Advanced Word Vector

Speaker : 蘇佳益

Advisor : 陳聰毅

國立高雄科技大學建工校區電子工程系

https://github.com/chiayisu/Artificial_Intelligence_Course

Citation

- This material mainly summarizes the lesson learned from Stanford University CS224N Lecture 2.

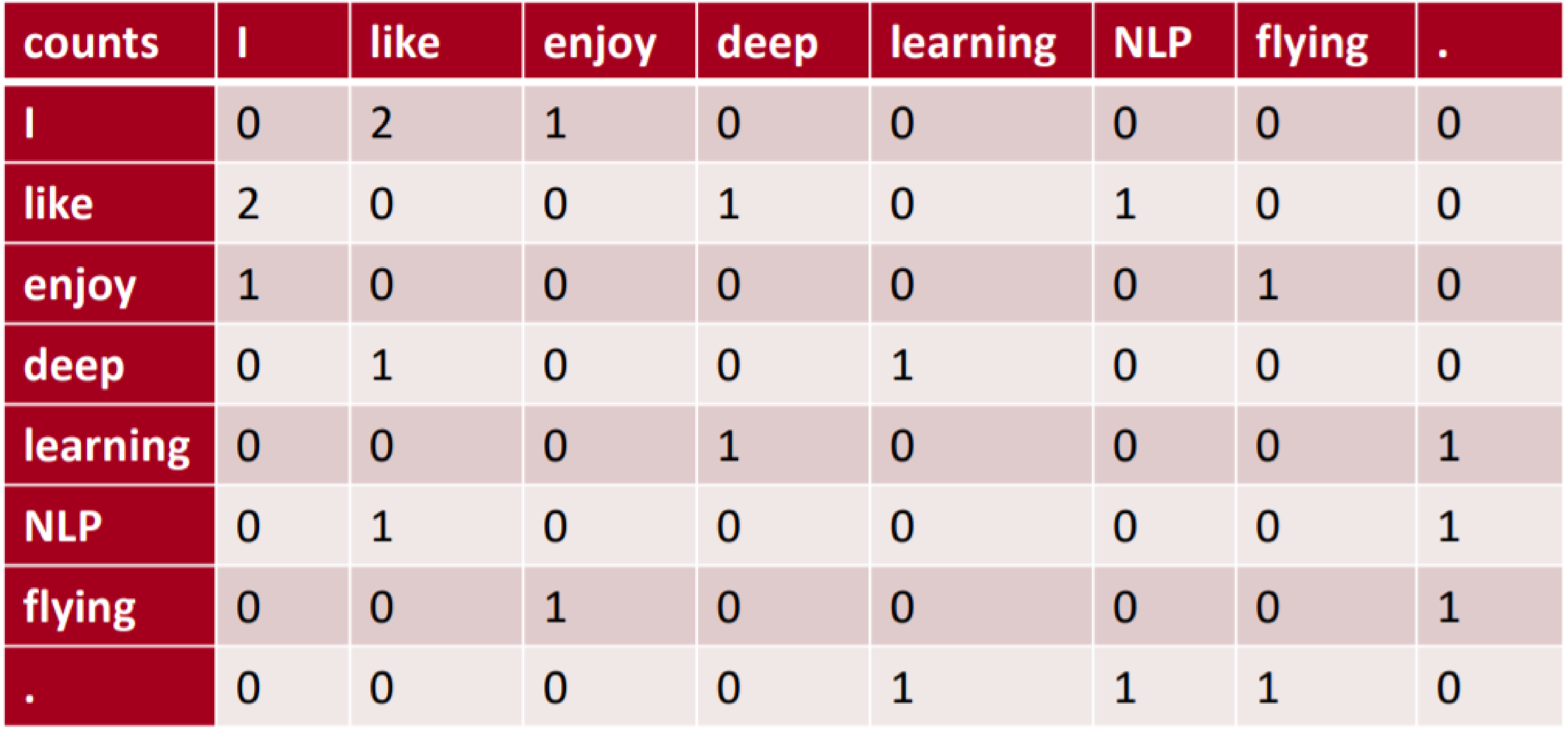
Agenda

- Why Not Co-Occurrence Windows?
- GloVe: Global Vectors for Word Representation
- Evaluation of Word Vector
- Linear Algebraic Structure of Word Senses, with Applications to Polysemy

Why Not Co-Occurrence Windows?

- Some people may think “why not using co-occurrence windows?”
- Main Reason: several disadvantages
- We will explore those disadvantages
- Suppose we have the following sentence:
  - I like deep learning.
  - I like NLP.
  - I enjoy flying.
- We will use windows size 1. (Usually 5-10)

Co-Occurrence Matrix



counts	I	like	enjoy	deep	learning	NLP	flying	.
I	0	2	1	0	0	0	0	0
like	2	0	0	1	0	1	0	0
enjoy	1	0	0	0	0	0	1	0
deep	0	1	0	0	1	0	0	0
learning	0	0	0	1	0	0	0	1
NLP	0	1	0	0	0	0	0	1
flying	0	0	1	0	0	0	0	1
.	0	0	0	0	1	1	1	0

Problems

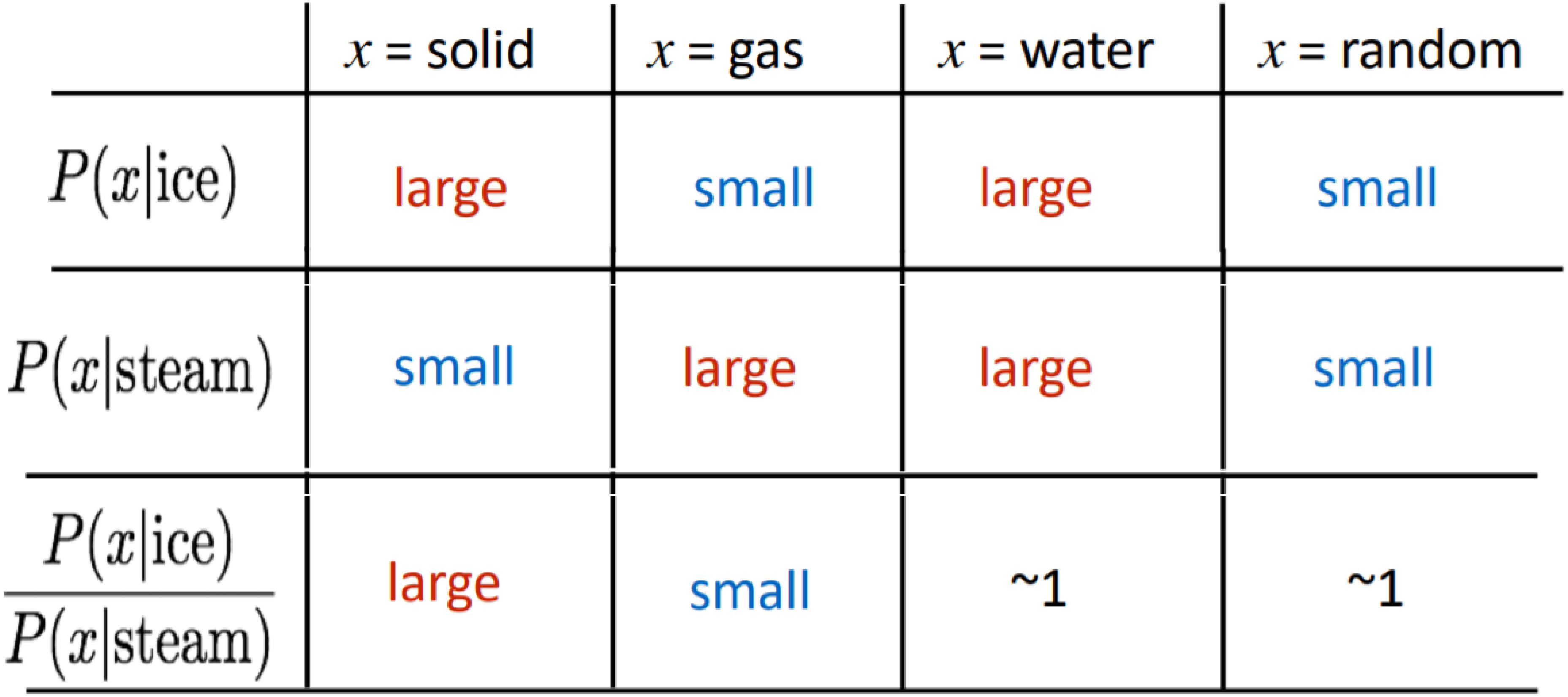
- Storage : As vocabulary size increases, we need more storage. Google usually uses 1 billion vocabularies to train the model.
- Sparsity : As the matrix shown in last slide, we see many zero elements in the matrix.
- Not Robust Enough : The sparsity problem may also make our model not robust.
- Solution?
    - Dimension Reduction. (Still not work well)

