# Word Representation

# One-Hot Word Representation

Given a fixed set of vocabulary $V = w_1, w_2, \ldots, w_{|V|}$, one very intuitive way to represent a word w is to encode it with a $|V|$ −dimensional vector $w$, where each dimension of $w$ is either $0$ or $1$. Only one dimension in $w$ can be $1$ while all the other dimensions are $0$. Formally, each dimension of w can be represented as

$$\mathbf{w}_i = \begin{cases} 1 & \text{if } w = w_i \\ 0 & \text{otherwise.} \end{cases}$$

# One-Hot Word Representation

What are the weaknesses of one-hot vectors?

# Distributed Word Representation

Probabilistic language models can be used to assign a probability to a sentence in many NLP tasks

Machine Translation:

– P(high winds tonight) > P(large winds tonight)

Spell Correction:

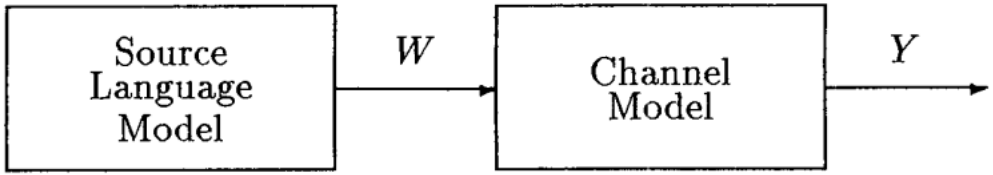
– Then office is about ten minutes from here

– P(The Office is) > P(Then office is)

Speech Recognition:

– P(I saw a van) >> P(eyes awe of an)

# Distributed Word Representation




$$\Pr(W) \times \Pr(Y \mid W) = \Pr(W, Y)$$

# Distributed Word Representation

An n-gram is a sequence of $N$ words:

– A $1-gram$ (unigram) is a single word sequence of words like “I” or “ do”.

– A $2-gram$ (bigram) is a two-word sequence of words like “I do”, “do my”, or ”my homework”.

– A $3-gram$ (trigram) is a three-word sequence of words like “I do my”, or “do my homework”.

# Distributed Word Representation

probabilities of entire sequences like $P(w_1, w_2, \ldots, w_n)$

$$P(w_{1:n}) = P(w_1)P(w_2|w_1)P(w_3|w_{1:2})\ldots P(w_n|w_{1:n-1})$$
$$= \prod_{k=1}^{n} P(w_k|w_{1:k-1})$$

$$P(w_n|w_{1:n-1}) \approx P(w_n|w_{n-N+1:n-1})$$

$$P(w_n|w_{1:n-1}) \approx P(w_n|w_{n-N+1:n-1})$$

$N$ here to mean the $n-gram$ size

# Distributed Word Representation

Maximum likelihood estimation:

$$P(w_i) = \frac{C(w_i)}{\sum_w C(w)}$$

$$P(w\_i|w_{i-1}) = \frac{C(w_{i-1},w_i)}{\sum_w C(w_{i-1},w)}$$

$$P(w\_i|w_{i-1}, w_{i-2}) = \frac{C(w_{i-1},w_{i-2},w_i)}{\sum_w C(w_{i-1},w_{i-2},w)}$$

Where, $C(w_i)$ is the count of $w_i$ in the corpus

# Distributed Word Representation

Maximum likelihood estimation:

$$P(w_i) = \frac{C(w_i)}{\sum_w C(w)}$$

$$P(w_i|w_{i-1}) = \frac{C(w_{i-1},.w_i)}{\sum_w C(w_{i-1}, w)}$$

$$P(w_i|w_{i-1}, w_{i-2}) = \frac{C(w_{i-1}, w_{i-2}, w_i)}{\sum_w C(w_{i-1},w_{i-2}, w)}$$

$$P(w_n|w_{n-N+1:n-1}) = \frac{C(w_{n-N+1:n-1}\ w_n)}{C(w_{n-N+1:n-1})}$$

# Distributed Word Representation

Let's work through an example using a mini-corpus of three sentences. We'll first need to augment each sentence with a special symbol $< s >$ at the beginning of the sentence, to give us the bigram context of the first word. We'll also need a special end-symbol $</s >$

<s> I am Sam </s>
<s> Sam I am </s>
<s> I do not like green eggs and ham </s>

What is P(I|<s>), P(Sam|<s>), P(am|I), P(</s>|Sam), P(Sam|am), P(do|I)

# Distributed Word Representation

Berkeley Restaurant Project corpus of 9332 sentences

| | i | want | to | eat | chinese | food | lunch | spend |
|---|---|---|---|---|---|---|---|---|
| **i** | 5 | 827 | 0 | 9 | 0 | 0 | 0 | 2 |
| **want** | 2 | 0 | 608 | 1 | 6 | 6 | 5 | 1 |
| **to** | 2 | 0 | 4 | 686 | 2 | 0 | 6 | 211 |
| **eat** | 0 | 0 | 2 | 0 | 16 | 2 | 42 | 0 |
| **chinese** | 1 | 0 | 0 | 0 | 0 | 82 | 1 | 0 |
| **food** | 15 | 0 | 15 | 0 | 1 | 4 | 0 | 0 |
| **lunch** | 2 | 0 | 0 | 0 | 0 | 1 | 0 | 0 |
| **spend** | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |

| | i | want | to | eat | chinese | food | lunch | spend |
|---|---|---|---|---|---|---|---|---|
| **i** | 0.002 | 0.33 | 0 | 0.0036 | 0 | 0 | 0 | 0.00079 |
| **want** | 0.0022 | 0 | 0.66 | 0.0011 | 0.0065 | 0.0065 | 0.0054 | 0.0011 |
| **to** | 0.00083 | 0 | 0.0017 | 0.28 | 0.00083 | 0 | 0.0025 | 0.087 |
| **eat** | 0 | 0 | 0.0027 | 0 | 0.021 | 0.0027 | 0.056 | 0 |
| **chinese** | 0.0063 | 0 | 0 | 0 | 0 | 0.52 | 0.0063 | 0 |
| **food** | 0.014 | 0 | 0.014 | 0 | 0.00092 | 0.0037 | 0 | 0 |
| **lunch** | 0.0059 | 0 | 0 | 0 | 0 | 0.0029 | 0 | 0 |
| **spend** | 0.0036 | 0 | 0.0036 | 0 | 0 | 0 | 0 | 0 |

| i | want | to | eat | chinese | food | lunch | spend |
|---|---|---|---|---|---|---|---|
| 2533 | 927 | 2417 | 746 | 158 | 1093 | 341 | 278 |

