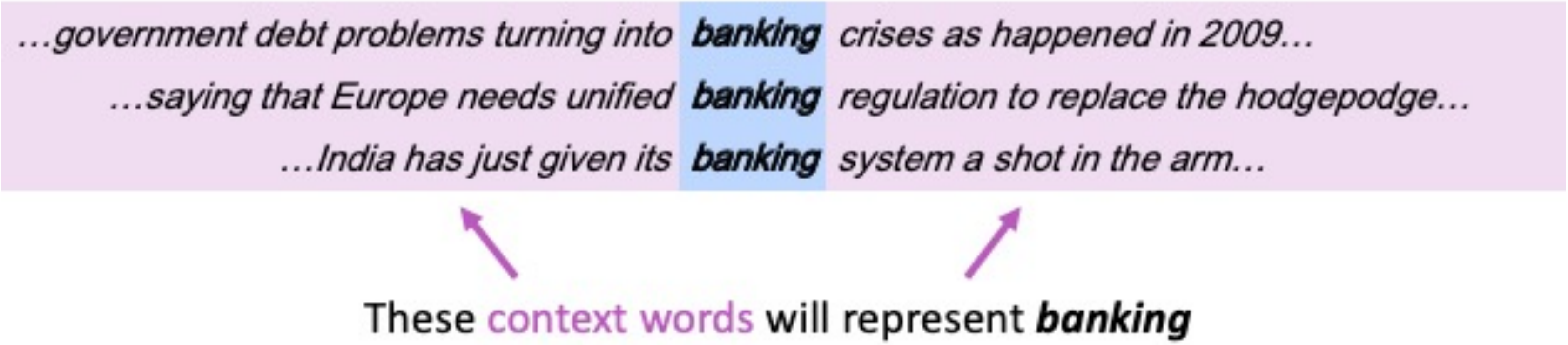
Lecture 03:
Word Embeddings

Overview

- Word meaning (semantics)
- Frequency based representation
- Sparse vector representation
- Dense vector representation (Distributional semantics)

Recap….

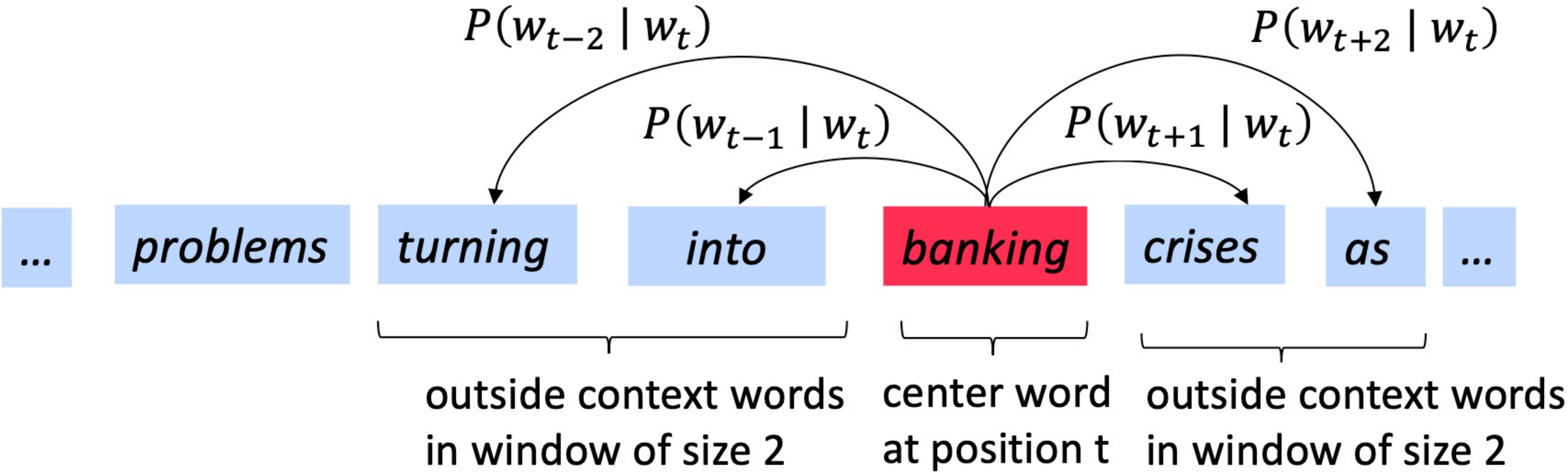
"You shall know a word by the company it keeps." Firth 1957



Based on Chris Manning

Recap....

- Example: when center word is *banking*



Based on Chris Manning

Recap….