Glove: Global Vectors for Word Representation

Glove (Pennington et al.): Introduction

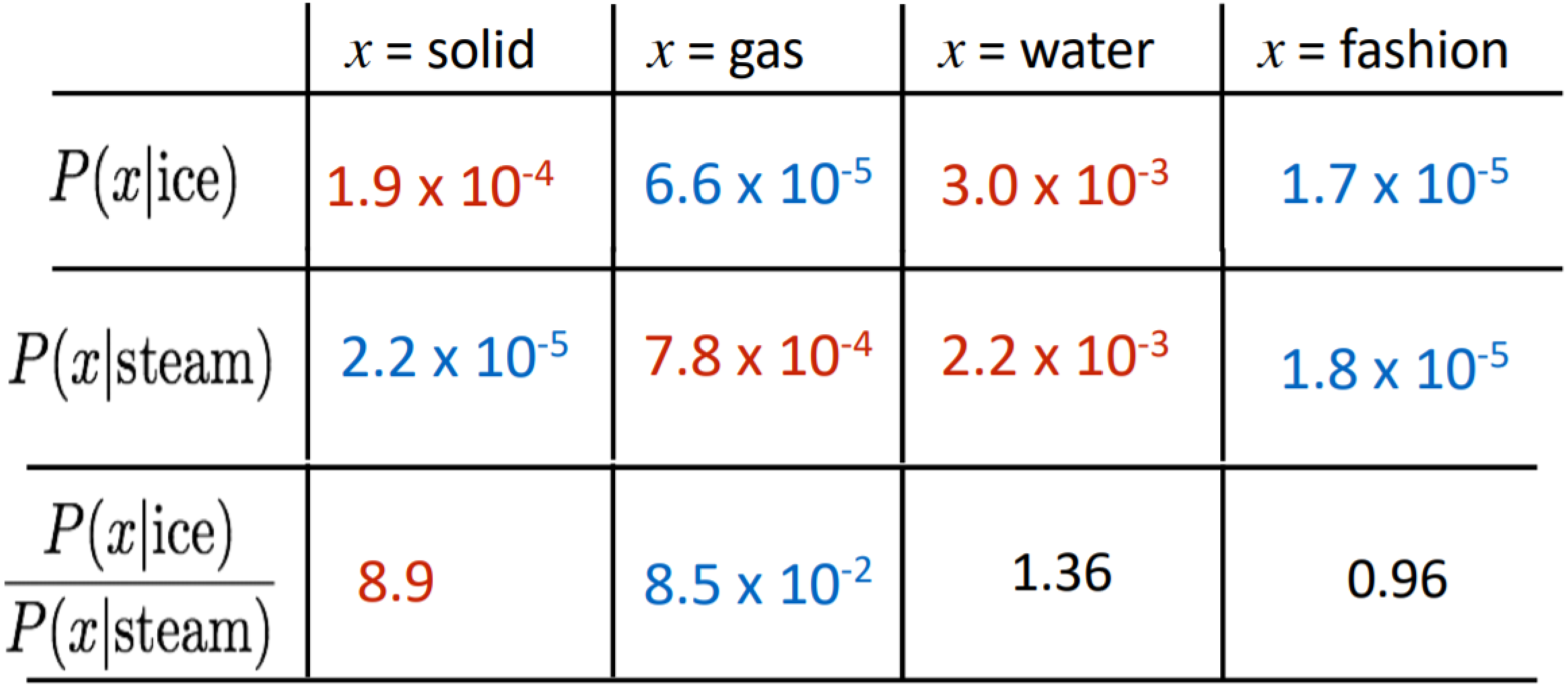
- Two different classes of methods for word embedding
  - count-based method (e.g. LSA)
  - windows-based models (e.g. Skip-Gram)
- Count-Based Method
  - makes good use of global information
  - However: Only performs well on similarity tasks
- Windows-Based Models
  - Fail make use of global information
  - However: perform well on many tasks
- GLOVE is trained on global information. It outperforms current model as well.

Glove Model: Motivation



	x = solid	x = gas	x = water	x = random
$P(x\|\text{ice})$	large	small	large	small
$P(x\|\text{steam})$	small	large	large	small
$\frac{P(x\|\text{ice})}{P(x\|\text{steam})}$	large	small	~1	~1

Glove Model: Motivation (CONT.)



	x = solid	x = gas	x = water	x = fashion
$P(x\|\text{ice})$	1.9 x 10^{-4}	6.6 x 10^{-5}	3.0 x 10^{-3}	1.7 x 10^{-5}
$P(x\|\text{steam})$	2.2 x 10^{-5}	7.8 x 10^{-4}	2.2 x 10^{-3}	1.8 x 10^{-5}
$\frac{P(x\|\text{ice})}{P(x\|\text{steam})}$	8.9	8.5 x 10^{-2}	1.36	0.96

Introduction to Notation

- X : Matrix of word co-occurrence count
- X_{ij} : Number of times word j occurs in the context of word i
- X_i : Number of times any word appears in the context of word i which is equal to $\sum_k X_{ik}$
- P_{ij} : Probability of word j appears in the context of word i which is equal to $P(j|i) = \frac{X_{ij}}{X_i}$

Summary of above Table

- Suppose we are interested in Thermodynamics. Let i = ice, j=steam, k = various probe words.
- k related to ice but not steam whose $\frac{P_{ik}}{P_{jk}}$ would be large, as we show in the table.
- k related to steam but not ice whose $\frac{P_{ik}}{P_{jk}}$ would be small, as we showed in the table.
- k related to both or neither whose $\frac{P_{ik}}{P_{jk}}$ would be approximate to 1, as we showed in the table.
- According to above summary, the paper (Pennington et al.) showed the importance of the co-occurrence window.

Formula Derivative

- $F\left(w_i, w_j, \widetilde{w}_k\right) = \frac{P_{ik}}{P_{jk}}$, where w are word vectors and $\widetilde{w}$ will be discussed later. F depends on a lot of unspecific factors.
- $F\left(w_i - w_j, \widetilde{w}_k\right) = \frac{P_{ik}}{P_{jk}}$, since the vector space is linear, we can do the vector difference. For now, F only depends on the difference of the two target words.
- $F\left((w_i - w_j)^T \widetilde{w}_k\right) = \frac{P_{ik}}{P_{jk}}$, the right-hand side is scalar, while the left-hand side is vector. It would obfuscate the linear structure that we try to capture so we do the dot product on the left-hand side.
- The F should satisfy the homomorphism so the formula should equal to

$F\left((w_i - w_j)^T \widetilde{w}_k\right) = \frac{F(w_i^T \widetilde{w}_k)}{F(w_j^T \widetilde{w}_k)}$, where $F\left(w_i^T \widetilde{w}_k\right) = P_{ik} = \frac{X_{ik}}{X_i}$

Formula Derivative

- The solution to above step is F=exp so our formula would be

  $w_i^T \widetilde{w}_k = \ln P_{ik} = \ln X_{ik} - \ln X_i$
- The above formula would exhibit the exchange symmetry if we don't include the $\ln X_i$. Fortunately, $\ln X_i$ is independent of k so we can absorb it into bias b_i. Also, the formula will be add additional bias $b_k$for $\widetilde{w}_k$. So our formula would be

  $w_i^T \widetilde{w}_k + b_k + b_i = \ln X_{ik}$ (1)
- Problem
  - log will be divergent whenever the argument is 0.
- Resolution
  - $\ln X_{ik}$ +1.
  - However: this will be the same as the co-occurrence algorithm. Each words have the same weights.
  - Therefore: this algorithm will be based on (1)

Formula Derivative

- The paper (Pennington et al.) employs a way called weighted least square regression model and a weighted function $f\left(X_{ij}\right)$ to resolve the problem. Our cost function is, therefore, as follows.

$J = \sum_{i,j}^{V} f\left(X_{ij}\right)\left(w_i^T \widetilde{w}_k + b_k + b_i - \ln X_{ik}\right)^2$, where the V is the size of vocabulary.

- $f\left(X_{ij}\right)$ should obey the following rules

1. f(0) is 0.
2. f(x) should be non-decreasing so that the low frequency words will not be overweighted.
3. f(x) should not be too large so that the words will not be overweighted as well.

Formula Derivative

- Although the tons of the functions satisfy the rule as paper mentioned, the function that works well is as follows,

$$f(x) = \begin{cases} \left(\frac{x}{x_{max}}\right)^{\propto} & \text{if x < } x_{max} \\ 1, otherwise \end{cases}$$

- According to the experiment, the paper found $\propto$ equals $\frac{3}{4}$ has the best result. Interestingly, it is consistent with the experiment in the word2vector paper (Mikolov et al.)

Relationship to Other Model

- Recall that in the word vector model (Mikolov et al.), they employed softmax to compute the probability of the word j appears in the context word i. The formula is as follow.

$$Q_{i,j} = \frac{\exp\left(u_j^T \hat{v}_i\right)}{\sum_{j=1}^{|V|} \exp\left(u_j^T \hat{v}_i\right)}$$

← The most costly one

- The global cross-entropy is calculated as follow :

$$J = -\sum_{i \in corpus} \sum_{j \epsilon context} \ln Q_{i,j}$$

- Since both word i and j probably appear multiple times in the corpus, we can group together in consideration of the efficiency.

$J = \sum_{i=1}^{W} \sum_{j=1}^{W} X_{ij} \ln Q_{i,j}$, where X is a co-occurrence matrix.

Relationship to Other Model

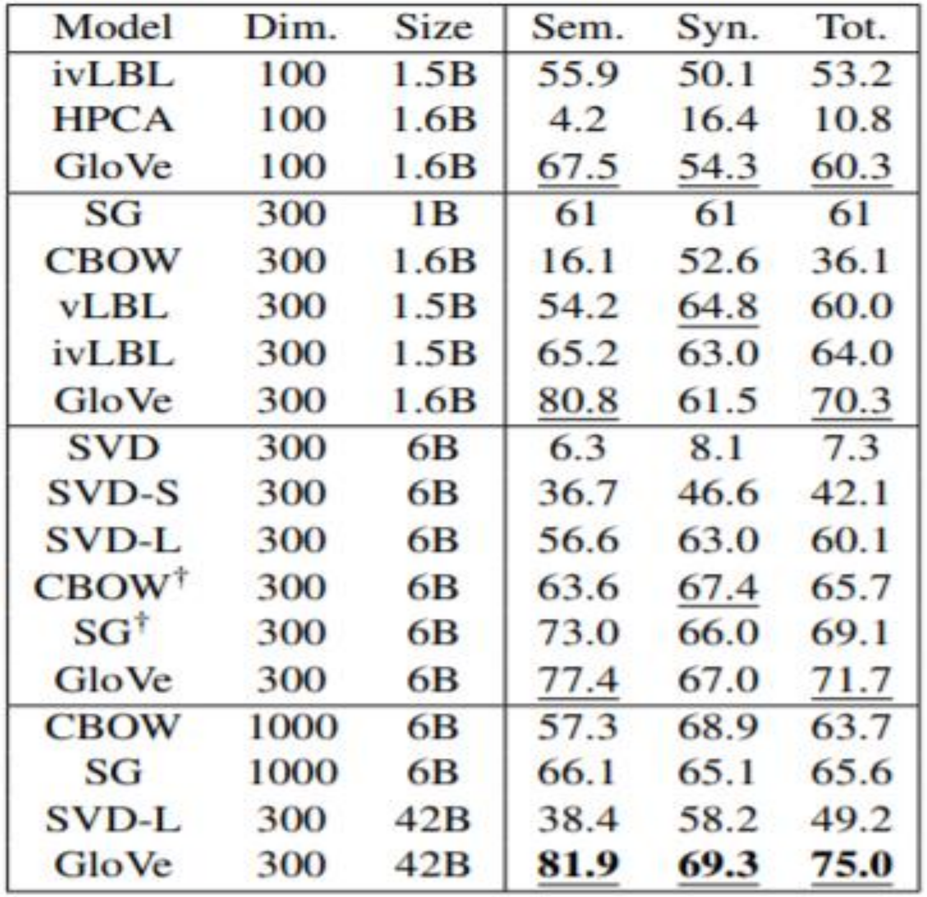
- Disadvantage of the cross-entropy loss
    - Computational Costly
    - Reason: requires to normalize
- The paper (Pennington et al.) employed the least square objective so that we don’t need to normalize. The formula is as follows :

