Statistical Natural Language Processing [COMP0087]

Word embeddings

Vasileios Lampos

Computer Science, UCL

lampos.net

About this lecture

- In this lecture:
  - Sparse and dense vector space representations for words
  - `word2vec` with skip-gram (*and negative sampling*)
- **Reading / Lecture based on:** Chapter 6 of "*Speech and Language Processing*" (SLP) by Jurafsky and Martin (2023) — web.stanford.edu/~jurafsky/slp3/
- Clipped slides: lampos.net/teaching
- Additional material
  - `word2vec` — see arxiv.org/abs/1301.3781 and proceedings.neurips.cc/paper/2013/file/9aa42b31882ec039965f3c4923ce901b-Paper.pdf
  - probabilistic topic models — see youtube.com/watch?v=yK7nN3FcgUs

Word embeddings by counting

- Specify word co-occurrence **context window** in a corpus
- $+/-4$ words around the target word is a common setting

"Another Brick in the Wall" part 2 is a Pink Floyd song from "The Wall" album that was released as a single in 1979 and while it was banned by at least one authoritarian regime, it managed to sell more than 4 million copies worldwide.

Word embeddings by counting

- Specify word co-occurrence **context window** in a corpus
- $+/-4$ words around the target word is a common setting
- short context window $\rightarrow$ syntax / grammar aware representation

"Another Brick in the Wall" part 2 is a Pink Floyd song from "The Wall" album that was released as a single in 1979 and while it was banned by at least one authoritarian regime, it managed to sell more than 4 million copies worldwide.

Word embeddings by counting

- Specify word co-occurrence **context window** in a corpus
- $+/-4$ words around the target word is a common setting
- short context window → syntax / grammar aware representation
- long context window → more abstraction / meaning / semantics

> *"Another Brick in the Wall" part 2 is a Pink Floyd song from "The Wall" album that was released as a single in 1979 and while it was banned by at least one authoritarian regime, it managed to sell more than 4 million copies worldwide.*

Word co-occurrence matrix $\quad \mathbf{C} \in \mathbb{N}^{|\mathscr{V}| \times |\mathscr{V}|}$

$$
|\mathscr{V}| \quad
\begin{array}{c c}
 & \begin{array}{cccccc} \text{a} & \text{an} & & c_j & & \text{zoo} \end{array} \\
\begin{array}{c} \text{a} \\ \text{an} \\ \\ \text{zoo} \end{array} &
\begin{bmatrix}
20 & 50 & \cdots & 0 & \cdots & 200 \\
50 & 10 & \cdots & 0 & \cdots & 100 \\
\vdots & \vdots & \ddots & \ddots & \ddots & \vdots \\
200 & 100 & \cdots & 0 & \cdots & 2
\end{bmatrix}
\end{array}
$$

Word embeddings by counting

Word co-occurrence matrix $\quad \mathbf{C} \in \mathbb{N}^{|\mathcal{V}| \times |\mathcal{V}|}$

$$
|\mathcal{V}| \quad
\begin{array}{c c}
 & \begin{array}{cccccc} \text{a} & \text{an} & & c_j & & \text{zoo} \end{array} \\
\begin{array}{c} \text{a} \\ \text{an} \\ \\ \text{zoo} \end{array} &
\begin{bmatrix}
20 & 50 & \cdots & 0 & \cdots & 200 \\
50 & 10 & \cdots & 0 & \cdots & 100 \\
\vdots & \vdots & \ddots & \ddots & \ddots & \vdots \\
200 & 100 & \cdots & 0 & \cdots & 2
\end{bmatrix}
\end{array}
$$

- given a corpus, count the amount of times words co-occur within the specified context windows

Word embeddings by counting

Word co-occurrence matrix $\quad \mathbf{C} \in \mathbb{N}^{|\mathscr{V}| \times |\mathscr{V}|}$

$$
|\mathscr{V}| \quad
\begin{array}{c c}
 & \begin{array}{cccccc} \text{a} & \text{an} & & c_j & & \text{zoo} \end{array} \\
\begin{array}{c} \text{a} \\ \text{an} \\ \\ \text{zoo} \end{array} &
\begin{bmatrix}
20 & 50 & \cdots & 0 & \cdots & 200 \\
50 & 10 & \cdots & 0 & \cdots & 100 \\
\vdots & \vdots & \ddots & \ddots & \ddots & \vdots \\
200 & 100 & \cdots & 0 & \cdots & 2
\end{bmatrix}
\end{array}
$$

word embeddings

- given a corpus, count the amount of times words co-occur within the specified context windows
- generates primitive word embeddings

Word embeddings by counting

Word co-occurrence matrix $\mathbf{C} \in \mathbb{N}^{|\mathscr{V}| \times |\mathscr{V}|}$

$$
|\mathscr{V}| \quad
\begin{array}{c c}
 & \begin{array}{cccccc} \text{a} & \text{an} & & c_j & & \text{zoo} \end{array} \\
\begin{array}{c} \text{a} \\ \text{an} \\ \\ \text{zoo} \end{array} &
\begin{bmatrix}
20 & 50 & \cdots & 0 & \cdots & 200 \\
50 & 10 & \cdots & 0 & \cdots & 100 \\
\vdots & \vdots & \ddots & \ddots & \ddots & \vdots \\
200 & 100 & \cdots & 0 & \cdots & 2
\end{bmatrix}
\end{array}
$$

word embeddings

- given a corpus, count the amount of times words co-occur within the specified context windows
- generates primitive word embeddings
- sparse representation, sparser for shorter context windows
- high dimensional representation; depends on vocabulary size, $|\mathscr{V}|$

Word embeddings by counting — Pointwise Mutual Information (PMI)

Word co-occurrence matrix $\mathbf{C} \in \mathbb{N}^{|\mathscr{V}| \times |\mathscr{V}|}$

Word embeddings by counting — Pointwise Mutual Information (PMI)

Word co-occurrence matrix $\mathbf{C} \in \mathbb{N}^{|\mathscr{V}| \times |\mathscr{V}|}$

Word context matrix $\mathbf{Q} \in \mathbb{N}^{|\mathscr{V}| \times d}, \; d < |\mathscr{V}|$

Word embeddings by counting — Pointwise Mutual Information (PMI)

Word co-occurrence matrix $\mathbf{C} \in \mathbb{N}^{|\mathscr{V}| \times |\mathscr{V}|}$

Word context matrix $\mathbf{Q} \in \mathbb{N}^{|\mathscr{V}| \times d}, \; d < |\mathscr{V}|$

- Pointwise Mutual Information (**PMI**)
  *How often 2 events (**in NLP: words!**) co-occur compared to our expectation under the assumption that these events were independent*

Word embeddings by counting — Pointwise Mutual Information (PMI)

Word co-occurrence matrix $\mathbf{C} \in \mathbb{N}^{|\mathcal{V}| \times |\mathcal{V}|}$

Word context matrix $\mathbf{Q} \in \mathbb{N}^{|\mathcal{V}| \times d}, \ d < |\mathcal{V}|$

- Pointwise Mutual Information (**PMI**)
  *How often 2 events (**in NLP: words!**) co-occur compared to our expectation under the assumption that these events were independent*
- For a target word w_i and a context word c_j

$$\mathrm{PMI}(w_i, c_j) = \log \frac{p(w_i, c_j)}{p(w_i) \cdot p(c_j)}$$

if $\log_2$, then the units are bits!

Word embeddings by counting — PPMI

$$\text{PMI}(w_i, c_j) = \log \frac{p(w_i, c_j)}{p(w_i) \cdot p(c_j)}$$

Word embeddings by counting — PPMI

$$\mathrm{PMI}(w_i, c_j) = \log \frac{p(w_i, c_j)}{p(w_i) \cdot p(c_j)}$$

- PMI identifies strongly associated words even when less frequent
- PMI ranges in $(-\infty, +\infty)$
- $\log(\,\cdot\,)$ shrinks the range

Word embeddings by counting — PPMI

$$\text{PMI}(w_i, c_j) = \log \frac{p(w_i, c_j)}{p(w_i) \cdot p(c_j)}$$

- PMI identifies strongly associated words even when less frequent
- PMI ranges in $(-\infty, +\infty)$
- $\log(\cdot)$ shrinks the range
- **Negative PMI values** are harder to interpret and evaluate
  — "relatedness" is more comprehensive / objective
- Force positivity — Positive PMI (PPMI)

$$\text{PPMI}(w_i, c_j) = \max\left(\text{PMI}(w_i, c_j),\ 0\right)$$

Word embeddings by counting — PPMI

Word context matrix $\mathbf{Q} \in \mathbb{N}^{|\mathscr{V}| \times d},\ d < |\mathscr{V}|$

$$\mathrm{PMI}(w_i, c_j) = \log \frac{p(w_i, c_j)}{p(w_i) \cdot p(c_j)} \qquad \mathrm{PPMI}(w_i, c_j) = \max\Big(\mathrm{PMI}(w_i, c_j),\ 0\Big)$$

Word context matrix $\mathbf{Q} \in \mathbb{N}^{|\mathscr{V}| \times d},\ d < |\mathscr{V}|$

$$\mathrm{PMI}(w_i, c_j) = \log \frac{p(w_i, c_j)}{p(w_i) \cdot p(c_j)} \qquad \mathrm{PPMI}(w_i, c_j) = \max\left(\mathrm{PMI}(w_i, c_j),\ 0\right)$$

$$p(w_i, c_j) = \frac{q_{ij}}{\sum_{i=1}^{|\mathscr{V}|} \sum_{j=1}^{d} q_{ij}}$$

number of times w_i co-occurs with c_j divided by the total word count in $\mathbf{Q}$

Word context matrix $\qquad \mathbf{Q} \in \mathbb{N}^{|\mathscr{V}| \times d},\ d < |\mathscr{V}|$

$$\mathrm{PMI}(w_i, c_j) = \log \frac{p(w_i, c_j)}{p(w_i) \cdot p(c_j)} \qquad \mathrm{PPMI}(w_i, c_j) = \max\Big(\mathrm{PMI}(w_i, c_j),\ 0\Big)$$

$$p(w_i, c_j) = \frac{q_{ij}}{\sum_{i=1}^{|\mathscr{V}|} \sum_{j=1}^{d} q_{ij}}$$

number of times w_i co-occurs with c_j divided by the total word count in $\mathbf{Q}$

sum of i-th row of $\mathbf{Q}$

$$p(w_i) = \frac{\sum_{j=1}^{d} q_{ij}}{\sum_{i=1}^{|\mathscr{V}|} \sum_{j=1}^{d} q_{ij}}$$

Word embeddings by counting — PPMI

Word context matrix $\mathbf{Q} \in \mathbb{N}^{|\mathscr{V}| \times d},\ d < |\mathscr{V}|$

$$\mathrm{PMI}(w_i, c_j) = \log \frac{p(w_i, c_j)}{p(w_i) \cdot p(c_j)}$$

$$\mathrm{PPMI}(w_i, c_j) = \max\left(\mathrm{PMI}(w_i, c_j),\ 0\right)$$

$$p(w_i, c_j) = \frac{q_{ij}}{\sum_{i=1}^{|\mathscr{V}|} \sum_{j=1}^{d} q_{ij}}$$

number of times w_i co-occurs with c_j
divided by the total word count in $\mathbf{Q}$

sum of i-th row of $\mathbf{Q}$

$$p(w_i) = \frac{\sum_{j=1}^{d} q_{ij}}{\sum_{i=1}^{|\mathscr{V}|} \sum_{j=1}^{d} q_{ij}}$$

sum of j-th column of $\mathbf{Q}$

$$p(c_j) = \frac{\sum_{i=1}^{|\mathscr{V}|} q_{ij}}{\sum_{i=1}^{|\mathscr{V}|} \sum_{j=1}^{d} q_{ij}}$$

Word embeddings by counting — PPMI

Word context matrix $\mathbf{Q} \in \mathbb{N}^{|\mathcal{V}| \times d},\ d < |\mathcal{V}|$

$$\mathrm{PMI}(w_i, c_j) = \log \frac{p(w_i, c_j)}{p(w_i) \cdot p(c_j)}$$

replace with

$$\mathrm{PPMI}(w_i, c_j) = \max\left(\mathrm{PMI}(w_i, c_j),\ 0\right)$$

$$p(w_i, c_j) = \frac{q_{ij}}{\sum_{i=1}^{|\mathcal{V}|} \sum_{j=1}^{d} q_{ij}}$$

number of times w_i co-occurs with c_j divided by the total word count in $\mathbf{Q}$

sum of i-th row of $\mathbf{Q}$

$$p(w_i) = \frac{\sum_{j=1}^{d} q_{ij}}{\sum_{i=1}^{|\mathcal{V}|} \sum_{j=1}^{d} q_{ij}}$$

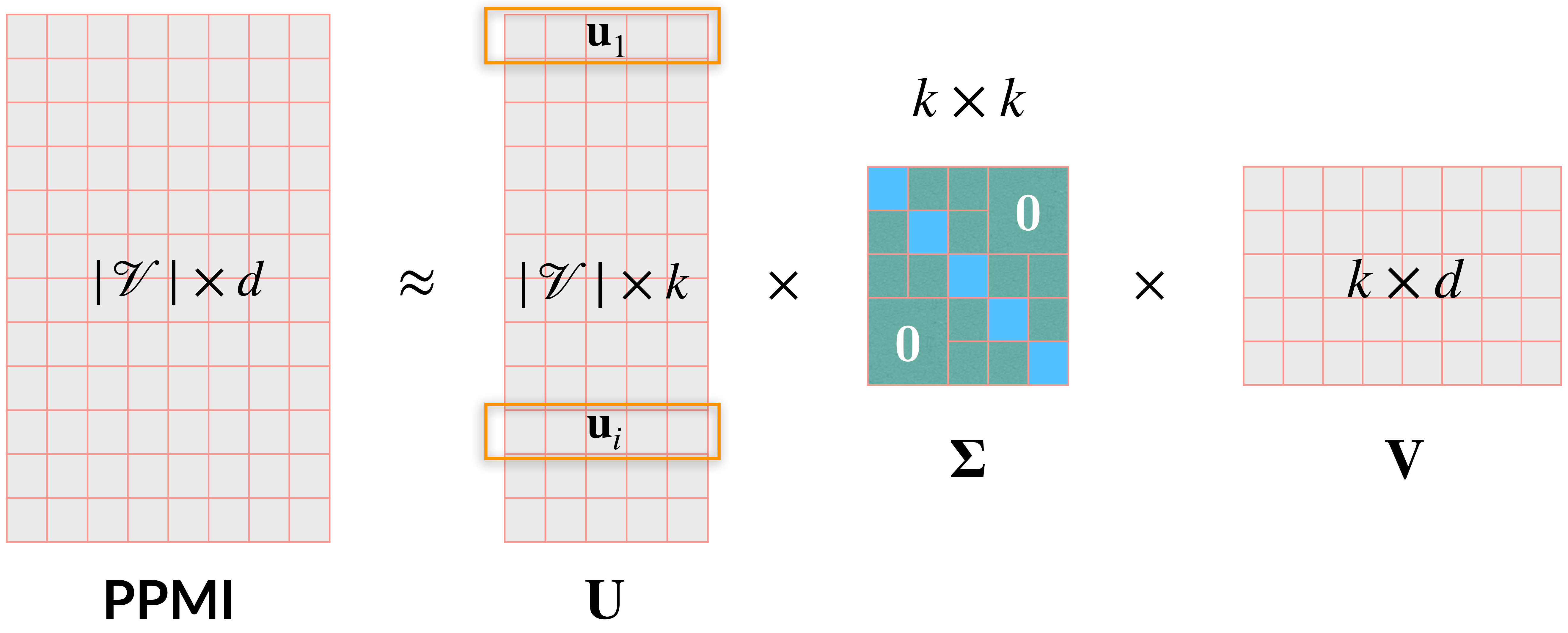
sum of j-th column of $\mathbf{Q}$

$$p(c_j) = \frac{\sum_{i=1}^{|\mathcal{V}|} q_{ij}}{\sum_{i=1}^{|\mathcal{V}|} \sum_{j=1}^{d} q_{ij}}$$

Singular
Value
Decomposition

(truncated)

