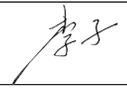
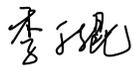
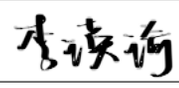
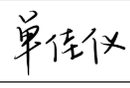
数据挖掘与最优化: Assignment 1

组长:	231502004 李子	完成内容	T2, T3	贡献度	20%	
组员:	221300066 季千焜	完成内容	T1	贡献度	20%	
组员:	221098024 李瑛琦	完成内容	T4	贡献度	20%	
组员:	221098145 李傲雪	完成内容	优化 T2, T3, T4	贡献度	20%	
组员:	221098071 单佳仪	完成内容	T4	贡献度	20%	

目录

实验进度

我们完成了所有内容。

T1

修改 gradient_descent 函数，在每次迭代中遍历候选步长并选择使目标函数最小的步长。以下是代码：

```python
def gradient_descent(max_iterations, threshold, w_init, obj_func, grad_func):
    w = w_init.copy()
    w_history = w.reshape(1, -1)  # 初始化为二维数组
    f_history = np.array([obj_func(w)])
    i = 0
    diff = 1.0e10

    while i < max_iterations and diff > threshold:
        grad = grad_func(w)
        lrs = np.linspace(0, 1, 101)
        best_lr = 0
        best_f = np.inf
        best_w = w
        # 遍历所有候选步长，寻找最优步长
        for lr in lrs:
            w_candidate = w - lr * grad
            f_candidate = obj_func(w_candidate)
            if f_candidate < best_f:
                best_f = f_candidate
                best_lr = lr
                best_w = w_candidate

        # 更新参数和记录历史
        w = best_w
        w_history = np.vstack((w_history, w.reshape(1, -1)))
        f_history = np.vstack((f_history, [best_f]))
        # 计算函数值差异
        if i == 0:
            diff = np.abs(f_history[1] - f_history[0])
        else:
            diff = np.abs(f_history[-1] - f_history[-2])
        i += 1
    return w_history, f_history
```

修改后代码整体运行时间从 0.2 秒左右缩短到 0.083 秒左右，效率显著提高。

T2

我们通过如下代码，首先构建了词典列表：

```python
def remove_numbers_from_file(input_file, output_file):
    with open(input_file, 'r', encoding='utf-8') as file:
        lines = file.readlines()

    with open(output_file, 'w', encoding='utf-8') as file:
        for line in lines:
            new_line = ''.join([char for char in line if not char.isdigit()])
            file.write(new_line)

if __name__ == "__main__":
    input_file = 'THUOCL_diming.txt'
    output_file = 'THUOCL_diming_no_numbers.txt'
    remove_numbers_from_file(input_file, output_file)
```

然后通过如下代码，将两个文件合并：

```python
def read_file(file_path):
    with open(file_path, 'r', encoding='utf-8') as file:
        return file.readlines()

def main():
    file1 = 'THUOCL_diming_no_numbers.txt'
    file2 = 'THUOCL_caijing_no_numbers.txt'

    content1 = read_file(file1)
    content2 = read_file(file2)

    combined_content = content1 + content2

    with open('combined_output.txt', 'w', encoding='utf-8') as output_file:
        output_file.writelines(combined_content)

if __name__ == "__main__":
    main()
```

接着修改 cut 代码，使其在仅剩单字的情况下不再切分：

```python
  import numpy as np
  def cut(dic,text):
    result=[]
    index=0
    text_length=len(text)
```

```
    length=[]
    for word in dic:
      length.append(len(word))
    m=max(length)
    while text_length>index:
      for size in np.arange(m+index,index,-1):
        piece=text[index:size]
        if piece in dic or len(piece) == 1:
          result.append(piece)
          index=size
          break
    print(result)
```