Word2vec parameters and computation

$$
\begin{bmatrix} \bullet & \bullet & \bullet & \bullet & \bullet \\ \bullet & \bullet & \bullet & \bullet & \bullet \\ \bullet & \bullet & \bullet & \bullet & \bullet \\ \bullet & \bullet & \bullet & \bullet & \bullet \\ \bullet & \bullet & \bullet & \bullet & \bullet \\ \bullet & \bullet & \bullet & \bullet & \bullet \end{bmatrix}
\quad
\begin{bmatrix} \bullet & \bullet & \bullet & \bullet & \bullet \\ \bullet & \bullet & \bullet & \bullet & \bullet \\ \bullet & \bullet & \bullet & \bullet & \bullet \\ \bullet & \bullet & \bullet & \bullet & \bullet \\ \bullet & \bullet & \bullet & \bullet & \bullet \\ \bullet & \bullet & \bullet & \bullet & \bullet \end{bmatrix}
\qquad
\begin{bmatrix} \bullet \\ \bullet \\ \bullet \\ \bullet \\ \bullet \\ \bullet \end{bmatrix}
\quad
\begin{bmatrix} \bullet \\ \bullet \\ \bullet \\ \bullet \\ \bullet \\ \bullet \end{bmatrix}
$$

U	V	$U.{v_4}^T$	$\mathrm{softmax}(U.{v_4}^T)$
outside	center	dot product	probabilities

!

Same predictions at each position

We want a model that gives a reasonably high probability estimate to *all* words that occur in the context (fairly often)

Based on Chris Manning

Recap....

For each position $t = 1, \ldots, T$, predict the context word within a window of fixed size m, given the centre word w_j

$$\text{Likelihood} = L(\theta) = \prod_{t=1}^{T} \prod_{\substack{-m \leq j \leq m \\ j \neq 0}} P\left(w_{t+j} \mid w_t; \theta\right)$$

θ is all variables to be optimized

sometimes called *cost* or *loss* function

The objective function $J(\theta)$ is the (average) negative log likelihood:

$$J(\theta) = -\frac{1}{T} \log L(\theta) = -\frac{1}{T} \sum_{t=1}^{T} \sum_{\substack{-m \leq j \leq m \\ j \neq 0}} \log P\left(w_{t+j} \mid w_t; \theta\right)$$

Based on Chris Manning

Recap....

Training word2vec

- To minimize $J(\theta)$ we need to calculate $P(w_{t+j}|w_t;\theta)$
- To do this we define 2 vectors per word *w*:
  - $v_w \in V$ when *w* is a center word
  - $u_w \in U$ when *w* is a context word
- Then for a center word *c* and a context word *o:*

$$P(o|c) = \frac{\exp(u_o^T v_c)}{\sum_{w \in V} \exp(u_w^T v_c)}$$

Recap….

Exponentiation makes anything positive

$$P(o|c) = \frac{\exp(\boxed{u_o^T v_c})}{\boxed{\sum_{w \in V} \exp(u_w^T v_c)}}$$

Dot product compares similarity of *o* and *c*.
$u^T v = u.v = \sum_{i=1}^{n} u_i v_i$
Larger dot product = larger probability

Normalize over entire vocabulary
to give probability distribution

- SoftMax maps arbitrary values x_i to a probability distribution p_i

$$\mathrm{softmax}(x_i) = \frac{\exp(x_i)}{\sum_{j=1}^{n} \exp(x_j)} = p_i$$

Based on Chris Manning

Training a word2vec model

To train the word2vec model we compute all vector gradients

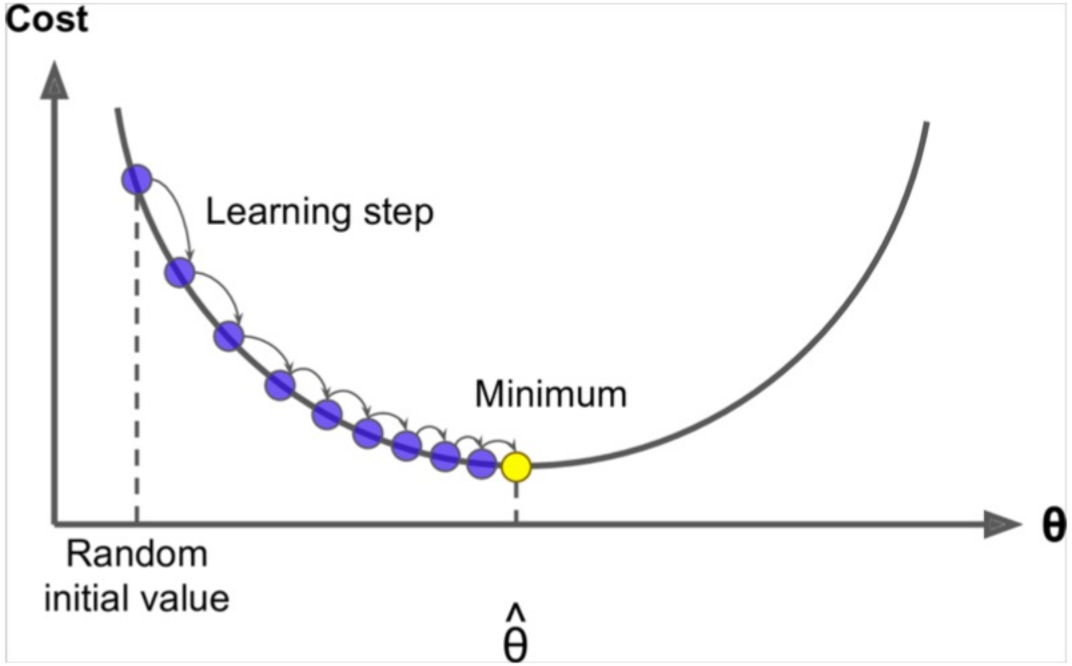
- Recall:
  - θ represents all model parameters
  - every word has two vectors
  - total of *V* vectors, each of *d*-dimension
    - *V*: vocabulary size
    - *d*: dimension of dense vector
- θ is optimized using gradient descend for
  - each center vector *v*
  - each outside vector *u*

