NLP Assignment1

Due Date: 1403-09-04

Dr. Leila Safari

University Of Zanjan, Department of Computer and Electrical Engineering

1- Temporal Pattern Extraction Using Regular Expressions (20 points) – software students

You are provided with a dataset in the file narrative_text.txt that contains a narrative text. This text includes:

- **phone numbers** in various Iranian formats (e.g., +989123456789, 09123456789, or 00989123456789),
- **date patterns** in multiple formats (e.g., YYYY-MM-DD and DD/MM/YYYY), and
- **time patterns** in HH:MM:SS or HH:MM formats (seconds may be optional).

Your task is to identify and extract all instances of these patterns using **regular expressions**.

Deliverable:

- Submit a Python script that performs the extraction and outputs the results for each category.

1- DNA Sequence Analysis using Regular Expressions and Basic Statistics (20 points) – bioinformatics students

You are provided with a dataset in the file sequences.fasta containing DNA sequences in FASTA format. Each sequence includes only the bases A, T, G, and C. Your task is to perform the following analyses:

1. **Pattern Extraction:** Use regular expressions to identify and extract complex sequences of interest. For example, sequences that start with 'A', contain exactly three 'G' bases (not necessarily consecutive), and end with a 'T'.
2. **Base Composition Analysis:** Calculate the frequency of each base (A, T, G, and C) across all sequences. Identify the sequence with the highest GC content and plot a histogram showing the distribution of GC content percentages across all sequences.

Deliverables:

- A Python script that:
    - Extracts specified complex patterns using regular expressions.
    - Calculates and prints the base composition frequencies.
    - Outputs the statistical summary of sequence lengths and the GC content histogram.

Text Classification using Naive Bayes on Sample Dataset – all students

2- Naive Bayes Classification

You are provided with a small training dataset consisting of documents with their corresponding class labels. The classes are either **c** (related to Chinese language) or **j** (related to Japan). Using the Naive Bayes classifier with **Laplace smoothing**, your task is to determine the class of a new test document based on the word frequencies in the training set.

Training Data:

Doc	Words	Class
1	Chinese Beijing Chinese	c
2	Chinese Chinese Shanghai	c
3	Chinese Macao	c
4	Tokyo Japan Chinese	j
5	Chinese Tokyo Shanghai	?

2-1: Handwritten Calculation (25 points)

Using the **Naive Bayes classification formula** and **Laplace smoothing**, calculate the probability that **Document 5** belongs to class **c** and class **j**.

Task:

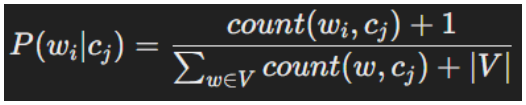
1. **Calculate the priors**:
    - P(c) and P(j)
2. **Calculate the conditional probabilities** for each word in **Document 5** (Chinese Tokyo Shanghai) for both classes **c** and **j**.
3. **Use Laplace smoothing** to compute the probabilities and determine the most likely class for **Document 5**.

Vocabulary:

The vocabulary consists of the following words: Chinese, Beijing, Shanghai, Macao, Tokyo, Japan. Use this for calculating the conditional probabilities.

Conditional Probabilities:

Use the formula for **Laplace smoothing**:



$$P(w_i|c_j) = \frac{count(w_i, c_j) + 1}{\sum_{w \in V} count(w, c_j) + |V|}$$

Where:

- count(wi,cj) is the count of word wiw_iwi in class cjc_jcj,
- V is the vocabulary size.

2-2: Python Code Implementation (25 points)

After performing the handwritten calculations, you will implement the Naive Bayes classification for **Document 5** using Python. You will follow the same steps programmatically as done manually.

Task:

1. **Write Python code** to:
   - Compute the **prior probabilities** for each class c and j,
   - Calculate the **conditional probabilities** for each word in the vocabulary for both classes, using **Laplace smoothing**.
2. **Apply Naive Bayes** to calculate P(c|d5) and P(j|d5) for **Document 5**.
3. **Output**:
   - The calculated probabilities for both classes.
   - The predicted class for **Document 5**.

Python Code Deliverables:

- Python code that implements the following steps:
  - Create a function to calculate the **prior probabilities** P(c) and P(j),
  - Create a function to compute the **conditional probabilities** using **Laplace smoothing** for each word in the vocabulary,
  - Create a function that applies the **Naive Bayes classification formula** and calculates P(c|d5) and P(j|d5),
  - Predict and print the most likely class for **Document 5**.

Code Output:

- Print the computed probabilities P(c|d5) and P(j|d5),
- Print the predicted class for **Document 5** based on the computed probabilities.

3- In-Depth Assignment: Sentiment Analysis Using Naive Bayes on Movie Reviews – all students

3. In this section of assignment, you will implement the Naive Bayes algorithm for text classification using the **movie review dataset** available in the NLTK library. You will apply the model to perform **sentiment analysis**, classifying reviews as either positive or negative. By the end of the assignment, you will understand the process of building a Naive Bayes model for text classification, preprocessing text data, and evaluating model performance using various metrics.

3-1: Data Preprocessing (20 points)

Explain the steps involved in preparing the text data for classification. Your preprocessing should include:

- **Tokenization** (splitting the text into individual words),
- **Lowercasing** (converting all words to lowercase),
- **Removing punctuation and stopwords** (words like "the," "and," etc.),
- **Stemming or Lemmatization** (optional).

You can use **Python code** that implements text classification using Naive Bayes on a sample dataset to gain familiarity with the technique.

Deliverables:

- A description of the preprocessing steps, with a focus on the vectorization technique.
- Python code snippets for tokenization, stopword removal, and vectorization.
- A brief justification for your choice of preprocessing and vectorization.

3-2: Model Implementation (20 points)

1. **Implement the Naive Bayes classifier** using the preprocessed dataset.
   - Split the dataset into **training and testing sets** (e.g., an 80/20 split) using train_test_split from sklearn.
   - Train a **Multinomial Naive Bayes model** on the training data, as it is suitable for text classification.
2. **Evaluate the model's performance**:
   - Calculate and report the following metrics: **accuracy**, **precision**, **recall**, and **F1-score** using sklearn metrics.
   - Visualize the **confusion matrix** using matplotlib or seaborn.
3. **Provide the code** for training, evaluation, and visualization.

*Optional: (10 points)

Additionally, use vectorization technique such as **Bag of Words (BoW)** or **TF-IDF** to convert the text into numerical features suitable for the Naive Bayes model. Use Python libraries like CountVectorizer or TfidfVectorizer from sklearn. Then compare this results with implemented model.

Deliverables:

- Python code for training the **Multinomial Naive Bayes classifier**,
- Evaluation metrics including **accuracy**, **precision**, **recall**, **F1-score**, and the **confusion matrix visualization**,
- Results and model evaluation discussed in the final report.

Good Luck!
TA Team (Moslem Amini, Rana Naibi)