Summarisation of Three Papers about NLP

Natural language processing (NLP) is one of the most important technologies of the information age. Understanding complex language utterances is also a crucial part of artificial intelligence. Applications of NLP are everywhere because people communicate most everything in language: web search, advertisement, emails, customer service, language translation, radiology reports, etc.

——Description of Stanford's NLP with Deep Learning Course

These three papers provide three models in sentiment analysis. The Semi-Supervised Recursive Autoencoders and the Recursive Deep Models are used for predicting sentiment distributions for phrase. The GloVe model is used for obtaining vector representations for words in a step.

In the first paper (Semi-Supervised Recursive Autoencoders for Predicting Sentiment Distributions), when doing text analysis, the vector space of a certain paragraph of text can be obtained according to the word vector, and then the vector representation of the whole paragraph can be obtained by analyzing it layer by layer.

In the Recursive Deep Models for Semantic Compositionality Over a Sentiment Treebank, it accurately captures the range of tree levels of negative effects and various positive and negative sentences. Recently I learned decision tree learning in machine learning course. But finally I find the treebank is much different from decision tree learning. Decision tree learning try to find the attribute to the classifies. Treebank is a parsed text corpus that annotates syntactic or semantic sentence structure. In the model, it is a mutually connected neural network.

The GloVe model is an unsupervised learning algorithm for obtaining vector representations for words. As a word2vec model it work by using Naive Bayes. It is a little difficult to me who have no idea with Skip-gram model and SVD. Maybe I can understand the GloVe model better by spending more time on learning a simple Skip-gram model. In a conclusion, these papers are significance in structure semantic understanding to computer.

Semi-Supervised Recursive Autoencoders for Predicting Sentiment Distributions

Abstract

In this paper a novel machine learning framework based on recursive autoencoders for sentence-level prediction of sentiment label distributions is introduced. For the record, the method doesn't learn vector space representations for words but phrases.

Introduction

The author point out that a one-dimensional scale does not accurately reflect the complexity of human emotions and sentiments. These are three issues to solve:

(i) Instead of using a bag-of-words representation, the model exploits hierarchical structure and uses compositional semantics to understand sentiment.

(ii) The system can be trained both on unlabeled domain data and on supervised sentiment data and does not require any language-specific sentiment lexica, parsers, etc.

(iii) Rather than limiting sentiment to a positive/negative scale, this model predicts a multidimensional distribution over several complex, interconnected sentiments.

Model

Aim: To find vector representations for variable-sized phrases in either unsupervised or semi-supervised training regimes.

Neural Word Representations: By exploring two settings, we can use the resulting matrix of word vectors L for subsequent tasks as follows. $x_i = Lb_k \in \mathbb{R}^n$. Mathematically, this look-up operation can be seen as a simple projection layer where we use a binary vector b which is zero in all positions except at the k_{th} index.

Traditional Recursive Autoencoders: Before give the model of author's, the paper introduces the traditional recursive autoencoders. This part explains the way to obtain a reduced dimensional vector representation for sentences.

(i) The first parent vector y_1 is computed from the children $(c_1,c_2) = (x_3,x_4)$:

$$p = f(W^{(1)}[c_1; c_2] + b^{(1)}) \quad (1)$$

(ii) To assessing how well this n-dimensional vector represents its children (by trying to reconstruct the children in a reconstruction layer):

$$[c_1'; c_2'] = W^{(2)}p + b^{(2)} \quad (2)$$

(iii) To minimize the reconstruction errors. For each pair, we compute the Euclidean distance between the original input and its reconstruction.

$$E_{rec}([c_1; c_2]) = \frac{1}{2}\left|\left|[c_1; c_2] - [c_1'; c_2']\right|\right|^2 \quad (3)$$

In my opinion, it is just as the error function in multilayer neural networks.