$$\theta = \begin{bmatrix} v_{length} \\ v_{long} \\ \vdots \\ v_{zebra} \\ u_{length} \\ u_{long} \\ \vdots \\ u_{zebra} \end{bmatrix} \in \mathbb{R}^{2dV}$$

Optimization: gradient descent

Recall: Cost function to minimize $J(\theta)$

- Idea: for current θ, calculate gradient of $J(\theta)$ take small steps in direction of negative gradient.



- Update equation

$$\boldsymbol{\theta}^{new} = \boldsymbol{\theta}^{old} - \boldsymbol{\alpha} \nabla_{\boldsymbol{\theta}} \boldsymbol{J}(\boldsymbol{\theta})$$

- Update equation for single parameter

$$\boldsymbol{\theta}_j^{new} = \boldsymbol{\theta}_j^{old} - \boldsymbol{\alpha} \frac{\partial}{\partial \boldsymbol{\theta}_j^{old}} \boldsymbol{J}(\boldsymbol{\theta})$$

α is the learning rate

Based on Chris Manning

Optimization: stochastic gradient descent

- $J(\theta)$ is a function of all windows in the corpus
  - Large vocabulary => potentially millions of words
  - Computing the gradient $\nabla_\theta J(\theta)$ is very expensive (common in DL models)

- Stochastic gradient descent
  - Repeatedly sample gradients in each window
  - each window have at most *2m + 1* words

    so $\nabla_\theta J(\theta)$ is sparse!

$$\nabla_\theta J_t(\theta) = \begin{bmatrix} 0 \\ \vdots \\ \nabla_{v_{like}} \\ \vdots \\ 0 \\ \nabla_{u_I} \\ \vdots \\ \nabla_{u_{learning}} \\ \vdots \end{bmatrix} \in \mathbb{R}^{2dV}$$

stochastic gradient descent

- $\nabla_\theta J(\theta)$ is a spare matrix mini batch training result to spare parameter update

- We might only update the word vectors that appear!

- Solution: either you need sparse matrix update operations to only update certain rows of full embedding matrices *U* and *V*, or you need to keep around a hash for word vectors