切分效果如下：

[' 据', ' 中', ' 指', ' 研究院', ' 统计', ',', ',', ' 自', '9', ' 月', '3', '0', ' 日', ' 人', ' 民', ' 银行', ' 发', ' 布', ' 消', ' 息', ' 决', ' 定', ' 下调', ' 首', ' 套', ' 个', ' 人', ' 住', ' 房', ' 公', ' 积', ' 金', ' 贷款', ' 利率', ' 以', ' 来', ',', ',', ' 截', ' 至', '1', '0', ' 月', '9', ' 日', ',', ',', ' 已', ' 有', ' 杭', ' 州', '、', ' 济', ' 南', '、', ' 吉', ' 林', ' 等', ' 至', ' 少', '3', '0', ' 个', ' 地', ' 区', ' 的', ' 公', ' 积', ' 金', ' 管', ' 理', ' 中', ' 心', ' 发', ' 布', ' 相', ' 关', ' 通', ' 知', ',', ',', ' 落', ' 实', ' 下调', ' 首', ' 套', ' 个', ' 人', ' 住', ' 房', ' 公', ' 积', ' 金', ' 贷款', ' 利率', '0', '.', '1', '5', ' 个', ' 百分点', '。', ' 近期', ',', ',', ' 包', ' 括', ' 下调', ' 首', ' 套', ' 个', ' 人', ' 住', ' 房', ' 公', ' 积', ' 金', ' 贷款', ' 利率', '、', ' 阶', ' 段', ' 性', ' 放', ' 宽', ' 部', ' 分', ' 城', ' 市', ' 首', ' 套', ' 住', ' 房贷', ' 款', ' 利率', ' 下', ' 限', '、', ' 支持', ' 居', ' 民', ' 换', ' 购', ' 住', ' 房', ' 个', ' 人', ' 所', ' 得', ' 税', ' 退', ' 税', ' 优惠', ' 政', ' 策', ' 等', ' 接', ' 连', ' 出台', '。']

显然可见效果并不是很好，因为我们的词典中并没有包含很多常用词汇。并且词典也有一定不全的问题。比如：词典中有“杭州市”，“杭州 xx”，但是没有“杭州”这个词。这就导致无法切分出“杭州”这个词。

T3

修改代码如下：

```python
import numpy as np
def cut(dic,text):
    result=[]
    index=0
    text_length=len(text)
    length=[]
    for word in dic:
        length.append(len(word))
    m=max(length)
    while text_length>index:
        for size in np.arange(m+index,index,-1):
            piece=text[index:size]
            if piece in dic or len(piece) == 1:
                result.append(piece)
                index=size
                break
    return result
def cut2(dic, text):
    result = []
    index = len(text)
    length = []
    for word in dic:
        length.append(len(word))
    m = max(length)
    while index > 0:
        for size in np.arange(max(0, index - m), index):
            piece = text[size:index]
            if piece in dic or len(piece) == 1:
                result.append(piece)
                index = size
                break
    result.reverse()
    return result
def cut3(dic, text):
    result1 = cut(dic, text)
    result2 = cut2(dic, text)
    if len(result1) == len(result2):
        print("SAME RESULT!")
    if len(result1) <= len(result2):
        return result1
```

```
    else:
        return result2

print(cut(dic,text))
print(cut2(dic,text))
print(cut3(dic,text))
```

得到输出：

[' 据', ' 中', ' 指', ' 研究院', ' 统计', ',', ',', ' 自', '9', ' 月', '3', '0', ' 日', ' 人', ' 民', ' 银行', ' 发', ' 布', ' 消', ' 息', ' 决', ' 定', ' 下调', ' 首', ' 套', ' 个', ' 人', ' 住', ' 房', ' 公', ' 积', ' 金', ' 贷款', ' 利率', ' 以', ' 来', ',', ',', ' 截', ' 至', '1', '0', ' 月', '9', ' 日', ',', ',', ' 已', ' 有', ' 杭', ' 州', '、', ',', ' 济', ' 南', '、', ',', ' 吉', ' 林', ' 等', ' 至', ' 少', '3', '0', ' 个', ' 地', ' 区', ' 的', ' 公', ' 积', ' 金', ' 管', ' 理', ' 中', ' 心', ' 发', ' 布', ' 相', ' 关', ' 通', ' 知', ',', ',', ' 落', ' 实', ' 下调', ' 首', ' 套', ' 个', ' 人', ' 住', ' 房', ' 公', ' 积', ' 金', ' 贷款', ' 利率', '0', ':', '1', '5', ' 个', ' 百分点', '。', ' 近期', ',', ',', ' 包', ' 括', ' 下调', ' 首', ' 套', ' 个', ' 人', ' 住', ' 房', ' 公', ' 积', ' 金', ' 贷款', ' 利率', '、', ' 阶', ' 段', ' 性', ' 放', ' 宽', ' 部', ' 分', ' 城', ' 市', ' 首', ' 套', ' 住', ' 房贷', ' 款', ' 利率', ' 下', ' 限', '、', ' 支持', ' 居', ' 民', ' 换', ' 购', ' 住', ' 房', ' 个', ' 人', ' 所', ' 得', ' 税', ' 退', ' 税', ' 优惠', ' 政', ' 策', ' 等', ' 接', ' 连', ' 出台', '。']

