# Vector Semantics & Embeddings

## 1. Word Meaning

In a larger context, what is the meaning of word means?

Take for example

Bank ← lemma

- this could have two senses/concept
  1. Financial Institutions
  2. Side of River

Above lemma is an example of polysemous lemma

### 1.1. Synonym Relation

- This relationship b/w similar concept is called Synonymy
- even though two words can be Synonym, there exact usage could still differ

My big Sister != My large Sister

### 1.2. Similarity Relation

- Words with similar meanings. Not Synonyms, but share some element of meaning

Car, bicycle
Cow, Horse

### 1.3. Word Relatedness

Words are related in different ways
Semantic frame/field

Coffee, tea : Similar
Coffee, cup : Related

# Lecture-1: Introduction & Word Vectors

## 1. Word Distributional Semantics

- A core idea of modern NLP is
  A word's meaning is given by that
  frequently appear close by
  ↳ distributional Hypothesis

- when a word $w$ appears in a text,
  its context is the set of words
  that appear nearby (within a fixed size-window)

## 2. Word 2 vec

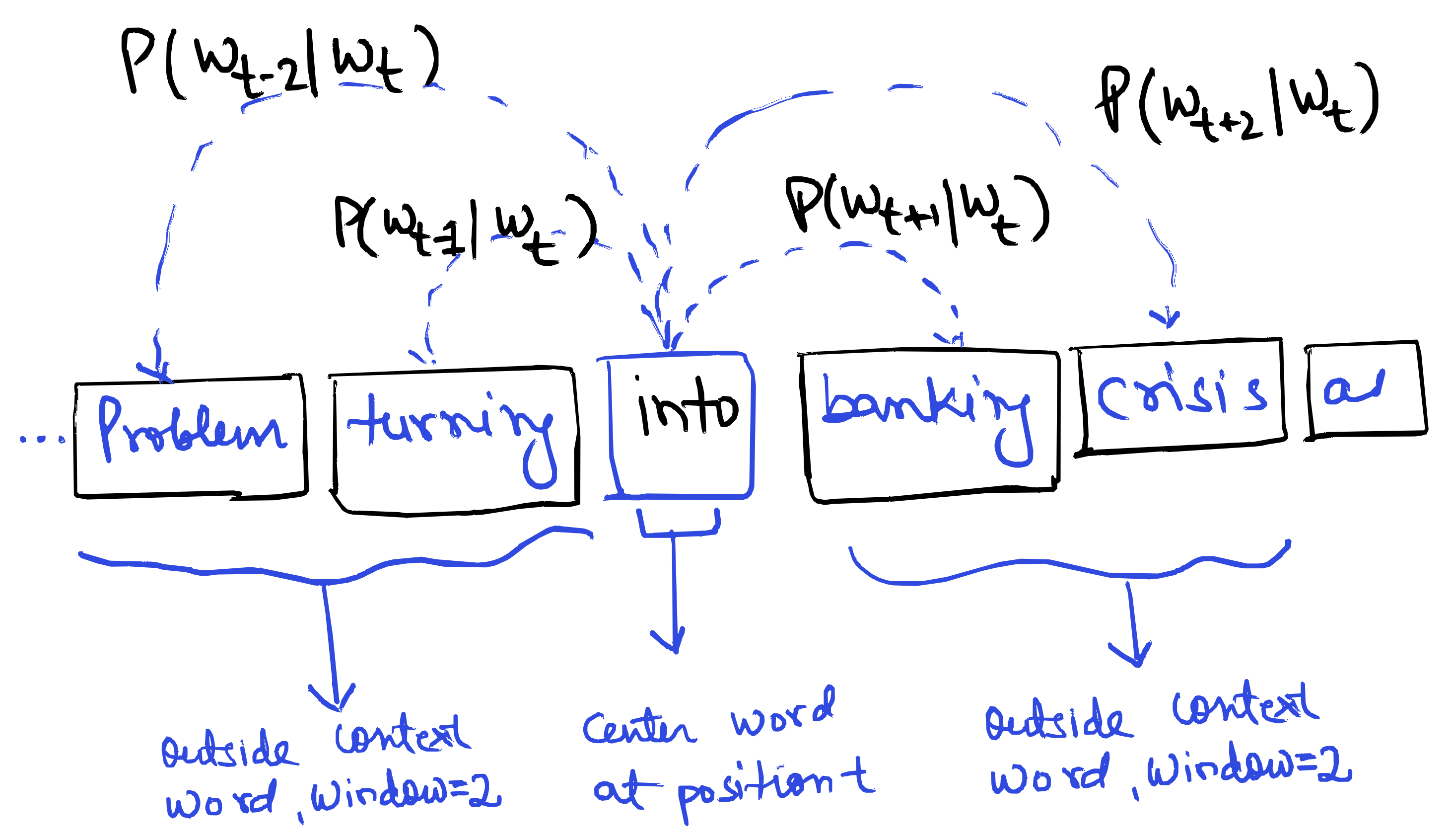
Core idea for word 2 vec is quite simple