$J = \sum_{i=1}^{w} \sum_{j=1}^{w} X_{ij} \left(\widetilde{P_{ij}} \widetilde{Q_{i,j}} \right)^2$, where $\widetilde{P_{ij}}$=X_{ij} and $\widetilde{Q_{i,j}}$ = $\exp\left(u_j^T \hat{v}_i \right)$ which are all unnormalized term.

Word Analogy Explanation

- Two types of word analogy problems
  - Semantics
  - Syntax
- Semantic Problem
  - a is to b as c is to _
  - San Francisco is to California as Chicago is to _
- Syntactic Problem
  - Good is to better as bad is to _
- These questions are answered by using
  - $w_b - w_a + w_c$ according to cosine similarity

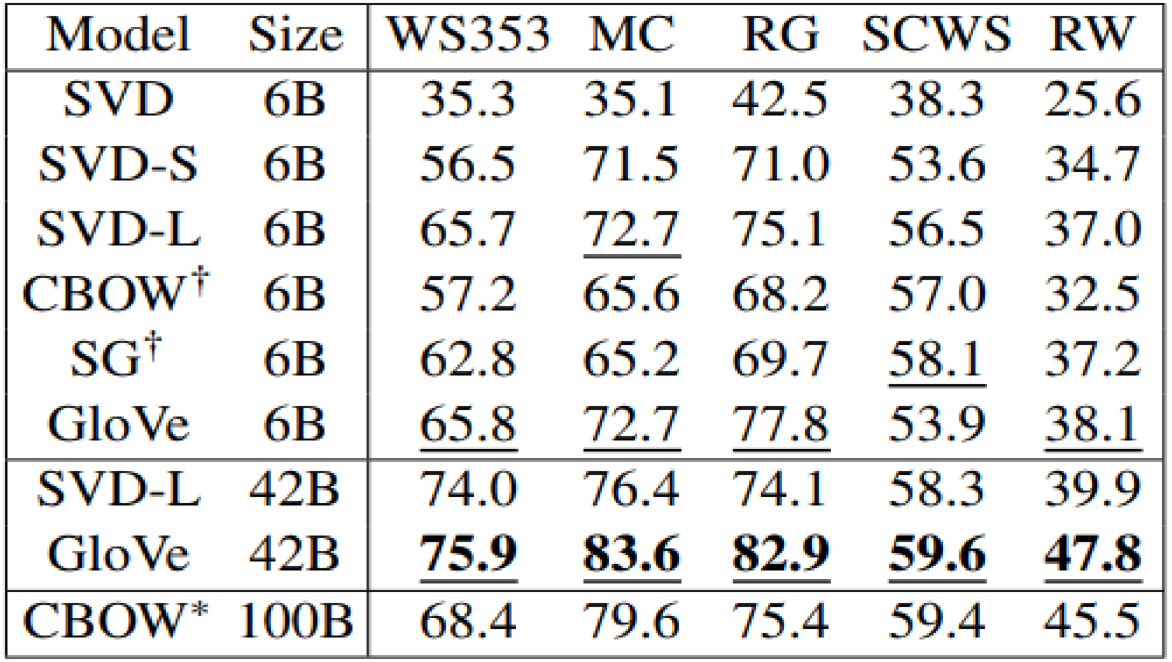
Experiment on Word Analogies



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**

Word Similarity

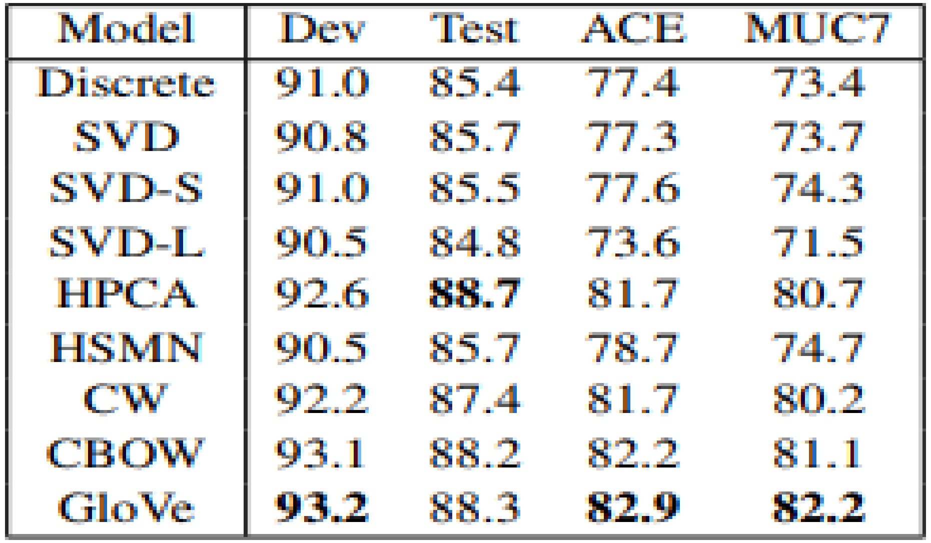
- Although word analogy is a major task, the paper also experiments the task in word similarity. The table below is the result of the experiment.



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

Name Entity Recognition

- NER Task in this experiment
    - Contains four types of entities: person, location, organization, and miscellaneous
    - Adopts BIO2 annotation standard
    - The preprocessing and setup are the same as the paper (Wang et al., 2013)
    - Moreover, 50-dimensional vectors are added for each word of a five-word context as continuous features
    - The model is trained with CRF.



Model	Dev	Test	ACE	MUC7
Discrete	91.0	85.4	77.4	73.4
SVD	90.8	85.7	77.3	73.7
SVD-S	91.0	85.5	77.6	74.3
SVD-L	90.5	84.8	73.6	71.5
HPCA	92.6	**88.7**	81.7	80.7
HSMN	90.5	85.7	78.7	74.7
CW	92.2	87.4	81.7	80.2
CBOW	93.1	88.2	82.2	81.1
GloVe	**93.2**	88.3	**82.9**	**82.2**

Corpora and Training Detail

- Corpora
  - 2010 wiki (1 billion tokens)
  - 2014 wiki (1.6 billion tokens)
  - Gigaword 5 (4.3 billion tokens)
  - Combination of the Gigaword5 + and wiki 2014 (6 billion tokens)
- Tokenization
  - Stanford Tokenizer
- Co-Occurrence Matrix X
  - In this step, the paper used decreasing weighted function which means the d word apart will contribute 1/d to the total count.

Corpora and Training Detail

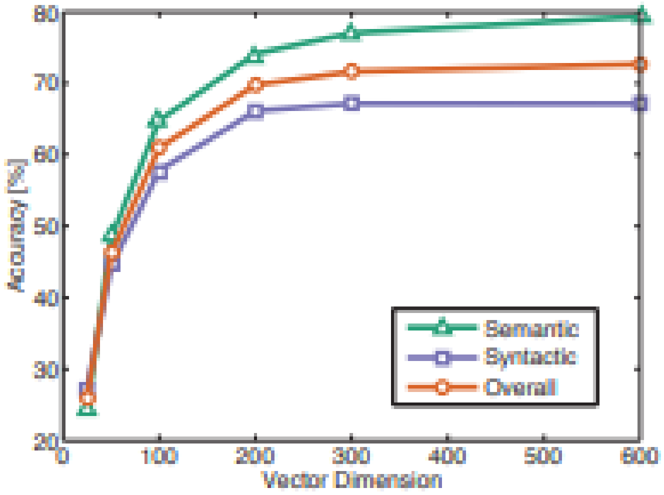
- Parameter Setting
  - $x_{max} = 100$
  - $\alpha = 3/4$
  - Learning rate = 0.05
  - Dimension is less that 300
  - The windows size is 10 to the right and left.

Discussion on $\widetilde{W}$ and W

- Two parameters are tantamount except their random initialization.
- Why add $\widetilde{W}$?
  - Because training multiple instances and combining the result can help to reduce the overfitting and noise.
- Thus: Pennington et al. use $\widetilde{W}$ + W as the word vector.

Model Analysis: Vector Length and Context Size

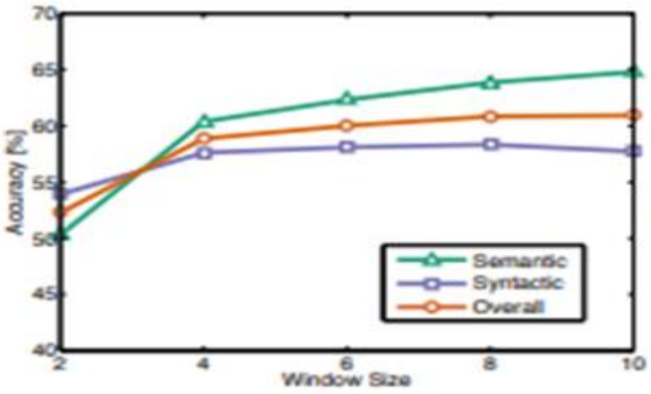
- The graph below shows that the vector larger than 200 dimensions will be smoothing.