# Distributed Word Representation

$$P(\texttt{i}|\texttt{<s>}) = 0.25 \qquad P(\texttt{english}|\texttt{want}) = 0.0011$$
$$P(\texttt{food}|\texttt{english}) = 0.5 \qquad P(\texttt{</s>}|\texttt{food}) = 0.68$$

Calculate with 2-gram

$$P(\texttt{<s> i want english food </s>})$$

| | i | want | to | eat | chinese | food | lunch | spend |
|---|---|---|---|---|---|---|---|---|
| **i** | 0.002 | 0.33 | 0 | 0.0036 | 0 | 0 | 0 | 0.00079 |
| **want** | 0.0022 | 0 | 0.66 | 0.0011 | 0.0065 | 0.0065 | 0.0054 | 0.0011 |
| **to** | 0.00083 | 0 | 0.0017 | 0.28 | 0.00083 | 0 | 0.0025 | 0.087 |
| **eat** | 0 | 0 | 0.0027 | 0 | 0.021 | 0.0027 | 0.056 | 0 |
| **chinese** | 0.0063 | 0 | 0 | 0 | 0 | 0.52 | 0.0063 | 0 |
| **food** | 0.014 | 0 | 0.014 | 0 | 0.00092 | 0.0037 | 0 | 0 |
| **lunch** | 0.0059 | 0 | 0 | 0 | 0 | 0.0029 | 0 | 0 |
| **spend** | 0.0036 | 0 | 0.0036 | 0 | 0 | 0 | 0 | 0 |

# Distributed Word Representation

$P(\texttt{i}\,|\,\texttt{<s>}) = 0.25$
$P(\texttt{english}\,|\,\texttt{want}) = 0.0011$
$P(\texttt{food}\,|\,\texttt{english}) = 0.5$
$P(\texttt{</s>}\,|\,\texttt{food}) = 0.68$

| | i | want | to | eat | chinese | food | lunch | spend |
|---|---|---|---|---|---|---|---|---|
| **i** | 0.002 | 0.33 | 0 | 0.0036 | 0 | 0 | 0 | 0.00079 |
| **want** | 0.0022 | 0 | 0.66 | 0.0011 | 0.0065 | 0.0065 | 0.0054 | 0.0011 |
| **to** | 0.00083 | 0 | 0.0017 | 0.28 | 0.00083 | 0 | 0.0025 | 0.087 |
| **eat** | 0 | 0 | 0.0027 | 0 | 0.021 | 0.0027 | 0.056 | 0 |
| **chinese** | 0.0063 | 0 | 0 | 0 | 0 | 0.52 | 0.0063 | 0 |
| **food** | 0.014 | 0 | 0.014 | 0 | 0.00092 | 0.0037 | 0 | 0 |
| **lunch** | 0.0059 | 0 | 0 | 0 | 0 | 0.0029 | 0 | 0 |
| **spend** | 0.0036 | 0 | 0.0036 | 0 | 0 | 0 | 0 | 0 |

$$
\begin{aligned}
&P(\texttt{<s> i want english food </s>}) \\
&\quad = P(\texttt{i}\,|\,\texttt{<s>})P(\texttt{want}\,|\,\texttt{i})P(\texttt{english}\,|\,\texttt{want}) \\
&\qquad\qquad P(\texttt{food}\,|\,\texttt{english})P(\texttt{</s>}\,|\,\texttt{food}) \\
&\quad = .25 \times .33 \times .0011 \times 0.5 \times 0.68 \\
&\quad = .000031
\end{aligned}
$$

# Latent Semantic Analysis

Latent Semantic Analysis (LSA) is a family of strategies derived from vector space models, which could capture word semantics much better. LSA aims to explore latent factors for words and documents by matrix factorization to improve the estimation of word similarities. The SVD of word-document matrix $M$ yields three matrices U, $\Sigma$ and $V$ such that:

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

# Latent Semantic Analysis

**Vector Space Model:**
LSA operates within the vector space model, where each document and term is represented as a vector in a high-dimensional space.
**Term-Document Matrix:**
Create a term-document matrix where each row corresponds to a term, each column corresponds to a document, and the matrix elements represent the frequency of a term in a document.
**Singular Value Decomposition (SVD):**
Apply Singular Value Decomposition to decompose the term-document matrix into three matrices: U (term space), Σ (diagonal matrix of singular values), and V to the powerT (document space).

