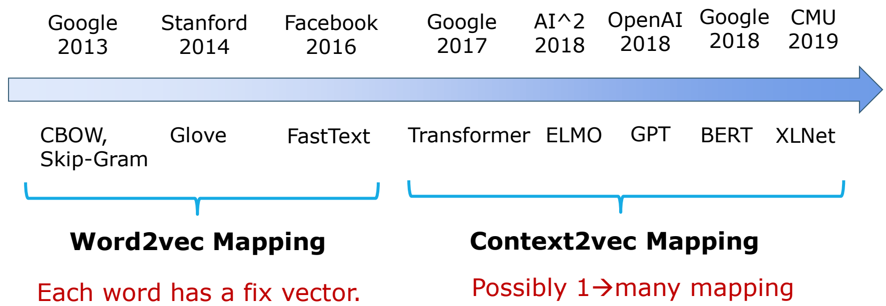
Deep Learning Techniques

DL1. Early Word Embedding (Word2Vec) Methods

1

Trend in Language Representation Learning

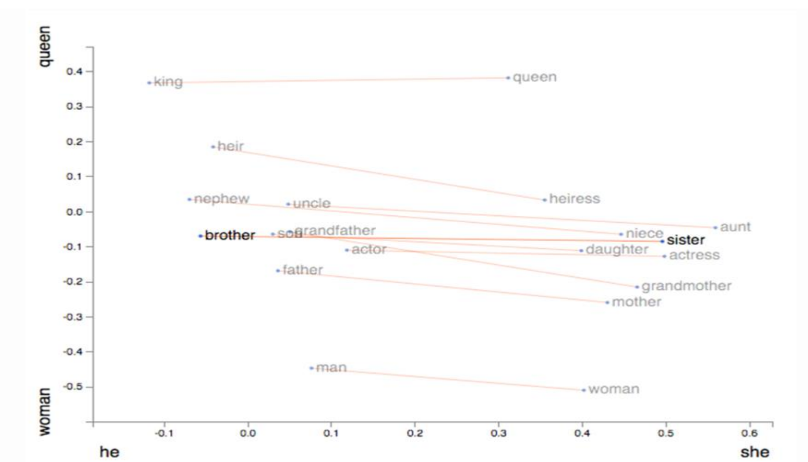


2

Motivation

- J.R. Firth's hypothesis (1957): *"You shall know a word by the company it keeps."*
- Word2vec mapping (2013 – 2016)
  - Finding a k-dimensional vector (e.g., k=300) for each word based on the words co-occurring with it in many documents.
  - Widely used in text mining and NLP tasks (machine translation, question answering, text classification, information retrieval, summarization, etc.)

Masculine-Feminine Vectors by GloVe (Projected to 2D via PCA)



- Each word is a vector (point).
- Each relation is a line.
- Parallel lines show some "analogy" among word pairs.

http://p.migdal.pl/2017/01/06/king-man-woman-queen-why.html

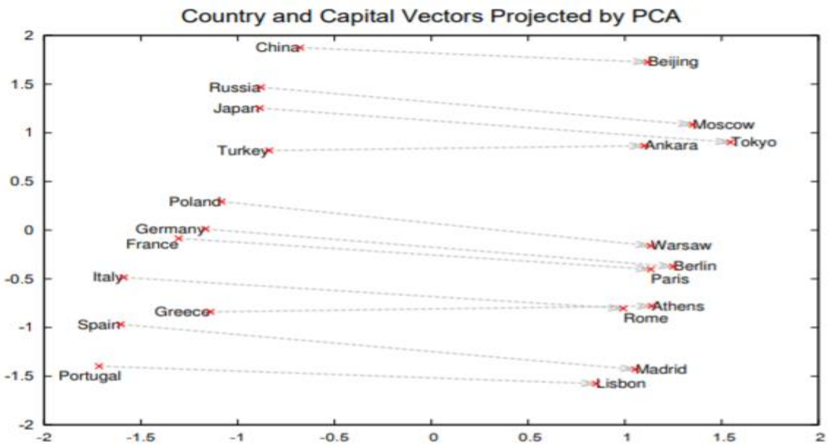


Figure 2: Two-dimensional PCA projection of the 1000-dimensional Skip-gram vectors of countries and their capital cities. The figure illustrates ability of the model to automatically organize concepts and learn implicitly the relationships between them, as during the training we did not provide any supervised information about what a capital city means.

