WordRank: Learning Word Embeddings via Robust Ranking

(Supplementary Material)

Shihao Ji, Hyokun Yun, Pinar Yanardag, Shin Matsushima, and S. V. N. Vishwanathan

A Parallel *WordRank*

Given number of p workers, we partition words $\mathcal{W}$ into p parts $\{\mathcal{W}^{(1)}, \mathcal{W}^{(2)}, \cdots, \mathcal{W}^{(p)}\}$ such that they are mutually exclusive, exhaustive and approximately equal-sized. This partition on $\mathcal{W}$ induces a partition on $\mathbf{U}$, Ω and Ξ as follows: $\mathbf{U}^{(q)} := \{\mathbf{u}_w\}_{w\in\mathcal{W}^{(q)}}$, $\Omega^{(q)} := \{(w,c) \in \Omega\}_{w\in\mathcal{W}^{(q)}}$, and $\Xi^{(q)} := \left\{\xi_{(w,c)}\right\}_{(w,c)\in\Omega^{(q)}}$ for $1 \le q \le p$. When the algorithm starts, $\mathbf{U}^{(q)}$, $\Omega^{(q)}$ and $\Xi^{(q)}$ are distributed to worker q.

At the beginning of each outer iteration, an approximately equal-sized partition $\{\mathcal{C}^{(1)}, \mathcal{C}^{(2)}, \cdots, \mathcal{C}^{(p)}\}$ on the context set $\mathcal{C}$ is sampled; note that this is independent of the partition on words $\mathcal{W}$. This induces a partition on context vectors $\mathbf{V}^{(1)}, \mathbf{V}^{(2)}, \cdots, \mathbf{V}^{(p)}$ defined as follows: $\mathbf{V}^{(q)} := \{\mathbf{v}_c\}_{c\in\mathcal{C}^{(q)}}$ for each q. Then, each $\mathbf{V}^{(q)}$ is distributed to each worker q. Now we define

$$\overline{J}^{(q)}(\mathbf{U}^{(q)}, \mathbf{V}^{(q)}, \Xi^{(q)}) = \sum_{(w,c)\in\Omega\cap\left(\mathcal{W}^{(q)}\times\mathcal{C}^{(q)}\right)} \sum_{c'\in\mathcal{C}^{(q)}\setminus\{c\}} j(w,c,c'), \quad (15)$$

where $j(w,c,c')$ was defined in (14). Note that $j(w,c,c')$ in the above equation only accesses $\mathbf{u}_w$, $\mathbf{v}_c$ and $\mathbf{v}_{c'}$ which belong to no sets other than $\mathbf{U}^{(q)}$ and $\mathbf{V}^{(q)}$, therefore worker q can run stochastic gradient descent updates on (15) for a predefined amount of time without having to communicate with other workers. The pseudo-code is illustrated in Algorithm 2.

Considering that the scope of each worker is always confined to a rather narrow set of observations $\Omega \cap \left(\mathcal{W}^{(q)} \times \mathcal{C}^{(q)}\right)$, it is somewhat surprising that Gemulla et al. (2011) proved that such an optimization scheme, which they call stratified stochastic gradient descent (SSGD), converges to the same local optimum a vanilla SGD would converge to. This is due to the fact that

$$\mathbb{E}\Big[\overline{J}^{(1)}(\mathbf{U}^{(1)}, \mathbf{V}^{(1)}, \Xi^{(1)}) + \overline{J}^{(2)}(\mathbf{U}^{(2)}, \mathbf{V}^{(2)}, \Xi^{(2)}) + \cdots + \overline{J}^{(p)}(\mathbf{U}^{(p)}, \mathbf{V}^{(p)}, \Xi^{(p)})\Big] \approx \overline{J}(\mathbf{U}, \mathbf{V}, \Xi), \quad (16)$$

if the expectation is taken over the sampling of the partitions of $\mathcal{C}$. This implies that the bias in each iteration due to narrowness of the scope will be washed out in a long run; this observation leads to the proof of convergence in Gemulla et al. (2011) using standard theoretical results from Yin and Kushner (2003).