SVD

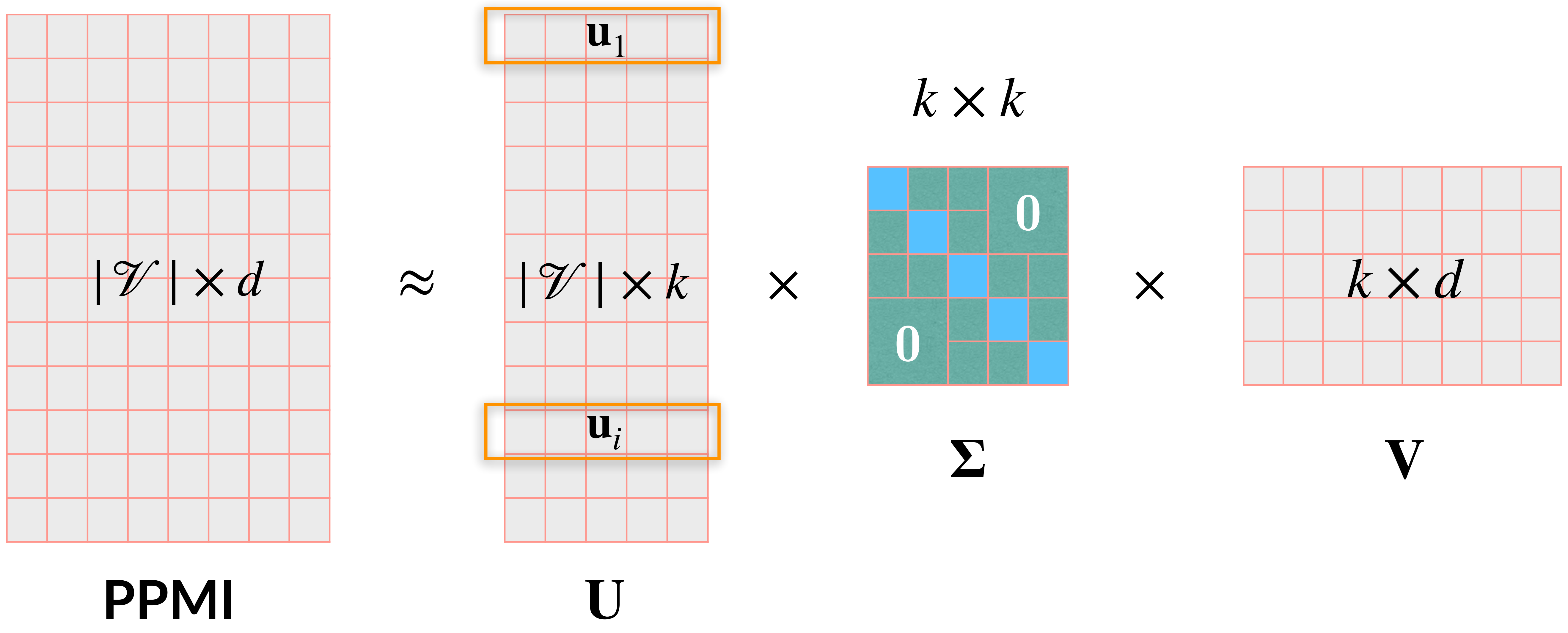


Word embeddings by matrix factorisation — SVD to PPMI

Singular **V**alue **D**ecomposition

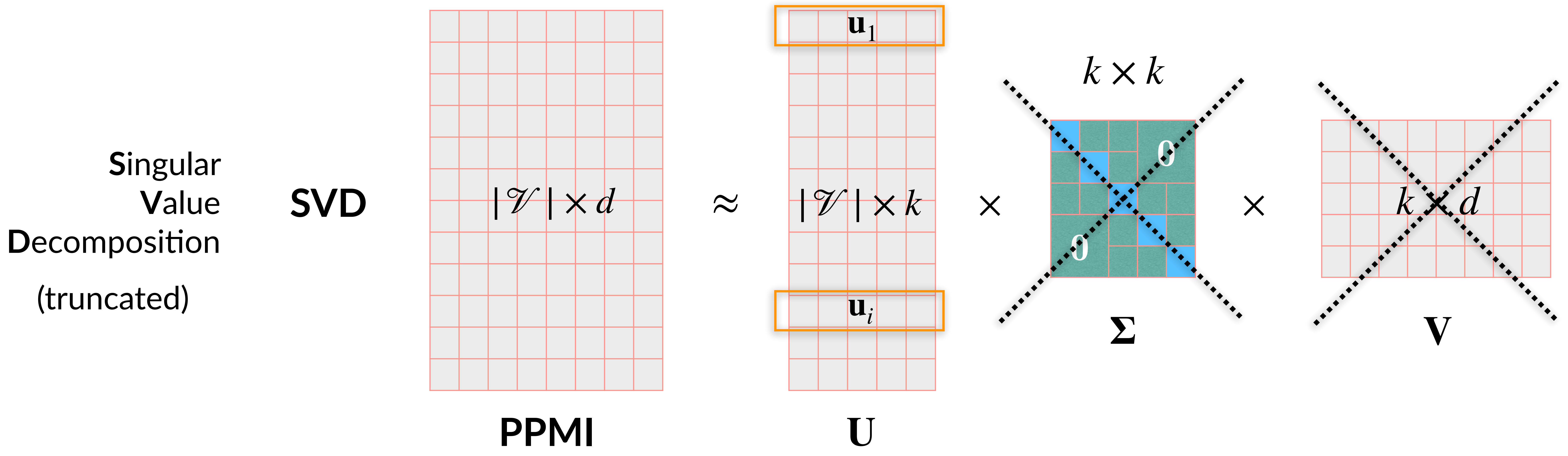
(truncated)

SVD



- $\mathbf{u}_i$: k-dimensional vector that represents word i in our vocabulary
  — **dense word embedding**
  — commonly, $k = 128$ to 1024, i.e. $\mathbf{u}_i$ is short and dense
  — matrices Σ and $\mathbf{V}$ are (*or could be*) thrown away

Word embeddings by matrix factorisation — SVD to PPMI



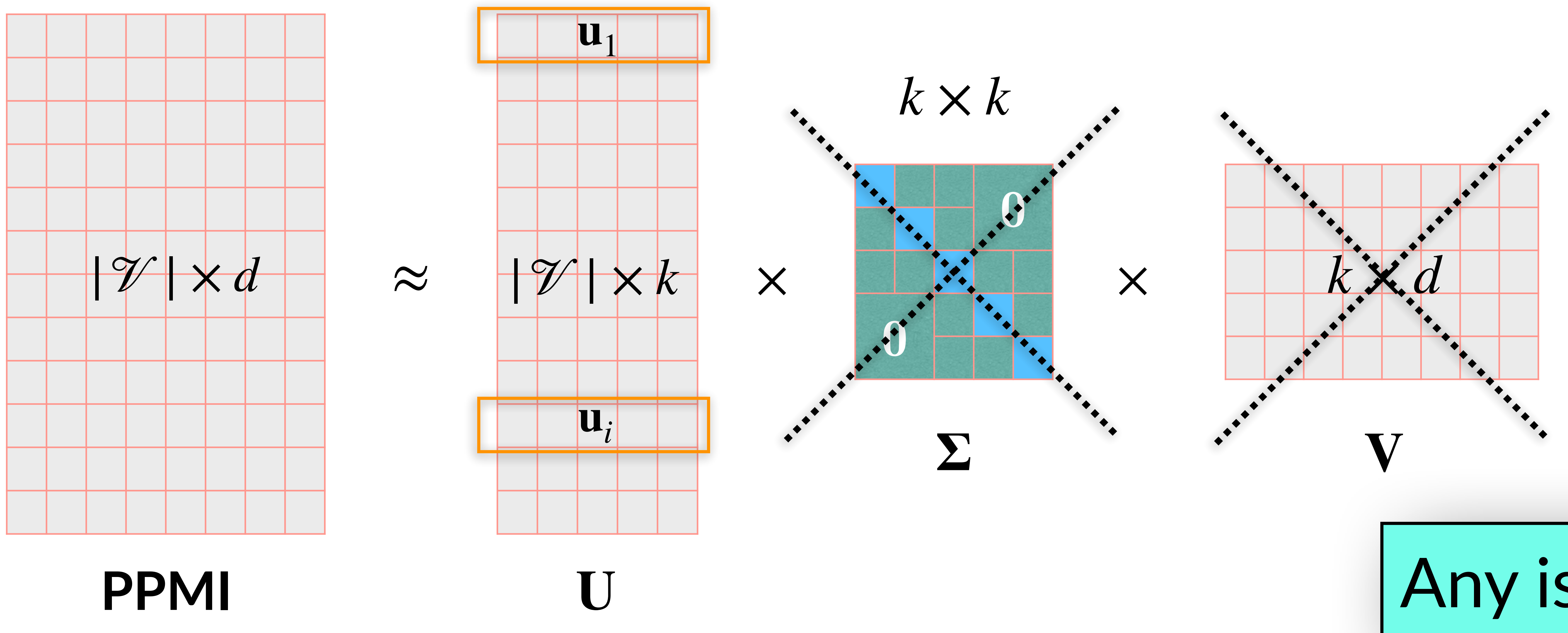
- $\mathbf{u}_i$: k-dimensional vector that represents word i in our vocabulary
  — **dense word embedding**
  — commonly, $k = 128$ to 1024, i.e. $\mathbf{u}_i$ is short and dense
  — matrices Σ and $\mathbf{V}$ are (*or could be*) thrown away

Word embeddings by matrix factorisation — SVD to PPMI

Singular
Value
Decomposition

(truncated)

SVD



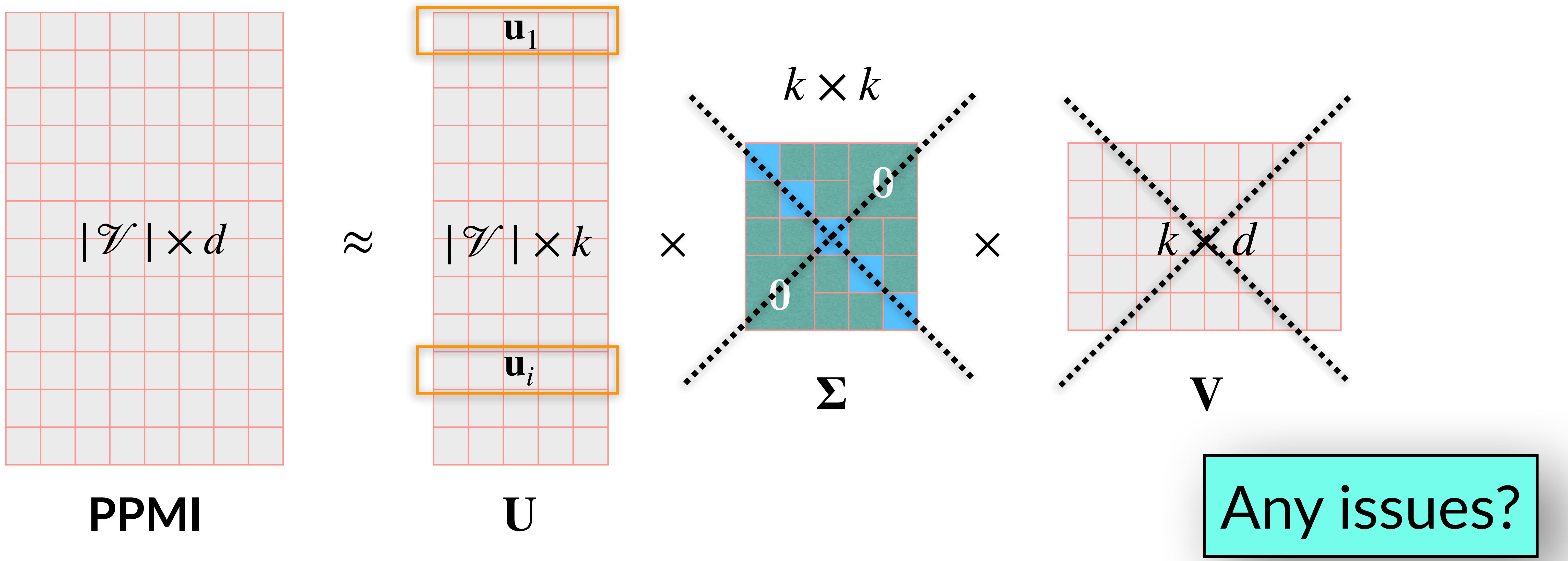
Any issues?

- $\mathbf{u}_i$: k-dimensional vector that represents word i in our vocabulary
  — **dense word embedding**
  — commonly, $k = 128$ to 1024, i.e. $\mathbf{u}_i$ is short and dense
  — matrices Σ and $\mathbf{V}$ are (*or could be*) thrown away

Singular
Value
Decomposition

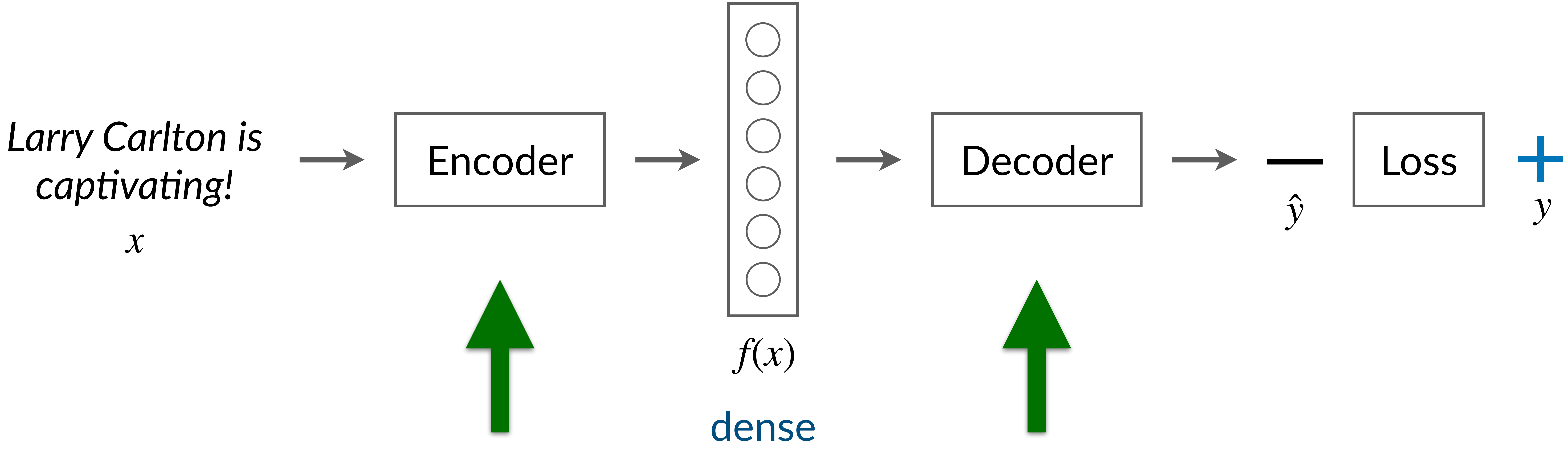
(truncated)

SVD

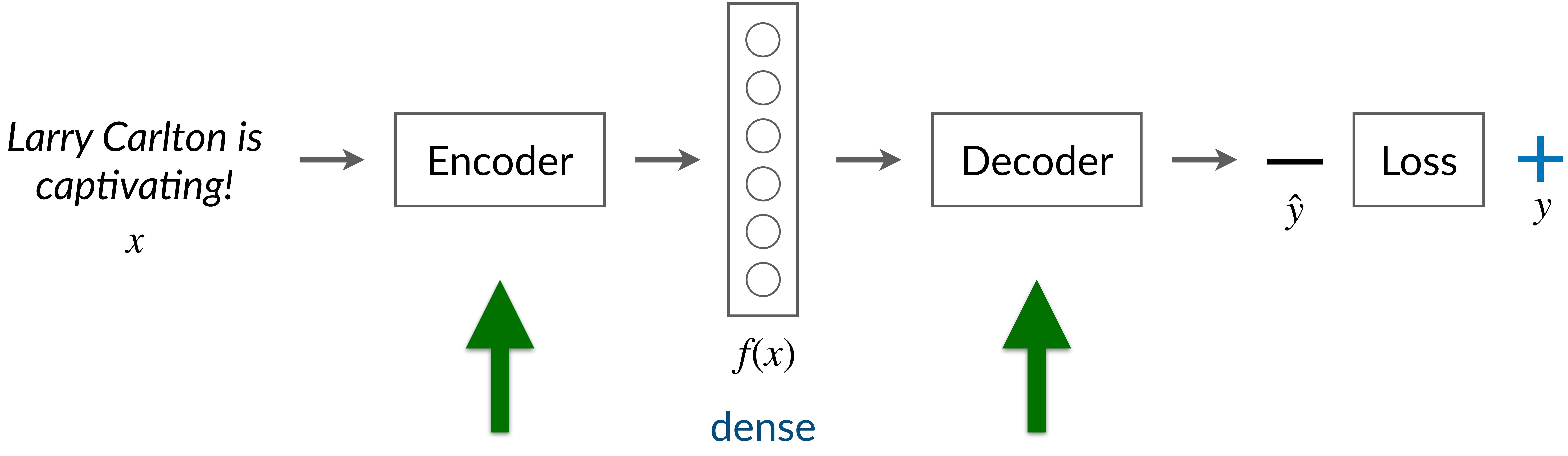


- $\mathbf{u}_i$: k-dimensional vector that represents word i in our vocabulary
  — **dense word embedding**
  — commonly, $k = 128$ to 1024, i.e. $\mathbf{u}_i$ is short and dense
  — matrices Σ and $\mathbf{V}$ are (*or could be*) thrown away
- **Downsides:** SVD has a significant computational cost, $\mathcal{O}\big(|\mathcal{V}| \cdot d \cdot k^2\big)$
  No intuition — what do the SVD embeddings represent?

