Effective Composition of Dense Features in Natural Language Processing

Xipeng Qiu
xpqiu@fudan.edu.cn
http://nlp.fudan.edu.cn/~xpqiu

Fudan University

August 25, 2015
CCIR 2015

Outline

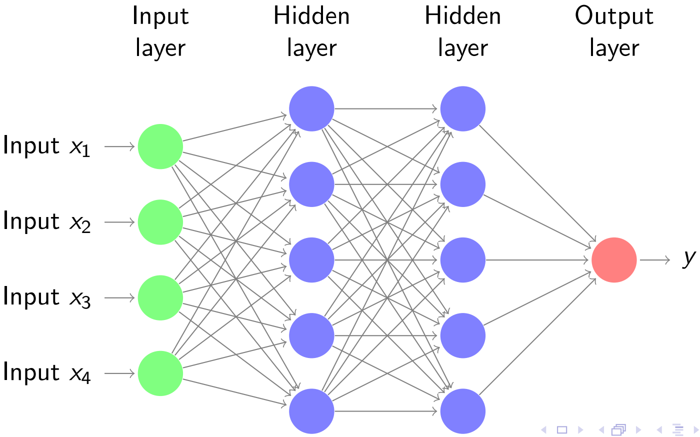
1. Basic Concepts
   - Machine Learning & Deep Learning
2. Neural Models for NLP
   - General Architecture
   - Various Models
   - Feature Composition Problems
3. Feature Compositions
   - Cube Activation Function
   - Neural Tensor Model
     - Multi-relational Data Embeddings
     - Neural Tensor Model
     - Training
     - Applications
   - Gated Recursive Neural Network
   - Attention Model

Basic Concepts of Machine Learning

- Input Data: (x_i, y_i),$1 \leq i \leq m$
- Model:
  - Linear Model: $y = f(x) = w^T x + b$
  - Generalized Linear Model: $y = f(x) = w^T \phi(x) + b$
  - Non-linear Model: Neural Network
- Criterion:
  - Loss Function:
    $L(y, f(x)) \rightarrow$ Optimization
  - Empirical Risk:
    $Q(\theta) = \frac{1}{m} \cdot \sum_{i=1}^{m} L(y_i, f(x_i, \theta)) \rightarrow$ Minimization
  - Regularization: $\|\theta\|^2$
- Objective Function: $Q(\theta) + \lambda \|\theta\|^2$

Basic Concepts of Deep Learning

- Model: Artificial Neural Network (ANN)
- Function: Non-linear function $y = \sigma(\sum_i w_i x_i + b)$



Traditional ML methods for NLP

Structured Learning

Structured Learning is the task of assigning a group of labels $\mathbf{y} = y_1, \ldots, y_n$ to a group of inputs $\mathbf{x} = x_1, \ldots, x_n$.
Given a sample $\mathbf{x}$, we define the feature $\Phi(\mathbf{x}, \mathbf{y})$. Thus, we can label $\mathbf{x}$ with a score function,

$$\hat{\mathbf{y}} = \arg\max_{\mathbf{y}} F(\mathbf{w}, \Phi(\mathbf{x}, \mathbf{y})), \tag{1}$$

where $\mathbf{w}$ is the parameter of function $F(\cdot)$.

General Neural Architectures for NLP

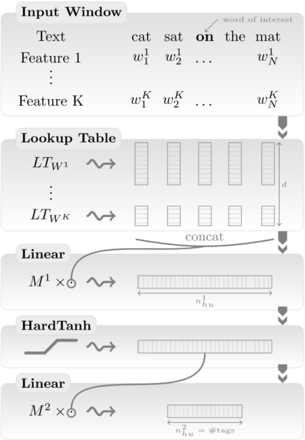
How to use neural network for the NLP tasks?

General Neural Architectures for NLP

How to use neural network for the NLP tasks?

