Word Representations

by
Cheng Zheng (UID: #304550266)
Junheng Hao (UID: #805025633)
Guangyu Zhou (UID: #204778222)

GloVe

Global Vectors for Word Representation

Jeffrey Pennington, Richard Socher, Christopher D. Manning

Presented by Guangyu Zhou

Outline

Papers: **Word embedding** -- GloVe: Global Vectors for Word Representation

1. **Motivation**
2. **Notations**
3. **Model Construction**
4. **Complexity**
5. **Experiments**

Preliminary: Word embedding

Word embedding refers to a kind of methods that learn a distributed dense vector for each word in a vocabulary.

1) Traditional word embedding methods first obtain the **co-occurrence matrix** then perform dimension reduction with PCA. (Global matrix factorization)
2) Recent methods use neural language models that directly learn word vectors by predicting the **context** words of the target word. (Local context window)

GloVe: Global Vectors for Word Representation

1. **Motivation**

Two main model families for learning word vectors are:

(1) **Global matrix factorization** -- Latent Semantic Analysis
(2) **Local context window** -- Skip-gram model (*Word2Vec*)

GloVe: Global Vectors for Word Representation

(1) **Global matrix factorization utilizes low rank approximations to decompose large matrix that capture statistical information about a corpus.**

Latent Semantic Analysis -- term-document matrix

Latent semantic analysis (LSA) is a technique in natural language processing, in particular distributional semantics, of **analyzing relationships between a set of documents and the terms** they contain by producing a set of concepts related to the documents and terms.

Latent Semantic Analysis

- Solve LSA by SVD

Map to a lower dimensional space

$$
\begin{array}{ccccccc}
 & X & & & U & \Sigma & V^T \\
 & (\mathbf{d}_j) & & & & & (\hat{\mathbf{d}}_j) \\
 & \downarrow & & & & & \downarrow \\
(\mathbf{t}_i^T) \rightarrow & \begin{bmatrix} x_{1,1} & \dots & x_{1,n} \\ \vdots & \ddots & \vdots \\ x_{m,1} & \dots & x_{m,n} \end{bmatrix} & = & (\hat{\mathbf{t}}_i^T) \rightarrow & \begin{bmatrix} \begin{bmatrix} \\ \mathbf{u}_1 \\ \\ \end{bmatrix} \dots \begin{bmatrix} \\ \mathbf{u}_l \\ \\ \end{bmatrix} \end{bmatrix} & \cdot \begin{bmatrix} \sigma_1 & \dots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \dots & \sigma_l \end{bmatrix} & \cdot \begin{bmatrix} [& \mathbf{v}_1 &] \\ & \vdots & \\ [& \mathbf{v}_l &] \end{bmatrix}
\end{array}
$$

1. Perform SVD on document-term adjacency matrix
2. Construct $C^k_{M \times N}$ by only keeping the largest k singular values in Σ non-zero

The two main model families for learning word vectors Pros & Cons

(1) **Global matrix factorization (LSA)**
  + Efficiently leverage statistical information
  - Do relatively poorly on the word analogy task, indicating **a sub-optimal vector space structure**.

(2) **Local context window (Skip-gram)**
  + Do better on the analogy task
  - **Poorly utilize statistics of the corpus** since they train on separate local context windows instead of on global co-occurrence counts.

GloVe: Global Vectors for Word Representation

1. Motivation

GloVe combines the advantages of two major model families by producing a new global **log-bilinear regression model**

It efficiently leverage statistical information by training only on **the nonzero elements in a word-word co-occurrence matrix**, rather than **entire sparse matrix** or on **individual context windows in a large corpus**

GloVe: Global Vectors for Word Representation

Log-bilinear Model

Given context $\boldsymbol{w}_{1:n-1}$, model predicts the next word $\boldsymbol{W}_n$ by linearly combining the representations of context.

$$\hat{r} = \sum_{i=1}^{n-1} C_i r_{w_i}$$

$\boldsymbol{r}_w$ is the real-valued word vector representing word $\boldsymbol{w}$

Then the description for the next word is computed based on **the similarity between the predicted distribution and representations of all words in the vocabulary**.

$$P(w_n = w | w_{1:n-1}) = \frac{\exp(\hat{r}^T r_w)}{\sum_j \exp(\hat{r}^T r_j)}$$

GloVe: Global Vectors for Word Representation

2. Notations

X : matrix of word-word co-occurrence counts

X_{ij} : the number of times word i occurs in the context of word j

W_i : vector form for word i

$X_i = \Sigma_k X_{ik}$: the number of times any word appears in the context of word i

$P_{ij} = P(j|i) = X_{ij} / X_i$: probability that word j appears in the context of word i

GloVe: Global Vectors for Word Representation

3. Model Construction

Step 1: Co-occurrence probabilities for target words ***ice*** and ***steam*** with selected context words from a **6 billion token corpus**. Only in the ratio does noise from non-discriminative words like ***water*** and ***fashion*** cancel out.

Argument: the appropriate **starting point for word vector learning** should be with **ratios of co-occurrence probabilities** rather than **the probabilities themselves**.

