Assignment: Word Embeddings Comparison, Visualization, and Evaluation

Due Date: 15/11

Objective

In this assignment, you will preprocess a text dataset, generate and compare Word2Vec CBOW, Word2Vec Skip-gram, and GloVe embeddings, and visualize them. Additionally, you'll evaluate the embeddings to determine which model best captures the semantic relationships within the dataset.

Instructions

1. **Dataset**:

   https://www.kaggle.com/datasets/jp797498e/twitter-entity-sentiment-analysis

2. **Preprocessing**:
   - Clean the text data by:
     - Removing punctuation, stop words, and special characters.
     - Lowercasing all text.
     - Tokenizing sentences into words.
     - lemmatize or stem words to normalize the vocabulary.
   - Include screenshots or code snippets in your Word document or notebook cells to show each preprocessing step.
3. **Word Embedding Models**:
   - Implement or load the following word embeddings:
     - **Word2Vec CBOW**: Uses context to predict a target word.
     - **Word2Vec Skip-gram**: Predicts the surrounding context from a given word.
     - **GloVe**: Uses word co-occurrence statistics to capture global word relationships..
4. **Comparative Analysis**:
   - Document the differences between embeddings generated by CBOW, Skip-gram, and GloVe. Use screenshots in your Word document or notebook cells to show differences in word relationships or similarities among the embeddings.
5. **Visualization**:
   - Use **PCA** and **t-SNE** to reduce embedding dimensions and create 2D visualizations.
   - Plot the embeddings to observe clustering and spatial relationships.
   - Include screenshots of these plots, and explain any patterns observed in each model's clusters.
6. **Evaluation**:
   - Assess each embedding method based on:
     - **Semantic Similarity**: How well each model captures similar words or related concepts (e.g., are synonyms close together?).
     - **Topic Coherence**: Are words from the same topic or context (e.g., technology, emotions) grouped together?

        - **Performance on Simple Tasks**: For an extra step, you could use a small, simple classification task (if feasible) to compare accuracy across embeddings (optional).
    - Document your findings, highlighting which model best captures the relationships within the dataset. Discuss any limitations or observations.
7. **Interpretation and Conclusion**:
    - Summarize your findings on which embedding model you believe is the most effective for this dataset.
    - Discuss any improvements or alternative methods you might consider if given more time or resources.

Submission Requirements:

- Submit a Word document with screenshots and explanations, and a Notebook with annotated cells showing each step.
- Ensure all code is well-commented, and any visualizations include captions to explain key findings.
- Include a final evaluation section summarizing which embedding performed best and why.

Libraries

1. **Data Preprocessing**:
    - **NLTK** (`nltk`): Useful for tokenization, removing stop words, and other text preprocessing tasks.
    - **spaCy** (`spacy`): Provides advanced NLP tools, including lemmatization and efficient tokenization.
    - **re** (Python's regular expressions library): Helpful for cleaning text by removing unwanted characters and patterns.
2. **Word Embedding Models**:
    - **Word2Vec CBOW and Skip-gram**:
        - **Gensim** (`gensim.models.Word2Vec`): Gensim is a popular library for implementing both CBOW and Skip-gram Word2Vec models. It offers functions to train custom embeddings as well as load pre-trained models.
    - **GloVe**:
        - **Gensim** (`gensim.models.KeyedVectors`): GloVe embeddings can be loaded using Gensim if pre-trained embeddings (e.g., from Stanford NLP) are available.
        - **torchtext** (for PyTorch users): Allows loading GloVe embeddings directly for deep learning tasks if using PyTorch.
3. **Dimensionality Reduction and Visualization**:
    - **PCA**:
        - **Scikit-Learn** (`sklearn.decomposition.PCA`): Scikit-Learn provides an easy-to-use PCA function for reducing word embeddings to two dimensions for visualization.

    - **t-SNE**:
        - **Scikit-Learn** (`sklearn.manifold.TSNE`): Also available in Scikit-Learn, t-SNE is effective for visualizing high-dimensional embeddings.
4. **Plotting and Visualization**:
    - **Matplotlib** (`matplotlib.pyplot`): Essential for creating 2D scatter plots for PCA or t-SNE visualizations.
    - **Seaborn** (`seaborn`): An extension of Matplotlib, which provides additional options for enhancing visualizations and color-coding clusters.