Distributed Representation

General Neural Architectures for NLP



1. represent the words/features with dense vectors (embeddings) by lookup table;
2. concatenate the vectors;
3. classify/match/rank with multi-layer neural networks.

from [Collobert et al., 2011]

Difference with the traditional methods

	Traditional methods	Neural methods
Features	Discrete Vector (One-hot Representation)	Dense Vector (Distributed Representation)
	High-dimension	Low-dimension
Classifier	Linear	Non-Linear

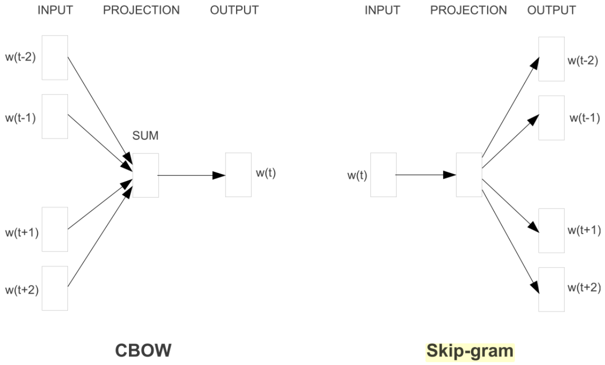
Beyond word/feature embeddings

Can we encode the phrase, sentence, paragraph, or even document into the distributed representation?

General Neural Architectures for NLP

- **Word Level**
  - NNLM
  - C&W
  - CBOW & Skip-Gram
- **Sentence Level**
  - **NBOW**
  - **Sequence Models**: Recurrent NN, LSTM, Paragraph Vector
  - **Topoligical Models**: Recursive NN,
  - **Convolutional Models**: DCNN
- **Document Level**
  - **NBOW**
  - **Hierachical Models** two-level CNN
  - **Sequence Models** LSTM, Paragraph Vector

Skip-Gram Model



from [Mikolov et al., 2013]

Skip-Gram Model

Given a pair of words (w, c), the probability that the word c is observed in the context of the target word w is given by

$$Pr(D = 1|w, c) = \frac{1}{1 + \exp(-\mathbf{w}^T\mathbf{c})},$$

where **w** and **c** are embedding vectors of w and c respectively. The probability of not observing word c in the context of w is given by,

$$Pr(D = 0|w, c) = 1 - \frac{1}{1 + \exp(-\mathbf{w}^T\mathbf{c})}.$$

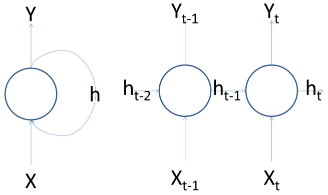
Skip-Gram Model with Negative Sampling

Given a training set $\mathcal{D}$, the word embeddings are learned by maximizing the following objective function:

$$J(\theta) = \sum_{w,c \in \mathcal{D}} Pr(D = 1|w, c) + \sum_{w,c \in \mathcal{D}'} Pr(D = 0|w, c),$$

where the set $\mathcal{D}'$ is randomly sampled negative examples, assuming they are all incorrect.

Recurrent Neural Network (RNN) [Elman, 1990]

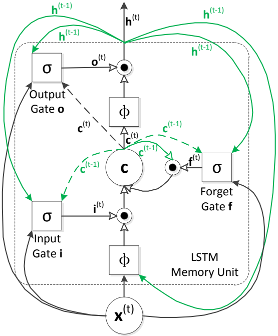


The RNN has recurrent hidden states whose output at each time is dependent on that of the previous time.

More formally, given a sequence $\mathbf{x}^{(1:n)} = (\mathbf{x}^{(1)}, \ldots, \mathbf{x}^{(t)}, \ldots, \mathbf{x}^{(n)})$, the RNN updates its recurrent hidden state $\mathbf{h}^{(t)}$ by

$$\mathbf{h}^{(t)} = \mathbf{g}(\mathbf{U}\mathbf{h}^{(t-1)} + \mathbf{W}\mathbf{x}^{(t)} + \mathbf{b}),$$

