# Semi-unsupervised approach toward learning meaning

Phuc Xuan Nguyen (pxn002@ucsd.edu)
Panqu Wang(pawang@ucsd.edu)
Saekwang Nam(s9nam@ucsd.edu)

March 24, 2012

**Abstract**

In this report, we implement the recursive auto encoder (RAE) method for learning meaning of sentences. We present the missing details for calculating the gradients and for back-propagation techniques. We implement a preliminary experiment to verify the correctness of our derivation for gradients. We get 64.72% accuracy on test set using the dimension of meaning $d = 40$.

## 1 Introduction

The ability to understand the meaning of text documents, especially ones with short length, has been a challenging task. Many forms of short-length texts, such as Twitter messages and product reviews, has become widely available online and the meaning of these tie directly into business and everyday life. Socher et. al.(2011) introduces a novel approach using semi-unsupervised auto encoders toward understanding meanings of sentences. In this report, we will discuss the theory and perform experiments with our own implementation of the framework.

Section 2 will give a brief theoretical overview. We assume that the readers are comfortable with the theory behind Socher's papers. In section 3, we will discuss the preliminary experiments to gain confidence in implementation. In section 4, we will discuss the details of experiments on the movie review dataset.

## 2 Theoretical overview

### 2.1 Representing meaning

The syntactic structure of a sentence in the English language is often represented as a binary tree, in which each word represents a leaf node. Suppose each word is represented as a vector in $R^d$, the value of parent node, given its two children, $x_1$ and $x_2$, is defined as

$$p = tanh(W_1 x_1 + W_2 x_2 + b)$$

.The matrices $W_1$ and $W_2$ and the bias vectors $b$ are the parameters to be trained.

### 2.2 Unsupervised auto encoder

Assume that we have a list of word vectors, $x = \{x_1, ..., x_n\}$, we want to construct a binary tree representing the syntactic structure of the sentence. A combined node is constructed from two nodes, $x_1$and $x_2$, as

$$p = tanh(W_1 x_1 + W_2 x + b).$$

To reconstruct its input, this child node gives rise to two reconstruction nodes, $x_1'$ and $x_2'$, which is mathematically defined as

$$x_1' = tanh(U_1 p + c_1),$$

and,

$$x_2' = tanh(U_2 p + c_2).$$

$U_1$ and $U_2$ are the parameter matrix in $R^{d \times d}$ and $c_1$,$c_2$ are the corresponding bias vectors. The reconstruction error is then defined as

$$E_{rec}([x_1; x_2]; \theta) = \frac{n_1}{n_1 + n_2} ||x_1 - x_1'||^2 + \frac{n_2}{n_1 + n_2} ||x_2 - x_2'||^2,$$

where $n_1$ and $n_2$ are the number of words that are covered by $x_1$and $x_2$ respectively.

## 2.3 Tree structure selection process

The previous section defined the reconstruction error on a given tree structure. When no tree structure is given, we need to construct a tree from the leaf nodes that minimizes reconstruction errors. Socher suggests a greedy approximation to this task. At every step, we consider $n - 1$ pairs consecutive words and pick the one with the lowest reconstruction errors. We repeat this process until there is only one choice left.

## 2.4 Label error

The auto encoder discussed in the previous section is completely unsupervised. It provides a representation for meanings of different sentences. To incorporate the meaning label into the framework, a linear model is added on the meaning of nodes. A softmax layer, or multiclass logistic regression, is laid on top of these nodes. Given $x_k$ as the meaning of a node $k$ and $r$ as the number of alternative labels, the value for the softmax layer is defined as

$$\bar{p} = softmax(V x_k),$$

where $V$ is the parameter matrix in $R^{r \times d}$. The error on these nodes are defined by the log loss of the predictions (cross-entropy errors), or mathematically,

$$E_{ce}(k) = -\sum_{i=1}^{r} t_i log p_i,$$

where $\bar{t}$ is the vector of true label values.

The total error at a node $k$ is defined as

$$E_{total} = \alpha E_{rec} + (1 - \alpha) E_{ce},$$

where $\alpha$ is a hyperparameter representing the trade off between reconstruction error and cross-entropy errors.

## 2.5 Parameters learning

Let $\theta = \{W_1, W_2, U_1, U_2, V, b, c_1, c_2, L\}$ be the parameters where $L$ is the randomly-generated word vectors. We define the loss function as

$$J = \frac{1}{N} \sum_{(x,t)} E(x, t; \theta) + \frac{\lambda}{2} ||\theta||^2,$$

where N is the number of training examples and $\lambda$ is the regularization strength.

The derivative with respect to $\theta$ follows as,

$$\frac{\partial J}{\partial \theta} = \frac{1}{N} \sum_{(x,t)} \frac{\partial E(x, t; \theta)}{\partial \theta} + \lambda \theta$$

In the next section, we will discuss how to calculate this derivative efficiently.

