第三讲 - 文本统计量和向量化表示

张建章

阿里巴巴商学院
杭州师范大学

2023-02-22

目录

语料库构建：将原始文本预处理后，按照结构化的方式组织和保存。

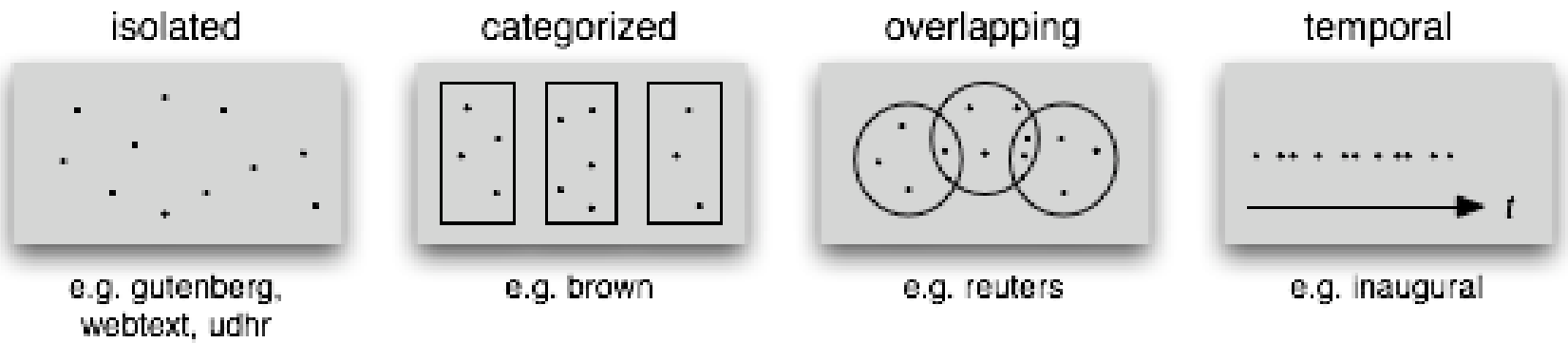


图 1: 常见语料库组织结构

语料库管理：从多种视图高效检索语料库，检索结果可进一步用于文本统计分析和向量化。

本讲以中文新闻作为示例语料讲解语料库的构建和管理，并在此基础上学习常见的文本统计量和文本向量化表示方法。该语料来自搜狗实验室发布的新闻分类数据集。

新闻语料预处理

主要包括：分句、分词、词性标注。使用 `spaCy` (高效、开源、维护更新及时、支持多种语言和多种 NLP 任务，支持加载调用已训练好的模型和训练自己的模型) 进行预处理，示例用法如下：

```
import spacy
nlp = spacy.load("zh_core_web_sm")

text = '''2023年短道速滑世锦赛落幕，
↪  在最后一个项目男子5000米接力决赛中，中国队夺得金牌。
↪  这是本届世锦赛中国队夺得的首金。'''

doc = nlp(text)
for sent in doc.sents:
    for token in sent:
        print(token.text, token.pos_)
    print('-'*20)
```

定义一个文本预处理函数，应用到每条新闻文本上：

```
def tagging(text):
    f_list = []
    paragraphs = html.unescape(text).strip().split('\n')
    for paragraph in paragraphs:
        if paragraph.strip():
            p_list = []
            doc = nlp(paragraph.strip())
            for sent in doc.sents:
                t_list = []
                for token in sent:
                    t_list.append(token.text+'/'+token.pos_)
                p_list.append(' '.join(t_list))
            f_list.append('\n'.join(p_list))
    tagged_text = '\n\n'.join(f_list)
    return tagged_text
```

对所有新闻文本进行预处理，并保存预处理结果：

```
import os
raw_data_dir = './cn_news/'
tagged_data_dir = './cn_news_tagged/'
categories = os.listdir(raw_data_dir)
for category in categories:
    files = os.listdir(raw_data_dir + category)
    for file in files:
        file_path = raw_data_dir + category + '/' + file
        if not os.path.exists(tagged_data_dir + category):
            os.mkdir(tagged_data_dir + category)
        new_file_path = tagged_data_dir + category + '/' + file
        with open(file_path, encoding='gbk',errors='ignore') as
        ↪  f:
            raw_text = f.read()
        tagged_text = tagging(raw_text)
        with open(new_file_path,'w') as f:
            f.write(tagged_text)
```

