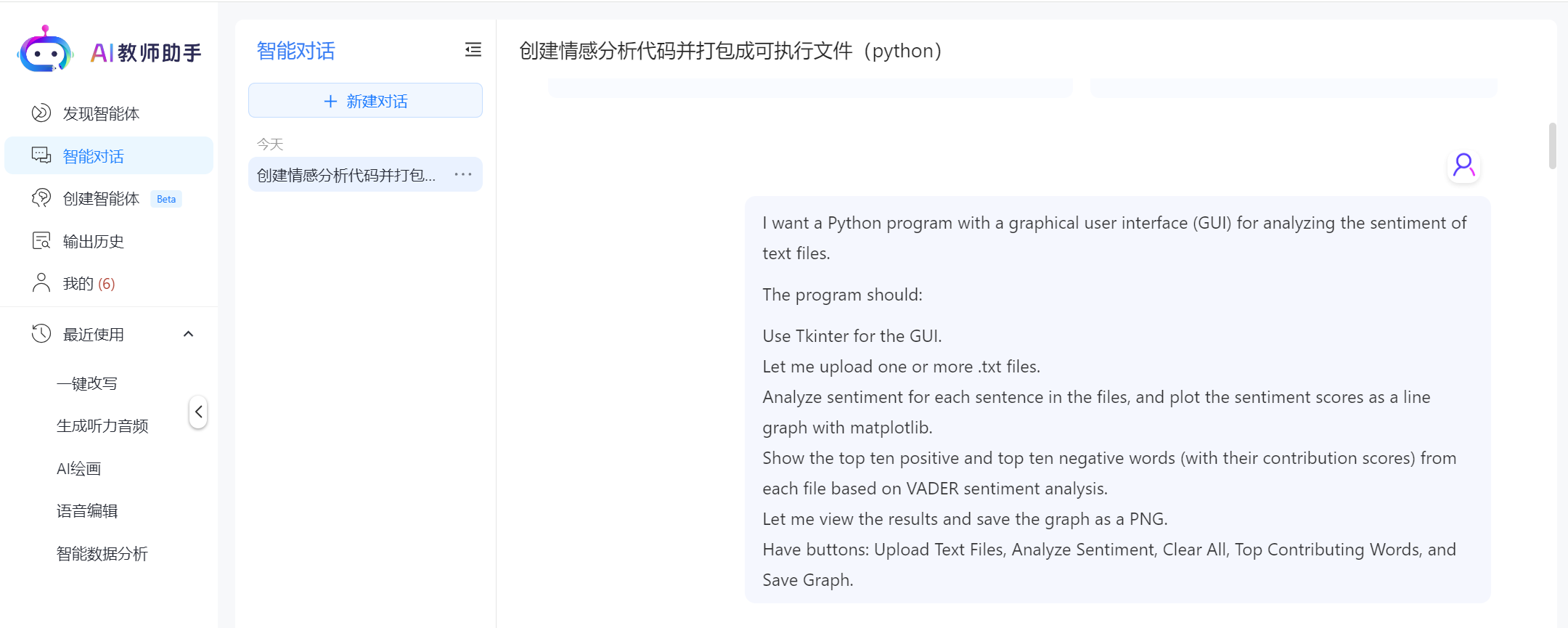
说明：

* 以下操作的前提是您的电脑已安装 Python。目前示例使用的是 Python 3.10（Windows 版本），也可使用更新版本。各版本 Python 下载地址：[https://www.python.org/downloads/windows/](https://www.python.org/downloads/windows/" \t "_blank)
* 文本编辑器使用的是 Notepad++，下载地址：[https://notepad-plus-plus.org/downloads/](https://notepad-plus-plus.org/downloads/%EF%BC%89%E3%80%82)
* 大语言模型对话使用的是外研社“AI教师助手”，模型版本为智能对话-4.0-pro。网页地址：[https://aigc.unipus.cn/home](https://aigc.unipus.cn/home" \t "_blank)

# **创建情感分析代码并打包成可执行文件（python）**

## 第一次提示词



*I want a Python program with a graphical user interface (GUI) for analyzing the sentiment of text files.*

*The program should:*

*Use Tkinter for the GUI.  
Let me upload one or more .txt files.  
Analyze sentiment for each sentence in the files, and plot the sentiment scores as a line graph with matplotlib.  
Show the top ten positive and top ten negative words (with their contribution scores) from each file based on VADER sentiment analysis.  
Let me view the results and save the graph as a PNG.  
Have buttons: Upload Text Files, Analyze Sentiment, Clear All, Top Contributing Words, and Save Graph.*

## 第一次提示词生成的代码（将生成的代码复制并保存成python文件，例如：sentsentiment1.py， 当前例子中保存在D:/myresearch/demo路径下，也可以保存在其它路径，请尽量使用英文路径名。



## 来自智能对话-4.0-pro的消息:

Below is the Python program that meets your requirements. The program uses Tkinter for the GUI, matplotlib for plotting, and nltk's VADER sentiment analysis tool for analyzing sentiment. Make sure you have the required libraries installed: nltk, matplotlib, and tkinter.

import tkinter as tk  
from tkinter import filedialog, messagebox  
import os  
from nltk.sentiment.vader import SentimentIntensityAnalyzer  
import matplotlib.pyplot as plt  
from collections import Counter  
  