The NLP view (*for this lecture*)



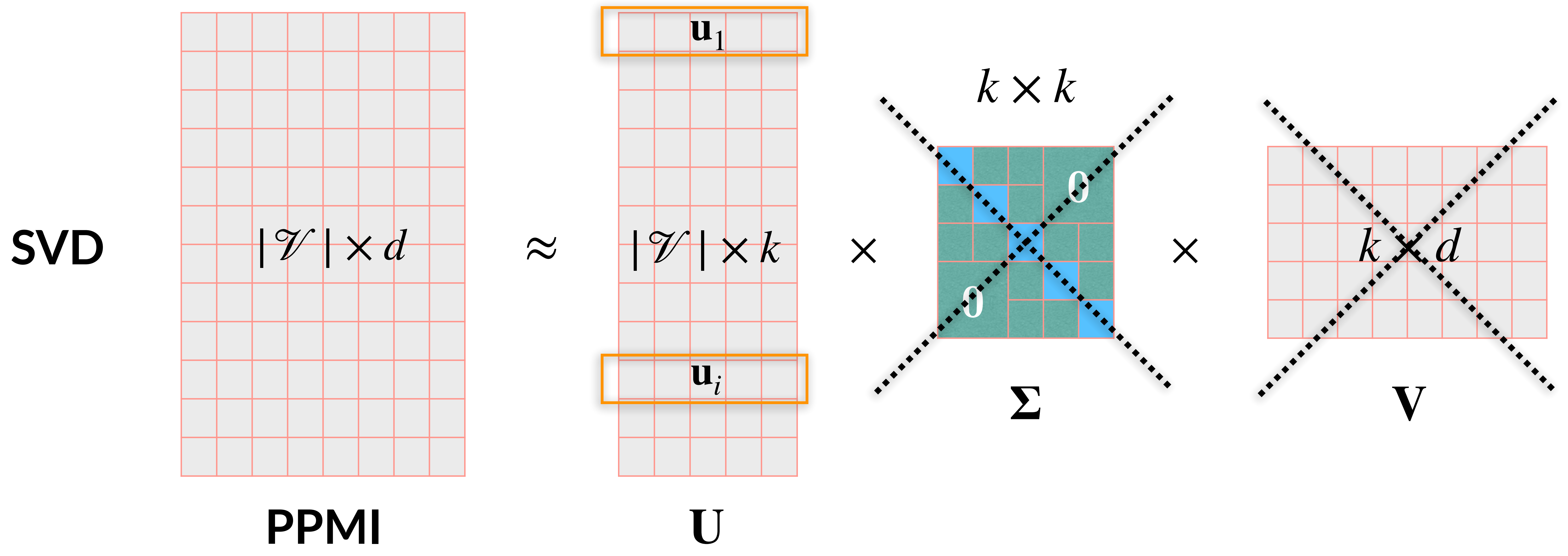
The NLP view (*for this lecture*)



`word2vec`

(and many other methods)

Word embeddings by matrix factorisation — SVD to PPMI



- Interesting to know: *A variant of* `word2vec` *(skip-gram with negative sampling that will see next) is implicitly factorising a word-context matrix, whose cells are the pointwise mutual information (PMI) of the respective word and context pairs, shifted by a global constant*
- More in papers.nips.cc/paper_files/paper/2014/file/feab05aa91085b7a8012516bc3533958-Paper.pdf

Word embeddings by prediction

… said that “Hey Jude” is Beatles’ most famous song, but…

Word embeddings by prediction

… said that "Hey Jude" is **Beatles**' most famous song, but…

centre word

w_t

Word embeddings by prediction

… said that "Hey Jude" is **Beatles**' most famous song, but…

centre word

w_t

context radius

$L = 3$

Word embeddings by prediction

context words $\mathbf{c} = [w_{t-3} \quad w_{t-2} \quad w_{t-1} \quad w_{t+1} \quad w_{t+2} \quad w_{t+3}]$

… said that "Hey Jude" is **Beatles**' most famous song, but…

centre word

w_t

context radius

$L = 3$

Word embeddings by prediction

context words $\mathbf{c} = [w_{t-3} \quad w_{t-2} \quad w_{t-1} \quad w_{t+1} \quad w_{t+2} \quad w_{t+3}]$

… said that "Hey Jude" is **Beatles'** most famous song, but…

centre word

w_t

context radius

$L = 3$

Prediction tasks

$p(\mathbf{c} \mid w_t) = ?$

or

$p(w_t \mid \mathbf{c}) = ?$

Word embeddings by prediction

context words $\mathbf{c} = [w_{t-3} \quad w_{t-2} \quad w_{t-1} \quad w_{t+1} \quad w_{t+2} \quad w_{t+3}]$

… said that "Hey Jude" is **Beatles'** most famous song, but…

centre word

w_t

context radius

$L = 3$

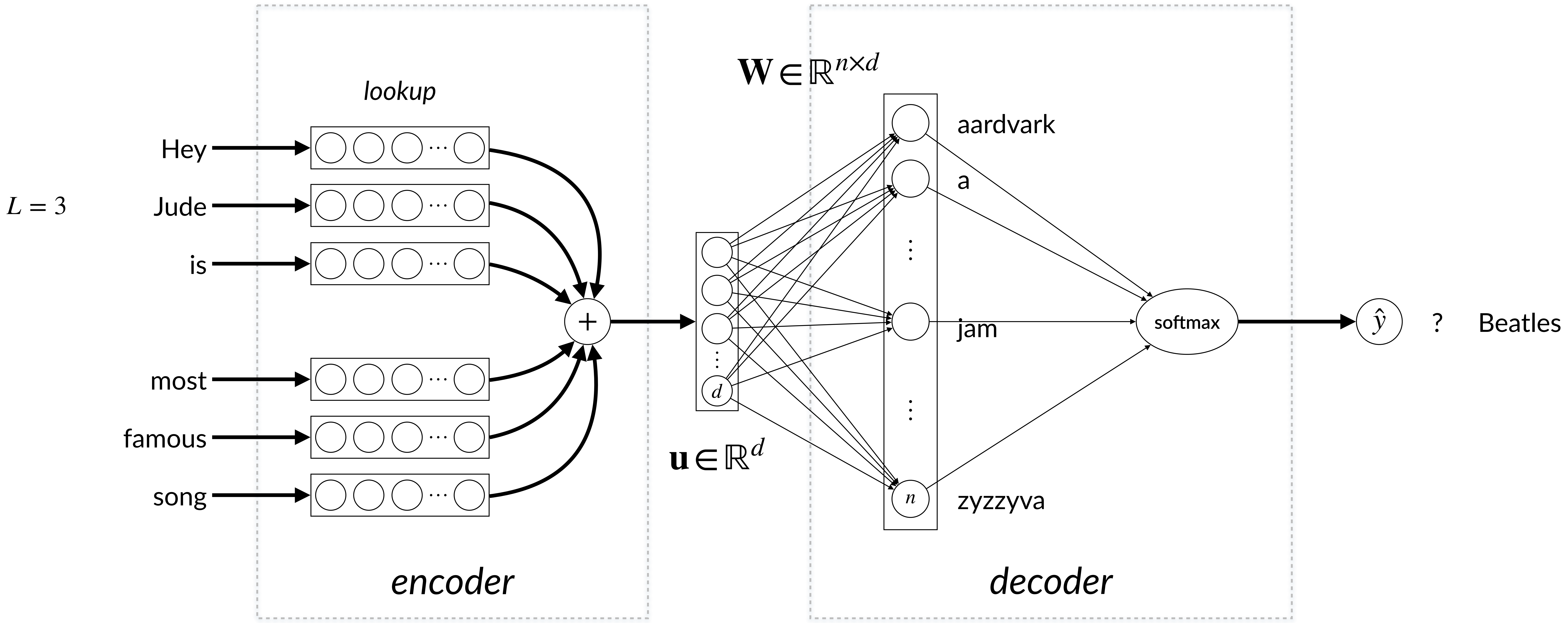
Prediction tasks

$p(\mathbf{c} \mid w_t) = ?$ skip-gram

or

$p(w_t \mid \mathbf{c}) = ?$ Continuous Bag of Words (CBOW)

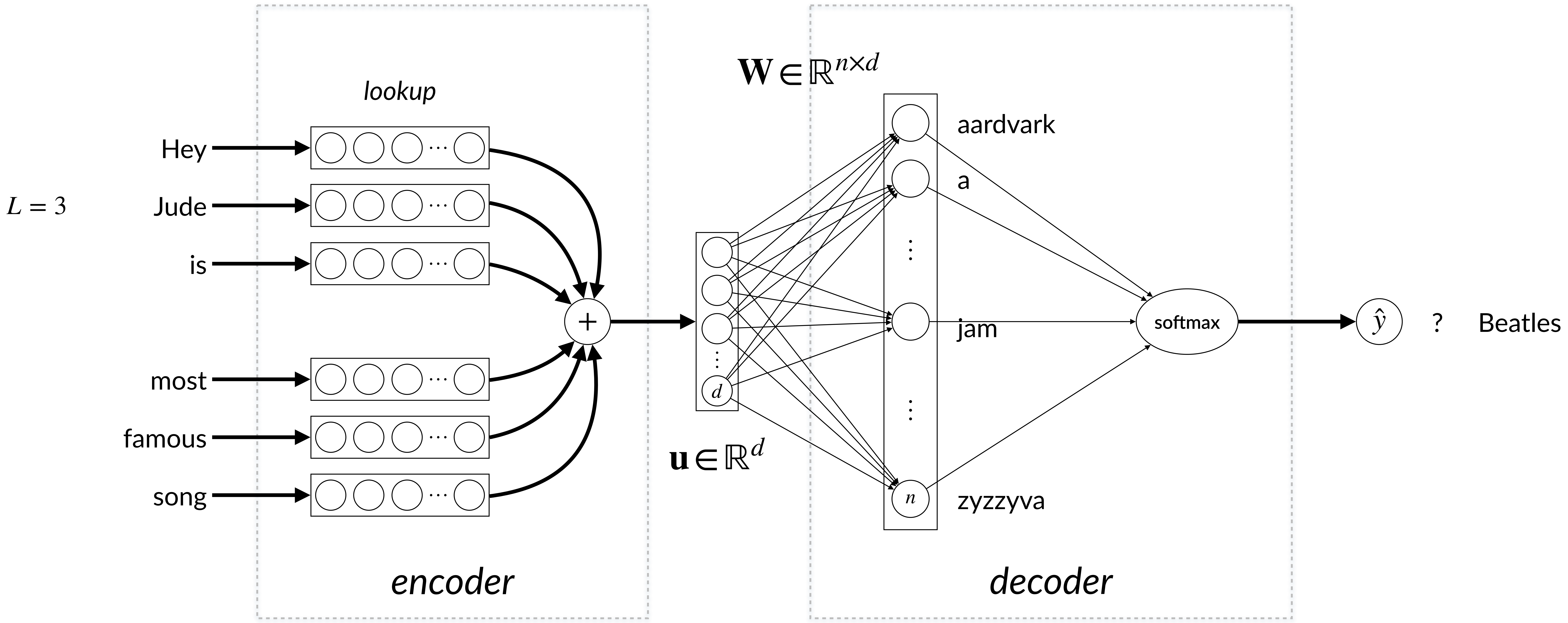
word2vec — Continuous Bag of Words (CBOW)



Text window: [Hey, Jude, is, **Beatles**, most, famous, song]

word2vec — Continuous Bag of Words (CBOW)

What do n and d denote?

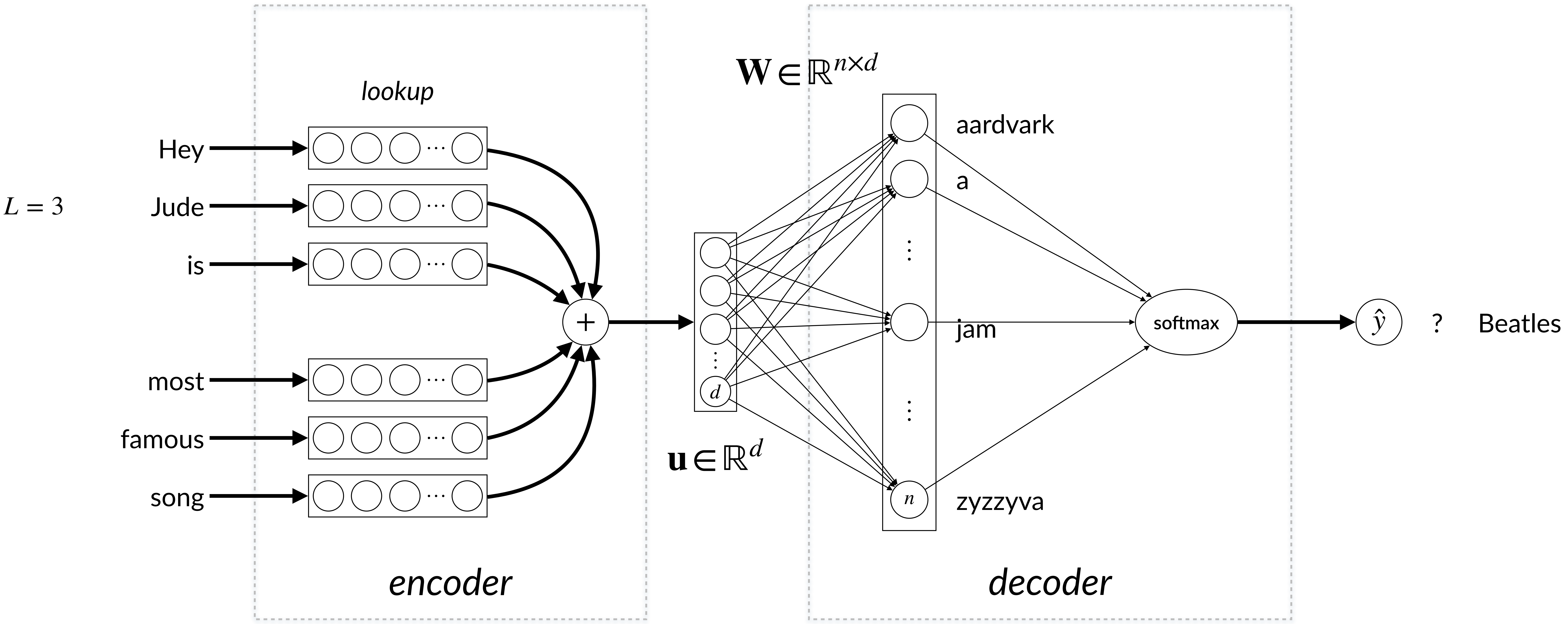


Text window: [Hey, Jude, is, **Beatles**, most, famous, song]

word2vec — Continuous Bag of Words (CBOW)

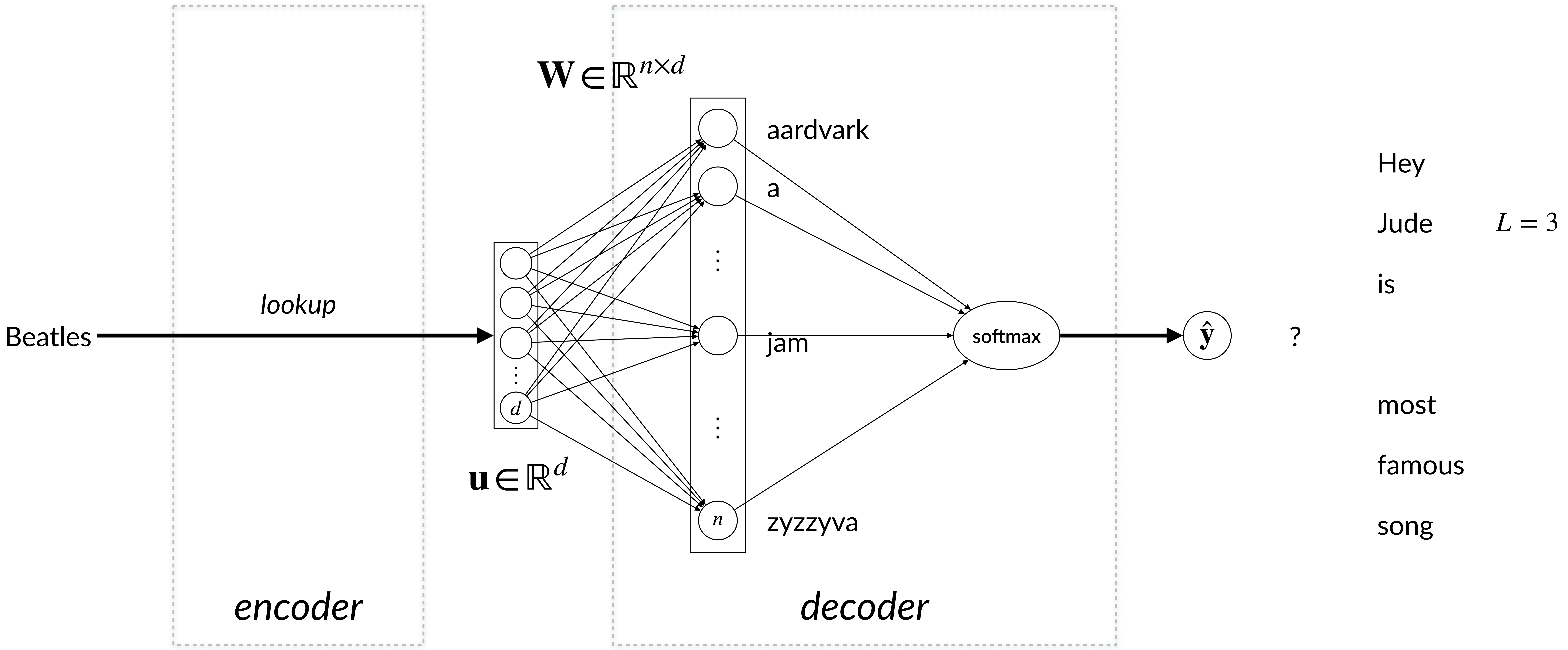
What do n and d denote?

Why do we use the softmax at the very end?



