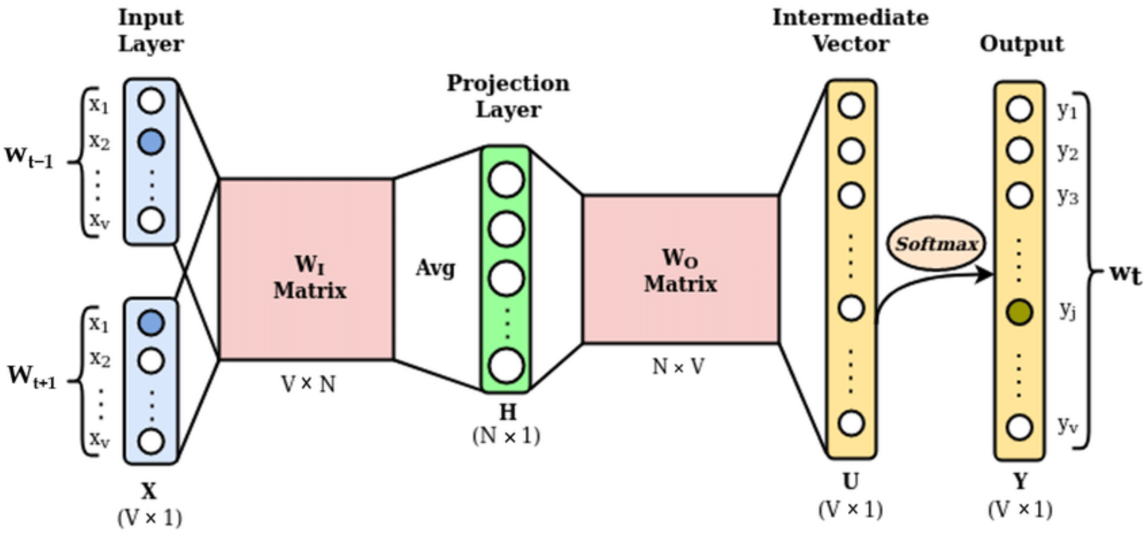
ASSIGNMENT 5: CONTINUOUS BAG OF WORDS (CBOW) MODEL

The **Continuous Bag of Words (CBOW)** model is a popular word embedding model in deep learning, especially in **Natural Language Processing (NLP)**. It is a part of the **Word2Vec** family, used for learning vector representations of words, which capture semantic and syntactic information. The CBOW model helps understand the relationships between words by predicting a target word based on its surrounding context words.



CBOW Model Explanation

In CBOW, given a set of context words (surrounding words), the model aims to predict the target word. For example, in the sentence: "The cat sat on the mat," if "cat," "on," "the," and "mat" are the context words, the model will try to predict "sat."

Working of the CBOW Model

1. **Input Layer**:
    - The input consists of context words surrounding a target word. For example, if we use a context window of size 2, there would be two words on each side of the target word (4 context words total).
    - These context words are represented as **one-hot encoded vectors** in a vocabulary-size dimensional space.
2. **Hidden Layer**:
    - The hidden layer has **no activation function** (it's linear).
    - The layer reduces the dimensionality of each word vector to a smaller dimension, which is essentially the size of the embedding vector we want for each word.

3. **Projection/Embedding Matrix**:

   - Each word in the vocabulary is associated with a dense vector (embedding) that is learned by the model.

   - The CBOW model uses a shared embedding matrix for all input words.

   - When a context word is input, its one-hot vector representation is multiplied by this embedding matrix to get a dense representation.

4. **Averaging**:

   - The embeddings of the context words are **averaged** to get a single context vector, which represents the collective meaning of the surrounding words.

5. **Output Layer**:

   - The output layer is a **softmax** layer that computes probabilities for each word in the vocabulary.

   - The word with the highest probability is the model's prediction for the target word.

6. **Loss Function**:

   - CBOW uses the **cross-entropy loss** to calculate the difference between the predicted word probabilities and the actual target word.

   - During training, this loss is minimized, updating the embedding vectors to better predict target words based on context.

Important Terms

- **Context Words**: The words surrounding the target word. In CBOW, these words are used to predict the target word.

- **Target Word**: The central word that the model tries to predict.

- **Window Size**: Defines the number of context words considered on either side of the target word. A window size of 2 means two context words before and two after the target word.

- **Word Embedding**: A dense vector representation of a word that captures its meaning based on context. Embeddings are learned during the training process.

- **One-Hot Encoding**: A binary representation of words, where each word is represented by a vector of 0s and 1s, with a unique position set to 1 for each word.

- **Embedding Matrix**: A matrix where each row represents the dense vector of a word in the vocabulary. It is initialized randomly and learned during training.

- **Softmax Function**: Used in the output layer to convert raw scores into probabilities, selecting the target word based on maximum likelihood.
- **Cross-Entropy Loss**: The loss function used to evaluate the difference between the predicted word probability distribution and the actual target word.

This code effectively preprocesses text data, creates a word cloud, generates context-target pairs for CBOW training, and initializes a random embedding matrix. Key terms and steps to note:

- **Stopwords**: Frequently used words removed to focus on meaningful terms.
- **Regular Expressions (Regex)**: Used to clean and standardize text.
- **Vocabulary**: The set of unique words in the text.
- **One-Hot Encoding** (conceptual, though not used here): Mapping words to indices.
- **Word Embedding**: Dense vector representation initialized randomly here, but optimized during model training.