append(word)
    func.append(len(used_words))

print(len(func))
print(func[-1])
points_x = []
points_y = []
for i in range(0, len(func), 1):
    points_x.append(i)
    points_y.append(func[i])

plt.get_current_fig_manager().window.wm_geometry('1400x750+50+50')
plt.scatter(points_x, points_y)
plt.show()

print(used_words[-1000:-1])
```

Ln: 185 Col: 27

Война и мир анализ (текстом)

```
used_words = []
func = []
func.append(0)
for word in words_filtered:
    if word not in used_words:
        used_words.append(word)
    func.append(len(used_words))
print(len(func))
print(func[-1])
points_x = []
points_y = []
```

```
for i in range(0, len(func), 1):
    points_x.append(i)
    points_y.append(func[i])

plt.get_current_fig_manager().window.wm_geometry('1400x750+50+50')
plt.scatter(points_x, points_y)
plt.show()

print(used_words[-1000:-1])
```

Посмотрим словарь последних шагов

Посмотрим, какие слова добавились в словарь Войны и мира в конце.

Python 3.7.2 Shell

File Edit Shell Debug Options Window Help

```
['kiril', 'sokol', 'permanent', 'platoche', 'convenience', 'soot', 'rounder', 'h
andiwork', 'assignation', 'squeaky', 'smirch', 'positive', 'reservation', 'super
fluity', 'destroys', 'insolubly', 'daydream', 'hardship', 'alertness', 'emaciati
on', 'dysentery', 'semidark', 'dram-da-da-dam', 'dam-dam', 'mistrustfully', 'ide
ntification', 'blasius', 'kham', 'vniki', 'unburned', 'heathen', 'paling', 'nesk
', 'chny', 'beauharnais', 'debouch', 'ord', 'transmoskv', 'converge', 'crossway'
, 'setting', 'bad-looking', 'wench', 'rouge', 'swaggeringly', 'colic', 'roughnes
s', 'reaction', 'worsen', 'immortal', 'oscillate', 'limitless', 'rokhov', 'brous
sier', 'russo-french', 'augezd', 'paroxysm', 'undiscerning', 'prose', 'testimony
', 'working', 'revolves', 'arist', 'figner', 'sesl', 'vin', 'rovsk', 'bolkhov',
```

Ln: 42 Col: 4

Очень много имён и непонятных слов русского или французского языков.

Корпус языка

При исследовании словаря текста не очень интересно изучать географические названия, имена, слова из других языков и т.д. Лучше всего их убрать. Простого способа нет. Но мы попробуем для этой цели использовать корпус английского языка.

Установим словарь слов:

nltk.download('words’)

Затем используем его, как фильтр. В дальнейшем построим уже известным нам способом график.

Война и мир новый фильтр

lection6_runner.py - E:\Works\Victor\Students\infres\2020\Lection6\lection6_runner.py (3.7.2)

File Edit Format Run Options Window Help

```
# nltk.download('words')

lang_words = set(nltk.corpus.words.words())
words_filtered = []
for w in words:
    if w not in stop_words and w in lang_words:
        if re.match(r'^(?:\w|-)*[qwertyuiopasdfghjklzxcvbnm](?:\w|-)*$', w):
            words_filtered.append(lemmatizer.lemmatize(w, get_wordnet_pos(w)))
```

Ln: 194 Col: 78

Война и мир новый фильтр (текстом)

```
# nltk.download('words')