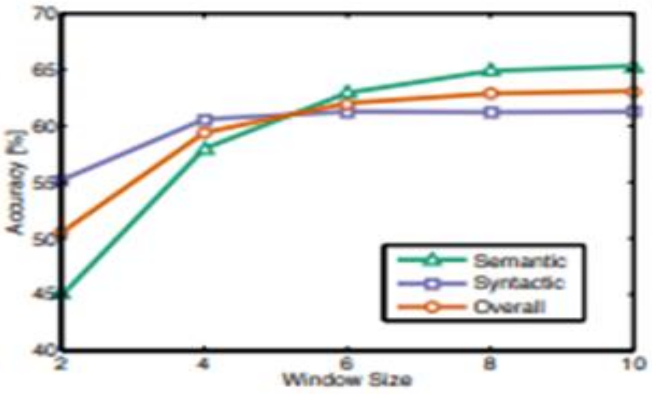
(a) Symmetric context

Model Analysis: Vector Length and Context Size

- The graphs below demonstrate
  - Syntactic task performs better on small and asymmetric windows
  - Semantic task performs better on larger window size
- it is better to take global information into account on semantic task.



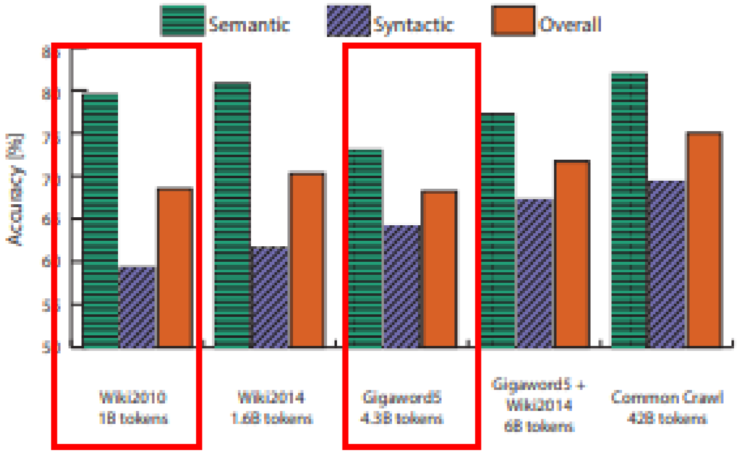
(b) Symmetric context



(c) Asymmetric context

Model Analysis: Corpus Size

- Accuracy
  - Larger corpus size yields higher accuracy on syntactic tasks
  - However: Semantic tasks don’t always follow this rule
- Example
  - Accuracy of larger Giagaword corpus doesn’t have higher accuracy than smaller wiki corpus
  - Reason: Giagaword (news) don’t assimilate new knowlege



Evaluation of Word Vector

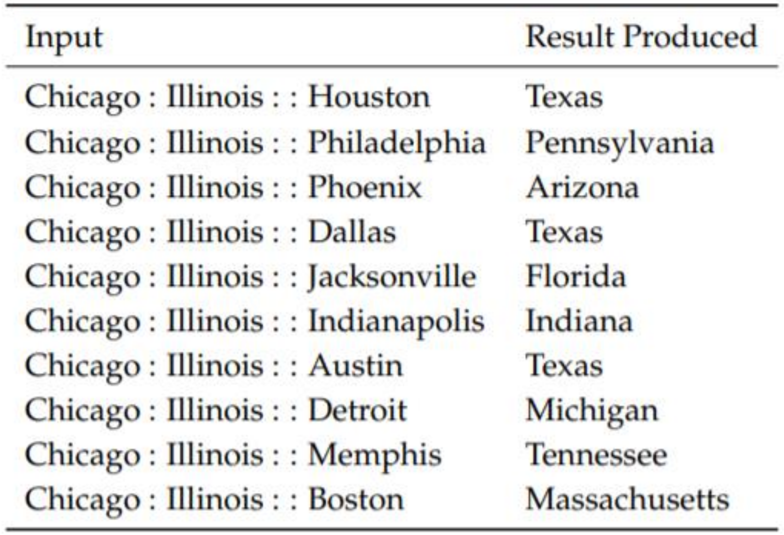
Methods of Evaluation

- So far, we don’t know how to evaluate word vector.
- Two Methods to Evaluate Word Vector
    - Intrinsic Evaluation
    - Extrinsic Evaluation

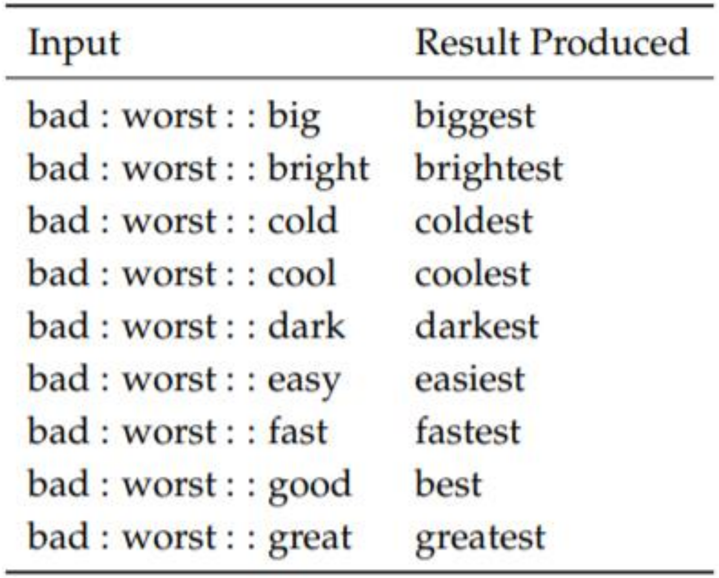
Intrinsic Evaluation

- Intrinsic Evaluation
  - Evaluate word vector on sub-system
  - Fast to compute
  - Help to understand sub-system
  - Need the correlation with the real task.
- Reason to use Intrinsic Evaluation
  - costly to evaluate the word vector with the entire system e.g. QA system

Intrinsic Evaluation – Word Analogy



Input	Result Produced
Chicago : Illinois : : Houston	Texas
Chicago : Illinois : : Philadelphia	Pennsylvania
Chicago : Illinois : : Phoenix	Arizona
Chicago : Illinois : : Dallas	Texas
Chicago : Illinois : : Jacksonville	Florida
Chicago : Illinois : : Indianapolis	Indiana
Chicago : Illinois : : Austin	Texas
Chicago : Illinois : : Detroit	Michigan
Chicago : Illinois : : Memphis	Tennessee
Chicago : Illinois : : Boston	Massachusetts

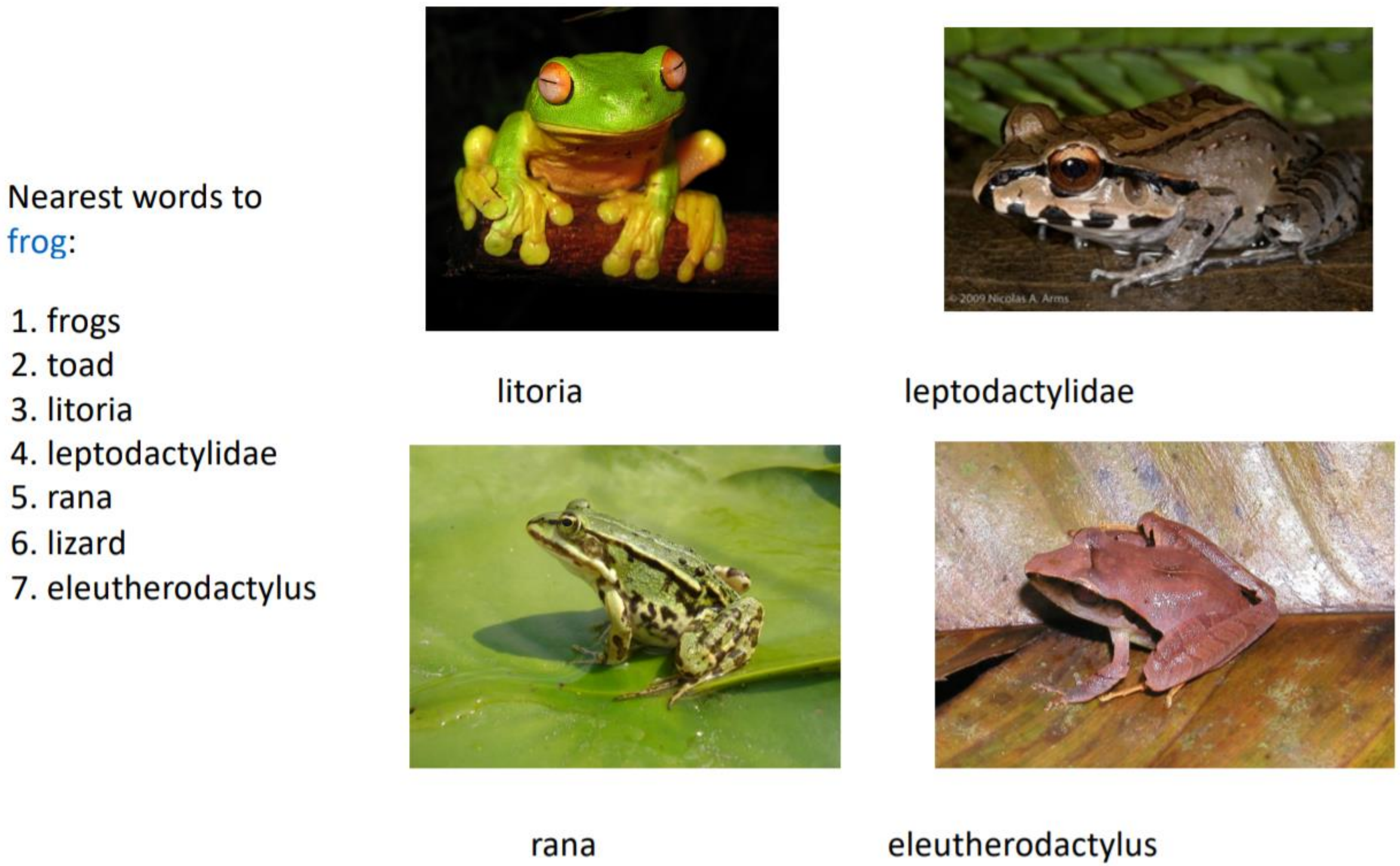


Input	Result Produced
bad : worst : : big	biggest
bad : worst : : bright	brightest
bad : worst : : cold	coldest
bad : worst : : cool	coolest
bad : worst : : dark	darkest
bad : worst : : easy	easiest
bad : worst : : fast	fastest
bad : worst : : good	best
bad : worst : : great	greatest

Extrinsic Evaluation

- Extrinsic evaluation
  - Evaluate word vector on real task
  - Slow to compute (opposite to intrinsic)
  - Hard to evaluate because it’s hard to know which system is problematic.