[' 据', ' 中', ' 指', ' 研究院', ' 统计', ',', ',', ' 自', '9', ' 月', '3', '0', ' 日', ' 人', ' 民', ' 银行', ' 发', ' 布', ' 消', ' 息', ' 决', ' 定', ' 下调', ' 首', ' 套', ' 个', ' 人', ' 住', ' 房', ' 公', ' 积', ' 金', ' 贷款', ' 利率', ' 以', ' 来', ',', ',', ' 截', ' 至', '1', '0', ' 月', '9', ' 日', ',', ',', ' 已', ' 有', ' 杭', ' 州', '、', ',', ' 济', ' 南', '、', ',', ' 吉', ' 林', ' 等', ' 至', ' 少', '3', '0', ' 个', ' 地', ' 区', ' 的', ' 公', ' 积', ' 金', ' 管', ' 理', ' 中', ' 心', ' 发', ' 布', ' 相', ' 关', ' 通', ' 知', ',', ',', ' 落', ' 实', ' 下调', ' 首', ' 套', ' 个', ' 人', ' 住', ' 房', ' 公', ' 积', ' 金', ' 贷款', ' 利率', '0', ':', '1', '5', ' 个', ' 百分点', '。', ' 近期', ',', ',', ' 包', ' 括', ' 下调', ' 首', ' 套', ' 个', ' 人', ' 住', ' 房', ' 公', ' 积', ' 金', ' 贷款', ' 利率', '、', ' 阶', ' 段', ' 性', ' 放', ' 宽', ' 部', ' 分', ' 城', ' 市', ' 首', ' 套', ' 住', ' 房', ' 贷款', ' 利率', ' 下', ' 限', '、', ' 支持', ' 居', ' 民', ' 换', ' 购', ' 住', ' 房', ' 个', ' 人', ' 所', ' 得', ' 税', ' 退', ' 税', ' 优惠', ' 政', ' 策', ' 等', ' 接', ' 连', ' 出台', '。']

SAME RESULT!

[' 据', ' 中', ' 指', ' 研究院', ' 统计', ',', ',', ' 自', '9', ' 月', '3', '0', ' 日', ' 人', ' 民', ' 银行', ' 发', ' 布', ' 消', ' 息', ' 决', ' 定', ' 下调', ' 首', ' 套', ' 个', ' 人', ' 住', ' 房', ' 公', ' 积', ' 金', ' 贷款', ' 利率', ' 以', ' 来', ',', ',', ' 截', ' 至', '1', '0', ' 月', '9', ' 日', ',', ',', ' 已', ' 有', ' 杭', ' 州', '、', ',', ' 济', ' 南', '、', ',', ' 吉', ' 林', ' 等', ' 至', ' 少', '3', '0', ' 个', ' 地', ' 区', ' 的', ' 公', ' 积', ' 金', ' 管', ' 理', ' 中', ' 心', ' 发', ' 布', ' 相', ' 关', ' 通', ' 知', ',', ',', ' 落', ' 实', ' 下调', ' 首', ' 套', ' 个', ' 人', ' 住', ' 房', ' 公', ' 积', ' 金', ' 贷款', ' 利率', '0', ':', '1', '5', ' 个', ' 百分点', '。', ' 近期', ',', ',', ' 包', ' 括', ' 下调', ' 首', ' 套', ' 个', ' 人', ' 住', ' 房', ' 公', ' 积', ' 金', ' 贷款', ' 利率', '、', ' 阶', ' 段', ' 性', ' 放', ' 宽', ' 部', ' 分', ' 城', ' 市', ' 首', ' 套', ' 住', ' 房贷', ' 款', ' 利率', ' 下', ' 限', '、', ' 支持', ' 居', ' 民', ' 换', ' 购', ' 住', ' 房', ' 个', ' 人', ' 所', ' 得', ' 税', ' 退', ' 税', ' 优惠', ' 政', ' 策', ' 等', ' 接', ' 连', ' 出台', '。']

