Word2Vec

Contents

What is Word2Vec?

Word2Vec is one of the word embedding techniques in the Natural Language Processing. It was developed by Tomas Mikolov and his team [Mikolov et al., 2013] at Google. Word2Vec is a shallow, two-layer neural network applied to text by vectorizing its words. The input of Word2Vec is a text corpus [**must be in the English language**] and the output is a set of vectors, typically of several hundred dimensions.

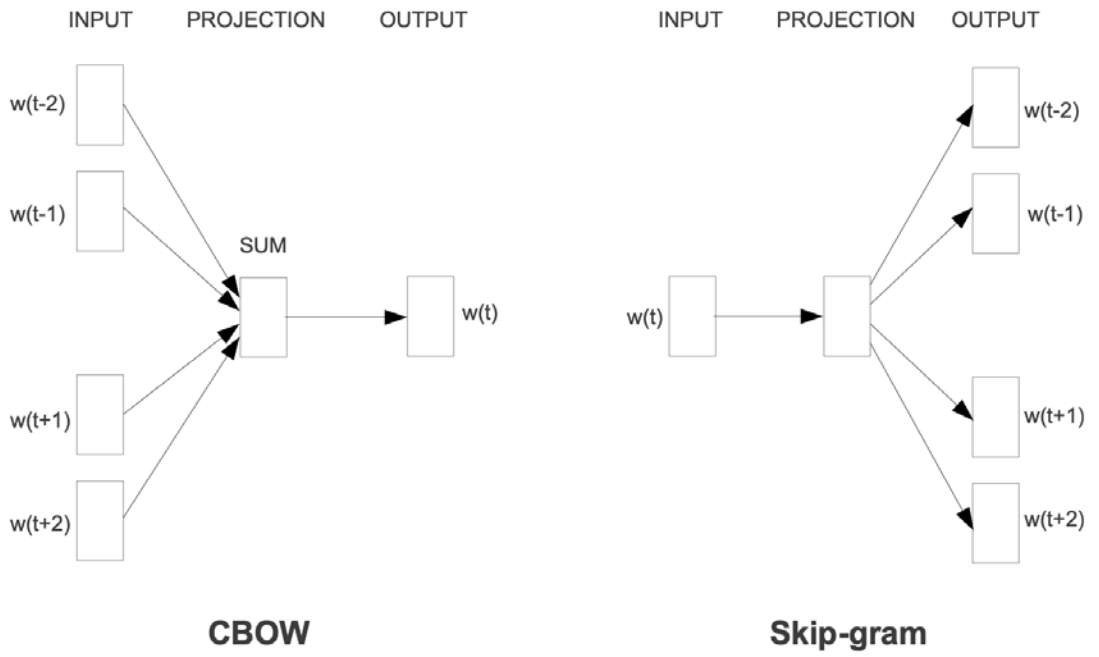
Word2Vec turns text into a numerical form, the vectors of similar words clustered together in vector space. The vectors used to represent words are called **neural word embeddings**. So, a neural word embedding represents a word with numbers. By vectorizing words, Word2Vec makes natural language computer-readable; on vectors we can perform powerful mathematical operations in order to detect similarities among words mathematically, **without human intervention**. Word vectors will place similar words close to each other in space. Thus, the words cat, dog, and chicken would most likely cluster in one corner, while car, road, and toll cluster in another.

Similar things and ideas are shown to be “close.” Their relative meanings have been translated to measurable distances. Qualities become quantities, and algorithms can do their work. But similarity is just the basis of many association that Word2Vec can learn. For example, it can gauge relations between words of one language, and map them to another.

Given enough data, usage, and contexts, Word2Vec can make highly accurate guesses about a word’s meaning based on past occurrences. Those guesses can be used to establish a word’s association with other words (e.g. “man” is to “boy” and “woman” is to “girl”), or cluster documents and classify them by topic. Those clusters can form the basis of search, sentiment

analysis, and recommendations in such diverse fields as scientific research, legal discovery, e-commerce, and customer relationship management.