# Ensure VADER data is downloaded  
import nltk  
nltk.download('vader\_lexicon')  
  
# Initialize Sentiment Analyzer  
sid = SentimentIntensityAnalyzer()  
  
class SentimentAnalyzerApp:  
 def \_\_init\_\_(self, root):  
 self.root = root  
 self.root.title("Sentiment Analyzer")  
   
 self.files = []  
 self.sentiment\_data = {}  
 self.top\_words = {}  
   
 # Buttons  
 self.upload\_button = tk.Button(root, text="Upload Text Files", command=self.upload\_files)  
 self.upload\_button.pack(pady=5)  
   
 self.analyze\_button = tk.Button(root, text="Analyze Sentiment", command=self.analyze\_sentiment)  
 self.analyze\_button.pack(pady=5)  
   
 self.clear\_button = tk.Button(root, text="Clear All", command=self.clear\_all)  
 self.clear\_button.pack(pady=5)  
   
 self.top\_words\_button = tk.Button(root, text="Top Contributing Words", command=self.show\_top\_words)  
 self.top\_words\_button.pack(pady=5)  
   
 self.save\_graph\_button = tk.Button(root, text="Save Graph", command=self.save\_graph)  
 self.save\_graph\_button.pack(pady=5)  
   
 # Text Box for displaying results  
 self.results\_text = tk.Text(root, wrap=tk.WORD, height=20, width=80)  
 self.results\_text.pack(pady=10)  
   
 def upload\_files(self):  
 self.files = filedialog.askopenfilenames(filetypes=[("Text files", "\*.txt")])  
 if self.files:  
 self.results\_text.insert(tk.END, f"Uploaded files:\n")  
 for file in self.files:  
 self.results\_text.insert(tk.END, f"{file}\n")  
 else:  
 self.results\_text.insert(tk.END, "No files uploaded.\n")  
   
 def analyze\_sentiment(self):  
 if not self.files:  
 messagebox.showwarning("No Files", "Please upload text files first.")  
 return  
   
 self.sentiment\_data = {}  
 self.top\_words = {}  
   
 for file in self.files:  
 with open(file, 'r', encoding='utf-8') as f:  
 text = f.read()  
   
 sentences = text.split('.')  
 sentiment\_scores = []  
 word\_contributions = Counter()  
   
 for sentence in sentences:  
 if sentence.strip():  
 score = sid.polarity\_scores(sentence)  
 sentiment\_scores.append(score['compound'])  
   
 # Analyze word contributions  
 words = sentence.split()  
 for word in words:  
 word\_score = sid.polarity\_scores(word)['compound']  
 if word\_score != 0:  
 word\_contributions[word] += word\_score  
   
 self.sentiment\_data[file] = sentiment\_scores  
 self.top\_words[file] = word\_contributions.most\_common()  
   
 # Plot sentiment scores  
 plt.plot(sentiment\_scores, label=os.path.basename(file))  
   
 plt.title("Sentiment Analysis")  
 plt.xlabel("Sentence Index")  
 plt.ylabel("Sentiment Score")  
 plt.legend()  
 plt.show()  
   
 self.results\_text.insert(tk.END, "Sentiment analysis completed.\n")  
   
 def clear\_all(self):  
 self.files = []  
 self.sentiment\_data = {}  
 self.top\_words = {}  
 self.results\_text.delete(1.0, tk.END)  
 plt.clf()  
 self.results\_text.insert(tk.END, "Cleared all data.\n")  
   
 def show\_top\_words(self):  
 if not self.top\_words:  
 messagebox.showwarning("No Analysis", "Please analyze sentiment first.")  
 return  
   
 self.results\_text.insert(tk.END, "Top Contributing Words:\n")  
 for file, words in self.top\_words.items():  
 self.results\_text.insert(tk.END, f"\nFile: {file}\n")  
 self.results\_text.insert(tk.END, "Top 10 Positive Words:\n")  
 positive\_words = [word for word in words if word[1] > 0][:10]  
 for word, score in positive\_words:  
 self.results\_text.insert(tk.END, f"{word}: {score:.2f}\n")  
   
 self.results\_text.insert(tk.END, "Top 10 Negative Words:\n")  
 negative\_words = [word for word in words if word[1] < 0][:10]  
 for word, score in negative\_words:  
 self.results\_text.insert(tk.END, f"{word}: {score:.2f}\n")  
   
 def save\_graph(self):  
 if not self.sentiment\_data:  
 messagebox.showwarning("No Analysis", "Please analyze sentiment first.")  
 return  
   
 save\_path = filedialog.asksaveasfilename(defaultextension=".png", filetypes=[("PNG files", "\*.png")])  
 if save\_path:  
 plt.savefig(save\_path)  
 self.results\_text.insert(tk.END, f"Graph saved as {save\_path}\n")  
 else:  
 self.results\_text.insert(tk.END, "Save operation canceled.\n")  
  
# Create the Tkinter root window  
root = tk.Tk()  
app = SentimentAnalyzerApp(root)  
root.mainloop()

### Instructions to Run the Code:

1. **Install Required Libraries**:  
   Ensure you have the required libraries installed. You can install them using:

* pip install nltk matplotlib

1. **Run the Program**:  
   Save the code to a file (e.g., sentiment\_analyzer\_gui.py) and run it:

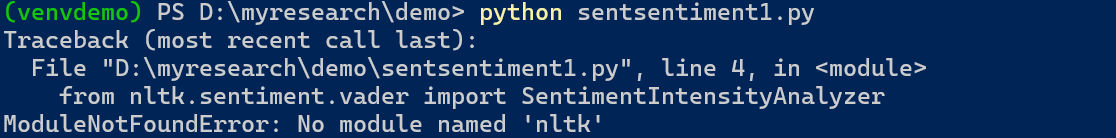
* python sentiment\_analyzer\_gui.py

1. **Use the GUI**:
   * Click "Upload Text Files" to upload .txt files.
   * Click "Analyze Sentiment" to perform sentiment analysis and plot the results.
   * Click "Top Contributing Words" to view the top positive and negative words.
   * Click "Save Graph" to save the sentiment plot as a PNG file.
   * Click "Clear All" to reset the application.

