If Content is King,
Context is God!

Word Embeddings
Mar 26th, 2019

Applied Deep Learning

YUN-NUNG (VIVIAN) CHEN HTTP://ADL.MIULAB.TW

國立臺灣大學
National Taiwan University

Slides credited from Dr. Richard Socher

Review

Meaning Representations in Computers

Knowledge-based representation

Corpus-based representation

- ✓ Atomic symbol
- ✓ Neighbors
  - High-dimensional sparse word vector
  - Low-dimensional dense word vector
    - Method 1 – dimension reduction
    - Method 2 – direct learning

Meaning Representations in Computers

Knowledge-based representation

Corpus-based representation

- ✓ Atomic symbol
- ✓ Neighbors
  - High-dimensional sparse word vector
  - Low-dimensional dense word vector
    - Method 1 – dimension reduction
    - Method 2 – direct learning

Corpus-based representation

Atomic symbols: ***one-hot*** representation

$$\text{car} \quad [0\ 0\ 0\ 0\ 0\ 0\ \underset{\uparrow\ \text{car}}{1}\ 0\ 0\ \ldots\ 0]$$

Issues: difficult to compute the similarity (i.e. comparing "car" and "motorcycle")

$$\underbrace{[0\ 0\ 0\ 0\ 0\ 0\ 1\ 0\ 0\ \ldots\ 0]}_{\text{car}}\ \text{AND}\ \underbrace{[0\ 0\ 1\ 0\ 0\ 0\ 0\ 0\ 0\ \ldots\ 0]}_{\text{motorcycle}} = 0$$

Idea: words with similar meanings often have similar neighbors

Meaning Representations in Computers

Knowledge-based representation

Corpus-based representation

- ✓ Atomic symbol
- ✓ Neighbors
  - High-dimensional sparse word vector
  - Low-dimensional dense word vector
    - Method 1 – dimension reduction
    - Method 2 – direct learning

Window-based Co-occurrence Matrix

Example

- Window length=1
- Left or right context
- Corpus:

I love NTU.
I love deep learning.
I enjoy learning.

similarity > 0

Counts	I	love	enjoy	NTU	deep	learning
I	0	2	1	0	0	0
love	2	0	0	1	1	0
enjoy	1	0	0	0	0	1
NTU	0	1	0	0	0	0
deep	0	1	0	0	0	1
learning	0	0	1	0	1	0

Issues:

- matrix size increases with vocabulary
- high dimensional
- sparsity → poor robustness

Idea: low dimensional word vector

Meaning Representations in Computers

Knowledge-based representation

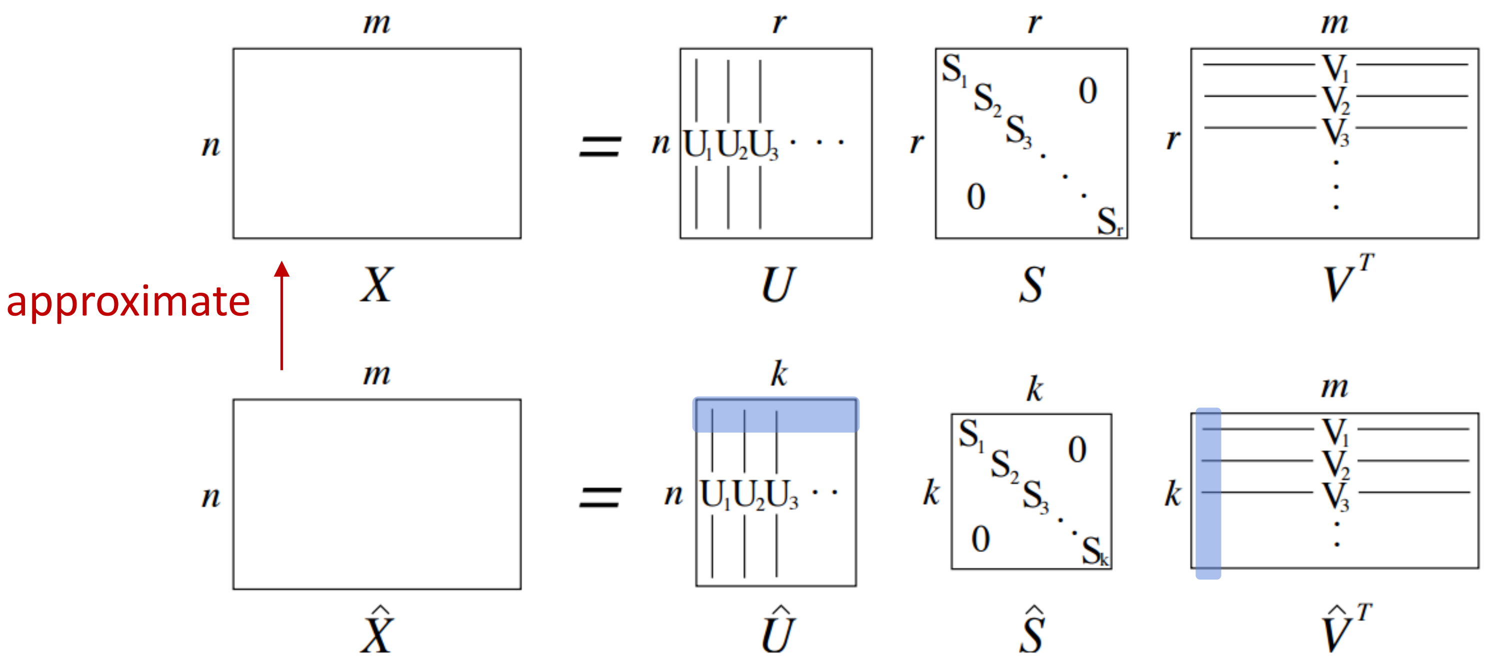
Corpus-based representation

- ✓ Atomic symbol
- ✓ Neighbors
  - High-dimensional sparse word vector
  - Low-dimensional dense word vector
    - Method 1 – dimension reduction
    - Method 2 – direct learning

Low-Dimensional Dense Word Vector

Method 1: dimension reduction on the matrix

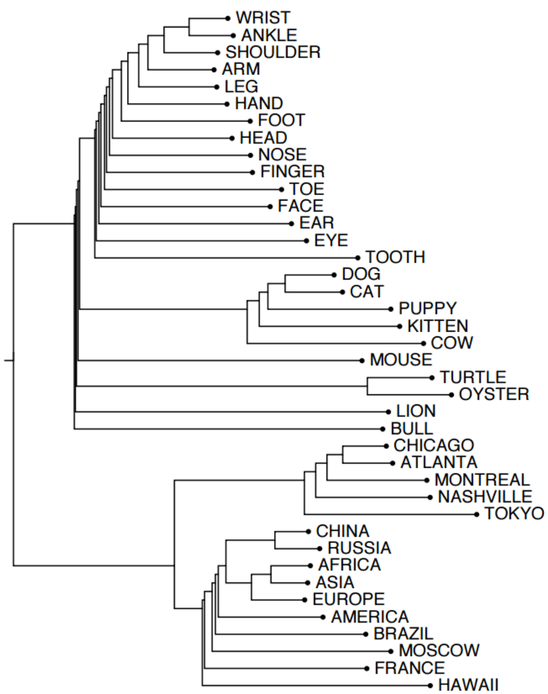
Singular Value Decomposition (SVD) of co-occurrence matrix X