- we have large ~~txt~~ corpus
- each word is represented by vector
- Go through each position $t$ in text.
  - ↳ center word: $c$
  - ↳ Context (outside) words: $o$

- use similarity of word vectors $c$ & $o$, & calc. $P(o|c)$
  ↳ keep adj. word vectors to maximize this problem.

# 2.1. Overview

Example for computing $P(w_{t+j} \mid w_t)$




For each position $t = 1, \dots, T$, predict context words within a window of fixed size "m", given center word "$w_t$".

likelihood function below is

$$L(\theta) = \prod_{t=1}^{T} \prod_{\substack{-m \le j \le m \\ j \ne 0}} P(w_{t+j} \mid w_t ; \theta)$$

and goal is to maximize it.

for computational reason, $\log L(\theta)$ is better behaved for optimization & -ve sign is added (so minimization)

Objective func: $J(\theta)$

$$J(\theta) = -\frac{1}{T} \sum_{t=1}^{T} \sum_{\substack{-m \le j \le m \\ j \ne 0}} \log P(w_{t+j} \mid w_t ; \theta)$$

# 2.2. Estimating Probability

Question: How to calculate $P(w_{t+j} \mid w_t, \theta)$

↳ use 2 vectors per word $w$

- $v_w$, when $w$ is center word
- $u_w$, when $w$ is context word

↳ Above word vectors are part of all parameters $\theta$

$P(o \mid c)$: Probability of context word "o" given center word "c"

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

② exponential make's everything positive

① dot product compare's similarity b/w o and c

$$\frac{\exp(u_o^T v_c)}{\sum_{w \in V} \exp(u_w^T v_c)}$$

③ Normalize over entire vocabulary to give pdf.

Softmax maps arbitrary $x_i$ to a p.d.f $p_i$

↳ "Max" amplifies Prob. of largest "$x_i$"

↳ "Soft" because some values to small $x_i$

## 2.3. Model Training via Grad. descent

Model training is done as we gradually adjust parameters to minimize loss

- $\theta$ represents all model parameters, in a long vector
- If each word is projected in "d" dimension, vocabulary size "V"
- Each word needs representation when it is
  - ↳ context
  - ↳ center
- Number of Params ($\theta$): $2 \times d \times V$

$$\theta = \begin{bmatrix} v_{aardvark} \\ v_a \\ \vdots \\ v_{zebra} \\ \vdots \\ u_{aardvark} \\ u_a \\ \vdots \\ u_{zebra} \end{bmatrix} \in \mathbb{R}^{2dV}$$

- This are optimized by gradient descent

### 2.3.1. Estimating the Gradient

$$J(\theta) = -\frac{1}{T}\sum_{t}\sum_{-j\leq m\leq j} \log P(w_{t+j}|w_t)$$

Probability of outside word "o", given center word "c", is

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

#### 2.3.1.1. Derivative w.r.t center word

$$\frac{\partial J_\theta}{\partial v_c} = -\frac{1}{T}\sum\sum \frac{\partial \log P(w_{t+j}|w_t)}{\partial v_c}$$

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

$$\frac{\partial \log P(w_{t+j}|w_t)}{\partial v_c} = \frac{\partial N^r}{\partial v_c} - \frac{\partial d^r}{\partial v_c}$$

Numerator & denominator

**Numerator**

$$\frac{\partial}{\partial v_c}\left(\log \exp(v_o^T v_c)\right)$$

$$\hookrightarrow \frac{\left(\frac{\partial \exp(v_o^T v_c)}{\partial v_c}\right)}{\exp(v_o^T v_c)}$$

$$\hookrightarrow \frac{\cancel{\exp(v_o^T v_c)}}{\cancel{\exp(v_o^T v_c)}} \times \frac{\partial(v_o^T v_c)}{\partial v_c}$$

$$\hookrightarrow v_o$$

**Denominator**

$$\frac{\partial}{\partial v_c}\left(\log \sum_{m\in V}\exp(u_m^T v_c)\right)$$

$$\rightarrow \frac{\frac{\partial}{\partial v_c}\left(\sum_{m=1}^{n}\exp(u_m^T v_c)\right)}{\sum_{m=1}^{n}\exp(u_m^T v_c)}$$