Probability and Ratio	$k = solid$	$k = gas$	$k = water$	$k = fashion$
$P(k\|ice)$	1.9×10^{-4}	6.6×10^{-5}	3.0×10^{-3}	1.7×10^{-5}
$P(k\|steam)$	2.2×10^{-5}	7.8×10^{-4}	2.2×10^{-3}	1.8×10^{-5}
$P(k\|ice)/P(k\|steam)$	8.9	8.5×10^{-2}	1.36	0.96

GloVe: Global Vectors for Word Representation

Step 2:

Noting: the ratio P_{ik} / P_{jk} depends on three words *i, j, k. w and* $\tilde{w}$, *are learned independently.*

⇒ **General model form**:

$$F(w_i, w_j, \tilde{w}_k) = \frac{P_{ik}}{P_{jk}}$$

GloVe: Global Vectors for Word Representation

Step 3:

Intention: encode information of ratio P_{ik} / P_{jk} into function F

⇒ **Most natural way to do - vector difference**:

$$F(w_i - w_j, \tilde{w}_k) = \frac{P_{ik}}{P_{jk}}$$

GloVe: Global Vectors for Word Representation

Step 4:

Imbalance: the right-hand side of ***F*** is a **scalar**, while parameters in left-hand side are **vectors**.

⇒ **To avoid this issue**, take the **dot product**

$$F\left((w_i - w_j)^T \tilde{w}_k\right) = \frac{P_{ik}}{P_{jk}}$$

GloVe: Global Vectors for Word Representation

Step 5:

Note: for word-word co-occurrence matrices X_{ij}, the distinction between a word and a context word is arbitrary and that we are **free to exchange the two roles**.

To do so consistently, we must not only exchange $w \leftrightarrow \tilde{w}$ but also $X \leftrightarrow X^T$

⇒Our final model should be **invariant under this relabeling** ⇒ **store symmetry**

GloVe: Global Vectors for Word Representation

Step 6:

$$\boxed{F\left((w_i - w_j)^T \tilde{w}_k\right) = \frac{P_{ik}}{P_{jk}},}$$

$$F\left((w_i - w_j)^T \tilde{w}_k\right) = \frac{F(w_i^T \tilde{w}_k)}{F(w_j^T \tilde{w}_k)}$$

$$F(w_i^T \tilde{w}_k) = P_{ik} = \frac{X_{ik}}{X_i}$$

$$F = \exp$$

$$w_i^T \tilde{w}_k = \log(P_{ik}) = \log(X_{ik}) - \log(X_i)$$

GloVe: Global Vectors for Word Representation

Step 7:

⇒ continue to **store symmetry**

$$w_i^T \tilde{w}_k = \log(P_{ik}) = \log(X_{ik}) - \boxed{\log(X_i)}$$

$$w_i^T \tilde{w}_k + \boxed{b_i + \tilde{b}_k} = \log(X_{ik})$$

GloVe: Global Vectors for Word Representation

Step 8:

Problem: $w_i^T \tilde{w}_k + b_i + \tilde{b}_k = \log(X_{ik})$ $\log(X_{ik}) \rightarrow \log(1 + X_{ik})$

1. Actually ill-defined since the **logarithm diverges** whenever its argument is 0.
2. Additive shifting? => model weights **all co-occurrences equally**

⇒ Introducing **a weighting function** into the cost function of a least square problem

$$J = \sum_{i,j=1}^{V} f\left(X_{ij}\right)\left(w_i^T \tilde{w}_j + b_i + \tilde{b}_j - \log X_{ij}\right)^2$$

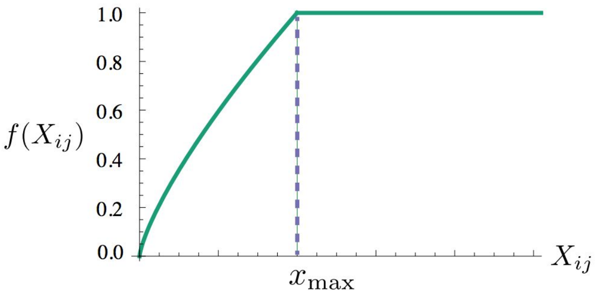
GloVe: Global Vectors for Word Representation

Step 9:

Constraints: *f(0) = 0*; non-decreasing; relatively small for large values of *x*

⇒ One class of functions that work well

$$f(x) = \begin{cases} (x/x_{\max})^{\alpha} & \text{if } x < x_{\max} \\ 1 & \text{otherwise} \end{cases}$$



GloVe: Global Vectors for Word Representation

Model Construction summary

Procedures : generate inspirations from example ⇒ maths representation ⇒ continuously adding constraints to polish up the cost function

1. Vector difference
2. Inner product
3. Keep symmetry
4. Avoid logarithmic divergence
5. Adjust weights

GloVe: Global Vectors for Word Representation

4. Complexity

Cost Function: $J = \sum_{i,j=1}^{V} f\left(X_{ij}\right)\left(w_i^T \tilde{w}_j + b_i + \tilde{b}_j - \log X_{ij}\right)^2$,

The computational complexity of the model depends on the number of nonzero elements in the matrix ***X***.

$$O\,(\,|C|\,) =?\; O\,(\,|V|^{2}\,)$$

$$X_{ij} = \frac{k}{(r_{ij})^\alpha}$$

$$|C| \sim \sum_{ij} X_{ij} = \sum_{r=1}^{|X|} \frac{k}{r^\alpha} = k H_{|X|,\alpha}$$

$$|C| \sim |X|^\alpha \, H_{|X|,\alpha}$$

$$|X| = \begin{cases} O(|C|) & \text{if } \alpha < 1 \\ O(|C|^{1/\alpha}) & \text{if } \alpha > 1 \end{cases}$$

GloVe: Global Vectors for Word Representation

5. Experiments

Tasks:

Word analogies
Word similarity
Named entity recognition

Corpus:

2010 Wikipedia dump with 1 billion tokens;
2014 Wikipedia dump with 1.6 billion tokens;
Gigaword 5 which has 4.3 billion tokens;
Combination Gigaword5 and Wikipedia2014;
42 billion tokens of web data, from Common Crawl.