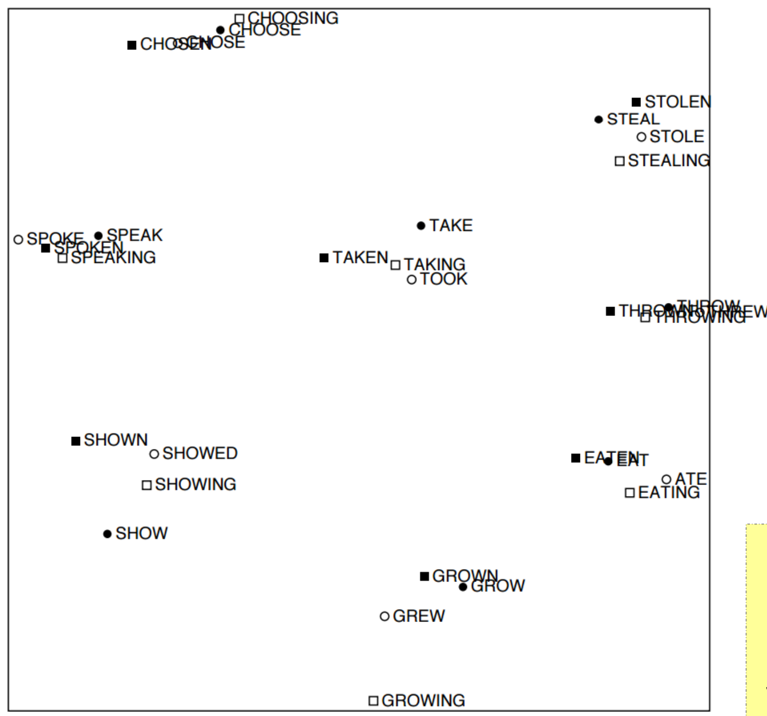
Low-Dimensional Dense Word Vector

Method 1: dimension reduction on the matrix

Singular Value Decomposition (SVD) of co-occurrence matrix X



semantic relations



syntactic relations

Issues:
- computationally expensive: $O(mn^2)$ when n<m for n x m matrix
- difficult to add new words

Idea: directly learn low-dimensional word vectors

Word Representation

Knowledge-based representation

Corpus-based representation

- ✓ Atomic symbol
- ✓ Neighbors
  - High-dimensional sparse word vector
  - Low-dimensional dense word vector
    - Method 1 – dimension reduction
    - Method 2 – direct learning → word embedding

Word Embedding

Method 2: directly learn low-dimensional word vectors

- Learning representations by back-propagation. (Rumelhart et al., 1986)
- A neural probabilistic language model (Bengio et al., 2003)
- NLP (almost) from Scratch (Collobert & Weston, 2008)
- Recent and most popular models: word2vec (Mikolov et al. 2013) and Glove (Pennington et al., 2014)

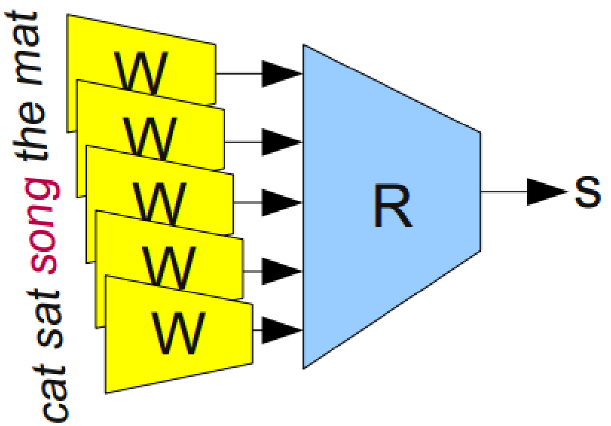
Word Embedding Benefit

Given an unlabeled training corpus, produce a vector for each word that encodes its semantic information. These vectors are useful because:

① semantic similarity between two words can be calculated as the cosine similarity between their corresponding word vectors

② word vectors as powerful features for various supervised NLP tasks since the vectors contain semantic information

③ propagate any information into them via neural networks and update during training



Word2Vec Skip-Gram

Mikolov et al., "Distributed representations of words and phrases and their compositionality," in *NIPS*, 2013.

Mikolov et al., "Efficient estimation of word representations in vector space," in *ICLR Workshop*, 2013.

Word2Vec – Skip-Gram Model

Goal: predict surrounding words within a window of each word

Objective function: maximize the probability of any context word given the current center word

$$w_1, w_2, \cdots, \underbrace{w_{t-m}, \cdots, w_{t-1}, w_t, w_{t+1}, \cdots, w_{t+m}}_{\text{context window}}, \cdots, w_{T-1}, w_T$$

w_I w_O

$$p(w_{O,1}, w_{O,2}, \cdots, w_{O,C} \mid w_I) = \prod_{c=1}^{C} p(w_{O,c} \mid w_I)$$

target word vector

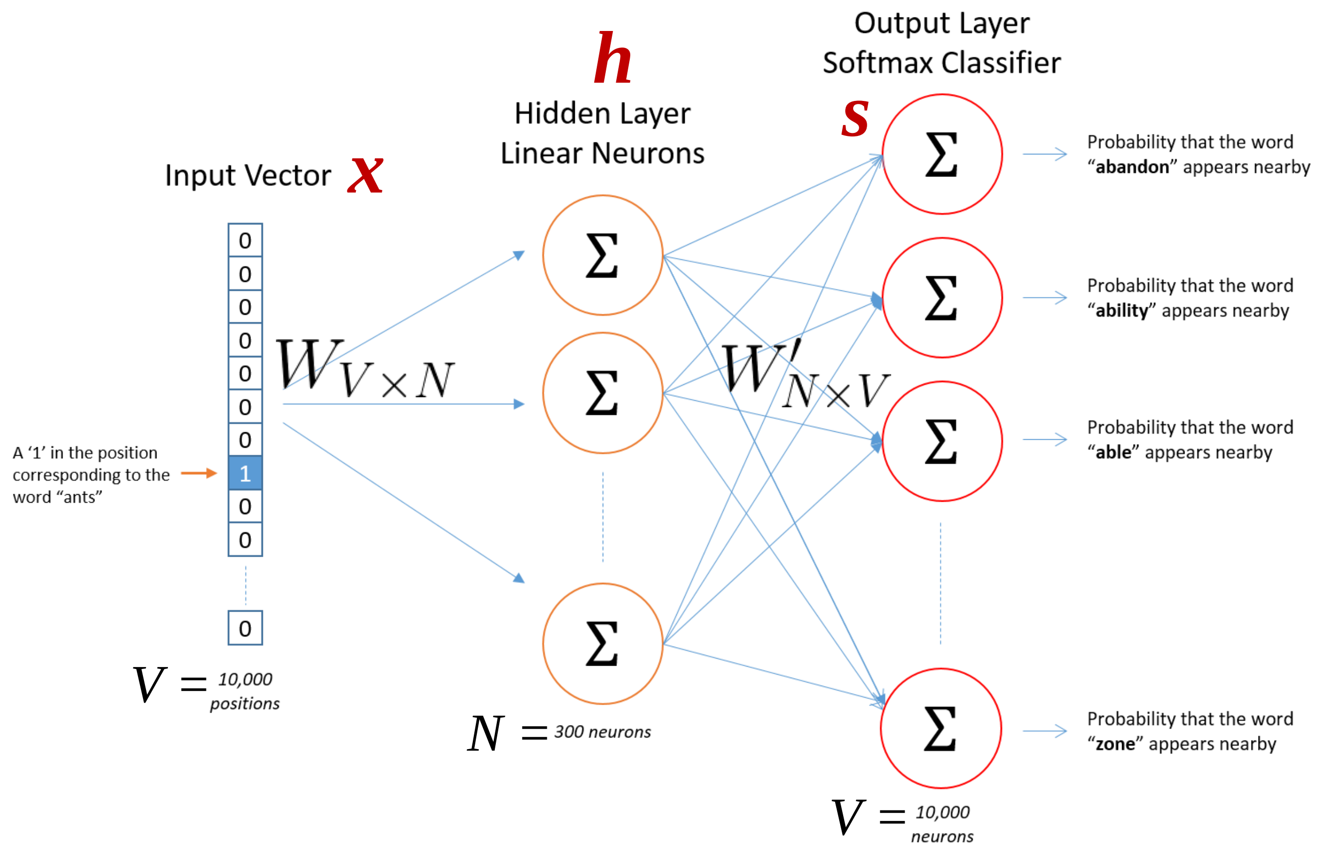
$$C(\theta) = -\sum_{w_I} \sum_{c=1}^{C} \log p(w_{O,c} \mid w_I) \quad p(w_O \mid w_I) = \frac{\exp(v'^{T}_{w_O} v_{w_I})}{\sum_j \exp(v'^{T}_{w_j} v_{w_I})}$$