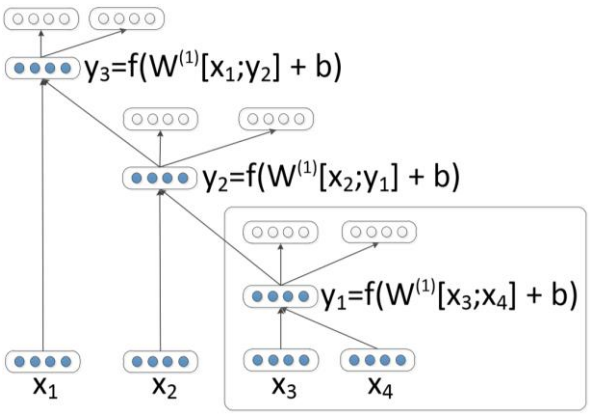

Figure 1

Unsupervised Recursive Autoencoder for Structure Prediction: In this case, we have no tree structure given for the input vectors in x. And the reconstruction errors changes into:

$$RAE_\theta(x) = \arg\min_{y \in A(x)} \sum_{s \in T(y)} E_{rec}([c_1; c_2]_s) \tag{4}$$

(i) Greedy Unsupervised RAE. In this way, it selects the pair which had the lowest reconstruction error (Erec) and its parent representation p will represent this phrase and replace both children in the sentence word list.

(ii) Weighted Reconstruction.To solve the problem with simply using the reconstruction error of both children equally as describe in TRA's (iii) is that each child could represent a different number of previously collapsed words and is hence of bigger importance for the overall meaning reconstruction of the sentence. By re-defining the reconstruction error is:

$$E_{rec}([c_1; c_2]; \theta) = \frac{n_1}{n_1 + n_2} \left|\left|c_1 - c_1'\right|\right|^2 + \frac{n_2}{n_1 + n_2} \left|\left|c_2 - c_2'\right|\right|^2 \tag{5}$$

(iii) Length Normalization. $p = \frac{p}{||p||}$

Semi-Supervised Recursive Autoencoders: Extending RAEs to a semi-supervised setting in order to predict a sentence or phrase-level target distribution t.

(i) Add on top of each parent node a simple softmax layer to predict class distributions

$$d(p; \theta) = \text{softmax}(W^{label} p) \tag{6}$$

(ii) Illustration of an RAE unit at a nonterminal tree node. Deep nodes show the supervised softmax layer for label distribution prediction

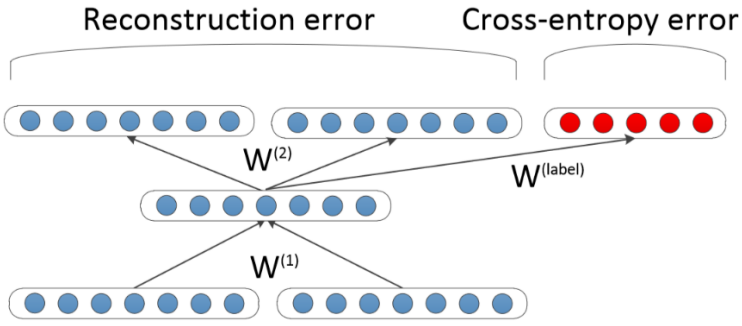

Figure 2

(iii) Let t_k be the k_{th} element of the multinomial target label distribution t for one entry. The softmax layer's outputs are interpreted as conditional probabilities $d_k = p(k|[c1;c2])$, hence the cross-entropy error is

$$E_{cE}(p, t; \theta) = -\sum_{k=1}^{K} t_k \log d_k(p; \theta) \tag{7}$$

(iv) the final semi-supervised RAE objective over (sentences, label) pairs (x,t) in a corpus becomes

$$J = \frac{1}{N} \sum_{(x,t)} E(x, t; \theta) + \frac{\lambda}{2} ||\theta||^2 \tag{8}$$

$$E(x,t;\theta) = \sum_{s \in T(RAE_\theta(x))} E([c_1;c_2]_s, p_s, t, \theta) \tag{9}$$

(v) The error at each nonterminal node is the weighted sum of reconstruction and cross-entropy errors

$$E([c_1;c_2]_s, p_s, t, \theta) = \alpha E_{rec}([c_1;c_2]_s;\theta) + (1-\alpha)E_{cE}(p_s,t;\theta) \tag{10}$$

Learning: We used gradient descent to set the parameters:

$$\frac{\partial J}{\partial \theta} = \frac{1}{N}\sum_{(x,t)} \frac{\partial E(x,t;\theta)}{\partial \theta} + \lambda\theta \tag{11}$$

