1. Explain One-Hot Encoding

One hot encoding is a process by which categorical variables are converted into a form that could be provided to ML algorithms to do a better job in prediction.

1. Explain Bag of Words

The Bag of Words model is a popular one that uses word frequency or occurrences to train a classifier. This methodology generates a matrix of occurrences for documents or phrases, regardless of their grammatical structure or word order. A bag-of-words is a text representation that describes the frequency with which words appear in a document. It entails two steps: A list of terms that are well-known. A metric for determining the existence of well-known terms. Because any information about the sequence or structure of words in the document is deleted, it is referred to as a "bag" of words. The model simply cares about whether or not recognised terms appear in the document, not where they appear

1. Explain Bag of N-Grams

Text N-grams are commonly used in text mining and natural language processing. They're essentially a collection of co-occurring words within a specific frame, and when computing the n-grams, you usually advance one word although you can move X words forward in more advanced scenarios.

1. Explain TF-IDF

TF-IDF also called Term Frequency-Inverse Document Frequency helps us get the importance of a particular word relative to other words in the corpus. It's a common scoring metric in information retrieval (IR) and summarization. TF-IDF converts words into vectors and adds semantic information, resulting in weighted unusual words that may be utilised in a variety of NLP applications.

1. What is OOV problem?

In Natural Language Processing (NLP), the OOV (Out-of-Vocabulary) problem refers to the challenge of encountering words or tokens during text processing that are not present in the vocabulary or lexicon of the language model or NLP system.

When training NLP models, a vocabulary is typically created by compiling a set of known words from a large corpus of text. This vocabulary serves as a reference for the model to understand and generate text. However, in real-world scenarios, there will inevitably be words or tokens that are not included in the vocabulary. These could be newly coined terms, domain-specific jargon, rare words, or misspellings.

1. What are word embeddings?

Word embeddings are a representation of words or phrases in a numerical form that captures the semantic relationships and contextual meaning between words. They are a key component in Natural Language Processing (NLP) tasks such as language modeling, text classification, machine translation, and sentiment analysis. Traditionally, NLP models have used one-hot encoding to represent words, where each word is represented by a sparse binary vector with a dimension equal to the size of the vocabulary. However, this representation fails to capture the meaning and relationships between words. Word embeddings, on the other hand, aim to represent words in a continuous vector space where similar words are closer together and dissimilar words are farther apart. These embeddings are typically generated through unsupervised learning algorithms, such as Word2Vec, GloVe, or FastText, which learn representations based on the distributional properties of words in large corpora

1. Explain Continuous bag of words (CBOW)

Continuous Bag of Words (CBOW) is a model architecture used for generating word embeddings in Natural Language Processing (NLP). It aims to predict a target word based on its surrounding context words within a given window size.

In CBOW, the training process involves two main steps:

Building the Training Data: The training data for CBOW consists of input-context pairs. Given a sentence or a sequence of words, the model creates training samples by taking a target word and its surrounding context words within a fixed window size. For example, consider the sentence: "The cat sat on the mat." If the window size is 2, the training pairs could be:

Input: ["The", "cat", "on", "the"], Target: "sat"

Input: ["cat", "sat", "the", "mat"], Target: "on"

Training the CBOW Model: The CBOW model takes the context words as input and aims to predict the target word. It consists of an embedding layer, a hidden layer, and an output layer.

a. Embedding Layer: The input context words are converted into their respective word embeddings. Each word in the context is represented by a dense vector, typically of fixed length, which captures the semantic meaning of the word.

b. Hidden Layer: The embeddings of the context words are averaged together to create a single vector representation. This vector is passed through a hidden layer with activation functions such as ReLU or sigmoid, which captures the non-linear relationships between the input and output.

c. Output Layer: The output layer is a softmax layer that produces a probability distribution over the entire vocabulary. The probabilities indicate the likelihood of each word being the target word given the context. The model is trained to maximize the probability of the correct target word.

During training, the model's parameters (weights) are adjusted using optimization algorithms like stochastic gradient descent (SGD) or Adam. The objective is to minimize the loss between the predicted probabilities and the actual target word.