outside target word

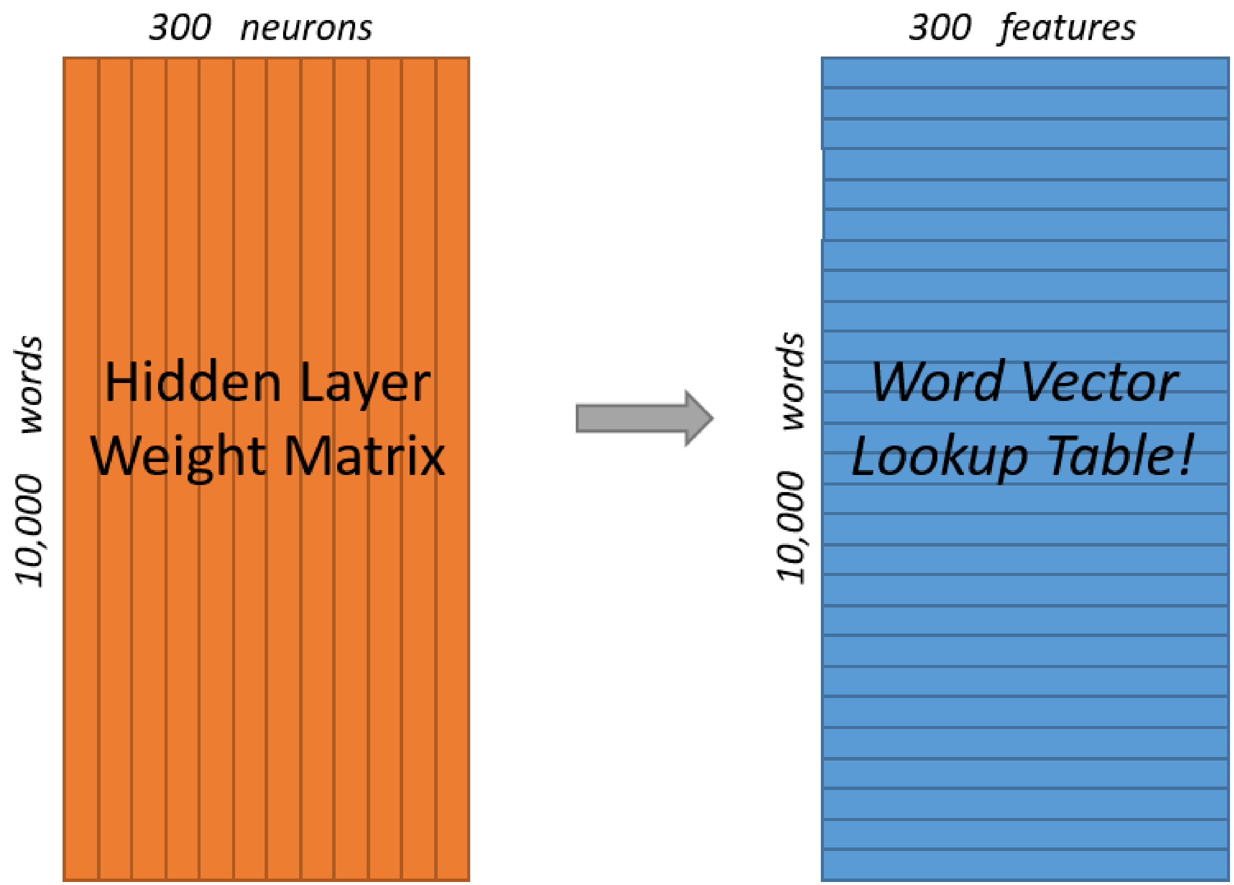
Benefit: faster, easily incorporate a new sentence/document or add a word to vocab

Word2Vec Skip-Gram Illustration

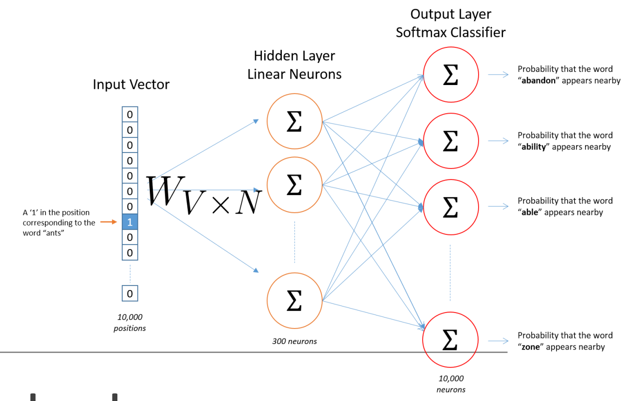
Goal: predict surrounding words within a window of each word