lang_words = set(nltk.corpus.words.words())
words_filtered = []
for w in words:
    if w not in stop_words and w in lang_words:
        if re.match(r'^(?:\w|-)*[qwertyuiopasdfghjklzxcvbnm](?:\w|-)*$', w):
            words_filtered.append(lemmatizer.lemmatize(w, get_wordnet_pos(w)))
```

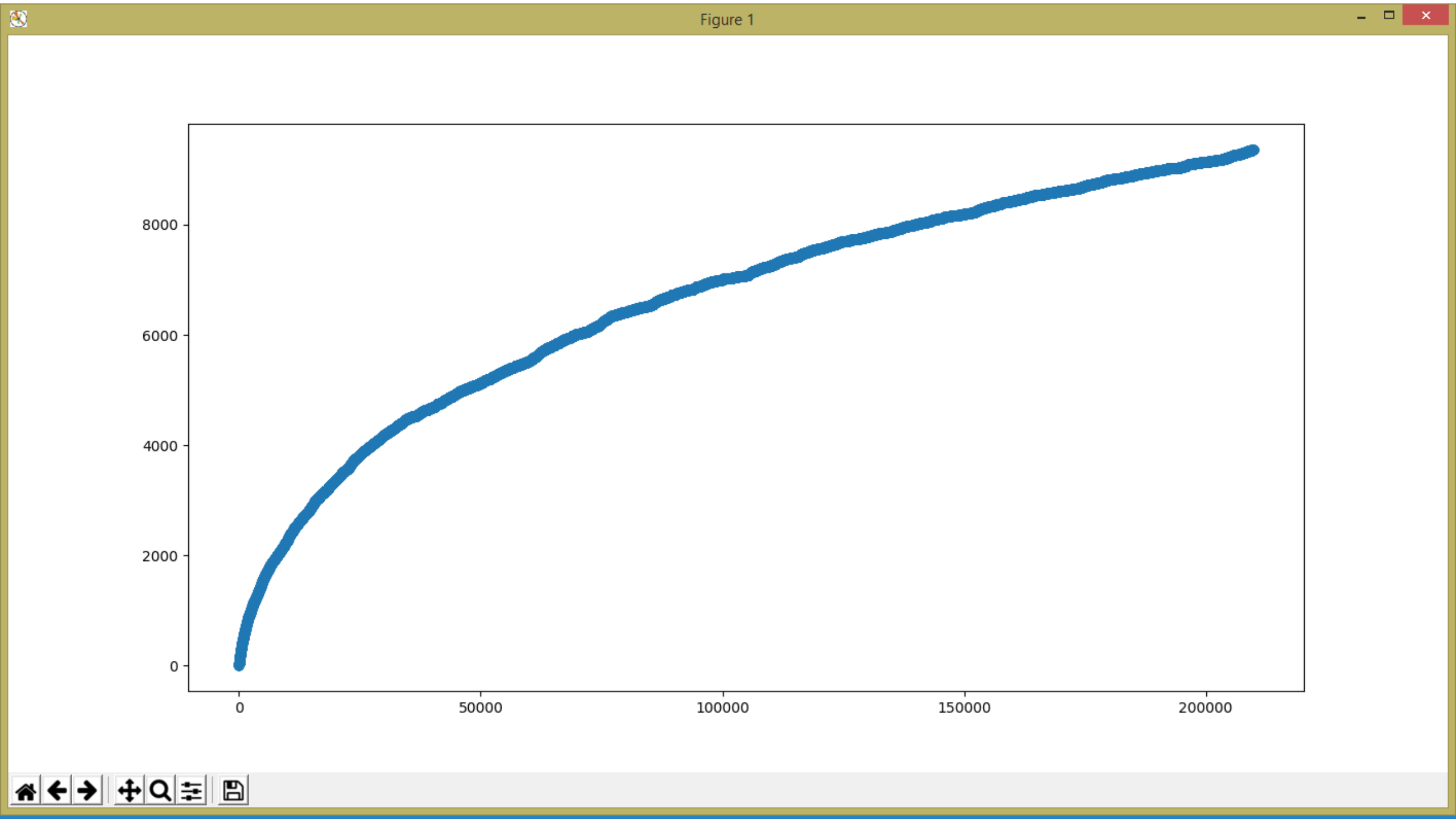


Figure 1
8000
6000
4000
2000
0
0
50000
100000
150000
200000

Где ступени?

Почему нет ступеней? Язык Л.Н. Толстого очень богат. Прием сцены Войны и мир плавно перемещаются, дополняясь новыми событиями и местами. Вдобавок, свой вклад внёс и литературный перевод на английский язык. Для хорошего автора в литературном произведении может и не быть «лестницы».