GloVe: Global Vectors for Word Representation

5. Experiments

frog

frogs

toad

litoria

leptodactylidae

rana

lizard

eleutherodactylus

3. litoria

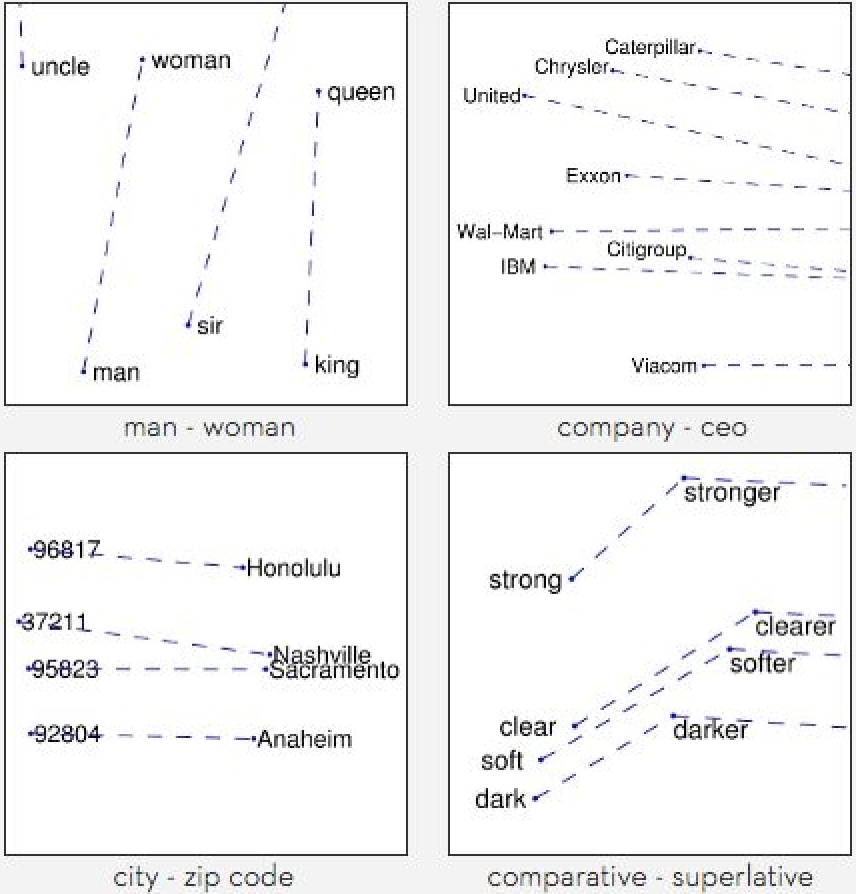
4. leptodactylidae

5. rana

7. eleutherodactylus

GloVe: Global Vectors for Word Representation

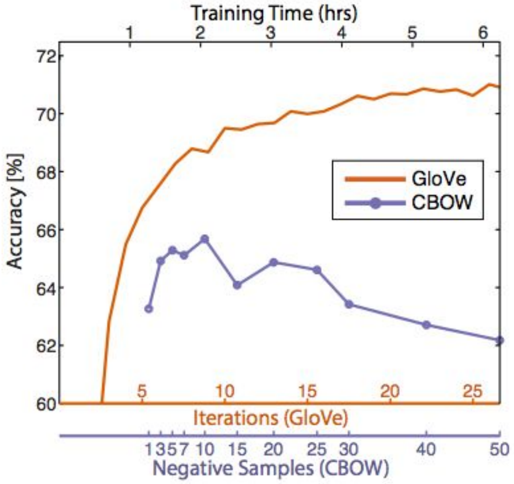
5. Experiments



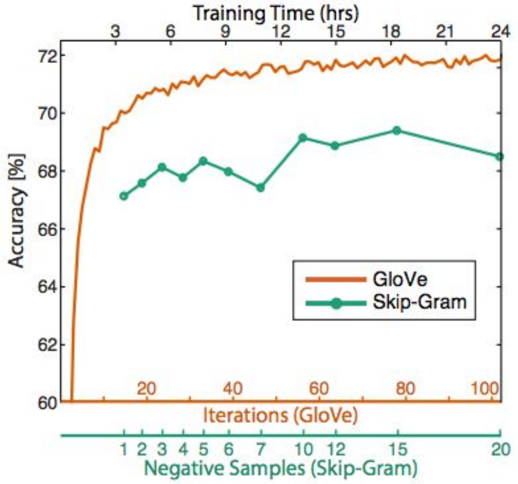
GloVe: Global Vectors for Word Representation

6. Comparison with Word2Vec

Model	Size	WS353	MC	RG	SCWS	RW
SVD	6B	35.3	35.1	42.5	38.3	25.6
SVD-S	6B	56.5	71.5	71.0	53.6	34.7
SVD-L	6B	65.7	72.7	75.1	56.5	37.0
CBOW†	6B	57.2	65.6	68.2	57.0	32.5
SG†	6B	62.8	65.2	69.7	58.1	37.2
GloVe	6B	65.8	72.7	77.8	53.9	38.1
SVD-L	42B	74.0	76.4	74.1	58.3	39.9
GloVe	42B	**75.9**	**83.6**	**82.9**	**59.6**	**47.8**
CBOW*	100B	68.4	79.6	75.4	59.4	45.5



(a) GloVe vs CBOW



(b) GloVe vs Skip-Gram

Conclusion:

1. ***GloVe*** combines advantages from global matrix factorization & local context window.
2. This well-written paper gives us a complete view of model intuition and construction.
3. ***GloVe* outperforms *Word2Vec***.

