

Speech Analysis Project Documentation

Project Overview

Speech Analysis is a Python-based project designed to analyze patterns of dialogue in literary texts. It extracts quoted speeches from Homer's *Iliad* and *Odyssey* and examines their frequency, length, and variation across the narrative. The goal is to understand how speech distribution contributes to storytelling structure.

Core Concepts: - Literary text mining and dialogue extraction - Statistical analysis of speech patterns - Visualization of narrative trends

Features & Objectives

1. **Automated Speech Extraction** — Uses regex to detect quoted passages.
 2. **Quantitative Metrics** — Counts total speeches and computes average length.
 3. **Visual Comparisons** — Displays dialogue patterns between *Iliad* and *Odyssey*.
 4. **Narrative Progression Analysis** — Shows speech length changes over time.
 5. **Data Visualization** — Bar charts, histograms, and smoothed line plots.
-

Usage Instructions

Run the Program

```
python speech_analysis.py
```

Input Files Required

- `../data/Homer_Illiad.txt`
- `../data/Homer_Odyssey.txt`

These should be plain text files (UTF-8 encoded). The program automatically loads, cleans, and analyzes them.

Code Structure

```
SpeechAnalysis/  
├─ speech_analysis.py      # Main analysis script
```

```
|— data/                # Contains text and word list files
|— outputs/            # Visualizations and results
```

Key Components: - **Data loading:** Reads both Homeric texts. - **Speech extraction:** Detects quoted text. - **Analysis functions:** Measures and summarizes speech characteristics. - **Visualization module:** Generates clear charts.

Main Functions

1. Loading the Texts

```
def load_texts(local_illiad: str = '../data/Homer_Illiad.txt',
               local_odyssey: str = '../data/Homer_Odyssey.txt') -> tuple[str,
str]:
    p1, p2 = Path(local_illiad), Path(local_odyssey)
    if not p1.exists() or not p2.exists():
        raise FileNotFoundError("Missing Homer text file(s).")
    return p1.read_text(encoding='utf-8', errors='ignore'),
p2.read_text(encoding='utf-8', errors='ignore')
```

Description: Loads the two epics safely and checks for missing files.

2. Extracting Quoted Speeches

```
def extract_speeches(text: str) -> list[str]:
    matches = re.findall(r'[""](.+?)[""]', text, flags=re.DOTALL)
    return matches
```

Explanation: Finds all dialogue passages enclosed in quotation marks. Supports both straight (") and curly (") quotes.

3. Analyzing Dialogue Characteristics

```
def analyze_speeches(text: str) -> tuple[int, float]:
    speeches = extract_speeches(text)
    if not speeches:
        return 0, 0.0
    lengths = [len(s.split()) for s in speeches]
    return len(speeches), mean(lengths)
```

Output: - Total number of speeches - Average words per speech

4. Collecting Speech Length Data

```
def get_speech_lengths(text: str) -> list[int]:  
    return [len(s.split()) for s in extract_speeches(text)]
```

Purpose: Generates a list of speech lengths used in later visualizations.

Visualizations

1. Average Speech Length Comparison

```
plt.figure(figsize=(6, 5))  
plt.bar(["Iliad", "Odyssey"], [avg_illiad, avg_odyssey], color=["skyblue",  
"salmon"])  
plt.title("Average Speech Length (words)")  
plt.ylabel("Average Length")  
plt.grid(axis="y", alpha=0.3)  
plt.show()
```

Interpretation: Compares the mean dialogue length between the two epics.

2. Speech Length Over Narrative Time

```
def smooth(y, window=25):  
    w = max(1, min(window, y.size))  
    kernel = np.ones(w) / w  
    return np.convolve(y, kernel, mode="same")  
  
plt.plot(smooth(illiad_lengths, 25), label="Iliad (avg)")  
plt.plot(smooth(odyssey_lengths, 25), label="Odyssey (avg)")  
plt.legend(); plt.title("Speech Length Over Narrative Time"); plt.show()
```

Interpretation: Displays how speech length fluctuates along the narrative timeline.

3. Speech Length Distribution

```
plt.hist(illiad_lengths, bins=50, alpha=0.6, label="Iliad")
plt.hist(odyssey_lengths, bins=50, alpha=0.6, label="Odyssey")
plt.title("Speech Length Distribution")
plt.xlabel("Speech length (words)"); plt.ylabel("Density"); plt.legend();
plt.show()
```

Interpretation: Shows overall distribution patterns and variance in dialogue length.

Troubleshooting Guide

Issue	Solution
Missing file error	Verify both Homer texts exist in the <code>data/</code> folder.
No speeches extracted	Ensure text contains quotation marks (<code>"</code> or <code>"</code>).
Empty plots	Check regex pattern or adjust smoothing window.
Encoding errors	Save text files as UTF-8.

Challenges & Solutions

Challenge	Solution
Inconsistent quotation marks	Added regex that supports multiple styles.
Extremely long speeches skewed averages	Used optional smoothing and median comparisons.
Visualization clutter	Limited chart bins and added smoothing function.

Additional Features

- Adjustable regex for different quote conventions.
- Optional smoothing window control.
- Histogram and bar plot exports.
- Extendable for any literary corpus.

Future Improvements

- Integrate natural language processing (NLP) for context tagging.
 - Add per-character speech statistics.
 - Include sentiment or thematic analysis of speeches.
 - Develop GUI for easier dataset management.
-

Conclusion

This Speech Analysis project combines Python text processing and visualization techniques to explore dialogue structures within epic literature. It provides insight into how narrative pacing and speech frequency shape the storytelling rhythm of Homer's works.

Authors: Project inspired by Applied NLP coursework on Homeric analysis.

End of Document