Recursive Neural Networks for sentiment analysis

1 Motivation

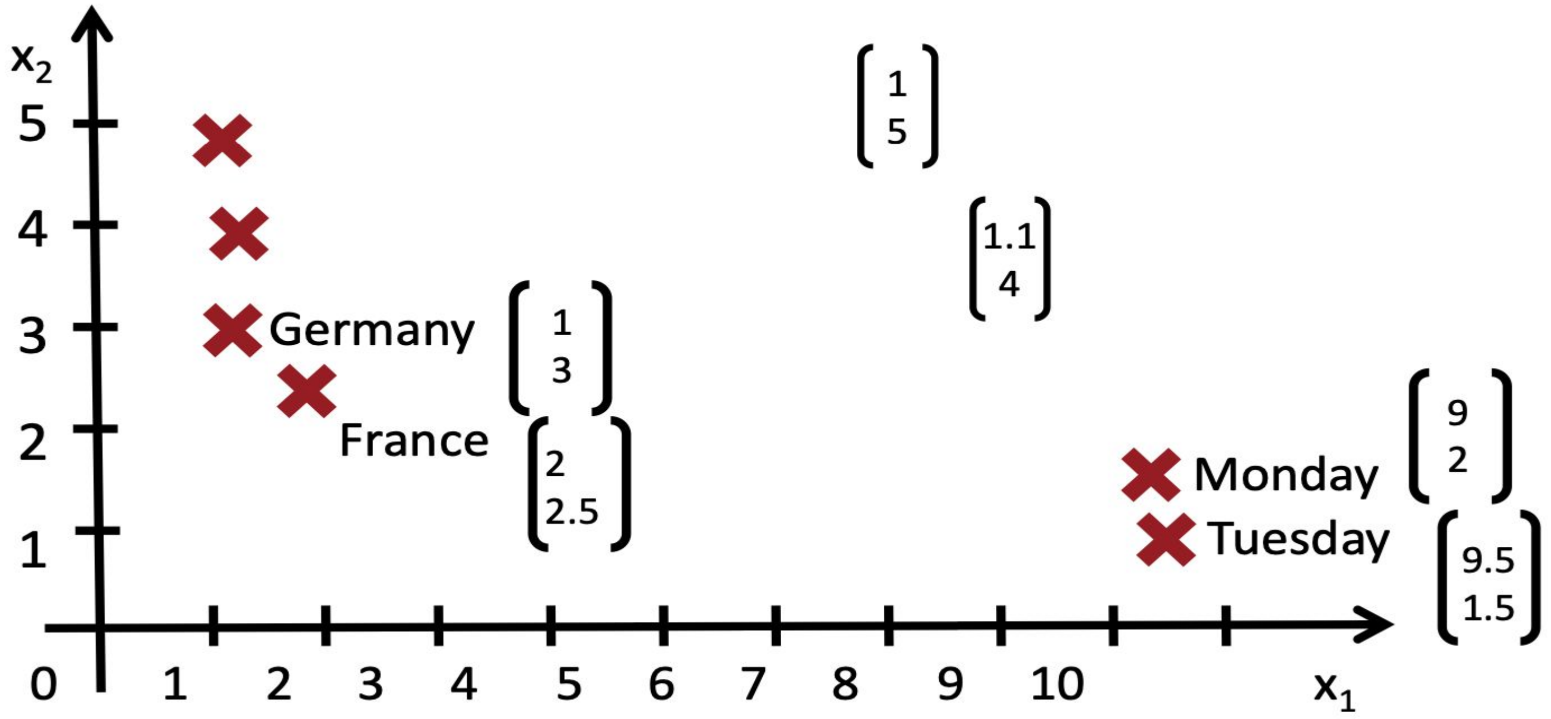
2 Recursive Neural Network

3 Some examples

Motivation

1 Building the mapping of phrase in vector space

- **Given**: word embeddings (w2v, Glove, fasttext, etc)
- How to construct the mapping of the phrase (sequence of words) to the vector space?



adapted from Deep Learning for NLP (without Magic), Richard Socher and Christopher Manning

1 Why do we need vectors of phrases?

- Word embeddings are built with the help of distributional techniques and could capture the similarity across words, but *cannot capture the meaning of the whole phrase.*

- Many models (Bag of Word, LSA, LDA) ignore word order, they are good for the Information Retrieval, Topic Modelling etc.