The Word Analogy Task

- Given a word pair and a word in another pair, find the word for "?"

$$\mathcal{R}(\text{man, woman}) \approx \mathcal{R}\,(\text{king, ?})$$

- Given word embeddings, system finds the answer as

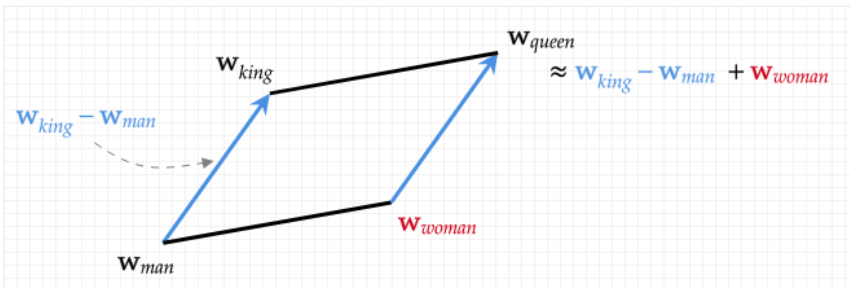
$$w_?^* = argmin_{w_?}\|(w_{man} - w_{woman}) + (w_{king} - w_?)\|$$

$$w_{man} - w_{woman} \approx w_{king} - w_?$$

$$w_? \approx w_{king} - w_{man} + w_{woman}$$

Word Analogy: A Geometrical View

- We can draw a parallelogram as



7

Neural Word Embedding Methods

- CBOW and Skip-Gram
- GloVe (Global Vectors for Word Representation)

8

Training Data Generation

- Consider a sliding window over a sequence of words as

  "we have classes every Tuesday and Thursday on machine learning …"

  w_{i-2} w_{i-1} w_i w_{i+1} w_{i+2}

  - The middle one is called the target word (w_i), and the surrounding ones ($\boldsymbol{w}_{<i,}\ \boldsymbol{w}_{>i}$) together are called the context of the target word.

- Apply the sliding window over a large corpus of text we obtain many <word, context> pairs as the training set.

CBOW vs. Skip-Gram

- **CBOW** (predicting each target word given its context)

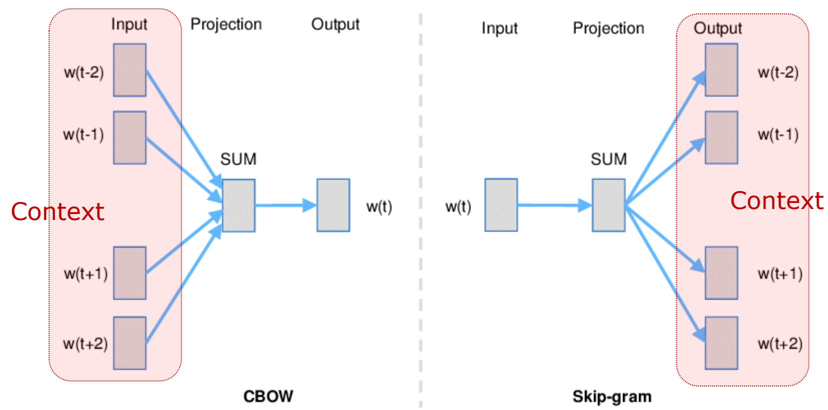
$$\max_{\theta} \sum_i logP_\theta(w_i |\ \underbrace{\boldsymbol{w}_{<i,}\ \boldsymbol{w}_{>i})}_{context\ \boldsymbol{c}_i} \qquad c_i = \{w_j : \mathrm{j} \in i \pm k\}$$

- **Skip-Gram** (predicting the context given a target word)

$$\max_{\theta} \sum_i logP_\theta(c_i|w_i) = \sum_i \sum_{w_j \epsilon c_i} logP_\theta\left(w_j|w_i\right)$$

- Both methods ignore the word order in the context window.