Two model architectures



There are two architectures of training Word2Vec: CBOW (Continuous Bag-of-Words) and Skip-gram. Both are to learn the underlying word representation of each word by using neural networks. In the CBOW model, the continuous distributed representations of context (or surrounding words) are combined to predict the target word. While in the Skip-gram model, instead of predicting the target word based on the context, the word is used to predict the context (certain rage begore and after). Generally, the CBOW is much more faster and slightly better accuracy for the frequent words. However, the Skip-gram produces more accurate results on large datasets, and it also works well with a small amount of the training data, represents well even rare words or phrases.

Parameters

There are dozen parameters to train Word2Vec, and the NLP Suite makes user to set up four parameters below:

- **learning model architecture**:
  the two architectures for training model – CBOW and Skip-gram
- **vector size**:
  the dimensionality of the resulting word vectors; if you have a large corpus (> few billion tokens) you can go up to 100-300 dimensions. Generally word vectors with more dimensions give better result. (default: 100)

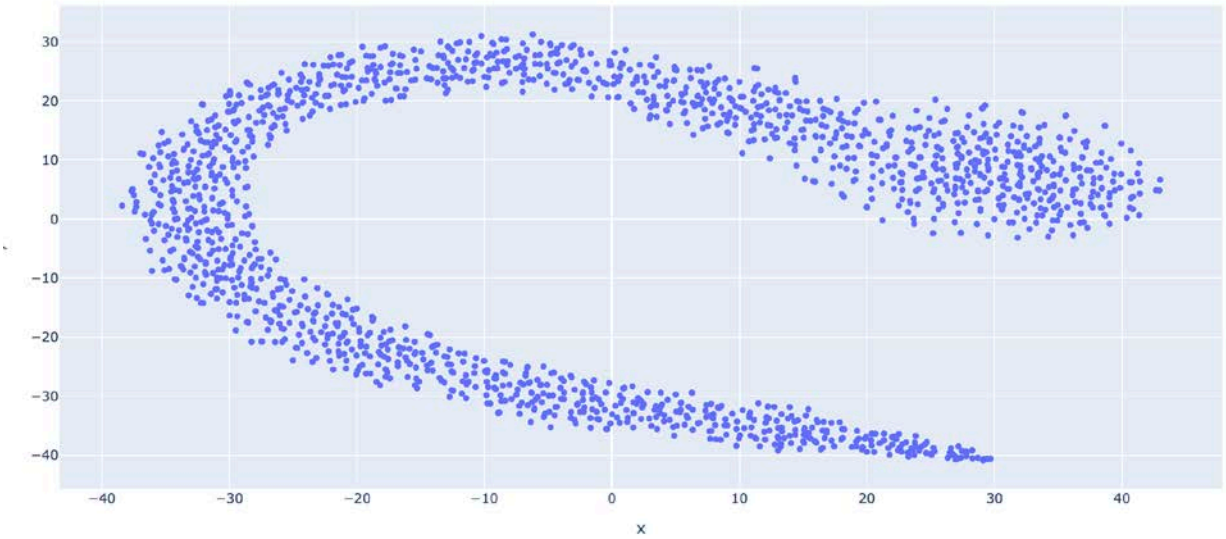
- **window size**:
  maximum distance between the current and predicted word within a sentence; in other words, how many words come before and after your given word. It vary depending on what you are interested in. For instance, if you are more interested in embeddings that embody semantic meaning, smaller window sizes work better (default: 5).
- **minimum count**:
  ignores all words with total frequency lower than this; if all the frequencies of each word are lower than this, it occurs error (default: 5).

Result

Word2Vec in the NLP Suite will create one csv file and one html file. The csv file is containing 7 columns: a word found in the corpus (column 1), the lemmatized word (column 2) , the resulting vector for the word (column 3), the sentence ID of the word (column 4), the sentence where the word appears (column 5), the document ID of the word (column 6), and the document where the word appears (column 6). The resulting vector of each word are multi-dimension, so the NLP Suite will reduce them in order to represent them in the two dimensional space.

Visualization

The resulting vectors for each word will be displayed in the two dimensional Cartesian space in the html file. You can close up by selecting certain space. Look carefully to see whether the spatial distribution of words suggests connections you had not thought about.



References

Mikolov, Tomas, Kai Chen, Greg Corrado, Jeffrey Dean. 2013. *Efficient Estimation of Word Representations in Vector Space*. CoRR. abs/1301.378

Gensim API Reference: Word2vec embeddings,
https://radimrehurek.com/gensim/models/word2vec.html