### 2.5.1 Back-propagation

**Modification** For back propagation algorithm to work properly, we add extra error nodes on top of each reconstruction output node. Suppose that input nodes $x_1$ and $x_2$ feed into $p$, which feeds into nodes $x_1'$ and $x_2'$, we add two new output nodes $e_1$ and $e_2$ to the network. Nodes $x_1$ and $x_1'$ feeds into $e_1$. Nodes $x_2$ and $x_2'$ feeds into $e_2$. In matrix form,

$$e_1 = [I, -I][x_1'; x_1]$$

and

$$e_2 = [I, -I][x_2'; x_2].$$

The error defined on these nodes are

$$J = ||e||^2$$

**Mathematical derivation** Given the values for all nodes collected through the forward pass through the neural network, we follow the following steps to calculate the derivatives discussed in the previous section. The notation $\delta$ and $a$ below is based on Elkan's lecture notes (2012) and we assume the reader is comfortable with it.

1. Compute the delta vector for each output node
   (a) For the reconstruction output node $i$, which was reconstructed from $U_1$, the delta is
   $$\delta_k = 2\alpha a_k$$
   (b) For a cross-entropy output node $i$, the delta vector is
   $$\delta = (1-\alpha) \times (-t + softmax(a) \sum_{m=1}^{r} t_m),$$
2. With $\circ$ defined as pointwise multiplication, we compute the delta vector of each internal node as
   (a) For the root node of the tree, using $tanh'(x) = 1 - tanh^2(x)$,
   $$\delta_r = (1-a^2) \circ [(\delta_{rec}^1)^T U_1 + (\delta_{rec}^2)^T U_2 + (\delta_{ce})^T V]^T$$
   (b) Given a node $i$ where feeds into the internal node $k$ and the output node $l$, the delta can be computed as follows:
   $$\begin{aligned} \delta_i^{(1)} &= (1-a^2) \circ [(\delta_{rec}^1)^T U_1 + (\delta_{rec}^2)^T U_2 + (\delta_{ce})^T V + \delta_k^T W_1 - (\delta_l^1)^T]^T \\ \delta_i^{(2)} &= (1-a^2) \circ [(\delta_{rec}^1)^T U_1 + (\delta_{rec}^2)^T U_2 + (\delta_{ce})^T V + \delta_k^T W_2 - (\delta_l^2)^T]^T \end{aligned}$$
3. At each node, we can compute the respective gradient by multiply the calculated $\delta$ by the corresponding matrix. Given $q$ as the value of node feed into the current node,
   (a) the gradient calculation for $X \in \{U_1, U_{2,}, W_1, W_2, V\}$ are
   $$\frac{\partial J}{\partial X} = \delta X$$
   (b) the gradient calculation for $d \in \{b, c_1, c_2\}$ is
   $$\frac{\partial J}{\partial d} = \delta$$
   (c) the gradient calculation for $L$ is
   $$\frac{\partial J}{\partial L} = \delta W_{1or2}$$

Since the objective function is not necessarily continuous and a step in the gradient direction might not decrease objective, we use L-BFGS quasi-Newton method to optimize the parameters as suggested by Socher.

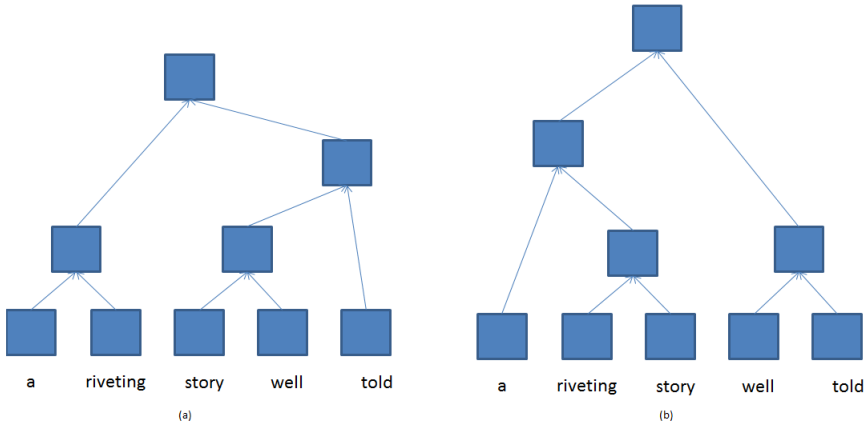


Figure 1: The before (a) and after r(b) tree structure of the same sentences
The "after" tree fits the human understand better than the first. This proves the effect the reconstruction parameters in constructing meaning.

# 3 Preliminary experiments

Due to the complicated nature of the algorithm, we want to make sure that each component of the algorithm works correctly. We run the forward algorithm on a small example($d = 3$, $r = 2$, $\alpha = .2$, $\lambda = .02$, 2 training examples, dictionary of size 5. We check the feature values and the errors at each node against the hand-worked values.