# Latent Semantic Analysis

Let $X$ be a matrix where element $(i, j)$ describes the occurrence of term $i$ in document $j$ (this can be, for example, the frequency). $X$ will look like

$$
\begin{array}{cc}
 & \mathbf{d}_j \\
 & \downarrow \\
\mathbf{t}_i^T \rightarrow & \begin{bmatrix}
x_{1,1} & \dots & x_{1,j} & \dots & x_{1,n} \\
\vdots & \ddots & \vdots & \ddots & \vdots \\
x_{i,1} & \dots & x_{i,j} & \dots & x_{i,n} \\
\vdots & \ddots & \vdots & \ddots & \vdots \\
x_{m,1} & \dots & x_{m,j} & \dots & x_{m,n}
\end{bmatrix}
\end{array}
$$

# Latent Semantic Analysis

Now a row in this matrix will be a vector corresponding to a term, giving its relation to each document:

$$\mathbf{t}_i^T = \begin{bmatrix} x_{i,1} & \dots & x_{i,j} & \dots & x_{i,n} \end{bmatrix}$$

Likewise, a column in this matrix will be a vector corresponding to a document, giving its relation to each term:

$$\mathbf{d}_j = \begin{bmatrix} x_{1,j} \\ \vdots \\ x_{i,j} \\ \vdots \\ x_{m,j} \end{bmatrix}$$

# Latent Semantic Analysis

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

$$\begin{matrix} & X & & U & & \Sigma & & V^T \\ & (\mathbf{d}_j) & & & & & & (\hat{\mathbf{d}}_j) \\ & \downarrow & & & & & & \downarrow \\ (\mathbf{t}_i^T) \rightarrow & \begin{bmatrix} x_{1,1} & \dots & x_{1,j} & \dots & x_{1,n} \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ x_{i,1} & \dots & x_{i,j} & \dots & x_{i,n} \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ x_{m,1} & \dots & x_{m,j} & \dots & x_{m,n} \end{bmatrix} & = & (\hat{\mathbf{t}}_i^T) \rightarrow & \begin{bmatrix} \begin{bmatrix} \\ \mathbf{u}_1 \\ \\ \end{bmatrix} \dots \begin{bmatrix} \\ \mathbf{u}_l \\ \\ \end{bmatrix} \end{bmatrix} & \cdot & \begin{bmatrix} \sigma_1 & \dots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \dots & \sigma_l \end{bmatrix} & \cdot & \begin{bmatrix} [ & \mathbf{v}_1 & ] \\ & \vdots & \\ [ & \mathbf{v}_l & ] \end{bmatrix} \end{matrix}$$

# Latent Semantic Analysis

| | cheese | bread | goat | sheep |
|---|---|---|---|---|
| cheese | 14 | 7 | 5 | 1 |
| bread | 7 | 12 | 0 | 0 |
| goat | 5 | 0 | 8 | 12 |
| sheep | 1 | 0 | 12 | 2 |

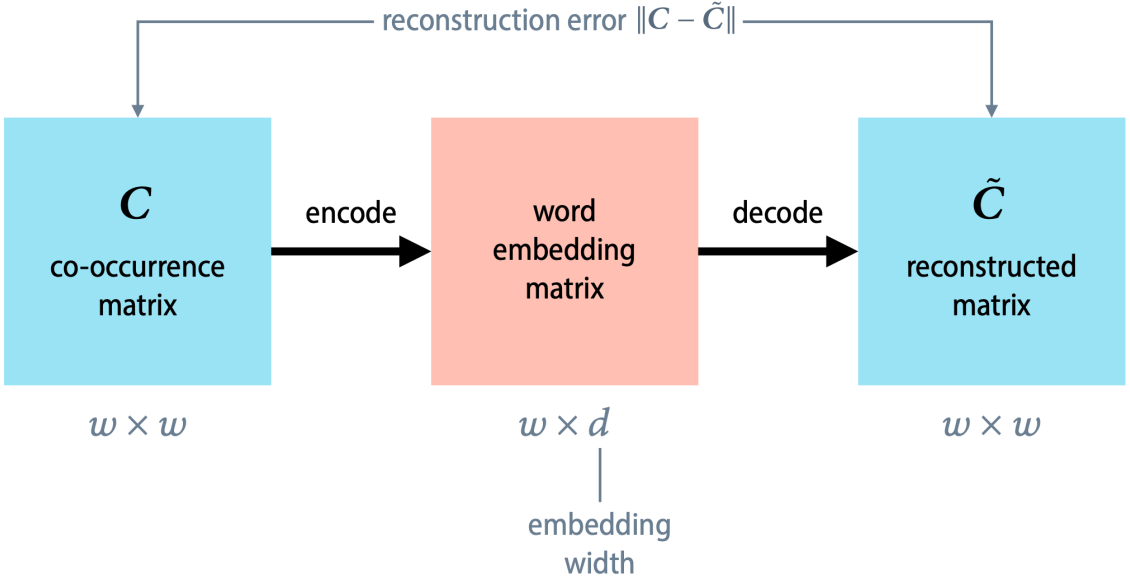
word vector for *cheese*

# Latent Semantic Analysis

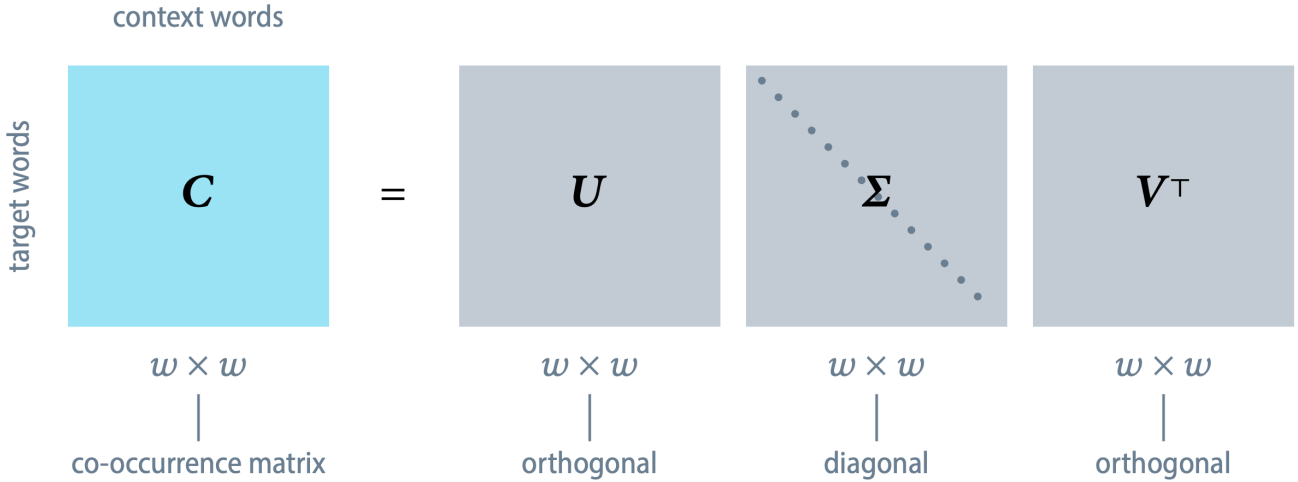
The row vectors of co-occurrence matrices are high-dimensional, but we want word embeddings to be low-dimensional.
Ex. Hundreds instead of hundreds of thousands of dimensions