构建新闻分类标注语料库

使用 NLTK 提供的语料库 API，从前述的新闻标注结果文件构建分类标注语料库。

```
from nltk.corpus.reader import CategorizedTaggedCorpusReader

creader = CategorizedTaggedCorpusReader('./cn_news_tagged/',
↪  '.*',cat_pattern = r'(.+)/.+txt')
```

与文件读写和字符串操作的简单组合相比，NLTK 提供的语料库构建 API 效率更高，支持多种语料库检索和查询操作。

语料库管理

NLTK 提供的常用的语料库管理功能如下：

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

图 2: NLTK 中常用的语料库管理方法

语料库检索

上下文是确定词语语义 (语义消歧) 的关键信息，也可用于确定两个词语的相似度 (思考英语阅读理解中你对陌生/罕见词含义的推断过程)。下面代码使用 NLTK 中的语料库方法实现上述两个功能。

```
# 通过上下文理解单词monstrous的意思
# text1为小说白鲸记文本
from nltk.book import *
text1.concordance("monstrous")

# of Whale - Bones ; for Whales of a monstrous size are
↪  oftentimes cast up dead u
# ll over with a heathenish array of monstrous clubs and spears
↪  . Some were thick
# ere to enter upon those still more monstrous stories of them
↪  which are to be fo

# 鲸鱼骨头; 因为<巨大的>鲸鱼经常被抛出死去
# 到处都是异教徒的<怪异的>棍棒和长矛。有些很厚
# 在进入他们将要讲述的那些更<可怕的>故事之前
```

语料库检索

```
# text1为小说白鲸记文本
# text2为小说理智与情感文本
from nltk.book import *
text1.similar("monstrous")
# mean part maddens doleful gamesome subtly uncommon careful
↪  untoward exasperate loving passing mouldy christian few true
↪  mystifying imperial modifies contemptible
text2.similar("monstrous")
# very heartily so exceedingly remarkably as vast a great
↪  amazingly extremely good sweet

# 寻找多个词语的公共上下文, a very/monstrous lucky
text2.common_contexts(["monstrous", "very"])
a_pretty is_pretty am_glad be_glad a_lucky
```

在小说 < 大白鲸 > 和 < 理智与爱情 > 中单词 *monstrous* 的含义明显不同，在 < 理智与爱情 > 中，其含义偏积极，有时还会像 *very* 一样作为语气增强词使用。

