**A single word with the same spelling and pronunciation (homonyms) can be used in multiple contexts and a potential solution to the above problem is computing word representations.**

Word2Vec

The main idea behind it is that you train a model on the context on each word, so similar words will have similar numerical representations.

Just like a normal feed-forward densely connected neural network(NN) where you have a set of independent variables and a target dependent variable that you are trying to predict, you first break your sentence into words(tokenize) and create a number of pairs of words, depending on the window size. So one of the combination could be a pair of words such as ('cat','purr'), where cat is the independent variable(X) and 'purr' is the target dependent variable(Y) we are aiming to predict.

We feed the 'cat' into the NN through an embedding layer initialized with random weights, and pass it through the softmax layer with ultimate aim of predicting 'purr'. The optimization method such as SGD minimize the loss function "(target word | context words)" which seeks to minimize the loss of predicting the target words given the context words. If we do this with enough epochs, the weights in the embedding layer would eventually represent the vocabulary of word vectors, which is the "coordinates" of the words in this geometric vector space.

The above example assumes the skip-gram model. For the Continuous bag of words(CBOW), we would basically be predicting a word given the context.

GLOVE

GLOVE works similarly as Word2Vec. While you can see above that Word2Vec is a "predictive" model that predicts context given word, GLOVE learns by constructing a co-occurrence matrix (words X context) that basically count how frequently a word appears in a context. Since it's going to be a gigantic matrix, we factorize this matrix to achieve a lower-dimension representation. There's a lot of details that goes in GLOVE but that's the rough idea.

FastText

<https://www.analyticsvidhya.com/blog/2017/07/word-representations-text-classification-using-fasttext-nlp-facebook/>

Develped by facebook( word embedding and sentence classification)

FastText is quite different from the above 2 embeddings. While Word2Vec and GLOVE treats each word as the smallest unit to train on, FastText uses n-gram characters as the smallest unit. For example, the word vector ,"apple", could be broken down into separate word vectors units as "ap","app","ple". The biggest benefit of using FastText is that it generate better word embeddings for rare words, or even words not seen during training because the n-gram character vectors are shared with other words. This is something that Word2Vec and GLOVE cannot achieve.

https://medium.com/huggingface/universal-word-sentence-embeddings-ce48ddc8fc3a