Batch IS NOT Heavy: Learning Word Representations From All Samples

1Xin Xin, 1Fajie Yuan, 2Xiangnan He1 Joemon Jose

1 School of Computing Science, University of Glasgow
2 School of Computing, National University of Singapore

Presented by Xin Xin@ACL 2018, July 17, 2018
Word Representations

- Representing words has become a basis for many NLP tasks.
  - One-hot encoding
    - Large dimensionality
    - Sparse representation (most zeros)
  - Dense word embedding
    - 100~400 dins with real-valued vectors
    - Semantic and syntactic meaning in latent space
Learning word embedding

• Predictive models:
  – Word2vec: CBOW & Skip-gram

• Count-based models:
  – GloVe: Biased MF on word co-occurrence statistics
Learning word embedding

• Training of Skip-gram
  – Predicting proper context $c$ given target word $w$
  – Negative sampling to introduce negative $c$
    • Word frequency-based sampling distribution
  – SGD to perform optimization

• Limitations
  – Sampling is a biased approach
    • Chen et al. (2018) recently found that replacing the original sampler with an adaptive sampler could result in better performance
  – SGD fluctuates heavily
Limitations

- Sampled negative instances have great influence

- Sample size and sampling distribution have great impact
- Smaller corpora tend to require a large sample size
Motivations

• Can we drop out negative sampling and directly learn from whole data?

• With whole data considered, can we design an efficient learning scheme to perform optimization?
Contributions

• We directly learn word embeddings from whole data without any sampling
  – All observed (positive) and unobserved (negative) \((w, c)\) pairs are considered
  – Fine-grained weights for negative pairs

• We propose an efficient training algorithm to tackle the huge whole data
  – Keeps the same complexity with sampling based methods
  – More stable convergence
Loss function for all data

- Count-based loss function
- Account for all examples without any sampling

\[ L = \sum_{(w,c) \in S} \alpha_{wc}^+(r_{wc}^+ - U_w \tilde{U}_c^T)^2 + \sum_{(w,c) \in (V \times V) \setminus S} \alpha_{wc}^-(r^- - U_w \tilde{U}_c^T)^2 \]

- \(S\): set of positive \((w, c)\) co-occurrence pairs
- \(V\): vocabulary
- \(U_w(\tilde{U}_c)\): embedding vectors for word (context)
- \(\alpha_{wc}^+, \alpha_{wc}^-\): weights for positive(negative) \((w, c)\) pairs
- \(r_{wc}^+, r^-\): target values for positive(negative) \((w, c)\) pairs
Difficulties to Optimize

• Time complexity

\[
L = \sum_{(w,c) \in S} \alpha_{wc}^+ (r_{wc}^+ - U_w \tilde{U}_c^T)^2 + \sum_{(w,c) \in (V \times V) \setminus S} \alpha_{wc}^- (r^- - U_w \tilde{U}_c^T)^2
\]

\[L_P \quad L_N\]

– \(O(|V|^2k)\) : easily reach tens of billions (e.g., with a 100K vocabulary, \(|V|^2\) reaches 10 billion, \(k\): embedding size)

A more efficient training algorithm needs to be developed
Difficulties to Optimize

\[
L = \sum_{(w,c) \in S} \alpha^+_{wc} (r^+_{wc} - U_w \tilde{U}_c^T)^2 \quad + \quad \sum_{(w,c) \in (V \times V) \setminus S} \alpha^-_{wc} (r^- - U_w \tilde{U}_c^T)^2
\]

\(|V| \times |V|\) interactions

Breaking

1. Loss Partition
2. Product Decouple
Loss Partition

\[ L = \sum_{(w,c) \in S} \alpha_{wc}^+(r_{wc}^+ - U_w \tilde{U}_c^T)^2 + \sum_{(w,c) \in (V \times V) \setminus S} \alpha_{wc}^- (r^- - U_w \tilde{U}_c^T)^2 \]

\[ L_P \]
\[ L_N \]

