2Evaluation Word Embedding Generation Method

Dong Liu
University of Notre Dame
Indiana, USA
dliu9@nd.edu

Meng Jiang
University of Notre Dame
Indiana, USA
mjiang2@nd.edu

***Abstract*—Word embedding is a mapping from words in the text space to vectors in numerical vector space. The key points of word embedding representation is how to generate a good representation of semantic / syntactic context and how to model of the relationship between the context and the target word. There are some famous word embedding generation methods, such as TF-IDF, CBOW, Skip-Gram, FastText, Glove, BERT. Those models are all word embedding pretrained model which will be executed before the NLP tasks are implemented, which means that the word embeddings generated by those methods are task-insensitive. In this paper we design a model that use the multi-task learning technique for word embedding generation which produce well-performed task-sensitive word embedding. Due to the multi-task learning in word embedding generation will have two objectives which are the word embedding evaluation and specific task objective, so we called this method as 2Evaluation Word Embedding Generation.**

***Index Terms*—Word Embedding Generation, Avoid Over-smoothing Problem**

I. Introduction

1. word embedding 2. 3. 4. 5.

Word embedding generation is a basic problem of NLP, many NLP tasks computed numerically based on embeddings. It is a mathematical representation to the semantics meaning of the words. With the form of real-valued vectors, one key idea is that to make word vectors that have the similar meaning closer while for those retaining of different meanings more far away. The performance of word embedding is so important that it can deeply influence the performance of series down-steam tasks like machine translation, QA, etc. There are several impressive algorithms that have shown good performance, however, as any methods have a limitation on the specific tasks, there are still room of improvement for the word embedding generation.

Nowadays, most of NLP tasks have relative small number of objectives compared to the large number of input corpus. In this paper, we solve this problem by introducing word vector evaluation into down-stream NLP tasks, which we called "2Evaluation" word embedding generation. In experiments, this method performs well.

II. Typical Word Embedding Generation Techniques

A. CBOW

The CBOW method uses the context to predict the central word (usually k/2 words before one word and k/2 words after it). From an intuitive perspective, each word pull the information from its context neighbors and aggregate the information to form its own embedding.

B. SkipGram

Different from CBOW, the SkipGram Method is to use a push mode, after each update, the words will send its information to its neighbors, neighbors will make changes according to the received information.

C. FastText

The word and n-gram vectors of the entire document are superimposed and averaged to obtain the document vector, and then the document vector is used for softmax multi-classification.

D. Glove

Construct the word-word co-occurrence matrix. The co-occurrence is established within a fixed window range. After a given range, a V*V matrix can be obtained, where V is the vocabulary size. (Although the density of the matrix is better than the word-document matrix, most of them are also 0)

This method has an objective function shown as below:

$$J = \sum_{i,j=1}^{V} f(X_{ij})(w_i^T w_j + b_i + b_j - log(X_{ij}))^2$$

III. Typical Word Embedding Evaluation Functions

There are several ways for word embedding evaluation nowadays.

A. Word Analogy++ Loss

In the broadest sense, a word analogy is a statement of the form "a is to b as x is to y", which asserts that a and x can be transformed in the same way to get b and y, and vice-versa. Given that this is just an invertible transformation, we can state it more formally:

A word analogy f is an invertible transformation that holds over a set of ordered pairs S if $(x, y) \in S, f(x) = y \bigwedge f^1(y) = x$. When f is of the form $\vec{x} \longmapsto \check{x} + \vec{r}$, it is a linear word analogy.

These linear word analogies, such as $\vec{king}$+$\vec{woman}$ $\vec{man}$ $\vec{queen}$, are what we'd like to explain. When they hold exactly, they form a parallelogram structure in the vector space (by definition):

This work is an output of iSURE program (University of Notre Dame) 2022 of Dong Liu, under the direction of Professor Meng Jiang.

Parallelograms have several useful properties that we can exploit. For one, a quadrilateral is a parallelogram iff each pair of opposite sides is equal in length (and in higher-dimensional spaces, all four points are coplanar). This means that

A linear word analogy holds exactly over a set of ordered word pairs S iff $||x-y||^2$ is the same for every word pair, $||a-x||^2 = ||b-y||^2$ for any two word pairs, and the vectors of all words in S are coplanar.

Word analogy is a method to evaluate the performance of generated word embedding. When we do word analogy evaluation, the corpus for evaluation is a set of sentences and four semantic-correlated words are selected in each sentence. The task is to use the previous four words to predict the fourth word.

$$w_4 = w_1 - w_2 + w_3$$

B. Spearman's rank correlation coefficient

Sparseman's Loss is a word embedding evaluation method. It is based on the word similarity peers. The evaluation corpus is a large set consists of multiple pairs with similar meaning and their similarity score. The first step of calculation of Sparseman's loss is to calculate the difference of the calculted similarity score and the original setting score, and then we need to rank the all together, in the end the Sparseman's loss will be calculated by the cross entropy form of rank.