CBOW and Skip-Gram for word2vec



11

CBOW Architecture

- **Input**: Each input word is represented as one-hot vector $x_j \in \{0,1\}^V$(all the elements are 0 except one), where is the vocabulary size.
- **Hidden Layer**: context embedding $h \in R^N$ $(N \ll V)$ is calculated as

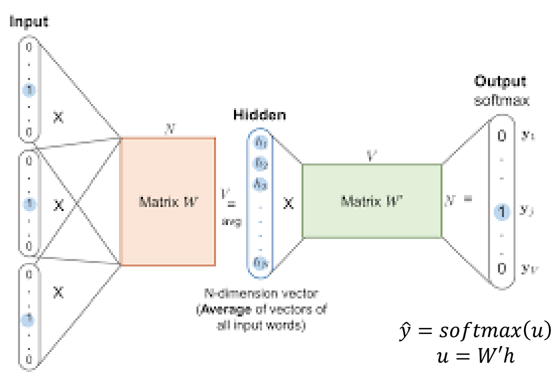
$$h = \frac{1}{|c|} \sum_{j \in c} W^T x_j$$

where $W \in R^{V \times N}$ is a matrix of learnable parameters.

- **Output Layer**: the predicted probabilistic distribution of candidate words as

$$\hat{y} = (\hat{y}_1, \hat{y}_2, \ldots, \hat{y}_V) = softmax(W'h)$$

where $W' \in R^{N \times V}$ is another matrix of learnable parameters.



$$\hat{y} = softmax(u)$$
$$u = W'h$$

12

CBOL with a Compact Input

- Input Layer (with a merged vector)

$$\boldsymbol{x} = \frac{1}{|c|}\left(\boldsymbol{x}_{w_1} + \boldsymbol{x}_{w_2} + \cdots + \boldsymbol{x}_{w_{|c|}}\right)$$

- Hidden layer

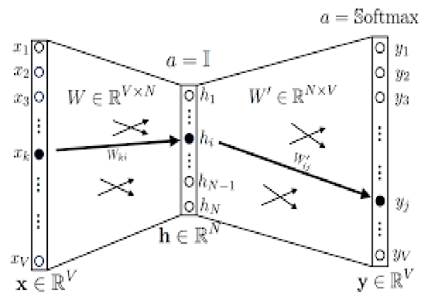
$$\boldsymbol{h} = W^T\ \boldsymbol{x}$$

$$W\ = \frac{1}{|c|}\left(W^T\boldsymbol{x}_{w_1} + W^T\boldsymbol{x}_{w_2} + \cdots + W^T\boldsymbol{x}_{w_{|c|}}\right)$$

- Model Parameters

$$\Theta = (W, W')$$

- Matrix W ∈ $\mathrm{R}^{\mathrm{V\times N}}$ contains all the word embeddings (each row is the embedding of a word).



$\hat{y} = softmax(u)$

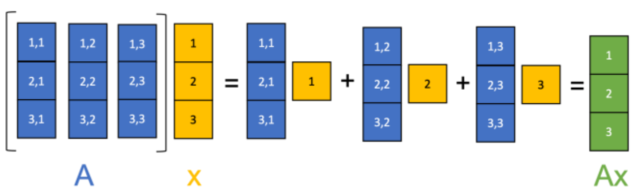
9/3/2024 @Yiming Yang, lecture on Deep Learning for Text Mining 13

13

Word-embedding matrix W

- Hidden layer

$$\boldsymbol{h} = W^T\ \boldsymbol{x} = \frac{1}{|c|}W^T(\boldsymbol{x}_{w_1} + \boldsymbol{x}_{w_2} + \cdots + \boldsymbol{x}_{w_{|c|}})$$



In CBOW, $A = W^T$ and x is the sum of a few one-hot vectors.

Ax picks a few columns of A to sum up (and then average over) for context embedding. Each column of W^T (each row of is W) is the embedding of a word.

