Note on Word Representation

Chao Yang
Microsoft
Suzhou
yangchao.42@outlook.com

Abstract

Word representation (especially the word2vec tools released by Mikolov) have drawn much attention recent years. In this note, I will give some essentials about word-context matrix, SGNS and Glove.

1 Introduction

The most basic word representation method is to use the word itself, just as what we use every day. A more economic method is to set a vocabulary list and use the word's index as its representation[1]. Although the later could convert the word into a useful mathematical form[2] , it catches no semantic information.

There is another set of methods which put each word into a continuous space in which the similar words will get close to each other. A common assumption is that the words having similar context will have the similar meaning, or close vector representation. Usually, a word's context refers to the words surrounding this word in a sentence[3]. In this note, we ignore the order of the surrounding word, or saying we make an exchangeable assumption. This is also called as bag-of-words contexts.

The bag-of-words contexts could be extracted freely from any texts without any annotation. That means the resource from internet could be used directly. And when we get vector representation of word unsupervised, maybe it could be used in off-the -shelf model as an additional feature[4].

2 Model

In this section two kind of models are introduced, the count-based model and the log-bilinear model. The count-based method is very intuitive, but also very effective. In further, Common matrix factorization method could help map the high dimension vector into a low dimension space spanned by more efficient basis. The log-bilinear model introduced here are two recent famous models called Skip-Gram and Glove. The Skip-Gram model could also be described as a three-layer neural network in which the hidden layer's activation function is identify function. It should be noted that all the count-based and log-bilinear models only use bag-of-words contexts information. In fact, there are some essential relations between them.

Contexts Because all these methods use contexts to find the word vector representation, I will first introduce this concept. The basic contexts of a word are the surrounding words. e.g. in sentence " Of cities filled with the foolish", the contexts of "cities" are { of, filled, with, the, foolish}. You could remove some stop words and do tokenization, the contexts may be { fill, foolish}. you could also fix the windows size or use a dynamic windows size or any tricks you like. Bigrams could also

[1] We could further use some sophisticated index such as Huffman code to save more energy.

[2] in machine learning literature, this is called as one-hot representation, which is very useful for formalization

[3] But it could also be other information about the word.

[4] Actually this is not easy as what you think at first glance

be used as contexts. Notice that besides surrounding words, other infos like synatic role in parsing tree could also be used as context[5].

Notice For each word, it has two vector representations. One for the word itself and one for used as context[6].

2.1 word-context matrix

The intuitive idea to represent a word is directly use the contexts of this word. So we could construct a matrix of the counts of word-context co-occurrence. Then each row is the representation of a word by its context counts. We use M^{count} to notate this matrix.

Table 1: The word-context count matrix M^{count}

	c_1	c_2	...	$c_{\|V_c\|}$
w_1	17	0	...	1
w_2	1	42	...	0
...	...	...	...	...
$w_{\|V_w\|}$	2	89	...	0

Another view. The idea behind the word-context explicit representation is clear. Let's revisit this from a different view. First, we make some assumption:

- Any word w could be represented by a vector $\vec{w}$ and any context could be represented by a vector $\vec{c}$.
- $\vec{w} \cdot \vec{c}$ could be used as an association metric of the word w and context c
- The words having similar context will have the similar meaning, or close vector representation

Then for two words w_i and w_j , if they are similar in semantic space, for most context c_k, $\vec{w}_i \cdot \vec{c}_k$ and $\vec{w}_j \cdot \vec{c}_k$ will have similar value. If we put all the products $\vec{w} \cdot \vec{c}$ into a matrix M , in which $M_{ik} = \vec{w}_i \cdot \vec{c}_k$, we get

$$M = WC^{\mathrm{T}} \tag{1}$$

where

$$W = \begin{pmatrix} \vec{w}_1^{\mathrm{T}} \\ \vec{w}_2^{\mathrm{T}} \\ ... \\ \vec{w}_{|V_w|}^{\mathrm{T}} \end{pmatrix}, C^{\mathrm{T}} = \begin{pmatrix} \vec{c}_1 & \vec{c}_2 & ... & \vec{c}_{|V_c|} \end{pmatrix} \tag{2}$$

Let's revisit the word-context count matrix. If we let $M = M^{count}$, then it means we use the co-occurrence count of w_i and c_k $\#(w_i, c_k)$ to measure $\vec{w}_i \cdot \vec{c}_k$. Actually, raw count is not a good measure of the association of word and context, researchers have suggested a better metric called PMI. which is defined as the log ration between w and c's joint probability and the product of their marginal probabilities:

$$PMI(w_i, c_k) = \log \frac{p(w_i, c_k)}{p(w_i)p(c_k)} \tag{3}$$

In further someone finds Positive PMI(PPMI) is better:

$$PPMI(w_i, c_k) = max\{PMI(w_i, c_k), 0\} \tag{4}$$

Notice that PPMI has a disadvantage the it has a big bias on rare context.

