2

Установление авторства с помощью стилометрии

Стилометрия — это статистический анализ стилистики посредством компьютерного анализа текста. В ее основе лежит тот факт, что у каждого человека есть уникальный, устойчивый и распознаваемый стиль письма. К нему относятся используемый лексикон, нюансы пунктуации, средняя длина предложений и слов, а также другие характеристики.

Одна из распространенных сфер применения стилометрии — установление авторства. Вы когда-нибудь задумывались, а действительно ли именно Шекспир написал все пьесы, принадлежащие, как считается, его перу? Или кто был автором песни «In my life» — Джон Леннон или Пол Маккартни? А может быть, Роберт Гэлбрейт, автор книги «Зов кукушки», — это псевдоним писательницы Дж. К. Роулинг? Стилометрия способна дать ответы на такие вопросы.

Эта техника применяется для снятия обвинений в убийстве и даже помогла вычислить и осудить Унабомбера[1] в 1996 году. Ну и, конечно же, она повсеместно используется для выявления плагиата и определения эмоциональной окраски

[1] Американский математик, социальный критик, философ, террорист и неолуддит, известный своей кампанией по рассылке бомб почтой. — *Примеч. ред.*

слов, например в постах в социальных сетях. Стилометрия даже годится для обнаружения признаков депрессии и суицидальных наклонностей.

В этой главе мы с вами при помощи техник стилометрии определим, кто является автором романа «Затерянный мир» — Артур Конан Дойл или Г. Д. Уэллс.

Проект #2: «Собака Баскервилей», «Война миров» и «Затерянный мир»

Сэр Артур Конан Дойл (1859 — 1930) больше всего известен историями о Шерлоке Холмсе, которые считаются эталоном в детективном жанре. Г. Д. Уэллс (1866 — 1946) прославился несколькими новаторскими научно-фантастическими романами — «Война миров», «Машина времени», «Человек-невидимка» и «Остров доктора Моро».

В 1912 году *Strand Magazine* опубликовал «Затерянный мир», серийную версию научно-фантастического романа. Это история экспедиции в бассейн Амазонки под предводительством профессора зоологии Джорджа Эдварда Челленджера, который утверждал, что там водятся динозавры. Собственно, их он там и обнаружил, а также племя злобных обезьяноподобных существ.

Несмотря на то что в реальности автор романа хорошо известен, давайте предположим, что это не так, и наша задача — установить его личность. Эксперты сузили область поиска до двух имен, Дойл и Уэллс. Уэллс выглядит предпочтительнее, потому что «Затерянный мир» относится к жанру научной фантастики, который для этого автора характерен. При этом в книге также встречаются жестокие пещерные люди, напоминающие морлоков из его же романа «Машина времени» (1895). Дойл же, наоборот, известен своими детективными историями и исторической фантастикой.

ЗАДАЧА

Написать на Python программу, использующую стилометрию, чтобы определить, кто является автором романа «Затерянный мир» — Артур Конан Дойл или Г. Д. Уэллс.

Стратегия

Дисциплина *обработка естественного языка (natural language processing, NLP)* занимается взаимодействиями между точным и структурированным языком

компьютеров и специфичным, зачастую неоднозначным «естественным» языком, используемым людьми. NLP применяется для машинного перевода, определения спама, понимания поисковых запросов, а также предиктивного распознавания текста для пользователей сотовых телефонов.

Наиболее типичными тестами NLP на авторство анализируются следующие особенности текста.

- **Длина слов.** График распределения частотности длин слов в документе.
- **Стоп-слова.** График распределения частотности стоп-слов (короткие, внеконтекстные функциональные слова вроде *the*, *but*, *if*).
- **Части речи.** График распределения частотности слов на основе их синтаксической роли (существительные, местоимения, глаголы, обстоятельства, определения и пр.).
- **Наиболее распространенные слова.** Сравнение наиболее часто встречающихся в тексте слов.
- **Коэффициент Жаккара.** Статистика, используемая для оценки сходства и разнообразия выборки.