9/3/2024 @Yiming Yang, lecture on Deep Learning for Text Mining 14

14

Training Process

- Denote Set: $\mathcal{D} = \{(x_i, y_i)\}$ context-word pairs.
- Model Initialization: Randomly initialize W and W'
- Model Update: For each pair $(x, y) \in \mathcal{D}$
  - Forward Propagation: Fix W and W', compute $h,\ u$ and $\hat{y}$ given x;
  - Backpropagation: Fix h, u and $\hat{y}$, update W and W' with mini-batch gradients.

Model Parameter Optimization

- Iterative update

$$W_{kl}{}^{(new)} := W_{kl}{}^{(old)} - \eta \nabla_{W_{kl}} \left(\frac{1}{|B|} \sum_{y_i \in B} l_\theta(\hat{y}_i, y_i) \right)$$

$$W'_{ij}{}^{(new)} := W'_{ij}{}^{(old)} - \eta \nabla_{W'_{ij}} \left(\frac{1}{|B|} \sum_{y_i \in B} l_\theta(\hat{y}_i, y_i) \right)$$

where $l_\theta\ (\hat{y}_i, y_i)$ is the loss on each training pair;

B is a randomly sampled mini-batch from the training set.

Loss Function (on a single training pair for simplicity)

- Cross entropy loss

$$l(\hat{y}, y) = -\sum_{j=1}^{V} y_j log \hat{y}_j = -\log \hat{y}_{j^*}$$

where j^* is the index of the target word in training pair (x_i, y_i).

- Example

$y = (\ 0 \quad 1 \quad 0 \quad 0 \quad 0)$ $\quad j^*=2$

$\hat{y} = (0.1 \quad 0.5 \quad 0.2 \quad 0.1 \quad 0.1)$

$L(\hat{y}, y) = -\log \hat{y}_2 = -\log 0.5$

9/3/2024 @Yiming Yang, lecture on Deep Learning for Text Mining 17

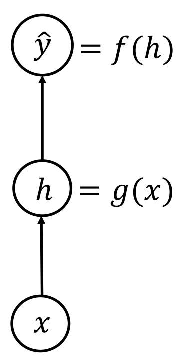
17

[L.P. Morency: CMU 11-777]

Gradient Computation

- Chain rule

$$\frac{df}{dx} = \frac{df}{dh}\frac{dh}{dx}$$



07/23/2017 @Yiming Yang, lecture on Deep Learning for Text Mining 18

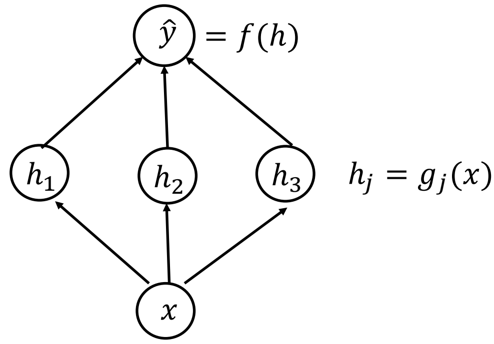
18

[L.P. Morency: CMU 11-777]

Gradient Computation (cont’d)

- Chain rule

$$\frac{df}{dx} = \sum_j \frac{\partial f}{\partial h_j} \frac{dh_j}{dx}$$



[L.P. Morency: CMU 11-777]

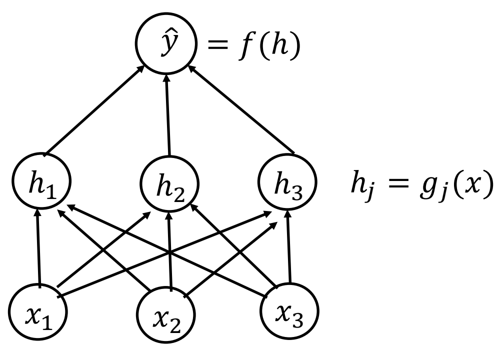
Gradient Computation (cont’d)

- Chain rule

$$\frac{\partial f}{\partial x_1} = \sum_j \frac{\partial f}{\partial h_j} \frac{\partial h_j}{\partial x_1}$$

$$\frac{\partial f}{\partial x_2} = \sum_j \frac{\partial f}{\partial h_j} \frac{\partial h_j}{\partial x_2}$$

$$\frac{\partial f}{\partial x_3} = \sum_j \frac{\partial f}{\partial h_j} \frac{\partial h_j}{\partial x_3}$$