- For the tasks like sentiment analysis, paraphrase detection, etc we need to add compositionality of the language for better capturing the meaning of the whole phrase.

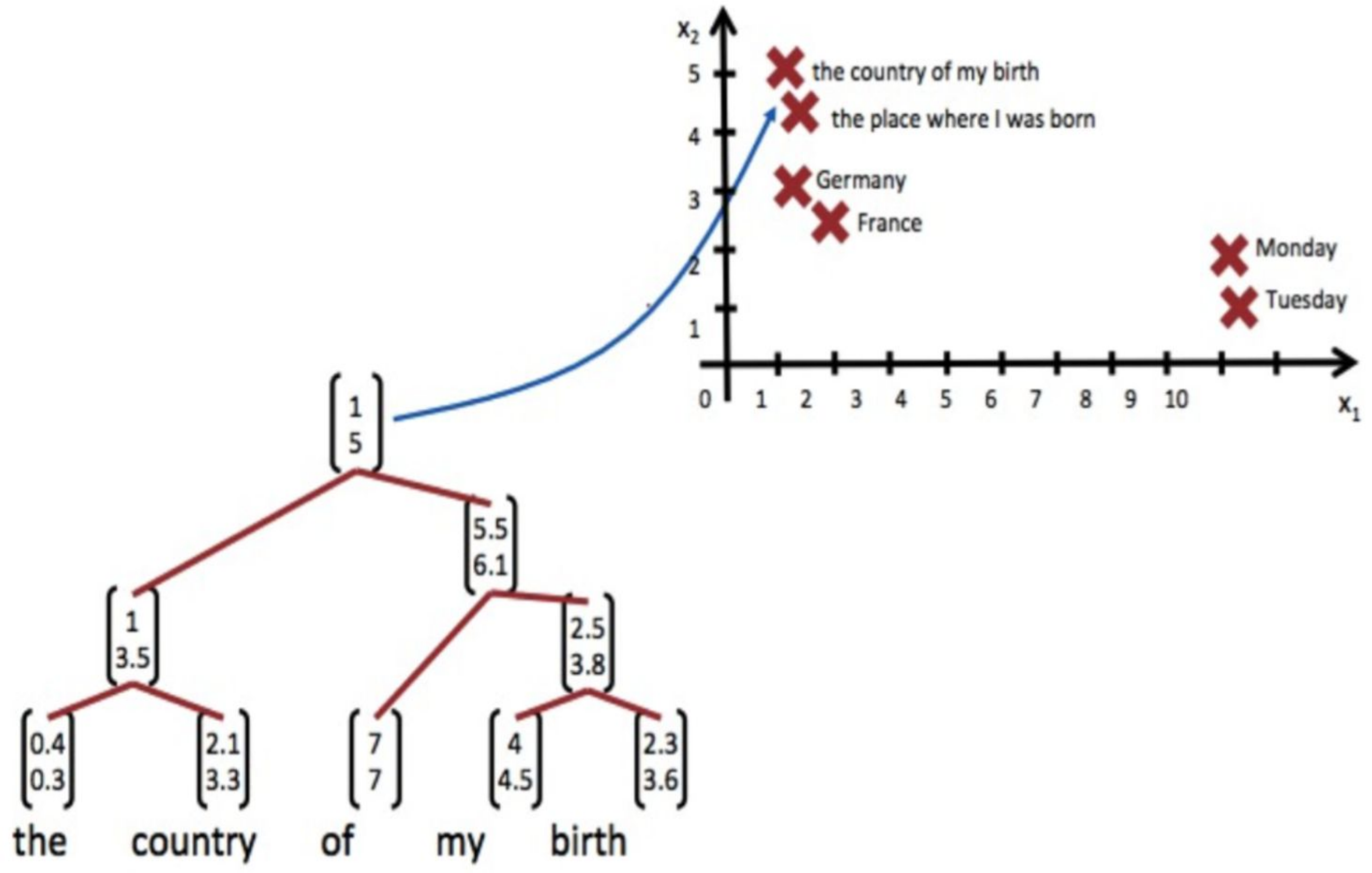
The **Principle** of **Semantic Compositionality** is the **principle** that the meaning of a (syntactically complex) whole is a function only of the meanings of its (syntactic) parts together with the manner in which these parts were combined (*Pelletier, Francis Jeffry. "The principle of semantic compositionality."*).