Text window: [Hey, Jude, is, **Beatles**, most, famous, song]

word2vec — skip-gram



Text window: [Hey, Jude, is, **Beatles**, most, famous, song]

word2vec — Target and context word embeddings

context words $[w_{t-3} \quad w_{t-2} \quad w_{t-1} \quad w_{t+1} \quad w_{t+2} \quad w_{t+3}]$

… said that "Hey Jude" is **Beatles**' most famous song, but…

target word

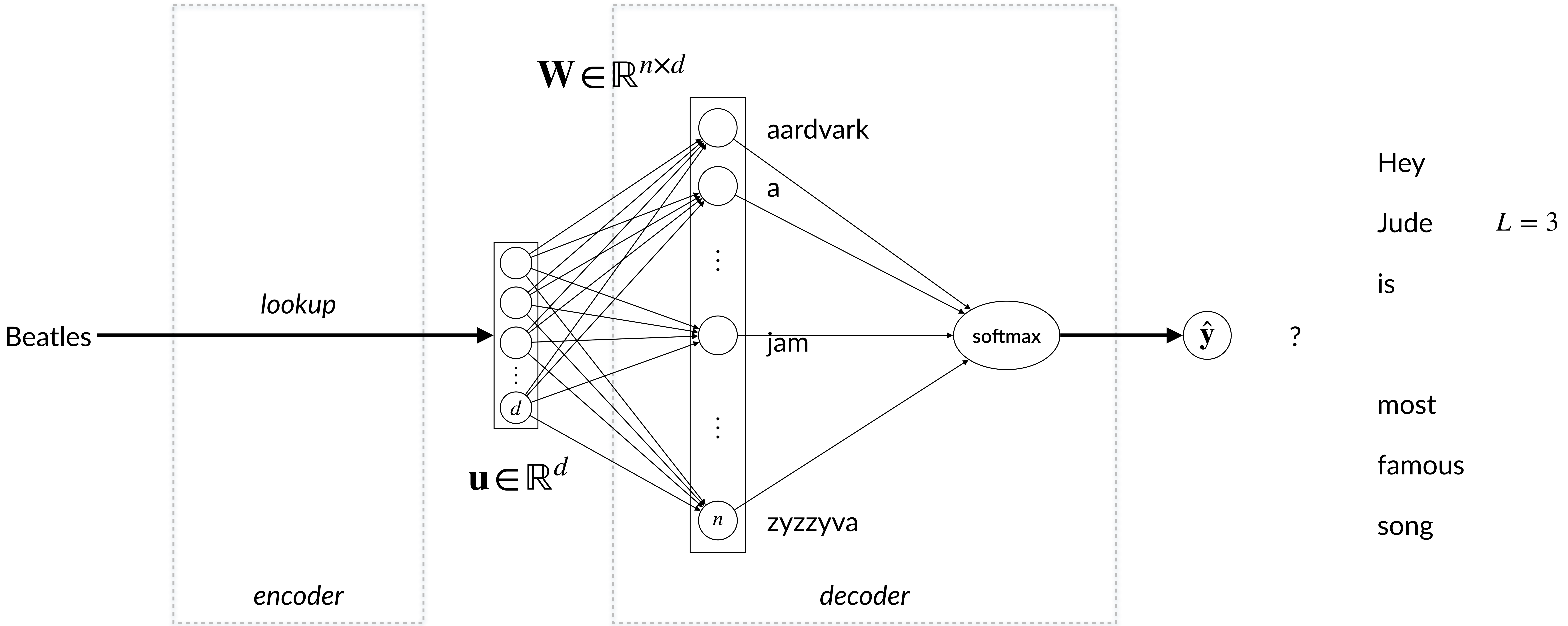
w_t

context radius

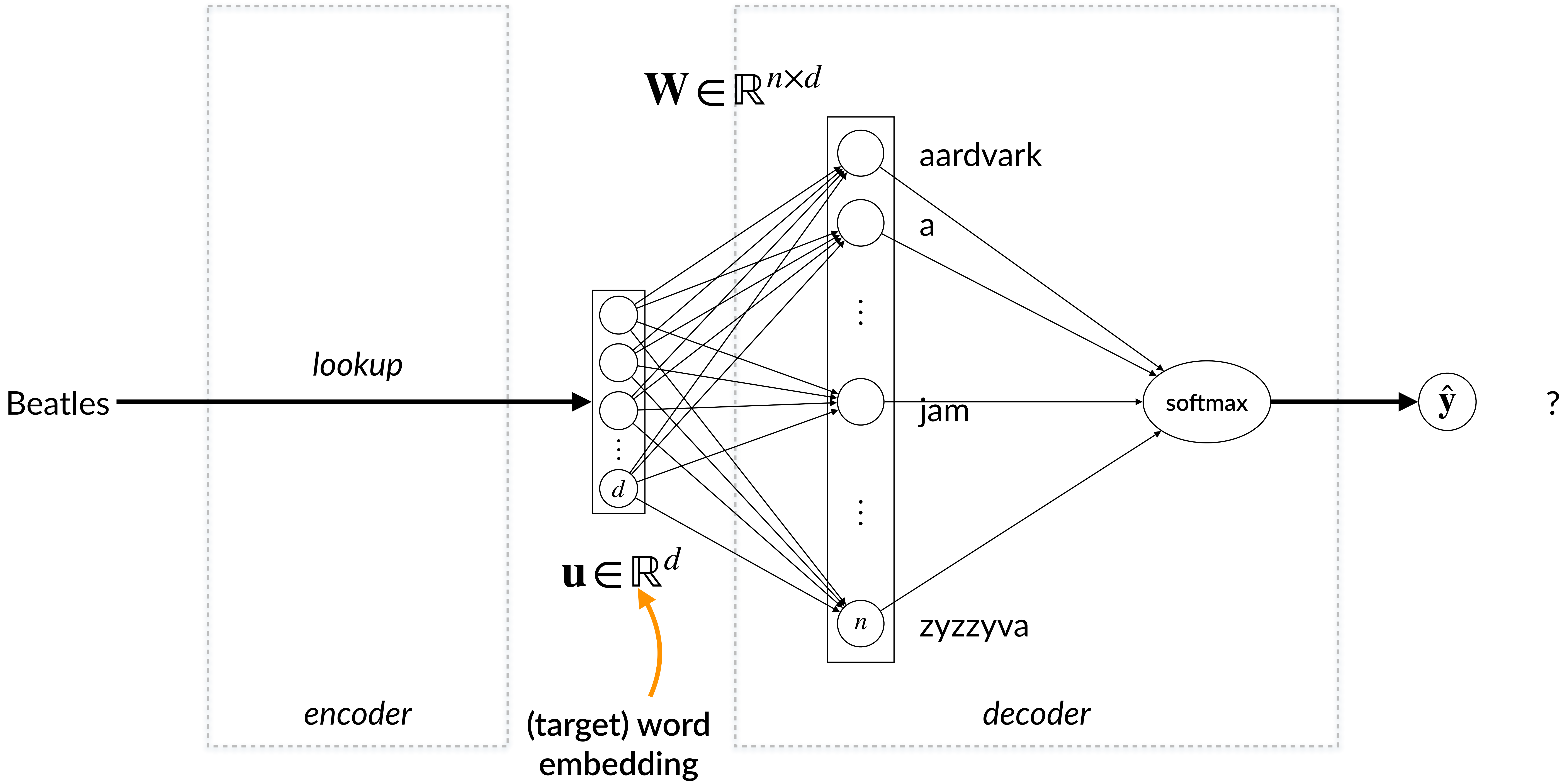
$L = 3$

context word $i \rightarrow \mathbf{c}_i \in \mathbb{R}^d$ $\quad \mathbf{C} \in \mathbb{R}^{n \times d}$ $\quad$ context word embeddings

target word $j \rightarrow \mathbf{u}_j \in \mathbb{R}^d$ $\quad \mathbf{U} \in \mathbb{R}^{d \times n}$ $\quad$ target word embeddings


$\mathbf{W} \in \mathbb{R}^{n \times d}$
aardvark
a
jam
zyzzyva
lookup
Beatles
d
n
$\mathbf{u} \in \mathbb{R}^d$
softmax
$\hat{\mathbf{y}}$
?
Hey
Jude
$L = 3$
is
most
famous
song
encoder
decoder

word2vec — skip-gram



Hey

Jude $\quad L = 3$

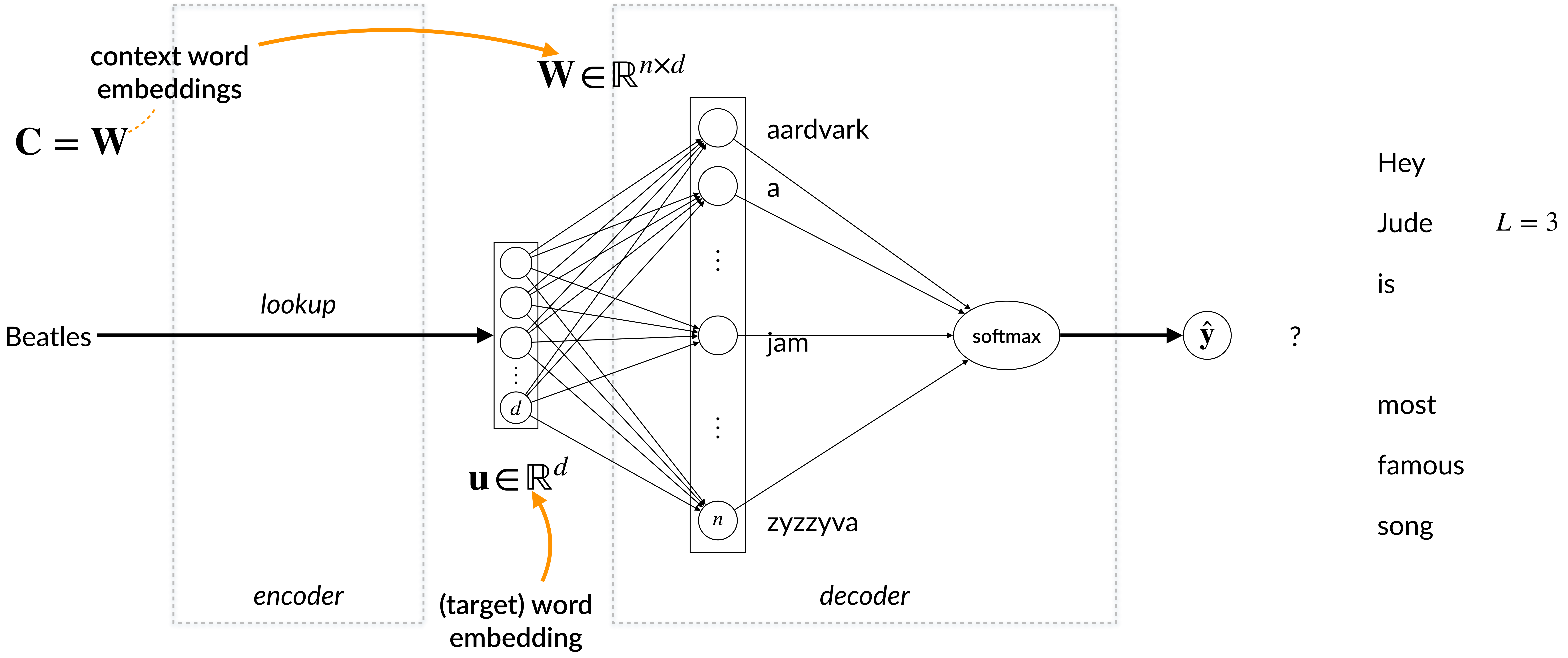
is

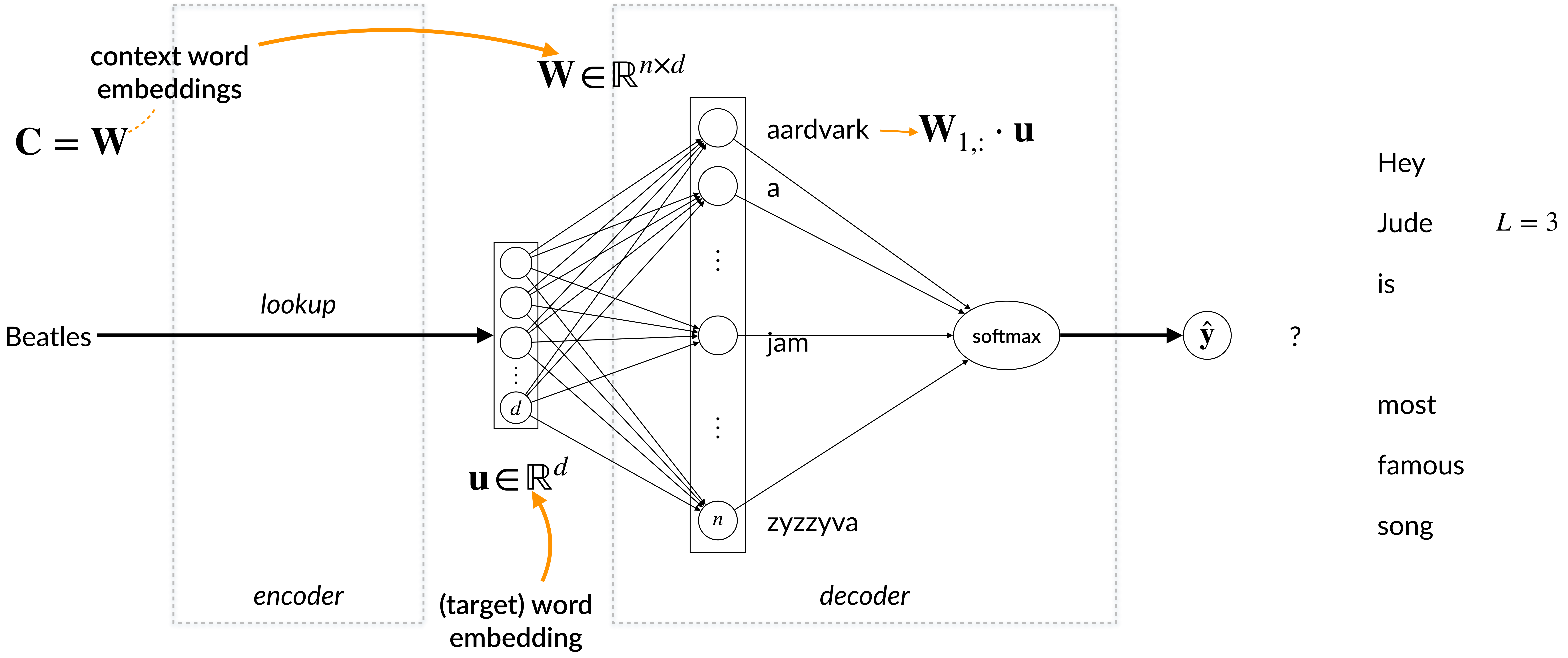
most

famous

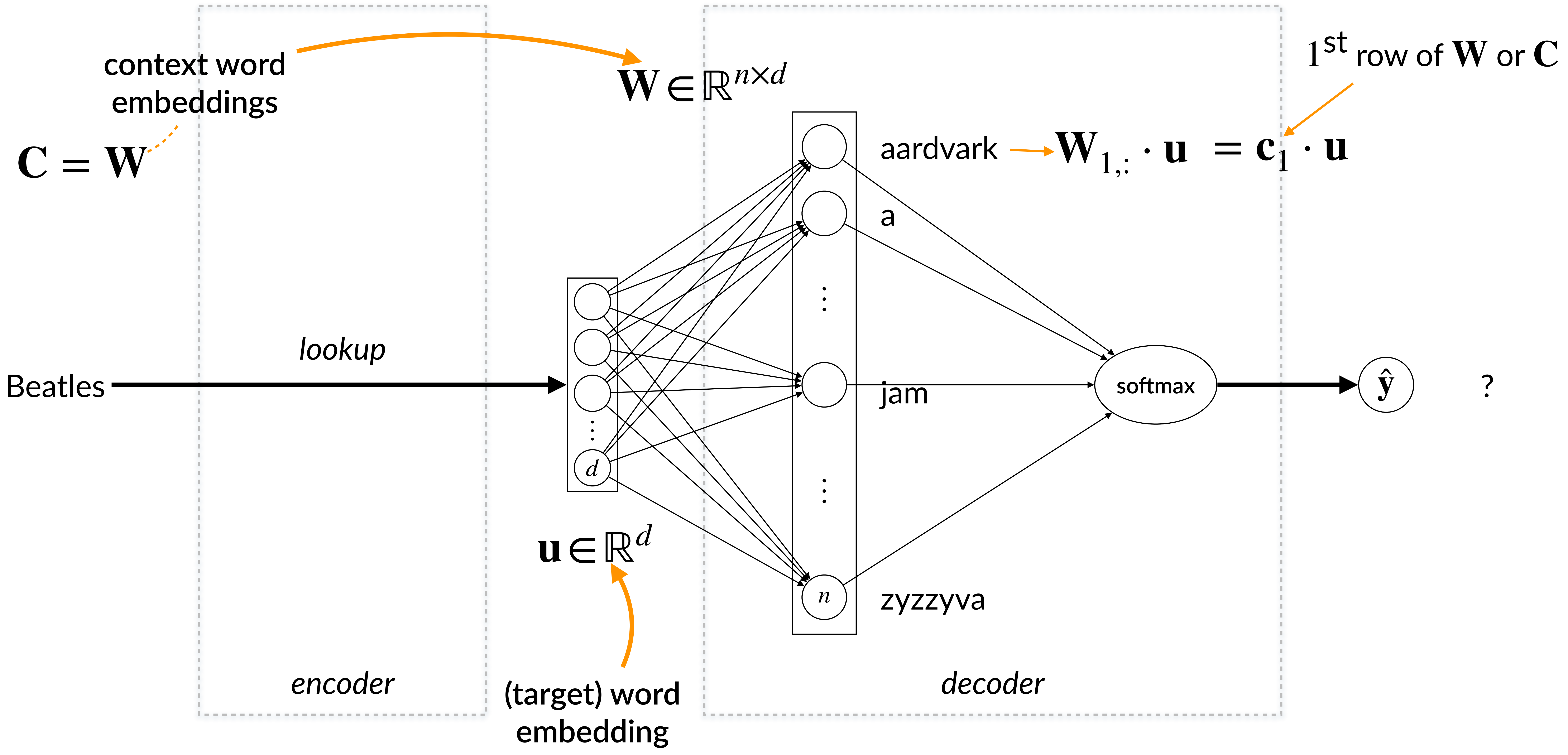
song

word2vec — skip-gram




context word
embeddings
$\mathbf{C} = \mathbf{W}$
$\mathbf{W} \in \mathbb{R}^{n \times d}$
aardvark
$\mathbf{W}_{1,:} \cdot \mathbf{u}$
a
jam
zyzzyva
n
d
lookup
Beatles
softmax
$\hat{\mathbf{y}}$
?
$\mathbf{u} \in \mathbb{R}^d$
(target) word
embedding
encoder
decoder
Hey
Jude
is
most
famous
song
$L = 3$