↓ Putting derivative inside

$$\rightarrow \frac{\sum_{m=1}^{n}\left(\frac{\partial}{\partial v_c}\exp(u_m^T v_c)\right)}{\sum_{m=1}^{n}\exp(u_m^T v_c)}$$

$$\rightarrow \frac{\sum_{m=1}^{n}\left(u_m \cdot \exp(u_m^T v_c)\right)}{\sum_{m=1}^{n}\exp(u_m^T v_c)}$$

$$\rightarrow \sum_{m=1}^{n}\underbrace{\left(\frac{\exp(u_m^T v_c)}{\sum \exp(u_m^T v_c)}\right)}_{\text{this is } P(m|c)} u_m$$

$$\rightarrow \underbrace{\sum_{m=1}^{n} u_m P(m|c)}_{\text{this is } E[u_m]}$$

combining $N^r$ & $D^r$

$$\nabla_{v_c} \log P(o|c)$$

$$\hookrightarrow u_o - E[u_m]$$

↳ observed - Expected

## 2.4. Gradient descent

update eqn for single param $\theta_j$

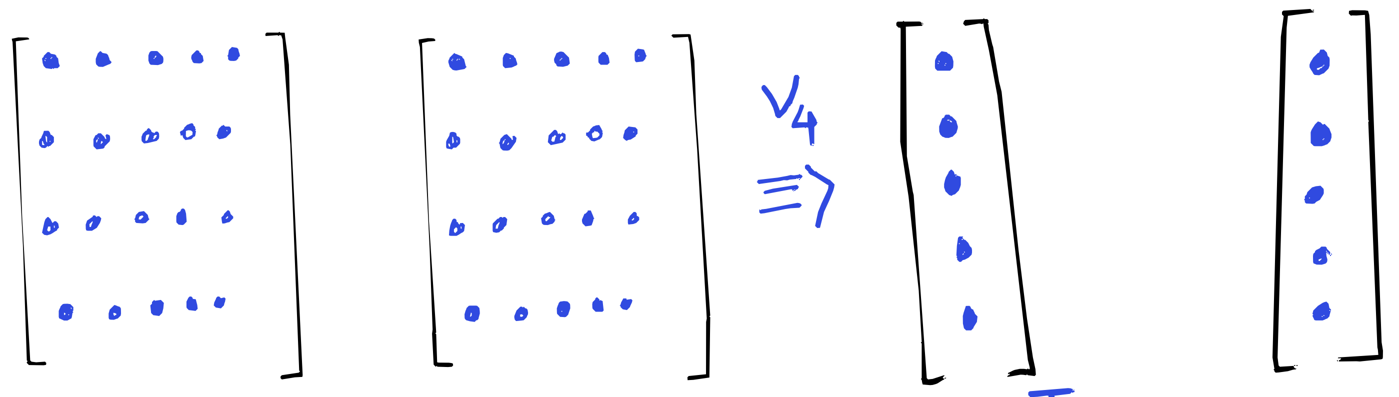
$$\theta_j^{new} = \theta_j^{old} - \alpha \frac{\partial L}{\partial \theta_j}$$

$\alpha$: learning rate

In Practice we use, SGD (stochastic gradient descent)

↳ we sample 32 sample's, compute gradient based on that

## 2.5. Word2Vec Parameter's




U outside | V center | $U.V_4^T$ dot.Prod. | softmax$(U.V_4^T)$ Probability

this is like "Bag of words" Model, where we don't care on structure of sentence.

there are two model variants:

- Skip-gram
  → Predict outside word given center
- continious Bag of Words
  → Predict center word from bag-of-context words

loss function for training

→ Naive Softmax

→ heirarchial Softmax

→ Negative sampling

## 2.6.1. Skip-Gram with Neg. Sampling

- In system of 2013, normalization term was computationally expensive

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

↑ Big sum over many words

- K negative samples (conditioned on word probabilities)