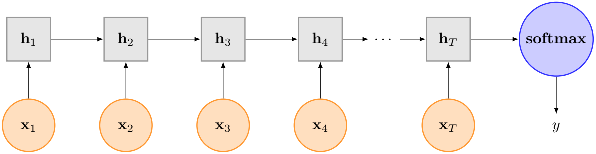
Long Short Term Memory Neural Network (LSTM) [Hochreiter and Schmidhuber, 1997]



The core of the LSTM model is a memory cell **c** encoding memory at every time step of what inputs have been observed up to this step.
The behavior of the cell is controlled by three "gates":

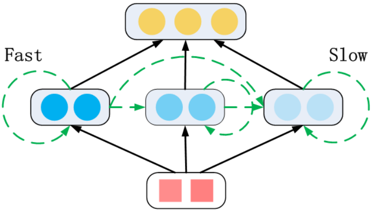
- input gate **i**
- output gate **o**
- forget gate **f**

Unfolded LSTM for Text Classification



Drawback: long-term dependencies need to be transmitted one-by-one along the sequence.

Multi-Timescale LSTM

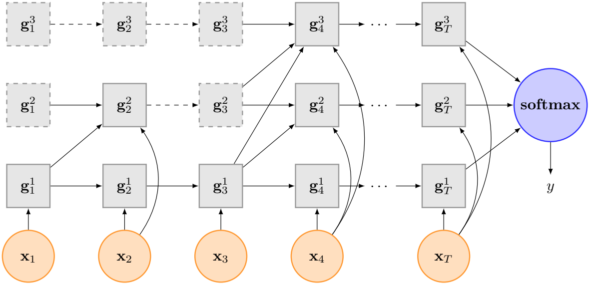


(a) Fast-to-Slow Strategy (b) Slow-to-Fast Strategy

Figure: Two feedback strategies of our model. The dashed line shows the feedback connection, and the solid link shows the connection at current time.

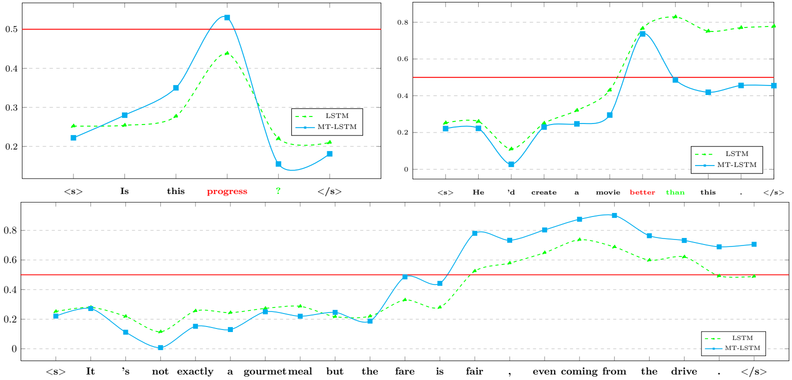
from [Liu et al., 2015a]

Unfolded Multi-Timescale LSTM with Fast-to-Slow Feedback Strategy

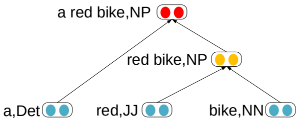


from [Liu et al., 2015a]

LSTM for Sentiment Analysis



Recursive Neural Network (RecNN) [Socher et al., 2013]



Topological models compose the sentence representation following a given topological structure over the words.

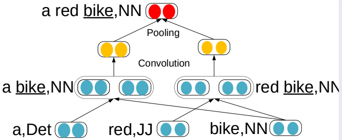
Given a labeled binary parse tree, $((p_2 \rightarrow ap_1), (p_1 \rightarrow bc))$, the node representations are computed by

$$\mathbf{p}_1 = f(\mathbf{W}\begin{bmatrix} \mathbf{b} \\ \mathbf{c} \end{bmatrix}),$$

$$\mathbf{p}_2 = f(\mathbf{W}\begin{bmatrix} \mathbf{a} \\ \mathbf{p}_1 \end{bmatrix}).$$