[5] in Skip-gram and Glove, the models are described using word context. But they could also involve other types.

[6] in Skip-gram, these two vectors are called input vector representation and output vector representation

SVD After we decide which measurement to use in M, we need to find a factorization of M to get W and C[7]. We could assign W and C like this to get a trivial solution.

$$W = M, C = \Lambda \tag{5}$$

This means all $\vec{c}$ are one-hot vectors (also orthonormal). In this factorization, W will be very sparse. To get a denser representation from this sparse representation, a matrix factorization method called SVD is applied. In SVD, M will be first factorized into $U \cdot \Sigma \cdot V^{\mathrm{T}}$, where U and V are orthonormal and Σ is a diagonal matrix of eigenvalues in decreasing order. By keeping only the top d elements of Σ, we obtain $M_d = U_d \cdot \Sigma_d \cdot V_d^{\mathrm{T}}$. M_d is a low-rank approximation of M.

Now the words could be represented in a new linear space $W^{SVD} = U_d \cdot \Sigma_d$, spanned by a set of more compact basics $C^{SVD} = V_d$.

2.2 Skip Gram Negative Sampling

In mikolov's paper, the objective function of Skip Gram is defined as:

$$argmax_\theta \sum_{t=1}^{\mathrm{T}} \sum_{\substack{-l \le j \le l \\ j \neq 0}} \log p(w_{t+j}|w_t) \tag{6}$$

we could use a more general formation:

$$argmax_\theta \sum_{t=1}^{\mathrm{T}} \sum_{c' \in Context(w_t)} \log p(c'|w_t) \tag{7}$$

Then (6) could be view as a special case of (7) in which $Context(w_t) = \{w_{t+j}| -l \le j \le l, j \neq 0\}$.

The conditional probability of a context given its word is defined as:

$$p(c|w) = \frac{exp(\vec{c} \cdot \vec{w})}{\sum\limits_{c' \in V_c} exp(\vec{c'} \cdot \vec{w})} \tag{8}$$

Neural view Skip Gram is called predict-based model, for it models the prediction probability of a context given a word. It is also called as neural network-based model, for the origin paper of Mikolov gives a description under neural network framework. Equation (6) and (8) could be viewed as a three layer neural network:

- Input layer with one-hot input.
- Hidden layer with identity activation function.
- Softmax output layer.
- The transition between the input layer and hidden layer is the matrix W
- The transition between the hidden layer and output layer is the matrix C

Optimization The objective (6) could be computed by gradient descent directly. But it is expensive due to the summation $\sum\limits_{c' \in V_c} exp(\vec{c'} \cdot \vec{w})$. [8] So Mikolov used an easier objective[9] to replace the $\log p(c|w)$:

$$\log \sigma(\vec{c} \cdot \vec{w}) + \sum_{i=1}^{k} E_{c_i \in P_n(w)} \log \sigma(-\vec{c}_i \cdot \vec{w}) \tag{9}$$

[7] We could view the above as a framework. What we need to do is how to set M and how to do factorization.

[8] There are some approximation methods such as hierarchical softmax and NCE for this problem. But this note will not elaborate on these methods.

[9] The explanation of this objective will not be covered in this note currently.

$P_n(w)$ is the non-context probability distribution of word w. (9) change the output layer's activation function form softmax to sigmoid. And at each iteration, it considers the current context and randomly use another k non-contexts (called as negative samples) to update the parameters.

Trick of SGNS

- dynamic context
- negative sampling number
- negative samples distribution smoothing

2.3 Glove

We will not derive the detials of Glove here. From some reasonable assumptions of the data distribution and restrict the model space by some constraints, Pennington give that a reasonable representation should be:

$$\vec{w}_i \cdot \vec{c}_k + b_i + b_k = \log\left(\#(w_i, c_k)\right) \tag{10}$$

Letting $\vec{w}_i = \{\vec{w}_i, b_i, 1\}$ and $\vec{c}_k = \{\vec{c}_k, 1, b_k\}$, we will get

$$\vec{w}_i \cdot \vec{c}_k = \log\left(\#(w_i, c_k)\right) \tag{11}$$

To solve this, Glove use a weighted least square objective:

$$\sum_i \sum_k \beta\left(\#(w_i, c_k)\right)\left(\vec{w}_i \cdot \vec{c}_k - \log\left(\#(w_i, c_k)\right)\right)^2 \tag{12}$$

$\beta(\#(w_i, c_k))$ is a weight function that you could play with. In Glove, $\beta(x) = min\left\{1, (x/x_{max})^\gamma\right\}$.