One idea is to use dimensionality reduction and find a lowerdimensional representation of the co-occurrence matrix.

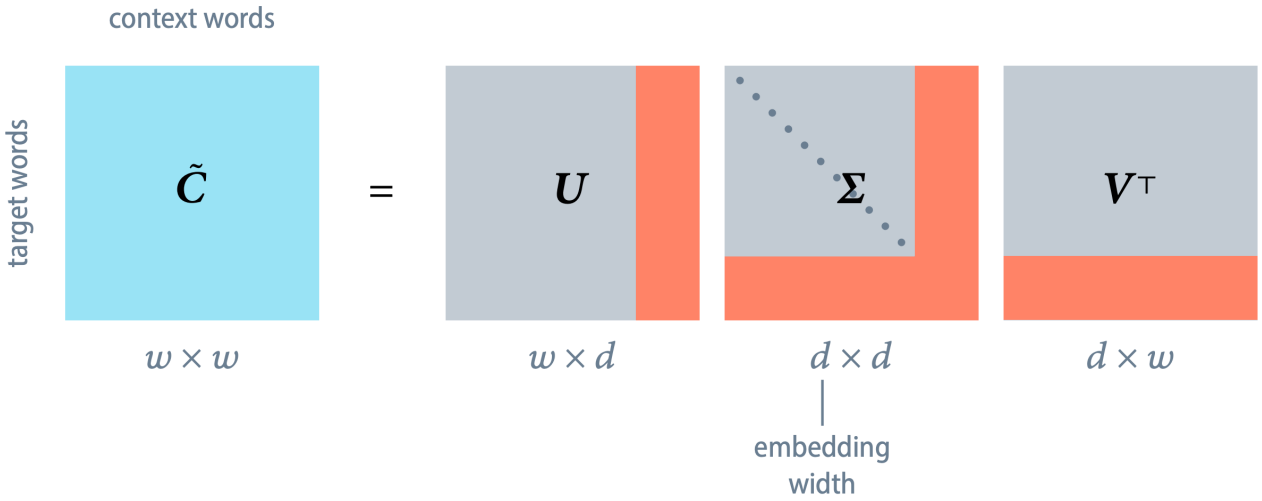
# Latent Semantic Analysis



# Latent Semantic Analysis



# Latent Semantic Analysis



# Latent Semantic Analysis

| | | | |
|---|---|---|---|
| 14 | 7 | 5 | 1 |
| 7 | 12 | 0 | 0 |
| 5 | 0 | 8 | 12 |
| 1 | 0 | 12 | 2 |

$U$

| | | | |
|---|---|---|---|
| −0.678 | 0.293 | 0.124 | 0.662 |
| −0.445 | 0.562 | −0.044 | −0.696 |
| −0.491 | −0.591 | −0.628 | −0.124 |
| −0.318 | −0.499 | 0.767 | −0.248 |

$\Sigma$

| | | | |
|---|---|---|---|
| 22.676 | 0.000 | 0.000 | 0.000 |
| 0.000 | 15.644 | 0.000 | 0.000 |
| 0.000 | 0.000 | 7.655 | 0.000 |
| 0.000 | 0.000 | 0.000 | 5.335 |

$V^\top$

| | | | |
|---|---|---|---|
| −0.678 | −0.445 | −0.491 | −0.318 |
| 0.293 | 0.562 | −0.591 | −0.499 |
| −0.124 | 0.044 | 0.628 | −0.767 |
| 0.662 | −0.696 | −0.124 | −0.248 |

# Latent Semantic Analysis

$U$

| | | | |
|---|---|---|---|
| −0.678 | 0.293 | 0.124 | 0.662 |
| −0.445 | 0.562 | −0.044 | −0.696 |
| −0.491 | −0.591 | −0.628 | −0.124 |
| −0.318 | −0.499 | 0.767 | −0.248 |

$\Sigma$

| | | | |
|---|---|---|---|
| 22.676 | 0.000 | 0.000 | 0.000 |
| 0.000 | 15.644 | 0.000 | 0.000 |
| 0.000 | 0.000 | 7.655 | 0.000 |
| 0.000 | 0.000 | 0.000 | 5.335 |

$V^T$

| | | | |
|---|---|---|---|
| −0.678 | −0.445 | −0.491 | −0.318 |
| 0.293 | 0.562 | −0.591 | −0.499 |
| −0.124 | 0.044 | 0.628 | −0.767 |
| 0.662 | −0.696 | −0.124 | −0.248 |

reconstruction d = 3

| | | | |
|---|---|---|---|
| 11.659 | 9.459 | 5.439 | 1.878 |
| 9.459 | 9.418 | −0.461 | −0.922 |
| 5.439 | −0.461 | 7.918 | 11.835 |
| 1.878 | −0.922 | 11.835 | 1.671 |

reconstruction error: 1.779

| | | | |
|---|---|---|---|
| 14 | 7 | 5 | 1 |
| 7 | 12 | 0 | 0 |
| 5 | 0 | 8 | 12 |
| 1 | 0 | 12 | 2 |