A variant of RecNN for Dependency Parse Tree [Zhu et al., 2015]

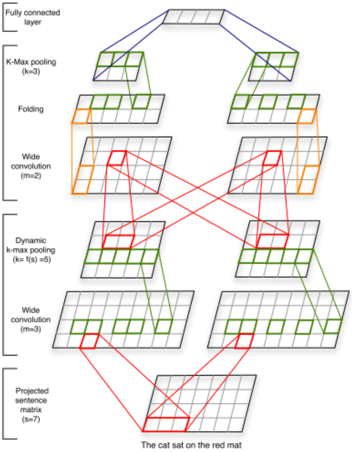
Recursive neural network can only process the binary combination and is not suitable for dependency parsing.



Recursive Convolutional Neural Network

- introducing the convolution and pooling layers;
- modeling the complicated interactions of the head word and its children.

Convolutional Neural Network (CNN)



Key steps

- Convolution
- (optional) Folding
- Pooling

Various models

- DCNN (k-max pooling) [Kalchbrenner et al., 2014]
- CNN (binary pooling) [Hu et al., 2014]
- ...

Overview of state-of-the-art neural models in NLP

Not "Really" Deep Learning in NLP

- Most of the neural models is very shallow in NLP.
- The major benefit is introducing dense representation.
- The feature composition is also quite simple.
  - Concatenation
  - Sum/Average
  - Bilinear model

Quite Simple Feature Composition

Given two embeddings **a** and **b**,

1. how to calculate their similarity/relevence/relation?
   1. Concatenation
   $$\mathbf{a} \oplus \mathbf{b} \rightarrow \text{ANN} \rightarrow \text{output}$$
   2. Bilinear
   $$\mathbf{a}^T \mathbf{M} \mathbf{b} \rightarrow \text{output}$$
2. how to use them in classification task?
   1. Concatenation
   $$\mathbf{a} \oplus \mathbf{b} \rightarrow \text{ANN} \rightarrow \text{output}$$
   2. Sum/Average
   $$\mathbf{a} + \mathbf{b} \rightarrow \text{ANN} \rightarrow \text{output}$$

Problem

How to enhance the neural model without increasing the network depth?

Solution 1: Cube Activation Function [Chen and Manning, 2014]

How to enhance the neural model without increasing the network depth?

A simple solution: Cube Activation Function

Suppose $\mathbf{x} = \mathbf{a} \oplus \mathbf{b}$,

$$\begin{aligned}\mathbf{f}(\mathbf{wx} + b) &= (w_1 x_1 + \cdots + w_m x_m + b)^3 \\ &= \sum_{i,j,k} (w_i w_j w_k) x_i x_j x_k + \sum_{i,j} b(w_i w_j) x_i x_j + \cdots\end{aligned}$$

Solution 2: Neural Tensor Model [Chen et al., 2013]

How to enhance the neural model without increasing the network depth?

Another intuitive solution: Neural Tensor Model

$$s(\mathbf{a}, \mathbf{b}) = \mathbf{u}^T \mathbf{f}(\mathbf{a}^T \mathbf{W}^{[1:k]} \mathbf{b} + \mathbf{V}^T (\mathbf{a} \oplus \mathbf{b}) + \mathbf{c})$$

Neural Tensor Model

The original Neural Tensor Model is proposed to model the multi-relational data embeddings [Chen et al., 2013] .

Multi-relational Data Embeddings

Multi-relational Data (e_1, e_2, r): a pair of entities (e_1, e_2) and their relation mention r.

The basic idea is that, the relationship between two entities corresponds to a translation between the embeddings of entities, that is, $\mathbf{e}_1 + \mathbf{r} \approx \mathbf{e}_2$ when (e_1, e_2, r) holds.

- the embeddings $\mathbf{r}$ of relation mention r
- the entity embeddings $\mathbf{e}_1$, $\mathbf{e}_2$ of the entity pair e_1, e_2

We can get them by a lookup table respectively.