Hidden Layer Weight Matrix → Word Embedding Matrix $W_{V \times N}$



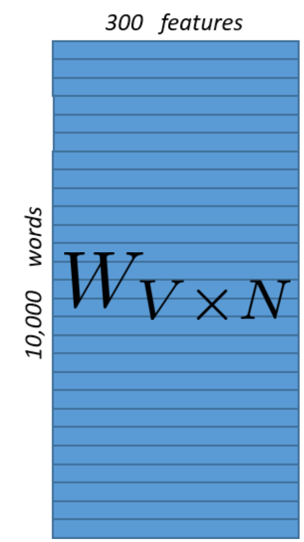
Weight Matrix Relation



Hidden layer weight matrix = word vector lookup

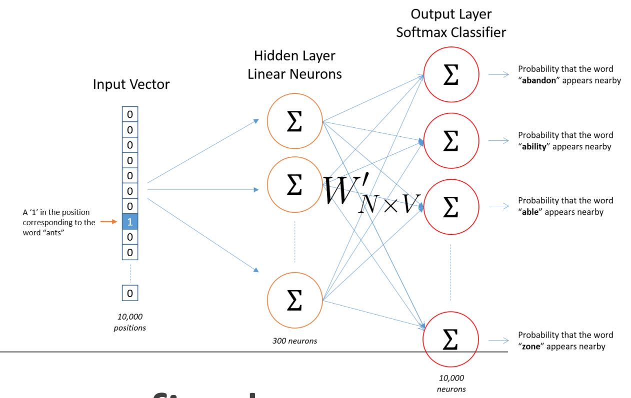
$$h = x^T W = W_{(k,.)} := v_{w_I}$$

$$\begin{bmatrix} 0 & 0 & 0 & 1 & 0 \end{bmatrix} \times \begin{bmatrix} 17 & 24 & 1 \\ 23 & 5 & 7 \\ 4 & 6 & 13 \\ 10 & 12 & 19 \\ 11 & 18 & 25 \end{bmatrix} = \begin{bmatrix} 10 & 12 & 19 \end{bmatrix}$$

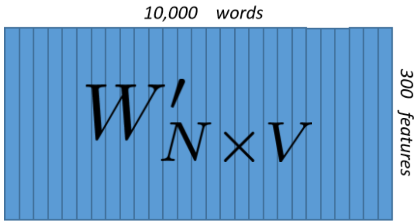


Each vocabulary entry has two vectors: as a **target** word and as a **context** word

Weight Matrix Relation



Output layer weight matrix = weighted sum as final score

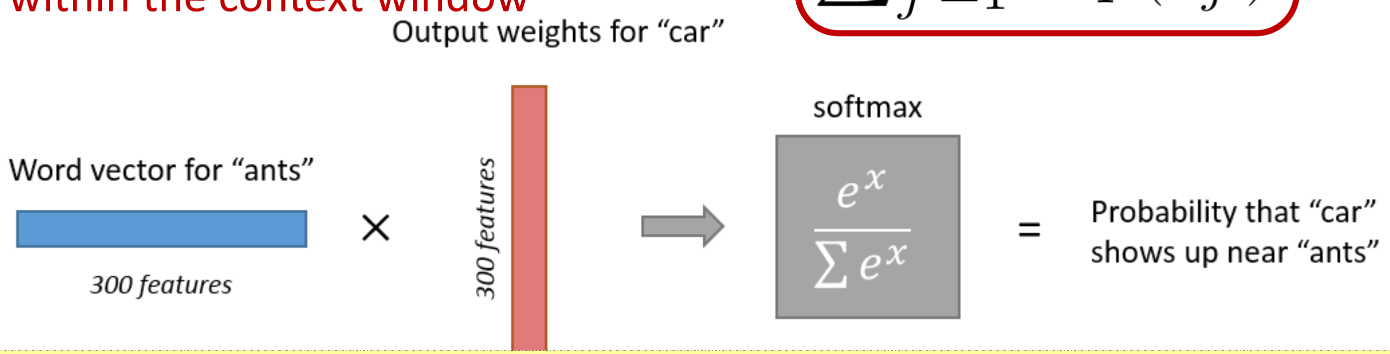


$$s_j = h v'_{w_j}$$

$$p(w_j = w_{O,c} \mid w_I) = y_{jc} = \boxed{\frac{\exp(s_{jc})}{\sum_{j'=1}^{V} \exp(s_{j'})}}$$

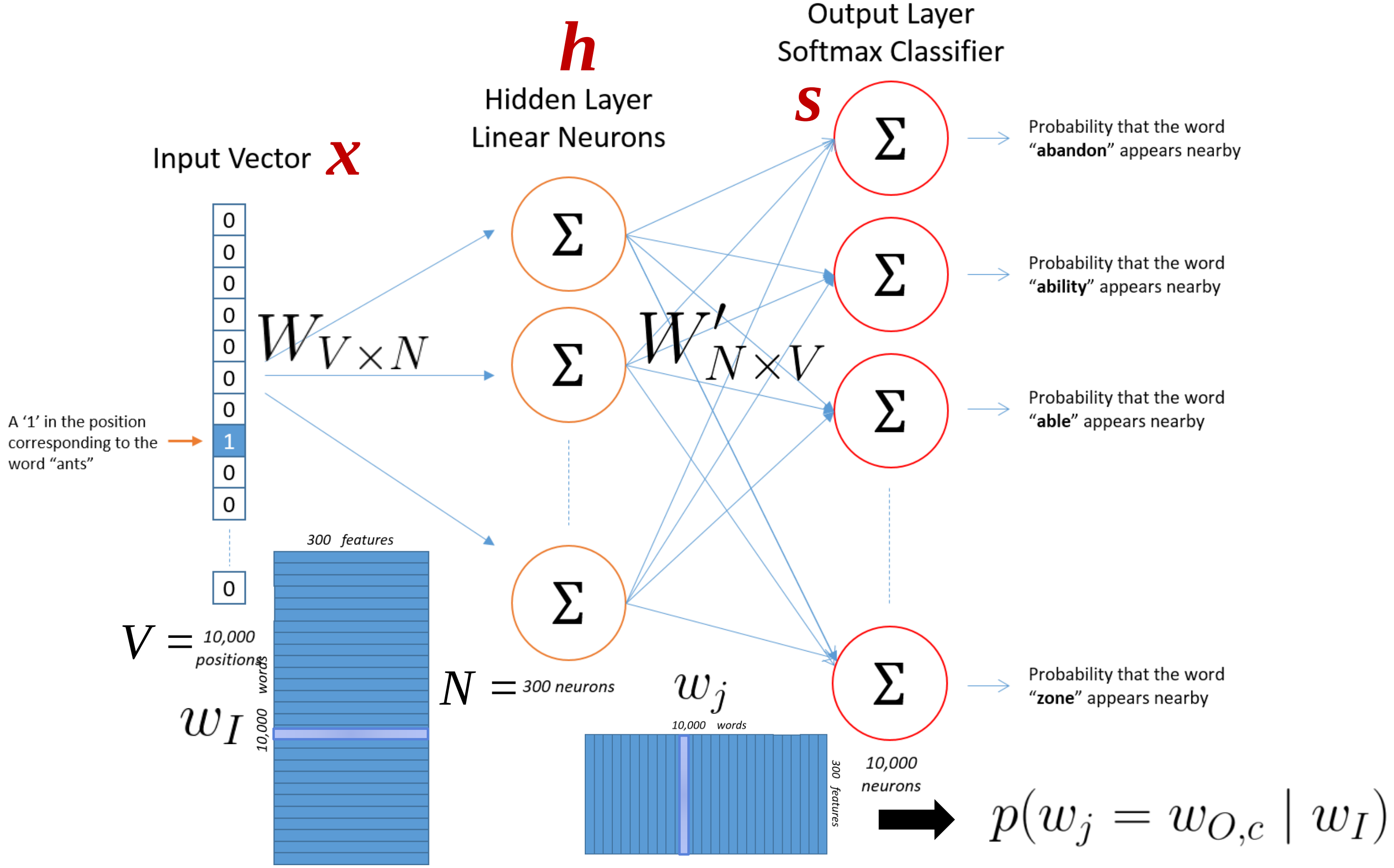
softmax

within the context window



Each vocabulary entry has two vectors: as a **target** word and as a **context** word

