Neural Word Embeddings from Scratch

Xin Li$^{1,2}$

$^1$NLP Center, Tencent AI Lab

$^2$Dept. of System Engineering & Engineering Management, The Chinese University of Hong Kong

2018-04-09
Outline

1. What is Word Embedding?

2. Neural Word Embeddings Revisit
   - Classical NLM
   - Word2Vec
   - GloVe

3. Bridging Skip-Gram and Matrix Factorization
   - SG-NS as Implicit Matrix Factorization
   - SVD over shifted PPMI matrix

4. Advanced Techniques for Learning Word Representations
   - General-Purpose Word Representations–By Ziyi
   - Task-Specific Word Representations–By Deng
What is Word Embedding?

- Word Embedding refers to low dimensional, real-valued dense vectors encoding the semantic information of word.
- Generally, the concepts **Word Embeddings, Distributed Word Representations** and **Dense Word Vectors** can be used interchangeably.

---

**John Rupert Firth (linguist)**

“You shall know a **word** by the company it keeps”.

**Karl Marx (philosopher)**

“The **human essence** is no abstraction inherent in each single individual. *In its reality it is the ensemble of the social relations*.”
What is Word Embedding?

Word Embedding is the by-product of neural language model.

- Definition of language model:

\[
p(w_1:T) = \prod_{t=1}^{T} p(w_t|w_1:t-1)
\]

- Neural Language Model (NLM) is the language model where the conditional probability is modeled by neural networks.
Outline

1. What is Word Embedding?

2. Neural Word Embeddings Revisit
   - Classical NLM
   - Word2Vec
   - GloVe

3. Bridging Skip-Gram and Matrix Factorization
   - SG-NS as Implicit Matrix Factorization
   - SVD over shifted PPMI matrix

4. Advanced Techniques for Learning Word Representations
   - General-Purpose Word Representations—By Ziyi
   - Task-Specific Word Representations—By Deng
Training objective is to maximize the log-likelihood:

\[
L = \frac{1}{T} \sum \log[p(w_t|w_1:t-n+1)]
\]

**Figure:** MLP-LM. n is set to 3.
Conv-LM
Collobert and Weston., ICML 2008

Figure: Conv-LM. $x_t$ denotes the middle word of $S$. $S^{x'}$ is obtained by replacing middle word of $S$ with $x'$.

Training objective is to minimize the rank-type loss:

$$ L = \sum_{S \in \mathcal{D}} \sum_{x' \in \mathcal{V}} \max(0, 1 - p(y = 1|S) + p(y = 1|S^{x'})) $$
Training objective is same to MLP-LM (i.e., maximum likelihood).

\[ p(w_t|w_1, \ldots, w_{t-1}) = \text{softmax}(P_h_{t-1} + b) \]

\[ h_{t-3} = f(Wx_{t-1} + Uh_{t-2} + b) \]

**Figure:** RNN-LM.
Outline

1. What is Word Embedding?

2. Neural Word Embeddings Revisit
   - Classical NLM
   - Word2Vec
   - GloVe

3. Bridging Skip-Gram and Matrix Factorization
   - SG-NS as Implicit Matrix Factorization
   - SVD over shifted PPMI matrix

4. Advanced Techniques for Learning Word Representations
   - General-Purpose Word Representations—By Ziyi
   - Task-Specific Word Representations—By Deng
Word2Vec involves two different models, namely, **CBOV** and **SG**:

1. **CBOV** (Continuous Bag-of-Words): Using context words to predict the middle word.
2. **SG** (Skip-Gram): Using middle word to predict the context words.
The main feature of Word2Vec (IMO) is that it is a non-MLE framework, i.e., its aim is not to model the joint probability of the input words.

**CBOW:** \[ L = \frac{1}{T} \sum_{t=1}^{T} \log[p(w_t|w_{t-c:t-1}, w_{t+1:t+c})] \]

**SG:** \[ L = \frac{1}{T} \sum_{t=1}^{T} \sum_{-c \leq j \leq c} \log[p(w_{t+j}|w_t)] \]

Word2Vec is the first model for learning word embeddings from unlabeled data!!!
Word2Vec
(Mikolov et al., ICLR 2013 & NIPS 2013)

Some extensions for improving Word2Vec:

- **HSoftmax** (Hierarchical Softmax)
  1. Full softmax layer is too “fat” since it needs to evaluate $|\mathcal{V}|$ (generally more than 1M in the large corpus) output nodes.
  2. **HSoftmax** takes advantage of binary tree representation of output layer and only needs to evaluate $\log_2(|\mathcal{V}|)$ nodes.

\[
p(w_2|w_i) = \prod_{j=1}^{3} \sigma(\mathbb{I}(n(w_2, j + 1) = ch(n(w_2, j)))v_{n(w_2, j)}^\top x_i)
\]

**Figure:** Binary Tree

**Figure:** Huffman Tree example

Mikolov uses **Huffman Tree** to construct the hierarchical structure.
Word2Vec
(Mikolov et al., ICLR 2013 & NIPS 2013)

- **NS** (Negative Sampling) is another alternative for speeding up training.
  1. Formulating $|\mathcal{V}|$-class classification problem as a binary classification problem. ("word prediction" $\implies$ "co-occurrence relation prediction")
  2. Training with $k$ additional corrupted samples for each positive sample.

$$L = \log \sigma(x_O^\top x_I) + \sum_{i=1}^{k} \mathbb{E}_{w^* \sim P_n(w)}[\log \sigma(-x_i^*\top x_I)]$$

- **Subsampling**: Most frequent words usually provide less information and randomly discarding them should speedup training and improve performance.
Outline

1. What is Word Embedding?

2. Neural Word Embeddings Revisit
   - Classical NLM
   - Word2Vec
   - GloVe

3. Bridging Skip-Gram and Matrix Factorization
   - SG-NS as Implicit Matrix Factorization
   - SVD over shifted PPMI matrix

4. Advanced Techniques for Learning Word Representations
   - General-Purpose Word Representations—By Ziyi
   - Task-Specific Word Representations—By Deng
Motivations of GloVe:

- Global co-occurrence count is the primary information for generating word embeddings. (Word2Vec ignores this kind of information)
- Only using co-occurrence information is not enough to distinguish relevant words from irrelevant words.

<table>
<thead>
<tr>
<th>Probability and Ratio</th>
<th>$k = \text{solid}$</th>
<th>$k = \text{gas}$</th>
<th>$k = \text{water}$</th>
<th>$k = \text{fashion}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$P(k</td>
<td>\text{ice})$</td>
<td>$1.9 \times 10^{-4}$</td>
<td>$6.6 \times 10^{-5}$</td>
<td>$3.0 \times 10^{-3}$</td>
</tr>
<tr>
<td>$P(k</td>
<td>\text{steam})$</td>
<td>$2.2 \times 10^{-5}$</td>
<td>$7.8 \times 10^{-4}$</td>
<td>$2.2 \times 10^{-3}$</td>
</tr>
<tr>
<td>$P(k</td>
<td>\text{ice})/P(k</td>
<td>\text{steam})$</td>
<td>8.9</td>
<td>$8.5 \times 10^{-2}$</td>
</tr>
</tbody>
</table>

The appropriate starting point for learning word vectors should be ratios of co-occurrence probabilities!!!


\[ F(x_i, x_j, \hat{x}_k) = p_{ik}/p_{jk} \]

\[ \implies F(x_i - x_j, \hat{x}_k) = p_{ik}/p_{jk} \]

\[ \implies F((x_i - x_j)^\top \hat{x}_k) = p_{ik}/p_{jk} \]

(1)

The roles of word and context word should be exchangeable, thus:

\[ F((x_i - x_j)^\top \hat{x}_k) = F(x_i^\top \hat{x}_k)/F(x_j^\top \hat{x}_k) \]

(2)

According to (1) and (2):

\[ F(x_i^\top \hat{x}_k) = p_{ik} \implies x_i^\top \hat{x}_k = \log(C_{ik}) - \log(C_i) \]

(3)

\[ \implies x_i^\top \hat{x}_k + b_i + \hat{b}_k = \log(C_{ik}) \]

Training objective: \[ J = \sum_{i, k=1}^{|

V|} f(C_{ik})(x_i^\top \hat{x}_k + b_i + \hat{b}_k - \log(C_{ik}))^2 \]

(4)

where \( f(C_{ik}) \) is a weight function to filter the noise from rare co-occurrences.
Outline

1. What is Word Embedding?

2. Neural Word Embeddings Revisit
   - Classical NLM
   - Word2Vec
   - GloVe

3. Bridging Skip-Gram and Matrix Factorization
   - SG-NS as Implicit Matrix Factorization
   - SVD over shifted PPMI matrix

4. Advanced Techniques for Learning Word Representations
   - General-Purpose Word Representations—By Ziyi
   - Task-Specific Word Representations—By Deng
Skip-Gram with Negative Sampling (SG-NS) can be efficiently trained and achieve state-of-the-art results.

