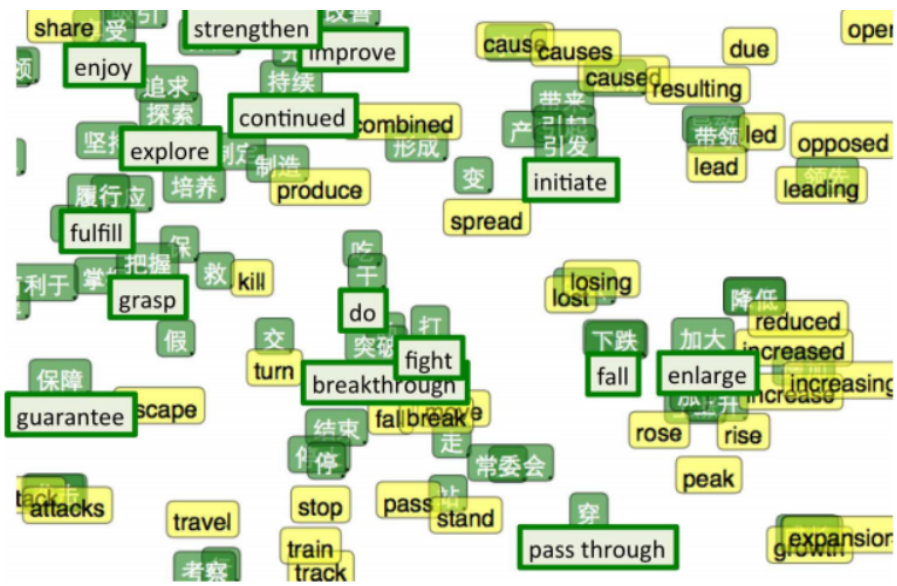
Word2Vec Implementation Issues

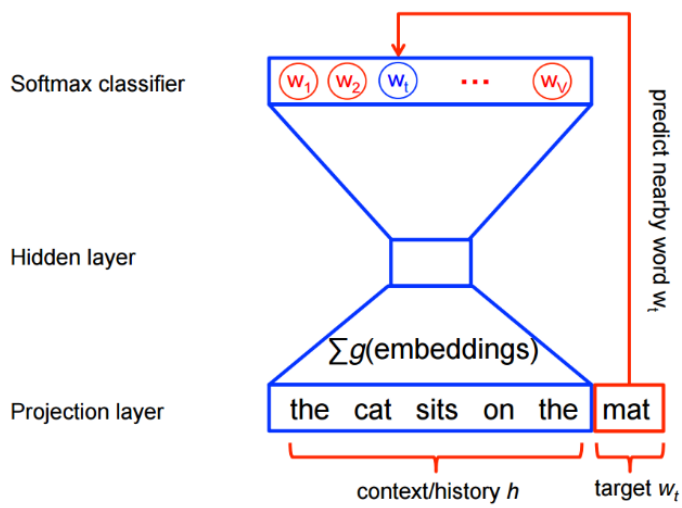
Word2vec algorithm was created in Google in 2013. This model takes in a corpus of text as input and gives out a vector space as output. Each word vector in this space represents a word in the corpus. Words with similar context end up as neighbors in the vector space. This model is extensively used in text classification and chatbot implementation. For instance, consider king—man + woman = queen. The relationship between the words is dependent on the distance between the words. The above algebraic equation shows us that vector("King") – vector("Man") + vector("Woman") result was closest to the vector representation of the word "Queen". So, man is to woman as king is to queen. Word2vec can also be used to embed Mandarin Chinese words and English words in the same space.



The figure above the t-SNE visualization of bilingual word embedding

The following are the implementation issues associated with word2vec:

- Word2vec has a high memory complexity. This is because the quickest way to construct a term co-occurrence matrix (tcm) is to store it in RAM as a hash map.
- One of the biggest drawbacks of the word2vec model is its incapacity to deal with unknown or out-of-vocabulary (OOV) words. If our model has not encountered a word before, it will not know how to interpret the word or build a vector for it.
- This model is also sensitive in the presence of noisy and sparse data. For instance, in a social media domain, such as Twitter, there are a lot of words that may be used only once or twice in a large corpus, and you are forced to use a random vector.

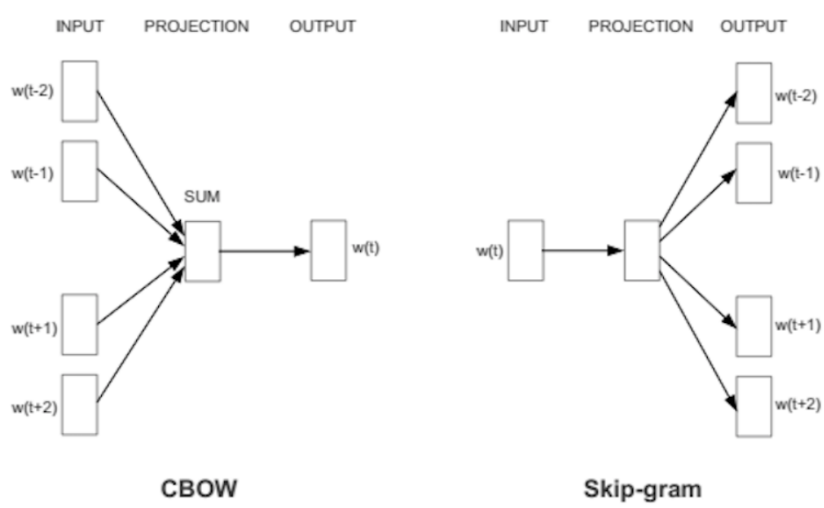


The figure above depicts how the word2vec model works using the softmax function

- Word2vec does not contain shared representations at sub-word levels. Word2vec represents each word as an independent vector, even if there exist words that are morphologically similar. For instance, if there are two words ending in 'ful', we know that it is probably an adjective since it is similar to 'beautiful', 'harmful'

and so on, but word2vec, as stated above, represents each word as an independent vector.

- Cross-lingual use of the same model is not possible with word2vec. Scaling to new languages requires new embedding matrices and does not enable parameter sharing.
- Word2vec model cannot be used to initialize state-of-art architectures that take character sequences as input. If we have a model that takes character-based input, you cannot benefit from pre-training as you are forced to randomize embeddings.
- This model is insensitive to word order. The skip-gram and the continuous bag-of-words CBOW model are used to implement the word2vec method, and both these methods discard word order information while accounting for context (Ling, Dyer, Black, Trancoso, 2015).



The figure above shows the two ways to implement word2vec model: Skip-gram (using word to predict a target context) and CBOW (using context to predict a target word)

- Word2vec model is not useful when dealing with large corpuses.

When we need to compute a single forward pass of our model, we need to sum through the entire corpus vocabulary to evaluate the softmax function. This is very time consuming in the presence of large datasets.

- For problems where semantic relatedness is important, word2vec is not the ideal method to use (Ling, Dyer, Black, Trancoso, 2015). For larger datasets, this model performs better, but the precision decreases.
- Word sense is not captured separately in the word2vec model. For instance, 'left' could mean to leave and a direction, but they are still represented in a single vector (Mikolov, Chen, Corrado, Dean).

Ling, W. Dyer, C. Black, A. Trancoso, I. Two/ Too Simple Adaptations of Word2Vec for Syntax Problems. 2015. Retrieved from: http://www.cs.cmu.edu

Mikolov, T. Chen, K. Corrado, G. Dean, J. Efficient Estimation of Word Representations in Vector Space. Retrieved from: https://arxiv.org