Neural Tensor Model [Chen et al., 2013]

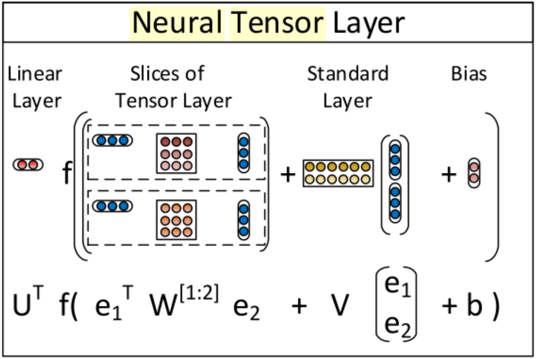
Neural Tensor Model gives a score function $s(\mathbf{e}_1, \mathbf{e}_2, \mathbf{r})$ to model the triplet (e_1, e_2, r) as followed:

$$s(e_1, e_2, r) = \mathbf{u}^T f(\mathbf{e}_1^T \mathbf{W}_r^{[1:k]} \mathbf{e}_2 + \mathbf{V}_r^T (\mathbf{e}_1 \oplus \mathbf{e}_2) + \mathbf{b}_r),$$

where $\mathbf{W}_r^{[1:k]} \in \mathbb{R}^{d \times d \times k}$ is a tensor, and the bilinear tensor product takes two vectors $\mathbf{e}_1$ and $\mathbf{e}_2$ as input, and generates a k-dimensional phrase vector $\mathbf{z}$ as output.

$$\mathbf{z} = \mathbf{e}_1^T \mathbf{M}_r^{[1:k]} \mathbf{e}_2$$

Neural Tensor Model



from [Chen et al., 2013]

Training

The objective function is based on the idea that the similarity scores of observed triples in the training set should be larger than those of any other triples.

$$\mathscr{L} = \sum_{(\mathbf{e}_1,\mathbf{e}_2,\mathbf{r})\in\mathcal{D}} \sum_{(\mathbf{e}'_1,\mathbf{e}'_2,\mathbf{r})\in\mathcal{D}'} [\gamma - s(\mathbf{e}_1,\mathbf{e}_2,\mathbf{r}) + s(\mathbf{e}'_1,\mathbf{e}'_2,\mathbf{r})]_+$$

where $[x]_+$ denotes the positive part of x; $\gamma > 0$ is a margin hyper-parameter; D is all triplet extracted from the plain text; D' denotes all the corrupted triplets, which is composed of training triplets with either the e_1 or e_2 replaced by a random entity (but not both at the same time).

Tensor Factorization

The tensor operation complexity is $O(d^2k)$. To remedy this, we use a tensor factorization approach that factorizes each tensor slice as the product of two low-rank matrices. Formally, each tensor slice $\mathbf{M}^{[i]} \in \mathbb{R}^{d \times d}$ is factorized into two low rank matrix $\mathbf{P}^{[i]} \in \mathbb{R}^{d \times r}$ and $\mathbf{Q}^{[i]} \in \mathbb{R}^{r \times d}$:

$$\mathbf{M}^{[i]} = \mathbf{P}^{[i]}\mathbf{Q}^{[i]}, 1 \leq i \leq k$$

where $r \ll d$ is the number of factors.

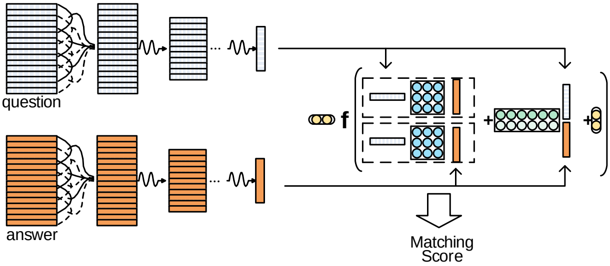
$$g(w, c, t) = \mathbf{u}^T f(\mathbf{w}^T \mathbf{P}^{[1:k]}\mathbf{Q}^{[1:k]}\mathbf{t} + \mathbf{V}_c^T(\mathbf{w} \oplus \mathbf{t}) + \mathbf{b}_c).$$

The complexity of the tensor operation is now $O(rdk)$.