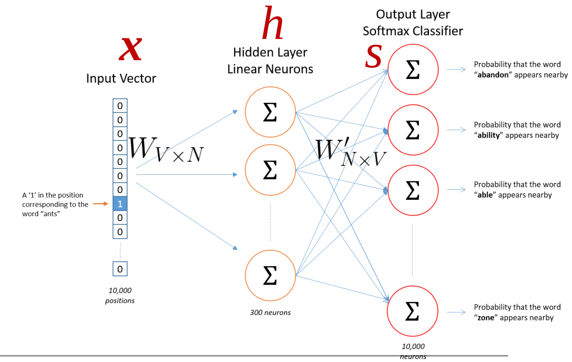
Word2Vec Skip-Gram Illustration



Loss Function

Given a target word (w_I)

$$
\begin{aligned}
C(\theta) &= -\log p(w_{O,1}, w_{O,2}, \cdots, w_{O,C} \mid w_I) \\
&= -\log \prod_{c=1}^{C} \frac{\exp(s_{j_c})}{\sum_{j'=1}^{V} \exp(s_{j'})} \\
&= -\sum_{c=1}^{C} s_{j_c} + C \log \sum_{j'=1}^{V} \exp(s_{j'})
\end{aligned}
$$



SGD Update for W'

Given a target word (w_I)

$$\frac{\partial C(\theta)}{\partial w'_{ij}} = \sum_{c=1}^{C} \frac{\partial C(\theta)}{\partial s_{j_c}} \frac{\partial s_{j_c}}{\partial w'_{ij}} = \sum_{c=1}^{C} (y_{j_c} - t_{j_c}) \cdot h_i$$

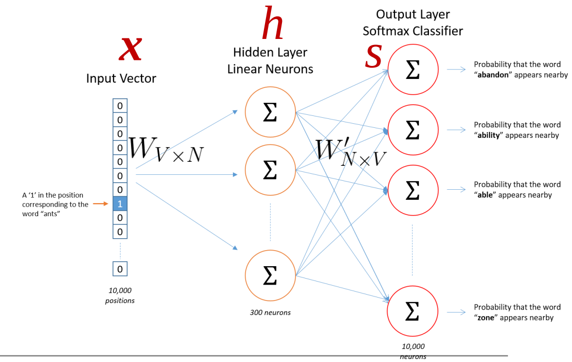
$$\boxed{s_j = {v'_{w_j}}^T \cdot h}$$

$$\frac{\partial C(\theta)}{\partial s_{j_c}} = y_{j_c} - t_{j_c} := e_{j_c}$$

error term

=1, when w_{jc} is within the context window
=0, otherwise

$$\boxed{{w'_{ij}}^{(t+1)} = {w'_{ij}}^{(t)} - \eta \cdot \sum_{c=1}^{C} (y_{j_c} - t_{j_c}) \cdot h_i}$$



SGD Update for W

$$\frac{\partial C(\theta)}{\partial w_{ki}} = \frac{\partial C(\theta)}{\partial h_i}\frac{\partial h_i}{\partial w_{ki}} = \sum_{j=1}^{V}\sum_{c=1}^{C}(y_{j_c} - t_{j_c}) \cdot w'_{ij} \cdot x_k$$

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

$$\frac{\partial C(\theta)}{\partial h_i} = \sum_{j=1}^{V}\frac{\partial C(\theta)}{\partial s_j}\frac{\partial s_j}{\partial h_i} = \sum_{j=1}^{V}\sum_{c=1}^{C}(y_{j_c} - t_{j_c}) \cdot w'_{ij}$$

$$\boxed{s_j = {v'_{w_j}}^T \cdot h}$$

$$w_{ij}^{(t+1)} = w_{ij}^{(t)} - \eta \cdot \sum_{j=1}^{V}\sum_{c=1}^{C}(y_{j_c} - t_{j_c}) \cdot w'_{ij} \cdot x_j$$

SGD Update

$$w'^{(t+1)}_{ij} = w'^{(t)}_{ij} - \eta \cdot \sum_{c=1}^{C} (y_{j_c} - t_{j_c}) \cdot h_i$$

$$EI_j = \sum_{c=1}^{C} (y_{j_c} - t_{j_c})$$

$$v'^{(t+1)}_{w_j} = v'^{(t)}_{w_j} - \eta \cdot EI_j \cdot h$$

$$w^{(t+1)}_{ij} = w^{(t)}_{ij} - \eta \cdot \sum_{j=1}^{V} \sum_{c=1}^{C} (y_{j_c} - t_{j_c}) \cdot w'_{ij} \cdot x_j$$

$$EH_i = \sum_{j=1}^{V} EI_j \cdot w'_{ij} \cdot x_j$$

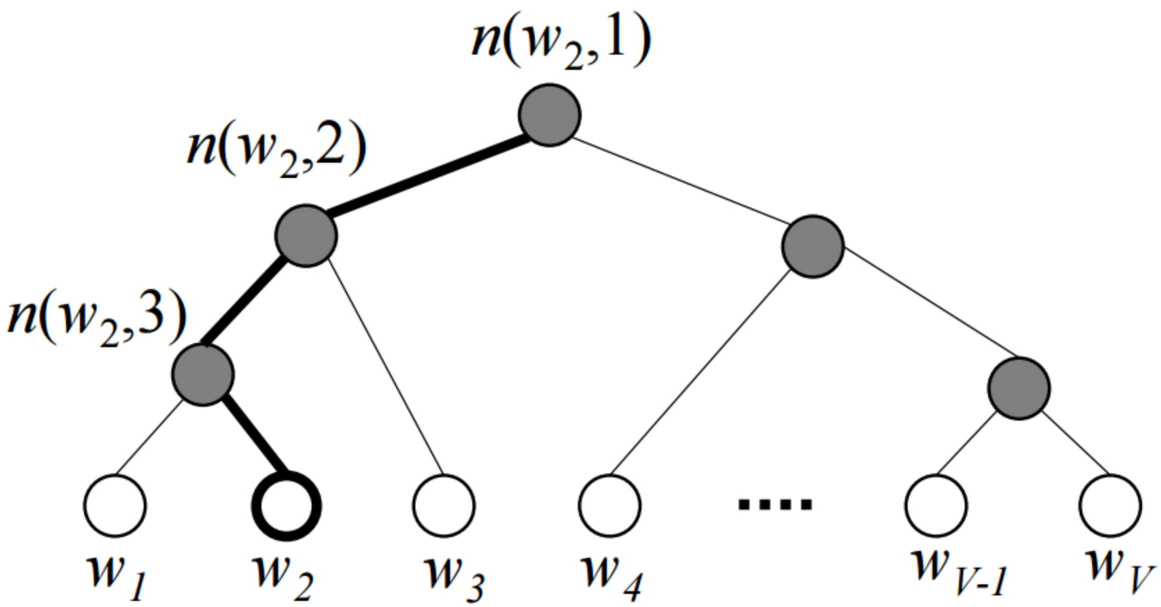
$$v^{(t+1)}_{w_I} = v^{(t)}_{w_I} - \eta \cdot EH^T$$

large vocabularies or large training corpora → expensive computations

limit the number of output vectors that must be updated per training instance
→ hierarchical softmax, sampling