1 Building the mapping of sentence in vector space

Use this principle:

The meaning (vector) of a sentence is determined by

1. the meanings of its words;
2. the rules that combine them.



x_2
x_1
0
1
2
3
4
5
6
7
8
9
10
the country of my birth
the place where I was born
Germany
France
Monday
Tuesday
$\begin{bmatrix}1\\5\end{bmatrix}$
$\begin{bmatrix}1\\3.5\end{bmatrix}$
$\begin{bmatrix}5.5\\6.1\end{bmatrix}$
$\begin{bmatrix}2.5\\3.8\end{bmatrix}$
$\begin{bmatrix}0.4\\0.3\end{bmatrix}$
$\begin{bmatrix}2.1\\3.3\end{bmatrix}$
$\begin{bmatrix}7\\7\end{bmatrix}$
$\begin{bmatrix}4\\4.5\end{bmatrix}$
$\begin{bmatrix}2.3\\3.6\end{bmatrix}$
the
country
of
my
birth

adapted from Deep Learning for NLP (without Magic), Richard Socher and Christopher Manning

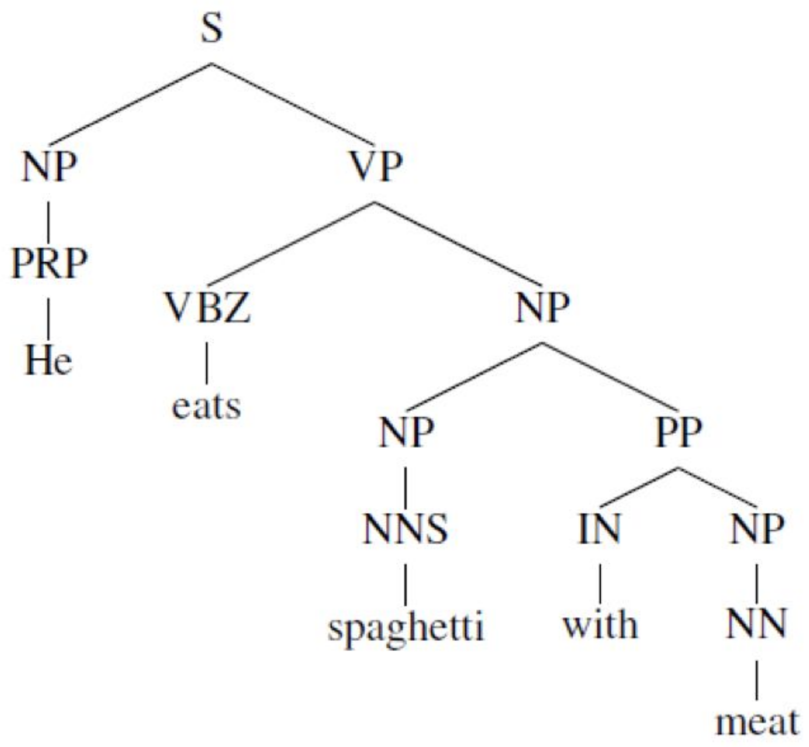
1

How to map the sentence to the vector?

- Use hand-crafted features (why is is not so efficient was explained in the Lecture 4);
- Use word embeddings and MLP (Neural Networks like DAN) - good idea, but ignores information about structure of the sentences.
- Need the way to combine the meaning of the words (given by the word embeddings) and rules that is representing the structure of the sentence.

Possible solution:
use the syntactic parsers to represent the structure of the sentence (The lectures about parsing are included in the course).