Если стили письма Дойла и Уэллса отличаются, тогда этих пяти тестов будет достаточно, чтобы выявить их различие. Более подробно о каждом из тестов мы поговорим, когда будем писать код.

Чтобы охватить и проанализировать стиль автора, потребуется образец *корпуса*, иначе говоря — тела текста. Для определения стиля Дойла мы используем известный роман о Шерлоке Холмсе «Собака Баскервилей», опубликованный в 1902 году. Для Уэллса же возьмем книгу «Война миров», выпущенную в 1898 году. Каждый из этих романов содержит более 50 000 слов, чего вполне хватит для репрезентативной статистической выборки. После этого мы сравним выборку каждого автора с произведением *«Затерянный мир»*, чтобы определить, какой стиль для него ближе.

Сам процесс стилометрии будем выполнять с помощью *Natural Language Toolkit (NLTK)*, популярного набора программ и библиотек для работы с данными на естественном языке в Python. Он бесплатный и работает в Windows, macOS, а также Linux. NLTK был разработан в 2001 году в рамках курса компьютерной лингвистики в Университете Пенсильвании и далее развивался усилиями десятков добровольных участников. Более подробно можете узнать об этом проекте на сайте http://www.nltk.org/.

Установка NLTK

Инструкции по установке вы найдете на сайте http://www.nltk.org/install.html. В случае с Windows откройте PowerShell и установите пакет с помощью Preferred Installer Program (pip).

```
python -m pip install nltk
```

Если у вас установлено несколько версий Python, необходимо указать версию. Вот команда для Python 3.7:

```
py -3.7 -m pip install nltk
```

Чтобы убедиться в успешности установки, откройте интерактивную оболочку Python и введите:

```
>>> import nltk
>>>
```

Если ошибок не возникнет, то все в порядке. В противном случае следуйте инструкциям по установке на http://www.nltk.org/install.html.

Скачивание токенизатора

Для выполнения стилометрических тестов потребуется разбивать тексты — их *корпусы* — на отдельные слова, называемые *токенами*. На момент написания книги метод `word_tokenize()` в NLTK неявно вызывает `sent_tokenize()`, используемую для фрагментирования корпуса на отдельные предложения. Для обработки `sent_tokenize()` вам потребуется *Punkt Tokenizer Models*. Несмотря на то что это часть NLTK, нужно скачать и установить ее отдельно с помощью удобного NLTK Downloader. Для его запуска введите в оболочке Python:

```
>>> import nltk
>>> nltk.download()
```

Должно открыться окно NLTK Downloader (рис. 2.1). Перейдите во вкладку Models либо All Packages вверху; затем щелкните punkt в столбце Identifier. Прокрутите окно вниз и установите Download Directory для вашей платформы (см. http://www.nltk.org/data.html). В завершение щелкните на кнопке Download для скачивания Punkt Tokenizer Models.

Заметьте, что пакеты NLTK можно также скачать напрямую из оболочки. Вот пример:

```
>>> import nltk
>>> nltk.download('punkt')
```