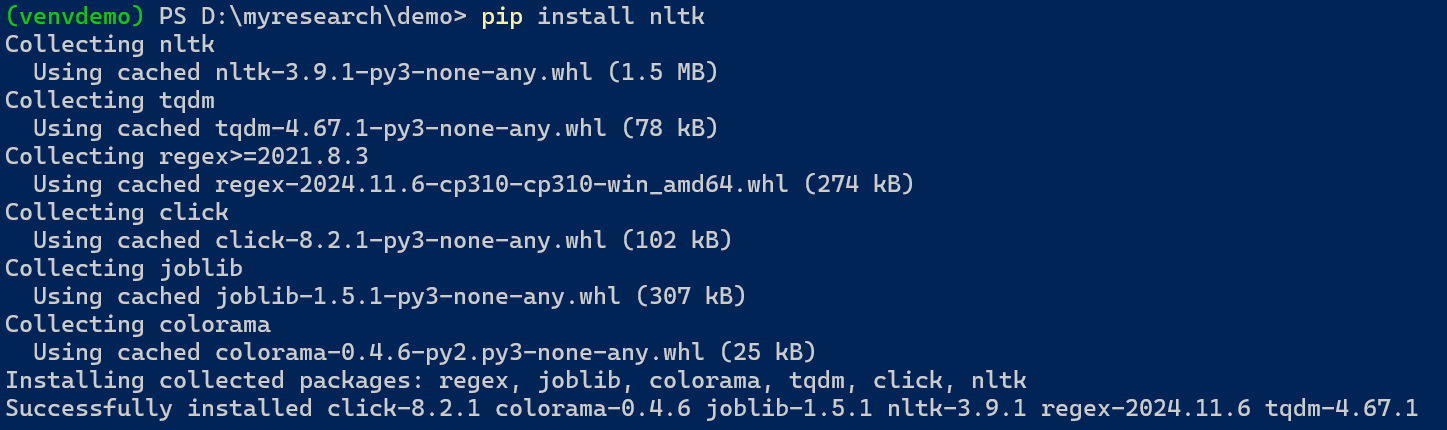
Let me know if you need further assistance!



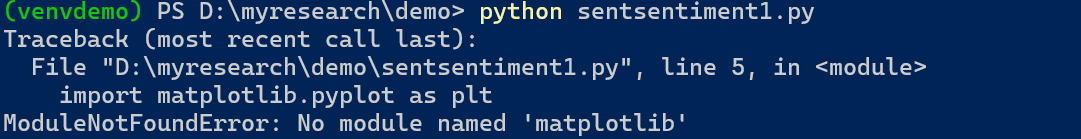
## 运行第一次提示词生成的代码（sentsentiment1.py），系统提示未安装nltk库。



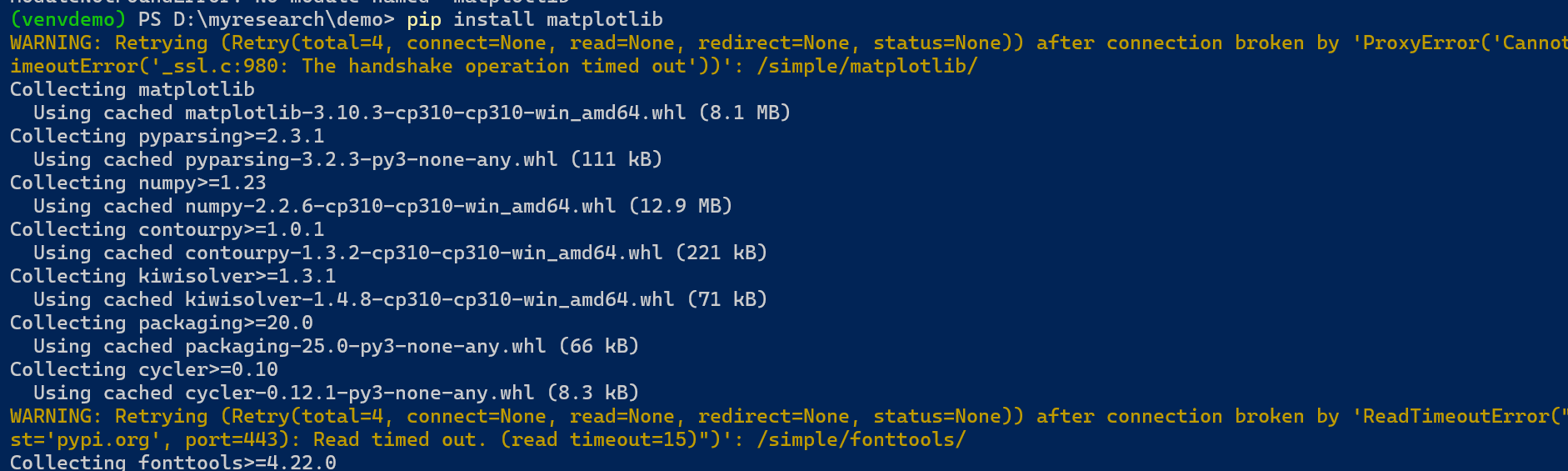
## 根据运行结果按需要安装nltk库（输入pip install nltk即可）