To confirm the accuracy of the back-propagation step, we compare the obtained gradients against the numerical gradients,

$$\frac{\partial J}{\partial w_{ij}} = \frac{J(w_{ij} + \epsilon) - J(w_{ij} - \epsilon)}{2\epsilon} + O(\epsilon^2),$$

where $\epsilon = .0001$. In the small training set described above, the average error over the parameter is in the order of$10^{-8}$.

To further gain confidence in our implementation, we randomly pick a sentence from the test set. We construct a tree with the initial randomized parameters and, then, construct another tree of the same sentence with the trained parameter. The former tree structure is expected to erratic. The latter tree, however, will fit human understanding better. An example of those trees are shown in figure 1.

# 4 Experiment

## 4.1 Final classifiers

After training the parameters, we are able to extract features for each nodes inside the tree. The final classifier use the features of the top nodes $t \in R^d$ and the average of all the nodes, $a = \frac{1}{2n-1}\sum_{i=1}^{2n-1} a_i$ in the tree. Mathematically, the decision is based on

$$\bar{p} = softmax([V, V][t; a]).$$

## 4.2 Design and result

The movie reviews dataset (MR)(Pang and Lee, 2005) has been preprocessed by Socher (2011) into a Matlab data file. All of our experiments are performed on these Matlab files. We split the data set into train (65%), validation (10%), and test (25%). Due to time constraint, we set the dimension of the meaning vectors at

| False positives |
|---|
| . . . about as exciting to watch as two last-place basketball teams playing one another on the final day of the season . |
| you're too conscious of the effort it takes to be this spontaneous . |
| the cast is so low-wattage that none of the characters comes off as big . . . and the setting remains indistinct . |
| borrows from other movies like it in the most ordinary and obvious fashion . |
| shows that jackie chan is getting older , and that's something i would rather live in denial about |
| **False negatives** |
| despite its faults , gangs UNK(excels) in spectacle and pacing . |
| alan and his fellow survivors are idiosyncratic enough to lift the movie above its playwriting 101 premise . |
| it ain't art , by a long shot , but unlike last year's lame musketeer , this dumas adaptation entertains . |
| a mess , but it's a sincere mess |
| UNK(touché) ! |

Table 1: Examples of false positives and false negatives.
UNK is short for unknown. The word in the parenthesis next to UNK is the actual word from the original dataset.

| Negative | Positive |
|---|---|
| bore; weak and ineffective; stupid; | good; funny; effective; cute; rocks; |
| too silly; bad; awkward; too bad; | pleasant; very funny; powerful |
| sloppy; mess; cliche | best; sweet; touching |

Table 2: Visualization of semantic vectors. Words and phrases that has the top probabilities for negative and positive meanings.

$d = 40$, instead of 100-dimension vectors. We expect the performance to be lower than Socher. At the same time, we, however, expect our results would still hold reasonable meanings.

Each word vector was generated independently from a Gaussian distribution with mean of 0 and covariance matrix of $.002I$. We want the values of these vectors to be small so that it stays within the unsaturated part of the sigmoid functions.

We perform grid search on the strength of regularization, $\lambda$, and the error weight, $\alpha$ with the inaccuracy as the objective function. The ranges of values for $\lambda$ and $\alpha$ are $[0, 0.1]$ with the increment of .01 and $[0, 1]$ with the increment of .1. The best values for the settings are $\alpha = .3$ and $\lambda = .02$.

After training the weight matrix, we use the classifiers described in the last section to predict the labels of the training examples. We obtained the classification rate of 64.72% on the test set. Table 1 shows 5 examples from the false positive and false negative sets.

We also want to visualize which phrases have strong effect on the final prediction. In table 2, we show the list of words and phrases that have the high probability of being negative and positives.

## 4.3 Discussion

As expected, classification rate is lower than the figure reported in Socher's paper as the words vector's dimension is higher in the latter. Looking through the false positives and false negatives are often as informative as looking at the accuracy performance figures. The learning technique seems to fail when the sentence contains sarcasm. According to the encoders' results, unknown words often carries bad meaning. If the positive keyword in the sentence is unknown, false negative is bound to happen.

# 5 Conclusion

In this report, we present an overview of a semi-unsupervised learning framework toward understanding meaning of sentences. We discuss the elements of unsupervised auto encoder method and how to add label classification in the framework. We show the results of different experiments run by our implementation of the auto encoder.

# References


[1] Semi-Supervised Recursive Autoencoders for Predicting Sentiment Distributions Richard Socher, Jeffrey Pennington, Eric Huang, Andrew Y. Ng, and Christopher D. Manning Conference on Empirical Methods in Natural Language Processing (EMNLP 2011, Oral)

[2] B. Pang and L. Lee. 2005. Seeing stars: Exploiting class relationships for sentiment categorization with respect to rating scales. In ACL, pages 115–124.

[3] Learning the meanings of sentences. Lecture notes. Charles Elkan, March 16, 2012.