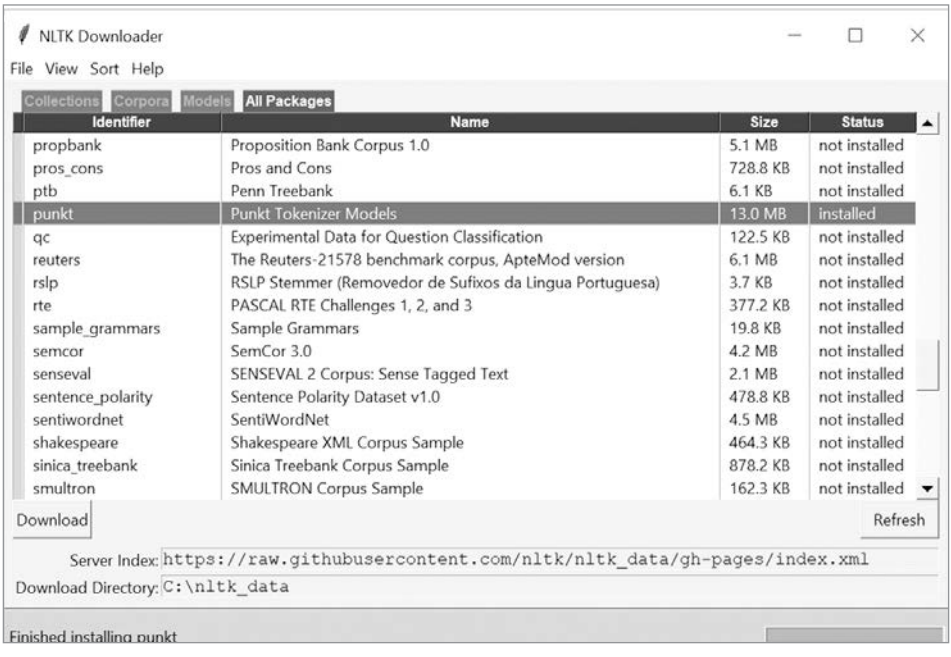

Рис. 2.1. Скачивание Punkt Tokenizer Models

Вам также потребуется обращаться к Stopwords Corpus, который можно скачать тем же способом.

Скачивание Stopwords Corpus

Перейдите во вкладку Corpora в окне NLTK Downloader и скачайте Stopwords Corpus, как показано на рис. 2.2.

В качестве альтернативы можно использовать оболочку.

```
>>> import nltk
>>> nltk.download('stopwords')
```

Давайте скачаем еще один пакет, который поможет анализировать части речи — существительные и глаголы. В окне NLTK Downloader перейдите во вкладку All Packages и скачайте Averaged Perceptron Tagger.

Чтобы сделать это с помощью оболочки, введите:

```
>>> import nltk
>>> nltk.download('averaged_perceptron_tagger')
```