Experiments

The paper first describes the new experience project (EP) dataset, results of standard classification tasks on this dataset and how to predict its sentiment label distributions. The paper then shows results on other commonly used datasets and conclude with an analysis of the important parameters of the model. The accuracy of predicting the class with most votes is

Method	Accuracy
Random	20.0
Most Frequent	38.1
Baseline 1: Binary BoW	46.4
Baseline 2: Features	47.0
Baseline 3: Word Vectors	45.5
RAE (our method)	**50.1**

Figure 3

What's more accuracy of sentiment classification on movie review polarity (MR) and the MPQA dataset is

Method	MR	MPQA
Voting with two lexica	63.1	81.7
Rule-based reversal on trees	62.9	82.8
Bag of features with reversal	76.4	84.1
Tree-CRF (Nakagawa et al,'10)	77.3	86.1
RAE (random word init.)	76.8	85.7
RAE (our method)	**77.7**	**86.4**

Figure 4

Related Work

Autoencoders and Deep Learning. One of neural networks. Autoencoders learn a reduced dimensional representation of fixed-size inputs such as image patches or bag-of-word representations of text documents.

Sentiment Analysis. Pang etal. were one of the first to experiment with sentiment classification. Most previous work is centered around a given sentiment lexicon or building one via heuristics, manual annotation or machine learning techniques. In contrast, this model do not require an initial or constructed sentiment lexicon of positive and negative words. Furthermore, this model Predicted in terms of a multinomial distribution over several interconnected.

Recursive Deep Models for Semantic Compositionality Over a Sentiment Treebank

Abstract

This paper give us a model having great improving in capturing the effects of negation and its scope at tree levels for positive and negative phrases.

Introduction

There are two datasets:

(i) **The Stanford Sentiment Treebank:** the first corpus with fully labeled parse trees that allows for a complete analysis of the compositional effects of sentiment in language. It solves the problem of capture the meaning of longer phrases.

(ii) **The Recursive Neural Tensor Network:** which capture the compositional effects with higher accuracy.

Model

Stanford Sentiment Treebank. It includes labels for every syntactically plausible phrase in thousands of sentences, allowing us to train and evaluate compositional models. And it divides into 5-class : positive, somewhat positive, neutral, somewhat negative or negative.

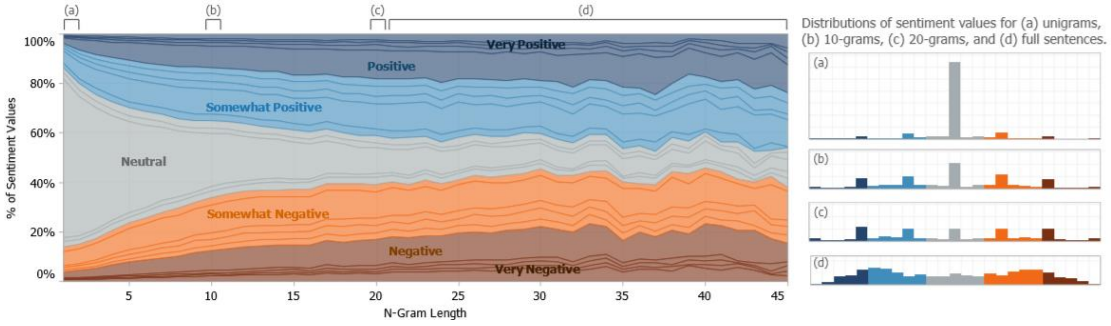

Figure 5

Recursive Neural Models. This model computes compositional vector representations for phrases of variable length and syntactic type.