说明正向和逆向切分结果一致。自然三种不同分词方法的结果也一致。

T4

(i) 将数据导入 Python，分别统计正面评论和负面评论的条数。以下是代码：

```
import re
import jieba as jb
import matplotlib.pyplot as plt
import numpy as np
with open('中文酒店评论.txt', 'r', encoding='utf-8') as file:
    content = file.read()
p_content = content.split("\n")
gd_num = 0
bd_num = 0
regex1 = "^1"
for line in p_content:
    if re.search(regex1,line) is not None: # 如果当前行匹配regex的结果不是None
        gd_num += 1# 就打印这行信息
    else:
        bd_num += 1
print("正面评论数量：",gd_num)
print("负面评论数量：",bd_num)
```

得到输出：

正面评论数量：7000 负面评论数量：3001

(ii) 分别对每条评论进行分词并统计词数（或句子长度）。

```
jb_cmt = list()#分词结果
wd_num = list()#词数
len_cmt = list()#句子长度
for line in p_content[1:]:
    cmt = line.split("    ")[1]
    jb_cut = jb.lcut(cmt, cut_all = False)
    jb_cmt.append(jb_cut)
    wd_num.append(len(jb_cut))
    len_cmt.append(len(cmt))
```

(iii) 分别绘制正面评论和负面评论句子长度的直方图。

```
# 绘制直方图
gdata = len_cmt[0:gd_num]
bdata = len_cmt[gd_num:]
plt.hist(gdata, color='blue', alpha=0.7)  # bins 表示柱子的数量
plt.title('Positive comment histogram')  # 设置标题
plt.xlabel('Wordage')  # 设置 x 轴标签
plt.ylabel('Frequency')  # 设置 y 轴标签
plt.show()  # 显示图表
```

```
plt.hist(bdata, color='red', alpha=0.7)
plt.title('Negative comment histogram')  # 设置标题
plt.xlabel('Wordage')  # 设置 x 轴标签
plt.ylabel('Frequency')  # 设置 y 轴标签
plt.show()  # 显示图表
```

