Research Report on CBOW (Continuous Bag of Words)

It is a neural network architecture used in natural language processing (NLP) to predict a target word based on its surrounding context words. It belongs to the family of word embedding techniques and is popularly associated with models like Word2Vec.

**General Architecture**

Input (word indices) --> Embedding Layer --> Lambda Layer --> Dense Output Layer (Softmax)

**Corpus Processing**

Normalization, Tokenization, Filtering

**Context and Target Pair Generation**

CBOW operates by predicting a target word based on its surrounding context words. For each word in the corpus:

Define a context window (ex: 4) around the target word to capture neighboring words.

Treat each word in the context window as a target word and its surrounding words as context words to form pairs.

Initialize word embeddings with random weights, typically in the shape of the embedding size.(ex: 100)

**In-Depth CBOW Algorithm(**  here 21 is the vocab size)

**Context Embedding**

The Context\_emb (shape: 1×100) in CBOW represents the average of embeddings for context words surrounding a target word. For example, if we consider a context window size of 4 and an embedding dimension of 100, Context\_emb would be computed by averaging the embeddings of these 4 context words.

**Logits Calculation**

The Logits (shape: 1×21) are computed as the dot product of Context\_emb with the word embeddings matrix (W\_emb). Each element in Logits corresponds to the raw score or unnormalized probability of each word in the vocabulary being the target word, given the context. The word with the highest logit score is considered the most likely target word.

**Softmax Activation**

To convert logits into probabilities, the softmax function is applied:

A black background with white text

Description automatically generated

Here, zj represents the logits for the word j, and V is the vocabulary size. Softmax normalization ensures that the resulting probabilities indicate the likelihood of each word in the vocabulary being the target word given the context.

**Target Identification**

The target word is identified using the index of the one-hot encoded vector, where the correct target word is represented as a vector with a single '1' and '0's elsewhere.

**Loss Calculation**

The loss is calculated using the negative log-likelihood of the predicted probability for the target word. This penalizes the model more when it predicts low probabilities for the correct target word.

**Gradient Calculation**

During backpropagation, gradients are calculated to update the weights (W\_emb). Specifically:

* dlogits[0, target] -= 1 adjust the gradient to reflect the error in predicting the correct target word.
* np.dot(dlogits.T, context\_emb) calculates the gradient of W\_emb based on how changes in logits affect the loss, propagated back through the network.

**Word Similarity and Euclidean Distances**

- After training, utilize Euclidean distances to measure similarities between word embeddings. Lower distances indicate closer semantic relationships between words.

A black background with white symbols

Description automatically generated

Where wi and wj are embeddings of words i and j, and D is the embedding dimensionality.

***revisit***

**1. Training the model with 1000+ words**

**2. Skipgram implementation and working**