Experiment No: 05

Aim: Implement the continuous Bag of Words (CBOW) model for text recognition in a given dataset.

Problem Statement:

Implement the Continuous Bag Of Words (CBOW) Model. Task to build the model are.

a. Data preparation

b. Generate training data

c. Train model

d. Output

Objective:

- BuildCBOW model to predict the current word given context words within a specific window.
- Evaluate Model.

Theory:

Word Embedding is a language modeling technique used for mapping words to vectors of real numbers. Word embedding can be generated using various methods like neural networks, co-occurrence matrix, probabilistic models, etc. The basic idea of word embedding is words that occur in similar context tend to be closer to each other in vector space.Word2Vec consists of models for generating word embedding.

These models are shallow two-layer neural networks having one input layer, one hidden layer, and one output layer. Word2Vec utilizes two architectures:

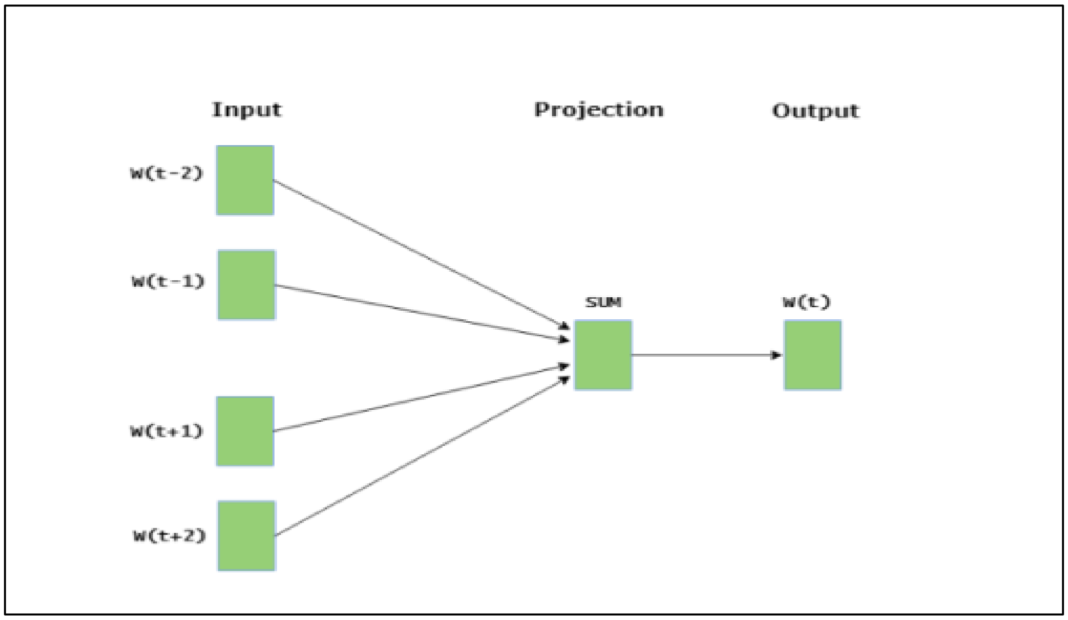
1. CBOW (Continuous Bag of Words)

2. Skip Gram

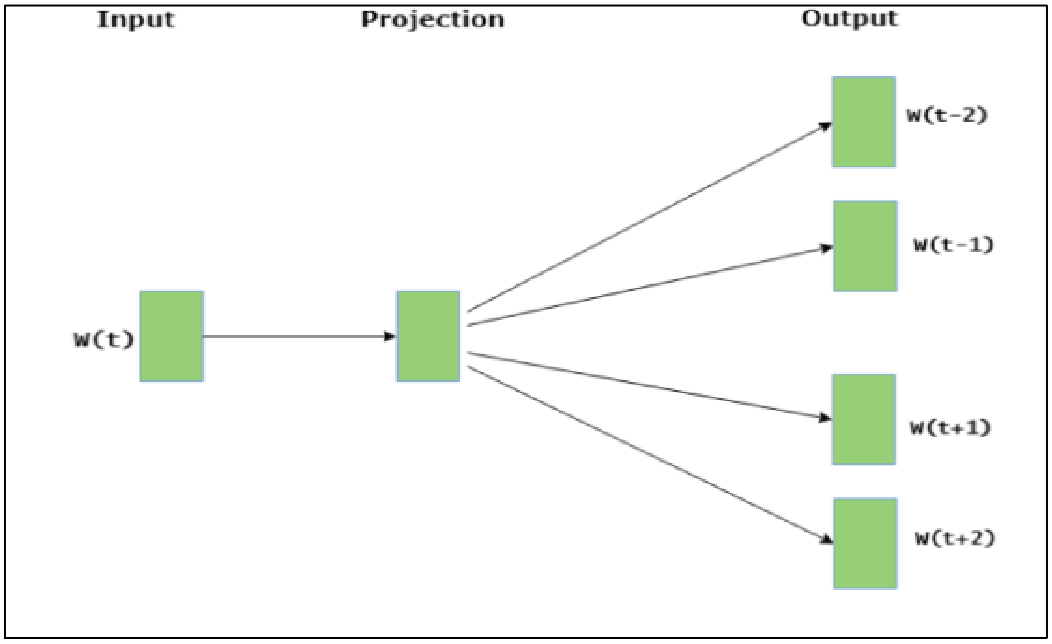
CBOW (Continuous Bag of Words): The CBOW model tries to understand the context of the words and takes this as input. It then tries to predict words that are contextually accurate.