得到直方图：

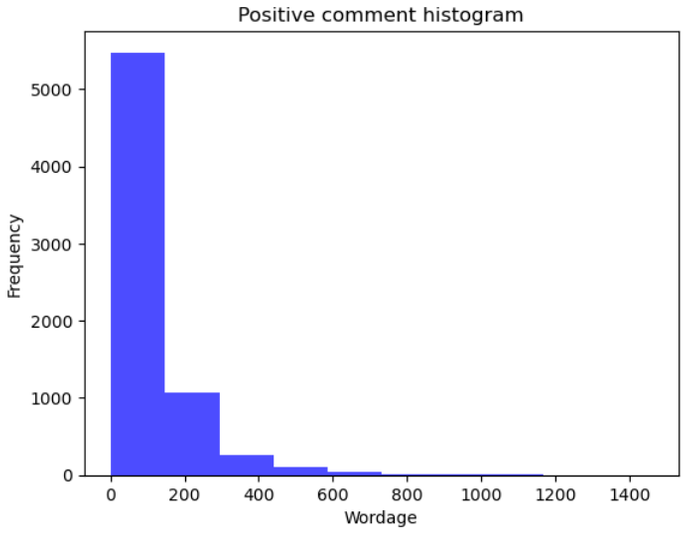


图 1: Positive Comment

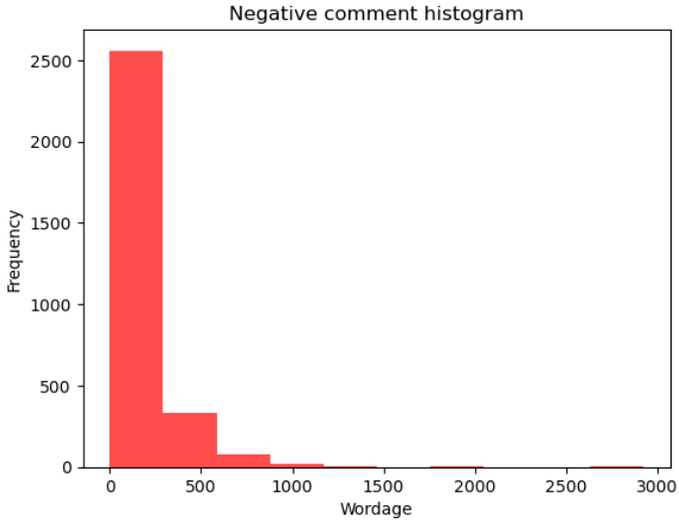


图 2: Negative Comment

(iv) 删除 (ii) 中每条评论分词后得到的非形容词，只保留每条评论分词后得到的形容词。

```
import jieba.posseg as pseg
def extract_adjectives(text):
    words = pseg.lcut(text)
    adjectives = [word.word for word in words if word.flag == 'a']
```

```python
    return adjectives
jb_cmtg = list()#正面分形容词结果
jb_cmtb = list()#负面分形容词结果
for line in p_content[1:]:
    cmt = line.split("	")[1]
    jb_cut = extract_adjectives(cmt)
    if line.split("	")[0]=='1':
        jb_cmtg.append(jb_cut)
    else:
        jb_cmtb.append(jb_cut)
```

(v) 分别绘制正面评论和负面评论高频形容词的词云图。

```python
from wordcloud import WordCloud
#创建证明评价词云对象
gcloud=list()
for gw in jb_cmtg:
    gwords_str = " ".join(gw)
    gcloud.append(gwords_str)

gwordcloud = WordCloud(background_color="white", width=800, height=600, font_path='font.ttf').
    generate(" ".join(gcloud))

plt.imshow(gwordcloud, interpolation='bilinear')
plt.axis("off")
plt.show()
```

得到正面评价词云图：

图 3: 正面评价词云图

```python
#创建负面评价词云对象
```

```
bcloud=list()
for bw in jb_cmtb:
    bwords_str = " ".join(bw)
    bcloud.append(bwords_str)
bwordcloud = WordCloud(background_color="white", width=800, height=600, font_path='font.ttf').
    generate(" ".join(bcloud))
plt.imshow(bwordcloud, interpolation='bilinear')
plt.axis("off")
plt.show()
```

得到负面评价词云图：

图 4: 负面评价词云图

T2 与 T3 优化

正向最大匹配法优化点：

- 加入结巴词库（dict.txt.big）扩大现有词典内容
- 利用正则表达式，完善日期、数字的分词

相关库

```
import jieba
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import jieba.posseg as pseg
import re
import itertools
from wordcloud import WordCloud
```

字典与文档

```
# 字典
dic_financial=pd.read_csv(".\THUOCL_caijing.txt",sep="\t")
dic_financial.columns = ['word', 'count']
dic_financial=dic_financial['word']

dic_place=pd.read_csv(".\THUOCL_diming.txt",sep="\t")
dic_place.columns = ['word', 'count']
dic_place=dic_place['word']

def load_jieba_dict(path):
    word_list = []  # 改用列表存储
    with open(path, 'r', encoding='utf-8') as f:
        for line in f:
            word = line.split()[0]  # 取每行第一个词
            word_list.append(word)
    return word_list  # 返回列表

my_dict = load_jieba_dict("dict.txt.big")  #
dict=my_dict+list(dic_financial)+list(dic_place)+['吉林']

# 文档
text = "据中指研究院统计，自9月30日人民银行发布消息决定下调首套个人住房公积金贷款利率以来，截至10月9
    日，已有杭州、济南、吉林等至少30个地区的公积金管理中心发布相关通知，落实下调首套个人住房公积金贷款
```

```
利率0.15个百分点。近期，包括下调首套个人住房公积金贷款利率、阶段性放宽部分城市首套住房贷款利率下限、支持居民换购住房个人所得税退税优惠政策等接连出台。"
```