Main idea:
let's represent a Neural Network structure as Directed Acyclic Graph, given by the parse tree.



picture from the tutorial “Wrap up: LSTMs and Recursive Neural Networks”

Recursive Neural Networks

2

Recursive Neural Networks

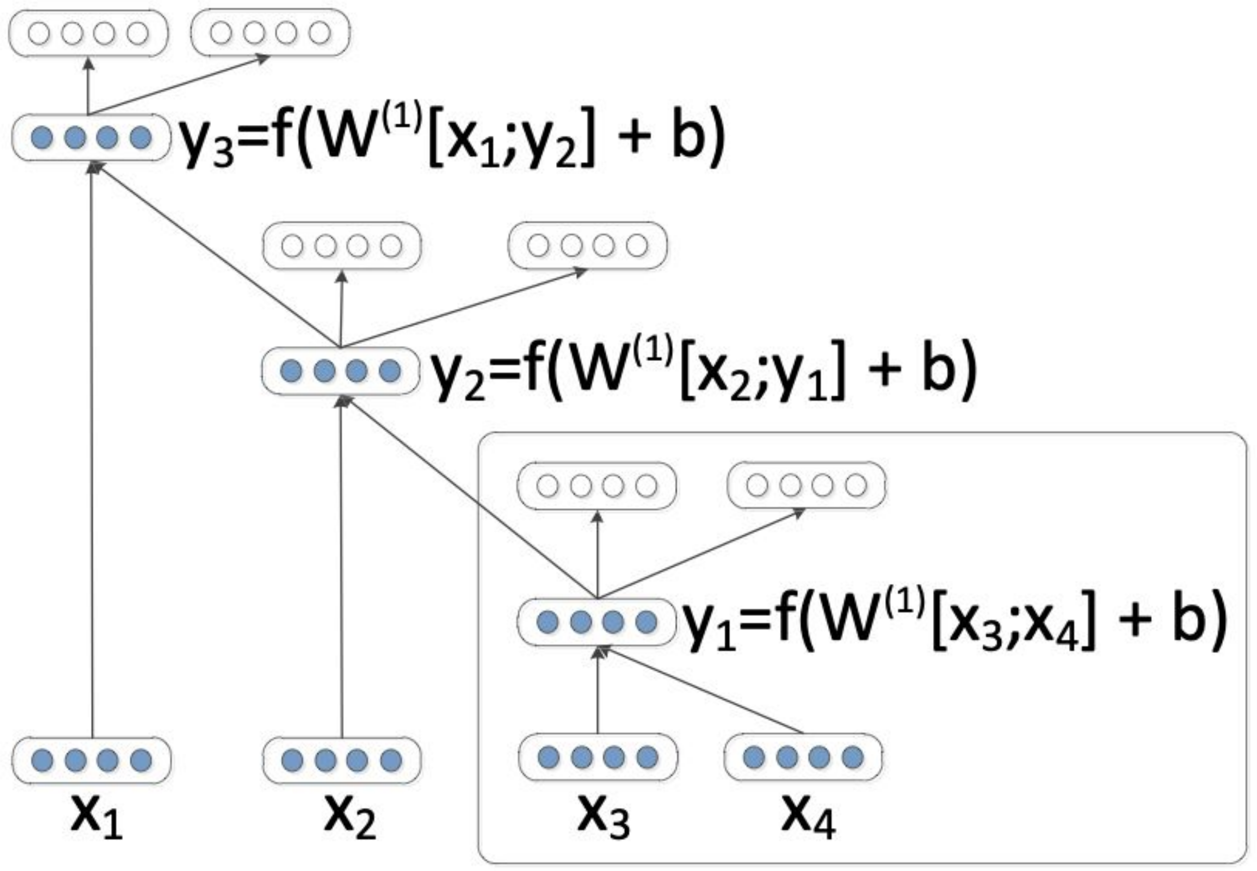
- Focused on compositional representation learning of hierarchical structure, features and predictions.

- Using the information about syntactic structure of the language helps to obtain better vector representation of the sentence, exploits hierarchical structure and uses compositional semantics to understand sentiment.

- All kind of parsed trees can be used to obtain the tree structure (constituency and dependency).