Improving Distributional Similarity with Lessons Learned from Word Embeddings

Levy, Omer et al.Transactions of the Association for Computational Linguistics 3 (2015): 211-225.

The Contributions of Word Embeddings

Novel Algorithms

(objective + training method)

- Skip Grams + Negative Sampling(SGNS)
- CBOW + Hierarchical Softmax
- Noise Contrastive Estimation
- GloVe
- …

New Hyperparameters

(preprocessing, smoothing, etc.)

- Subsampling
- Dynamic Context Windows
- Context Distribution Smoothing
- Adding Context Vectors
- …

What's really improving performance?

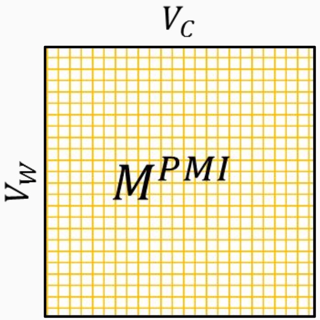
Count-based word embedding models

PPMI Matrix (positive pointwise mutual information)

- Each row is a word w in vocabulary V_W
- Each column is a context c in V_C

$$PMI(w,c) = \log \frac{\hat{P}(w,c)}{\hat{P}(w)\hat{P}(c)} = \log \frac{\#(w,c)\cdot|D|}{\#(w)\cdot\#(c)}$$

$$PPMI(w,c) = \max\left(PMI\left(w,c\right),0\right)$$



Singular Value Decomposition(SVD)

Factorize PPMI into $U \cdot \Sigma \cdot V^{\top}$ ($UU^{\top}$=I, Σ diagonal eigenvalue matrix, $VV^{\top}$=I)

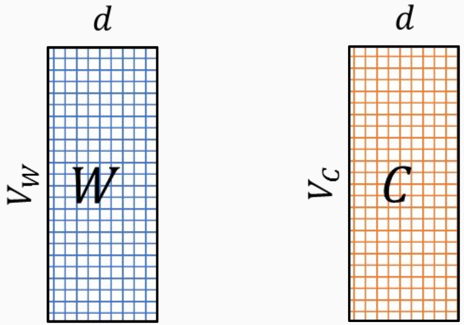
Keeping only top d element of Σ:

$$U_d \cdot \Sigma_d \cdot V_d^{\top}$$

$$W^{\text{SVD}} = U_d \cdot \Sigma_d \qquad C^{\text{SVD}} = V_d$$

What is SGNS learning?

Take SGNS's embedding matrices(W and C)



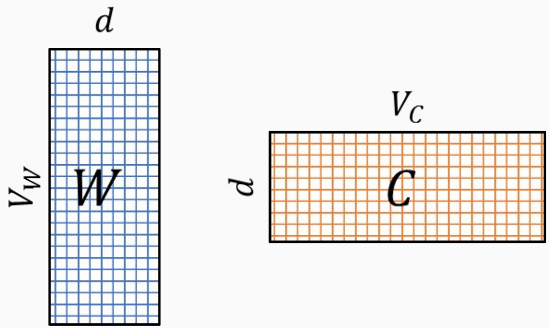
"Neural Word Embeddings as Implicit Matrix Factorization"
Levy & Goldberg, NIPS 2014

What is SGNS learning?

Take SGNS's embedding matrices(W and C)

Multiply them

What do you get?

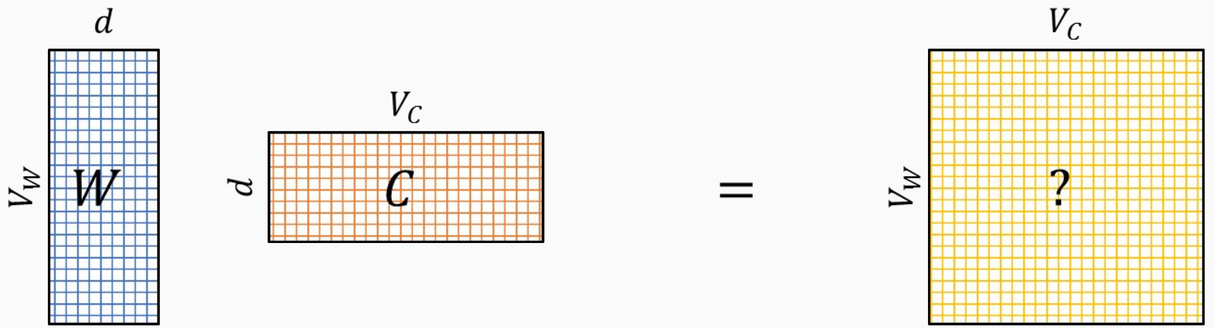


"Neural Word Embeddings as Implicit Matrix Factorization"
Levy & Goldberg, NIPS 2014

What is SGNS learning?

A $V_W \times V_C$ matrix

Each cell describes the relation between a specific word-context pair



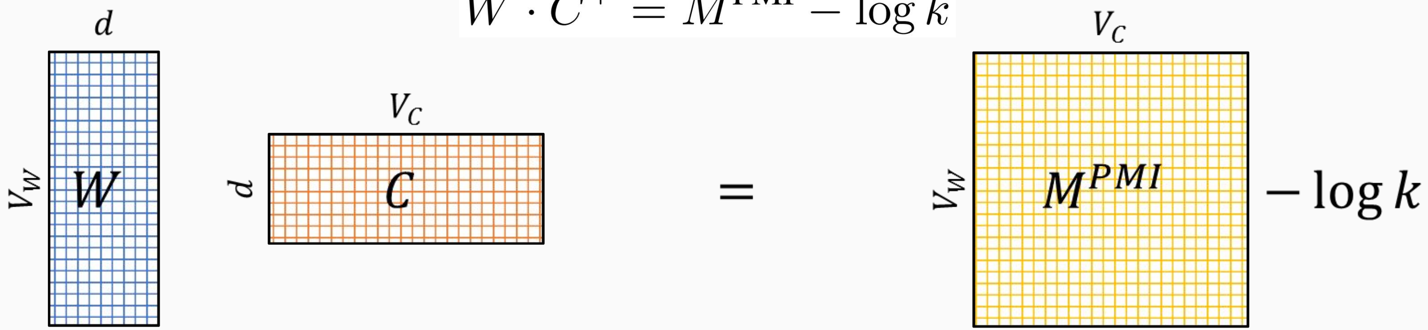
"Neural Word Embeddings as Implicit Matrix Factorization"
Levy & Goldberg, NIPS 2014

What is SGNS learning?