$$P(w) = \frac{f^{3/4}(w)}{\sum_{w \in V} f^{3/4}(w)}$$

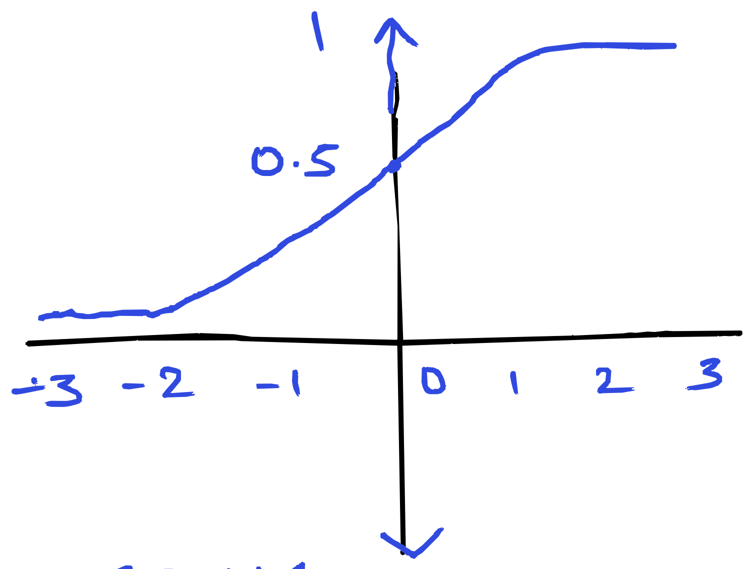
$f(w)$ is unigram probability of $w$, where $w \in V$

- Maximize probability of real outside word; minimize prob. of random word

$$J(u_o, v_c, U) = -\log \sigma(u_o^T v_c) - \sum_{k \in \{K \text{ sampled indices}\}} \log \sigma(-u_k^T v_c)$$

Sigmoid rather than softmax

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




- SGD step with SGNS
  - → take gradients at each window
  - → each window word have at most
    - ↳ $2m + 1$ words
    - ↳ $2km$ negative words

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

# 3. why not capture co-occurence counts directly

word2vec iterates through whole corpus many time, why don't we accumulate stats of words appear near to each other

How often word "octopus" appears to "jellyfish"

We can have window based co-occurence matrix

toy example
window = 1, symmetric

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

| Counts | I | like | enjoy | deep |
|---|---|---|---|---|
| I | 0 | 2 | 1 | 0 |
| like | 2 | 0 | 0 | 1 |
| enjoy | 1 | 0 | 0 | 0 |
| deep | 0 | 1 | 0 | 0 |

Building co-occurence matrix is expensive!
for vocab $\in V$, size $\approx V \times V$

## 3.1. Co-occurence Vectors

→ Simple count based matrix are big, expensive to compute & sparse

→ uses PCA & S.V.D to reduce dimen.

X : co-occurence matrix
U : orthonormal
V : vector

$$X = U \Sigma V^T$$

we want to retain only "k" singular values
$\hat{X}$ approximation of X in terms of OLS

→ expensive to compute for large matrix

### 3.1.1. Some Hacks

COALS model (2006)
→ log to counts
→ use pearson correlation instead of counts, set -ve corr to 0

- Interesting semantic Patterns also observed

this was inspiration of GLoVe

# 3.2. GLoVe Model

Main Idia : encoding meaning components in Vector difference

→ Ratio of co-occurrence probabilities can encode meaning components

↳ this needs to captured as linear meaning components in Vector space

| | x = solid | x = gas | x = water | x = random |
|---|---|---|---|---|
| $P(x \mid ice)$ | large | small | large | small |
| $P(x \mid steam)$ | small | large | large | small |

$\frac{P(x \mid ice)}{P(x \mid steam)}$ : Large small ~1 ~1

for real corpus, this was observed

(x=fashion)

$\frac{P(x \mid ice)}{P(x \mid steam)}$ : 8.9 $8.5 \times 10^{-2}$ 1.32 0.96

- to make it linear, log is taken.

$$w_i \cdot w_j = \log P(i \mid j)$$

vector diff. beng

$$w_x \cdot (w_a - w_b) = w_x \cdot w_a - w_x \cdot w_b = \log \frac{P(x \mid a)}{P(x \mid b)}$$

$$\log P(x \mid a) - \log P(x \mid b)$$

loss function :

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

$w_i$ : word-vector for main word $i$, $b_i$ is bias

$\tilde{w}_j$ : word-vec. for context word $j$, $\tilde{b}_j$ is bias

$X_{ij}$ : How often $j$ appears near word $i$

$\log X_{ij}$ : target we want to appr

$f(X_{ij})$ : weighting function

↳ decide how Imp. the word pair is

↳ down weights rare co-occur. pairs

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

typically,

$x_{max} = 100$

$\alpha = 0.75$

# 4. Word Vectors Evaluation

In N.L.P there are 2 kinds of evaluation,
Intrinsic vs. Extrinsic

→ Intrinsic eval helps specific tests like
MMLU

→ Extrinsic eval test on real task

## 4.1. Intrinsic evals

→ Analogy King : Man : : Queen : Women

→ word sim eval

## 4.2. Extrinsic evals

→ Named Entity Recognication