Проверим для Шарлотты Бронте. На странице

https://www.gutenberg.org/files/1260/1260-h/1260-h.htm

Можно найти текст произведения Джейн Эйр. Проверим его.

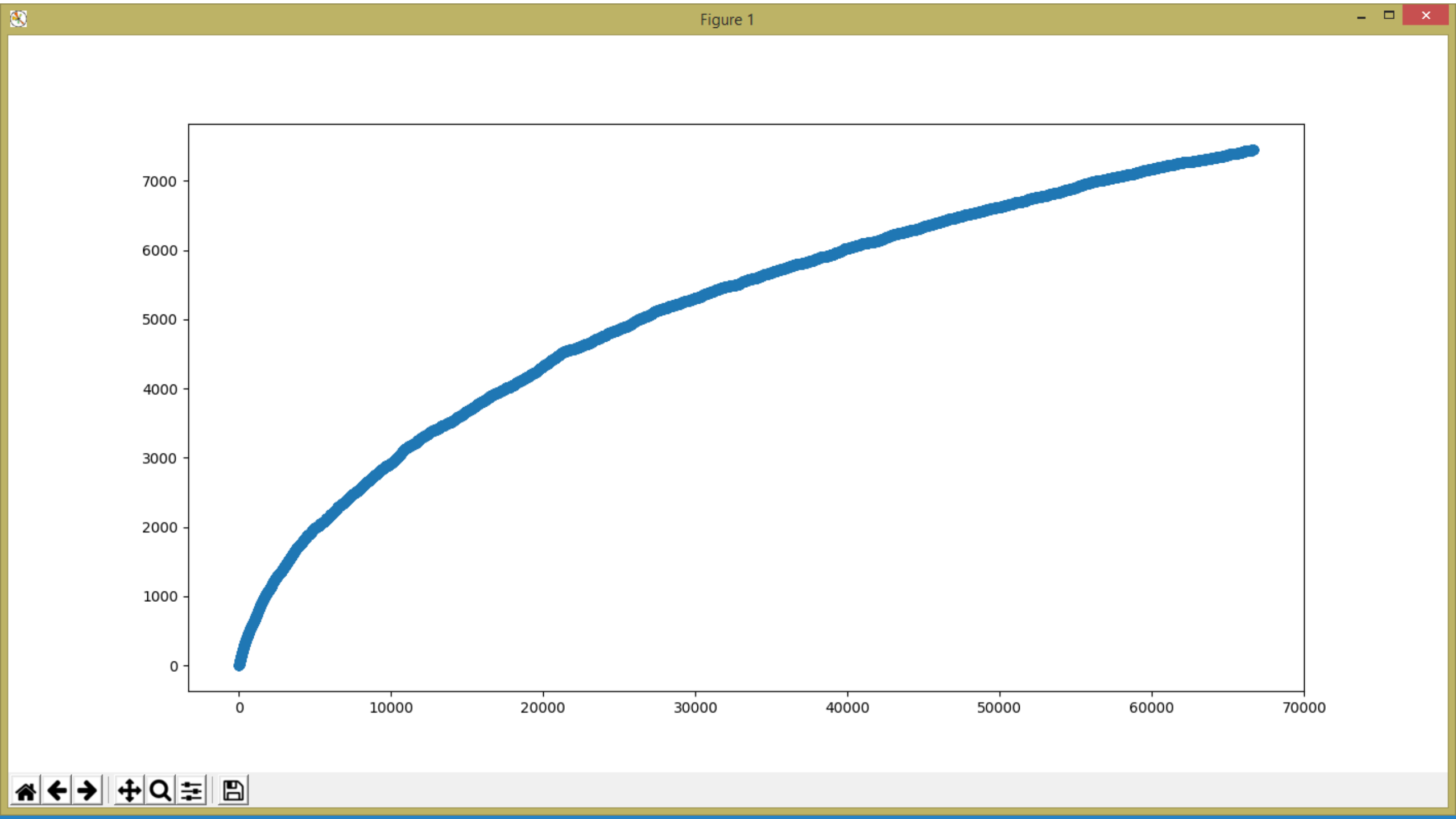


Figure 1
7000
6000
5000
4000
3000
2000
1000
0
0
10000
20000
30000
40000
50000
60000
70000

Где ступени?

Ступеней по прежнему нет. Неужели Хипс ошибся?

Возьмём нелитературный текст. Хороший пример – конституция Великобритании. Её можно скачать на странице

https://www.ippr.org/files/images/media/files/publication/2014/01/the-constitution-of-the-united-kingdom_1991-2014_1420.pdf