The proof shows for large enough d and enough iterations

SGNS factorizes word-context PMI matrix, shifted by a global constant.

$$W \cdot C^\top = M^{\text{PMI}} - \log k$$



"Neural Word Embeddings as Implicit Matrix Factorization"
Levy & Goldberg, NIPS 2014

The Contributions of Word Embeddings

Novel Algorithms

(objective + training method)

- Skip Grams + Negative Sampling(SGNS)
- CBOW + Hierarchical Softmax
- Noise Contrastive Estimation
- GloVe
- …

New Hyperparameters

(preprocessing, smoothing, etc.)

- Subsampling
- Dynamic Context Windows
- Context Distribution Smoothing
- Adding Context Vectors
- …

What's really improving performance?

New Hyperparameters

- **Preprocessing** **(word2vec)**
  - Dynamic Context Windows
  - Subsampling
  - Deleting Rare Words
- **Postprocessing** **(GloVe)**
  - Adding Context Vectors
- **Association Metric** **(SGNS)**
  - Shifted PMI
  - Context Distribution Smoothing

Dynamic Context Windows

Marco saw a furry little wampimuk hiding in the tree.

	saw	a	furry	little	wampimuk	hiding	in	the	tree
word2vec:	$\frac{1}{4}$	$\frac{2}{4}$	$\frac{3}{4}$	$\frac{4}{4}$		$\frac{4}{4}$	$\frac{3}{4}$	$\frac{2}{4}$	$\frac{1}{4}$
GloVe:	$\frac{1}{4}$	$\frac{1}{3}$	$\frac{1}{2}$	$\frac{1}{1}$		$\frac{1}{1}$	$\frac{1}{2}$	$\frac{1}{3}$	$\frac{1}{4}$
Aggressive:	$\frac{1}{8}$	$\frac{1}{4}$	$\frac{1}{2}$	$\frac{1}{1}$		$\frac{1}{1}$	$\frac{1}{2}$	$\frac{1}{4}$	$\frac{1}{8}$

The Word-Space Model *(Sahlgren, 2006)*

Adding Context Vectors

- SGNS creates word vectors **w**
- SGNS creates auxiliary context vectors **c**
  - So do GloVe and SVD

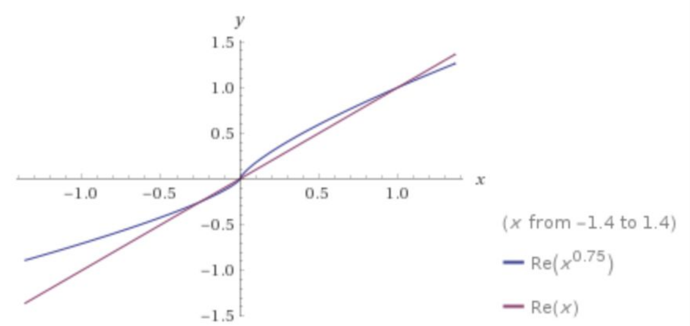
- Instead of just w
- Represent a word as: **w** + **c**

- Adding second order similarity($\mathbf{w}_x \cdot \mathbf{w}_y$, $\mathbf{c}_x \cdot \mathbf{c}_y$)

Context Distribution Smoothing

- SGNS samples c'~P to form negative (w,c') examples
- In practice, it's a smoothed unigram distribution

$$P^{0.75}(c) = \frac{(\#c)^{0.75}}{\sum_{c' \in V_C} (\#c')^{0.75}}$$



- Adding bias towards rare words
- Applicable to PMI:

$$PMI_\alpha(w,c) = \log \frac{\hat{P}(w,c)}{\hat{P}(w)\hat{P}_\alpha(c)}$$

$$\hat{P}_\alpha(c) = \frac{\#(c)^\alpha}{\sum_c \#(c)^\alpha}$$

Recap: All Hyperparameters

- **Preprocessing**
  - Dynamic Context Windows (`dyn`)
  - Subsampling (`sub`)
  - Deleting Rare Words (`del`)
- **Postprocessing**
  - Adding Context Vectors (`w+c`)
- **Association Metric**
  - Shifted PMI (`neg`)
  - Context Distribution Smoothing (`cds`)
- **Others**
  - Context window size (`win`)
  - Eigenvalue Weighting (`eig`)
  - Vector Normalization (`nrm`)

Systematic Experiments Setup

9 Hyperparameters

Hyper-parameter	Explored Values	Applicable Methods
win	2, 5, 10	All
dyn	none, with	All
sub	none, dirty, clean†	All
del	none, with†	All
neg	1, 5, 15	PPMI, SVD, SGNS
cds	1, 0.75	PPMI, SVD, SGNS
w+c	only w, $w + c$	SVD, SGNS, GloVe
eig	0, 0.5, 1	SVD
nrm	none†, row, col†, both†	All

4 Word Representation Algorithms

1. PPMI (Sparse)
2. SVD
3. SGNS
4. GloVe

8 Benchmarks

- 6 Word Similarity Task Datasets
- 2 Analogy Task Dataset (Google, MSR)

Lots of experiments...