word2vec

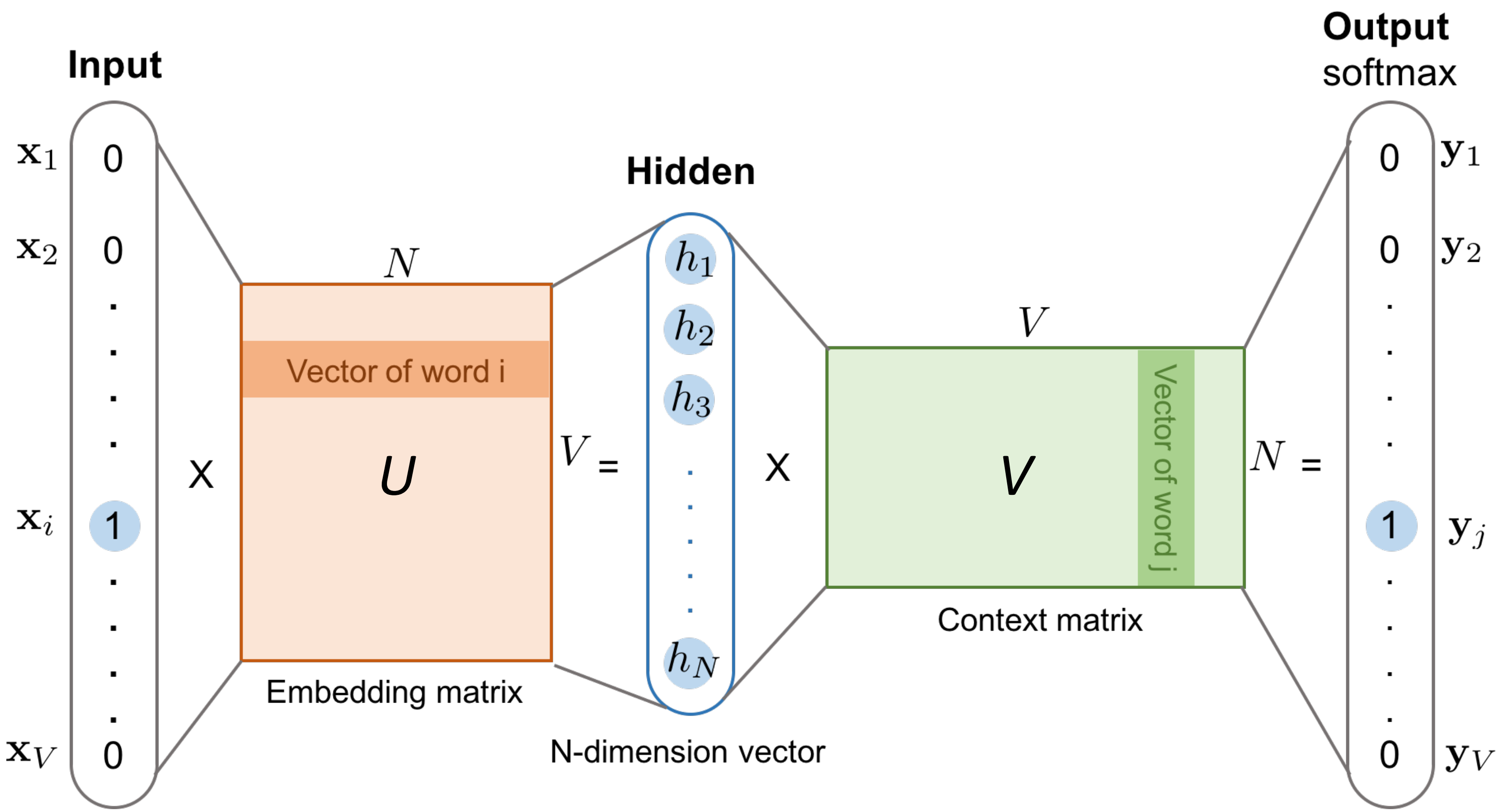
Two variants of word2vec algorithm:

1. Skip-Gram: use the current word *w* to predict its context
2. Continuous Bag of Word (CBOW): uses the context words to predict the current word *w*

Skip gram model

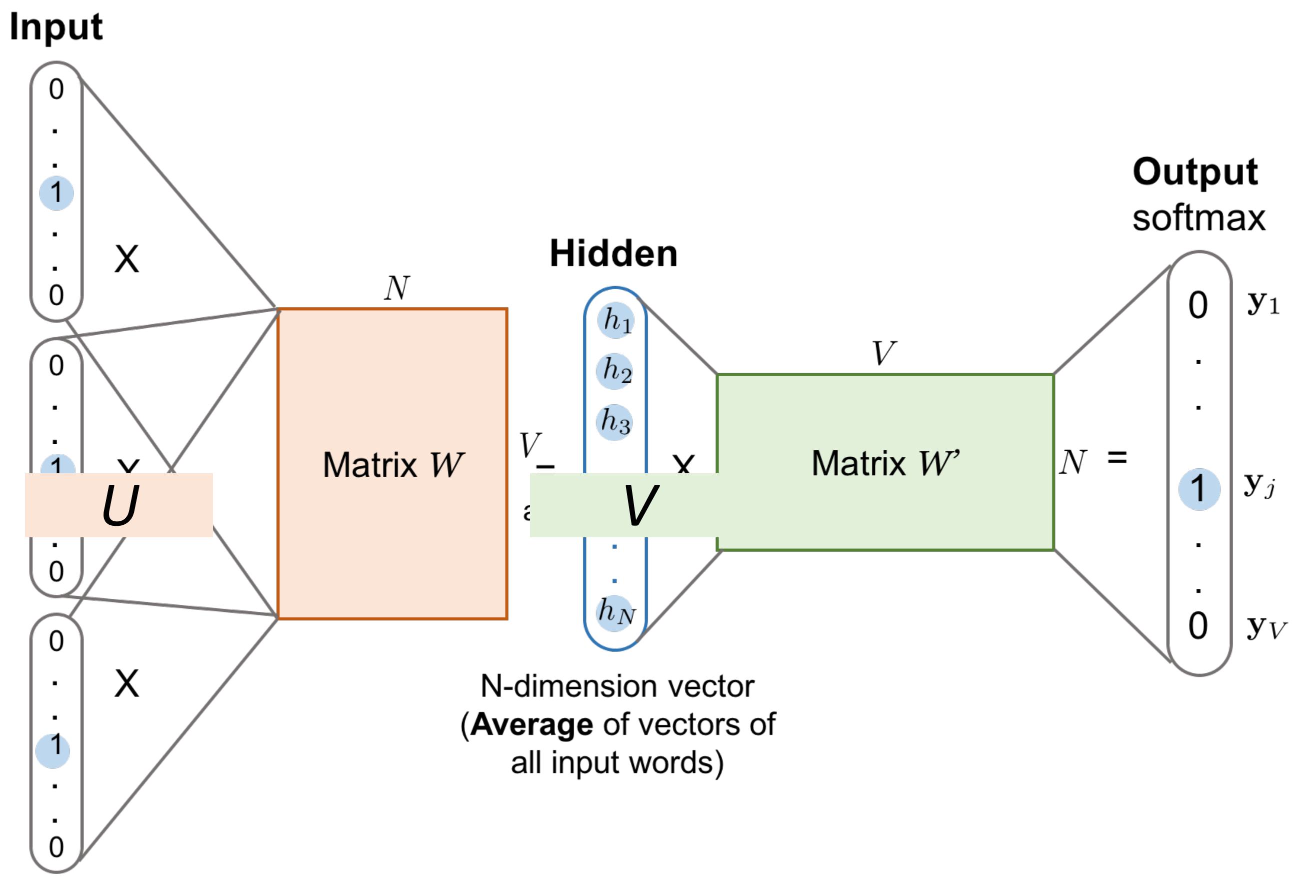
Notation:

- *N:* word embedding size
- *V*: vocabulary size
- $x, y \in \mathbb{R}^{V x\ 1}$: one-hot encoded words
- $U \in \mathbb{R}^{V x\ N}$: word embedding matrix
- $V \in \mathbb{R}^{N x\ V}$: word context matrix



CBOW model

In CBOW embedding



Skip-gram – negative sampling

- Naïve skip-gram model: modelling technique we have been discussing
- Problems:
  - Simple but expensive training method
  - With large vocabularies it might not be scalable

Skip-gram model with negative sampling

The normalization factor is computationally expensive

$$P(o|c) = \frac{\exp(u_o^T v_c)}{\boxed{\sum_{w \in V} \exp(u_w^T v_c)}}$$

Normalization factor

- Define positive pairs: centre word and word in its context window
- Define negative pairs: the centre word paired with a random word
  - Sample ***k*** words to decrease the number of training examples.
- Train a binary logistic regressions model to predict context words

Skip-gram model with negative sampling

- The negative sampling objective function is given by

maximizing the probability of two words co-occurring

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

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

- Maximize probability that real outside word appears, minimize probability that random words appear around centre word

Skip-gram model with negative sampling

- Sigmoid function is used to convert the estimates to probabilities

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

- To eliminate the negative effect of very frequent words such as "in", "the" a simple subsampling approach used

$$P(w) = U(w)^{3/4}/Z$$

- $U(w)$: is a unigram distribution of context words.
- The power ensures less frequent words be samples more often.

Skip-gram model with negative sampling

- Since $J_t(\theta)$ maximizes co-occurrence between words, why not measure it directly by taking the co-occurrence count?
- *U* and *V* matrix are both co-occurrence matrix

What is a co-occurrences matrix?

- Records the co-occurrence frequencies between a pair of words.
- Two type
  - Window based (commonly between 5 – 10 words)
  - Document based

Co-occurrence matrix

Two types of word co-occurrence matrix

- Row vector describes usage of word in a corpus of text
- Can be seen as coordinates of the point in an n-dimensional Euclidian space

Corpus

- I like deep learning.
- I like NLP.
- I enjoy flying.

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 with co-occurrences

- Left or right context is irrelevant
- Matrix size increases with vocabulary size
- Very high dimension: requires a lot of storage
- Matrix is extremely sparse since most words do not co-occur
- Models are less robust:
    - It is hard to incorporate out of sample (**new**) words or documents