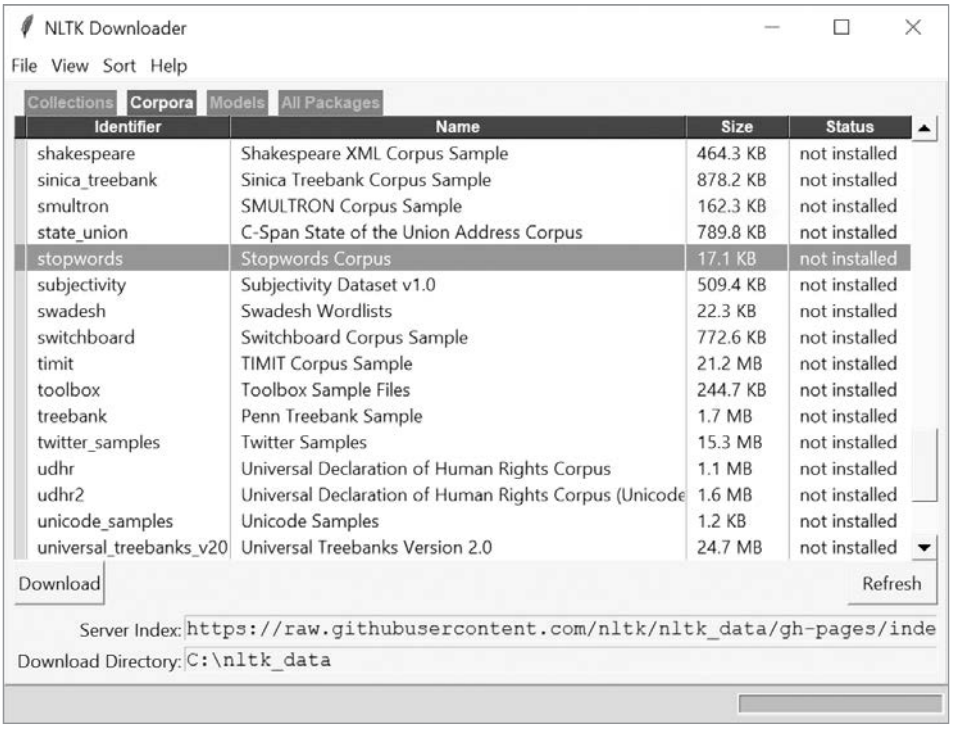

Рис. 2.2. Скачивание Stopwords Corpus

Когда NLTK завершит скачивание, выйдите из окна NLTK Downloader и введите в интерактивной оболочке Python:

```
>>> from nltk import punkt
```

А затем:

```
>>> from nltk.corpus import stopwords
```

Если ошибки не возникнет, значит, модели и корпус стоп-слов скачаны успешно.

В завершение для построения графиков потребуется `matplotlib`. Если вы ее еще не установили, то смотрите инструкции по установке научных пакетов на с. 32.

Корпусы текстов

Текстовые файлы для книг «Собака Баскервилей» (hound.txt), «Война миров» (war.txt) и «Затерянный мир» (lost.txt), а также код книги доступны на странице https://nostarch.com/real-world-python/.

Они были взяты с Project Gutenberg (https://www.gutenberg.org/), прекрасного источника общедоступной литературы. Чтобы вы могли с ходу начать использовать эти тексты, я убрал из них излишний материал, такой как содержание, названия глав, информация об авторских правах и пр.

Код стилометрии

Программа stylometry.py, которую мы напишем далее, загружает текстовые файлы в виде строк, токенизирует их на слова, после чего выполняет пять стилометрических анализов, перечисленных на с. 59. В качестве результата программа будет выводить набор графиков и сообщений оболочки, которые помогут определить автора книги «Затерянный мир».

Разместите программу в одном каталоге с тремя скачанными текстовыми файлами. Если вы не хотите вводить код самостоятельно, просто загрузите код, используя его доступную для скачивания версию с https://nostarch.com/real-world-python/.

Импорт модулей и определение функции main()

Код листинга 2.1 импортирует NLTK и `matplotlib`, назначает константу и определяет функцию `main()` для запуска программы. Используемые в `main()` функции будут подробно описаны в этой главе позднее.

Листинг 2.1. Импорт модулей и определение функции main()

stylometry.py, часть 1

```
import nltk
from nltk.corpus import stopwords
import matplotlib.pyplot as plt

LINES = ['-', ':', '--'] # Стиль линий для графиков.

def main():
  ❶ strings_by_author = dict()
    strings_by_author['doyle'] = text_to_string('hound.txt')
    strings_by_author['wells'] = text_to_string('war.txt')
    strings_by_author['unknown'] = text_to_string('lost.txt')

    print(strings_by_author['doyle'][:300])

  ❷ words_by_author = make_word_dict(strings_by_author)
    len_shortest_corpus = find_shortest_corpus(words_by_author)
  ❸ word_length_test(words_by_author, len_shortest_corpus)
    stopwords_test(words_by_author, len_shortest_corpus)
    parts_of_speech_test(words_by_author, len_shortest_corpus)
    vocab_test(words_by_author)
    jaccard_test(words_by_author, len_shortest_corpus)
```

Вначале выполняется импорт NLTK и Stopwords Corpus. Далее импортируется `matplotlib`.

После мы создаем переменную `LINES`. По соглашению ее имя прописывается заглавными буквами, это указывает, что она используется в качестве константы. Функция `matplotlib` по умолчанию рисует графики в цвете, но при этом все равно нужно задать список символов для людей, страдающих цветовой слепотой, а также для этой черно-белой книги.

