AIG230 Natural Language Processing

Assignment 5	Prof: David Quispe	Winter 2026
Due Date: Feb 18 @ 11:59 PM	**Seneca Polytechnic**	Max Grade: 100 points

Instructions

In this assignment you will build a small end-to-end NLP pipeline using a corpus from NLTK. You will preprocess text, create classic vector representations, train an n-gram statistical language model, and train word embeddings. Your goal is to demonstrate that you can implement each technique, interpret outputs, and communicate results clearly.

Use this NLTK corpus:

- Option A (recommended): Gutenberg - 'austen-emma.txt' or 'carroll-alice.txt'
- Option B: Brown corpus (choose 1-2 categories, e.g., 'news' and 'romance')

Choose **ONE** option and state your choice at the top of your notebook/report.

You must complete all three parts below. Each part builds on the previous parts.

Part A - Text Preprocessing (50%)

Goal: Prepare a clean token stream and justify your choices.

A1. Load the corpus

- Load the raw text (or sentences) from your chosen NLTK corpus and print:
- Total number of characters (if raw text) or total number of sentences (if sentence-based)
- Total number of tokens BEFORE preprocessing

A2. Preprocess

Create a preprocessing function that performs:

- Lowercasing
- Tokenization
- Removal of punctuation tokens
- Optional: stopword removal (if you choose to remove stopwords, explain why)
- Optional: stemming OR lemmatization (choose one if you use it, explain why)

Report these statistics AFTER preprocessing:

- Total number of tokens
- Vocabulary size (unique tokens)
- Top 20 most frequent tokens (with counts)

A3. Reflection (short)

In 5-8 sentences, explain how your preprocessing choices could affect downstream tasks (vectorization, language modeling, and embeddings).

4. Part B - Text Representation (25%)

Goal: Compare Bag-of-Words and TF-IDF representations and interpret the results.

B1. Create documents

Split your corpus into documents. Choose ONE of the following strategies and justify it:

- Use each chapter (if available) or fixed-size chunks (e.g., 500-1000 tokens) as a document.
- Use each sentence as a document (works best for Brown with many sentences).

B2. Vectorize

Using scikit-learn:

- Create a Bag-of-Words (CountVectorizer) matrix
- Create a TF-IDF (TfidfVectorizer) matrix
- Report the shape of each matrix (num_docs x vocab_size)
- Show the top 15 TF-IDF terms for 2 different documents and interpret why they appear

B3. Similarity

Compute cosine similarity between documents using TF-IDF. Show:

- The most similar pair of documents (excluding self-similarity) with the similarity score
- One example of a surprising similarity (if any) and a short explanation
- A small similarity table (at least 5 documents) or a heatmap (optional)

6. Part C - Word Embeddings (25%)

Goal: Train Word2Vec embeddings and explore semantic similarity.

C1. Prepare training data

Create a list of sentences, where each sentence is a list of tokens, using your preprocessing pipeline. You may keep stopwords for embeddings (recommended) unless you have a clear reason not to.

C2. Train Word2Vec

Train a Word2Vec model and report your chosen hyperparameters:

- vector_size (e.g., 100)
- window (e.g., 5)
- min_count (e.g., 3)
- sg (0 for CBOW, 1 for skip-gram)
- epochs

C3. Explore similarity

Pick 5 target words that occur frequently in your corpus. For each word, show:

- Top 10 most similar words (cosine similarity)
- A short interpretation: do the neighbors make sense given your corpus?

C4. Analogies (vector arithmetic)

Run 3 analogy queries. Example format:

- king - man + woman ≈ ?

If your corpus is small, analogies may be weak. If results are poor, explain why (corpus size, domain, frequency).

Requirements and Preparation

Software:

- Python 3.10+
- Jupyter Notebook (or Google Colab)
- NLTK
- NumPy, pandas
- scikit-learn (for vectorization and evaluation)
- gensim (for Word2Vec)
- matplotlib (optional for plots)

Keep your code runnable from top to bottom (Restart + Run All should work).
Do not hardcode local file paths; load data only through NLTK.

Deliverables

You must submit the following on Blackboard:

- **One GitHub repository** containing:
    - All completed notebooks clearly stating the sections presented in the instructions
    - All the answer to the questions on the instructions

Submission Instructions

1. Upload your completed work to a GitHub repository
2. On **Blackboard**, submit:
    - A link to your GitHub repository
3. Add the instructor as a **collaborator** on your GitHub repository

Late submissions may not be accepted, as this is an in-class lab.

Academic Integrity

All submitted work must be your own. You may discuss concepts with classmates, but all code submitted must reflect your individual work unless explicitly stated otherwise.