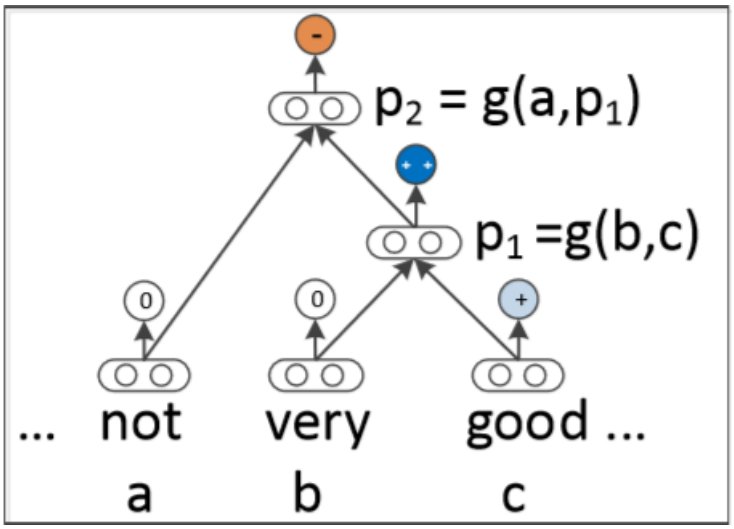

Figure 6

For classification into five classes, we compute the posterior probability over labels given the word vector via:

$$y^a = \text{softmax}(W_s a) \tag{12}$$

RNN: Recursive Neural Network.

RNNs use the following equations to compute the parent vectors:

$$p_1 = f\left(W\begin{bmatrix} b \\ c \end{bmatrix}\right), p_2 = f\left(W\begin{bmatrix} a \\ p_1 \end{bmatrix}\right) \tag{13}$$

This model uses the same compositionality function as the recursive autoencoder (Socher et al., 2011b) and recursive auto-associate memories (Pollack,1990). The only difference to the former model is that we fix the tree structures and ignore the reconstruction loss.

MV-RNN: Matrix-Vector RNN.

The aim is to represent every word and longer phrase in a parse tree as both a vector and a matrix.

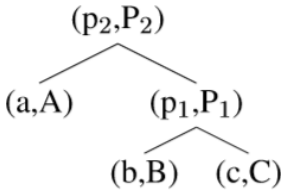

Figure 7 The tree with nodes

MV-RNNs use the following equations to compute the parent vectors:

$$p_1 = f\left(W\begin{bmatrix} Cb \\ Bc \end{bmatrix}\right), P_1 = f\left(W_M\begin{bmatrix} B \\ C \end{bmatrix}\right) \tag{14}$$

RNTN: Recursive Neural Tensor Network.

The main idea is to use the same, tensor-based composition function for all nodes.

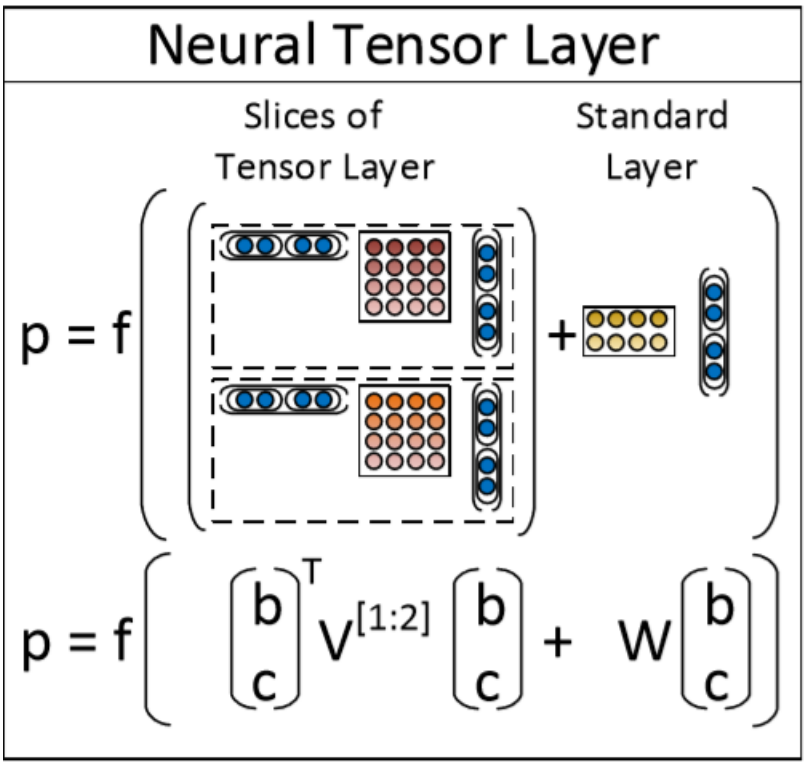

Figure 8 A single tensor layer

The output of a tensor product $h \in R^d$ via the following vectorized notation and the equivalent but more detailed notation for each slice $V^{[i]} \in R^{d \times d}$:

$$h = \begin{bmatrix} b \\ c \end{bmatrix}^T V^{[1:d]} \begin{bmatrix} b \\ c \end{bmatrix}; h_i = \begin{bmatrix} b \\ c \end{bmatrix}^T V^{[i]} \begin{bmatrix} b \\ c \end{bmatrix} \tag{15}$$

The RNTN uses this definition for computing p1:

$$p_1 = f\left(\begin{bmatrix} b \\ c \end{bmatrix}^T V^{[1:d]} \begin{bmatrix} b \\ c \end{bmatrix} + W \begin{bmatrix} b \\ c \end{bmatrix}\right) \tag{16}$$

The next parent vector p2 in the tri-gram will be computed with the same weights:

$$p_2 = f\left(\begin{bmatrix} a \\ p_1 \end{bmatrix}^T V^{[1:d]} \begin{bmatrix} a \\ p_1 \end{bmatrix} + W \begin{bmatrix} a \\ p_1 \end{bmatrix}\right) \tag{17}$$

In this model, the tensor can directly relate input vectors

Tensor Backprop through Structure.

This is equivalent (up to a constant) to minimizing the KL-divergence between the two distributions. The error as a function of the RNTN parameters θ = (V, W, Ws, L) for a sentence is:

$$E(\theta) = \sum_i \sum_j t_j^i \log y_j^i + \lambda \|\theta\|^2 \tag{18}$$

The softmax error vector at node i:

$$\delta^{i,s} = \left(W_s^T (y^i - t^i)\right) \otimes f'(x^i) \tag{19}$$

The remaining derivatives can only be computed in a top-down fashion from the top node through the tree and into the leaf nodes. The full derivative for V and W is the sum of the derivatives at each

of the nodes. For the derivative of each slice k = 1,...,d, we get:

$$\frac{\partial E^{p_2}}{\partial V^{[k]}} = \delta_k^{p_2,com} \begin{bmatrix} a \\ p_1 \end{bmatrix} \begin{bmatrix} a \\ p_1 \end{bmatrix}^T \tag{20}$$

And for the children:

$$\delta^{p_2,down} = \left(W^T \delta^{p_2,com} + S \right) \otimes f' \left(\begin{bmatrix} a \\ p_1 \end{bmatrix} \right) \tag{21}$$

where we define

$$S = \sum_{k=1}^{d} \delta_k^{p_2,com} \left(V^{[k]} + \left(V^{[k]} \right)^T \right) \begin{bmatrix} a \\ p_1 \end{bmatrix} \tag{22}$$

In particular, we have

$$\delta^{p_1,com} = \delta^{p_1,s} + \delta^{p_2,down}[d+1:2d] \tag{23}$$

The full derivative for slice $V_{[k]}$ for this trigram tree:

$$\frac{\partial E}{\partial V^{[k]}} = \frac{E^{p_2}}{\partial V^{[k]}} + \delta_k^{p_1,com} \begin{bmatrix} b \\ c \end{bmatrix} \begin{bmatrix} b \\ c \end{bmatrix}^T \tag{24}$$

The W is the same as V.

Experiments

RNTN prediction of positive and negative (bottom right) sentences and their negation