In statistics, Spearman's rank correlation coefficient or Spearman's ρ, named after Charles Spearman and often denoted by the Greek letter ρ (rho) or as r_s r_s, is a nonparametric measure of rank correlation (statistical dependence between the rankings of two variables). It assesses how well the relationship between two variables can be described using a monotonic function.

The Spearman correlation between two variables is equal to the Pearson correlation between the rank values of those two variables; while Pearson's correlation assesses linear relationships, Spearman's correlation assesses monotonic relationships (whether linear or not). If there are no repeated data values, a perfect Spearman correlation of +1 or 1 occurs when each of the variables is a perfect monotone function of the other.

The Spearman correlation coefficient is defined as the Pearson correlation coefficient between the rank variables.[3]

For a sample of size n, the n raw scores X_i, Y_i are converted to ranks $\mathrm{R}(X_i)$,$\mathrm{R}(Y_i)\mathrm{R}(X_i), \mathrm{R}(Y_i)$, and r_s is computed as $r_s = \rho_{\mathrm{R}(X),\mathrm{R}(Y)} = \frac{\mathrm{cov}(\mathrm{R}(X),\mathrm{R}(Y))}{\sigma_{\mathrm{R}(X)}\sigma_{\mathrm{R}(Y)}}$,$r_s = \rho_{\mathrm{R}(X),\mathrm{R}(Y)} = \frac{\mathrm{cov}(\mathrm{R}(X),\mathrm{R}(Y))}{\sigma_{\mathrm{R}(X)}\sigma_{\mathrm{R}(Y)}}$, where ρ denotes the usual Pearson correlation coefficient, but applied to the rank variables, $\mathrm{cov}(\mathrm{R}(X),\mathrm{R}(Y))\mathrm{cov}(\mathrm{R}(X),\mathrm{R}(Y))$ is the covariance of the rank variables, $\sigma_{\mathrm{R}(X)}\sigma_{\mathrm{R}(X)}$ and $\sigma_{\mathrm{R}(Y)}\sigma_{\mathrm{R}(Y)}$ are the standard deviations of the rank variables. Only if all n ranks are distinct integers, it can be computed using the popular formula $r_s = 1 - \frac{6\sum d_i^2}{n(n^2-1)}$,$r_s = 1 - \frac{6\sum d_i^2}{n(n^2-1)}$, where $d_i = \mathrm{R}(X_i) - \mathrm{R}(Y_i)d_i = \mathrm{R}(X_i) - \mathrm{R}(Y_i)$ is the difference between the two ranks of each observation, n is the number of observations.

IV. Word Analogy Loss Re-Construction

To solve the problem the word analogy loss is only designed to take co-occurence as the evaluation method for consideration, we designed addtional two loss to enhance the explanation of the word analogy semantic loss. Loss

A. Loss1

semantic / syntactic loss

B. Loss2

To solve the problem of un-derivative, we design a derivative loss

C. Loss3

To solve the problem of 1-hot coding to evaluation, we design a loss more smooth and can show more some more relevant words connection either.

V. Model Construction

A. The Designation of the loss

loss = CrossEntropy(orignal values, prediction)
-WordAnalogyLoss-Spearman's rank correlation coefficient

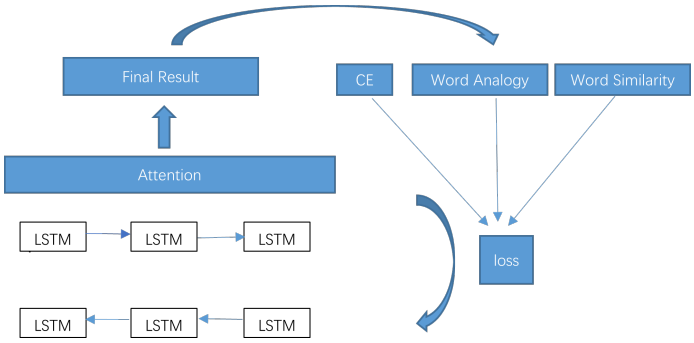

Fig. 1. Loss Model Designation

B. The BiLSTM-Attention Netowrk

In this section we proposed Attention-BiLSTM model, the main components of the model are shown as below:

(1) Input layer: input sentence to this model;

(2) Embedding layer: map each word into a low dimension vector;

(3) LSTM layer: utilize BiiLSTM to get high level features from step (2);

(4) Attention layer: produce a weight vector, and merge word-level features from each time step into a sentence-level feature vector, by multiplying the weight vector;

(5) Output layer: the sentence-level feature vector is finally used for classification.