Result from Stanford Power Point



Nearest words to frog:

1. frogs
2. toad
3. litoria
4. leptodactylidae
5. rana
6. lizard
7. eleutherodactylus

litoria

leptodactylidae

rana

eleutherodactylus

Vector Space Model (VSM)

Information Retrieve

- Definition of Information Retrieve
- The difference between “Structured Data” and “Unstructured Data”
- Algorithm for Information Retrieve
  - Boolean Retrieve
  - Vector Space Model
- What’s the difference between Boolean Retrieve and Vector Space Model?
  - The Boolean Retrieve only tells us “if the documents contain key words that we type.” The Vector Space Model tells us “the similarity between our documents and query.”

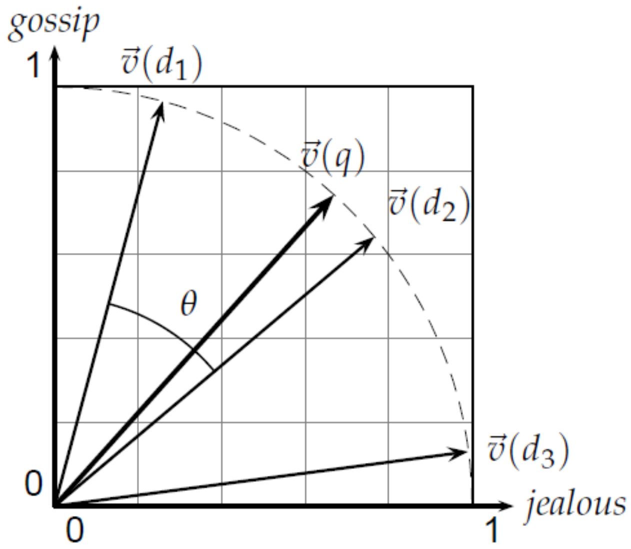
Vector Space Model: Introduction

- How do we know the similarity between two documents?
- Definition of Vector Space Model
  - The vector Space Model represents a set of document as vectors in a common vector space.
- Application of Vector Space Model
  - Information Retrieve
  - Text Classification
  - Text Clustering

a set of features

Vector Space Model: Cosine Similarity

- Cosine Similarity



- Formula of Cosine Similarity

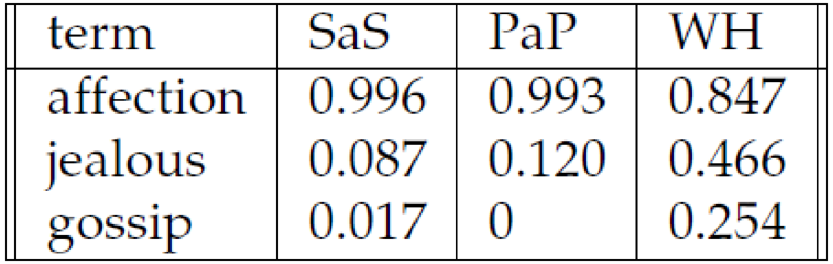
$$sim(d_1, d_2) = \frac{\overrightarrow{v_1}\,(d_1) * \overrightarrow{v_2}(d_2)}{|\overrightarrow{v_1}\,(d_1)|\,|\overrightarrow{v_2}(d_2)|}$$

Vector Space Model: Cosine Similarity

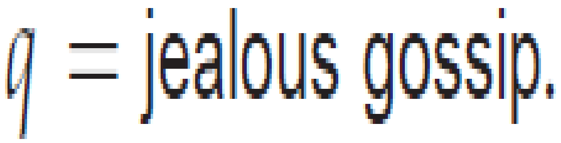
- the numerator is known as dot product(inner product). It is defined as follows.
  - $\vec{x} * \vec{y} = \sum_{i=1}^{M} x_i y_i$
- the denominator is known as Euclidean Length. The Euclidean Length of d is defined as follows.
  - $\sqrt{\sum_{i=1}^{M} \overrightarrow{V_i}^2 (d)}$
- the normalization of Cosine Similarity
  - We can also normalize $\overrightarrow{v_1}\,(d_1)$ to unit vector. Thus, we only need to calculate the dot product for Cosine Similarity. The formula will be as follows.
  - $sim(d_1, d_2) = \overrightarrow{v_1}\,(d_1) * \overrightarrow{v_2}(d_2)$
  - the length of unit vector is 1.

Vector Space Model: Example

- Suppose we have the documents and query below. What is the most similar document for the query below.



term	SaS	PaP	WH
affection	0.996	0.993	0.847
jealous	0.087	0.120	0.466
gossip	0.017	0	0.254



q = jealous gossip.

- Similarity between each documents
  - WH – 0.509
  - PaP – 0.085
  - SaS – 0.074

the most similar document

Linear Algebraic Structure of Word Senses, with Applications to Polysemy

Introduction

- Recent non-linear word embedding methods (Mikolov et al.; Pennington et al.)
  - obfuscate different senses of polysemous words
- However: Arora et al. showed that word senses
  - Consist in linear superposition within word embedding.
  - Satisfy the formula bellow in modern word vectors
- $v_{word} \approx \alpha_1 * v_{word1} + \alpha_2 * v_{word2} + \cdots$

Gaussian Walk Model

- Arora et al.,2016 hypothesizes that there’s micro-topics (“What is being talked about”) called discourse in the corpus.
- Thus: The Gaussian Walk Model is modified from the paper (Arora et al.,2016).
- The parameters include a vector $v_w \in R^d$ for each word w.
- The discourse is acquired from a Gaussian distribution with mean 0 and co-variance $\sum$.

Gaussian Walk Model (CONT.)

- A window of n words are, thus, generated as follows:

$P_r[w_1, w_2, \ldots, w_n|c] = \prod_{i=1}^{n} P_r[w_i|c]$

$P_r[w_i|c] = \boxed{exp\left(c * v_{w_i}\right)/Z_c}$, where $Z_c = \sum_w \exp(\langle v_w, c\rangle)$ - (1)

Because of Gaussian Distribution

Paper (Arora et al.) also assumes the partition function concentrates in the senses $Z_c \approx Zexp(\|c^2\|)$, where Z is a constant, which is proved in (Arora et al.,2016)

Gaussian Walk Model (Theorem 1)

- Theorem 1 assumes above generative model and let s denote the random variable of n words. Then, the linear transformation A such that

$$v_w \approx AE\left[\frac{1}{n}\sum_{w_{i \in s}} v_{w_i} | w \in s\right]$$

Theorem 1: Proof

- According to Law of Expectation, we have

$E[c_s|w \in s] = E[E[c_s|s = w_1 \ldots w_n]|w \in s]$ - (2)

- By Bayes’ rule, the assumption on the distribution of c, and the partition function, we can derive

$$
\begin{aligned}
p(c|w) &\,\alpha\, p(w|c)p(c) \\
&\alpha\ \frac{1}{Z_c}\exp(\langle v_w, c\rangle) * \boxed{exp\left(\frac{-1}{2} c^T \Sigma^{-1} c\right)} \\
&\approx \frac{1}{Z}\exp\left(\langle v_w, c\rangle - c^T\left(\frac{1}{2}\Sigma^{-1} + I\right)c\right)
\end{aligned}
$$

Multi-Variable Gaussian Function

Theorem 1: Proof

- Thus, the mean of c|w is

  $E[c|w] \approx (\sum^{-1} + 2I)^{-1} v_w$ - (3)
- The derivative of above formula is as follows, but we will not calculate it in detail.
  - The expected value is the integral of its probability density function, so we have $\int_{-\infty}^{\infty} c * p(c|w)\, dc$

Theorem 1: Proof

- Next, compute $E[c_s|s = w_1 \ldots w_n]$ with the same assumptions.

$$p(c|w_1, \ldots, w_n)$$

$$\alpha\, p(w_1, \ldots, w_n|c) * p(c)$$

$$\alpha\, p(c) \prod_{i=1}^{n} p(w_i|c)$$

$$\approx \frac{1}{z^n} \exp\left(\sum_{i=1}^{n} v_{w_i}^T c - c^T \left(\frac{1}{2} \Sigma^{-1} + nI \right) c \right)$$

Theorem 1: Proof

- Thus, the mean of c|$w_1 \dots w_n$is

  $E[c|w] \approx (\sum^{-1} + 2I)^{-1} \sum_{i=1}^{n} v_{w_i}$ - (4)

- Plug in equation (3) and (4) into (2), we have

$$(\sum^{-1} + 2I)^{-1} v_w \approx E\left[\boxed{(\sum^{-1} + 2I)^{-1}} \sum_{i=1}^{n} v_{w_i}\right] \longleftarrow$$

Constant

- Because it is only a constant inside the square, according to theorem of expected value, the formula can be rearranged as follows.

$$(\sum^{-1} + 2I)^{-1} v_w \approx (\sum^{-1} + 2I)^{-1} E\left[\sum_{i=1}^{n} v_{w_i}\right]$$

- Thus, A is equal to $\frac{(\sum^{-1}+2I)^{-1}}{(\sum^{-1}+2I)^{-1}} * n$

Experimental Notes

- Arora et al. employs SIF embedding (Arora et al., 2016) to do the experiment because it has better result.
- SIF embedding employs tf-idf to calculate weighted average.
- Therefore, generative formula is as follows.