Algorithm 2 Distributed *WordRank* algorithm.

η: step size
repeat
  `//Start outer iteration`
  Sample a partition over contexts $\mathcal{C}^{(1)}, \cdots, \mathcal{C}^{(q)}$
  `//Step 1:Update` $\mathbf{U}, \mathbf{V}$ `in parall.`
  for all machine $q \in \{1, \cdots, p\}$ **do in parallel**
    Fetch all $\mathbf{v}_c \in \mathbf{V}^{(q)}$
    repeat
      Sample (w,c) uniformly from $\Omega^{(q)} \cap \left(\mathcal{W}^{(q)} \times \mathcal{C}^{(q)}\right)$
      Sample c' uniformly from $\mathcal{C}^{(q)} \setminus \{c\}$
      `//following three updates are done simultaneously`
      $\mathbf{u}_w \leftarrow \mathbf{u}_w - \eta \cdot r_{w,c} \cdot \rho'(\xi_{w,c}^{-1}) \cdot \ell'\left(\langle \mathbf{u}_w, \mathbf{v}_c - \mathbf{v}_{c'}\rangle\right) \cdot (\mathbf{v}_c - \mathbf{v}_{c'})$
      $\mathbf{v}_c \leftarrow \mathbf{v}_c - \eta \cdot r_{w,c} \cdot \rho'(\xi_{w,c}^{-1}) \cdot \ell'\left(\langle \mathbf{u}_w, \mathbf{v}_c - \mathbf{v}_{c'}\rangle\right) \cdot \mathbf{u}_w$
      $\mathbf{v}_{c'} \leftarrow \mathbf{v}_{c'} + \eta \cdot r_{w,c} \cdot \rho'(\xi_{w,c}^{-1}) \cdot \ell'\left(\langle \mathbf{u}_w, \mathbf{v}_c - \mathbf{v}_{c'}\rangle\right) \cdot \mathbf{u}_w$
    until predefined time limit is exceeded
  end for
  `//Step 2: Update` Ξ `in parallel`
  for all machine $q \in \{1, \cdots, p\}$ **do in parallel**
    Fetch all $\mathbf{v}_c \in \mathbf{V}$
    for $w \in \mathcal{W}^{(q)}$ **do**
      for $c \in \mathcal{C}$ **do**
      $\xi_{w,c} = \alpha / \left(\sum_{c'\in\mathcal{C}\setminus\{c\}} \ell\left(\langle \mathbf{u}_w, \mathbf{v}_c - \mathbf{v}_{c'}\rangle\right) + \beta\right)$
      end for
    end for
  end for
until $\mathbf{U}$, $\mathbf{V}$ and Ξ are converged

Corpus Size	WS-353 (Word Similarity)			Google (Word Analogy)		
	word2vec	*GloVe*	*WordRank*	*word2vec*	*GloVe*	*WordRank*
17M	66.8	47.8	70.4	39.2	30.4	44.5
32M	64.1	47.8	68.4	42.3	30.9	52.1
64M	67.5	55.0	70.8	53.5	42.0	59.9
128M	70.7	54.5	72.8	59.8	50.4	65.1
256M	72.0	59.5	72.4	67.6	60.3	68.6
512M	72.3	64.5	74.1	70.6	66.4	70.6
1.0B	73.3	68.3	74.0	70.4	68.7	70.8
1.6B	71.8	69.5	74.1	72.1	70.4	71.7
7.2B	73.4	70.9	75.2/77.4[1]	75.1[2]	75.6[2]	76.0[2,3]

[1] When ρ_0 is used, corresponding to setting $\xi = 1$ in training and no ξ update
[2] Use 3CosMul instead of regular 3CosAdd for evaluation
[3] Use $\mathbf{u}_w$ instead of default $\mathbf{u}_w + \mathbf{v}_c$ as word representation for evaluation

Table 4: Performance of *word2vec*, *GloVe* and *WordRank* on datasets with increasing sizes; evaluated on WS-353 word similarity benchmark and Google word analogy benchmark.

B Additional Experimental Details

Table 4 is the tabular view of the data plotted in Figure 2 to provide additional experimental details.