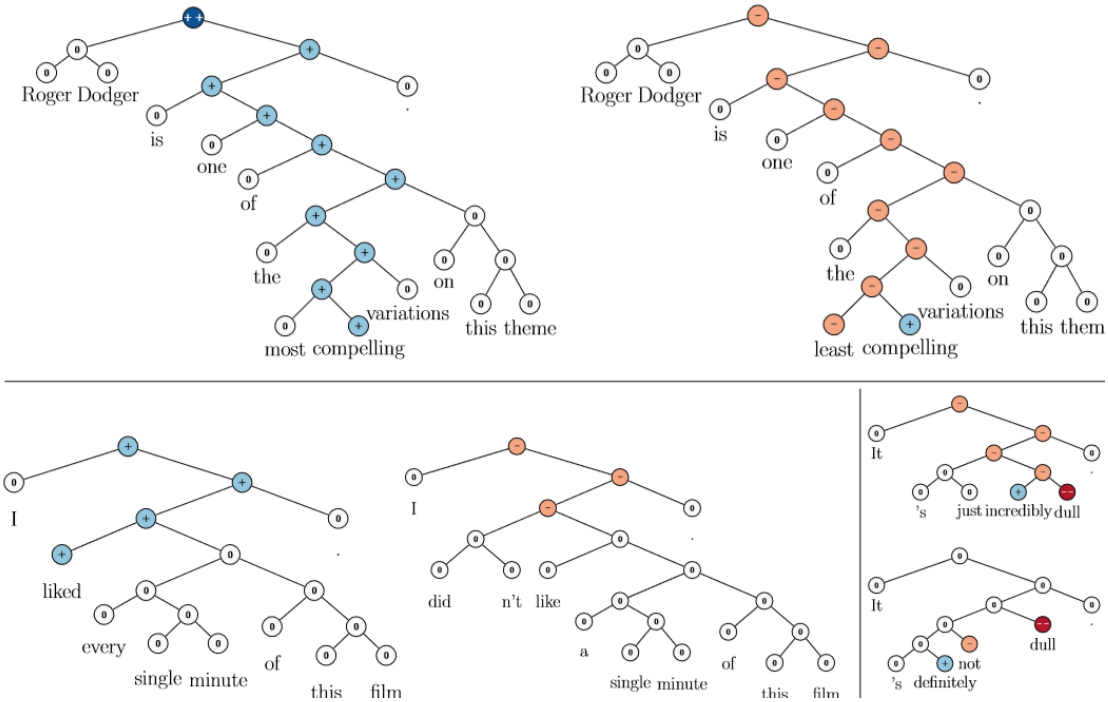

Figure 9

Change in activations for negations. Only the RNTN correctly captures both types. It decreases positive sentiment more when it is negated and learns that negating negative phrases (such as not terrible) should increase neutral and positive activations

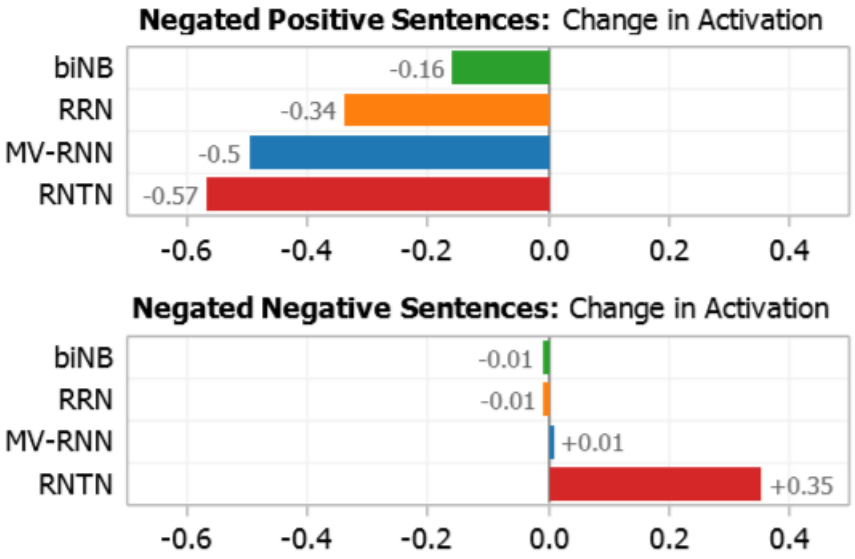

Figure 10

The correctly picks the negative and positive examples.

