Word2Vec

Son Tran, Khoi Le, Uyen Le

January 23rd, 2022

1 Inner Workings of Word2Vec

The word2vec models contains 2 main layers, namely the hidden layer and the output layer. Each of the two layers is a matrix whose vectors are exactly word representations of words in the corpus. The two layers' dimension are reversed, i.e. if the hidden layer's matrix has size of 10000×300, then the output layer's matrix has size of 300×10000. Although their sizes are similar, their purposes are not the same as the hidden layer tries to convert every word into a vector with predefined dimensionality, while the output layer tries to calculate the probability of picking words in the corpus to be nearby the chosen center word. The word2vec models' task is to optimize such that nearby words in a sentence should have high probability output, i.e the word embedding of nearby words are similar.

Let's talk about the two layers in details:

1. The hidden layer: this layer takes input as one-hot vectors which has the size of $1\times$ *size of corpus* containing all 0s, except a 1 corresponding with the index of a word in a corpus. Therefore, the hidden layer's matrix has size of *size of corpus* $\times$ *number of features* that tries to convert each word into a $1\times$ *number of features* vector.

2. The output layer: this layer takes the output vectors of the hidden layer. It performs matrix multiplication and then applies softmax function to return the probability of nearby words to appear around the center word.

The vectors in each layer's matrix are exactly word embeddings. However, the vectors representing the same word in two matrices are different. Yet, the two vectors in two matrices that represent two nearby words in two matrices are the same, which facilitates the model to achieve their ultimate goal of adjusting word vectors of nearby words to be similar.

2 Word2Vec Model Evaluation

The methods of evaluating a word2vec model can be varied, but we only use two main benchmarks, namely word similarity and word analogy.

1. Word similarity: This benchmark consider the similarity of words. For example, "cars" and "automobiles" should have high similarity score because they are related, while "noon" and "string" should have low score. The score are calculated by cosine distance between vectors, which can be expressed as $cos(\overrightarrow{v_1}, \overrightarrow{v_2}) = \frac{|\overrightarrow{v_1} \cdot \overrightarrow{v_2}|}{|\overrightarrow{v_1}||\overrightarrow{v_2}|}$. The higher the score means that the two vectors $\overrightarrow{v_1}$ and $\overrightarrow{v_2}$ are similar, which also implies that the words that the two vectors represent are closely related.

word1	word2	similarity
car	automobile	3.92
gem	jewel	3.84
journey	voyage	3.84
boy	lad	3.76
coast	shore	3.70
asylum	madhouse	3.61
magician	wizard	3.50
midday	noon	3.42
furnace	stove	3.11
food	fruit	3.08
bird	cock	3.05

Figure 1: Examples of word similarity test set

2. Word analogy: This benchmark deals with the connections between words. Specifically, given a pair of words that have connection with each other and a third word, the model will have to predict a fourth word based on the third one and the connection from the previous two. One famous example is given the pair "man"-"woman" and the word "king", the fourth word is mostly going to be "queen".

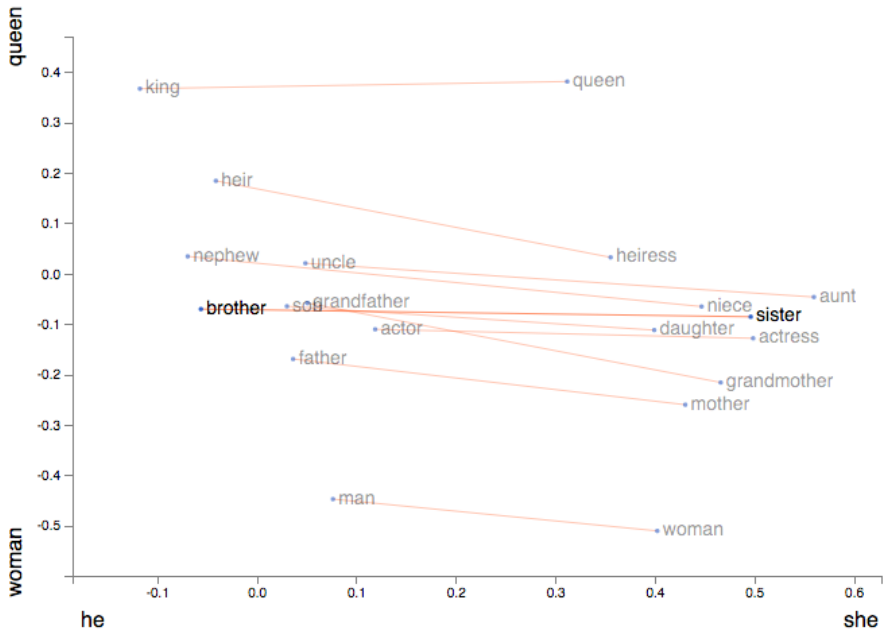

Figure 2: Examples of word analogy test set

3 Intuitions

3.1 Why does Word2Vec work?

(a) Evaluated on the Synonym Benchmark

The goal of Word2Vec is that when optimized, the model will try to synchronize word embeddings of nearby words in two different matrices. For example, the word vectors of words "I" in one matrix and that of "am" in the other matrix should be similar since "I" and "am" appear near each other most of the time. Given this similarity in vector representations of nearby words, it is supposed that words with similar contexts should have similar word embeddings. Hence, synonyms which have the same contexts should have similar word embeddings, which eventually indicates that they are indeed synonyms.