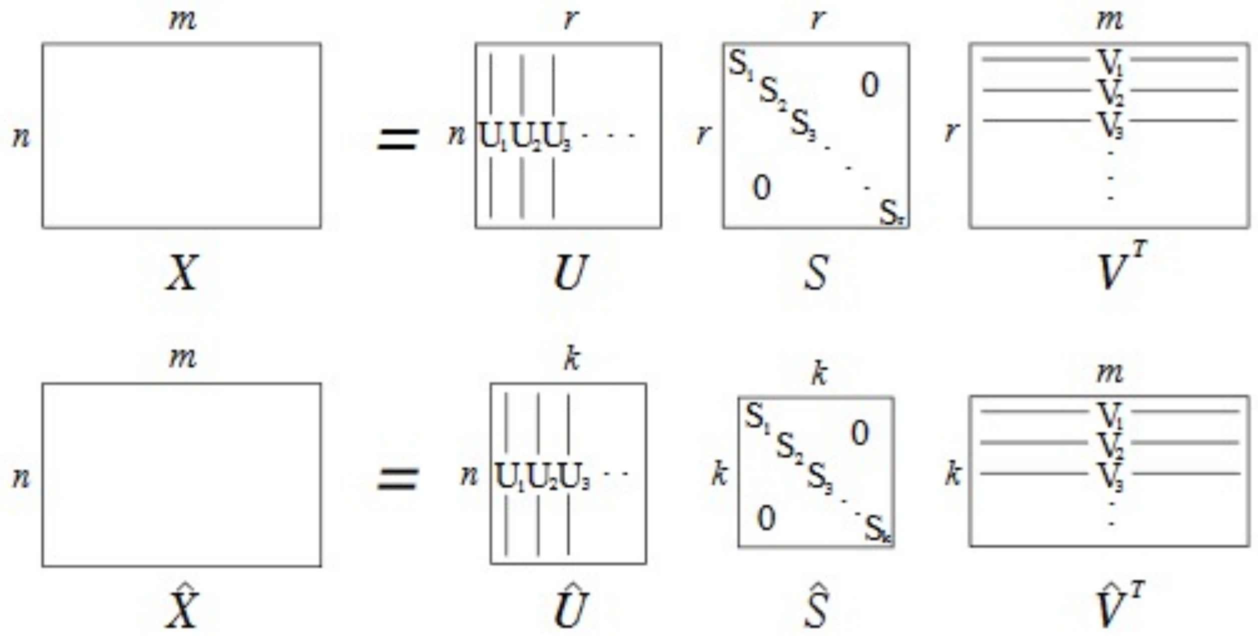
Solutions: co-occurrences

Create low dimensional vectors

- Convert co-occurrence matrix into a dense matrix of smaller dimension that captures most information
- Scale counts in the cell to minimize bias from high frequency words
- Use correlation instead of of raw counts
- How do we reduce high-dimensional co-occurrence matrix?

Singular value decomposition – SVD (LSA)

SVD factorizes a matrix X into $U\Sigma V^T$, where U and V are orthonormal



- Retain only k singular values in order to generalize.
- $\hat{X}$ is the best ranked k approximation to X in terms off least square

GloVe

An objective that attempts to create a semantic space with linear structure

	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

Probability ratios are more important than probabilities

GloVe

- ratio gives us some insight on the co-relation of the target word to a context word

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

Very small or large:

solid is related to ice but not steam, or
gas is related to steam but not ice

close to 1:

water is highly related to ice and steam, or
fashion is not related to ice or steam.

GloVe

- Try to find word embeddings such that (roughly)

$$\left(v_{c_1} - v_{c_2}\right)^T u_0 = \frac{P_{c_1\,o}}{P_{c_2\,o}}$$

- $P_{c\,o}$ is the probability of an output word *o* given a centre word *c*

- For example

$$v_{ice} - v_{stean} \approx u_{solid}$$
$$v_{steam} - v_{ice} \approx u_{gas}$$

Count based vs prediction-based word vectors

SVD – LSA

- Faster to train and it efficiently use word statistics
- Primarily used to capture word similarity
- Disproportionate importance given to large counts

Word2vec

- Scales with corpus size but with Inefficient usage of statistics
- Generate improved performance on other tasks
- Can capture complex patterns beyond word similarity

Evaluating word2vec model

- Intrinsic:
  - Evaluation on a specific/intermediate subtask
  - Fast to compute
  - Helps to understand that system
  - Not clear if really helpful unless correlation to real task is established
- Extrinsic:
  - Evaluation on a real task
  - Can take along time to compute accuracy
  - Unclear if the subsystem is the problem or its interaction or other subsystems
  - If replacing exactly one subsystem with another improves accuracy