# Latent Semantic Analysis

$U$

| | | | |
|---|---|---|---|
| −0.678 | 0.293 | 0.124 | 0.662 |
| −0.445 | 0.562 | −0.044 | −0.696 |
| −0.491 | −0.591 | −0.628 | −0.124 |
| −0.318 | −0.499 | 0.767 | −0.248 |

$\Sigma$

| | | | |
|---|---|---|---|
| 22.676 | 0.000 | 0.000 | 0.000 |
| 0.000 | 15.644 | 0.000 | 0.000 |
| 0.000 | 0.000 | 7.655 | 0.000 |
| 0.000 | 0.000 | 0.000 | 5.335 |

$V^\top$

| | | | |
|---|---|---|---|
| −0.678 | −0.445 | −0.491 | −0.318 |
| 0.293 | 0.562 | −0.591 | −0.499 |
| −0.124 | 0.044 | 0.628 | −0.767 |
| 0.662 | −0.696 | −0.124 | −0.248 |

reconstruction d = 3

| | | | |
|---|---|---|---|
| 11.659 | 9.459 | 5.439 | 1.878 |
| 9.459 | 9.418 | −0.461 | −0.922 |
| 5.439 | −0.461 | 7.918 | 11.835 |
| 1.878 | −0.922 | 11.835 | 1.671 |

reconstruction error: 1.779

reconstruction d = 2

| | | | |
|---|---|---|---|
| 11.776 | 9.417 | 4.845 | 2.605 |
| 9.417 | 9.432 | −0.249 | −1.181 |
| 4.845 | −0.249 | 10.934 | 8.148 |
| 2.605 | −1.181 | 8.148 | 6.178 |

reconstruction error: 5.442

| | | | |
|---|---|---|---|
| 14 | 7 | 5 | 1 |
| 7 | 12 | 0 | 0 |
| 5 | 0 | 8 | 12 |
| 1 | 0 | 12 | 2 |

# Latent Semantic Analysis

What are the LSA Limitation?

# Latent Semantic Analysis

- Computing the singular value decomposition is expensive; as a rule of thumb, the running time is in O(max(w, d)*min(w, d)^2)
- The method is not incremental – the decomposition needs to be recomputed from scratch whenever new data arrives.

# Pointwise mutual information

- Raw counts give too much weight to function words such as the, she, has, and too little weight to cheese, bread, sheep.
- We measure the associative strength between a target word w and a context c using pointwise mutual information:

$$\text{PMI}(w, c) = \log \frac{P(w, c)}{P(w)P(c)}$$

# Pointwise mutual information

- Raw counts give too much weight to function words such as the, she, has, and too little weight to cheese, bread, sheep.
- We measure the associative strength between a target word w and a context c using pointwise mutual information:

$$\text{PMI}(w, c) = \log \frac{P(w, c)}{P(w)P(c)}$$

# Pointwise mutual information

| | cheese | bread | goat | sheep |
|---|---|---|---|---|
| cheese | 14 | 7 | 5 | 1 |
| bread | 7 | 12 | 0 | 0 |
| goat | 5 | 0 | 8 | 12 |
| sheep | 1 | 0 | 12 | 2 |

# Pointwise mutual information

$$P(cheese) = \frac{27}{86}$$

| | cheese | bread | goat | sheep | |
|---|---|---|---|---|---|
| cheese | 14 | 7 | 5 | 1 | 27 |
| bread | 7 | 12 | 0 | 0 | |
| goat | 5 | 0 | 8 | 12 | |
| sheep | 1 | 0 | 12 | 2 | |
| | | | | | Σ 86 |

# Pointwise mutual information

$$P(bread) = \frac{19}{86}$$

| | cheese | bread | goat | sheep |
|---|---|---|---|---|
| cheese | 14 | 7 | 5 | 1 |
| bread | 7 | 12 | 0 | 0 |
| goat | 5 | 0 | 8 | 12 |
| sheep | 1 | 0 | 12 | 2 |

19 Σ 86

# Pointwise mutual information

$$P(cheese, bread) = \frac{14}{86}$$

| | cheese | bread | goat | sheep |
|---|---|---|---|---|
| cheese | 14 | 7 | 5 | 1 |
| bread | 7 | 12 | 0 | 0 |
| goat | 5 | 0 | 8 | 12 |
| sheep | 1 | 0 | 12 | 2 |

Σ 86

# Pointwise mutual information

| | cheese | bread | goat | sheep | |
|---|---|---|---|---|---|
| cheese | $\frac{14/86}{27/86 \cdot 27/86}$ | $\frac{7/86}{27/86 \cdot 19/86}$ | $\frac{5/86}{27/86 \cdot 25/86}$ | $\frac{1/86}{27/86 \cdot 15/86}$ | 27 |
| bread | $\frac{7/86}{19/86 \cdot 27/86}$ | $\frac{12/86}{19/86 \cdot 19/86}$ | $\frac{0/86}{19/86 \cdot 25/86}$ | $\frac{0/86}{19/86 \cdot 15/86}$ | 19 |
| goat | $\frac{5/86}{25/86 \cdot 27/86}$ | $\frac{0/86}{25/86 \cdot 19/86}$ | $\frac{8/86}{25/86 \cdot 25/86}$ | $\frac{12/86}{25/86 \cdot 15/86}$ | 25 |
| sheep | $\frac{1/86}{15/86 \cdot 27/86}$ | $\frac{0/86}{15/86 \cdot 19/86}$ | $\frac{12/86}{15/86 \cdot 25/86}$ | $\frac{2/86}{15/86 \cdot 15/86}$ | 15 |
| | 27 | 19 | 25 | 15 | Σ 86 |

# Pointwise mutual information

| | cheese | bread | goat | sheep |
|---|---|---|---|---|
| cheese | log 1.65 | log 1.17 | log 0.64 | log 0.21 |
| bread | log 1.17 | log 2.86 | log 0.00 | log 0.00 |
| goat | log 0.64 | log 0.00 | log 1.10 | log 2.75 |
| sheep | log 0.21 | log 0.00 | log 2.75 | log 0.76 |

# Positive pointwise mutual information (PPMI)

- Because the rows of co-occurrence matrices are sparse, many PMI values will be log $0 = -\infty$.
- A common approach is to use positive pointwise mutual information (PPMI), and replace all negative values with zero

$$\text{PPMI}(w, c) = \max\big(\text{PMI}(w, c), 0\big)$$

# Word2vec

Based on the assumption that the meaning of a word can be learned from its context, CBOW optimizes the embeddings so that they can predict a target word given its context words. Skip-gram, on the contrary, learns the **embeddings** that can predict the context words given a target word.

