Project Title:

Training Word2Vec method to learn word embeddings and evaluate them.

Description of the corpus:

The chosen corpus for this analysis is the Brown Corpus from the Natural Language Toolkit (NLTK), a comprehensive collection of American English texts widely used in natural language processing research and applications.

The Brown Corpus comprises diverse genres of text, including fiction, news, reviews, and more. With a total of 57,340 sentences, it provides a substantial and varied dataset for language modeling and analysis.

Preprocessing Steps:

- Sentence Tokenization: Sentences were tokenized using NLTK's nltk.corpus.brown.sents() method, treating each sentence as an individual unit for analysis.

- Lowercasing: All tokens in each sentence were converted to lowercase using list comprehension. This step ensures uniformity and reduces dimensionality in the dataset.

- Lemmatization: The WordNet lemmatizer was applied to each token, reducing words to their base or root form for enhanced semantic analysis.

Task 1:

first_model:

- Skip-Gram (sg=1): The Skip-Gram architecture was chosen for first_model, where the model predicts the context words given the target word. This approach is often effective in capturing syntactic relationships and is suitable for small to medium-sized datasets like the Brown Corpus.

- Epochs (epochs=5): The model was trained for 5 epochs, striking a balance between computational efficiency and model convergence. A moderate number of epochs helps prevent overfitting and ensures the model generalizes well to unseen data.

- Vector Size (vector_size=100): Word vectors were set to be of dimension 100. This size strikes a balance between capturing intricate semantic relationships and computational efficiency.

- Minimum Count (min_count=5): Words occurring less than 5 times were excluded from the vocabulary. This helps in focusing on frequently occurring words, enhancing the quality of learned embeddings.

second_model:

- Continuous Bag of Words (sg=0): The Continuous Bag of Words (CBOW) architecture was chosen for second_model. CBOW predicts the target word given its context, making it suitable for corpora where context carries substantial meaning, such as the Brown Corpus.

- Epochs (epochs=5): Similar to first_model, second_model was trained for 5 epochs for consistency and to ensure a fair comparison between the two models.

- Vector Size (vector_size=100): The dimensionality of the word vectors was set to 100, aligning with first_model for a fair comparative analysis.

- Minimum Count (min_count=5): Like first_model, a minimum count of 5 was chosen to filter out less frequent words and focus on building meaningful embeddings.

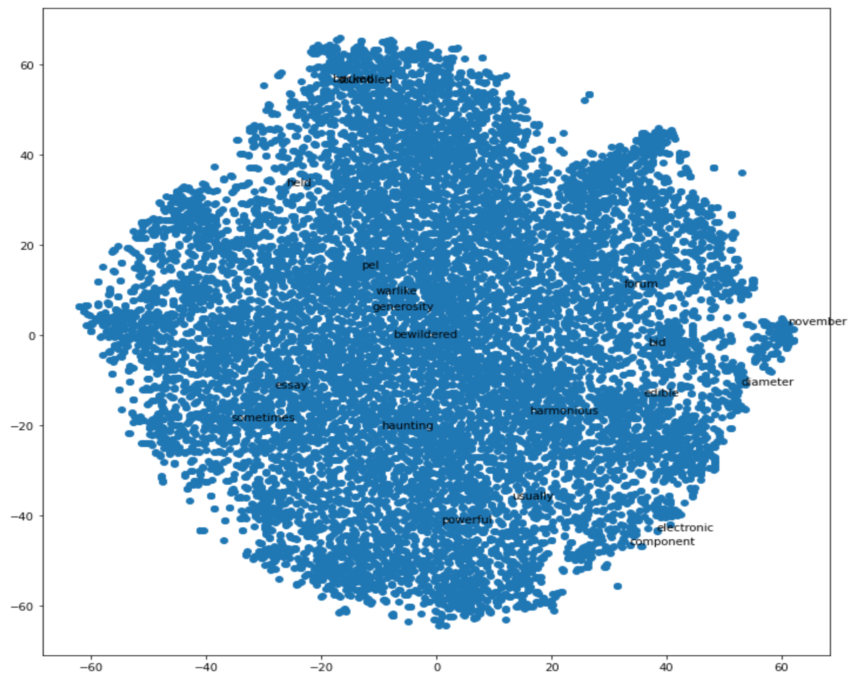
Rational:

- Architecture Choice (Skip-Gram vs. CBOW): The choice between Skip-Gram and CBOW depends on the nature of the corpus. Skip-Gram is often suitable for capturing syntactic relationships, while CBOW is effective in semantic-rich contexts. By training models with both architectures, we aim to evaluate their respective performances on the Brown Corpus.