Once the CBOW model is trained, the weights in the embedding layer represent the word embeddings. These embeddings can be used in downstream NLP tasks, where they capture semantic relationships and contextual meaning between words.

CBOW is often compared to another popular word embedding model called Skip-gram, where the objective is to predict context words given a target word. CBOW is generally faster to train and performs well on syntactic tasks and capturing word similarity, while Skip-gram is better at capturing rare words and semantic relationships.

1. Explain SkipGram

Skip-gram is a model architecture used for generating word embeddings in Natural Language Processing (NLP). It aims to predict the context words given a target word, in contrast to the Continuous Bag of Words (CBOW) model which predicts the target word given the context.

The training process of Skip-gram involves two main steps:

1. Building the Training Data: Similar to CBOW, the training data for Skip-gram consists of input-context pairs. Given a sentence or a sequence of words, the model creates training samples by selecting a target word and a set of context words within a fixed window size. For example, consider the sentence: "The cat sat on the mat." If the window size is 2, the training pairs could be:

Input: "sat", Context: ["The", "cat", "on", "the"]

Input: "on", Context: ["cat", "sat", "the", "mat"]

2.Training the Skip-gram Model: The Skip-gram model takes the target word as input and aims to predict the context words. It consists of an embedding layer, a hidden layer, and an output layer.

a. Embedding Layer: The target word is converted into its word embedding, which is a dense vector representation capturing its semantic meaning.

b. Hidden Layer: The word embedding is passed through a hidden layer with activation functions such as ReLU or sigmoid. This layer captures the non-linear relationships between the input and output.

c. Output Layer: The output layer is a softmax layer that produces a probability distribution over the vocabulary. Each word in the vocabulary is assigned a probability representing the likelihood of it being a context word given the target word. The model is trained to maximize the probability of the correct context words.

During training, the model's parameters (weights) are adjusted using optimization algorithms like stochastic gradient descent (SGD) or Adam. The objective is to minimize the loss between the predicted probabilities and the actual context words.

Once the Skip-gram model is trained, the weights in the embedding layer represent the word embeddings. These embeddings capture semantic relationships and contextual meaning between words. They can be used in various downstream NLP tasks such as text classification, sentiment analysis, and machine translation.

Skip-gram is known for performing well on capturing semantic relationships and representing rare words effectively. It is particularly useful in scenarios where capturing the meaning of less frequent words or dealing with large vocabularies is important. However, it may require more training time compared to CBOW due to the higher complexity of predicting multiple context words from a single target word.

1. Explain Glove Embeddings.

GloVe (Global Vectors for Word Representation) is a word embedding technique commonly used in Natural Language Processing (NLP) tasks. It aims to generate word embeddings by leveraging global co-occurrence statistics of words in a large corpus of text.

The key idea behind GloVe is to capture the semantic relationships between words based on their co-occurrence probabilities. It combines the advantages of global matrix factorization methods and local context window-based methods.

The process of generating GloVe embeddings involves the following steps:

1. Co-occurrence Matrix:

First, a co-occurrence matrix is constructed based on word-word co-occurrence statistics. The matrix represents the frequency of word pairs occurring together within a specific context window in the corpus. The size of the context window determines the extent of the word's surrounding context considered for co-occurrence.

2. Probability Distribution:

From the co-occurrence matrix, a probability distribution is derived by normalizing the co-occurrence counts. This distribution represents the probability of observing a particular word given another word in its context.

3.Objective Function:

GloVe introduces an objective function that captures the relationship between word vectors based on their co-occurrence probabilities. The goal is to learn word embeddings that satisfy the following equation: word\_vector\_i \* word\_vector\_j = log(co\_occurrence\_probability\_ij)

The objective function is designed to minimize the difference between the dot product of word vectors and the logarithm of the corresponding co-occurrence probabilities.

4.Training:

The GloVe model is trained by optimizing the objective function using techniques such as gradient descent. The training process adjusts the word vectors iteratively to minimize the loss between the predicted dot products and the logarithm of the co-occurrence probabilities.

5.Obtaining GloVe Embeddings:

After training, the word vectors in the GloVe model represent the embeddings. These embeddings capture the semantic relationships between words based on their co-occurrence statistics. They tend to reflect both global semantic relationships and local context information.