Learning Word Embeddings for Low-resource Languages by PU Learning

Chao Jiang, Hsiang-Fu Yu, Cho-Jui Hsieh, Kai-Wei Chang
Word Embeddings are useful

• Many successful stories
  • Named entity recognition
  • Document ranking
  • Sentiment analysis
  • Question answering
  • Image captioning

• Pre-trained word vectors have been widely used
  • GloVe [Pennington+14]: 3900+ citations
  • Word2Vec [Mikolov+13]: 7600+ citations
Existing English Embeddings are trained on a large collection of text

- Word2Vec is trained on the Google News dataset. 100 billion tokens
- GloVe is trained on a crawled corpus. 840 billion tokens
How about other language?
How about other language?

- Number of Wikipedia articles in different languages:
  - English: ~2.5 M
  - German: ~800 K
  - French: ~700 K
  - Czech: ~100 K
  - Danish: ~95K
  - Chichewa: 58

High-resource languages: 23 languages have more than 100K articles.

Low-resource languages: 60 languages have 10K ~ 100K articles.

Very low-resource languages: 183 languages have less than 10K articles.
Sparsity of the co-occurrence matrix

• Word Embeddings are trained based on co-occurrence statistics
• When training corpus is small
  • Many word pairs are unobserved
  • Co-occurrence matrix is very sparse
• Example: The text8 data
  • 17,000,000 tokens and 71,000 distinct words
  • Co-occurrence matrix has more than 5,000,000,000 entries, >99% are zeros.
Zeros in the co-occurrence matrix

- True zeros
  - Word pairs which are unlikely to co-occur
- Missing entries
  - Word pairs can co-occur
  - Unobserved in the training data

<table>
<thead>
<tr>
<th></th>
<th>alien</th>
<th>table</th>
<th>...</th>
<th>cake</th>
<th>space</th>
</tr>
</thead>
<tbody>
<tr>
<td>alien</td>
<td>0.8</td>
<td>0.1</td>
<td>-</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>table</td>
<td>0</td>
<td>0</td>
<td>-</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>...</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>cake</td>
<td>0</td>
<td>0.2</td>
<td>-</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>space</td>
<td>0</td>
<td>0</td>
<td>-</td>
<td>0</td>
<td>0.7</td>
</tr>
</tbody>
</table>
Motivation

• Small size text corpus
  ⇒ Extremely sparse co-occurrence matrix

• Existing approaches do not use unobserved word pairs effectively
  • E.g., Word2Vec subsamples only some negative word pairs (negative sampling)

• Similar problem is faced by recommendation system
  • User-Product matrix
  • Positive Unlabeled learning
Our contributions

1. Propose a **PU-Learning framework** for training word embedding

2. Design an efficient learning algorithm to deal with **all** negative pairs

3. Demonstrate that unobserved word pairs provide valuable information
PU-Learning for Training Word Embedding
PU Learning Framework

1. **Pre-processing:**
   Building co-occurrence matrix

2. **Matrix factorization by PU-Learning**

3. **Post-processing**
Step 1 – Building co-occurrence matrix

- Count words co-occurrence statistics
- We follow [Levy+15] to scale the co-occurrence counts by PPMI metric

<table>
<thead>
<tr>
<th></th>
<th>cat</th>
<th>dog</th>
<th>table</th>
<th>happy</th>
</tr>
</thead>
<tbody>
<tr>
<td>cat</td>
<td>0.8</td>
<td>0.1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>black</td>
<td>0</td>
<td>0</td>
<td>-</td>
<td>0</td>
</tr>
<tr>
<td>likes</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>milk</td>
<td>0</td>
<td>0</td>
<td>-</td>
<td>0</td>
</tr>
<tr>
<td>0.2</td>
<td>0</td>
<td>0</td>
<td>-</td>
<td>0.2</td>
</tr>
<tr>
<td>context word</td>
<td>cat</td>
<td>dog</td>
<td>...</td>
<td>table</td>
</tr>
<tr>
<td>--------------</td>
<td>-----</td>
<td>-----</td>
<td>-----</td>
<td>-------</td>
</tr>
<tr>
<td>black</td>
<td>0.8</td>
<td>0.1</td>
<td>-</td>
<td>0</td>
</tr>
<tr>
<td>blue</td>
<td>0.5</td>
<td>0</td>
<td>-</td>
<td>0</td>
</tr>
<tr>
<td>...</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>yellow</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>milk</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Center word</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Step 2 - PU-Learning for matrix factorization

<table>
<thead>
<tr>
<th>context word</th>
<th>cat</th>
<th>dog</th>
<th>...</th>
<th>table</th>
<th>happy</th>
</tr>
</thead>
<tbody>
<tr>
<td>black</td>
<td>0.8</td>
<td>0.1</td>
<td>-</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>blue</td>
<td>0</td>
<td>0</td>
<td>-</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>...</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>yellow</td>
<td>0</td>
<td>0</td>
<td>-</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>milk</td>
<td>0.2</td>
<td>0</td>
<td>-</td>
<td>0</td>
<td>0.2</td>
</tr>
</tbody>
</table>