word2vec — skip-gram



Hey

Jude $L = 3$

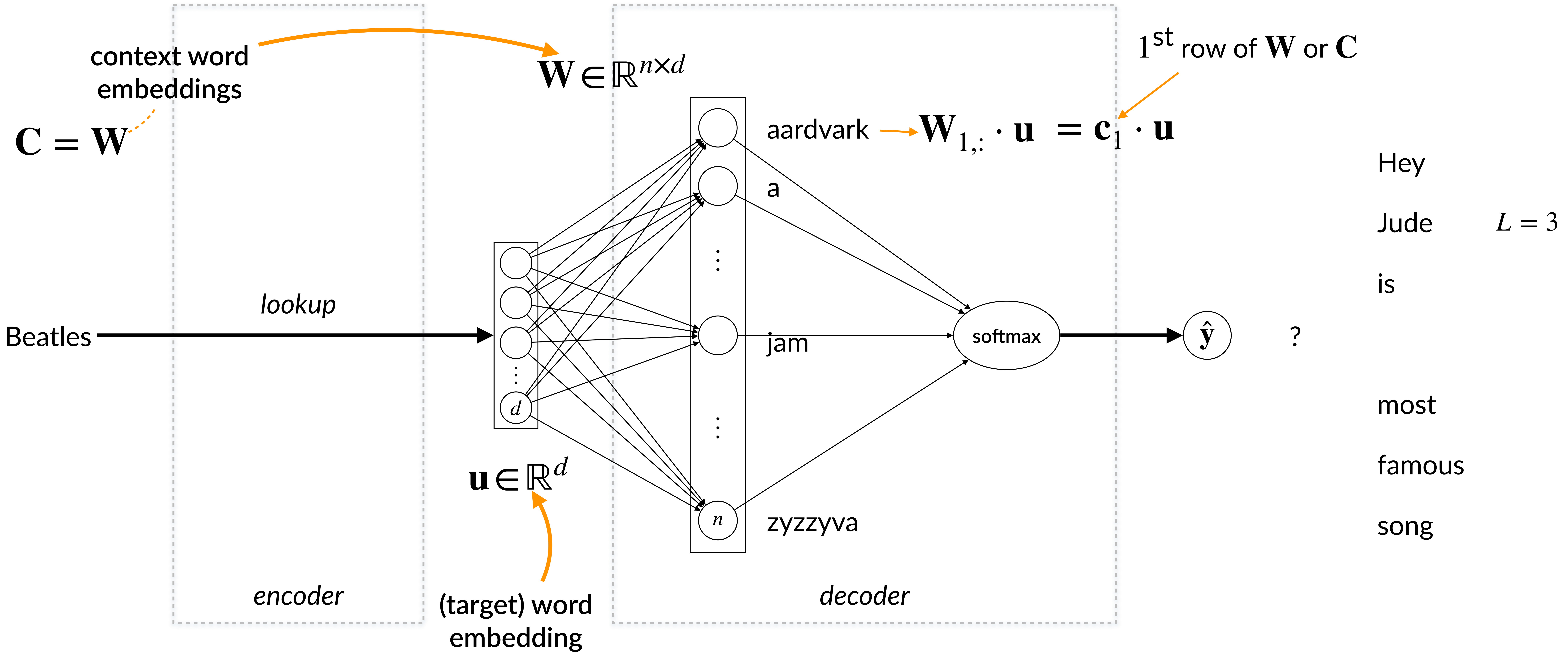
is

most

famous

song

word2vec — skip-gram



Objective: maximise the *dot product* $\in \mathbb{R}$ between context and centre words $\quad \mathrm{sim}(c_i, u_j) = \mathbf{c}_i \cdot \mathbf{u}_j$

word2vec — skip-gram

Imagine our corpus is a sequence of T tokens

$$w_1, w_2, \dots, w_T$$

word2vec — skip-gram

Imagine our corpus is a sequence of T tokens

How big is T ?

$$w_1, w_2, \ldots, w_T$$

Imagine our corpus is a sequence of T tokens

How big is T ?

$$w_1, w_2, \ldots, w_T$$

if our context radius $L = 2$ and our target word is w_t

word2vec — skip-gram

Imagine our corpus is a sequence of T tokens

How big is T ?

$$w_1, w_2, \ldots, w_T$$

if our context radius $L = 2$ and our target word is w_t

skip-gram aims to maximise this

$$p(w_{t-2} \mid w_t) \cdot p(w_{t-1} \mid w_t) \cdot p(w_{t+1} \mid w_t) \cdot p(w_{t+2} \mid w_t)$$

word2vec — skip-gram

Imagine our corpus is a sequence of T tokens

How big is T ?

$$w_1, w_2, \ldots, w_T$$

if our context radius $L = 2$ and our target word is w_t

skip-gram aims to maximise this

words are independent from each other

$$p(w_{t-2} \mid w_t) \cdot p(w_{t-1} \mid w_t) \cdot p(w_{t+1} \mid w_t) \cdot p(w_{t+2} \mid w_t)$$

word2vec — skip-gram

Imagine our corpus is a sequence of T tokens

How big is T ?

$$w_1, w_2, \ldots, w_T$$

if our context radius $L = 2$ and our target word is w_t

skip-gram aims to maximise this

words are independent from each other

$$p(w_{t-2} \mid w_t) \cdot p(w_{t-1} \mid w_t) \cdot p(w_{t+1} \mid w_t) \cdot p(w_{t+2} \mid w_t)$$

Does it matter if a word comes before or after w_t?

word2vec — skip-gram

Imagine our corpus is a sequence of T tokens

How big is T?

$$w_1, w_2, \ldots, w_T$$

if our context radius $L = 2$ and our target word is w_t

skip-gram aims to maximise this

words are independent from each other

$$p(w_{t-2} \mid w_t) \cdot p(w_{t-1} \mid w_t) \cdot p(w_{t+1} \mid w_t) \cdot p(w_{t+2} \mid w_t)$$

Does it matter if a word comes before or after w_t?

$$= \prod_{i=-L,\, i \neq 0}^{L} p(w_{t-i} \mid w_t)$$

word2vec — skip-gram

Imagine our corpus is a sequence of T tokens $w_1, w_2, \ldots, w_T$

for one context window $\max \prod_{i=-L,\, i\neq 0}^{L} p(w_{t-i} \mid w_t)$

word2vec — skip-gram

Imagine our corpus is a sequence of T tokens $w_1, w_2, \ldots, w_T$

for one context window $\max \prod_{i=-L, i\neq 0}^{L} p(w_{t-i} \mid w_t)$

across the entire corpus

word2vec — skip-gram

Imagine our corpus is a sequence of T tokens $w_1, w_2, \dots, w_T$

for one context window $\max \prod_{i=-L, i\neq 0}^{L} p(w_{t-i} \mid w_t)$

across the entire corpus $\max \frac{1}{T} \prod_{t=1}^{T} \prod_{i=-L, i\neq 0}^{L} p(w_{t-i} \mid w_t)$

word2vec — skip-gram

Imagine our corpus is a sequence of T tokens $\quad w_1, w_2, \ldots, w_T$

for one context window $\quad \max \prod_{i=-L,\, i\neq 0}^{L} p(w_{t-i} \mid w_t)$

across the entire corpus $\quad \max \frac{1}{T} \prod_{t=1}^{T} \prod_{i=-L,\, i\neq 0}^{L} p(w_{t-i} \mid w_t)$

let's work with the $\log$

why?

word2vec — skip-gram

Imagine our corpus is a sequence of T tokens $\quad w_1, w_2, \ldots, w_T$

for one context window $\quad \max \prod_{i=-L,\, i\neq 0}^{L} p(w_{t-i} \,|\, w_t)$

across the entire corpus $\quad \max \frac{1}{T} \prod_{t=1}^{T} \prod_{i=-L,\, i\neq 0}^{L} p(w_{t-i} \,|\, w_t)$

let's work with the $\log$ $\quad \max \frac{1}{T} \log \left(\prod_{t=1}^{T} \prod_{i=-L,\, i\neq 0}^{L} p(w_{t-i} \,|\, w_t) \right)$

why?

word2vec — skip-gram

Imagine our corpus is a sequence of T tokens $w_1, w_2, \ldots, w_T$

for one context window
$$\max \prod_{i=-L,\, i\neq 0}^{L} p(w_{t-i} \mid w_t)$$

across the entire corpus
$$\max \frac{1}{T} \prod_{t=1}^{T} \prod_{i=-L,\, i\neq 0}^{L} p(w_{t-i} \mid w_t)$$

let's work with the $\log$
$$\max \frac{1}{T} \log \left(\prod_{t=1}^{T} \prod_{i=-L,\, i\neq 0}^{L} p(w_{t-i} \mid w_t) \right) = \max \frac{1}{T} \sum_{t=1}^{T} \sum_{i=-L,\, i\neq 0}^{L} \log \left(p(w_{t-i} \mid w_t) \right)$$

why?

Imagine our corpus is a sequence of T tokens $w_1, w_2, \ldots, w_T$

let's work with the $\log$ $$\max \frac{1}{T} \log \left(\prod_{t=1}^{T} \prod_{i=-L,\, i \neq 0}^{L} p(w_{t-i} \,|\, w_t) \right) = \max \frac{1}{T} \sum_{t=1}^{T} \sum_{i=-L,\, i \neq 0}^{L} \log \left(p(w_{t-i} \,|\, w_t) \right)$$

Imagine our corpus is a sequence of T tokens $w_1, w_2, \ldots, w_T$

let's work with the $\log$

$$\max \frac{1}{T} \log \left(\prod_{t=1}^{T} \prod_{i=-L,\, i\neq 0}^{L} p(w_{t-i} \mid w_t) \right) = \max \frac{1}{T} \sum_{t=1}^{T} \sum_{i=-L,\, i\neq 0}^{L} \log \left(p(w_{t-i} \mid w_t) \right)$$

minimise this

$$\min - \frac{1}{T} \sum_{t=1}^{T} \sum_{i=-L,\, i\neq 0}^{L} \log \left(p(w_{t-i} \mid w_t) \right)$$

word2vec — skip-gram

Imagine our corpus is a sequence of T tokens $\quad w_1, w_2, \ldots, w_T$

let's work with the $\log$

$$\max \frac{1}{T} \log \left(\prod_{t=1}^{T} \prod_{i=-L,\, i\neq 0}^{L} p(w_{t-i} \,|\, w_t) \right) = \max \frac{1}{T} \sum_{t=1}^{T} \sum_{i=-L,\, i\neq 0}^{L} \log \big(p(w_{t-i} \,|\, w_t) \big)$$

minimise this

$$\min -\frac{1}{T} \sum_{t=1}^{T} \sum_{i=-L,\, i\neq 0}^{L} \log \big(p(w_{t-i} \,|\, w_t) \big)$$

- What are we minimising this against? Parameters of the model?
- How do we learn word embeddings from this?

word2vec — skip-gram

Imagine our corpus is a sequence of T tokens $w_1, w_2, \ldots, w_T$

$$\min -\frac{1}{T}\sum_{t=1}^{T}\sum_{i=-L,\, i\neq 0}^{L} \log\left(p(w_{t-i} \mid w_t)\right)$$

$$p(w_{t-i} \mid w_t)$$

context word target word

word2vec — skip-gram

$$p(w_{t-i} | w_t)$$

context word target word

word2vec — skip-gram

$$p(w_{t-i} \mid w_t)$$

context word $\qquad$ target word

Context word w_{t-i} is vocabulary word $c \in \mathscr{V}$

that has an embedding $\mathbf{c} \in \mathbb{R}^{1 \times d}$

assuming context embedding matrix $\mathbf{C} \in \mathbb{R}^{n \times d}$

word2vec — skip-gram

$$p(w_{t-i} \mid w_t) \qquad = p(c \mid u)$$

context word | target word

Context word w_{t-i} is vocabulary word $c \in \mathscr{V}$
that has an embedding $\mathbf{c} \in \mathbb{R}^{1 \times d}$
assuming context embedding matrix $\mathbf{C} \in \mathbb{R}^{n \times d}$

Target word w_t is $u \in \mathscr{V}$
with an embedding $\mathbf{u} \in \mathbb{R}^{d \times 1}$
assuming embedding matrix $\mathbf{U} \in \mathbb{R}^{d \times n}$

word2vec — skip-gram

$$p(w_{t-i} \mid w_t) \qquad = p(c \mid u)$$

context word target word

Context word w_{t-i} is vocabulary word $c \in \mathcal{V}$
that has an embedding $\mathbf{c} \in \mathbb{R}^{1 \times d}$
assuming context embedding matrix $\mathbf{C} \in \mathbb{R}^{n \times d}$

Target word w_t is $u \in \mathcal{V}$
with an embedding $\mathbf{u} \in \mathbb{R}^{d \times 1}$
assuming embedding matrix $\mathbf{U} \in \mathbb{R}^{d \times n}$

$$\text{sim}(w_{t-1}, w_t) = \text{sim}(c, u) = \mathbf{c} \cdot \mathbf{u}$$

dot product!

word2vec — skip-gram

$$p(w_{t-i} \mid w_t) \quad = p(c \mid u)$$

context word

target word

Context word w_{t-i} is vocabulary word $c \in \mathscr{V}$
that has an embedding $\mathbf{c} \in \mathbb{R}^{1 \times d}$
assuming context embedding matrix $\mathbf{C} \in \mathbb{R}^{n \times d}$

Target word w_t is $u \in \mathscr{V}$
with an embedding $\mathbf{u} \in \mathbb{R}^{d \times 1}$
assuming embedding matrix $\mathbf{U} \in \mathbb{R}^{d \times n}$

$$\mathrm{sim}(w_{t-1}, w_t) = \mathrm{sim}(c, u) = \mathbf{c} \cdot \mathbf{u}$$

dot product!

$$p(c \mid u) = \frac{\exp\left(\mathbf{c} \cdot \mathbf{u}\right)}{\sum_{\mathbf{c}_k \in \mathbf{C}} \exp\left(\mathbf{c}_k \cdot \mathbf{u}\right)}$$

normalise
using softmax

word2vec — skip-gram

$$p(w_{t-i} \mid w_t) \quad = p(c \mid u)$$

context word

target word

Context word w_{t-i} is vocabulary word $c \in \mathscr{V}$
that has an embedding $\mathbf{c} \in \mathbb{R}^{1 \times d}$
assuming context embedding matrix $\mathbf{C} \in \mathbb{R}^{n \times d}$

Target word w_t is $u \in \mathscr{V}$
with an embedding $\mathbf{u} \in \mathbb{R}^{d \times 1}$
assuming embedding matrix $\mathbf{U} \in \mathbb{R}^{d \times n}$

$$\mathrm{sim}(w_{t-1}, w_t) = \mathrm{sim}(c, u) = \mathbf{c} \cdot \mathbf{u}$$

dot product!