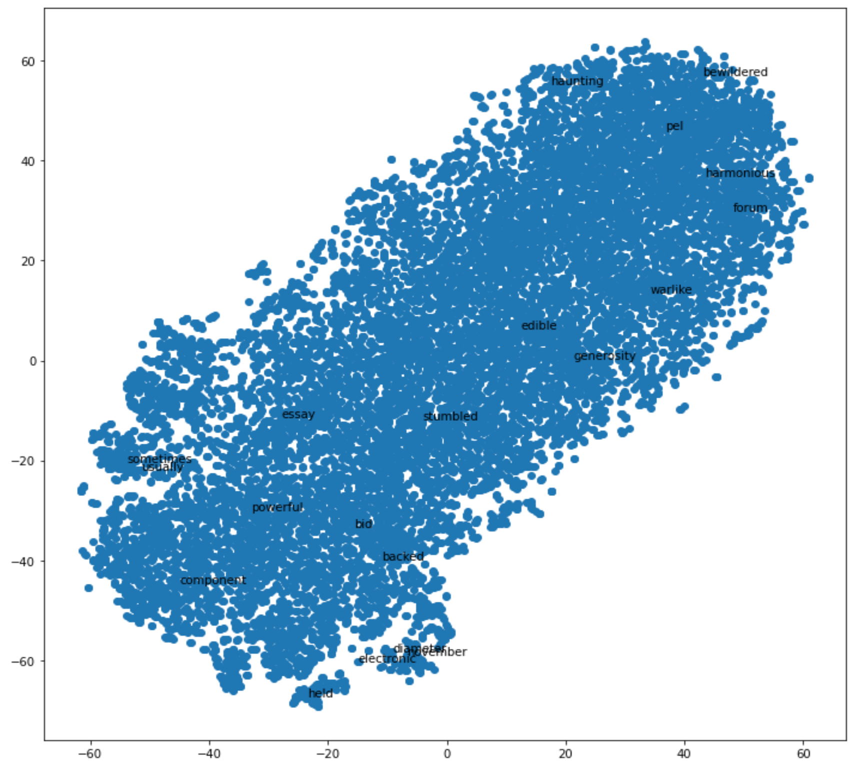
- Vector Size and Minimum Count: The decisions for vector size and minimum count were made to strike a balance between the model's capacity to capture intricate relationships and the need to manage computational resources. A vector size of 100 is commonly used, and a minimum count of 5 helps in excluding rare words that might not contribute significantly to the embeddings.

Task 2:

First_model:



Second_model:



A1_helper.py file is used for this task . The provided code, consisting of two functions (reduce_dimensions and plot_with_matplotlib), enables the exploration of spatial relationships between words within the context of a Word2Vec model. The purpose is to offer a clear and intuitive representation of the semantic and syntactic similarities captured by the embeddings.

The reduce_dimensions function applies t-SNE to reduce the dimensions of word vectors obtained from a Word2Vec model to 2D. The resulting coordinates (x_vals and y_vals) and word labels (labels) are returned.

The plot_with_matplotlib function utilizes Matplotlib to create a scatter plot of the 2D word embeddings. The plot size is set to 12x12 inches for clear visualization.

Comments and Observations:

- Proximity and Similarity:
  The proximity of points on the scatter plot reflects the semantic and syntactic similarity between words. Words that are conceptually or contextually similar should appear closer to each other, forming clusters or groups on the plot.
- Comparative Analysis:
  The visualization output is valuable for comparing word embeddings generated by different models or configurations. For instance, it allows a side-by-side examination of the spatial distribution of words from first_model and second_model.
- Outliers and Clusters:
  Outliers, represented by points that are positioned differently from the majority, may indicate words with unique relationships or meanings. Clusters of points suggest groups of words that share common characteristics or contexts.