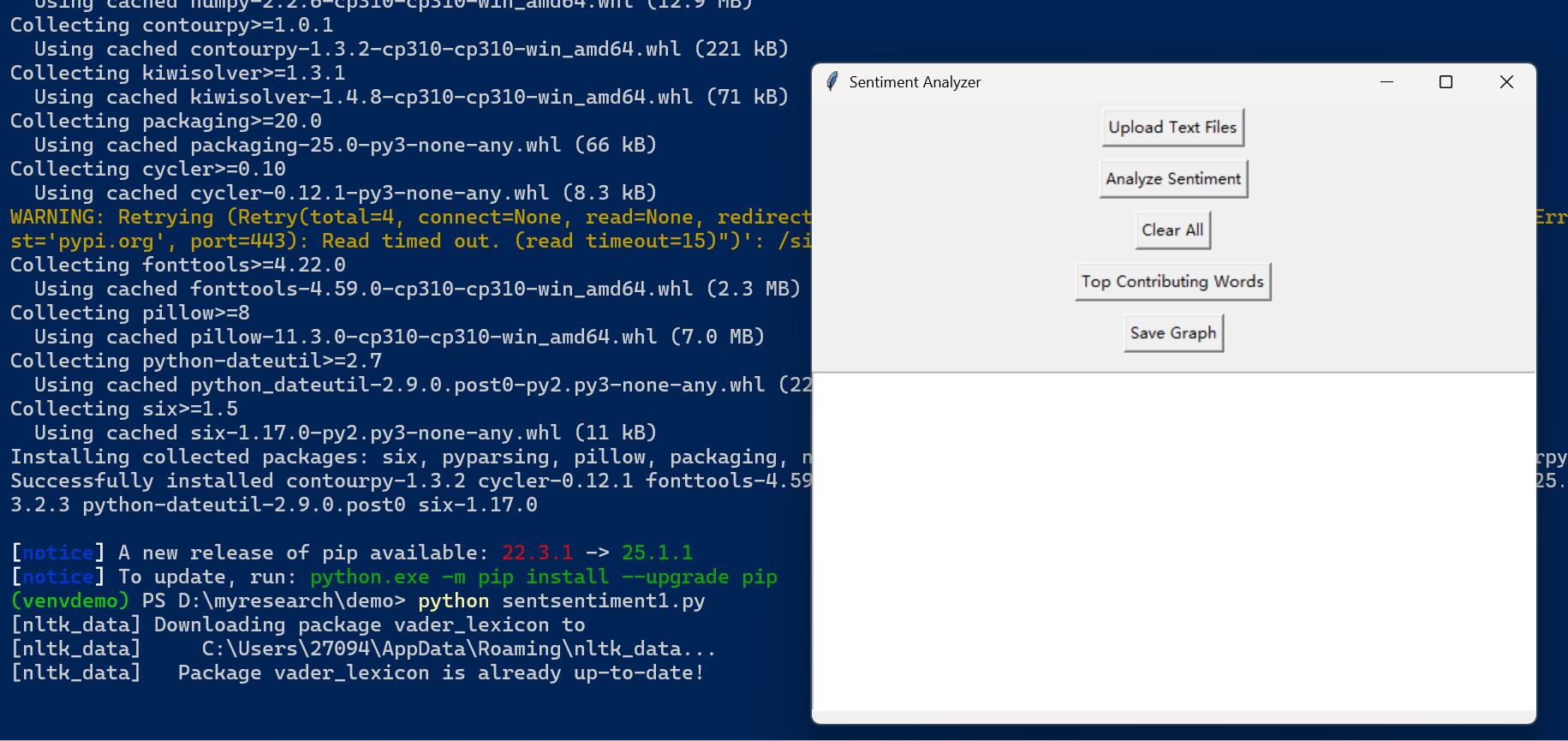
## 再次第一次提示词生成的代码（sentsentiment1.py），系统提示未安装matplotlib库