文本长度和词表规模

通过计算文本中词汇总数 (N) 和词表规模 (V)，可以计算不同类型文本的词汇丰富度。

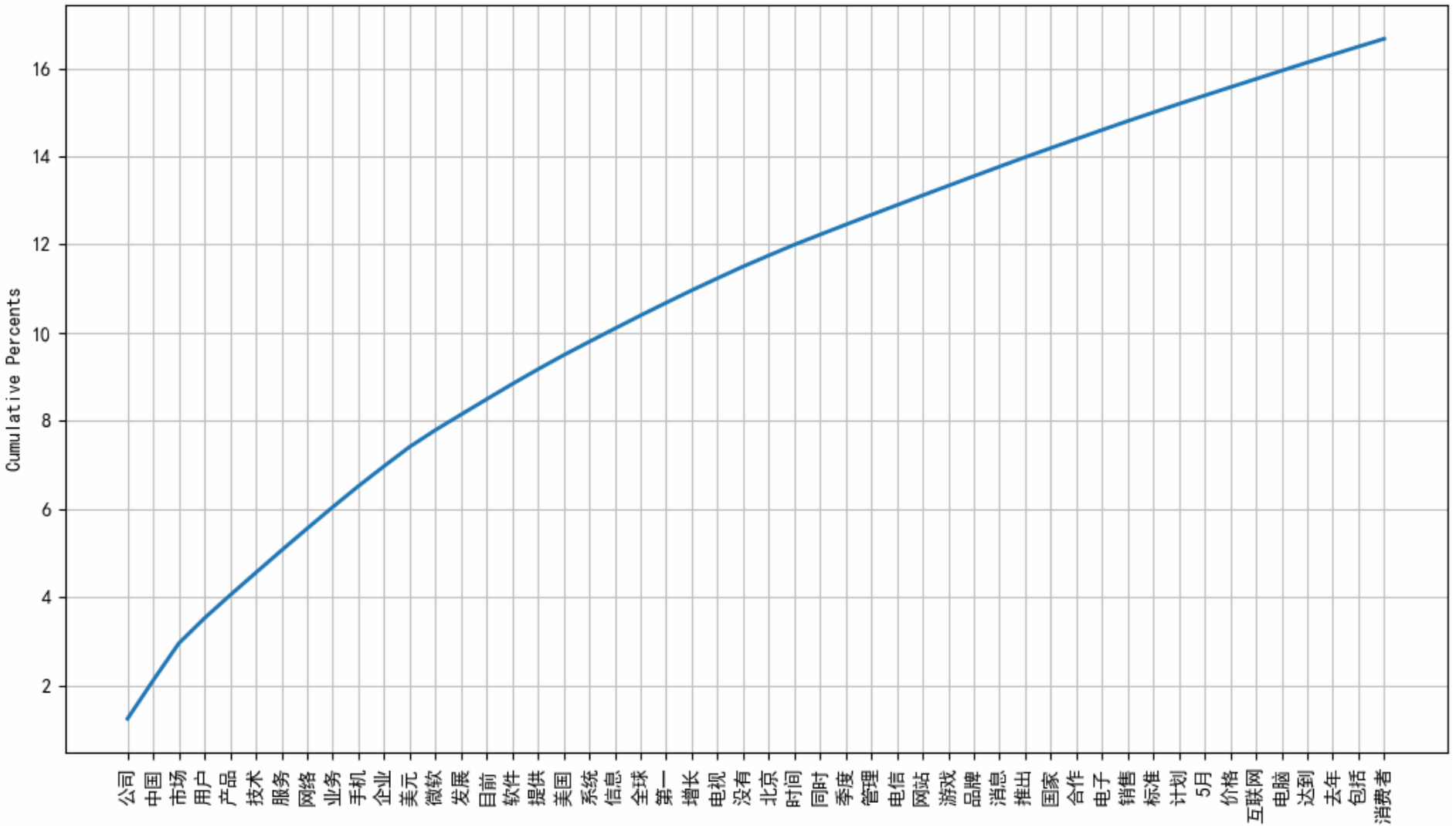
$$lexical\_richness = \frac{V}{N}$$

下面以中文新闻语料库为例，计算不同类别的词汇丰富度。

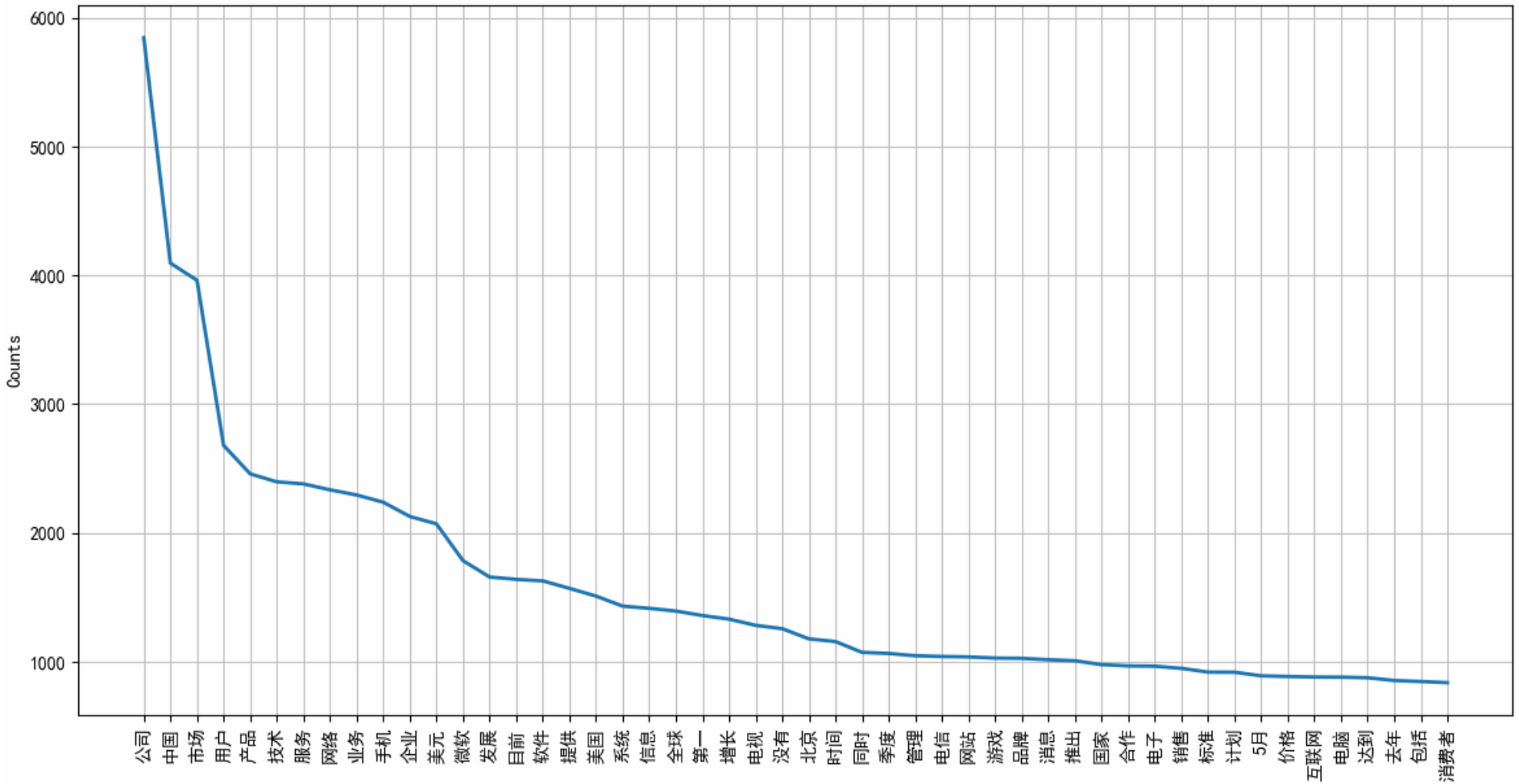
```
from nltk.corpus.reader import CategorizedTaggedCorpusReader

creader = CategorizedTaggedCorpusReader('./cn_news_tagged/',
↪  '.*',cat_pattern = r'(.+)/.+txt')
it_words = creader.tagged_words(categories=['IT'])
cult_words = creader.tagged_words(categories=['Culture'])
print(len(it_words), len(cult_words))
print(len(set(it_words)), len(set(cult_words)))
def lexical_diversity(text):
  return len(set(text)) / len(text)