- The major computation lies in \( L_N \)
  - Transfer \( L_N \)

\[ L_N = \sum_{w \in V} \sum_{c \in V} \alpha_c^- (r^- - U_w \tilde{U}_c^T)^2 - \sum_{(w,c) \in S} \alpha_c^- (r^- - U_w \tilde{U}_c^T)^2 \]

- Now, the major part falls in \( L_A \)
  - Merge with \( L_P \)
Product Decouple

• Inner product Decouple
  – Rewrite $L_A$ into $\tilde{L}_A$ with the constant part $\alpha_c^-(r^-)^2$ omitted

  $$\tilde{L}_A = \sum_{w \in V} \sum_{c \in V} \alpha_c^- \sum_{d=0}^{k} u_{wd}\tilde{u}_{cd} \sum_{d'=0}^{k} u_{wd'}\tilde{u}_{cd'}$$

  $$- 2r^- \sum_{w \in V} \sum_{c \in V} \alpha_c^- \sum_{d=0}^{k} u_{wd}\tilde{u}_{cd}$$

  $|V| \times |V|$ interactions between $u_w$ and $\tilde{u}_c$
Product Decouple

• Inner product Decouple
  – Rewrite $L_A$ into $\tilde{L}_A$ with the constant part $\alpha_c^- (r^-)^2$ omitted

\[
\tilde{L}_A = \sum_{w \in V} \sum_{c \in V} \alpha_c^- \sum_{d=0}^{k} u_{wd} \tilde{u}_{cd} \sum_{d'=0}^{k} u_{wd'} \tilde{u}_{cd'}
\]

\[-2r^- \sum_{w \in V} \sum_{c \in V} \alpha_c^- \sum_{d=0}^{k} u_{wd} \tilde{u}_{cd}\]

Commutative property
Product Decouple

• Inner product Decouple
  – Rewrite $L_A$ into $\tilde{L}_A$ with the constant part $\alpha_c^{-}(r^-)^2$ omitted

\[
\tilde{L}_A = \sum_{w \in V} \sum_{c \in V} \alpha_c^{-} \sum_{d=0}^{k} u_{wd} \tilde{u}_{cd} \sum_{d'=0}^{k} u_{wd'} \tilde{u}_{cd'}
\]

\[-2r^- \sum_{w \in V} \sum_{c \in V} \alpha_c^{-} \sum_{d=0}^{k} u_{wd} \tilde{u}_{cd}\]

Commutative property

\[
\tilde{L}_A = \sum_{d=0}^{k} \sum_{d'=0}^{k} \sum_{w \in V} u_{wd} u_{wd'} \sum_{c \in V} \alpha_c^{-} \tilde{u}_{cd} \tilde{u}_{cd'}
\]

\[-2r^- \sum_{d=0}^{k} u_{wd} \sum_{c \in V} \alpha_c^{-} \tilde{u}_{cd}\]
Product Decouple

- Inner product Decouple
  - Rewrite $L_A$ into $\tilde{L}_A$ with the constant part $\alpha_c^- (r^-)^2$ omitted
  
  \[
  \tilde{L}_A = \sum_{w \in V} \sum_{c \in V} \alpha_c^- \sum_{d=0}^{k} u_{wd} \tilde{u}_{cd} \sum_{d'=0}^{k} u_{wd'} \tilde{u}_{cd'} 
  - 2r^- \sum_{w \in V} \sum_{c \in V} \alpha_c^- \sum_{d=0}^{k} u_{wd} \tilde{u}_{cd} 
  \]

  Commutative property

  \[
  \tilde{L}_A = \sum_{d=0}^{k} \sum_{d'=0}^{k} \sum_{w \in V} u_{wd} u_{wd'} \sum_{c \in V} \alpha_c^- \tilde{u}_{cd} \tilde{u}_{cd'} 
  - 2r^- \sum_{d=0}^{k} \sum_{w \in V} u_{wd} \sum_{c \in V} \alpha_c^- \tilde{u}_{cd} 
  \]