Center word

<table>
<thead>
<tr>
<th>cat</th>
<th>dog</th>
<th>...</th>
<th>table</th>
<th>happy</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.3</td>
<td>0.1</td>
<td>...</td>
<td>0.2</td>
<td>0.3</td>
</tr>
<tr>
<td>0.1</td>
<td>0.1</td>
<td>...</td>
<td>0.1</td>
<td>0.1</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>black</th>
<th>blue</th>
<th>...</th>
<th>yellow</th>
<th>milk</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.1</td>
<td>0.1</td>
<td>-</td>
<td>0.1</td>
<td>0.1</td>
</tr>
<tr>
<td>0.1</td>
<td>0.2</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>0.1</td>
<td>0.1</td>
<td>-</td>
<td>0.1</td>
<td>-</td>
</tr>
<tr>
<td>0.2</td>
<td>0.2</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>

\[ A \approx W^T H \]
Step 2 - PU-Learning for matrix factorization

\[ A \approx W^T H \]

Weighting function

Reconstruction error

Regularization

\[
\min_{W,H} \sum_{i,j \in \Omega} C_{ij} \left( A_{ij} - w_i^T h_j - b^i - \hat{b}^i \right)^2 + \sum_i \lambda_i \| w_i \|^2 + \sum_j \bar{\lambda}_j \| h_j \|^2
\]
### Step 2 – Weighting function

![Table](image)

Three types of entries:

1. **Co-occurrence count > \( x_{max} \)**
   
   \[ C_{ij} = 1 \]

2. **Co-occurrence count \( \leq x_{max} \)**
   
   \[ C_{ij} = \frac{\text{count}}{x_{max}} \]

3. **Co-occurrence count = 0**
   
   \[ C_{ij} = \rho \]
Step 2 - PU-Learning for matrix factorization

\[
\min_{W,H} \sum_{i,j \in \Omega} C_{ij} (A_{ij} - w_i^T h_j - b^i - \hat{b}^j)^2 + \sum_i \lambda_i \|w_i\|^2 + \sum_j \bar{\lambda}_j \|h_j\|^2
\]

- We consider all entries
  - Both positive and zero entries
Step 2 - PU-Learning for matrix factorization

\[
\min_{W,H} \sum_{i,j \in \Omega} C_{ij} (A_{ij} - w_i^T h_j - b^i - \hat{b}^j)^2 + \sum_{i} \lambda_i \|w_i\|^2 + \sum_{j} \bar{\lambda}_j \|h_j\|^2
\]

- We design efficient coordinate descent algorithm (see paper for details)
Step 2 - PU-Learning for matrix factorization

\[
\min_{W,H} \sum_{i,j \in \Omega} C_{ij} \left( A_{ij} - w_i^T h_j - b^i - \hat{b}^j \right)^2 + \sum_i \lambda_i \| w_i \|^2 + \sum_j \bar{\lambda}_j \| h_j \|^2
\]

- We design efficient coordinate descent algorithm (see paper for details)
Step 2 - PU-Learning for matrix factorization

\begin{equation}
\min_{W,H} \sum_{i,j \in \Omega} C_{ij} (A_{ij} - w_i^T h_j - b^i - \hat{b}^j)^2 + \sum_i \lambda_i \|w_i\|^2 + \sum_j \bar{\lambda}_j \|h_j\|^2
\end{equation}

- We design efficient coordinate descent algorithm (see paper for details)
**Step 2 - PU-Learning for matrix factorization**

\[
\min_{W,H} \sum_{i,j \in \Omega} C_{ij} \left( A_{ij} - w_i^T h_j - b^i - \hat{b}^j \right)^2 + \sum_i \lambda_i \| w_i \|^2 + \sum_j \bar{\lambda}_j \| h_j \|^2
\]

- We design efficient coordinate descent algorithm (see paper for details)
Step 2 - PU-Learning for matrix factorization

\[ A \approx W^T H \]

- We design efficient coordinate descent algorithm (see paper for details)
Step 3 -- Post-processing

- Each word is represented by a word vector $w_i^T$ and a context vector $h_i$
- We follow [Pennington+14, Levy+15] to use the average of $w_i^T$ and $h_i$ as word vector for word $i$
Experiments
Results on English

Simulate the low-resource setting: Embedding is trained on a subset of Wikipedia with 32M tokens

Analogy Task on Google Dataset

Word Similarity Task on WS353
Results on Danish (more results in paper)

Danish Wikipedia with 64M tokens
Test set are translated by Google translation (w/ 90% accuracy verified by native speakers)

Analogy Task on Google Dataset

Word Similarity Task on WS353
**Interpretation of Parameters - $\rho$**

- Weight for zero entries in co-occurrence matrix
- Zero entries can be true 0 or missing
- $\rho$ reflects how confident that the zero entries are true zero
Take home messages

• A PU-Learning framework for learning word embedding in the low resource setting
• Unobserved word pairs provide valuable information

Thanks!