print(lexical_diversity(it_words), lexical_diversity(cult_words))
```

词语频率分布

计算词表中词语的出现频率 (Top 50)，绘制词语累积频率分布图。



计算词表中词语的出现频数 (Top 50)，绘制词语频数分布图 (长尾分布)，只有少量高频词，大多数词语出现的频率都很低 (词频曲线陡降并趋于平缓)。



词语频率分布

识别文本中的高频词语，并绘制词云 (Top 50)。

条件频率统计

统计词汇在给定条件下的频率，如统计某一词语在不同类别的新闻中出现的频率，某一词语在不同年份语料中出现的频率等。可以使用 `NLTK` 中的条件频率统计方法 `ConditionalFreqDist` 进行计算和绘图，将条件和词语表示为二元组形式，例如，(文本类别, 词语)。得分，货币，心情三个词语在不同类别新闻语料中的频率如下表。

	得分	货币	心情
Culture	9	62	143
Education	99	79	148
Finance	0	256	7
Health	6	0	192
IT	31	9	12
Military	1	5	28
Recruit	31	26	249
Sports	704	0	71
Travel	0	59	62

从具有时间跨度的语料库中按照时间维度统计词频，可以挖掘词语含义、使用模式随时间变化的趋势。下图显示了在美国总统就职演说语料库中，以 *america*-和 *citizen*-为前缀的单词的使用频率随时间变化的趋势。code