Gradient Computation (cont'd)

- the gradient (scalar-by-vector)

$$\nabla_x f = \left[\frac{\partial f}{\partial x_1} \quad \frac{\partial f}{\partial x_2} \quad \frac{\partial f}{\partial x_3}\right]$$

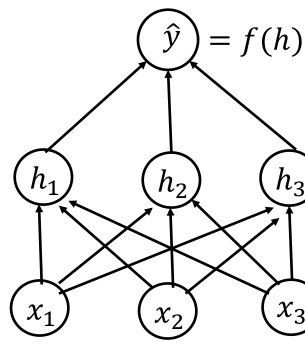
$$= \boxed{\nabla_{\boldsymbol{h}} \boldsymbol{f}} \boxed{\left(\frac{\partial \boldsymbol{h}}{\partial \boldsymbol{x}}\right)}$$

Gradient of f w.r.t. h

$$\left[\frac{\partial f}{\partial h_1} \quad \frac{\partial f}{\partial h_2} \quad \frac{\partial f}{\partial h_3}\right]$$

$$\begin{bmatrix} \frac{\partial h_1}{\partial x_1} & \frac{\partial h_1}{\partial x_2} & \frac{\partial h_1}{\partial x_3} \\ \frac{\partial h_2}{\partial x_1} & \frac{\partial h_2}{\partial x_2} & \frac{\partial h_2}{\partial x_3} \\ \frac{\partial h_3}{\partial x_1} & \frac{\partial h_3}{\partial x_2} & \frac{\partial h_3}{\partial x_3} \end{bmatrix}$$

Jacobian matrix of size $|h| \times |x|$



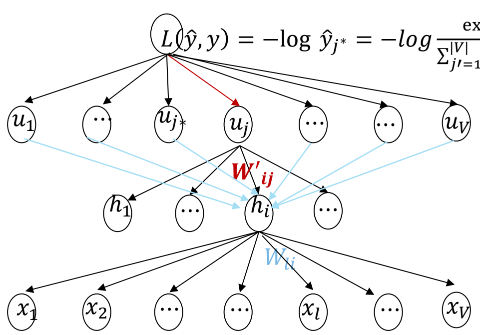
$h_j = g_j(x)$

@Yiming Yang, lecture on Deep Learning for Text Mining

21

Backpropagation in CBOL

$$L(\hat{y}, y) = -\log \hat{y}_{j^*} = -log \frac{\exp(u_{j*})}{\sum_{j'=1}^{|V|} \exp(u_{j'})}$$



$$\frac{\partial L}{\partial W'_{ij}} = \frac{\partial L}{\partial u_j} \cdot \frac{\partial u_j}{\partial W'_{ij}}$$

$$\frac{\partial L}{\partial W_{li}} = \sum_{j=1}^{V} \frac{\partial L}{\partial u_j} \frac{\partial u_j}{\partial h_i} \frac{\partial h_i}{\partial W_{li}}$$

$$u_j = W'_{1j} h_1 + W'_{2j} h_2 + \cdots + W'_{iN} h_N \qquad h_i = W_{1i}\, \bar{x}_1 + W_{2i}\, \bar{x}_2 + \cdots + W_{iV}\, \bar{x}_V$$

22

Backpropagation [Xin Rong, arXiv 2016]

- Partial derivative with respect to each element of W'

$$\frac{\partial L}{\partial W'_{ij}} = \frac{\partial L}{\partial u_j} \cdot \frac{\partial u_j}{\partial W'_{ij}}$$

$$\frac{\partial L}{\partial u_j} = \frac{\partial}{\partial u_j}\left(-log\frac{\exp(u_{j*})}{\sum_{j'=1}^{|V|}\exp(u_{j'})}\right) = \frac{\partial}{\partial u_j}\left(-u_{j*}\right) + \frac{\partial}{\partial u_j}\left(\log(\sum_{j'=1}^{V}\exp(u_{j'})\right)$$

$$= -\delta_j + \frac{\exp(u_j)}{\sum_{j'=1}^{V}\exp(u_{j'})} = -\delta_j + \hat{y}_j \qquad (1)$$