Is it expensive to compute the denominator of this?

$$p(c \mid u) = \frac{\exp\left(\mathbf{c} \cdot \mathbf{u}\right)}{\sum_{\mathbf{c}_k \in \mathbf{C}} \exp\left(\mathbf{c}_k \cdot \mathbf{u}\right)}$$

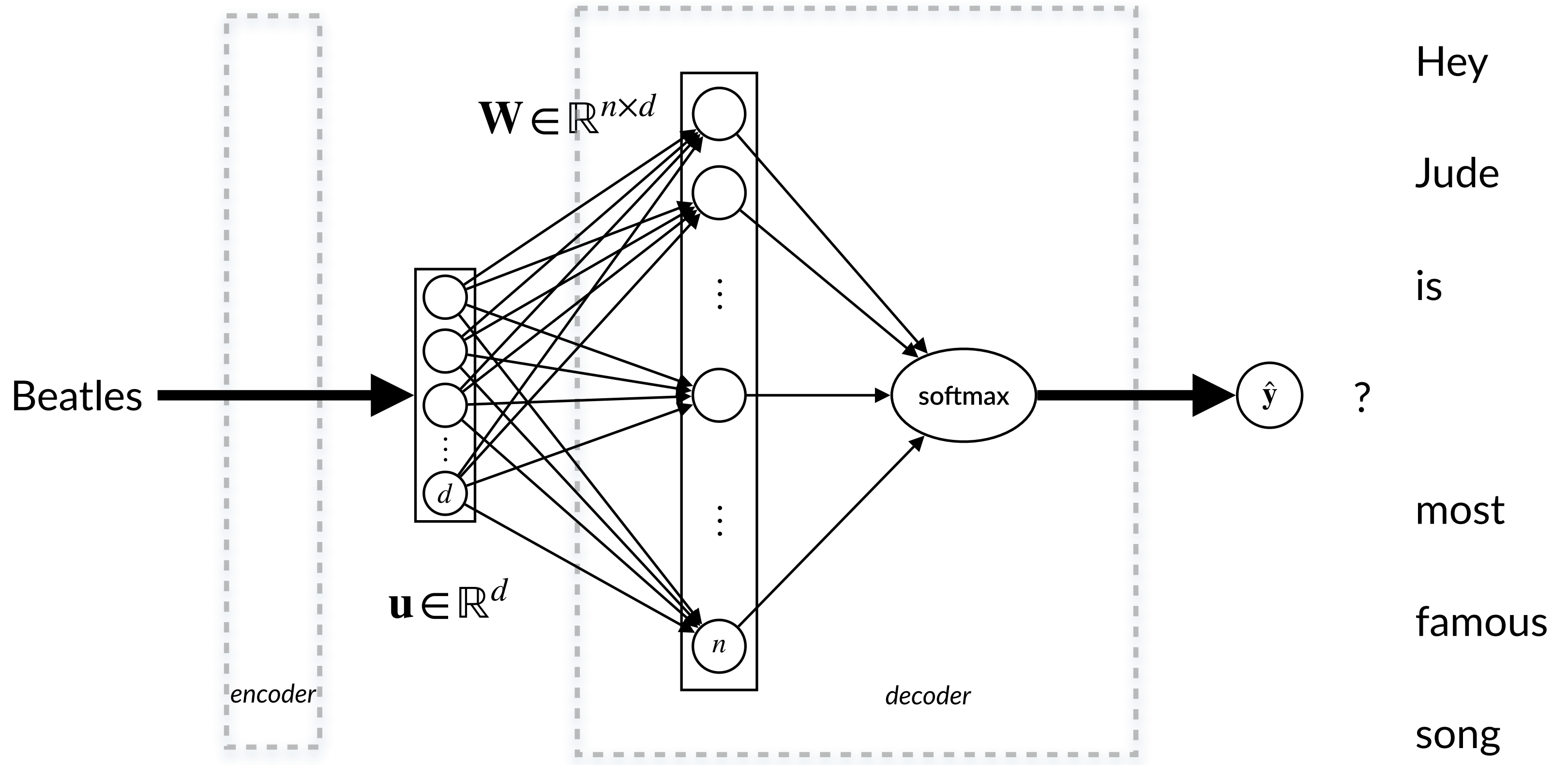
normalise
using softmax

word2vec — skip-gram

Imagine our corpus is a sequence of T tokens

$$w_1, w_2, \ldots, w_T$$

$$\min -\frac{1}{T}\sum_{t=1}^{T}\sum_{i=-L,\, i\neq 0}^{L} \log\left(p(w_{t-i}\,|\,w_t)\right)$$

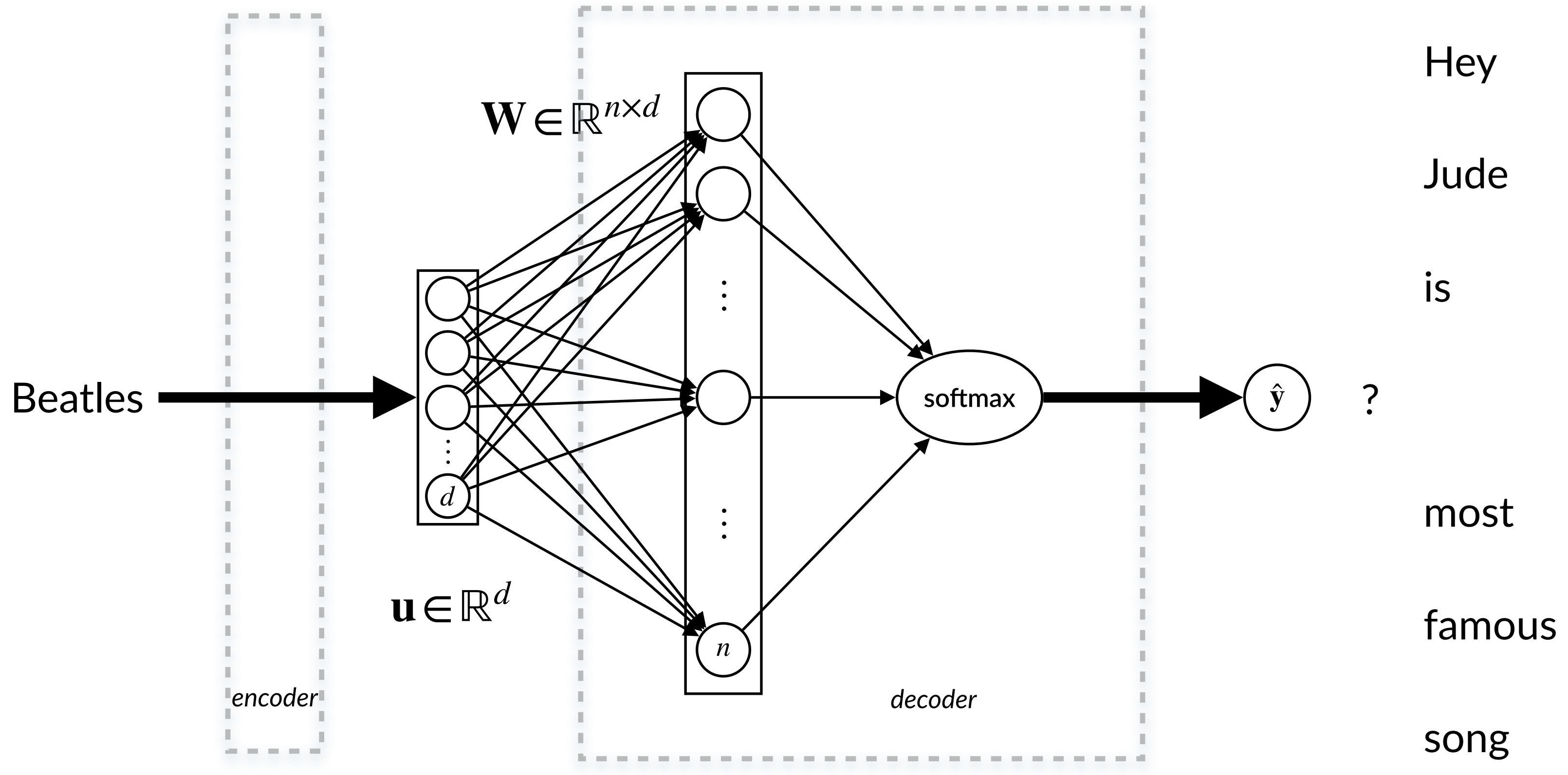


Imagine our corpus is a sequence of T tokens

$$w_1, w_2, \ldots, w_T$$

$$\min -\frac{1}{T}\sum_{t=1}^{T}\sum_{i=-L,\, i\neq 0}^{L} \log\left(p(w_{t-i}\,|\,w_t)\right)$$

let's insert the previous information

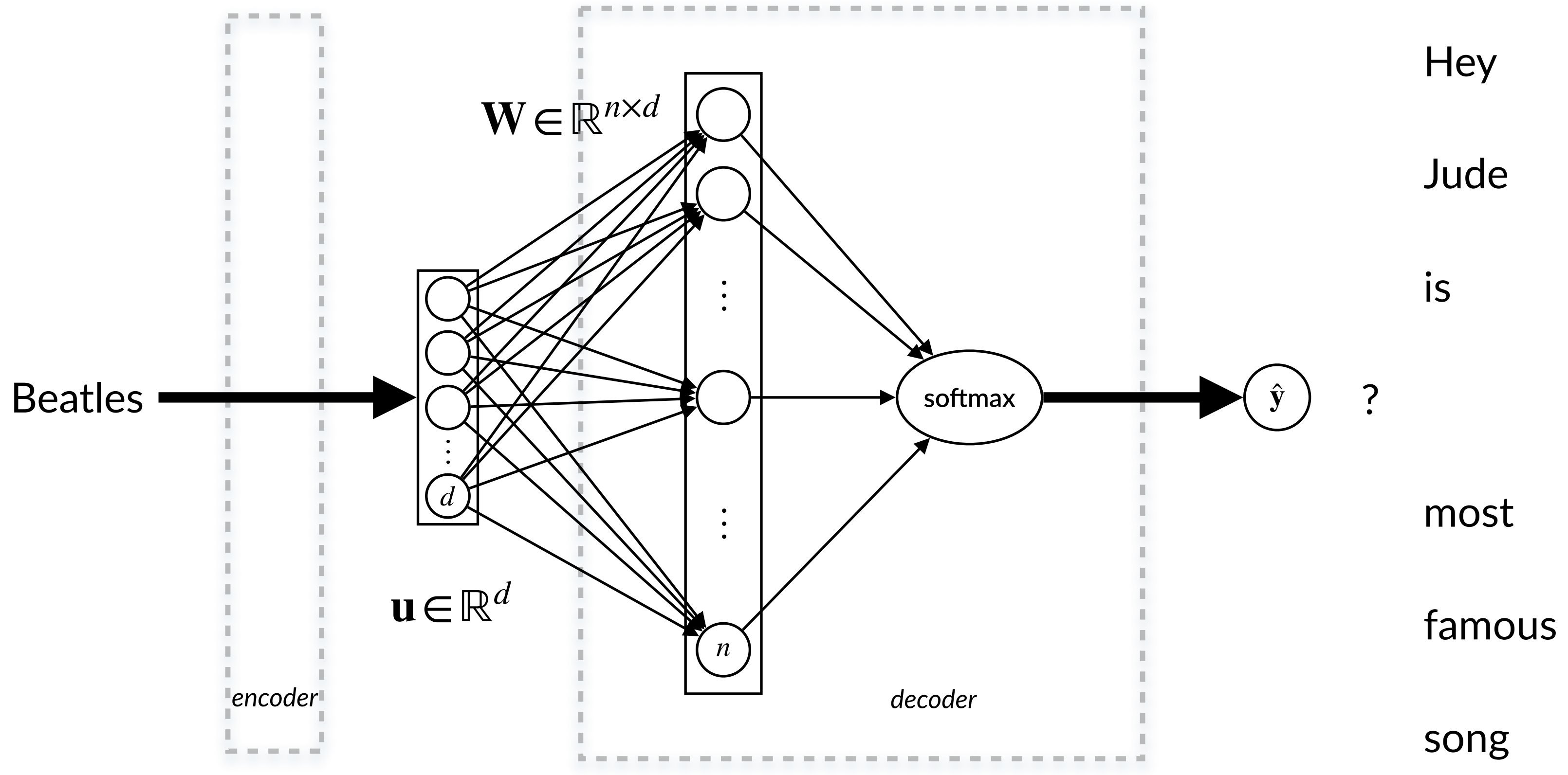


word2vec — skip-gram

Imagine our corpus is a sequence of T tokens

$$w_1, w_2, \ldots, w_T$$

$$\min -\frac{1}{T}\sum_{t=1}^{T}\sum_{i=-L,\, i\neq 0}^{L} \log\left(p(w_{t-i} \mid w_t)\right)$$



let's insert the previous information

Opt. task: $$\arg\min_{\mathbf{C},\mathbf{U}} -\frac{1}{T}\sum_{t=1}^{T}\sum_{i=-L,\, i\neq 0}^{L} \log\left(\frac{\exp\left(\mathbf{c}_{w_{t-i}} \cdot \mathbf{u}_{w_t}\right)}{\sum_{j=1}^{n}\exp\left(\mathbf{c}_j \cdot \mathbf{u}_{w_t}\right)}\right)$$

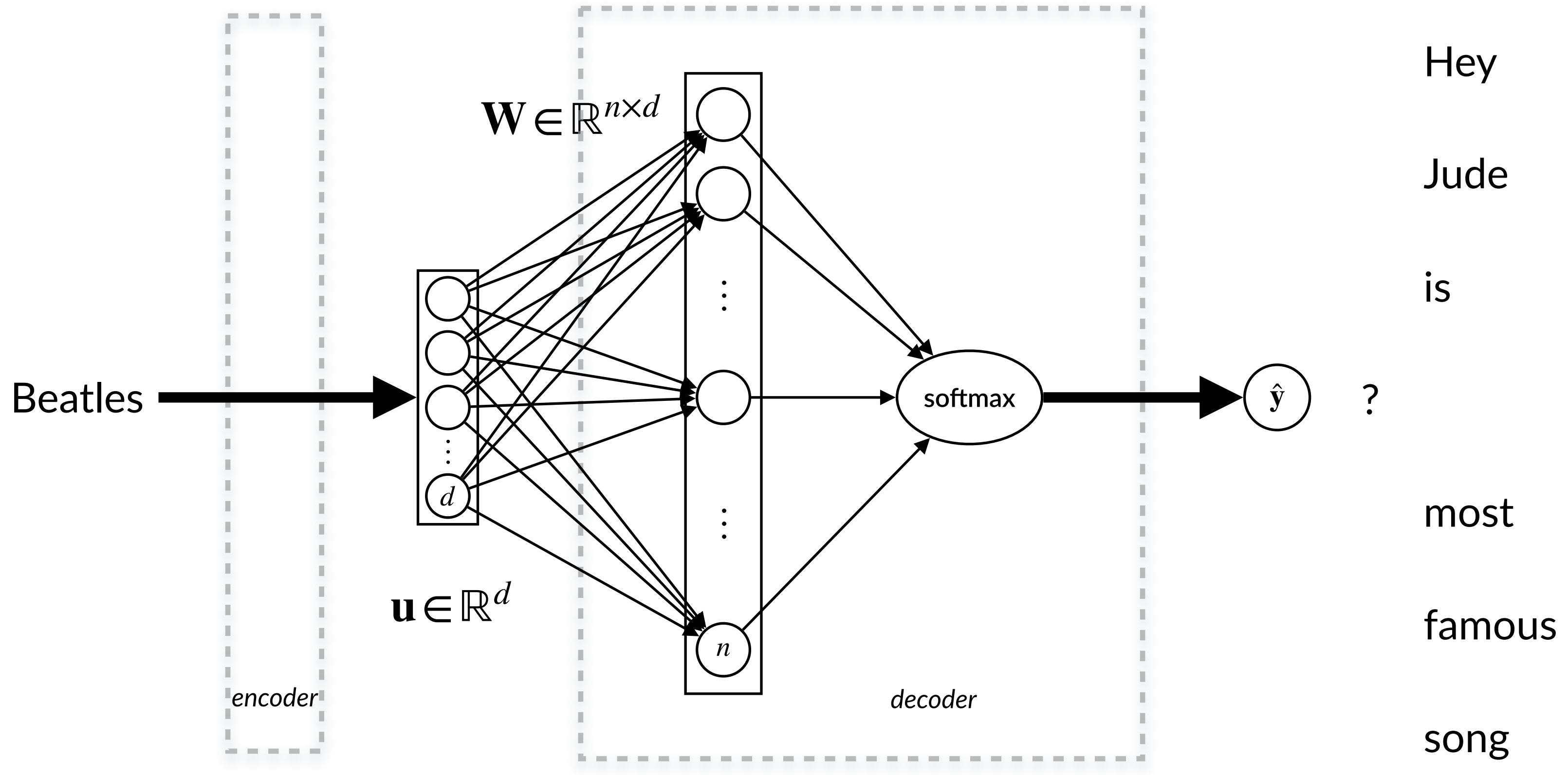
embedding matrices

word2vec — skip-gram

Imagine our corpus is a sequence of T tokens

$$w_1, w_2, \ldots, w_T$$

$$\min -\frac{1}{T}\sum_{t=1}^{T}\sum_{i=-L,\, i\neq 0}^{L} \log\left(p(w_{t-i} \mid w_t)\right)$$



let's insert the previous information

Opt. task: $$\arg\min_{\mathbf{C},\mathbf{U}} -\frac{1}{T}\sum_{t=1}^{T}\sum_{i=-L,\, i\neq 0}^{L} \log\left(\frac{\exp\left(\mathbf{c}_{w_{t-i}}\cdot \mathbf{u}_{w_t}\right)}{\sum_{j=1}^{n}\exp\left(\mathbf{c}_{j}\cdot \mathbf{u}_{w_t}\right)}\right)$$

embedding matrices

ranks all words in the vocabulary in terms of their probability of being within the context window

too expensive!!!