(In the plot associated with the first model, the terms 'usually' and 'sometimes' exhibit a notable separation, whereas in the plot linked to the second model, they are in close proximity to each other.)

Task 3:

Model	Correlation Scores
pre-trained Google News(wv)	((0.3908885216390469, 0.29823863007156837), SpearmanrResult(correlation=0.29538231539940274, pvalue=0.44030468679916945), 10.0)
first_model	((-0.10216455033122428, 0.778833119642074), SpearmanrResult(correlation=-0.2935903373670295, pvalue=0.4103240796393395), 0.0)
second_model	((0.1414891148135379, 0.6966145737476537), SpearmanrResult(correlation=0.20184335693983277, pvalue=0.5760205613999424), 0.0)

We used a Word2Vec model and evaluating its performance on word similarity tasks using the evaluate_word_pairs function. This function typically takes a file containing word pairs and human-

annotated similarity scores, and it calculates the similarity scores predicted by the model. The Pearson correlation coefficient is then computed between the predicted scores and the human-annotated scores.
Here's a interpretation of the results:

- Overall Performance:
  The pre-trained Google News(wv) model has a Pearson correlation of 0.39, indicating a moderate positive correlation between the model's predicted scores and human-annotated scores.
  The second model has a Pearson correlation of 0.141, which also suggests a positive correlation but weaker than the first model.
  The first model has a Pearson correlation of -0.102, indicating a negative correlation. This suggests that the model's predicted scores are inversely related to the human-annotated scores.

- Interpretation :
  - A positive correlation suggests that the model's predictions align with human judgments to some extent.
  - A negative correlation suggests that the model's predictions do not align well with human judgments.

- Magnitude of Correlation:
  - A correlation closer to 1 or -1 indicates a stronger relationship between the predicted and human-annotated scores.
  - A correlation closer to 0 indicates a weaker relationship.

- Consideration:
  It's important to note that the absolute values of the correlations might be more indicative of the model's performance rather than the sign. For instance, a correlation of -0.102 might be considered weak, but it depends on the specific task and dataset.

Task 4:

new_selected_words= ['bewildered', 'powerful', 'held', 'harmonious', 'stumbled']

results:

Model	result
pre-trained Google News(wv)	{'bewildered': ['perplexed', 'bemused', 'befuddled', 'baffled', 'dumbfounded'], 'powerful': ['Powerful', 'potent', 'influential', 'poweful', 'formidable'], 'held': ['holding', 'hold', 'Held', 'holds', 'attended'], 'harmonious': ['harmony', 'harmoniously', 'amity', 'harmonious_coexistence', 'congenial'], 'stumbled': ['stumbling', 'stumble', 'slipped', 'wandered', 'bumbled']}
first_model	{'bewildered': ['mastered', 'villain', 'sprayed', 'bravado', 'volunteered', 'symbolized'], 'powerful': ['minority', 'protective', 'receives', 'aggressive', 'rarely', 'regarded'], 'held': ['joined', 'struck', 'opened', 'cabinet', 'rang', 'gathered'], 'harmonious': ['observance', 'contraceptive', 'terminus', 'cohesion', 'inadequacy', 'affords'], 'stumbled': ['hurried', 'jerked', 'sank', 'smashed', 'leaning', 'raced']}

second_model	{'bewildered': ['sober', 'doctrine', 'statute', 'greatness', 'notion'], 'powerful': ['static', 'budgeting', 'decoration', 'occurring', 'obscure'], 'held': ['entered', 'followed', 'opened', 'dropped', 'reached'], 'harmonious': ['philosophical', 'well-being', 'secondary', 'administrative', "state's"], 'stumbled': ['hurried', 'smashed', 'climbed', 'schoolhouse', 'slipped']}

Explanation:

- These results showcase the semantic relationships captured by each Word2Vec model. Words with similar meanings or contexts are grouped together.
- Different models may yield different associations based on their training data and the specific context they have learned.
- The “topn parameter” controls the number of similar words returned for each query word.
- These results can be useful for tasks like word embedding-based semantic similarity and understanding the context in which words are used.