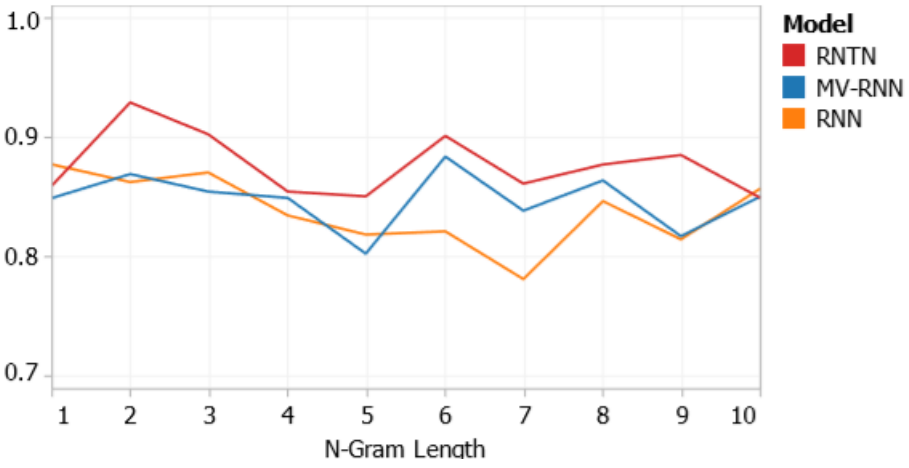

Figure 11

Related Work

Semantic Vector Spaces, Compositionality in Vector Spaces, Logical Form, Deep Learning, Sentiment Analysis (some may be repeat and we won't discussed here)

GloVe: Global Vectors for Word Representation

Abstract

This model efficiently leverages statistical information by training only on the nonzero elements in a word-word co-occurrence matrix. The model produces a vector space with meaningful substructure. And this model is used named entity recognition.

Introduction

The paper analyzes the model properties necessary to produce linear directions of meaning and argue that global log-bilinear regression models are appropriate for doing so.

Model

Let $X_i = \sum_k X_{ik}$ be the number of times any word appears in the context of word i.
Let $P_{ij} = P(j|i) = X_{ij}/X_i$ be the probability that word j appear in the context of word i.
Let $\text{ratio} = P_{ik}/P_{jk}$ means the ratio of two conditional probabilities.

ratio	j is relevant with k	j is irrelevant with k
i is relevant with k	Ratio $\approx$ 1	Ratio >>1
i is irrelevant with k	Ratio <<1	Ratio $\approx$ 1

ratio depends on three words i, j, and k, the most general model takes the form

$$F(w_i, w_j, \tilde{w}_k) = \frac{P_{ik}}{P_{jk}} \tag{25}$$

In this equation, the right-hand side is extracted from the corpus, and F may depend on some as-of-yet unspecified parameters. w and $\tilde{w}$ are equivalent and differ only as a result of their random initializations. To depend only on the difference of the two target words, modifying to

$$F(w_i - w_j, \tilde{w}_k) = \frac{P_{ik}}{P_{jk}} \tag{26}$$

To avoid obfuscating the linear structure we are trying to capture,

$$F\left((w_i - w_j)^T \tilde{w}_k\right) = \frac{P_{ik}}{P_{jk}} \tag{27}$$

First, we require that F be a homomorphism between the groups (R,+) and (R>0,×):

$$F\left((w_i - w_j)^T \tilde{w}_k\right) = \frac{F(w_i^T \tilde{w}_k)}{F(w_j^T \tilde{w}_k)} \tag{28}$$

The solution is F = exp, so

$$w_i^T \tilde{w}_k = \log(P_{ik}) = \log(X_{ik}) - \log(X_i) \tag{29}$$

As we know $w_i^T w_j = w_j^T w_i$, but $\log(P_{ij}) \neq \log(P_{ji})$. Hence, author adds an additional bias $\tilde{b}_k$ for $\tilde{w}_k$ restores the symmetry

$$w_i^T \tilde{w}_k + b_i + \tilde{b}_k = \log(X_{ik}) \tag{30}$$

We propose a new weighted least squares regression model that addresses these problems.

$$J = \sum_{i,j=1}^{V} f\left(X_{ij}\right)\left(w_i^T \tilde{w}_j + b_i + \tilde{b}_j - \log X_{ij}\right)^2 \tag{31}$$

Where V is the size of the vocabulary. One class of f(x) that author found to work well can be parameterized as

$$f(x) = \begin{cases} (x/x_{\max})^{\alpha} & \text{if } x < x_{\max} \\ 1 & \text{otherwise} \end{cases} \tag{32}$$