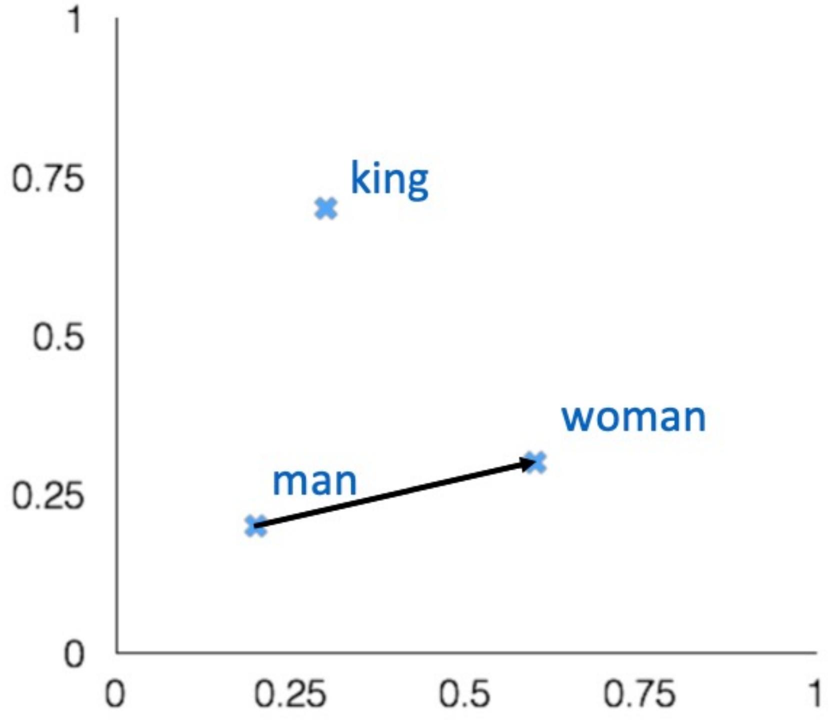
Intrinsic word vector evaluation

- Evaluate word vectors by how well their cosine distance after addition captures intuitive semantic and syntactic analogy questions.

a:b :: c:?

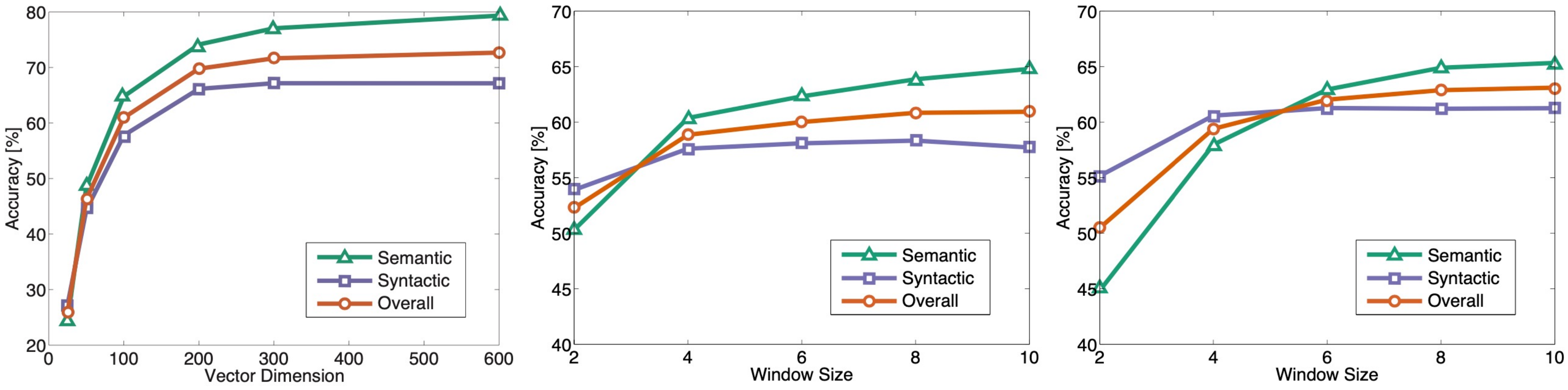
man:woman :: king:?

$$d = \arg\max_i \frac{(x_b - x_a + x_c)^T x_i}{||x_b - x_a + x_c||}$$



- Discarding the input words from the search!
- Problem: What if the information is there but not linear?