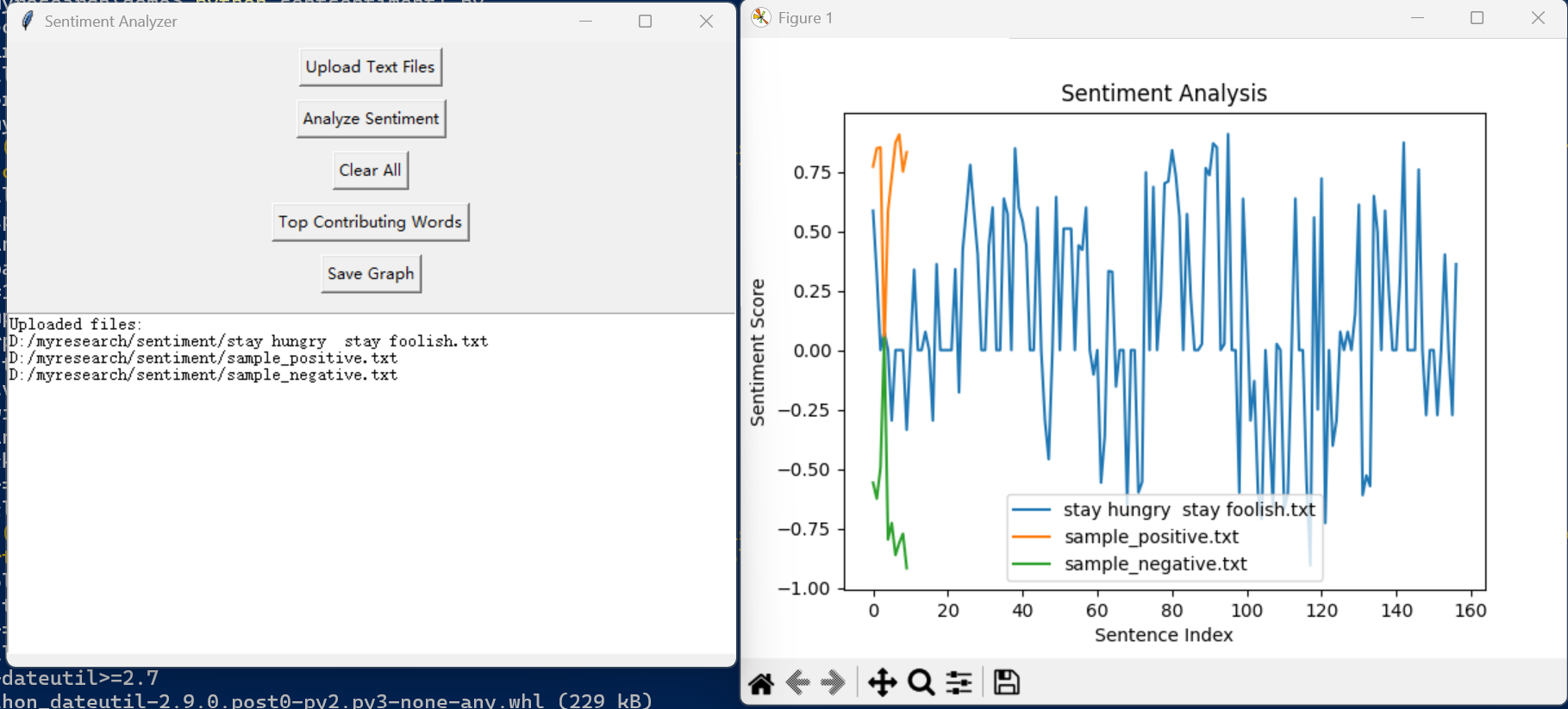
## 根据运行结果按需要安装matplotlib库（输入pip install matplotlib即可,需要一点时间）



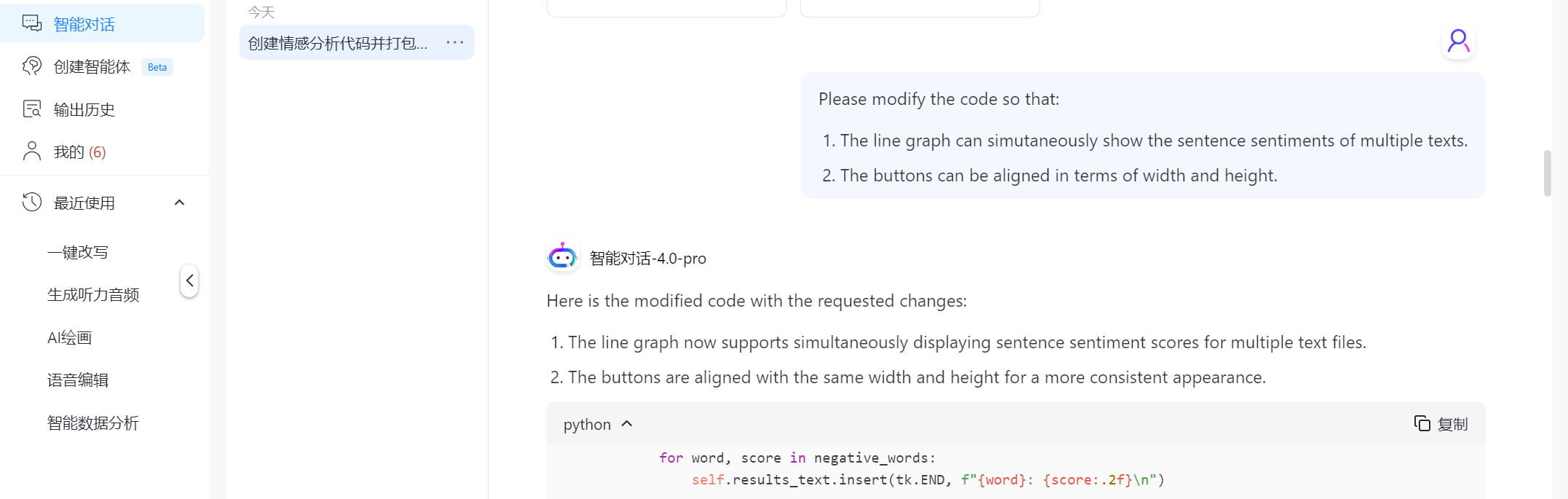
## 再次第一次提示词生成的代码（sentsentiment1.py），得到如下运行结果。