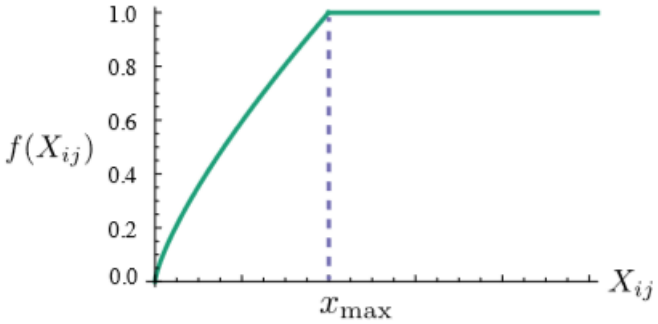

Figure 12 Weighting function f with α = 3/4

Relationship to Other Models. Author shows how window-based methods are related to their proposed model.

Complexity of the model. It concludes that the complexity of the model is much better than the worst case $O(V^2)$, and in fact it does somewhat better than the on-line window-based methods which scale like O(|C|).

Experiments

The paper conduct experiments on the word analogy task of Mikolov et al. (2013a), a variety of word similarity tasks, as described in (Luong et al., 2013), and on the CoNLL-2003 shared benchmark dataset for NER

Model	Size	WS353	MC	RG	SCWS	RW
SVD	6B	35.3	35.1	42.5	38.3	25.6
SVD-S	6B	56.5	71.5	71.0	53.6	34.7
SVD-L	6B	65.7	72.7	75.1	56.5	37.0
CBOW†	6B	57.2	65.6	68.2	57.0	32.5
SG†	6B	62.8	65.2	69.7	58.1	37.2
GloVe	6B	65.8	72.7	77.8	53.9	38.1
SVD-L	42B	74.0	76.4	74.1	58.3	39.9
GloVe	42B	**75.9**	**83.6**	**82.9**	**59.6**	**47.8**
CBOW*	100B	68.4	79.6	75.4	59.4	45.5

Figure 13 Results on the word analogy task, given as percent accuracy

The GloVe model performs significantly better than the other baselines, often with smaller vector sizes and smaller corpora.

Model	Size	WS353	MC	RG	SCWS	RW
SVD	6B	35.3	35.1	42.5	38.3	25.6
SVD-S	6B	56.5	71.5	71.0	53.6	34.7
SVD-L	6B	65.7	72.7	75.1	56.5	37.0
CBOW†	6B	57.2	65.6	68.2	57.0	32.5
SG†	6B	62.8	65.2	69.7	58.1	37.2
GloVe	6B	65.8	72.7	77.8	53.9	38.1
SVD-L	42B	74.0	76.4	74.1	58.3	39.9
GloVe	42B	**75.9**	**83.6**	**82.9**	**59.6**	**47.8**
CBOW*	100B	68.4	79.6	75.4	59.4	45.5

Figure 14 Spearman rank correlation on word similarity tasks

The table above shows results on five different word similarity datasets. GloVe outperforms CBOW while using a corpus less than half the size.

Model	Dev	Test	ACE	MUC7
Discrete	91.0	85.4	77.4	73.4
SVD	90.8	85.7	77.3	73.7
SVD-S	91.0	85.5	77.6	74.3
SVD-L	90.5	84.8	73.6	71.5
HPCA	92.6	**88.7**	81.7	80.7
HSMN	90.5	85.7	78.7	74.7
CW	92.2	87.4	81.7	80.2
CBOW	93.1	88.2	82.2	81.1
GloVe	**93.2**	88.3	**82.9**	**82.2**

Figure 15 F1 score on NER task with 50d vectors

Author concludes that the GloVe vectors are useful in downstream NLP tasks, as was first shown for neural vectors in (Turian et al., 2010).

Related Work

Matrix Factorization Methods. It is be used for generating low-dimensional word representations have roots stretching as far back as LSA.

Shallow Window-Based Methods. To learn word representations that aid in making predictions within local context windows. But compared with MFM it fails to take advantage of the vast amount of repetition in the data.