Hierarchical Softmax

Idea: compute the probability of leaf nodes using the paths



$$O(N) \rightarrow O(\log N)$$

Negative Sampling

Idea: only update a sample of output vectors

$$C(\theta) = -\log \sigma({v'_{w_O}}^T v_{w_I}) + \sum_{w_j \in \mathcal{W}_{\text{neg}}} \log \sigma({v'_{w_j}}^T v_{w_I})$$

$${v'_{w_j}}^{(t+1)} = {v'_{w_j}}^{(t)} - \eta \cdot EI_j \cdot h$$

$$EI_j = \sigma({v'_{w_j}}^T v_{w_I}) - t_j$$

$$v_{w_I}^{(t+1)} = v_{w_I}^{(t)} - \eta \cdot EH^T$$

$$EH = \sum_{w_j \in \{w_O\} \cup \mathcal{W}_{\text{neg}}} EI_j \cdot v'_{w_j}$$

$$w_j \in \{w_O\} \cup \mathcal{W}_{\text{neg}}$$

Negative Sampling

Sampling methods $w_j \in \{w_O\} \cup \mathcal{W}_{\text{neg}}$

- Random sampling
- Distribution sampling: w_j is sampled from $P(w)$

What is a good $P(w)$?

Idea: less frequent words sampled more often

Empirical setting: unigram model raised to the power of 3/4

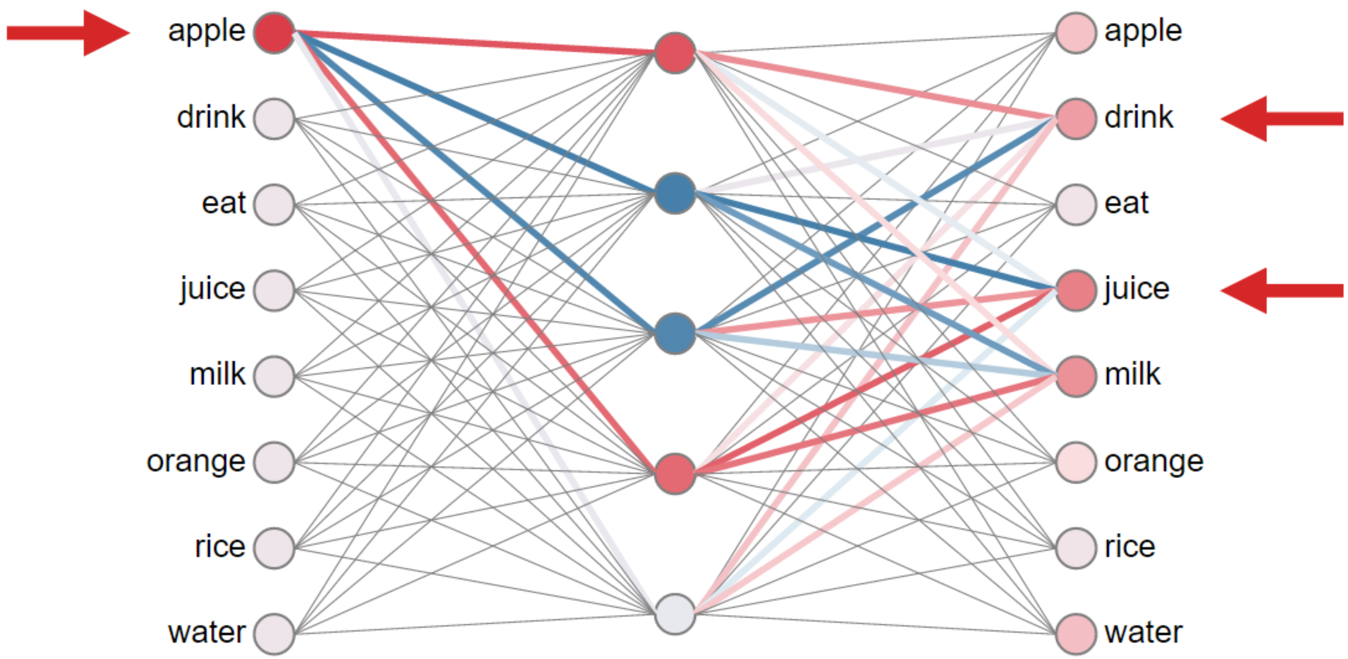
Word	Probability to be sampled for "neg"
is	$0.9^{3/4} = 0.92$
constitution	$0.09^{3/4} = 0.16$
bombastic	$0.01^{3/4} = 0.032$

Mikolov et al., "Distributed representations of words and phrases and their compositionality," in NIPS, 2013.

Word2Vec Skip-Gram Visualization

https://ronxin.github.io/wevi/

Skip-gram training data:
apple|drink^juice,orange|eat^apple,rice|drink^juice,juice|drink^milk,milk|drink^rice,water|drink^milk,juice|orange^apple,juice|apple^drink,milk|rice^drink,drink|milk^water,drink|water^juice,drink|juice^water



Word2Vec Variants

Skip-gram: predicting surrounding words given the target word (Mikolov+, 2013)

better

$$p(w_{t-m}, \cdots w_{t-1}, w_{t+1}, \cdots , w_{t+m} \mid w_t)$$

CBOW (continuous bag-of-words): predicting the target word given the surrounding words (Mikolov+, 2013)

$$p(w_t \mid w_{t-m}, \cdots w_{t-1}, w_{t+1}, \cdots , w_{t+m})$$

LM (Language modeling): predicting the next words given the proceeding contexts (Mikolov+, 2013)