$p(w|c) = \alpha\, p(w) + (1-\alpha)\frac{\exp(v_{w*}c)}{Z_c}$, where α is a hyperparameter. – (5)

Proof of Linear Assertion: Introduction

- Suppose we have a word w which has two different senses s_1 and s_2, where v_w is the word embedding for w.
- Hand-replace each senses with s_1 and s_2
- Train the separate word vector for s_1 and s_2 but keeping the others fixed.

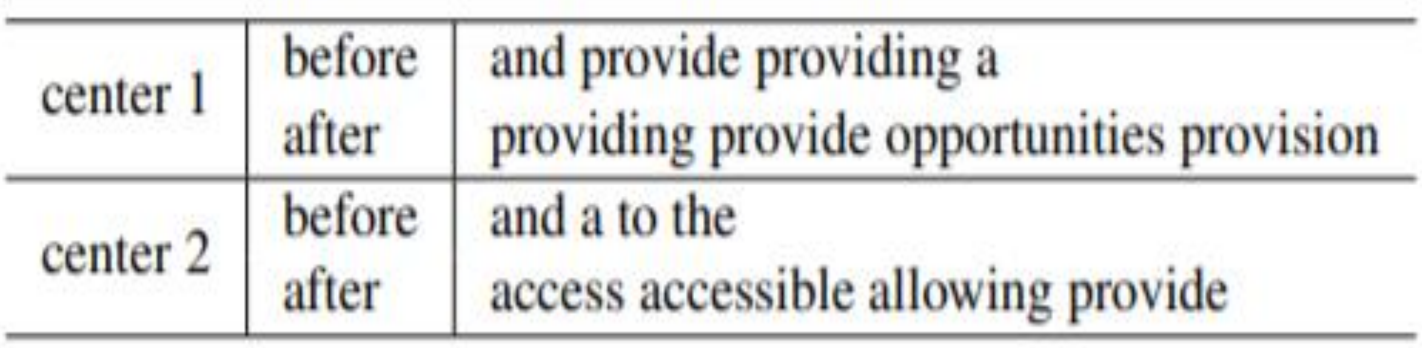
Proof of Linear Assertion: Theorem 2 (Main)

- Assume that our embedding will satisfy $||v_w - \overline{v_w}||_2 \rightarrow 0$ when the corpus length tends to be infinity, where $\overline{v_w} \approx \ \alpha v_{s1} + \beta v_{s2}$ - (6)

  $\alpha = \frac{f_1}{f_1+f_2}, \beta = \frac{f_2}{f_1+f_2}$, f_1 and f_2 are the frequency of $s_1\ and\ s_2$.
- Proof
  - The paper (Arora et al.) found after applying linear transformation, the sense s_1 will have higher value of α vice versa. Thus, $\overline{v_w}$ is a linear combination of v_{s1} and v_{s2}.

Theorem 2 (Main): Note

- Problem of older cluster-based approaches
  - Cluster center is not the sense of each cluster
  - Reason: Closest words are sometimes not meaningful.
- However: it becomes meaningful after applying linear transformation.



center 1	before	and provide providing a
	after	providing provide opportunities provision
center 2	before	and a to the
	after	access accessible allowing provide

Toward WSI: Atoms of Discourse

- Now we will discuss how to induce senses for all words with only word embedding and linear assertion.
- Each senses is interrelated with corresponding words in order to capture the senses.
  - E.g.: “ article of clothing” will have words like shoes and jackets.
- Generative model is formula (5).
- According to equation (1), because the probability of being output by discourse is decided by inner product, one expects that the vector for discourse will have higher inner products with all words relating to it.
- Thus, the optimizations are as follows.

Toward WSI: Atoms of Discourse

- Given word vectors $\{v_w\}$ and two integers k, m with k < m, find a set of unit vectors $A_1, A_2 \ldots A_m$ such that

✓ $v_w = \sum_{j=1}^{m} \alpha_{w,j} A_j + \eta_w \ - (7)$, where $\alpha_{w,j}$ is nonzero at most, and η_w is error vector.

✓ k is the sparsity parameters, and m is the number of atom.

✓ $\eta_w = \sum_w \boxed{\left\| v_w - \sum_{j=1}^{m} \alpha_{w,j} A_j \right\|_2^2}$ ← Square of Euclidean Distance

✓ Both $\alpha_{w,j} and\ A_j$ are unknown, and our goal is to minimize η_w. This is nonconvex optimization which is called sparse coding.

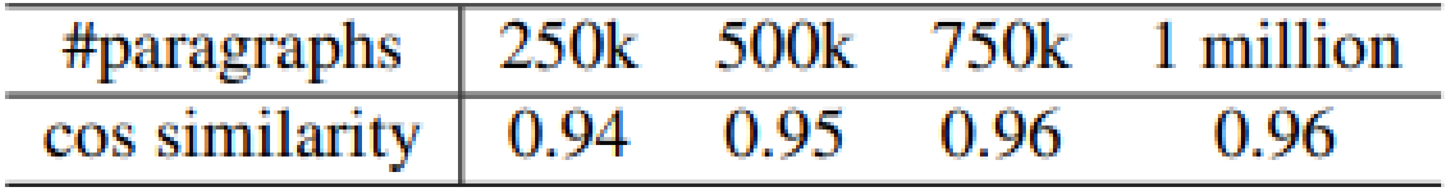
Test of Gaussian Work Model: Induced Embedding

- Suppose we have GloVe embedding
  - Let v_w denote the vector of word w.
  - Compute the SIF embedding for each words w'that is randomly sampled from wiki. The windows size is 20.
  - Let μ' be the average of all w'.
  - According to Gaussian Walk Model, there’s linear transformation mapping μ' to $v_{w'}$.
  - Thus, we need to solve the equation bellow.

✓ $argmin_A \sum_w ||A\mu_w - v_w||_2^2$ - (9), $A\mu_w$ is induced embedding.

Test of Gaussian Work Model: Induced Embedding

- The following table shows that
  - the cosine similarity between induced embedding and GloVe is large.
  - However: the similarity is small when we don’t apply linear transformation. It’s around 0.58.

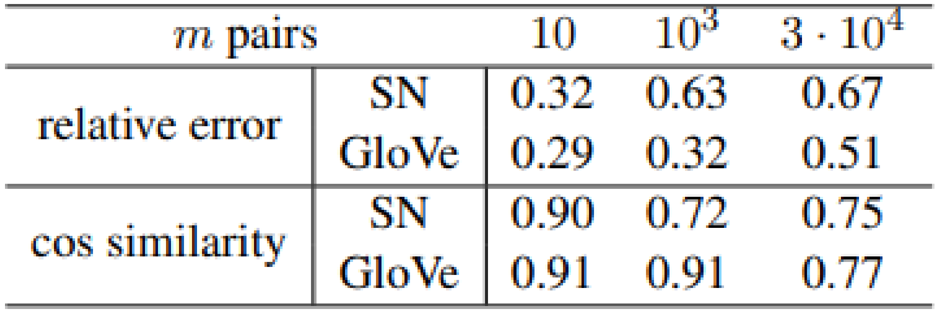


#paragraphs	250k	500k	750k	1 million
cos similarity	0.94	0.95	0.96	0.96

Test of Linear Assertion: Theorem 2 & Test 1

1. Train the word vector W_1 with wiki.
2. Randomly choose m pairs of non-repeated words.
3. Replace each with artificial polysemous words that have two different senses.
4. Train word vector W_2 on new corpus while fixed the words that were not be replaced with polysemous words. .
5. According to theorem 2, we can calculate the average relative error between vector for pusedoword μ_w and predicted vector v_w. The formula is as follows.

$\frac{1}{|S|}\sum_{w\epsilon S}\frac{\|\mu_w-v_w\|_2^2}{\|v_w\|_2^2}$, where S is a set of all pusedowords.

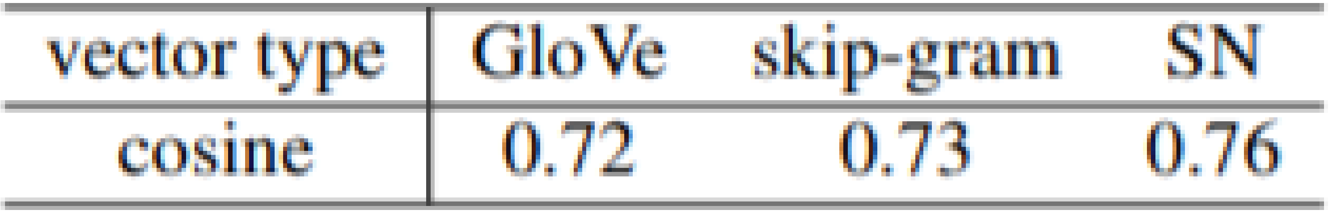


m pairs		10	10^3	$3 \cdot 10^4$
relative error	SN	0.32	0.63	0.67
	GloVe	0.29	0.32	0.51
cos similarity	SN	0.90	0.72	0.75
	GloVe	0.91	0.91	0.77

Test of Linear Assertion: Theorem 2 & Test 2

- This test
  - Is called proxy test but it’s a laboratory test.
  - Employs WordNet (Fellbaum, 1998) which includes various senses.
- Arora et al. trains SIF embedding followed by linear transformation.
- The linear assertion predicts that it lies close to span of senses vector.