2

The Recursive Neural Network



$y_3=f(W^{(1)}[x_1;y_2] + b)$
$y_2=f(W^{(1)}[x_2;y_1] + b)$
$y_1=f(W^{(1)}[x_3;x_4] + b)$
x_1
x_2
x_3
x_4

Given a tree structure and list of word vectors $(\mathbf{x}_1, \ldots, \mathbf{x}_m)$

where *m* - the sentence length, $\mathbf{x}_i \in \mathbb{R}^n$

$$\underbrace{\mathbf{y}_1}_{\mathbf{y}_1 \in \mathbb{R}^n} = \mathbf{f}(\underbrace{\mathbf{W}_1}_{\mathbf{W}_1 \in \mathbb{R}^{n \times 2n}} \underbrace{[\mathbf{x}_3, \mathbf{x}_4]}_{[\cdot,\cdot] \in \mathbb{R}^{2n}} + \mathbf{b}),$$

$$\underbrace{\mathbf{y}_2}_{\mathbf{y}_2 \in \mathbb{R}^n} = \mathbf{f}(\underbrace{\mathbf{W}_1}_{\mathbf{W}_1 \in \mathbb{R}^{n \times 2n}} \underbrace{[\mathbf{x}_2, \mathbf{y}_1]}_{[\cdot,\cdot] \in \mathbb{R}^{2n}} + \mathbf{b}),$$

$$\mathbf{y}_m = \ldots$$

f - activation function (tanh), $[\cdot, \cdot]$ - concatenation.

adapted from Semi-Supervised Recursive Autoencoders for Predicting Sentiment Distributions, Richard Socher et al., EMNLP 2011

2

The Recursive Neural Network

In general, the parent representation ***p*** is computed with the help of the children***:***

$$\mathbf{p} = \mathbf{f}(\mathbf{W}_1[\mathbf{c}_1, \mathbf{c}_2] + \mathbf{b})$$

then we could predict the probability:

$$\mathbf{pred} = \mathrm{softmax}(\mathbf{W}^{label}\mathbf{p})$$

$\mathbf{W}^{label}$- the output weight matrix

Objective function: cross-entropy between predicted labels and true labels $E_{ce}(\mathbf{pred}, \mathbf{true})$.

② In addition to the objective function - autoencoders

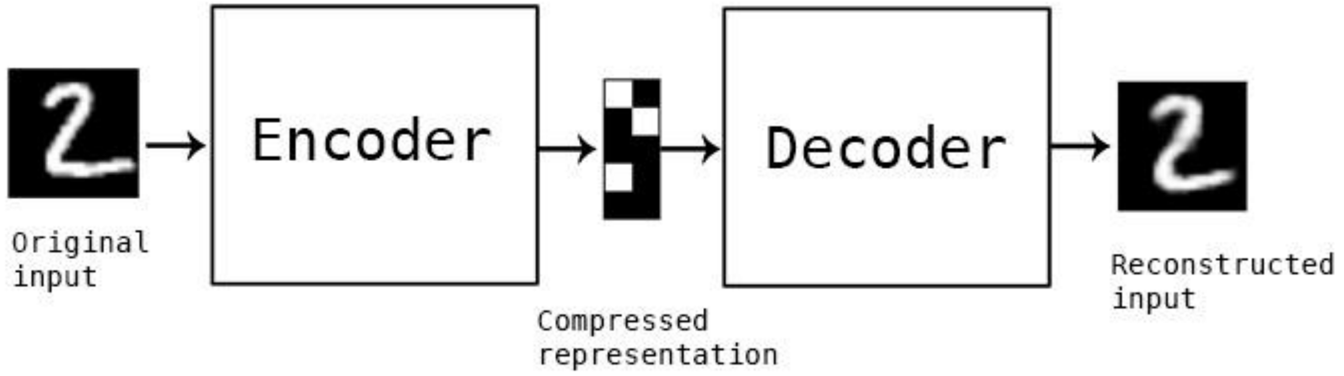
An autoencoder - the neural network for efficient dimensionality reduction in unsupervised manner.

$$\mathbf{X} \xrightarrow[\text{encode}]{\mathbf{f}} \mathbf{H} \xrightarrow[\text{decode}]{\mathbf{g}} \mathbf{X},$$

$$\mathbf{H} = \mathbf{f}(\mathbf{X}) = \sigma(\mathbf{W}_e\mathbf{X} + \mathbf{b}_e),$$

$\mathbf{r}(\mathbf{X}) = \mathbf{g}(\mathbf{f}(\mathbf{X})) = \sigma(\mathbf{W}_d\mathbf{H} + \mathbf{b}_d)$ — the reconstruction.