(δ_j is the Kronecker delta function, $\delta_j = 1$ if j=j*; otherwise $\delta_j = 0$)

$$\frac{\partial u_j}{\partial W'_{ij}} = \frac{\partial}{\partial W'_{ij}}\left(W'_{1j}h_1 + W'_{2j}h_2 + \cdots + W'_{ij}h_i + \cdots\right) = h_i \qquad (2)$$

- Combine (1) and (2), we have

$$\frac{\partial L}{\partial W'_{ij}} = (\hat{y}_j - \delta_j)h_i \qquad (3)$$

Backpropagation (cont'd)

- Partial derivative w.r.t. the elements of W

$$\frac{\partial L}{\partial W_{li}} = \frac{\partial L}{\partial h_i}\frac{\partial h_i}{\partial W_{li}} = \sum_{j=1}^{V}\frac{\partial L}{\partial u_j}\frac{\partial u_j}{\partial h_i}\frac{\partial h_i}{\partial W_{li}}$$

$$\frac{\partial L}{\partial u_j} = (\hat{y}_j - \delta_j) \qquad (1)$$

$$\frac{\partial u_j}{\partial h_i} = \frac{\partial}{\partial h_i}\left(W'_{1j}h_1 + W'_{2j}h_2 + \cdots + W'_{ij}h_i + \cdots\right) = W'_{ij} \qquad (2)$$

$$\frac{\partial h_i}{\partial W_{li}} = \frac{\partial}{\partial W_{li}}\left(W_{1i}x_1 + W_{2i}x_2 + \cdots + W_{li}x_l + \cdots\right) = x_l \qquad (3)$$

- Finally, $\frac{\partial L}{\partial W_{li}} = \sum_{j=1}^{V}(\hat{y}_j - \delta_j) \cdot W'_{ij}x_l$

Neural Word Embedding Methods

- ✓ CBOW and Skip-Gram
- GloVe (Global Vectors for Word Representation)

GloVe [Jeffrey Pennington et. al., EMNLP 2014]

- **Input data** $\boldsymbol{X} \in \boldsymbol{R}^{V \times V}$ (the matrix of weighted word co-occurrences in log scale)
  - Define each local context window as ± 10 words around center word i;
  - Weight word j in the window with $\frac{1}{k}$ if word j is k-words apart from word i;
  - Calculate X_{ij} as the global sum of the weight of word j in all the windows of word i;
  - Take the log scale of matrix X as $X_{ij} \xrightarrow{logscale} logX_{ij}$
- **Learnable matrix** $W \in R^{V \times N}$(word embeddings) and **bias vector** $\boldsymbol{b} \in R^V$
  - Each row of W of is the embedding of a word, denoted as w_i;
  - Each element of $\boldsymbol{b}$ is the bias term of a word, denoted as b_i.

GloVe [Jeffrey Pennington et. al., EMNLP 2014]

- **Training Objective**

$$\min_{W,b} \sum_{i,j=1}^{m} f(X_{ij})\big(\underbrace{w_i^T w_j + b_i + b_j}_{\hat{X}_{ij}} - X_{ij}\big)^2$$

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

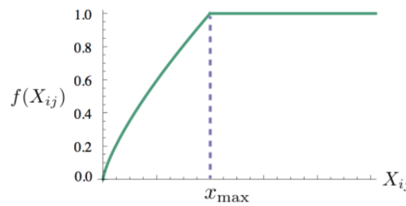


Figure 1: Weighting function f with $\alpha = 3/4$.

- **Algorithm**
  - stochastic gradient descent

27

Differences of GloVe compared to CBOW

- Local contextual word is weighted by $1/K$ (sensitive to word position)
- Giving more weights to larger cells (common word pairs) in training
- Despite the name, GloVe (Global Vectors for word representation) is still based on the local context around each target word
- In fact, all matrix factorization (MF) methods (such as SVD, BPR, etc.) can be applied to the GloVe's matrix to produce the word-embedding matrix $W \in R^{V \times N}$ (with $N \ll V$).