Results (in detail)

Method	WordSim Similarity	WordSim Relatedness	Bruni et al. MEN	Radinsky et al. M. Turk	Luong et al. Rare Words	Hill et al. SimLex	Google Add / Mul	MSR Add / Mul
PPMI	.709	.540	.688	.648	.393	.338	.491 / **.650**	.246 / .439
SVD	**.776**	**.658**	**.752**	.557	**.506**	**.422**	.452 / .498	.357 / .412
SGNS	.724	.587	.686	**.678**	.434	.401	.530 / .552	.578 / **.592**
GloVe	.666	.467	.659	.599	.403	.398	.442 / .465	.529 / .576

Table 2: Performance of each method across different tasks in the "vanilla" scenario (all hyperparameters set to default): $\texttt{win} = 2$; $\texttt{dyn} =$ none; $\texttt{sub} =$ none; $\texttt{neg} = 1$; $\texttt{cds} = 1$; $\texttt{w+c} =$ only w; $\texttt{eig} = 0.0$.

Method	WordSim Similarity	WordSim Relatedness	Bruni et al. MEN	Radinsky et al. M. Turk	Luong et al. Rare Words	Hill et al. SimLex	Google Add / Mul	MSR Add / Mul
PPMI	.755	**.688**	.745	**.686**	.423	.354	.553 / **.629**	.289 / .413
SVD	**.784**	.672	**.777**	.625	**.514**	.402	.547 / .587	.402 / .457
SGNS	.773	.623	.723	.676	.431	**.423**	.599 / .625	.514 / .546
GloVe	.667	.506	.685	.599	.372	.389	.539 / .563	.503 / **.559**
CBOW	.766	.613	.757	.663	.480	.412	.547 / .591	.557 / **.598**

Table 3: Performance of each method across different tasks using word2vec's recommended configuration: $\texttt{win} = 2$; $\texttt{dyn} =$ with; $\texttt{sub} =$ dirty; $\texttt{neg} = 5$; $\texttt{cds} = 0.75$; $\texttt{w+c} =$ only w; $\texttt{eig} = 0.0$. CBOW is presented for comparison.

Method	WordSim Similarity	WordSim Relatedness	Bruni et al. MEN	Radinsky et al. M. Turk	Luong et al. Rare Words	Hill et al. SimLex	Google Add / Mul	MSR Add / Mul
PPMI	.755	**.697**	.745	.686	.462	.393	.553 / .679	.306 / .535
SVD	**.793**	.691	**.778**	.666	**.514**	.432	.554 / .591	.408 / .468
SGNS	**.793**	.685	.774	**.693**	.470	**.438**	.676 / **.688**	.618 / **.645**
GloVe	.725	.604	.729	.632	.403	.398	.569 / .596	.533 / .580

Table 4: Performance of each method across different tasks using the best configuration for that method and task combination, assuming $\texttt{win} = 2$.

Results (in detail)

win	Method	WordSim Similarity	WordSim Relatedness	Bruni et al. MEN	Radinsky et al. M. Turk	Luong et al. Rare Words	Hill et al. SimLex	Google Add / Mul	MSR Add / Mul
2	PPMI	.732	**.699**	.744	.654	.457	.382	.552 / .677	.306 / .535
	SVD	.772	.671	**.777**	.647	**.508**	.425	.554 / .591	.408 / .468
	SGNS	**.789**	.675	.773	**.661**	.449	**.433**	.676 / **.689**	.617 / **.644**
	GloVe	.720	.605	.728	.606	.389	.388	.649 / .666	.540 / .591
5	PPMI	.732	**.706**	.738	**.668**	.442	.360	.518 / .649	.277 / .467
	SVD	.764	.679	**.776**	.639	**.499**	**.416**	.532 / .569	.369 / .424
	SGNS	**.772**	.690	.772	.663	.454	.403	.692 / **.714**	.605 / **.645**
	GloVe	.745	.617	.746	.631	.416	.389	.700 / .712	.541 / .599
10	PPMI	.735	**.701**	.741	.663	.235	.336	.532 / .605	.249 / .353
	SVD	.766	.681	.770	.628	**.312**	.419	.526 / .562	.356 / .406
	SGNS	**.794**	.700	**.775**	**.678**	.281	**.422**	.694 / .710	.520 / **.557**
	GloVe	.746	.643	.754	.616	.266	.375	.702 / **.712**	.463 / .519
10	SGNS-LS	.766	.681	**.781**	**.689**	**.451**	.414	.739 / **.758**	.690 / **.729**
	GloVe-LS	.678	.624	.752	.639	.361	.371	.732 / .750	.628 / .685

Table 5: Performance of each method across different tasks using 2-fold cross-validation for hyperparameter tuning. Configurations on large-scale (LS) corpora are also presented for comparison.