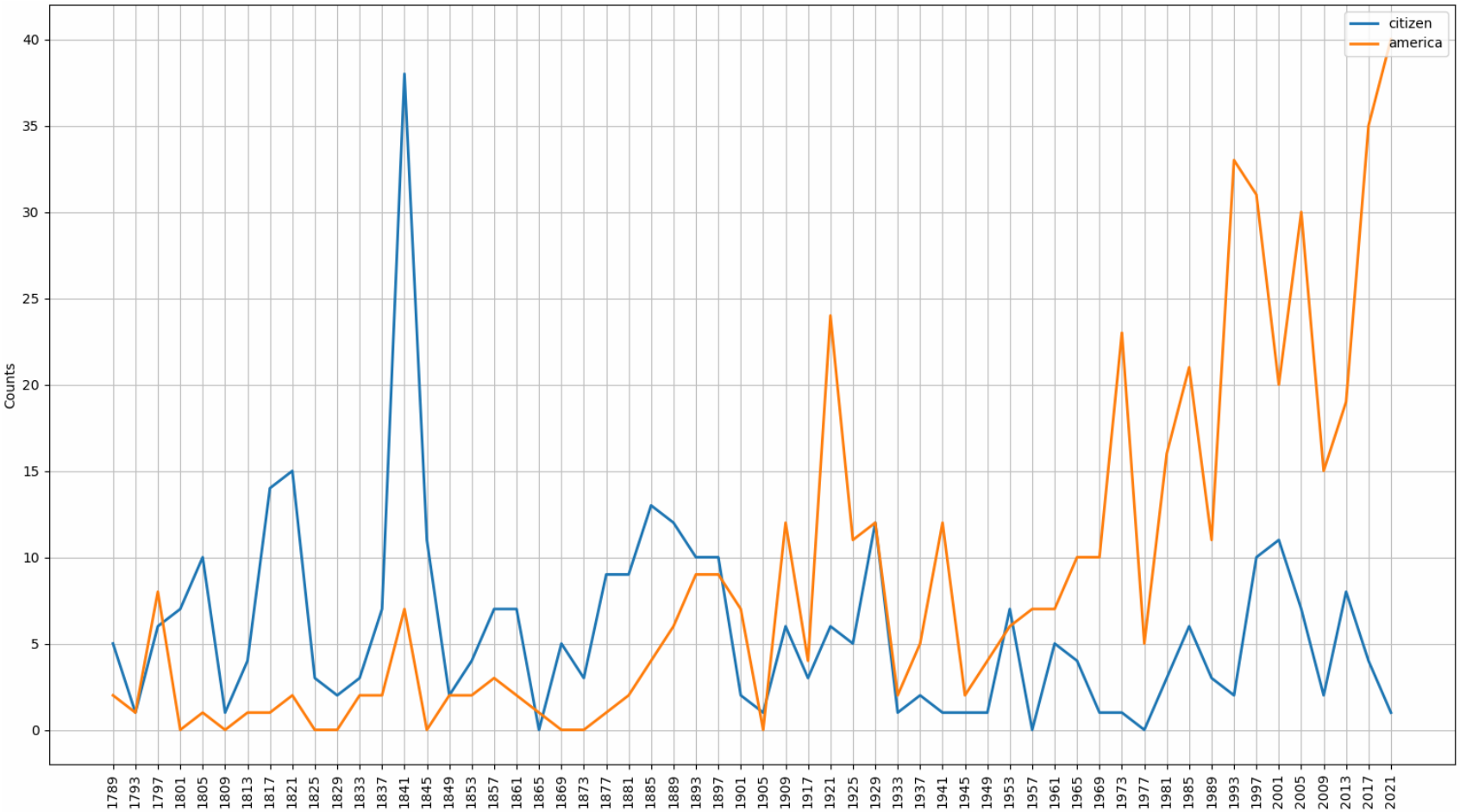


图 3: NLTK 中常用的语料库管理方法

1. 使用 NLTK 语料库 API 和提供的历年政府工作报告文本，构建政府工作报告语料库。

2. 结合本讲文本统计量的内容，探索历年政府工作报告中用词特点，结果以图或表方式呈现，并据此给出你的分析。

未完待续