Word2Vec – A Summary

Word2Vec is a natural language processing technique that creates word embeddings. Each word in a given corpus is represented as a vector (= word embedding) so that relationships between the vectors can be established. Word2Vec is a technique for learning word meanings by capturing semantic and syntactic relationships of words. The skip-gram model uses a neural network architecture consisting of three layers. With positive and negative samples, the model is trained to capture meaningful information of words. Additionally, the skip-gram model applies dimensionality reduction while still preserving the meaningful properties in the vectors. Overall, word2vec revolutionized the field of natural language processing by providing an efficient way for computers to understand the meaning of words. This summary will go through the key steps that Word2Vec takes to learn word vectors with a plane neural network and will elaborate all the important aspects that is needed to understand Word2Vec:

There are two algorithms in the software package Word2Vec, skip-gram and Continuous Bag of Words (CBOW), which leverage the context of the target words. Both variants utilize the surrounding words to represent the target word. Skip-gram predicts context words for a target word given as an input. Context words surround a target word in a text whereas a target word is the word that we want to learn the word embedding for. It is important to mention here that every word can sometimes be a context word and sometimes a target word. CBOW on the other hand, given a context, will predict which word is most likely to appear. In this summary, we will focus on the skip-gram model and its internal representations of words within the model's 3 layers: the input, the hidden and the output layer.

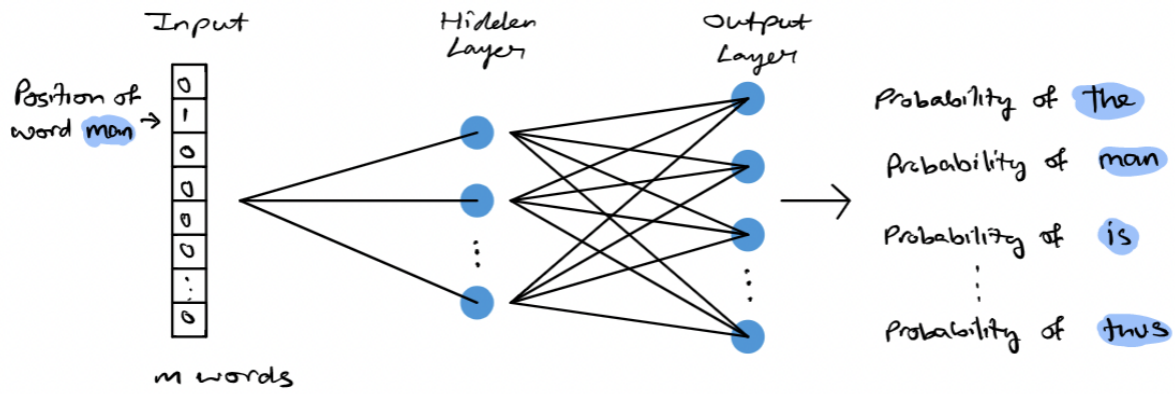

Figure 1: The skip-gram model (towardsdatascience.com)

In a skip-gram model, every word has two kinds of representations, a target and a context representation. The target representation is learned from the input to the hidden layer and the context representation is learned from the hidden layer to the output layer. The target representation captures the semantic meaning of a word and the context representation captures the contextual meaning of a given word. The output layer contains the vector representations of the potential target words.
The input layer of the skip-gram takes a one-hot encoding of the target word. The one-hot encoding is a binary vector where each word of the vocabulary is assigned to a unique index.
Next, the hidden layer holds the word representation of the target word, which was learned during the training of the model. During the training process the weights which are in the hidden layer are adjusted during the training to capture semantic relationships between the input and the desired output (context words). Words that either appear in the same context or have similar meanings usually have similar vector representations. During the training process, positive and negative samples are used to update the weights in the hidden layer. Weights in the hidden layer combine the input features to produce meaningful representations. Positive samples are real word pairs which means that a target word and a context word appear in the same window of text. A negative sample consists of a target word and context word that were randomly selected and are not in the same window of text. The goal of the model is to maximize the probability of the positive training samples and minimize the probability of negative

samples during training. The process of dimensionality reduction transforms data from high-dimensional feature space to a low-dimensional feature space. It is important that meaningful properties present in the input data are not lost during the transformation in the hidden layer. The output layer contains a probability distribution of the context words. Each word represents the probability of a specific context word given the target word.

To gain deeper understanding of how models works, one must know of some key differences of interpretable and uninterpretable word vectors: Firstly, interpretable word vectors are easily understood by humans such as one-hot encoded vectors and allow for transparency. Uninterpretable word vectors on the other side are not readable by 'the naked eye' as their properties are not immediately understandable. But uninterpretable word vectors may allow better performance.