# Continuous Bag-of-Words

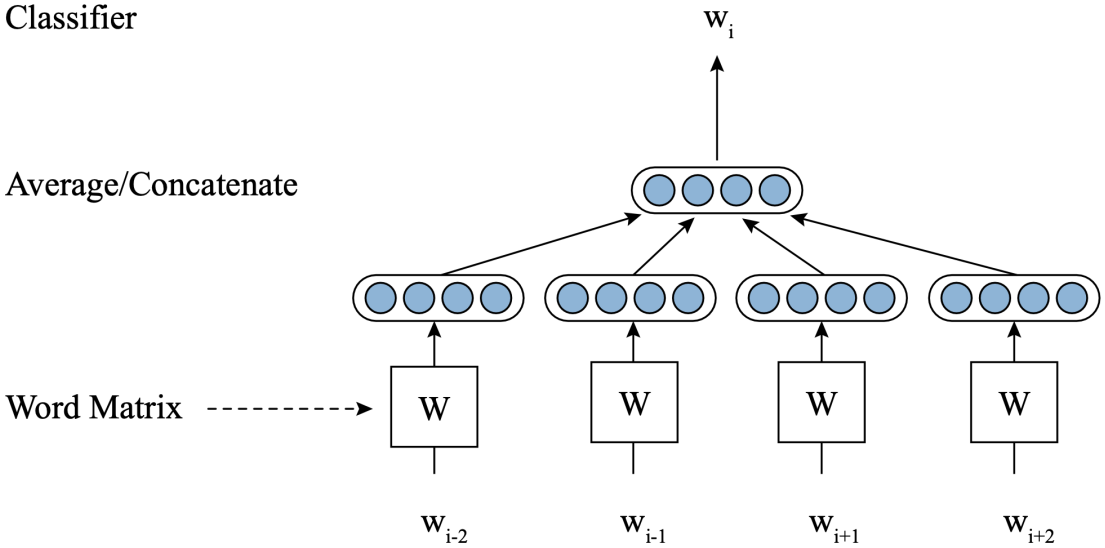
CBOW predicts the center word given a window of context. Formally, CBOW predicts $w_i$ according to its contexts as

$$P(w_i|w_{j(|j-i|\leq l, j\neq i)}) = \text{Softmax}\left(\mathbf{M}\left(\sum_{|j-i|\leq l, j\neq i} \mathbf{w}_j\right)\right)$$

Where $P\left(w_i\middle|w_{j(|j-i|\leq l, j\neq i)}\right) = Softmax\left(M\left(\sum_{|i-j|\leq l, j\neq i} w_i\right)\right)$ is the probability of word wi given its contexts, $l$ is the size of training contexts, M is the weight matrix in $R^{|V|\times M}$, $V$ indicates the vocabulary, and m is the dimension of the word vector. The CBOW model is optimized by minimizing the sum of negative log probabilities:

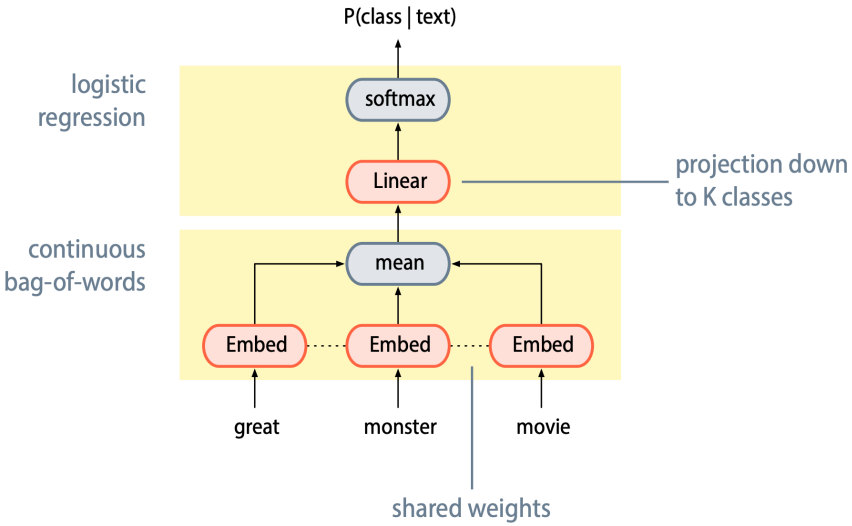
$$\mathscr{L} = -\sum_i \log P(w_i|w_{j(|j-i|\leq l, j\neq i)}).$$