Solution: *Let's change the objective function by using "negative sampling"!*

Given a target word u and another word v
model the probability of u and v appearing in the same context
$\Longrightarrow$ **binary classification**

Solution: *Let's change the objective function by using "negative sampling"!*

Given a target word u and another word v

model the probability of u and v appearing in the same context

$\Longrightarrow$ **binary classification**

they appear
in the same context

$$p(D = 1 \mid v, u) = \sigma(\mathbf{v} \cdot \mathbf{u}) = \frac{1}{1 + \exp(-\mathbf{v} \cdot \mathbf{u})}$$

Solution: *Let's change the objective function by using "negative sampling"!*

Given a target word u and another word v

model the probability of u and v appearing in the same context

$\Longrightarrow$ **binary classification**

they appear
in the same context

$$p(D = 1 \,|\, v, u) = \sigma(\mathbf{v} \cdot \mathbf{u}) = \frac{1}{1 + \exp(-\mathbf{v} \cdot \mathbf{u})}$$

they don't appear
in the same context

$$p(D = 0 \,|\, v, u) = 1 - p(D = 1 \,|\, v, u) = 1 - \sigma(\mathbf{v} \cdot \mathbf{u}) = \sigma(-\mathbf{v} \cdot \mathbf{u})$$

word2vec — skip-gram *with negative sampling*

Now, if u is our target word and c a context word
we want to maximise

$$\arg\max_{\mathbf{C},\mathbf{U}} \prod_{\{c,u\}\in\mathscr{D}} p(D = 1 \,|\, c, u)$$

where $\mathscr{D}$ holds all target-context word pairs in our corpus

Now, if u is our target word and c a context word
we want to maximise

$$\arg\max_{\mathbf{C},\mathbf{U}} \prod_{\{c,u\}\in\mathscr{D}} p(D = 1 \mid c, u)$$

where $\mathscr{D}$ holds all target-context word pairs in our corpus

$$\arg\max_{\mathbf{C},\mathbf{U}} \prod_{\{c,u\}\in\mathscr{D}} \sigma(\mathbf{c} \cdot \mathbf{u})$$

Now, if u is our target word and c a context word
we want to maximise

$$\arg\max_{\mathbf{C},\mathbf{U}} \prod_{\{c,u\}\in\mathcal{D}} p(D=1\,|\,c,u)$$

where $\mathcal{D}$ holds all target-context word pairs in our corpus

$$\arg\max_{\mathbf{C},\mathbf{U}} \prod_{\{c,u\}\in\mathcal{D}} \sigma(\mathbf{c}\cdot\mathbf{u}) \quad \xrightarrow{\log(\,\cdot\,)}$$

word2vec — skip-gram *with negative sampling*

Now, if u is our target word and c a context word
we want to maximise

$$\arg\max_{\mathbf{C},\mathbf{U}} \prod_{\{c,u\}\in\mathscr{D}} p(D=1 \mid c, u)$$

where $\mathscr{D}$ holds all target-context word pairs in our corpus

$$\arg\max_{\mathbf{C},\mathbf{U}} \prod_{\{c,u\}\in\mathscr{D}} \sigma(\mathbf{c}\cdot\mathbf{u}) \xrightarrow{\log(\,\cdot\,)} \arg\max_{\mathbf{C},\mathbf{U}} \sum_{\{c,u\}\in\mathscr{D}} \log\big(\sigma(\mathbf{c}\cdot\mathbf{u})\big)$$

word2vec — skip-gram *with negative sampling*

u is our target word and c a context word

$$\arg\max_{\mathbf{C},\mathbf{U}} \sum_{\{c,u\}\in\mathscr{D}} \log\big(\sigma(\mathbf{c}\cdot\mathbf{u})\big)$$

word2vec — skip-gram *with negative sampling*

u is our target word and c a context word

$$\underset{\mathbf{C},\mathbf{U}}{\arg\max} \sum_{\{c,u\}\in\mathscr{D}} \log\big(\sigma(\mathbf{c}\cdot\mathbf{u})\big)$$

but an undesirable setting that maximises this function is…

word2vec — skip-gram *with negative sampling*

u is our target word and c a context word

$$\arg\max_{\mathbf{C},\mathbf{U}} \sum_{\{c,u\}\in\mathcal{D}} \log\big(\sigma(\mathbf{c}\cdot\mathbf{u})\big)$$

but an undesirable setting that maximises this function is…

$\mathbf{c} = \mathbf{u}^\top$ and $\mathbf{c}\cdot\mathbf{u} = k,$ where $k \geq 40$

word2vec — skip-gram *with negative sampling*

u is our target word and c a context word

$$\underset{\mathbf{C},\mathbf{U}}{\arg\max} \sum_{\{c,u\}\in\mathscr{D}} \log\big(\sigma(\mathbf{c}\cdot\mathbf{u})\big)$$

but an undesirable setting that maximises this function is…

$\mathbf{c} = \mathbf{u}^\top$ and $\mathbf{c}\cdot\mathbf{u} = k,$ where $k \geq 40$

$\implies \sigma(\mathbf{c}\cdot\mathbf{u}) = \sigma(40) \approx 1$ logistic sigmoid's max value

word2vec — skip-gram *with negative sampling*

u is our target word and c a context word

$$\underset{\mathbf{C},\mathbf{U}}{\arg\max} \sum_{\{c,u\} \in \mathscr{D}} \log\big(\sigma(\mathbf{c} \cdot \mathbf{u})\big)$$

Fix: generate random pairs ($\mathscr{D}'$) and consider them as "***negative***" target-context pairs

$$\underset{\mathbf{C},\mathbf{U}}{\arg\max} \sum_{\{c,u\} \in \mathscr{D}} \log\big(\sigma(\mathbf{c} \cdot \mathbf{u})\big) + \sum_{\{c,u\} \in \mathscr{D}'} \log\big(\sigma(-\mathbf{c} \cdot \mathbf{u})\big)$$

word2vec — skip-gram *with negative sampling*

u is our target word and c a context word

$$\arg\max_{\mathbf{C},\mathbf{U}} \sum_{\{c,u\}\in\mathscr{D}} \log\big(\sigma(\mathbf{c}\cdot\mathbf{u})\big)$$

Fix: generate random pairs ($\mathscr{D}'$) and consider them as "***negative***" target-context pairs

$$\arg\max_{\mathbf{C},\mathbf{U}} \sum_{\{c,u\}\in\mathscr{D}} \log\big(\sigma(\mathbf{c}\cdot\mathbf{u})\big) + \sum_{\{c,u\}\in\mathscr{D}'} \log\big(\sigma(-\mathbf{c}\cdot\mathbf{u})\big)$$

minimise this!

$$\arg\min_{\mathbf{C},\mathbf{U}} -\left[\sum_{\{c,u\}\in\mathscr{D}} \log\big(\sigma(\mathbf{c}\cdot\mathbf{u})\big) + \sum_{\{c,u\}\in\mathscr{D}'} \log\big(\sigma(-\mathbf{c}\cdot\mathbf{u})\big)\right]$$

word2vec — skip-gram *with negative sampling*

Suppose we have a target word u, a valid context word c,
and k noise words h_i, $i \in \{1, \ldots, k\}$ (negative samples) chosen randomly

Suppose we have a target word u, a valid context word c,
and k noise words $h_i\,, i \in \{1, \ldots, k\}$ (negative samples) chosen randomly

Logistic cross-entropy loss

$$L_{\text{ce}} = -\left[\log\Big(\sigma(\mathbf{c}\cdot\mathbf{u})\Big) + \sum_{i=1}^{k}\log\Big(\sigma(-\mathbf{h}_i\cdot\mathbf{u})\Big)\right]$$

word2vec — skip-gram *with negative sampling*

Suppose we have a target word u, a valid context word c,
and k noise words h_i, $i \in \{1, \ldots, k\}$ (negative samples) chosen randomly

Logistic cross-entropy loss $$L_{\text{ce}} = -\left[\log\big(\sigma(\mathbf{c} \cdot \mathbf{u})\big) + \sum_{i=1}^{k} \log\big(\sigma(-\mathbf{h}_i \cdot \mathbf{u})\big)\right]$$

$k + 1$ context word embeddings

$$\frac{\partial L_{\text{ce}}}{\partial \mathbf{c}} = ?$$

$$\frac{\partial L_{\text{ce}}}{\partial \mathbf{h}_i} = ?$$

target word embedding

$$\frac{\partial L_{\text{ce}}}{\partial \mathbf{u}} = ?$$

word2vec — skip-gram *with negative sampling*

Suppose we have a target word u, a valid context word c,
and k noise words $h_i, i \in \{1,\ldots,k\}$ (negative samples) chosen randomly

$$L_{\text{ce}} = -\left[\log\big(\sigma(\mathbf{c}\cdot\mathbf{u})\big) + \sum_{i=1}^{k}\log\big(\sigma(-\mathbf{h}_i\cdot\mathbf{u})\big)\right]$$

$$\frac{\partial L_{\text{ce}}}{\partial \mathbf{c}} =$$

Suppose we have a target word u, a valid context word c,
and k noise words $h_i, i \in \{1, \ldots, k\}$ (negative samples) chosen randomly

$$L_{\text{ce}} = -\left[\log\big(\sigma(\mathbf{c} \cdot \mathbf{u})\big) + \sum_{i=1}^{k} \log\big(\sigma(-\mathbf{h}_i \cdot \mathbf{u})\big)\right]$$

$$0$$

$$\frac{\partial L_{\text{ce}}}{\partial \mathbf{c}} =$$

word2vec — skip-gram *with negative sampling*

Suppose we have a target word u, a valid context word c,
and k noise words $h_i\,, i \in \{1, \ldots, k\}$ (negative samples) chosen randomly

$$L_{\text{ce}} = -\left[\log\left(\sigma(\mathbf{c}\cdot\mathbf{u})\right) + \sum_{i=1}^{k}\log\left(\sigma(-\mathbf{h}_i\cdot\mathbf{u})\right)\right]$$

$$0$$

chain rule…!

$$\frac{\partial L_{\text{ce}}}{\partial \mathbf{c}} = \frac{1}{\sigma(\mathbf{c}\cdot\mathbf{u})}$$

word2vec — skip-gram *with negative sampling*

Suppose we have a target word u, a valid context word c,
and k noise words $h_i, i \in \{1, \ldots, k\}$ (negative samples) chosen randomly

$$L_{\text{ce}} = -\left[\log\big(\sigma(\mathbf{c} \cdot \mathbf{u})\big) + \sum_{i=1}^{k} \log\big(\sigma(-\mathbf{h}_i \cdot \mathbf{u})\big)\right]$$

0

reminder

$$\frac{d\sigma(x)}{dx} = \sigma(x) \cdot \big(1 - \sigma(x)\big)$$

chain rule...!

$$\frac{\partial L_{\text{ce}}}{\partial \mathbf{c}} = \frac{1}{\sigma(\mathbf{c} \cdot \mathbf{u})} \cdot \sigma(\mathbf{c} \cdot \mathbf{u}) \cdot \big(1 - \sigma(\mathbf{c} \cdot \mathbf{u})\big)$$

word2vec — skip-gram *with negative sampling*

Suppose we have a target word u, a valid context word c,
and k noise words h_i, $i \in \{1, \dots, k\}$ (negative samples) chosen randomly

$$L_{\text{ce}} = -\left[\log\big(\sigma(\mathbf{c} \cdot \mathbf{u})\big) + \sum_{i=1}^{k} \log\big(\sigma(-\mathbf{h}_i \cdot \mathbf{u})\big)\right] \rightarrow 0$$

reminder

$$\frac{d\sigma(x)}{dx} = \sigma(x) \cdot \big(1 - \sigma(x)\big)$$

chain rule...!

$$\frac{\partial L_{\text{ce}}}{\partial \mathbf{c}} = \frac{1}{\sigma(\mathbf{c} \cdot \mathbf{u})} \cdot \sigma(\mathbf{c} \cdot \mathbf{u}) \cdot \big(1 - \sigma(\mathbf{c} \cdot \mathbf{u})\big) \cdot \mathbf{u}$$

word2vec — skip-gram *with negative sampling*

Suppose we have a target word u, a valid context word c,
and k noise words h_i , $i \in \{1,\ldots,k\}$ (negative samples) chosen randomly

$$L_{\text{ce}} = -\left[\log\big(\sigma(\mathbf{c}\cdot\mathbf{u})\big) + \sum_{i=1}^{k}\log\big(\sigma(-\mathbf{h}_i\cdot\mathbf{u})\big)\right] \rightarrow 0$$

reminder

$$\frac{d\sigma(x)}{dx} = \sigma(x)\cdot\big(1-\sigma(x)\big)$$

chain rule...!

$$\frac{\partial L_{\text{ce}}}{\partial \mathbf{c}} = -\frac{1}{\sigma(\mathbf{c}\cdot\mathbf{u})}\cdot\sigma(\mathbf{c}\cdot\mathbf{u})\cdot\big(1-\sigma(\mathbf{c}\cdot\mathbf{u})\big)\cdot\mathbf{u}$$

word2vec — skip-gram *with negative sampling*

Suppose we have a target word u, a valid context word c,
and k noise words h_i, $i \in \{1, \ldots, k\}$ (negative samples) chosen randomly

$$L_{\text{ce}} = -\left[\log\big(\sigma(\mathbf{c}\cdot\mathbf{u})\big) + \cancel{\sum_{i=1}^{k}\log\big(\sigma(-\mathbf{h}_i\cdot\mathbf{u})\big)}\right] \rightarrow 0$$

reminder

$$\frac{d\sigma(x)}{dx} = \sigma(x)\cdot\big(1-\sigma(x)\big)$$

chain rule...!

$$\frac{\partial L_{\text{ce}}}{\partial \mathbf{c}} = -\frac{1}{\cancel{\sigma(\mathbf{c}\cdot\mathbf{u})}}\cdot\cancel{\sigma(\mathbf{c}\cdot\mathbf{u})}\cdot\big(1-\sigma(\mathbf{c}\cdot\mathbf{u})\big)\cdot\mathbf{u}$$

$$= \big(\sigma(\mathbf{c}\cdot\mathbf{u})-1\big)\cdot\mathbf{u}$$

word2vec — skip-gram *with negative sampling*

Suppose we have a target word u, a valid context word c,
and k noise words h_i , $i \in \{1, \ldots, k\}$ (negative samples) chosen randomly

$$L_{\text{ce}} = -\left[\log\big(\sigma(\mathbf{c} \cdot \mathbf{u})\big) + \sum_{i=1}^{k} \log\big(\sigma(-\mathbf{h}_i \cdot \mathbf{u})\big)\right]$$

word2vec — skip-gram *with negative sampling*

Suppose we have a target word u, a valid context word c,
and k noise words $h_i\,, i \in \{1,\ldots,k\}$ (negative samples) chosen randomly

$$L_{\text{ce}} = -\left[\log\big(\sigma(\mathbf{c}\cdot\mathbf{u})\big) + \sum_{i=1}^{k}\log\big(\sigma(-\mathbf{h}_i\cdot\mathbf{u})\big)\right]$$

$$\frac{\partial L_{\text{ce}}}{\partial \mathbf{h}_i} = \;?$$

word2vec — skip-gram *with negative sampling*

Suppose we have a target word u, a valid context word c,
and k noise words h_i , $i \in \{1, \ldots, k\}$ (negative samples) chosen randomly

$$L_{\text{ce}} = -\left[\log\big(\sigma(\mathbf{c} \cdot \mathbf{u})\big) + \sum_{i=1}^{k} \log\big(\sigma(-\mathbf{h}_i \cdot \mathbf{u})\big)\right]$$

$$\frac{\partial L_{\text{ce}}}{\partial \mathbf{h}_i} = ? = \sigma(\mathbf{h}_i \cdot \mathbf{u}) \cdot \mathbf{u}$$

word2vec — skip-gram *with negative sampling*

Suppose we have a target word u, a valid context word c,
and k noise words $h_i\,, i \in \{1, \ldots, k\}$ (negative samples) chosen randomly

$$L_{\text{ce}} = -\left[\log\left(\sigma(\mathbf{c}\cdot\mathbf{u})\right) + \sum_{i=1}^{k}\log\left(\sigma(-\mathbf{h}_i\cdot\mathbf{u})\right)\right]$$

$$\frac{\partial L_{\text{ce}}}{\partial \mathbf{h}_i} = \,?\, = \sigma(\mathbf{h}_i\cdot\mathbf{u})\cdot\mathbf{u}$$

$$\frac{\partial L_{\text{ce}}}{\partial \mathbf{u}} = \left(\sigma(\mathbf{c}\cdot\mathbf{u}) - 1\right)\mathbf{c} + \sum_{i=1}^{k}\left(\sigma(\mathbf{h}_i\cdot\mathbf{u})\cdot\mathbf{h}_i\right)$$

word2vec — skip-gram *with negative sampling*

Suppose we have a target word u, a valid context word c,
and k noise words $h_i\,, i \in \{1,\ldots,k\}$ (negative samples) chosen randomly

$$L_{\text{ce}} = -\left[\log\big(\sigma(\mathbf{c}\cdot\mathbf{u})\big) + \sum_{i=1}^{k}\log\big(\sigma(-\mathbf{h}_i\cdot\mathbf{u})\big)\right]$$

rows of the context embedding matrix $\mathbf{C}$

$$\mathbf{c}_{t+1} = \mathbf{c}_t - \alpha\left(\frac{\partial L_{\text{ce}}}{\partial \mathbf{c}}\right)_t = \mathbf{c}_t - \alpha\,\big(\sigma(\mathbf{c}_t\cdot\mathbf{u}_t) - 1\big)\cdot\mathbf{u}_t$$