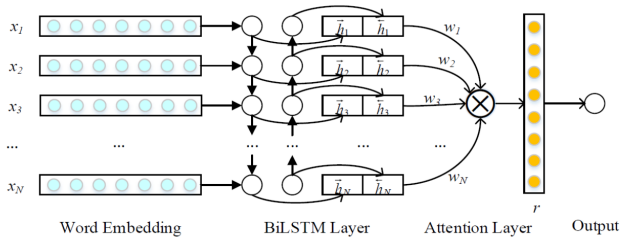

Fig. 2. BiLSTM-Attention Architecture

C. The Method of the supervised construction

Here we pass the derivative form of loss into the original loss to get a supervised loss function.

$$L = L + \frac{\partial L}{\partial x}$$

D. Regularization

In supervised machine learning, the model is trained on a subset of the data, the training data. The goal is to compute the target for each training example based on the training data. Overfitting occurs when a model learns signal and noise in the training data and does not perform well on new data for which the model was not trained. There are several ways to avoid the model from overfitting the training data, such as cross-validation sampling, reducing the number of features, pruning, regularization, etc. Regularization basically increases penalties as model complexity increases. The regularization parameter (lambda) penalizes all parameters except the intercept so that the model generalizes to the data and does not overfit.

Here we use L2 regularization to prevent overfitting. The formula of the L2 regularization is shown as below:

$$J(\theta) = [y_\theta(x) - y]^2 + [\theta_1^2 + \theta_2^2 + ...]$$

With a uniform expression the formula will be shown as below:

$$J(\theta) = [y_\theta(x) - y]^2 + \lambda \sum \theta_i^p$$

Regularization increases the penalty for higher terms as complexity increases. This will reduce the importance given to higher terms and will tend the model towards less complex equations.

The regression model using L1 regularization technique is called Lasso Regression, and the model using L2 is called Ridge Regression. The main difference between the two is the penalty term. Ridge regression was used in this experiment. Ridge regression adds the "squared magnitude" of the coefficients as a penalty term to the loss function. The highlighted part here represents the L2 regularization element. It is worth noting that the choice of lamda in ridge regression is important, if lambda is zero, the regression is close to OLS. If lambda is very large, it will add too many weights, resulting in underfitting.

Finally, to control the strength of this regularization, we use a parameter lambda, and use cross-validation to choose a better lambda. At this time, to unify this type of regularization method, we use p to represent the regularization of the parameter degree of transformation.

VI. Result and Discussion

A. Result

Here is an comparison table about the control-variable experiment.

TABLE I
DIFFEENT LOSS COMPARISON

Task	CE	CE-WA	CE-SL	CE-WA-SL
Sentiment Analysis	82.1%	83.7%	84.2%	88.6%
P.O.S Tagging	74.8%	78.9%	76.1%	83.2%

B. Discussion

Below we will analyze why we can got those results.

1) Why the eval-oriented performs well: The experiments performs in this paper is sentiment analysis and P.O.S tagging, actually they are just 2 or multiple class classification task. The original loss only contains too small class to present the abundant information that contained in the text. Moreover, we should use more interesint things like google docP

2) Why attention is good: I do comparison on the network to comparing the results of the BiLSTM network with an attention or not. The results shows that a network with an attention mechansim will have much better performance. After analysis, we thought that the attention will enhance the performance on weight adjustment on the unseen embedding.

TABLE II
ATTENTION OR NO-ATTENTION COMPARISON

Task	CE-Attention	CE
Sentiment Analysis	82.1%	80.3%
P.O.S Tagging	74.8%	71.2%

3) The effect of normalization: Since the loss in the model is combined of 3 losses, we want each loss has the nearly similar effect on the result so we use the normalization method to scale each loss into the same scale:0,2 to realize normalization.

Below shows an result comparison of this normalization methods.

TABLE III
NORMALIZATION EXPERIMENTS

Task	CE-WA-SL-norm	CE-WA-SL
Sentiment Analysis	82.1%	76.6%
P.O.S Tagging	74.8%	69.8%

4) The effect of regularization: Regularization will make the parameters tuning more normally. It will pay more attention on those smaller number parameters will pay less attention on those larger number parameters.

TABLE IV
REGULARIZATION EXPERIMENTS

Task	CE-WA-SL-reged	CE-WA-SL
Sentiment Analysis	82.1%	79.3%
P.O.S Tagging	74.8%	67.2%

5) The effect of supervised learning: Since the input is a natural language sequence, so here we use supervised learning to enhance the semantic meaning of text.

TABLE V
SUPERVISED LEARNING EXPERIMENTS

Task	CE-WA-SL-suped	CE-WA-SL
Sentiment Analysis	82.1%	80.7%
P.O.S Tagging	74.8%	69.4%

VII. FUTURE GOALS

A. More application of eval-oriented

The 2eval method that reveals that the task-direct oriented method may show a low performance than if we add the unit evaluation in the objective. In natural language processing, the unit orientation is word embedding evaluation. By word analogy and Spearsmans loss we can evaluate how the word embedding generate is and will also help to tune the parameters in a more efficient way, while direct task evaluation maybe too simple for the complex and large set of text for training.

For example, we can use some methods like

B. Some deep learning techniques to enhance the performance

In this paper we use regularization and VAE to enhance the performance, and we use normalization, regularization and supervised learning to improve the performance, those methods always useful for most cases when we construct neural networks.

C. Reinforcement learning to realize word embedding generation