During training the reconstruction loss is minimized: $[\| \mathbf{r}(\mathbf{X}) - \mathbf{X} \|_2^2]$



2

In addition to the objective function - autoencoders

Unfolding Recursive Autoencoder

Encoder:

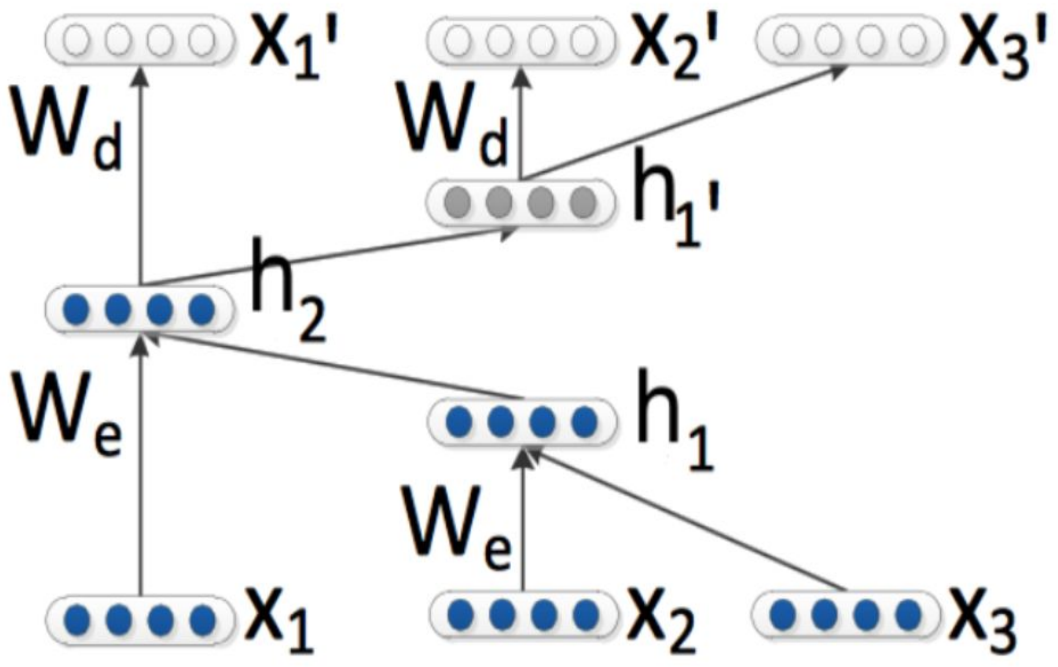
$$h_1 = \sigma(W_e[x_2, x_3] + b_e),$$

$$h_2 = \sigma(W_e[x_1, h_1] + b_e).$$

Decoder:

$$[x_1', h_1'] = \sigma(W_d h_2 + b_d),$$

$$[x_2', x_3'] = \sigma(W_d h_1' + b_d).$$



The reconstruction error: $E_{\text{RecNN}} = \| [x_1, x_2, x_3] - [x_1', x_2', x_3'] \|_2^2 .$

adapted from Dynamic Pooling and Unfolding Recursive Autoencoders for Paraphrase Detection, Richard Socher et al., NIPS 2011 and