any linear combination of vector

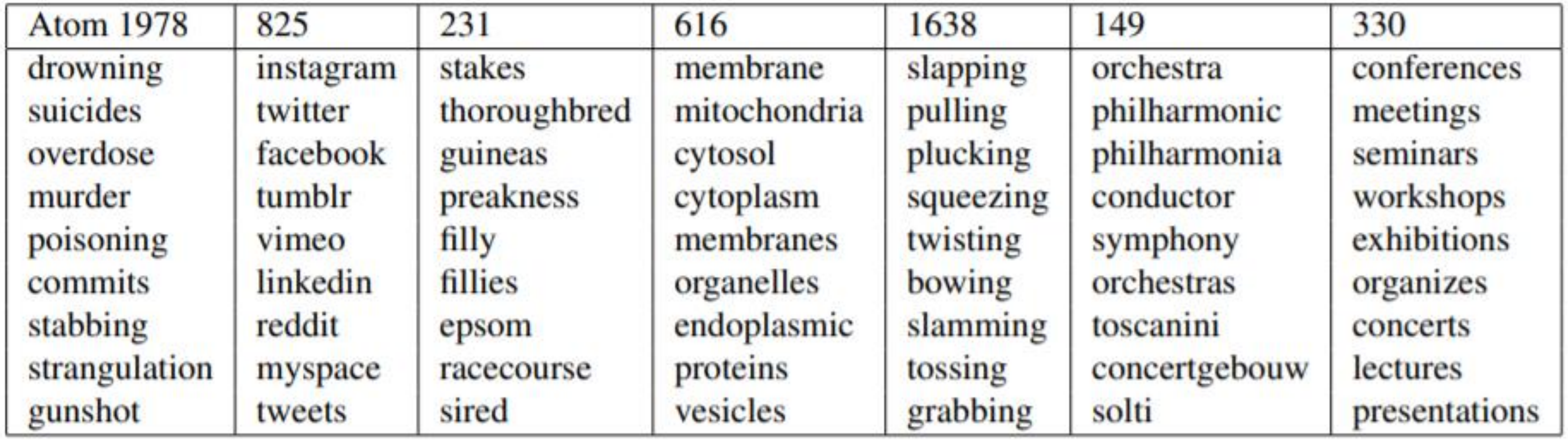


vector type	GloVe	skip-gram	SN
cosine	0.72	0.73	0.76

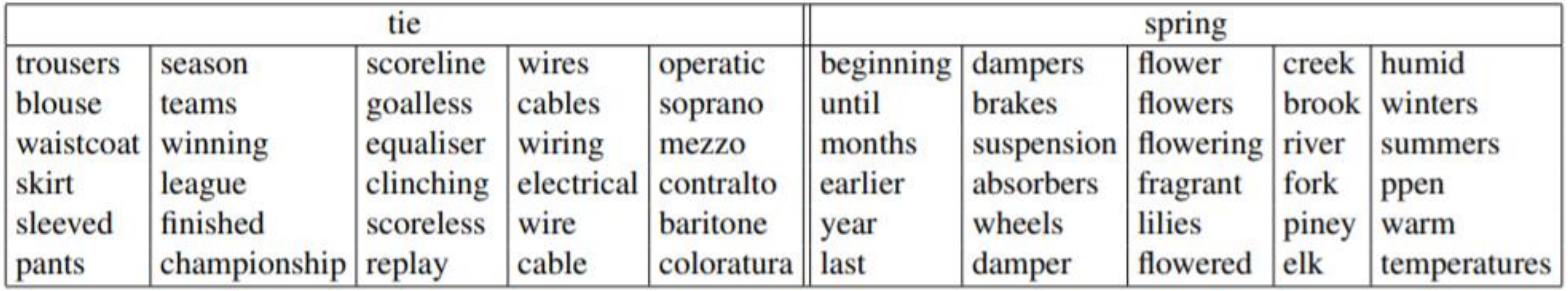
Experiments with Atoms of Discourse

- Experimental Embedding
  - 300-dimensional embeddings with SN objective (Arora et al.,2016)
  - Sparse coding solved by k-SVD is employed on this test.
- Experimental Parameters
  - The best sparsity parameter k is 5. (number of senses per word)
  - The number of atoms is 2000. (words around the sense)
  - Arora et al. also lets k be flexible for different words.
  - However: it doesn’t have better result than fixed k
  - Reason: it’s prone to be influenced by noise words. It will be hard to distinguish the difference between meaningful words and noise words if $\alpha_{w,j}$ is too small.
- $\alpha_{w,j}$ is hardly negative. According to the observation from the paper (Arora et al.), they think it’s because the appearance of words is best explained by what discourse is being used to generate it.
- It often makes mistakes in the last (or two) atoms in the last table.

Experiments with Atoms of Discourse



Atom 1978	825	231	616	1638	149	330
drowning	instagram	stakes	membrane	slapping	orchestra	conferences
suicides	twitter	thoroughbred	mitochondria	pulling	philharmonic	meetings
overdose	facebook	guineas	cytosol	plucking	philharmonia	seminars
murder	tumblr	preakness	cytoplasm	squeezing	conductor	workshops
poisoning	vimeo	filly	membranes	twisting	symphony	exhibitions
commits	linkedin	fillies	organelles	bowing	orchestras	organizes
stabbing	reddit	epsom	endoplasmic	slamming	toscanini	concerts
strangulation	myspace	racecourse	proteins	tossing	concertgebouw	lectures
gunshot	tweets	sired	vesicles	grabbing	solti	presentations



tie					spring				
trousers	season	scoreline	wires	operatic	beginning	dampers	flower	creek	humid
blouse	teams	goalless	cables	soprano	until	brakes	flowers	brook	winters
waistcoat	winning	equaliser	wiring	mezzo	months	suspension	flowering	river	summers
skirt	league	clinching	electrical	contralto	earlier	absorbers	fragrant	fork	ppen
sleeved	finished	scoreless	wire	baritone	year	wheels	lilies	piney	warm
pants	championship	replay	cable	coloratura	last	damper	flowered	elk	temperatures

Word Sense Induction

- This task is to cluster text so that the text that has the same sense will group together.
- The criteria for evaluation is F-Score and V-Measure. These two criteria tend to be opposite.
- The task computes μ_c for text c and applies k-means on this vector.
- Computation of μ_c

✓ For each atom a, word w, text c, there’s a joint distribution that describes atom a is the sense c in text c. Thus, according to Bayes rule

✓ p(a|c,w) α p(a|w) p(a|c)/p(a)

✓ p(a|w) is the coeffient in (6)

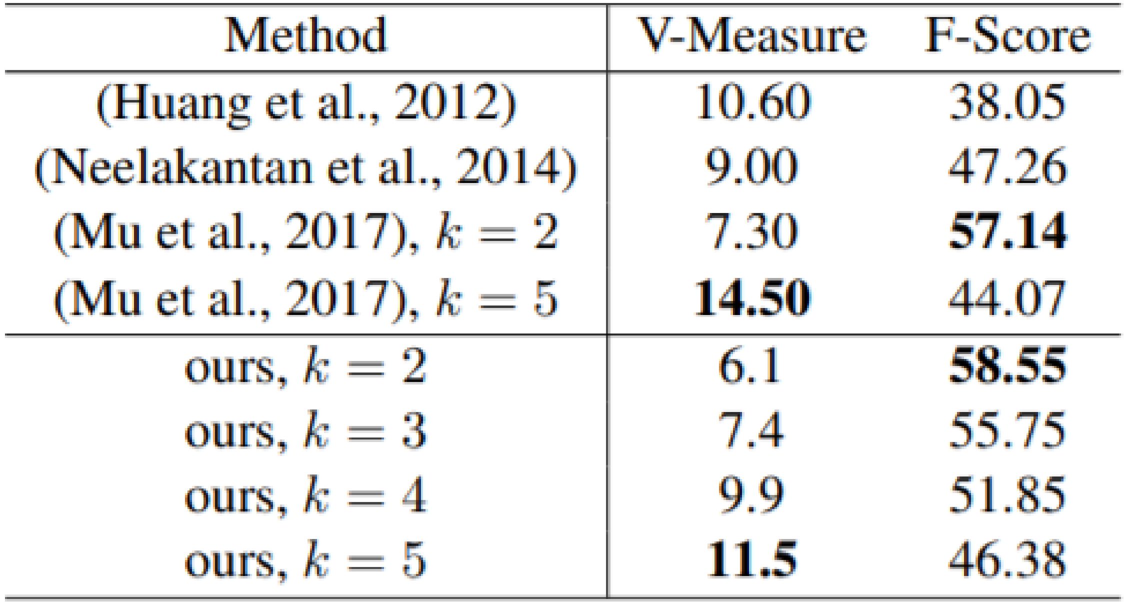
✓ p(a|c) is approximate to $\exp(< v_a, v_c >)$, where v_c is SIF embedding of c.

Word Sense Induction

- ✓ p(a) is equal to $E_c[p(a|c)]$
- ✓ similarity between c_1 and c_2 is defined as follow
- ✓ similarity(c_1, c_2)

  = $\left\langle \sum_{a_1} p(a_1|c_1, w)\, v_{a1}, \sum_{a_2} p(a_2|c_2, w)\, v_{a2} \right\rangle$ - (10)
- ✓ Thus,
- ✓ $\mu_c = \sum_a p(a|c, w) v_a$

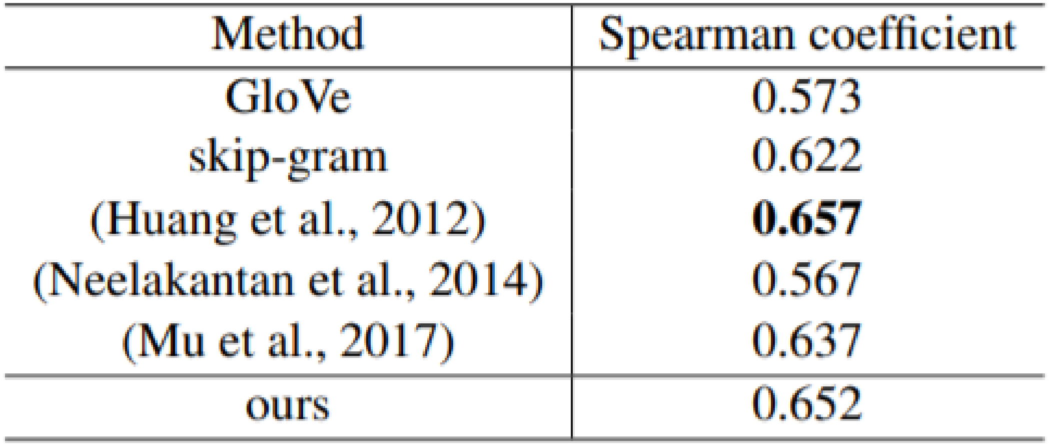
Word Sense Induction - Results



Method	V-Measure	F-Score
(Huang et al., 2012)	10.60	38.05
(Neelakantan et al., 2014)	9.00	47.26
(Mu et al., 2017), $k = 2$	7.30	**57.14**
(Mu et al., 2017), $k = 5$	**14.50**	44.07
ours, $k = 2$	6.1	**58.55**
ours, $k = 3$	7.4	55.75
ours, $k = 4$	9.9	51.85
ours, $k = 5$	**11.5**	46.38

Word Similarity in Context

- The dataset contains
  - 2000 pairs of words.
  - Predicted and the ground-truth similarity score.
- The criterion is the correlation between ground-truth and predicted one. The similarity is calculated by (10).
- The paper (Arora et al.) compares their results with recent word vectors.