The outputs of SG-NS are word embeddings $X^w$ and context word embeddings $X^c$ (ignored).

**Figure**: SG-NS

**Figure**: SVD for $d$-rank factorization
SG-NS as Implicit Matrix Factorization
Levy and Goldberg, NIPS 2014 & TACL 2015

Training objective of SG-NS:

\[
\ell(I, O) = \log \sigma(x_O^* \top x_I) + \sum_{i=1}^{k} \mathbb{E}_{w_i^* \sim p_n(w)}[\log \sigma(-x_i^* \top x_I)]
\]

\[
L = \sum_{I \in \mathcal{V}} \sum_{O \in \mathcal{V}_c} C_{IO} \ell(I, O)
\]

The optimal value is attained at:

\[
y = x_O^* \top x_I = \log\left(\frac{C_{IO} \star |\mathcal{D}|}{C_I \star C_O}\right) - \log k
\]

The first item is the point-wise mutual information (PMI) of word pair \((I, O)\).
The matrix $\mathbf{M}^{\text{PMI}_k}$, called “shifted PMI Matrix”, emerges as the optimal solution for SG-NS’s objective. Each cell of the matrix is defined below:

$$M_{ij}^{\text{PMI}_k} = X_i^w \cdot X_j^c = x_i \cdot x_j^* = \text{PMI}(i, j) - \log k$$

The objective of SG-NS can be regarded as a weighted matrix factorization problem over $\mathbf{M}^{\text{PMI}_k}$.

The matrices $\mathbf{M}_0^{\text{PMI}_k}$ and $\mathbf{M}^{\text{PPMI}_k}$ can be better alternatives of $\mathbf{M}^{\text{PMI}_k}$.

$$(\mathbf{M}_0^{\text{PMI}_k})_{ij} = \begin{cases} 0 & \text{if } C_{ij} = 0 \\ M_{ij}^{\text{PMI}_k} & \text{Otherwise} \end{cases}$$

$$M_{ij}^{\text{PPMI}_k} = \text{PPMI}(i, j) - \log k$$

$$\text{PPMI}(i, j) = \max(\text{PMI}(i, j), 0)$$
Outline

1. What is Word Embedding?

2. Neural Word Embeddings Revisit
   - Classical NLM
   - Word2Vec
   - GloVe

3. Bridging Skip-Gram and Matrix Factorization
   - SG-NS as Implicit Matrix Factorization
   - SVD over shifted PPMI matrix

4. Advanced Techniques for Learning Word Representations
   - General-Purpose Word Representations–By Ziyi
   - Task-Specific Word Representations–By Deng
SVD over shifted PPMI matrix

Levy and Goldberg, NIPS 2014 & TACL 2015

- **shifted** PPMI matrix:
  \[ M^{SPPMI_k} = SPPMI_k(i, j), \text{ where } SPPMI_k(i, j) = \max(PMI(i, j) - \log k, 0) \]

- Performing SVD over \( M^{SPPMI_k} \) and \( U \cdot \sqrt{\Sigma} \) is treated as word representations \( X^w \).
  - This method outperforms SG-NS on word similarity task!!!
Outline

1. What is Word Embedding?

2. Neural Word Embeddings Revisit
   - Classical NLM
   - Word2Vec
   - GloVe

3. Bridging Skip-Gram and Matrix Factorization
   - SG-NS as Implicit Matrix Factorization
   - SVD over shifted PPMI matrix

4. Advanced Techniques for Learning Word Representations
   - General-Purpose Word Representations—By Ziyi
   - Task-Specific Word Representations—By Deng
Outline

1. What is Word Embedding?

2. Neural Word Embeddings Revisit
   - Classical NLM
   - Word2Vec
   - GloVe

3. Bridging Skip-Gram and Matrix Factorization
   - SG-NS as Implicit Matrix Factorization
   - SVD over shifted PPMI matrix

4. Advanced Techniques for Learning Word Representations
   - General-Purpose Word Representations—By Ziyi
   - Task-Specific Word Representations—By Deng