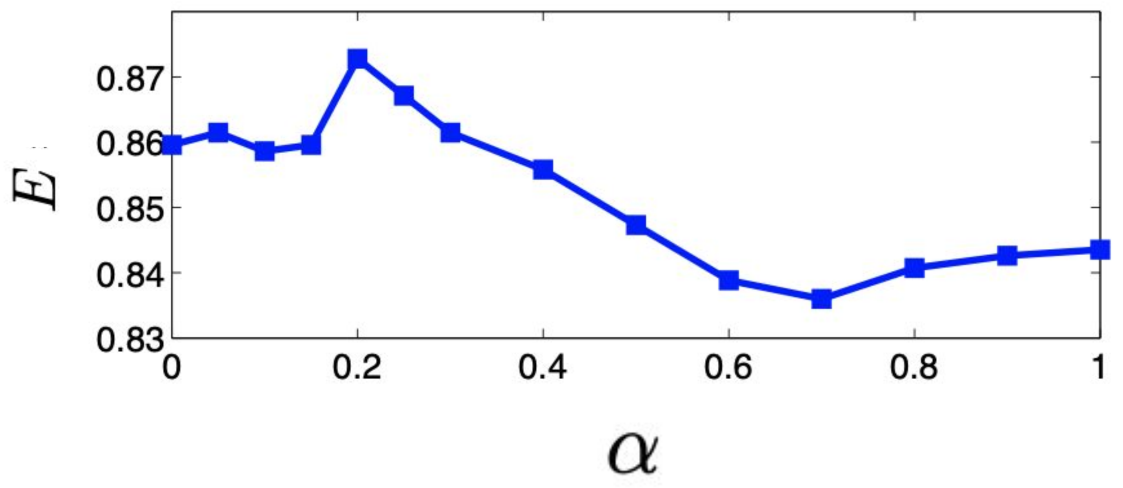
2 In addition to the objective function

The whole objective: $E = \alpha E_{recNN} + (1-\alpha) E_{ce}$

α - hyperparameter, that weighs reconstruction and cross-entropy error.

Why do we need the unsupervised part of the objective? How does it help in sentiment analysis or other classification tasks?

1. Prevents overfitting and helps to learn the language structure by itself;
2. Help to achieve the highest performance;
3. Other experiments showed that unsupervised training helps to catch syntactic and semantic information (see Socher, NIPS 2011).