  $u_w$ and $\tilde{u}_c$ are now independent
Product Decouple

- Fix one term and update the other

\[
\tilde{L}_A = \sum_{d=0}^{k} \sum_{d'=0}^{k} \sum_{w \in V} u_{wd} u_{wd'} \sum_{c \in V} \alpha^-_c \tilde{u}_{cd} \tilde{u}_{cd'} - 2r^- \sum_{d=0}^{k} \sum_{w \in V} u_{wd} \sum_{c \in V} \alpha^-_c \tilde{u}_{cd}
\]

- We can achieve a $|V| \times |V|$ to $|V| + |V|$ acceleration
  - Time complexity of $L_A$ reduces from $O(|V|^2 k)$ to $O(k^2 |V|)$
  - Embedding size $k$ is much smaller than vocabulary size $|V|$
Efficient training

• Total time complexity
  – The total time complexity is $O(|S|k + |V|k^2)$
  – $\frac{|S|k}{|V|k^2} = \frac{\bar{c}}{k} \gg 1$ ($\bar{c}$: the average number of positive contexts for a word)
  – The complexity is determined by the number of positive samples

We can train on whole data without any sampling but the time complexity is only determined by the positive part.
Experiments

• Evaluation tasks
  – Word analogy (semantic&syntactic)
    • King is to man as queen is to ?
  – Word Similarity
    • MEN,MC,RW,RG,WSim,WRel
  – QVEC (Tsvetkov et al., 2015)
    • Intrinsic evaluation based on feature alignment

• Training Corpora: Text8, NewsIR, Wiki

• Baseline: Skip-gram, Skip-gram with adaptive sampler, GloVe, LexVec (Salle et al., 2016).
Experiments

- Word analogy accuracy (%) on Text8

<table>
<thead>
<tr>
<th></th>
<th>Semantic</th>
<th>Syntactic</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>Skip-gram</td>
<td>47.51</td>
<td>32.26</td>
<td>38.60</td>
</tr>
<tr>
<td>Skip-gram-a</td>
<td>48.10</td>
<td>33.78</td>
<td>39.74</td>
</tr>
<tr>
<td>GloVe</td>
<td>45.11</td>
<td>26.89</td>
<td>34.47</td>
</tr>
<tr>
<td>LexVec</td>
<td>51.87</td>
<td>31.78</td>
<td>40.14</td>
</tr>
<tr>
<td>Our model</td>
<td>56.66</td>
<td>32.42</td>
<td>42.50</td>
</tr>
</tbody>
</table>

- Our model performs especially good
- GloVe performs poorly (lack of negative information)
- Syntactic performance is not so good as semantic performance
Experiments

• Word similarity & QVEC tasks on Text8

<table>
<thead>
<tr>
<th></th>
<th>MEN</th>
<th>MC</th>
<th>RW</th>
<th>RG</th>
<th>WSim</th>
<th>WRel</th>
<th>QVEC</th>
</tr>
</thead>
<tbody>
<tr>
<td>Skip-gram</td>
<td>0.6868</td>
<td>0.6776</td>
<td>0.3336</td>
<td>0.6904</td>
<td>0.7082</td>
<td>0.6539</td>
<td>0.3999</td>
</tr>
<tr>
<td>Skip-gram-a</td>
<td>0.6885</td>
<td>0.6667</td>
<td>0.3399</td>
<td><strong>0.7035</strong></td>
<td>0.7291</td>
<td>0.6708</td>
<td>0.4062</td>
</tr>
<tr>
<td>GloVe</td>
<td>0.4999</td>
<td>0.3349</td>
<td>0.2614</td>
<td>0.3367</td>
<td>0.5168</td>
<td>0.5115</td>
<td>0.3662</td>
</tr>
<tr>
<td>LexVec</td>
<td>0.6660</td>
<td>0.6267</td>
<td>0.2935</td>
<td>0.6076</td>
<td>0.7005</td>
<td>0.6862</td>
<td><strong>0.4211</strong></td>
</tr>
<tr>
<td>Our model</td>
<td><strong>0.6966</strong></td>
<td><strong>0.6975</strong></td>
<td><strong>0.3424</strong></td>
<td>0.6588</td>
<td><strong>0.7484</strong></td>
<td><strong>0.7002</strong></td>
<td><strong>0.4211</strong></td>
</tr>
</tbody>
</table>