Hyperparameter Settings

Classic Setting/Vanilla

- Preprocessing
  - <None>
- Postprocessing
  - <None>
- Association Metric
  - PMI/PPMI

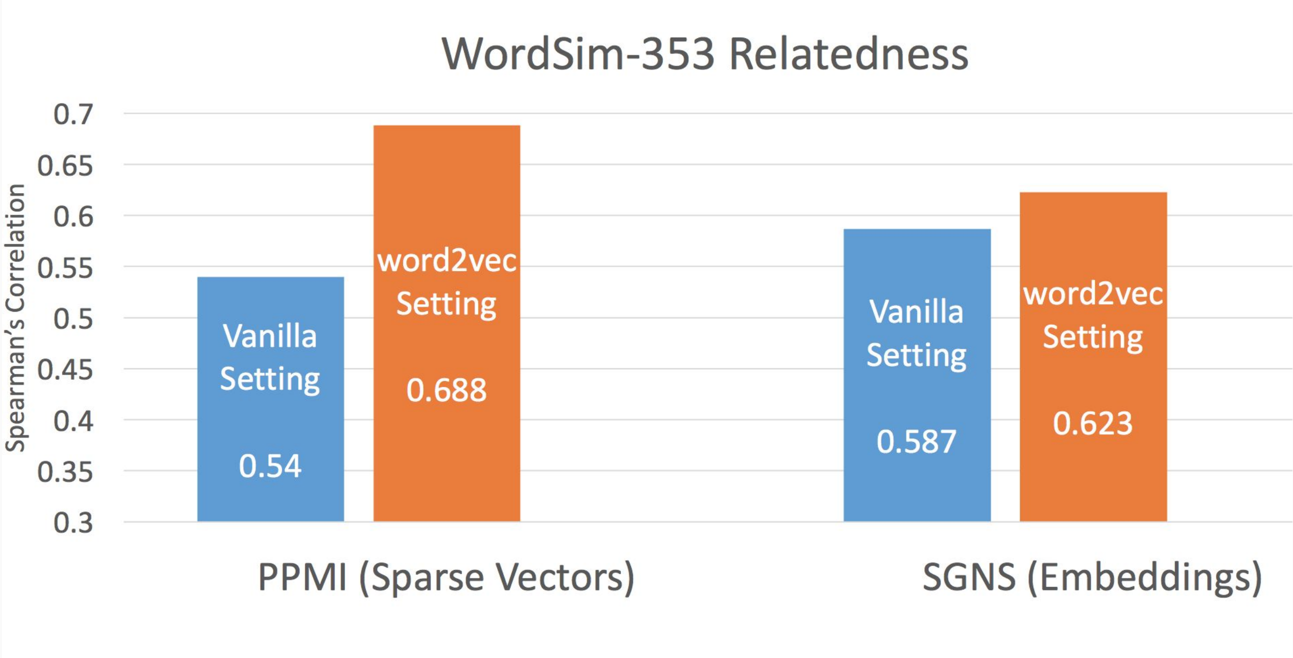
Recommended `word2vec` Setting

- Preprocessing
  - Dynamic Context Window
  - Subsampling
- Postprocessing
  - <None>
- Association Metric
  - Shifted PMI/PPMI
  - Context Distribution Smoothing

Optimal Setting

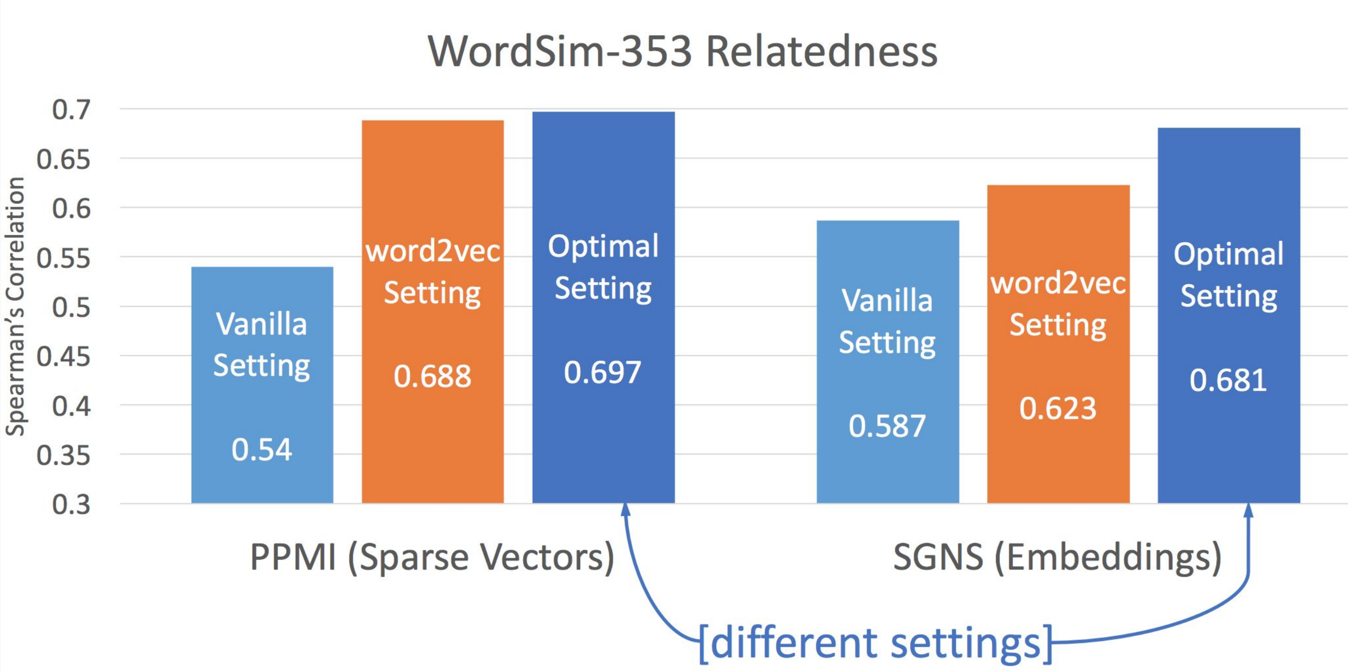
- Enable full range of hyper-parameters and choose the best performance

Experiments: Comparison



Note on measurement
Spearman's Correlation:
Pearson correlation coefficient between the ranked variables

Experiments: Hyperparameter Tuning



Overall Results

- Hyperparameters vs Algorithms
  - Hyperparameters often have stronger effects than algorithms.
- Hyperparameters vs Big Data
  - Hyperparameters often have stronger effects than more data.
- Questions on prior superiority claims
- ...