first

$$p(w_{t+1} \mid w_t)$$

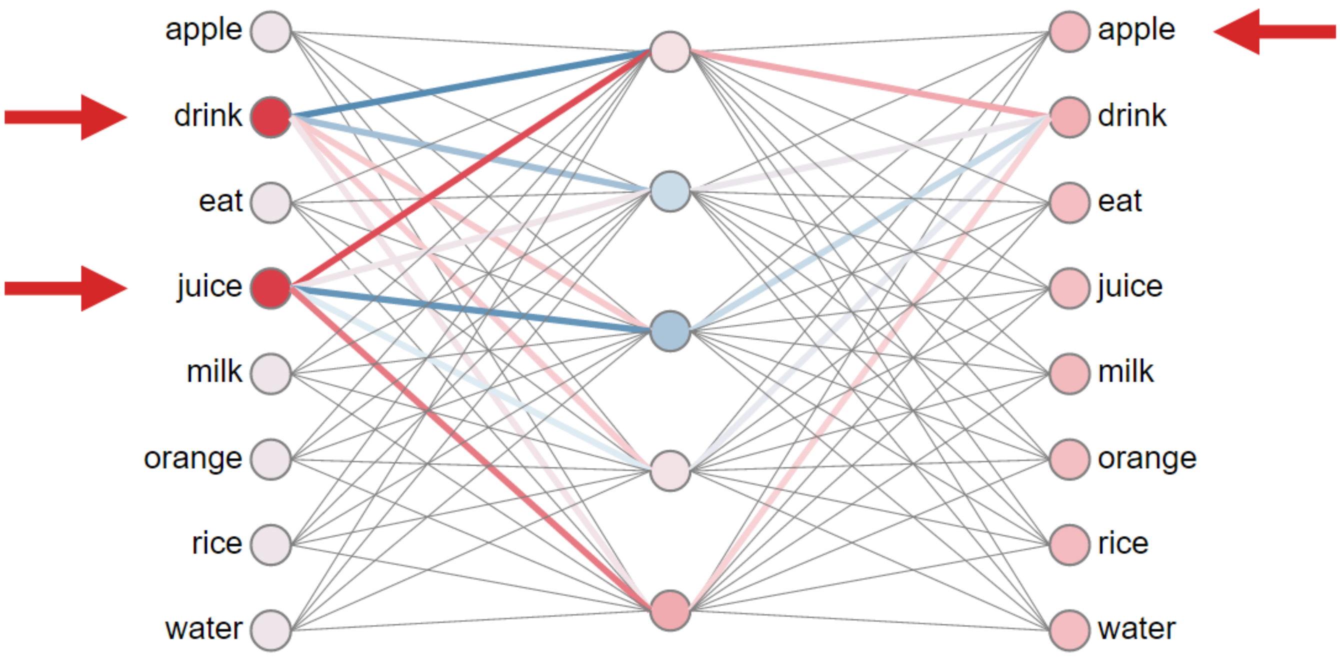
Practice the derivation by yourself!!

Mikolov et al., "Efficient estimation of word representations in vector space," in *ICLR Workshop*, 2013.
Mikolov et al., "Linguistic regularities in continuous space word representations," in *NAACL HLT*, 2013.

Word2Vec CBOW

Goal: predicting the target word given the surrounding words

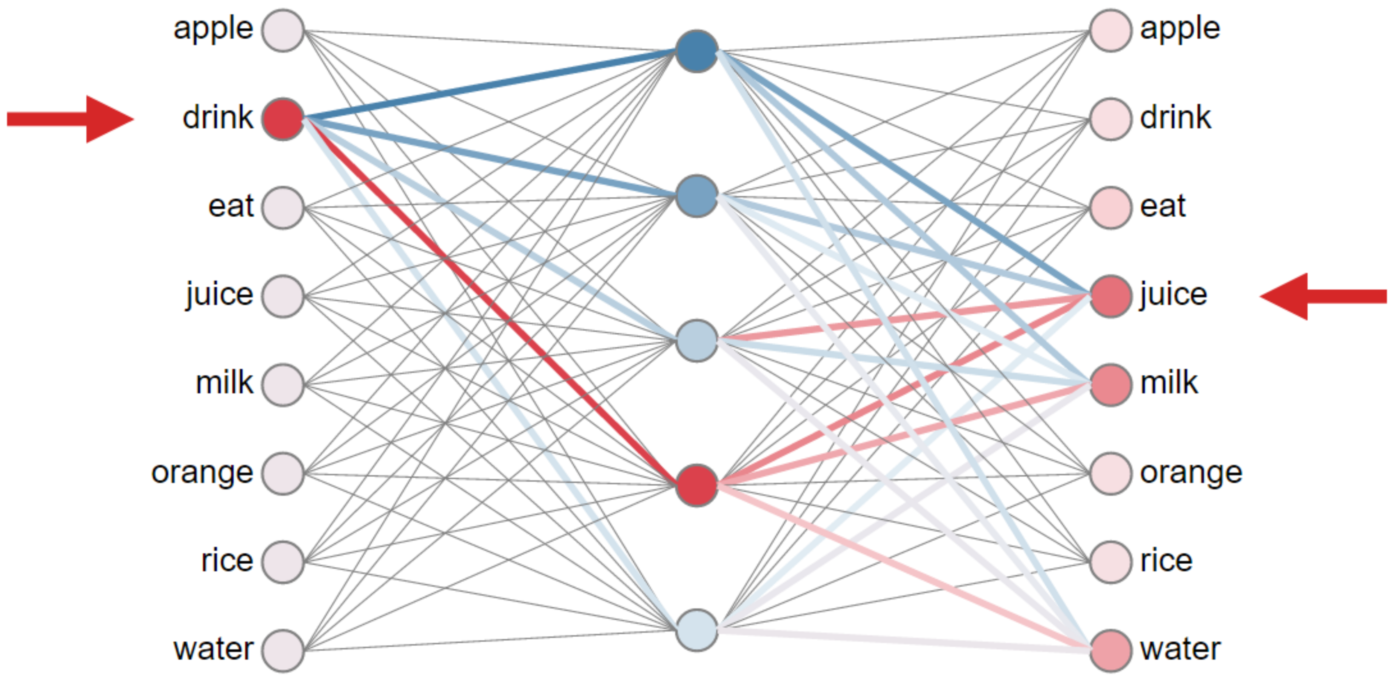
$$p(w_t \mid w_{t-m}, \cdots w_{t-1}, w_{t+1}, \cdots , w_{t+m})$$



Word2Vec LM

Goal: predicting the next words given the proceeding contexts

$$p(w_{t+1} \mid w_t)$$



Comparison

Count-based

- Example
  - LSA, HAL (Lund & Burgess), COALS (Rohde et al), Hellinger-PCA (Lebret & Collobert)
- Pros
  - ✓Fast training
  - ✓Efficient usage of statistics
- Cons
  - ✓Primarily used to capture word similarity
  - ✓Disproportionate importance given to large counts

Direct prediction

- Example
  - NNLM, HLBL, RNN, Skipgram/CBOW, (Bengio et al; Collobert & Weston; Huang et al; Mnih & Hinton; Mikolov et al; Mnih & Kavukcuoglu)
- Pros
  - ✓Generate improved performance on other tasks
  - ✓Capture complex patterns beyond word similarity
- Cons
  - ✓Benefits mainly from large corpus
  - ✓Inefficient usage of statistics

Combining the benefits from both worlds → GloVe

GloVe

Pennington et al., "GloVe: Global Vectors for Word Representation," in EMNLP, 2014.