28

Evaluation in word analogy tasks
[Jeffrey Pennington et. al., EMNLP 2014]

- Dataset contains 19,544 questions, divided into two subsets
  - Semantic Task: "Athens is to Greece as Berlin is to __?"
  - Syntactic Task: "dance is to dancing as fly is to __?"
- Input question takes the form of "a is to b as c is to __?"
- System ranks all the candidate words based on the cosine similarity of each word with respect to $w_? = w_b - w_a + w_c$ and select the top candidate.

Results [Jeffrey Pennington et. al., EMNLP 2014]

- Baseline Methods
  - HPCA: PMI version of LSA (PCA) [10]
  - vLBL, ivLBL: log-bilinear model [9]
  - SG: skip gram (another variant of w2v)
  - CBOW: continuous bag-of-words
  - SVD-S: take SVD of $\sqrt{X_{trunc}}$
  - SVD-L: take SVD of $\log(1 + X_{trunc})$
- Metric: Accuracy
- Size: number of tokens in training set

Model	Dim.	Size	Sem.	Syn.	Tot.
ivLBL	100	1.5B	55.9	50.1	53.2
HPCA	100	1.6B	4.2	16.4	10.8
GloVe	100	1.6B	67.5	54.3	60.3
SG	300	1B	61	61	61
CBOW	300	1.6B	16.1	52.6	36.1
vLBL	300	1.5B	54.2	64.8	60.0
ivLBL	300	1.5B	65.2	63.0	64.0
GloVe	300	1.6B	80.8	61.5	70.3
SVD	300	6B	6.3	8.1	7.3
SVD-S	300	6B	36.7	46.6	42.1
SVD-L	300	6B	56.6	63.0	60.1
CBOW†	300	6B	63.6	67.4	65.7
SG†	300	6B	73.0	66.0	69.1
GloVe	300	6B	77.4	67.0	71.7
CBOW	1000	6B	57.3	68.9	63.7
SG	1000	6B	66.1	65.1	65.6
SVD-L	300	42B	38.4	58.2	49.2
GloVe	300	42B	**81.9**	**69.3**	**75.0**

Input matrices for the SVD Baselines
[Jeffrey Pennington et. al., EMNLP 2014]

- "For the SVD baselines, we generate a truncated matrix X_{trunc} which retains the information of how frequently each word occurs with only the top 10,000 most frequent words."
- SVD: $X_{trunc} \in R^{V \times 10000}$ with $X_{trunc}[\mathrm{i},\mathrm{j}] := freq(j|i)$
- SVD_S: with $X_{SVD\_S}[\mathrm{i},\mathrm{j}] \coloneqq \sqrt{X_{trunc}[\mathrm{i},\mathrm{j}]}$
- SVD_L: with $X_{SVD\_L}[\mathrm{i},\mathrm{j}] \coloneqq \log(1 + X_{trunc}[\mathrm{i},\mathrm{j}])$

Concluding Remarks

- CBOW and SkipGram treat local context as a set of words (ignoring word order) and learns the word embeddings with a one-hidden-layer neural network.
- GloVe use a matrix to globally aggregates local co-occurrence counts (with proximity weights) and learns word embeddings via gradient descent.
- Those methods produces a fixed embedding for each word that cannot differentiate word meanings under different contexts
- Another limitation of those methods is that the embeddings are not task-driven.
- We will discuss more in the remaining DL lectures about dynamic word embedding

References

1. Distributed representations of words and phrases and their compositionality. Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, Jeffrey Dean, NIPS 2013
2. Efficient Estimation of Word Representations in Vector Space. Tomas Mikolov, Kai Chen, Greg Corrado, Jeffrey Dean ICLR Workshop 2013
3. Xin Rong: word2vec Parameter Learning Explained. CoRR abs/1411.2738 (2016)
4. GloVe: Global Vectors for Word Representation. Jeffrey Pennington, Richard Socher, Christopher D. Manning. EMNLP 2014