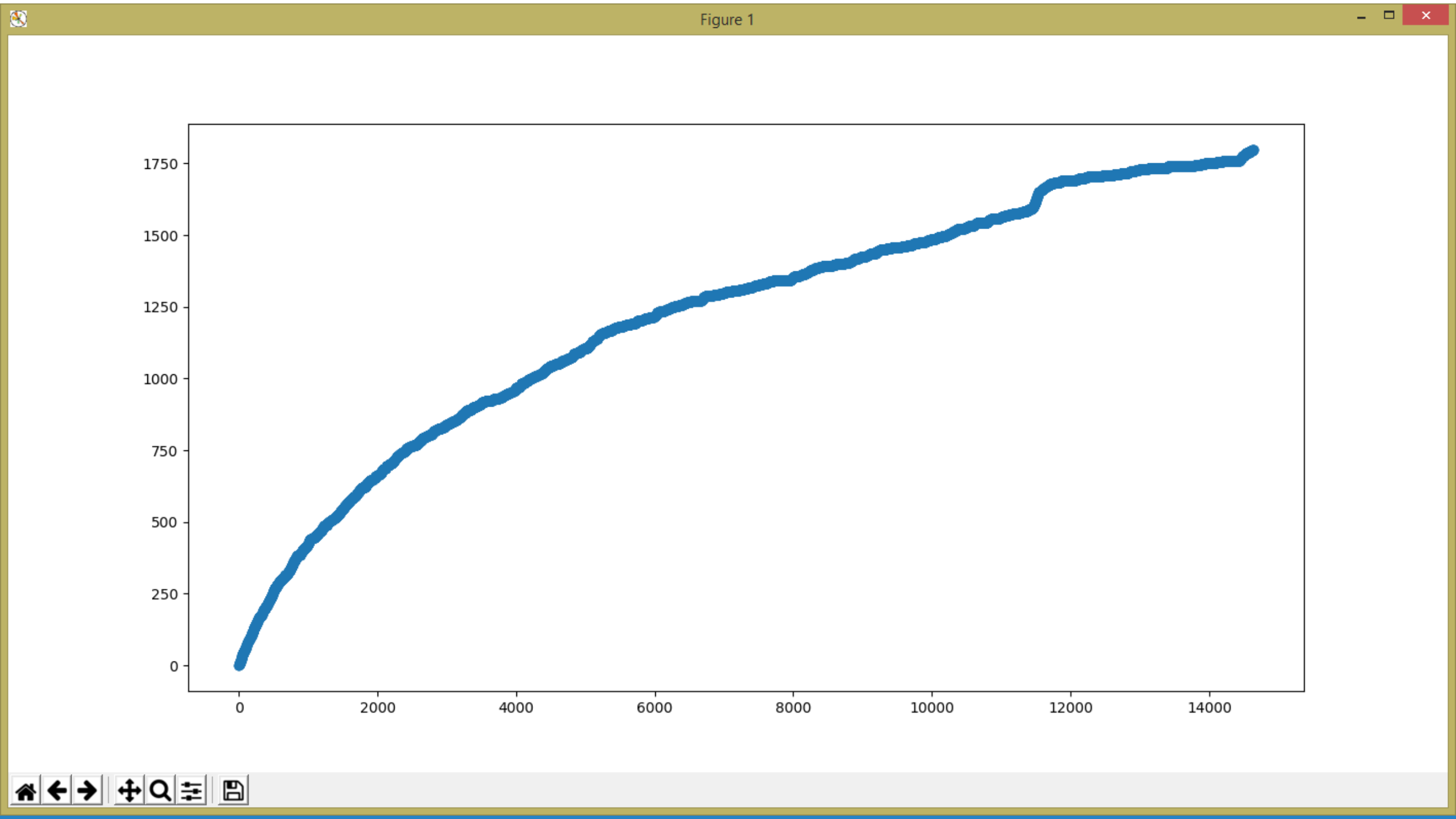


Figure 1
1750
1500
1250
1000
750
500
250
0
0
2000
4000
6000
8000
10000
12000
14000

Ступеньки есть!

Текст конституции хорошо показал разницу между известным классиком-литератором и реальными текстами. В реальных текстах закон Хипса успешно выполняется!

Полезные ссылки

1. https://stackoverflow.com/questions/55220455/convert-from-pdf-to-text-lines-and-words-are-broken
2. https://www.machinelearningplus.com/nlp/lemmatization-examples-python/

Полезные ссылки

7. https://pypi.org/project/python-Levenshtein/ – страница библиотеки python-Levenshtein на сайте pypi.