- CBOW model predicts the current word given context words within a specific window.

- The input layer contains the context words and the output layer contains the current word.
- The hidden layer contains the number of dimensions in which we want to represent the current word present at the output layer.



- Skip Gram : Skip gram predicts the surrounding context words within a specific window given the current word. The input layer contains the current word and the output layer contains the context words. The hidden layer contains the number of dimensions in which we want to represent the current word present at the input layer.



The basic idea of word embedding is words that occur in similar context tend to be closer to each other invector space. For generating word vectors in Python, modules needed are nltk and gensim. Run these commands in terminal to install nltk and gensim:

- pip install nltk
- pip install gensim

Following are the steps to implement the CBOW Model,

Step 1:Download or collect the dataset from https://www.gutenberg.org/files/11/11-0.txt

Step 2: Remove all special characters and digits

Step 3: Remove all punctuation marks

Step 4: Bring all letters to lower case

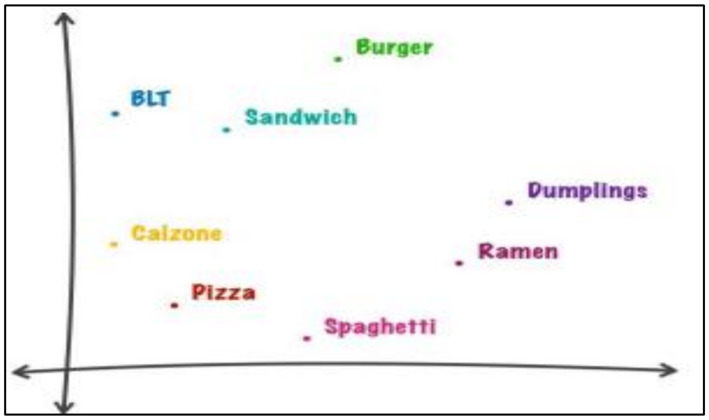
Step 5: Tokenize the document into sentences and words

Step 6: Remove all stop words.

Step 7: Train the model (Use Gensim word2vec for training the model)

To make words understood by machine learning algorithms, word embedding is used to map words into vectors of real numbers. There are various word embedding models and word2vec is one of them.In simple words, word2vec is a group of related models that are used to produce word embedding. These models are trained to construct the linguistic contexts of words. Word2vec takes a large corpus of text and produces a

vector space, with each unique word in the corpus being assigned to a corresponding vector in the space.



To understand how word2vec works, let's explore some methods that we can use with word2vec such as finding similarities between words or solving analogies like this

$$\boxed{f(\text{"canada"}) - f(\text{"us"}) = f(\text{"??"}) - f(\text{"hamburger"})}$$

Step 8: Load the dataset and Model

Step 9: Find the most similar Words Now we use model.most_similar() to find the top-N most similar words. Positive words contribute positively towards the similarity, negative words negatively. This method computes cosine similarity between a simple mean of the projection weight vectors of the given words and the vectors for each word in the model. The method corresponds to the word-analogy and distance scripts in the original word2vec implementation. If topn is False, most_similar returns the vector of similarity scores. restrict_vocab is an optional integer which limits the range of vectors which are searched for most-similar values. For example, restrict_vocab=10000 would only check the first 10000 words vectors in the vocabulary order. (This may be meaningful if you've
sorted the vocabulary by descending frequency.)

Step 10: Predict the output word [Use predict_output_word(context_word_list,topn=10) function]

Conclusion:

Thus we have implemented the CBOW model on collected dataset and found that word embedding approach is very useful for text recognition. CBOW can also be apply to convert the speech to text and has many more application in natural language processing.