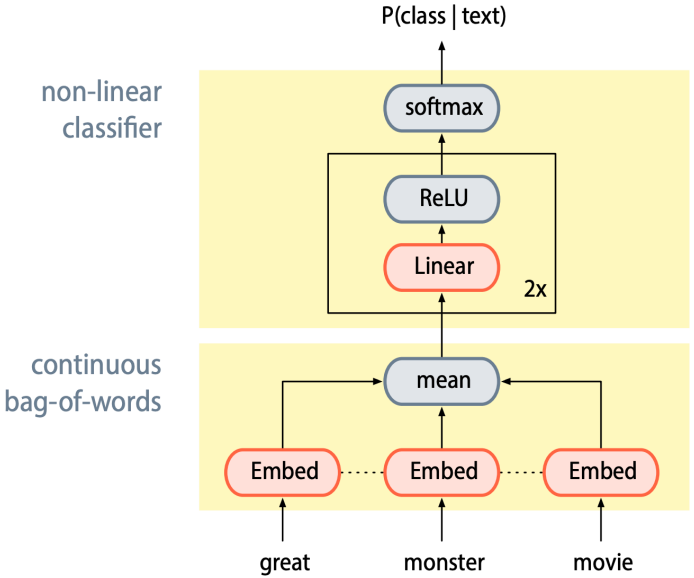
# Continuous Bag-of-Words




The architecture of CBOW model

# Continuous Bag-of-Words



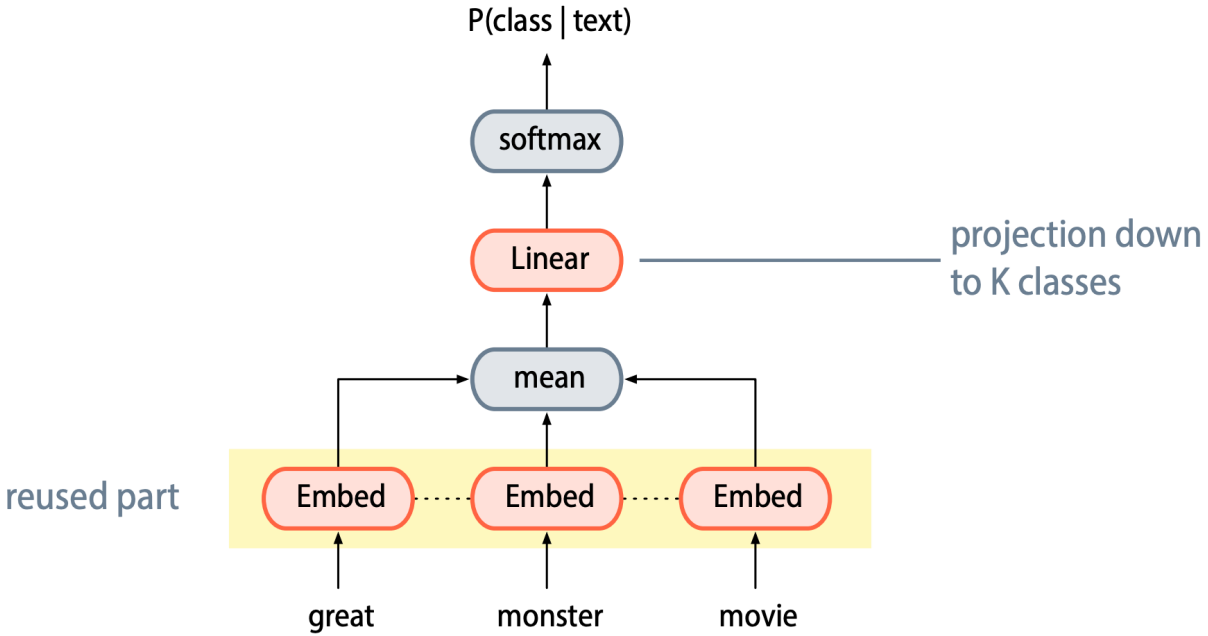



# Re-using pre-trained word embeddings

Pre-train embeddings on task A and use them to initialize the embedding layers of the network for task B. Then:

- Alternative 1: Train as usual, effectively fine-tuning the pretrained embeddings to the task at hand.
- Alternative 2: Freeze the weights of the embedding layers, to prevent the pre-trained embeddings from being modified.

# Continuous Bag-of-Words



# Skip-Gram

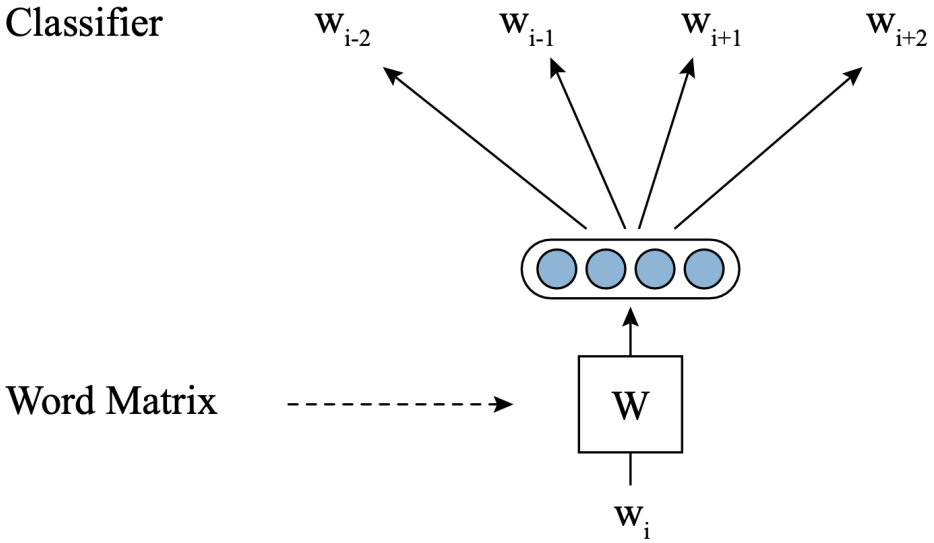
On the contrary to CBOW, Skip-gram predicts the context given the center word. Formally, given a word $w_i$ , Skip-gram predicts its context as

$$P(w_j|w_i) = \text{softmax}(\mathbf{M}\mathbf{w}_i)\big(|j-i| \le l, j \ne i\big)$$