新的正向最大匹配函数

```python
def cut_forward(dic,text):
    result=[]
    index=0
    text_length=len(text)
    max_length = max(len(word) for word in dic) if dic else 1

    # 正则匹配数字、日期、小数等模式
    patterns = [
        (r"\d+月\d+日", "DATE"),     # 匹配 "9月30日"
        (r"\d+\.\d+", "FLOAT"),      # 匹配 "0.15"
        (r"\d+", "INT")              # 匹配纯数字
    ]

    while text_length>index:
        # 优先用正则匹配数字/日期
        matched = False
        for pattern, _ in patterns:
            regex = re.compile(pattern)
            match = regex.match(text, index)
            if match:
                result.append(match.group())
                index = match.end()
                matched = True
                break
        if matched:
            continue

        current_max = min(max_length, text_length - index)
        for size in np.arange(current_max,0,-1):
            piece=text[index:index+size]
            if piece in dic or len(piece)==1:
                result.append(piece)
                index+=size
                break
    return(result)

print(cut_forward(dict,text))
```

得到输出：

[' 据', ' 中指', ' 研究院', ' 统计', ', ', ' 自', '9 月 30 日', ' 人民银行', ' 发布', ' 消息', ' 决定', ' 下调', ' 首套', ' 个人住房', ' 公积金', ' 贷款', ' 利率', ' 以来', ', ', ' 截至', '10 月 9 日', ', ', ' 已有', ' 杭州', '、', ' 济南', '、', ' 吉林', ' 等', ' 至少', '30', ' 个', ' 地区', ' 的', ' 公积金', ' 管理中心', ' 发布', ' 相关', ' 通知', ', ', ' 落实', ' 下调', ' 首套', ' 个人住房', ' 公积金', ' 贷款', ' 利率', '0.15', ' 个', ' 百分点', '。', ' 近期', ', ', ' 包括', ' 下调', ' 首套', ' 个人住房', ' 公积金', ' 贷款', ' 利率', '、', ' 阶段性', ' 放宽', ' 部分', ' 城市', ' 首套', ' 住房贷款', ' 利率', ' 下限', '、', ' 支持', ' 居民', ' 换购', ' 住房', ' 个人所得税', ' 退税', ' 优惠政策', ' 等', ' 接连', ' 出台', '。']

新的逆向最大匹配函数

```python
# 逆向匹配
def cut_reverse(dic, text):
    result = []
    index = len(text)  # 从文本末尾开始
    max_length = max(len(word) for word in dic) if dic else 1

    # 正则模式调整（增加右边界匹配）
    patterns = [
        (r"\d+月\d+日$", "DATE"),  # 增加$匹配右边界
        (r"\d+\.\d+$", "FLOAT"),
        (r"\d+$", "INT")
    ]

    while index > 0:
        current_start = max(0, index - max_length)
        current_window = text[current_start:index]

        # 逆向正则匹配（优先处理特殊模式）
        matched = False
        for pattern, _ in patterns:
            regex = re.compile(pattern)
            match = regex.search(current_window)  # 在窗口内搜索
            if match:
                matched_text = match.group()
                result.insert(0, matched_text)  # 插入结果列表头部
                index -= len(matched_text)
                matched = True
                break
        if matched:
            continue

        # 逆向词典匹配
        found = False
        for size in range(min(index, max_length), 0, -1):
```

```
            piece = text[index-size:index]
            if piece in dic or size == 1:
                result.insert(0, piece)  # 插入结果列表头部
                index -= size
                found = True
                break
        if not found:
            # 处理未匹配字符
            result.insert(0, text[index-1])
            index -= 1

    return result

print(cut_reverse(dict,text))
```

得到输出:

[' 据', ' 中指', ' 研究院', ' 统计', ', ', ' 自', '9 月 30 日', ' 人民银行', ' 发布', ' 消息', ' 决定', ' 下调', ' 首套', ' 个人住房', ' 公积金', ' 贷款', ' 利率', ' 以来', ', ', ' 截至', '10 月 9 日', ', ', ' 已有', ' 杭州', '、', ' 济南', '、', ' 吉林', ' 等', ' 至少', '30', ' 个', ' 地区', ' 的', ' 公积金', ' 管理中心', ' 发布', ' 相关', ' 通知', ', ', ' 落实', ' 下调', ' 首套', ' 个人住房', ' 公积金', ' 贷款', ' 利率', '0.15', ' 个', ' 百分点', '。', ' 近期', ', ', ' 包括', ' 下调', ' 首套', ' 个人住房', ' 公积金', ' 贷款', ' 利率', '、', ' 阶段性', ' 放宽', ' 部分', ' 城市', ' 首套', ' 住房贷款', ' 利率', ' 下限', '、', ' 支持', ' 居民', ' 换购', ' 住房', ' 个人所得税', ' 退税', ' 优惠政策', ' 等', ' 接连', ' 出台', '。']