– Similar results with word analogy tasks
– GloVe performs poorly on these two tasks
Experiments

• Word analogy accuracy (%) on NewsIR

<table>
<thead>
<tr>
<th></th>
<th>Semantic</th>
<th>Syntactic</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>Skip-gram</td>
<td>70.81</td>
<td>47.48</td>
<td>58.10</td>
</tr>
<tr>
<td>Skip-gram-a</td>
<td>71.74</td>
<td>48.71</td>
<td>59.20</td>
</tr>
<tr>
<td>GloVe</td>
<td>78.79</td>
<td>41.58</td>
<td>58.52</td>
</tr>
<tr>
<td>LexVec</td>
<td>76.11</td>
<td>39.09</td>
<td>55.95</td>
</tr>
<tr>
<td>Our model</td>
<td>78.47</td>
<td>48.33</td>
<td>61.57</td>
</tr>
</tbody>
</table>

• GloVe’s performance is improved
• The proposed model still over-performs GloVe
  – The importance of negative examples
Experiments

• Word similarity & QVEC tasks on NewsIR

<table>
<thead>
<tr>
<th>Model</th>
<th>MEN</th>
<th>MC</th>
<th>RW</th>
<th>RG</th>
<th>WSim</th>
<th>WRel</th>
<th>QVEC</th>
</tr>
</thead>
<tbody>
<tr>
<td>Skip-gram</td>
<td>0.7293</td>
<td>0.7328</td>
<td>0.3705</td>
<td>0.7184</td>
<td>0.7176</td>
<td>0.6147</td>
<td>0.4182</td>
</tr>
<tr>
<td>Skip-gram-a</td>
<td><strong>0.7409</strong></td>
<td>0.7513</td>
<td>0.3797</td>
<td>0.7508</td>
<td>0.7442</td>
<td>0.6398</td>
<td>0.4159</td>
</tr>
<tr>
<td>GloVe</td>
<td>0.5839</td>
<td>0.5637</td>
<td>0.2487</td>
<td>0.6284</td>
<td>0.6029</td>
<td>0.5329</td>
<td>0.3948</td>
</tr>
<tr>
<td>LexVec</td>
<td>0.7301</td>
<td><strong>0.8403</strong></td>
<td>0.3614</td>
<td><strong>0.8341</strong></td>
<td>0.7404</td>
<td><strong>0.6545</strong></td>
<td>0.4172</td>
</tr>
<tr>
<td>Our model</td>
<td>0.7407</td>
<td>0.7642</td>
<td><strong>0.4610</strong></td>
<td>0.7753</td>
<td><strong>0.7453</strong></td>
<td>0.6322</td>
<td><strong>0.4319</strong></td>
</tr>
</tbody>
</table>

– GloVe still performs poorly on these two tasks
Experiments

• Word analogy accuracy (%) on Wiki

<table>
<thead>
<tr>
<th></th>
<th>Semantic</th>
<th>Syntactic</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>Skip-gram</td>
<td>73.91</td>
<td>61.91</td>
<td>67.37</td>
</tr>
<tr>
<td>Skip-gram-a</td>
<td>75.11</td>
<td>61.94</td>
<td>67.92</td>
</tr>
<tr>
<td>GloVe</td>
<td>77.38</td>
<td>58.94</td>
<td>67.33</td>
</tr>
<tr>
<td>LexVec</td>
<td>76.31</td>
<td>56.83</td>
<td>65.48</td>
</tr>
<tr>
<td>Our model</td>
<td>77.64</td>
<td>60.96</td>
<td><strong>68.52</strong></td>
</tr>
</tbody>
</table>