GloVe

Idea: ratio of co-occurrence probability can encode meaning

P_{ij} is the probability that word w_j appears in the context of word w_i

$$P_{ij} = P(w_j \mid w_i) = X_{ij}/X_i$$

Relationship between the words w_i and w_j

	x = solid	x = gas	x = water	x = random
$P(x \mid \text{ice})$	large	small	large	small
$P(x \mid \text{stream})$	small	large	large	small
$\dfrac{P(x \mid \text{ice})}{P(x \mid \text{stream})}$	large	small	~ 1	~ 1

GloVe

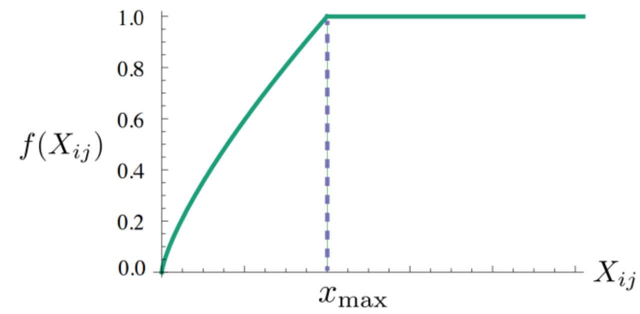
The relationship of w_i and w_j approximates the ratio of their co-occurrence probabilities with various w_k

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

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

$$F((v_{w_i} - v_{w_j})^T v'_{\tilde{w}_k}) = \frac{P_{ik}}{P_{jk}} \qquad F(\cdot) = \exp(\cdot)$$

$$v_{w_i} \cdot v'_{\tilde{w}_k} = v_{w_i}^T v'_{\tilde{w}_k} = \log P(w_k \mid w_i)$$



GloVe

$$\boxed{P_{ij} = X_{ij}/X_i}$$

$$v_{w_i} \cdot v'_{\tilde{w}_j} = v_{w_i}^T v'_{\tilde{w}_j} = \log P(w_j \mid w_i)$$
$$= \log P_{ij} = \log(X_{ij}) - \log(X_i)$$

$$v_{w_i}^T v'_{\tilde{w}_j} + b_i + \tilde{b}_j = \log(X_{ij})$$

$$C(\theta) = \sum_{i,j=1}^{V} f(P_{ij})(v_{w_i} \cdot v'_{\tilde{w}_j} - \log P_{ij})^2$$

$$C(\theta) = \sum_{i,j=1}^{V} f(X_{ij})(v_{w_i}^T v'_{\tilde{w}_j} + b_i + \tilde{b}_j - \log X_{ij})^2$$

fast training, scalable, good performance even with small corpus, and small vectors

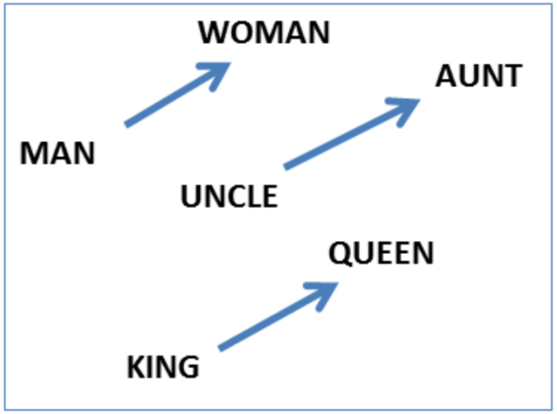
Word Vector Evaluation

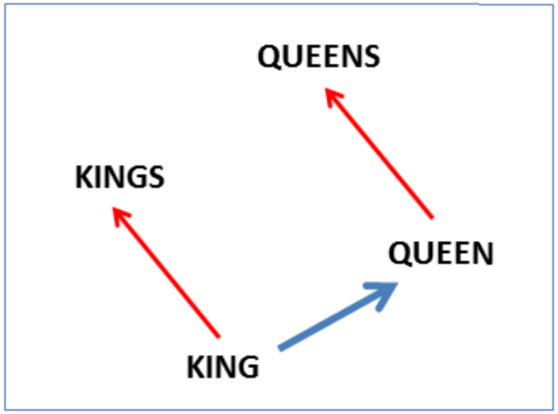
Intrinsic Evaluation – Word Analogies

Word linear relationship $w_A : w_B = w_C : w_x$

$$x = \arg\max_x \frac{(v_{w_B} - v_{w_A} + v_{w_C})^T v_{w_x}}{\left\| v_{w_B} - v_{w_A} + v_{w_C} \right\|}$$

Syntactic and **Semantic** example questions [link]





Issue: what if the information is there but not linear

Intrinsic Evaluation – Word Analogies

Word linear relationship $w_A : w_B = w_C : w_x$

Syntactic and **Semantic** example questions [link]

city---in---state

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 = Aus8n : Texas
Chicago : Illinois = Detroit : Michigan
Chicago : Illinois = Memphis : Tennessee
Chicago : Illinois = Boston : Massachusetts

Issue: different cities may have same name

capital---country

Abuja : Nigeria = Accra : Ghana
Abuja : Nigeria = Algiers : Algeria
Abuja : Nigeria = Amman : Jordan
Abuja : Nigeria = Ankara : Turkey
Abuja : Nigeria = Antananarivo : Madagascar
Abuja : Nigeria = Apia : Samoa
Abuja : Nigeria = Ashgabat : Turkmenistan
Abuja : Nigeria = Asmara : Eritrea
Abuja : Nigeria = Astana : Kazakhstan

Issue: can change with time

Intrinsic Evaluation – Word Analogies

Word linear relationship $w_A : w_B = w_C : w_x$

Syntactic and Semantic example questions [link]

superlative

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

past tense

dancing : danced = decreasing : decreased
dancing : danced = describing : described
dancing : danced = enhancing : enhanced
dancing : danced = falling : fell
dancing : danced = feeding : fed
dancing : danced = flying : flew
dancing : danced = generating : generated
dancing : danced = going : went
dancing : danced = hiding : hid
dancing : danced = hiding : hit

Intrinsic Evaluation – Word Correlation

Comparing word correlation with human-judged scores

Human-judged word correlation [link]

Word 1	Word 2	Human-Judged Score
tiger	cat	7.35
tiger	tiger	10.00
book	paper	7.46
computer	internet	7.58
plane	car	5.77
professor	doctor	6.62
stock	phone	1.62

Ambiguity: synonym or same word with different POSs

Extrinsic Evaluation – Subsequent Task

Goal: use word vectors in neural net models built for subsequent tasks

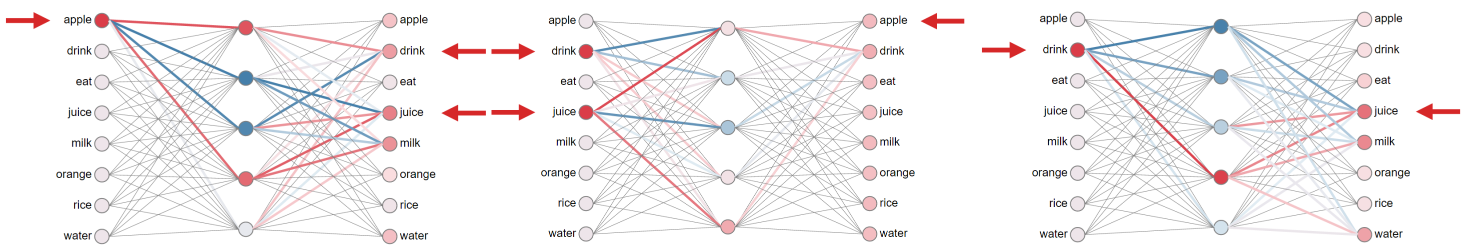
Benefit

- Ability to also classify words accurately
  - Ex. countries cluster together a classifying location words should be possible with word vectors
- Incorporate any information into them other tasks
  - Ex. project sentiment into words to find most positive/negative words in corpus

Concluding Remarks

Low dimensional word vector

- word2vec



Skip-gram CBOW LM

- GloVe: combining count-based and direct learning

Word vector evaluation

- Intrinsic: word analogy, word correlation
- Extrinsic: subsequent task