## 测试程序运行情况



## 第二次提示词，要求对外观做修改



## 

*Please modify the code so that:*

1. *The line graph can simutaneously show the sentence sentiments of multiple texts.*
2. *The buttons can be aligned in terms of width and height.*

## 来自智能对话-4.0-pro的消息:

Here is the modified code with the requested changes:

1. The line graph now supports simultaneously displaying sentence sentiment scores for multiple text files.
2. The buttons are aligned with the same width and height for a more consistent appearance.

import tkinter as tk  
from tkinter import filedialog, messagebox  
import os  
from nltk.sentiment.vader import SentimentIntensityAnalyzer  
import matplotlib.pyplot as plt  
from collections import Counter  
  
# Ensure VADER data is downloaded  
import nltk  
nltk.download('vader\_lexicon')  
  
# Initialize Sentiment Analyzer  
sid = SentimentIntensityAnalyzer()  
  
class SentimentAnalyzerApp:  
 def \_\_init\_\_(self, root):  
 self.root = root  
 self.root.title("Sentiment Analyzer")  
   
 self.files = []  
 self.sentiment\_data = {}  
 self.top\_words = {}  
   
 # Define button width and height  
 button\_width = 20  
 button\_height = 2  
   
 # Buttons  
 self.upload\_button = tk.Button(root, text="Upload Text Files", command=self.upload\_files, width=button\_width, height=button\_height)  
 self.upload\_button.pack(pady=5)  
   
 self.analyze\_button = tk.Button(root, text="Analyze Sentiment", command=self.analyze\_sentiment, width=button\_width, height=button\_height)  
 self.analyze\_button.pack(pady=5)  
   
 self.clear\_button = tk.Button(root, text="Clear All", command=self.clear\_all, width=button\_width, height=button\_height)  
 self.clear\_button.pack(pady=5)  
   
 self.top\_words\_button = tk.Button(root, text="Top Contributing Words", command=self.show\_top\_words, width=button\_width, height=button\_height)  
 self.top\_words\_button.pack(pady=5)  
   
 self.save\_graph\_button = tk.Button(root, text="Save Graph", command=self.save\_graph, width=button\_width, height=button\_height)  
 self.save\_graph\_button.pack(pady=5)  
   
 # Text Box for displaying results  
 self.results\_text = tk.Text(root, wrap=tk.WORD, height=20, width=80)  
 self.results\_text.pack(pady=10)  
   
 def upload\_files(self):  
 self.files = filedialog.askopenfilenames(filetypes=[("Text files", "\*.txt")])  
 if self.files:  
 self.results\_text.insert(tk.END, f"Uploaded files:\n")  
 for file in self.files:  
 self.results\_text.insert(tk.END, f"{file}\n")  
 else:  
 self.results\_text.insert(tk.END, "No files uploaded.\n")  
   
 def analyze\_sentiment(self):  
 if not self.files:  
 messagebox.showwarning("No Files", "Please upload text files first.")  
 return  
   
 self.sentiment\_data = {}  
 self.top\_words = {}  
   
 plt.figure(figsize=(10, 6)) # Set the figure size  
   
 for file in self.files:  
 with open(file, 'r', encoding='utf-8') as f:  
 text = f.read()  
   
 sentences = text.split('.')  
 sentiment\_scores = []  
 word\_contributions = Counter()  
   
 for sentence in sentences:  
 if sentence.strip():  
 score = sid.polarity\_scores(sentence)  
 sentiment\_scores.append(score['compound'])  
   
 # Analyze word contributions  
 words = sentence.split()  
 for word in words:  
 word\_score = sid.polarity\_scores(word)['compound']  
 if word\_score != 0:  
 word\_contributions[word] += word\_score  
   
 self.sentiment\_data[file] = sentiment\_scores  
 self.top\_words[file] = word\_contributions.most\_common()  
   
 # Plot sentiment scores for each file  
 plt.plot(sentiment\_scores, label=os.path.basename(file))  
   
 plt.title("Sentiment Analysis")  
 plt.xlabel("Sentence Index")  
 plt.ylabel("Sentiment Score")  
 plt.legend()  
 plt.grid()  
 plt.tight\_layout()  
 plt.show()  
   
 self.results\_text.insert(tk.END, "Sentiment analysis completed.\n")  
   
 def clear\_all(self):  
 self.files = []  
 self.sentiment\_data = {}  
 self.top\_words = {}  
 self.results\_text.delete(1.0, tk.END)  
 plt.clf()  
 self.results\_text.insert(tk.END, "Cleared all data.\n")  
   
 def show\_top\_words(self):  
 if not self.top\_words:  
 messagebox.showwarning("No Analysis", "Please analyze sentiment first.")  
 return  
   
 self.results\_text.insert(tk.END, "Top Contributing Words:\n")  
 for file, words in self.top\_words.items():  
 self.results\_text.insert(tk.END, f"\nFile: {file}\n")  
 self.results\_text.insert(tk.END, "Top 10 Positive Words:\n")  
 positive\_words = [word for word in words if word[1] > 0][:10]  
 for word, score in positive\_words:  
 self.results\_text.insert(tk.END, f"{word}: {score:.2f}\n")  
   
 self.results\_text.insert(tk.END, "Top 10 Negative Words:\n")  
 negative\_words = [word for word in words if word[1] < 0][:10]  
 for word, score in negative\_words:  
 self.results\_text.insert(tk.END, f"{word}: {score:.2f}\n")  
   
 def save\_graph(self):  
 if not self.sentiment\_data:  
 messagebox.showwarning("No Analysis", "Please analyze sentiment first.")  
 return  
   
 save\_path = filedialog.asksaveasfilename(defaultextension=".png", filetypes=[("PNG files", "\*.png")])  
 if save\_path:  
 plt.savefig(save\_path)  
 self.results\_text.insert(tk.END, f"Graph saved as {save\_path}\n")  
 else:  
 self.results\_text.insert(tk.END, "Save operation canceled.\n")  
  