For example, given two sentences: "I am a good student." and "I am an excellent student.". The two words "good" and "excellent" have almost the same context, so the word embeddings of "good" and "excellent" in the same matrix are similar to those of their context words. This implies that "good" and "excellent" should have similar word vectors. Therefore, the model subconsciously knows that 'good" and "excellent" are word synonyms, and so Word2Vec works well on the Synonym Benchmark.

(b) Evaluated on the Word Analogy Benchmark (?)

3.2 Negative Sampling on Word2Vec

Difference between Word2Vec with and without Negative Sampling

(a) Why Negative Sampling?

A word2vec neural network has a tremendous number of weights, all of which would be updated slightly by only a small portion of our training samples. Particularly, for each time a center word appears in the corpus and enters the model, the words that are not context words of the center words also get updated yet only slightly. Eventually, computing all of this would takes too much time and resources. Moreover, the results from pairs of unrelated words also do not have a large impact on the overall loss function of the model.

One solution to this problem is Negative Sampling, where the model chooses 1 positive word and 5-20 negative words for each positive (center) word to update. Negative words are chosen on account of their frequency in the corpus, so more frequent words are more likely to be negative words. This strategy allows us to only modify a small percentage of the weights, rather than all of them for each training sample.

(b) Why implementation works?
The negative samplings model contains two embedding layers, one is for the input words, i.e. the center words, while the other one is for the output words, i.e. the context words. Each pair of "positive" and "negative" sample will be fed into the two layers and the output vectors will be multiplied to get a score.

The negative sampling model works because for each centered word, the model tries to update the layers so that the scores of "positive" samples are increased, while the scores of "negative" samples are decreased. This means that the model tries to distinguish between words that are closed to each other and words that are not related. In addition, the model only chooses a small amount of negative samples to update based on probability, so repeated updates will be minimized.

3.3 Why is Word2Vec no longer used?

Evaluated on Contextualized Embedding

(a) Compared to Glove

(b) Compared to Elmo

(c) Compared to BERT family

4 Experiments

4.1 Khoi Le's experiment:

The method of word2vec (Mikolov et al., 2013) has been an useful model to map words to vectors, which promotes the rapid development of modern Natural Language Processing in the past decades. The model of Skip-gram contains mainly 2 neural layers, the hidden layer and the output layer. The hidden layer takes the input as one-hot vector and outputs a certain row of that matrix. Then, the selected row will be fetched into the output matrix, which is a softmax regression classifier. However, the number of parameters used in word2vec can grow up to 75 million if the word embedding is 300-dimensional vector, which is comparable to the super model of current state-of-the-art language models. As reported in original research of Mikolov et al. (2013), 3 epoch of Skip-gram with 300-dimensional vector cost one CPU 3 days to train. In this research, we will try a simple approach to reduce the number of parameters used in word2vec while preserving the performances of word2vec.
Instead of reducing the number of dimensions of word embedding vectors to reduce the number of parameters used, we, in this research, experiment a new approach of using a small intermediate vector. In other words, our solution is to split up the hidden layer matrix into two matrices. For instance, if the original hidden layer is a matrix of size 10000×300, meaning that our vocabulary has 10000 words and the number of features is 300, then, we can split it into matrix A with size 10000×20 and matrix B with size 20×300. The reason for this method is that many words might have closely similar meaning, so we can reduce the number of features in the first matrix to map them to even closely related vectors and we can still map it to have 300 features. This approach might improve the performance as well as the efficiency of our models.

4.2 Uyen Le's Experiment:

Title: Reducing context words and Optimizing Use of Weighted Matrices from Different Language Representations

Machine translation models have traditionally been designed with left-to-right prediction order in mind. Additionally, the working of Word2Vec does not allow for parameter sharing across languages, making it complicated to scale to new languages. Thus, we design this work with a view to optimizing the originial word2vec method to fit the purpose of machine translation. We hope to achieve this via first limiting context words to just preceding words and then combining two weighted matrices from two different languages in negative sampling to preserve cross-lingual translation efficiency from a well trained source language.

First, we reduce the window size in a normal monolingual training (English) and check for its efficiency. Secondly, we test this out in a cross-lingual work. We do this by training the source language (Vietnamese), then extracting the weighted matrix, and stacking it upon the weighted matrix of the target language (English, assume that we have access to its learned representations) in

the negative sampling process. This produces a Vietnamese-English matrix that we can use to project output vectors in cross-lingual translation.

We evaluate the results using the same tests as Mikolov's. We approach this problem from a Viet-to-Eng perspective so that the results are in English, which makes it easier to assess the efficiency based on previous works. We also anticipate some difficulties while training Vietnamese word2vec since the datasets are not easily accessible, and Vietnamese itself contains different grammatical and semantic structures from other Western languages.