Evaluation and hyperparameter



(a) Symmetric context (b) Symmetric context (c) Asymmetric context

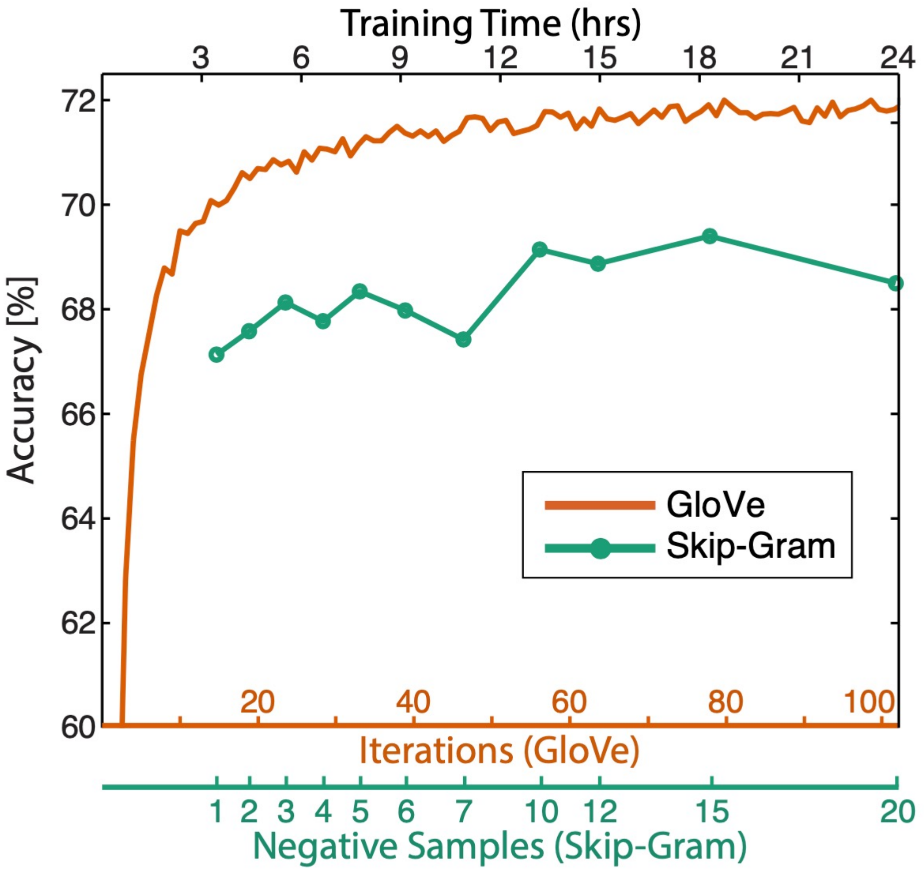
Dimensionality Window size Window size

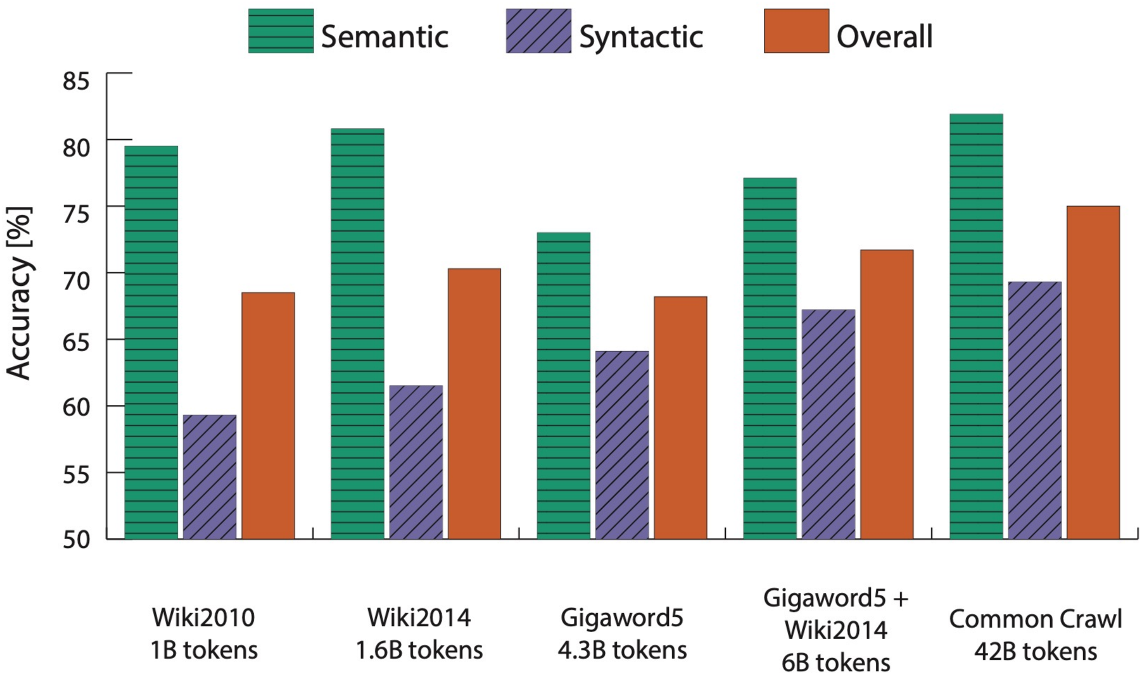
- Best dimension 300
- Asymmetric (only words to the left) context are not good
- A context window size of 8 is best for GloVe

Based on Chris Manning

Word2vec evaluation and hyperparameter

- More training and more data help improve the word vector.





Based on Chris Manning

Pre-trained word embeddings

- Word2vec: https://code.google.com/archive/p/word2vec/
- Fasttext: http://www.fasttext.cc/
- Glove: http://nlp.stanford.edu/projects/glove/
- Gensim: https://radimrehurek.com/gensim/