Word 2 vec tutorial

<http://adventuresinmachinelearning.com/word2vec-keras-tutorial/>

Basic idea:

Using skip-gram, for a target word, tries to predict the surrounding context words

Vocabulary size = number of words

Embedding size = number of weights for the hidden layer

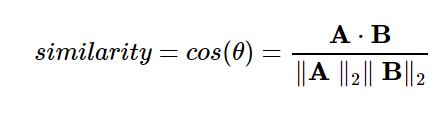
Softmax is really expensive 🡪 use negative sampling

Negative sampling:

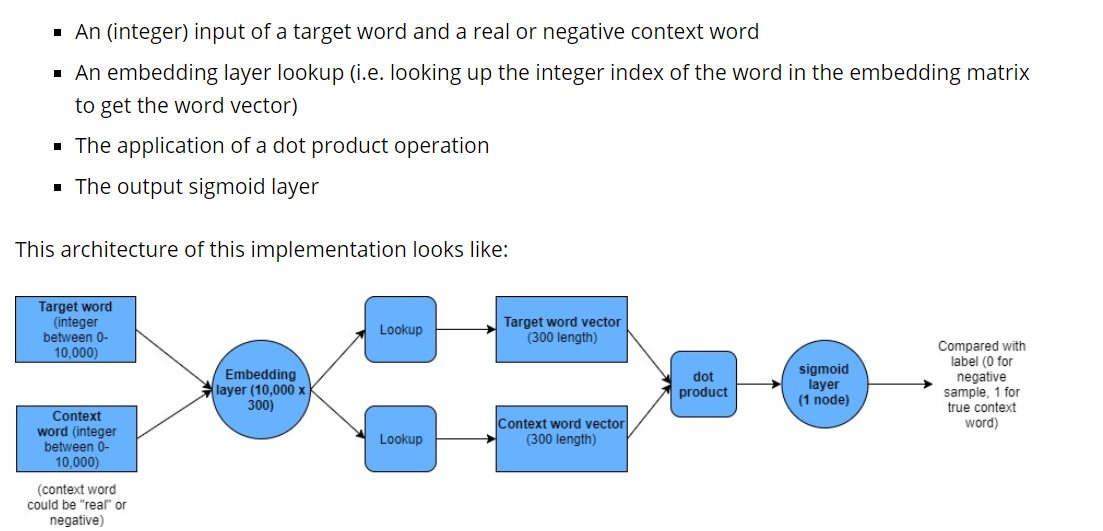
It works by reinforcing the strength of weights which link a target word to its context words, but rather than reducing the value of all those weights which aren’t in the context, it simply samples a small number of them – these are called the “negative samples”.

Embedded layer = hidden layer

We also need a way of ensuring that, as the network trains, words which are similar end up having similar embedding vectors. Therefore, we need a vector similarity score supplied to the output sigmoid layer – with similar vectors outputting a high score and un-similar vectors outputting a low score. The most typical similarity measure used between two vectors is the cosine similarity score:



For training the embedded layer:



Let’s go through this architecture more carefully. First, each of the words in our vocabulary is assigned an integer index between 0 and the size of our vocabulary (in this case, 10,000). We pass two words into the network, one the target word and the other either a word from the surrounding context or a negative sample. We “look up” these indexes as the rows of our embedding layer (10,000 x 300 weight tensor) to retrieve our 300 length word vectors. We then perform a dot product operation between these vectors to get the similarity. Finally, we output the similarity to a sigmoid layer to give us a 1 or 0 indicator which we can match with the label given to the Context word (1 for a true context word, 0 for a negative sample).

The back-propagation of our errors will work to update the embedding layer to ensure that words which are truly similar to each other (i.e. share contexts) have vectors such that they return high similarity scores. Let’s now implement this architecture in Keras and we can test whether this turns out to be the case.