新的双向匹配函数

```
# 双向匹配
def cut_2direction(dic, text):
    result1 = cut_forward(dic, text)
    result2 = cut_reverse(dic, text)
    if result1 == result2:
        print("SAME RESULT!")
    if len(result1) <= len(result2):
        return result1
    else:
        return result2

print(cut_2direction(dict,text))
```

得到输出: SAME RESULT!

[' 据', ' 中指', ' 研究院', ' 统计', ', ', ' 自', '9 月 30 日', ' 人民银行', ' 发布', ' 消息', ' 决定', ' 下调', ' 首套', ' 个人住房', ' 公积金', ' 贷款', ' 利率', ' 以来', ', ', ' 截至', '10 月 9 日', ', ', ' 已有', ' 杭州', '、', ' 济南', '、', ' 吉林', ' 等', ' 至少', '30', ' 个', ' 地区', ' 的', ' 公积金', ' 管理中心', ' 发布', ' 相关', ' 通知', ', ', ' 落实', ' 下调', ' 首套', ' 个人住房', ' 公积金', ' 贷款', ' 利率', '0.15', ' 个', ' 百分点', '。', ' 近期', ', ', '

包括', ' 下调', ' 首套', ' 个人住房', ' 公积金', ' 贷款', ' 利率', '、', ' 阶段性', ' 放宽', ' 部分', ' 城市', ' 首套', ' 住房贷款', ' 利率', ' 下限', '、', ' 支持', ' 居民', ' 换购', ' 住房', ' 个人所得税', ' 退税', ' 优惠政策', ' 等', ' 接连', ' 出台', '。']

T4 优化

优化点

- 完整所需内容位数据集，方便后续调用
- 部分代码调整

获取文件内容，并转换为 label，sentence 两列

```
# 读取整个文件内容
with open('中文酒店评论.txt', 'r', encoding='utf-8') as file:
    lines = file.readlines()

df_txt = []
lines=lines[1:] # 去除掉读入后的第一行数据“label  sentence”，和后续需要处理的格式不相符
for line in lines:
    # 去除行尾的换行符并按空格分割
    parts = line.strip().split(' ', 1)
    if len(parts) == 2:
        label, sentence = parts
        df_txt.append([int(label), sentence])
    else:
        label = parts[0]
        df_txt.append([int(label), ''])

df_txt = pd.DataFrame(df_txt, columns=['label', 'sentence'])
df_txt['sentence'] = df_txt['sentence'].apply(lambda x: x.strip())

print(df_txt.head(6))
```

只列出代码调整部分，输出图形相同，此处不放置

```
# (1)
gd_num = len(df_txt[df_txt['label']==1])
bd_num = len(df_txt[df_txt['label']==-1])

# (2)
df_txt['length'] = df_txt['sentence'].str.len()  # 添加句子长度列
df_txt['jieba'] = df_txt['sentence'].apply(lambda x: jieba.lcut(x)) # 添加分词结果列
df_txt['count'] = df_txt['jieba'].str.len() # 添加分词数量列

# (3)
gdata = df_txt[df_txt['label']==1]['length']
bdata = df_txt[df_txt['label']==-1]['length']
```

```
13
14 # (4)
15 # 添加形容词列
16 df_txt['jieba_with_pos'] = df_txt['sentence'].apply(lambda x: pseg.lcut(x))
17 df_txt['adj'] = df_txt['jieba_with_pos'].apply(lambda word_list: [pair.word for pair in word_list
       if pair.flag == 'a'])
18 df_txt.drop(columns=['jieba_with_pos'], inplace=True)
19
20 #  (5)
21 g_word = list(itertools.chain(*df_txt[df_txt['label']==1]['adj']))
22 b_word = list(itertools.chain(*df_txt[df_txt['label']==-1]['adj']))
```