Applications

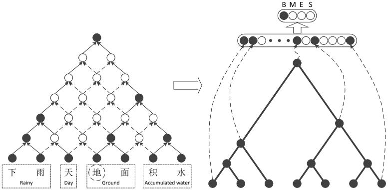
- Parsing [Socher et al., 2012] [Socher et al., 2013]
- Chinese Word Segmentation [Pei et al., 2014]
- Semantic Composition [Zhao et al., 2015b]
- Sentence Matching [Qiu and Huang, 2015]
- Context-Sensitive Word Embeddings [Liu et al., 2015b]
- ...

Convolutional Neural Tensor Network for CQA [Qiu and Huang, 2015]



Architecture of Convolutional Neural Tensor Network

Solution 3: Gated Recursive Neural Network [Chen et al., 2015a]



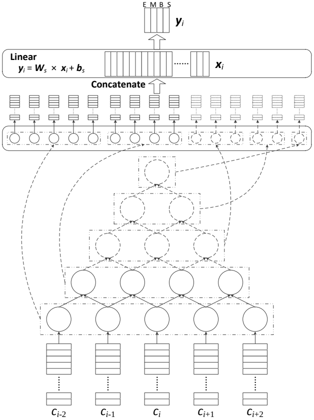
- DAG based Recursive Neural Network
- Gating mechanism

An relative complicated solution

GRNN models the complicated combinations of the features, which selects and preserves the useful combinations via reset and update gates.

A similar model: AdaSent [Zhao et al., 2015a]

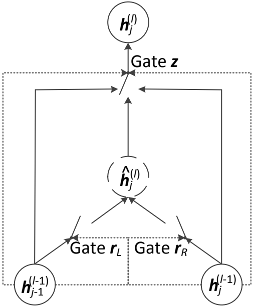
Gated Recursive Neural Network



The RecNN needs a topological structure to model a sequence, such as a syntactic tree.

GRNN uses a directed acyclic graph (DAG) to model the combinations of the input characters, in which two consecutive nodes in the lower layer are combined into a single node in the upper layer.

GRNN Unit



Two Gates

- reset gate
- update gate

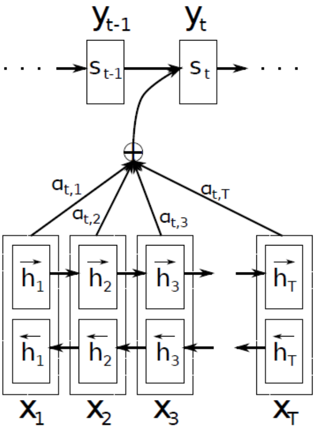
The new activation $\hat{\mathbf{h}}_j^l$ is computed as:

$$\hat{\mathbf{h}}_j^l = \tanh(\mathbf{W}_{\hat{\mathbf{h}}} \begin{bmatrix} \mathbf{r}_L \odot \mathbf{h}_{j-1}^{l-1} \\ \mathbf{r}_R \odot \mathbf{h}_j^{l-1} \end{bmatrix}).$$

Applications

- Chinese Word Segmentation [Chen et al., 2015a]
- Dependency Parsing [Chen et al., 2015c]
- Sentence Modeling [Chen et al., 2015b]
- ...

Attention Model



from [Bahdanau et al., 2014]

Similar to gate mechanism!

1. The context vector c_i is computed as a weighted sum of h_i:

$$c_i = \sum_{j=1}^{T} \alpha_{ij} h_j$$

2. The weight α_{ij} is computed by

$$\alpha_{ij} = \mathrm{softmax}(a(s_{i-1}; h_j))$$

Applications

- Machine Translation [Bahdanau et al., 2014, Meng et al., 2015]
- Speech Recognition [Chan et al., 2015]
- ...

Conclusion

Four intuitive ways to model the combination the distributed features (dense vectors)?