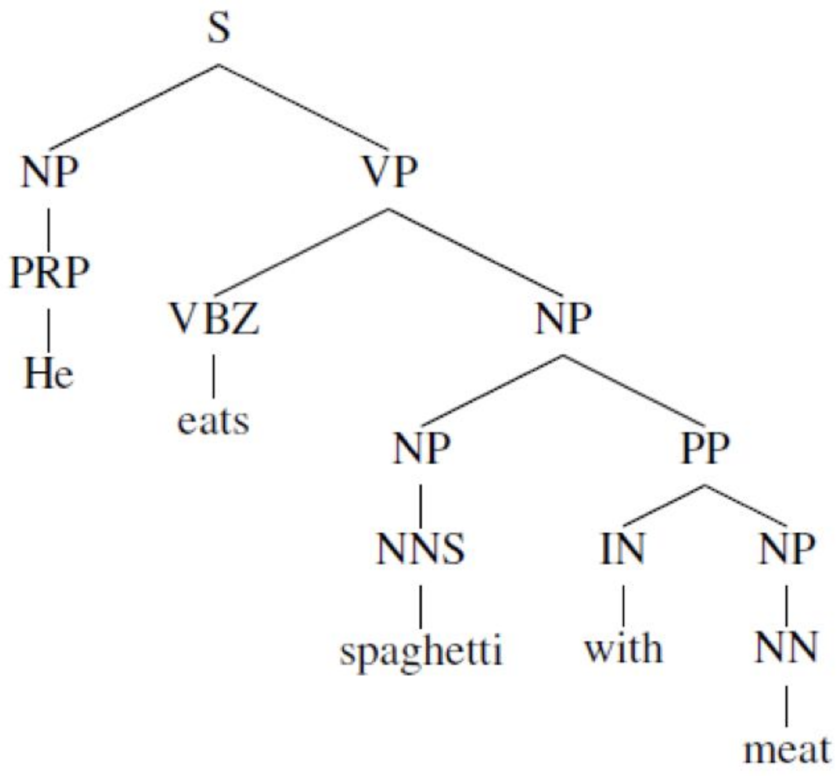


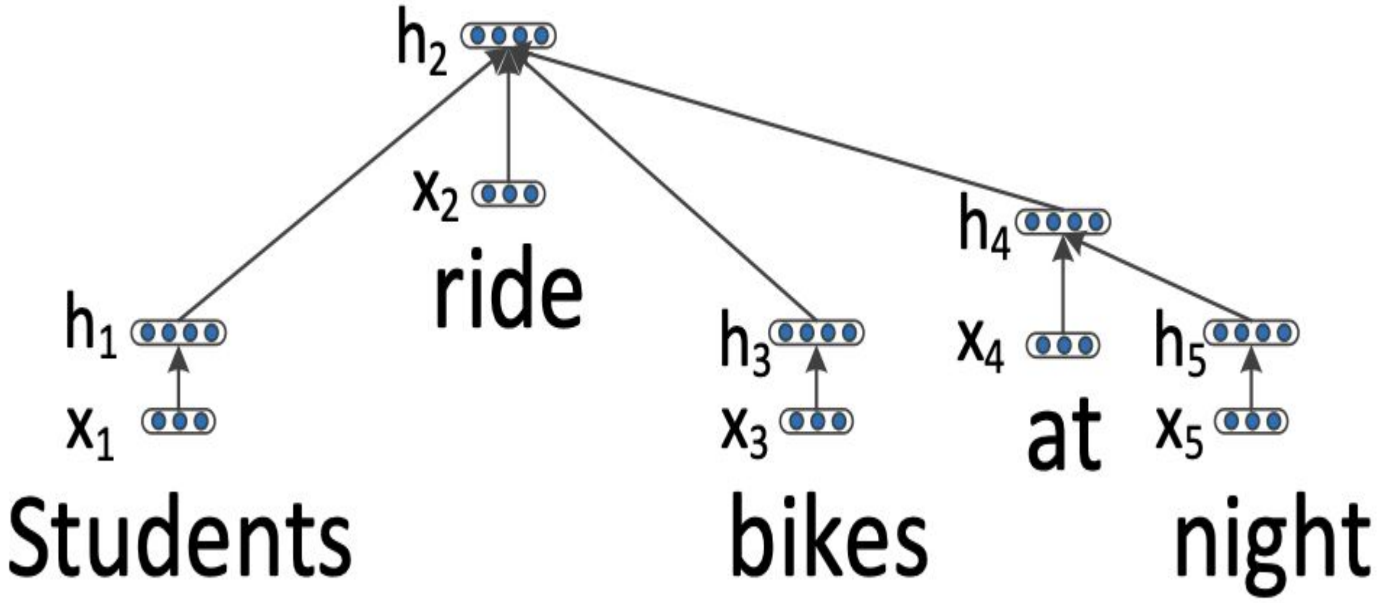
adapted from Semi-Supervised Recursive Autoencoders for Predicting Sentiment Distributions, Richard Socher et al., EMNLP 2011

2

How to choose the tree structure

- Using the trees, given by constituency and dependency parsers (depends on the task and what do you want to learn from the language):





adapted from Semi-Supervised Recursive Autoencoders for Predicting Sentiment Distributions, Richard Socher et al., EMNLP 2011

2

How to choose the tree structure

Greedy approximation that constructs such a tree (if no trees given):

1. For the sequence of *m* words $(\mathbf{x}_1, ..., \mathbf{x}_m)$ the autoencoder first takes the first pair of neighboring vectors $(\mathbf{x}_1, \mathbf{x}_2)$, concatenates them and gives them as input to the autoencoder.
2. Compute the reconstruction score for the first pair, save the score and the parent node.
3. Shift by one position and take pair $(\mathbf{x}_2, \mathbf{x}_3)$ and again compute a potential parent node and a score. This process repeats until it hits the last pair of words in the sentence (x_{m-1}, x_m) .
4. Select the pair with the lowest reconstruction score and its parent representation ***p*** will represent this phrase and replace both children in the sentence word list. Now we will have something like this (if tl $(\mathbf{x}_1, \mathbf{x}_2)$ r was ch $(\mathbf{p}, \mathbf{x}_3, ..., \mathbf{x}_m)$
5. The process repeats and treats the new vector ***p*** like any other input vector. The process finishes when final state will be achieved (all leaves were encoded in the one representation).

adapted from Semi-Supervised Recursive Autoencoders for Predicting Sentiment Distributions, Richard Socher et al., EMNLP 2011

Some nice examples

3

Results

Method	Movie Review	MPQA (opinions)
Voting with two lexica	63.1	81.7
Rule-based reversal on trees	62.9	82.8
Bag of features	76.4	84.1
RAE	**77.7**	**86.4**

Method	Stanford Sentiment Treebank
NB	81.8
SVM	79.4
VecAvg	80.1
Recursive Neural Net	**85.4**

3 To finalize

- The problem of this approach that it requires the given tree structure, and it could be tricky for many cases to obtain such structure.

- Parsers are quite slow, for some languages and grammars it is hard to find good parsers.

That is the reason why not all researchers use it for the representation learning and related classification tasks.

Fin