Lab 04: Speech Analysis Lab – Sentiment, Subjectivity, and Topic Modeling with Python

Course: Programming for Data Science
Professor: Dr. Carlos J. Costa

February 26, 2026

Introduction

In this lab, you will assume the role of a data analyst tasked with analyzing a famous speech. Your objective is to perform a detailed text analysis, including:

- Counting words and sentences,
- Computing sentiment and subjectivity,
- Identifying main topics via topic modeling,
- Visualizing the evolution of these metrics across the speech,
- Finally, refactoring your code into an object-oriented Python design.

This lab will help you combine text processing, data analysis, visualization, and software engineering principles in Python.

The speech to analyze is available at:
`https://raw.githubusercontent.com/masterfloss/text/refs/heads/main/jobs.txt`

Step 1: Reading and Preprocessing the Speech

1. Download the speech text from the URL.
2. Tokenize the speech into sentences using NLTK.
3. Count the total number of words in the speech.
4. Count the total number of sentences.
5. Display the first five sentences to verify correct tokenization.

Step 2: Sentiment and Subjectivity Analysis

1. Use `TextBlob` to compute the **sentiment polarity** for each sentence. Polarity ranges from -1 (negative) to 1 (positive).
2. Compute the **subjectivity** for each sentence. Subjectivity ranges from 0 (objective) to 1 (subjective).
3. Plot the evolution of sentiment across sentences.
4. Apply Gaussian smoothing to the sentiment scores to highlight trends.
5. Plot the smoothed evolution of subjectivity across sentences.

Step 3: Topic Modeling Across Speech Sections

1. Divide the speech into several sections (e.g., 5–6 sections of roughly equal length).
2. Use `CountVectorizer` to create a document-term matrix for these sections.
3. Fit a Latent Dirichlet Allocation (LDA) model to identify 3–4 main topics in the speech.
4. For each topic, list the top 10 words.
5. Plot the evolution of topic proportions across sections to visualize how themes change throughout the speech.

Step 4: Refactor into an Object-Oriented Approach

1. Design three classes:
    - `SpeechReader`: Responsible for downloading and tokenizing the speech.
    - `SpeechAnalyzer`: Responsible for computing sentiment, subjectivity, dividing into sections, and performing topic modeling.
    - `SpeechVisualizer`: Responsible for plotting sentiment, subjectivity, and topic evolution.
2. Use these classes to reproduce all previous analyses and visualizations.
3. Optionally, extend the design with additional methods, such as summarizing top topics per section or computing average sentiment and subjectivity per section.

Submission Requirements

- Submit your Python code as a `.py` file or a Jupyter Notebook (`.ipynb`).
- Include all analyses and visualizations from the previous steps.
- Ensure that the object-oriented implementation reproduces the step-by-step results.
- Be prepared to use any other text as source.
- Be prepared to explain:
    - Your design of classes and methods.
    - How you computed sentiment, subjectivity, and topics.
    - Any insights gained from visualizing trends across the speech.