- Cube Activation Function
- Neural Tensor Model
- Gated Recursive Neural Network
- Attention Model

Better models?

References I

D. Bahdanau, K. Cho, and Y. Bengio. Neural machine translation by jointly learning to align and translate. *ArXiv e-prints*, September 2014.

W. Chan, N. Jaitly, Q. V. Le, and O. Vinyals. Listen, Attend and Spell. *ArXiv e-prints*, August 2015.

Danqi Chen and Christopher D Manning. A fast and accurate dependency parser using neural networks. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 740–750, 2014.

Danqi Chen, Richard Socher, Christopher D Manning, and Andrew Y Ng. Learning new facts from knowledge bases with neural tensor networks and semantic word vectors. *arXiv preprint arXiv:1301.3618*, 2013.

Xinchi Chen, Xipeng Qiu, Chenxi Zhu, and Xuanjing Huang. Gated recursive neural network for Chinese word segmentation. In *Proceedings of Annual Meeting of the Association for Computational Linguistics*, 2015a.

References II

Xinchi Chen, Xipeng Qiu, Chenxi Zhu, Shiyu Wu, and Xuanjing Huang. Sentence modeling with gated recursive neural network. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, 2015b.

Xinchi Chen, Yaqian Zhou, Chenxi Zhu, Xipeng Qiu, and Xuanjing Huang. Transition-based dependency parsing using two heterogeneous gated recursive neural networks. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, 2015c.

Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. Natural language processing (almost) from scratch. *The Journal of Machine Learning Research*, 12:2493–2537, 2011.

Jeffrey L Elman. Finding structure in time. *Cognitive science*, 14(2):179–211, 1990.

Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

References III

Baotian Hu, Zhengdong Lu, Hang Li, and Qingcai Chen. Convolutional neural network architectures for matching natural language sentences. In *Advances in Neural Information Processing Systems*, 2014.

Nal Kalchbrenner, Edward Grefenstette, and Phil Blunsom. A convolutional neural network for modelling sentences. In *Proceedings of ACL*, 2014.

PengFei Liu, Xipeng Qiu, Xinchi Chen, Shiyu Wu, and Xuanjing Huang. Multi-timescale long short-term memory neural network for modelling sentences and documents. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, 2015a.

PengFei Liu, Xipeng Qiu, and Xuanjing Huang. Learning context-sensitive word embeddings with neural tensor skip-gram model. In *Proceedings of International Joint Conference on Artificial Intelligence*, 2015b.

F. Meng, Z. Lu, Z. Tu, H. Li, and Q. Liu. Neural Transformation Machine: A New Architecture for Sequence-to-Sequence Learning. *ArXiv e-prints*, June 2015.

References IV

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*, 2013.

Wenzhe Pei, Tao Ge, and Chang Baobao. Maxmargin tensor neural network for chinese word segmentation. In *Proceedings of ACL*, 2014.

Xipeng Qiu and Xuanjing Huang. Convolutional neural tensor network architecture for community-based question answering. In *Proceedings of International Joint Conference on Artificial Intelligence*, 2015.

Richard Socher, Brody Huval, Christopher D Manning, and Andrew Y Ng. Semantic compositionality through recursive matrix-vector spaces. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 1201–1211, 2012.

Richard Socher, John Bauer, Christopher D Manning, and Andrew Y Ng. Parsing with compositional vector grammars. In *In Proceedings of the ACL conference*. Citeseer, 2013.

References V

Han Zhao, Zhengdong Lu, and Pascal Poupart. Self-adaptive hierarchical sentence model. *arXiv preprint arXiv:1504.05070*, 2015a.

Yu Zhao, Zhiyuan Liu, and Maosong Sun. Phrase type sensitive tensor indexing model for semantic composition. In *AAAI*, 2015b.

Chenxi Zhu, Xipeng Qiu, Xinchi Chen, and Xuanjing Huang. A re-ranking model for dependency parser with recursive convolutional neural network. In *Proceedings of Annual Meeting of the Association for Computational Linguistics*, 2015.