• Models tend to have similar performance in large datasets
Experiments

• Word similarity & QVEC tasks on Wiki

<table>
<thead>
<tr>
<th></th>
<th>MEN</th>
<th>MC</th>
<th>RW</th>
<th>RG</th>
<th>WSim</th>
<th>WRel</th>
<th>QVEC</th>
</tr>
</thead>
<tbody>
<tr>
<td>Skip-gram</td>
<td>0.7564</td>
<td>0.8083</td>
<td>0.4311</td>
<td>0.7678</td>
<td><strong>0.7662</strong></td>
<td>0.6485</td>
<td>0.4306</td>
</tr>
<tr>
<td>Skip-gram-a</td>
<td><strong>0.7577</strong></td>
<td>0.7940</td>
<td>0.4379</td>
<td>0.7683</td>
<td>0.7110</td>
<td>0.6488</td>
<td>0.4464</td>
</tr>
<tr>
<td>GloVe</td>
<td>0.7370</td>
<td>0.7767</td>
<td>0.3197</td>
<td>0.7499</td>
<td>0.7359</td>
<td>0.6336</td>
<td>0.4206</td>
</tr>
<tr>
<td>LexVec</td>
<td>0.7256</td>
<td><strong>0.8219</strong></td>
<td>0.4383</td>
<td>0.7797</td>
<td>0.7548</td>
<td>0.6091</td>
<td>0.4396</td>
</tr>
<tr>
<td>Our model</td>
<td>0.7396</td>
<td>0.7840</td>
<td><strong>0.4966</strong></td>
<td><strong>0.7800</strong></td>
<td>0.7492</td>
<td><strong>0.6518</strong></td>
<td><strong>0.4489</strong></td>
</tr>
</tbody>
</table>

• To conclude
  – Our model performs especially good in smaller datasets
  – GloVe performs poorly on word similarity and QVEC tasks
  – The difference between models tends to become smaller in large datasets
Experiments

• Effect of weight parameters

\[ \alpha_{wc} = \alpha_c = \alpha_0 \frac{M_{\delta c}}{\sum_{c \in V} M_{\delta c}} \]

– Performance boosts when \( \alpha_0 \) becomes non-zero
  • Negative information is of vital importance
– Best performance achieves when \( \delta \) is around 0.75
  • Same with the power used in negative sampling
Experiments

- Running time on NewsIR corpus

<table>
<thead>
<tr>
<th></th>
<th>Single iter</th>
<th>Iteration</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>SG-3</td>
<td>259s</td>
<td>15</td>
<td>65m</td>
</tr>
<tr>
<td>SG-7</td>
<td>521s</td>
<td>15</td>
<td>131m</td>
</tr>
<tr>
<td>SG-10</td>
<td>715s</td>
<td>15</td>
<td>179m</td>
</tr>
<tr>
<td>Ours</td>
<td>388s</td>
<td>50</td>
<td>322m</td>
</tr>
</tbody>
</table>

- In a single iteration, our model has the same level of running time with skip-gram
- The proposed model need to run more iteration, resulting in a little longer total time
- Running time has almost a linear relationship with embedding size
  - $O(|S|k)$ (positive pairs) accounts for the main part in total $O(|S|k + |V|k^2)$ complexity
Conclusion & Future works

• Conclusion:
  – We proposed a new embedding method which can directly learn from whole data without any sampling
  – We developed a new learning scheme to perform efficient optimization
    • Complexity of learning from whole data is only determined by the positive part.

• Future works:
  – Generalize the proposed learning scheme to other loss functions
  – Full example learning for deep models
Thank you