Re-evaluating Prior Claims

1. Embedding vs count-based methods

--Don't Count, Predict! (Baroni et al., 2014)

Hyperparameter settings account for most of the reported gaps. Embeddings do not really outperform count-based methods (except MSR analogy task).

Re-evaluating Prior Claims

1. Embedding vs count-based methods
2. GloVe vs SGNS?

--GloVe (Pennington et al., 2014)

Difference on hyperparameter settings account for most of the reported gaps. It is observed that SGNS outperformed GloVe almost on every task.

Re-evaluating Prior Claims

1. Embedding vs count-based methods
2. GloVe vs SGNS
3. Linguistic Regularities in Word Representations

“PPMI vectors perform on par with SGNS on analogy tasks.” (Levy and Goldberg, 2014)

It holds on semantic tasks but not on syntactic analogies. In the aspect of syntactic analogies, there is a gap in favor of SGNS.

Examples: Syntactic: “*Good is to best as smart is to smartest*” (MSR)
Semantic: “*Paris is to France as Tokyo is to Japan*.” (Google)

Conclusions

1. Contributions of Word Embeddings: New hyperparameters more than novel algorithms, that is, hyperparameters mostly are improving performances.
2. Beneficial configurations and practical recommendations.
3. Primary suggestion: Look for new hyperparameters and adapt parameters across different models

References

1. Pennington, Jeffrey, et al. "Glove: Global vectors for word representation." *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*. 2014.
2. Levy, Omer, et al. "Improving distributional similarity with lessons learned from word embeddings." *Transactions of the Association for Computational Linguistics* 3 (2015): 211-225.
3. Levy, Omer, and Yoav Goldberg. "Neural word embedding as implicit matrix factorization." *Advances in neural information processing systems*. 2014.

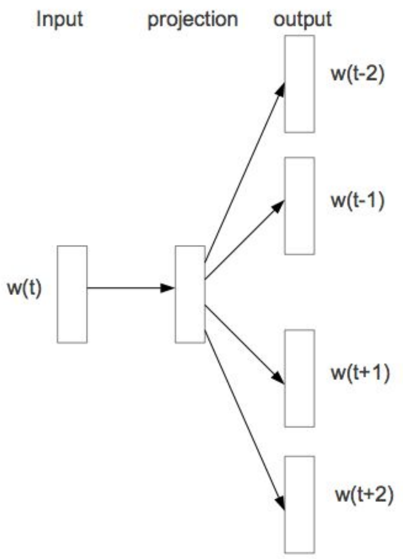
Thanks!
Q&A

Refresh: Skip-gram model

- Objective: Find word representations that are useful for predicting the surrounding words in a sentence or a document.
- Cost function:
  Maximize: $$\frac{1}{T}\sum_{t=1}^{T}\sum_{-c\leq j\leq c, j\neq 0}\log p(w_{t+j}|w_t)$$

  with probabilities defined as, $p(o|c) = \frac{\exp\left(u_o^T v_c\right)}{\sum_{w=1}^{W}\exp\left(u_w^T v_c\right)}$

  which is impractical because the cost of computing gradient is proportional to W.



Refresh:
Skip-gram + negative sampling = Word2Vec

Overall objective function: $J(\theta) = \frac{1}{T}\sum_{t=1}^{T} J_t(\theta)$

$$J_t(\theta) = \log\sigma\left(u_o^T v_c\right) + \sum_{i=1}^{k} \mathbb{E}_{j\sim P(w)}\left[\log\sigma\left(-u_j^T v_c\right)\right] = \log\sigma\left(u_o^T v_c\right) + \sum_{j\sim P(w)} \left[\log\sigma\left(-u_j^T v_c\right)\right]$$

Distinguish the target word W_o from draws from the noise distribution $P_n(w)$ using logistic regression

Supplementary Topic:

Link GloVe with `word2vec`

➔ Training Objects

◆ GloVe

Local cost function:

$$l_G(w_i, c_j) = f(\#(w_i, c_j))\left(W_i \cdot C_j^T + b_{W_i} + b_{C_j} - \log \#(w_i, c_j)\right)^2$$

where f(x) is weighting function, shown as:

$$f(x) = \begin{cases} (x/x_{\max})^\alpha & x < x_{\max} \\ 1 & \text{otherwise} \end{cases}$$

Optimal solution:

$$W_i \cdot C_j^T = \log \#(w_i, c_j) - b_{W_i} - b_{C_j}$$

◆ SGNS

Local objective:

$$l_S(w_i, c_j) = \#(w_i, c_j) \log \sigma\left(W_i \cdot C_j^T\right) + k \cdot \#(w_i) \cdot \frac{\#(c_j)}{\Sigma_w \#(w)} \log \sigma\left(-W_i \cdot C_j^T\right)$$

where

$$\sigma(x) = \frac{1}{1+e^{-x}}$$

Optimal Solution:

$$\begin{aligned} W_i \cdot C_j^T &= PMI(w_i, c_j) - \log k \\ &= \log \#(w_i, c_j) - \log \#(w_i) - \log \#(c_j) + \log \Sigma_w \#(w) - \log k. \end{aligned}$$

Credit: Linking GloVe with word2vec, by Tianze Shi and Zhiyuan Liu (Tsinghua University)