Where $P\big(w_i|w_j\big)$ is the probability of context word $w_j$ given $w_i$, and $M$ is the weight matrix. The loss function is defined similar to CBOW as

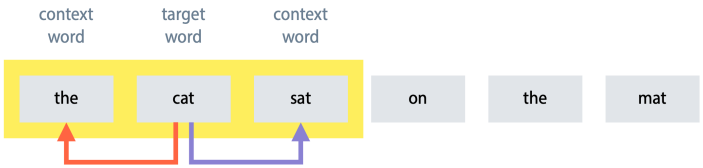
$$\mathscr{L} = -\sum_{i} \sum_{j(|j-i| \le l, j \ne i)} P(w_j|w_i).$$

# Skip-Gram



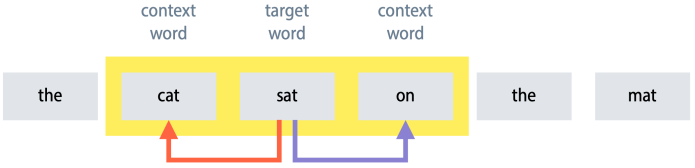

The architecture of skip-gram model

# Skip-Gram




$P(\text{the} \mid \text{cat}) \propto \boldsymbol{v}'^{\top}_{\text{the}} \boldsymbol{v}_{\text{cat}}$ $P(\text{sat} \mid \text{cat}) \propto \boldsymbol{v}'^{\top}_{\text{sat}} \boldsymbol{v}_{\text{cat}}$

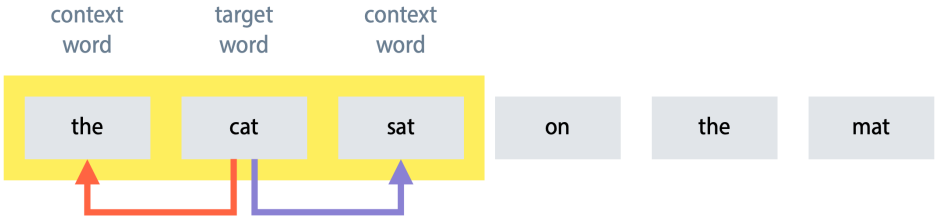
| | | | | | |
|---|---|---|---|---|---|
| target word vectors | $\boldsymbol{v}_{\text{cat}}$ | $\boldsymbol{v}_{\text{mat}}$ | $\boldsymbol{v}_{\text{on}}$ | $\boldsymbol{v}_{\text{sat}}$ | $\boldsymbol{v}_{\text{the}}$ |
| context word vectors | $\boldsymbol{v}'_{\text{cat}}$ | $\boldsymbol{v}'_{\text{mat}}$ | $\boldsymbol{v}'_{\text{on}}$ | $\boldsymbol{v}'_{\text{sat}}$ | $\boldsymbol{v}'_{\text{the}}$ |




$P(\text{cat} \mid \text{sat}) \propto \boldsymbol{v}'^{\top}_{\text{cat}} \boldsymbol{v}_{\text{sat}}$ $P(\text{on} \mid \text{sat}) \propto \boldsymbol{v}'^{\top}_{\text{on}} \boldsymbol{v}_{\text{sat}}$

| | | | | | |
|---|---|---|---|---|---|
| target word vectors | $\boldsymbol{v}_{\text{cat}}$ | $\boldsymbol{v}_{\text{mat}}$ | $\boldsymbol{v}_{\text{on}}$ | $\boldsymbol{v}_{\text{sat}}$ | $\boldsymbol{v}_{\text{the}}$ |
| context word vectors | $\boldsymbol{v}'_{\text{cat}}$ | $\boldsymbol{v}'_{\text{mat}}$ | $\boldsymbol{v}'_{\text{on}}$ | $\boldsymbol{v}'_{\text{sat}}$ | $\boldsymbol{v}'_{\text{the}}$ |

# Skip-Gram




$P(\text{the} \mid \text{cat}) \propto \boldsymbol{v}'^{\top}_{\text{the}} \boldsymbol{v}_{\text{cat}}$ $P(\text{sat} \mid \text{cat}) \propto \boldsymbol{v}'^{\top}_{\text{sat}} \boldsymbol{v}_{\text{cat}}$

| | | | | | |
|---|---|---|---|---|---|
| target word vectors | $\boldsymbol{v}_{\text{cat}}$ | $\boldsymbol{v}_{\text{mat}}$ | $\boldsymbol{v}_{\text{on}}$ | $\boldsymbol{v}_{\text{sat}}$ | $\boldsymbol{v}_{\text{the}}$ |
| context word vectors | $\boldsymbol{v}'_{\text{cat}}$ | $\boldsymbol{v}'_{\text{mat}}$ | $\boldsymbol{v}'_{\text{on}}$ | $\boldsymbol{v}'_{\text{sat}}$ | $\boldsymbol{v}'_{\text{the}}$ |

# Skip-gram with negative sampling

Maximize the probability of observed word–context pairs, while minimizing the probability of randomly drawn samples

$$J(\boldsymbol{\theta}) = -\frac{1}{N}\sum_{i=1}^{N}\sum_{\substack{-m\le j\le m\\ j\neq 0}} \log\sigma\left(\boldsymbol{v}'^{\top}_{w_{i+j}}\boldsymbol{v}_{w_i}\right) + \sum_{c\sim D}\log\sigma\left(-\boldsymbol{v}'^{\top}_{c}\boldsymbol{v}_{w_i}\right)$$

- $\frac{1}{N}$: length of the text
- $\log\sigma\left(\boldsymbol{v}'^{\top}_{w_{i+j}}\boldsymbol{v}_{w_i}\right)$: logistic function
- $\sum_{c\sim D}\log\sigma\left(-\boldsymbol{v}'^{\top}_{c}\boldsymbol{v}_{w_i}\right)$: negative samples

# Skip-gram with negative sampling