# Create the Tkinter root window  
root = tk.Tk()  
app = SentimentAnalyzerApp(root)  
root.mainloop()

### Key Changes:

1. **Simultaneous Line Graphs**:
   * The analyze\_sentiment method now plots the sentiment scores for each text file on the same graph. Each file is represented by a different line, and the legend shows the file names.
2. **Aligned Buttons**:
   * All buttons are given the same width and height parameters for consistent alignment. The width is set to 20, and the height is set to 2.

### Instructions to Run:

1. Install the required libraries:

* pip install nltk matplotlib

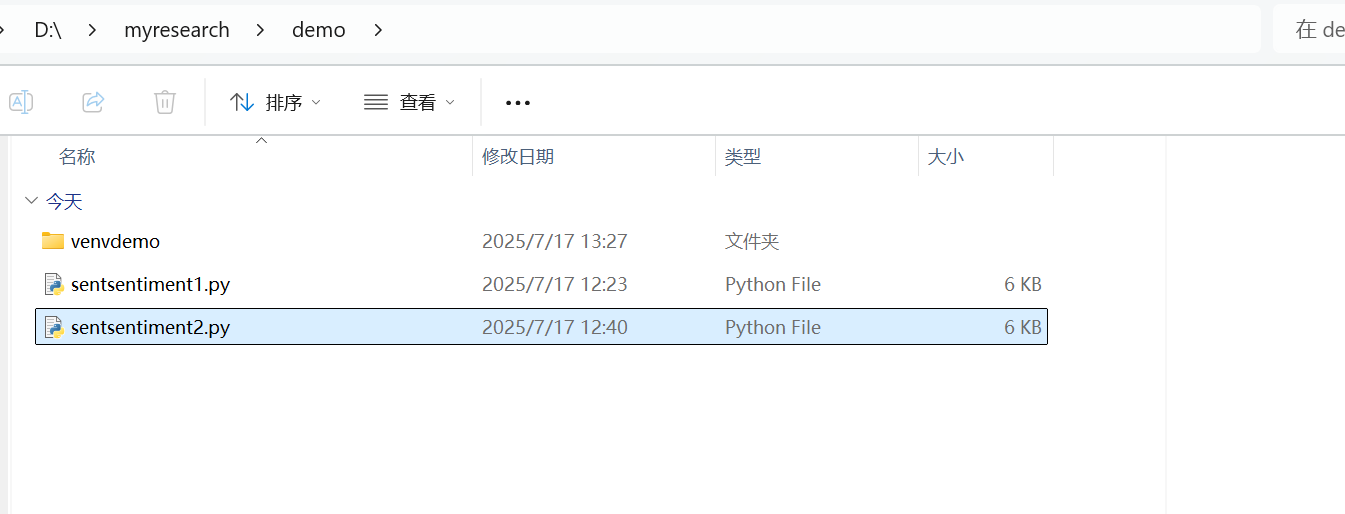
1. Save the code to a file (e.g., sentiment\_analyzer\_gui.py) and run it:

* python sentiment\_analyzer\_gui.py

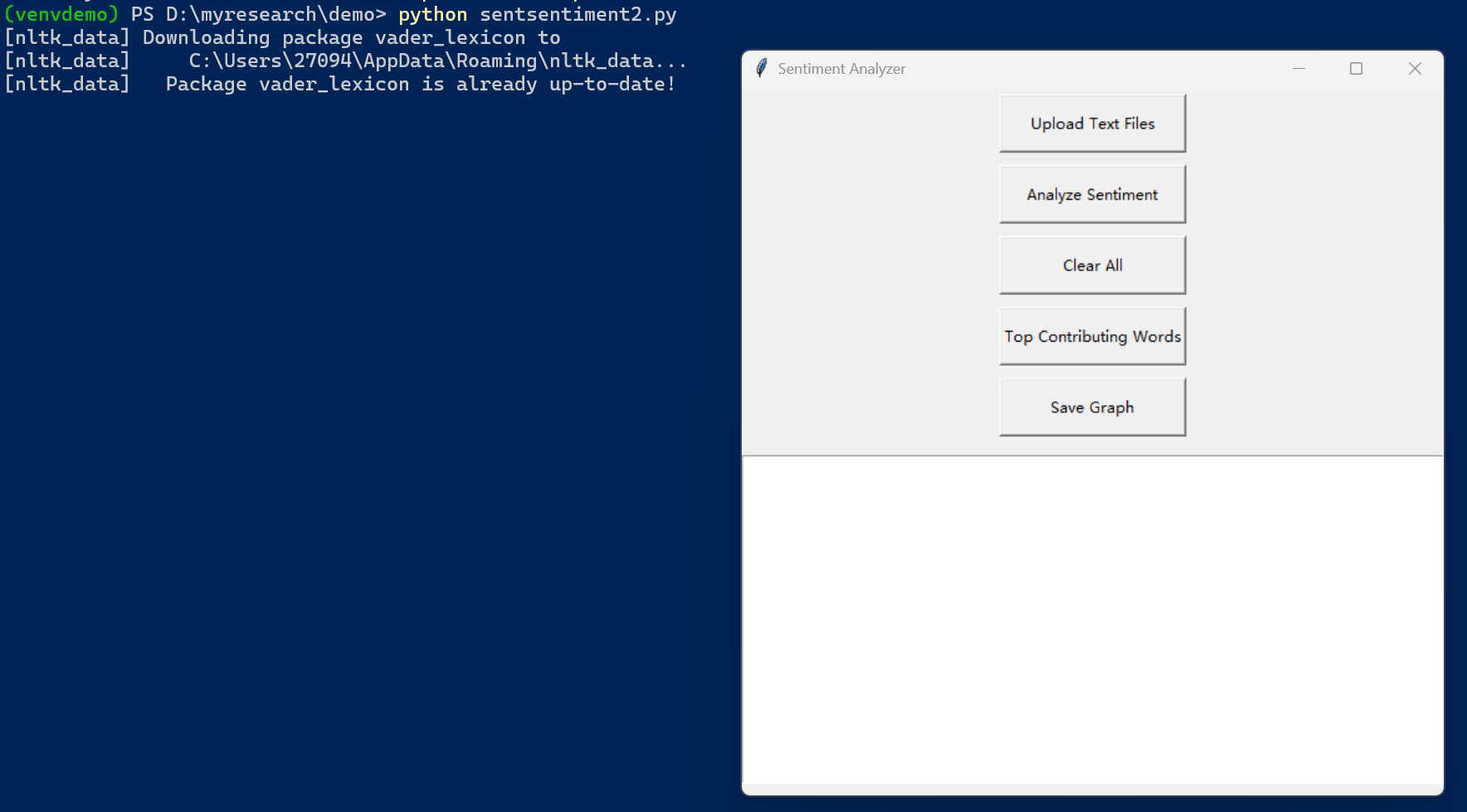
1. Use the GUI to upload .txt files, analyze sentiment, view results, and save the graph.

Let me know if you need further modifications or enhancements!

## 第二次提示词生成的代码（将生成的代码复制并保存成python文件，例如：sentsentiment2.py， 当前例子中保存在D:/myresearch/demo路径下，也可以保存在其它路径，请尽量使用英文路径名。



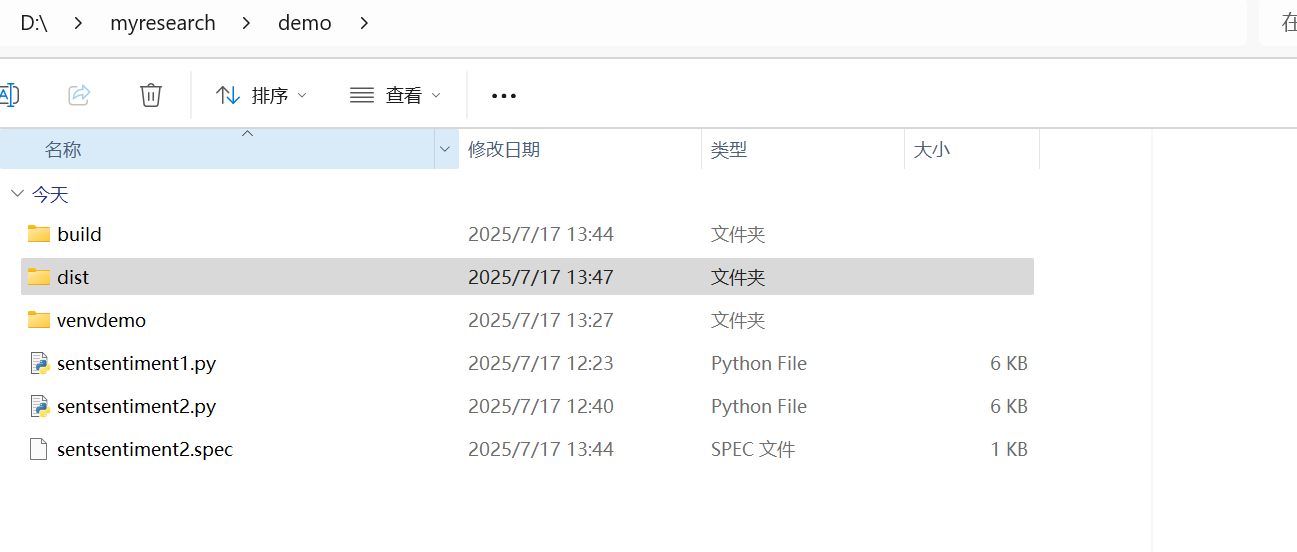
## 运行sentsentiment2.py， 查看外观。



## 根据自己需要进一步修改….

…….

## xx. 打包最终文件: pyinstaller --onefile –windowed sentsentiment2.py 需要一点时间。打包好的文件在源文件夹的dist里。



也可尝试：pyinstaller --onefile --windowed --add-data "nltk\_data/tokenizers;nltk\_data/tokenizers" --add-data "nltk\_data/sentiment;nltk\_data/sentiment" sentsentiment2.py

此时可将下载到本地的两个包打包到程序里。