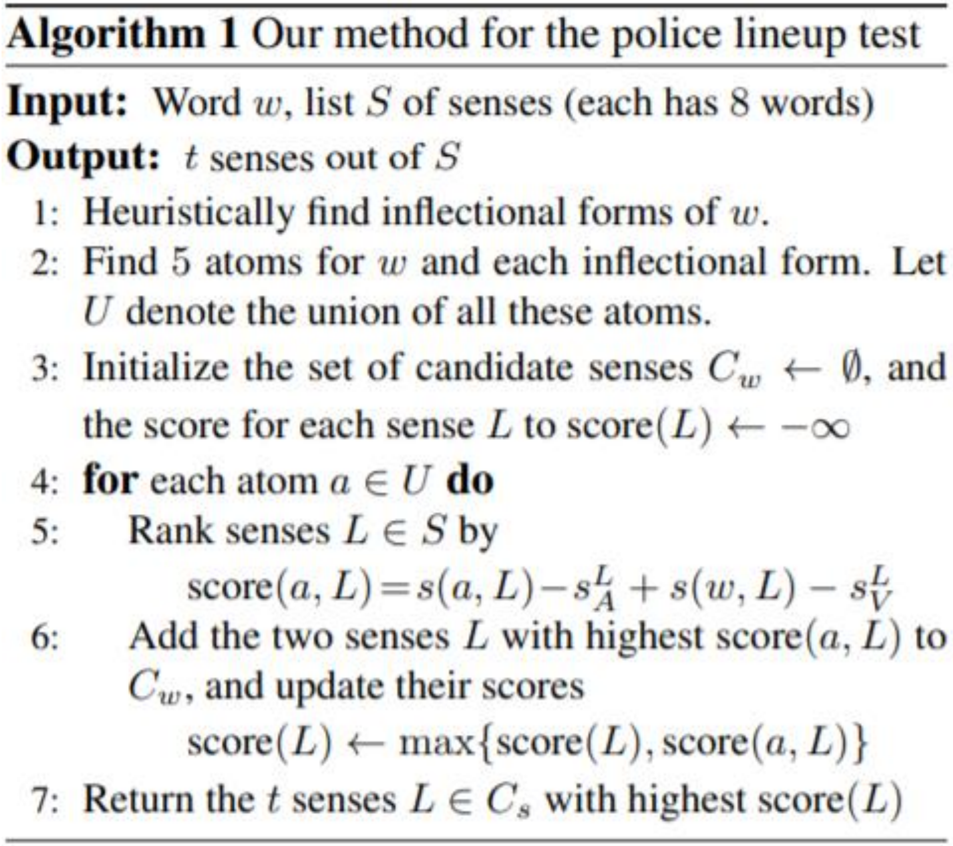
Method	Spearman coefficient
GloVe	0.573
skip-gram	0.622
(Huang et al., 2012)	**0.657**
(Neelakantan et al., 2014)	0.567
(Mu et al., 2017)	0.637
ours	0.652

?

Police Lineup

- The test uses 200 polysemous words and their 704 senses according to WordNet.
- Each senses, considered as ground-truth, is represented by 8 related words collected by college student from WordNet and online dictionary.
- Those words are not synonym.
- The quantitative test is as follows.
  1. pick one of these 200 polysemous words.
  2. pick true sense and randomly pick n senses from other words so that it contains n + 1 senses, where each contains 8 related words.
  3. Both algorithm and human need to identify the true sense.

Police Lineup: Algorithm



Algorithm 1 Our method for the police lineup test

```
Input: Word $w$, list $S$ of senses (each has 8 words)
Output: $t$ senses out of $S$
Heuristically find inflectional forms of $w$.
Find 5 atoms for $w$ and each inflectional form. Let $U$ denote the union of all these atoms.
Initialize the set of candidate senses $C_w \leftarrow \emptyset$, and the score for each sense $L$ to $\text{score}(L) \leftarrow -\infty$
for each atom $a \in U$ do
    Rank senses $L \in S$ by
        $\text{score}(a, L) = s(a, L) - s_A^L + s(w, L) - s_V^L$
    Add the two senses $L$ with highest $\text{score}(a, L)$ to $C_w$, and update their scores
        $\text{score}(L) \leftarrow \max\{\text{score}(L), \text{score}(a, L)\}$
Return the $t$ senses $L \in C_s$ with highest $\text{score}(L)$
```

Police Lineup: Calculation

✓ $s(x, Y) = \langle v_x, v_Y \rangle$, v_Y is SIF embedding. Y is a set of words.

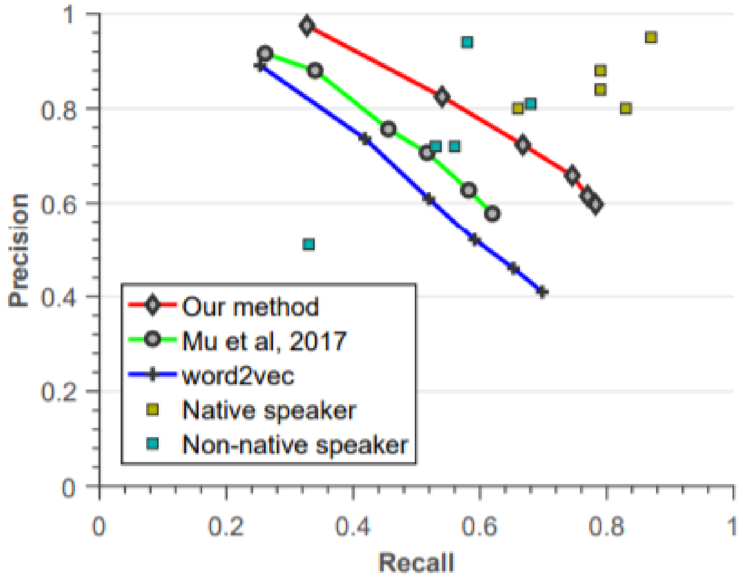
✓$s_A^Y = \frac{\sum_{a \varepsilon A} s(a,Y)}{|A|}$, where A are all atoms.

✓ Corpus contains stop words. The penalty terms s_A^L and s_V^L lower the score of sense L containing such words.

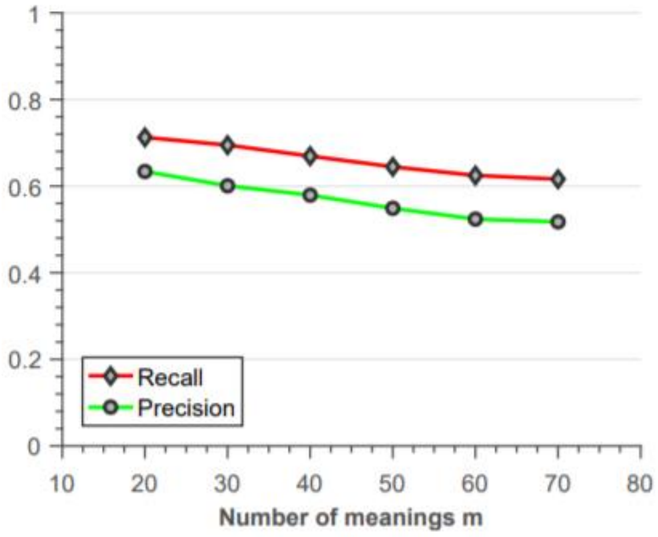
Police Lineup: Results

- The figure A suggests that
  - Arora et al’s method outperforms any competitors.
  - It is parallel to non-negative English speakers who are graduate student and major in science or engineering. Non-negative speakers have speak for 7 to 10 years.
- The figure B suggests that
  - It’s still reasonable when n = 50.

Police Lineup: Results



A



B

Assignment 4

1. Write a Cosine Similarity Algorithm with only Numpy - 50 points
2. Using the word vector that you calculated in last assignment to verify the code – 10 points
   - you only need to show the similarity between two words (5 sets)
   - The results should be the same as Gensim
3. Use SG vector to list top 10 similar words with your own similarity algorithm (5 sets) – 15 Points
4. Use CBOW vector to list top 10 similar words with your own similarity algorithm (5 sets) – 15 Points
5. Discuss the results – 10 Points
   - Why are some similarities between words positive?
   - Why are some similarities between words Negative?

Submission Requirements

- Email Subjects
    1. 姓名_學號_Assignment4
- Please use Jupyter Notebook and convert your result into HTML file.

Reference

- Manning et al., CS224n: Natural Language Processing with Deep Learning, Stanford University
- Pennington et al., GloVe: Global Vectors for Word Representation
- Mundra et al., Stanford University : Lecture 2 Notes
- Arora et al., Linear Algebraic Structure of Word Senses, with Applications to Polysemy