Определяем `main()` и запускаем программу. Шаги в данной функции почти так же читаемы, как псевдокод, и наглядно представляют действия программы. На первом этапе будет выполнена инициализация словаря для хранения текста каждого автора ❶. Функция `text_to_string()` загружает каждый корпус в этот словарь в виде строки. Имя каждого автора будет являться ключом словаря (для «Затерянного мира» используется `unknown`), а строка текста романа — значением. Например, вот ключ `Doyle` с сильно обрезанным строковым значением:

```
{'Doyle': 'Mr. Sherlock Holmes, who was usually very late in the mornings
--snip--'}
```

Сразу после заполнения словаря выводим 300 элементов для ключа `doyle`, чтобы убедиться, что все прошло правильно. На выводе должно получиться следующее:

```
Mr. Sherlock Holmes, who was usually very late in the mornings, save
upon those not infrequent occasions when he was up all night, was seated
at the breakfast table. I stood upon the hearth-rug and picked up the
stick which our visitor had left behind him the night before. It was a
fine, thick piec
```

После корректной загрузки корпусов текстов переходим к токенизации строк в слова. На данный момент Python не распознает слова, он работает с *символами*, такими как буквы, числа и знаки препинания. Чтобы это исправить, мы используем функцию `make_word_dict()`, которая в качестве аргумента будет получать `strings_by_author`, разбивать эти строки на слова и возвращать словарь `words_by_author` с фамилиями авторов в качестве ключей и списком слов в качестве значений ❷.

Стилометрия опирается на подсчет слов, следовательно, лучше всего она работает, когда каждый корпус имеет одинаковую длину. Есть несколько способов обеспечить такое сравнение подобного с подобным. С помощью *разбивки* мы разделим текст на блоки, скажем, по 5000 слов, и сравним эти блоки. Нормализацию можно также производить, используя вместо прямого подсчета относительную частотность или усекая все корпуса по размеру наименьшего из них.

Рассмотрим вариант с усечением. Передадим словарь в другую функцию, `find_shortest_corpus()`, которая вычисляет количество слов в списке каждого

автора и возвращает длину самого короткого корпуса. В табл. 2.1 показана длина каждого корпуса.

Таблица 2.1. Длина (количество слов) каждого корпуса

Корпус	Длина
Hound (Doyle)	58,387
War (Wells)	59,469
World (Unknown)	74,961

Поскольку наименьший корпус здесь представляет робастный датасет почти из 60 000 слов, мы используем переменную `len_shortest_path`, чтобы обрезать другие два корпуса до этой длины, и уже затем перейдем к анализу. При этом мы, конечно же, предполагаем, что обрезаемое содержание текстов не сильно отличается от оставляемого.

На следующих пяти строках вызываются функции, выполняющие стилометрический анализ, представленный в разделе «Стратегия» на с. 58. ❸ Все эти функции получают в качестве аргумента словарь `words_by_author`, и большая их часть также получает `len_shortest_corpus`. Данные функции мы рассмотрим, как только закончим подготовку текстов к анализу.

Загрузка текста и построение словаря слов

В листинге 2.2 определяются две функции. Первая считывает текстовый файл в виде строки. Вторая создает словарь с именем каждого автора в качестве ключа и его романом, токенизированным из непрерывной строки в отдельные слова, в качестве значения.

Листинг 2.2. Определение функций text_to_string() и make_word_dict()

stylometry.py, часть 2

```
    def text_to_string(filename):
        """Читаем текстовый файл и возвращаем строку."""
        with open(filename) as infile:
            return infile.read()

❶ def make_word_dict(strings_by_author):
        """Возвращаем словарь слов-токенов корпусов по автору."""
        words_by_author = dict()
        for author in strings_by_author:
            tokens = nltk.word_tokenize(strings_by_author[author])
          ❷ words_by_author[author] = ([token.lower() for token in tokens
                                        if token.isalpha()])
        return words_by_author
```