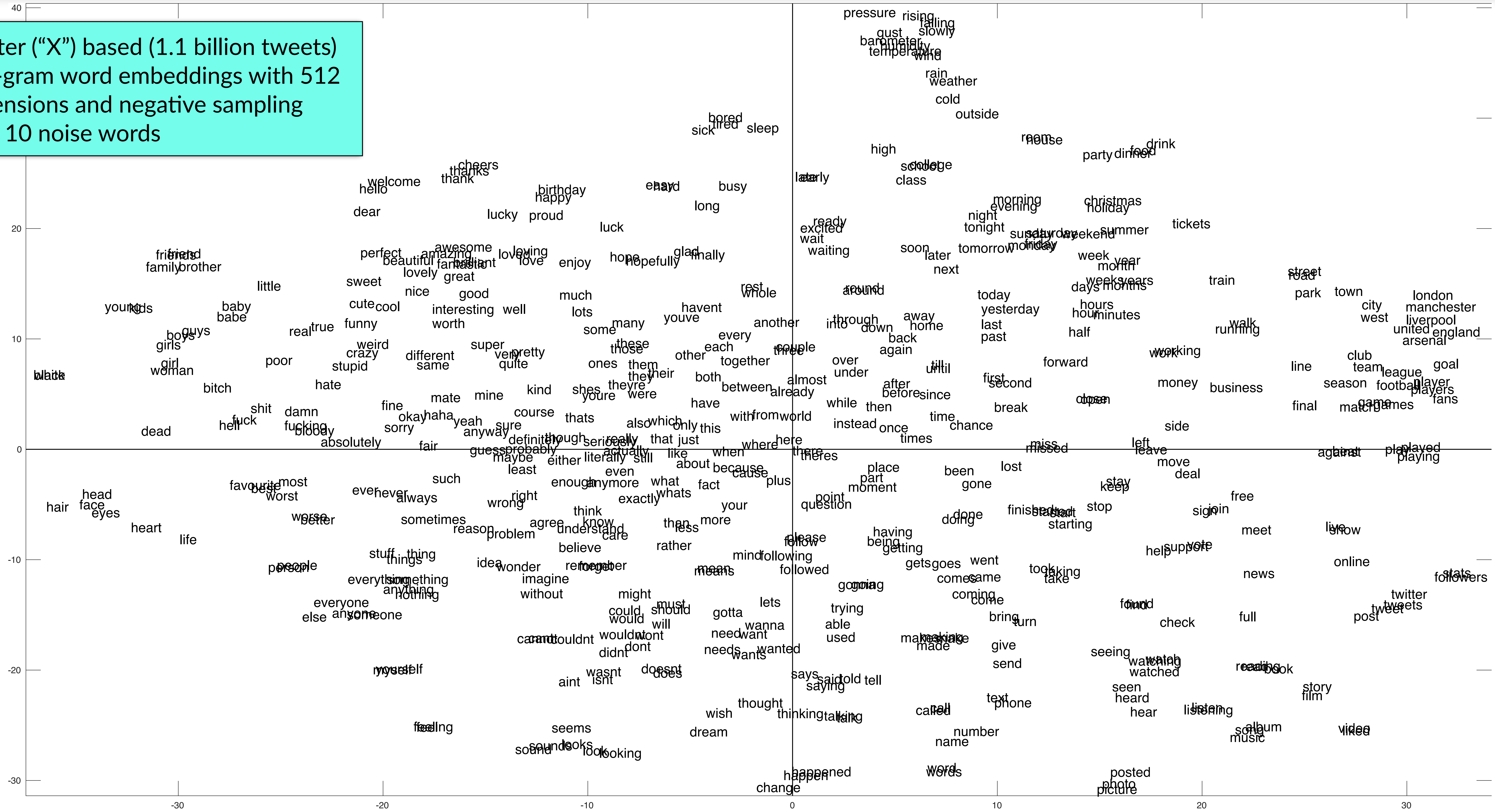
gradient descent with learning rate α

$$\mathbf{h}_{i;\,t+1} = \mathbf{h}_{i;\,t} - \alpha\,\sigma(\mathbf{h}_{i;\,t}\cdot\mathbf{u}_t)\,\mathbf{u}_t$$

$$\mathbf{u}_{t+1} = \mathbf{u}_t - \alpha\left[\big(\sigma(\mathbf{c}_t\cdot\mathbf{u}_t) - 1\big)\,\mathbf{c}_t + \sum_{i=1}^{k}\big(\sigma(\mathbf{h}_{i;\,t}\cdot\mathbf{u}_t)\cdot\mathbf{h}_{i;\,t}\big)\right]$$

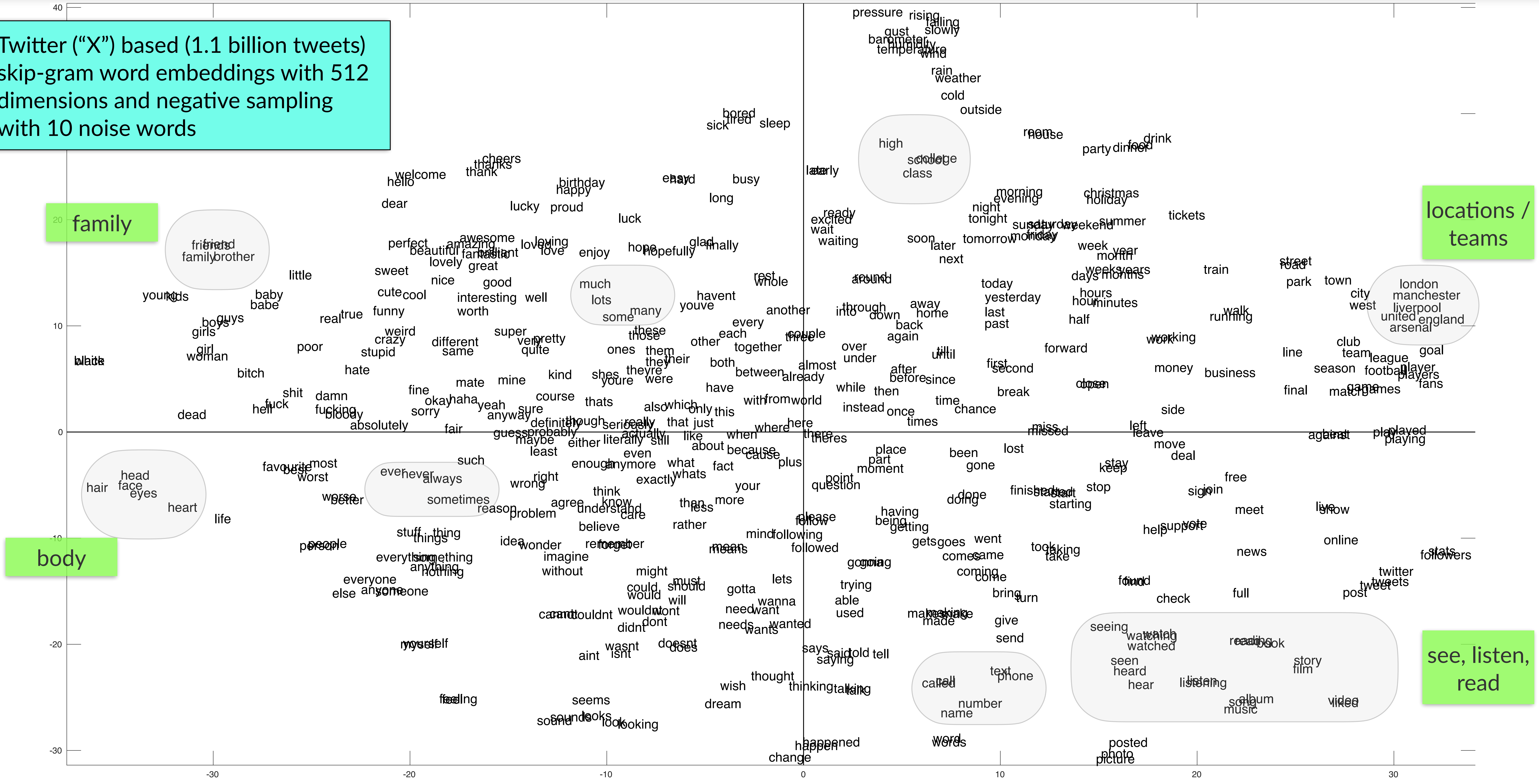
word2vec 2D projections

Twitter ("X") based (1.1 billion tweets) skip-gram word embeddings with 512 dimensions and negative sampling with 10 noise words



word2vec 2D projections

Twitter ("X") based (1.1 billion tweets) skip-gram word embeddings with 512 dimensions and negative sampling with 10 noise words



figshare.com/articles/dataset/UK_Twitter_word_embeddings_II_/5791650

Word analogies: The infamous "king — man + woman ≈ queen"

NB

Word embeddings tend to carry the biases or stereotypes of the corpora used to train them!

$$\text{vector}(‘\textbf{\textit{queen}}’) \approx \text{vector}(‘\textbf{\textit{king}}’) - \text{vector}(‘\textbf{\textit{man}}’) + \text{vector}(‘\textbf{\textit{woman}}’)$$

Word analogies: The infamous “king — man + woman ≈ queen”

NB

Word embeddings tend to carry the biases or stereotypes of the corpora used to train them!

$$\overbrace{\text{vector}(\text{‘}\textbf{\textit{queen}}\text{’})}^{b} \approx \overbrace{\text{vector}(\text{‘}\textbf{\textit{king}}\text{’})}^{a} - \overbrace{\text{vector}(\text{‘}\textbf{\textit{man}}\text{’})}^{a_p} + \overbrace{\text{vector}(\text{‘}\textbf{\textit{woman}}\text{’})}^{b_p}$$

Word analogies: The infamous “king — man + woman ≈ queen”

NB

Word embeddings tend to carry the biases or stereotypes of the corpora used to train them!

$$\underset{b}{\text{vector('}\textbf{\textit{queen}}\text{')}} \approx \underset{a}{\text{vector('}\textbf{\textit{king}}\text{')}} - \underset{a_p}{\text{vector('}\textbf{\textit{man}}\text{')}} + \underset{b_p}{\text{vector('}\textbf{\textit{woman}}\text{')}}$$

cosine similarity between ‘*queen*’ and ‘*king*’ - ‘*man*’ + ‘*woman*’

$$b = \arg\max_{b \in \mathscr{V}} \left(\cos \left(\mathbf{u}_b, \mathbf{u}_a - \mathbf{u}_{a_p} + \mathbf{u}_{b_p} \right) \right)$$

Compute cosine similarity between the composite embedding $\left(\mathbf{u}_a - \mathbf{u}_{a_p} + \mathbf{u}_{b_p} \right)$ and each other word embedding in our vocabulary; expect that $\mathbf{u}_b =$ vector(‘***queen***’) will have the greatest one.

Word analogies: The infamous “king — man + woman ≈ queen”

> **NB**
> Word embeddings tend to carry the biases or stereotypes of the corpora used to train them!

$$\underset{b}{\text{vector(‘}\textbf{\textit{queen}}\text{’)}} \approx \underset{a}{\text{vector(‘}\textbf{\textit{king}}\text{’)}} - \underset{a_p}{\text{vector(‘}\textbf{\textit{man}}\text{’)}} + \underset{b_p}{\text{vector(‘}\textbf{\textit{woman}}\text{’)}}$$

cosine similarity between ‘*queen*’ and ‘*king*’ - ‘*man*’ + ‘*woman*’

$$b = \arg\max_{b \in \mathscr{V}} \left(\cos\left(\mathbf{u}_b, \mathbf{u}_a - \mathbf{u}_{a_p} + \mathbf{u}_{b_p}\right)\right)$$

Compute cosine similarity between the composite embedding $\left(\mathbf{u}_a - \mathbf{u}_{a_p} + \mathbf{u}_{b_p}\right)$ and each other word embedding in our vocabulary; expect that $\mathbf{u}_b =$ vector(‘***queen***’) will have the greatest one.

This gives rise to the **word analogy**

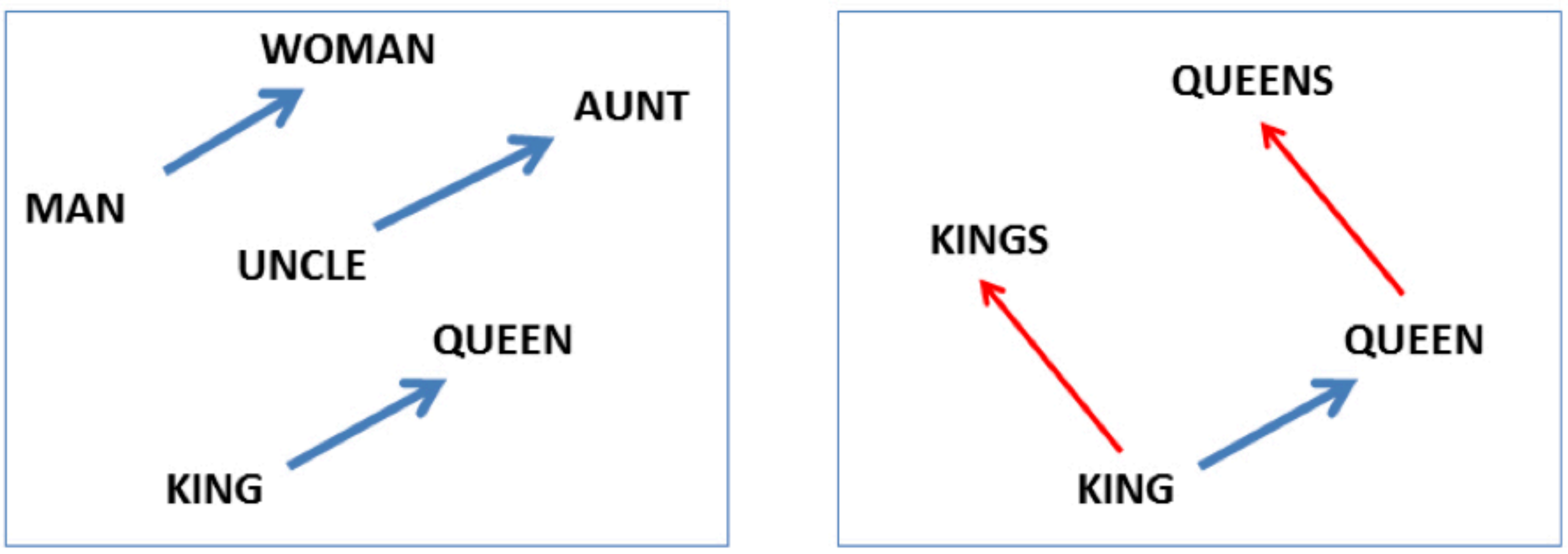
a_p is for a, what b_p is for b
or ‘*man*’ is for ‘*king*’, what ‘*woman*’ is for ‘*queen*’

Word analogies: The infamous “king — man + woman ≈ queen”

NB

Word embeddings tend to carry the biases or stereotypes of the corpora used to train them!

$$\underbrace{\text{vector('}\textbf{\textit{queen}}\text{')}}_{b} \approx \underbrace{\text{vector('}\textbf{\textit{king}}\text{')}}_{a} - \underbrace{\text{vector('}\textbf{\textit{man}}\text{')}}_{a_p} + \underbrace{\text{vector('}\textbf{\textit{woman}}\text{')}}_{b_p}$$



This gives rise to the **word analogy**

a_p is for a, what b_p is for b

or ‘*man*’ is for ‘*king*’, what ‘*woman*’ is for ‘*queen*’

Twitter word embeddings — Similarities

Top-5 most similar words using cosine similarity on word embeddings

- **Monday:** Tuesday, Thursday, Wednesday, Friday, Sunday
- **January:** February, August, October, March, June
- **red:** yellow, blue, purple, pink, green
- **we:** they, you, we've, our, us
- **espresso:** expresso, cappuccino, macchiato, latte, coffee
- **linux:** Unix, Centos, Debian, Ubuntu, Redhat
- **democracy:** democratic, dictatorship, democracies, socialism, undemocratic
- **loool:** looool, lool, loooool, looooool, loooooool
- **envi<u>rom</u>ent:** envi<u>ronm</u>ent, environments, env, enviro, habitats

Twitter word embeddings — Analogies

- **she** is to **her** what **he** is to ...

Twitter word embeddings — Analogies

- **she** is to **her** what **he** is to … [**his**, **him**, himself]

Twitter word embeddings — Analogies

- **she** is to **her** what **he** is to ... [**his**, **him**, himself]
- **Rome** is to **Italy** what **London** is to ... [**UK**, Denmark, Sweden]

Twitter word embeddings — Analogies

- **she** is to **her** what **he** is to ... [**his**, **him**, himself]
- **Rome** is to **Italy** what **London** is to ... [**UK**, Denmark, Sweden]
- **go** is for **went** what **do** is to... [**did**, doing, happened]

Twitter word embeddings — Analogies

- **she** is to **her** what **he** is to ... [**his**, **him**, himself]
- **Rome** is to **Italy** what **London** is to ... [**UK**, Denmark, Sweden]
- **go** is for **went** what **do** is to... [**did**, doing, happened]
- **big** is to **bigger** what **small** is to... [**smaller**, larger, tiny]

Twitter word embeddings — Analogies

- **she** is to **her** what **he** is to … [**his**, **him**, himself]
- **Rome** is to **Italy** what **London** is to … [**UK**, Denmark, Sweden]
- **go** is for **went** what **do** is to… [**did**, doing, happened]
- **big** is to **bigger** what **small** is to… [**smaller**, larger, tiny]
- **poet** is to **poem** what **author** is to… [**novel**, excerpt, memoir]

Evaluation of word embeddings

Intrinsic

- Easy, given, no need for additional effort
- Based on theoretical properties (linguistics), not always indicative of actual performance
- Word vector analogies (*seen in previous slides*)
- WordSim-353, SimLex-999
  word similarity by humans vs. trained word embeddings

Extrinsic

- Based on a downstream machine learning application (classification, regression)
- Not always easy or given $\Longrightarrow$ significant effort
- Is it the fault of the word embeddings or something else? Another sub-process that is failing, a task that is impossibly hard and so on. Requires an established, well-studied downstream task.

Evaluation of word embeddings

Intrinsic

- Easy, given, no need for additional effort
- Based on theoretical properties (linguistics), not always indicative of actual performance
- Word vector analogies (*seen in previous slides*)
- WordSim-353, SimLex-999
  word similarity by humans vs. trained word embeddings

most important???

Extrinsic

- Based on a downstream machine learning application (classification, regression)
- Not always easy or given $\implies$ significant effort
- Is it the fault of the word embeddings or something else? Another sub-process that is failing, a task that is impossibly hard and so on. Requires an established, well-studied downstream task.

Evaluation of word embeddings

Intrinsic

- Easy, given, no need for additional effort
- Based on theoretical properties (linguistics), not always indicative of actual performance
- Word vector analogies (*seen in previous slides*)
- WordSim-353, SimLex-999
  word similarity by humans vs. trained word embeddings

most important???

Extrinsic

- Based on a downstream machine learning application (classification, regression)
- Not always easy or given $\implies$ significant effort
- Is it the fault of the word embeddings or something else? Another sub-process that is failing, a task that is impossibly hard and so on. Requires an established, well-studied downstream task.

Other static word representation models

GloVe — aclanthology.org/D14-1162.pdf

- Global Vectors, uses ratios of probabilities from the word co-occurrence matrix
- not a neural network, bilinear model, scalable fast, not the best evaluation
- more optimisation functions?

$$\arg\min_{\mathbf{C},\mathbf{U}} \sum_{i \in \mathscr{V}} \sum_{j \in \mathscr{V}} f(x_{ij}) \left(\mathbf{c}_j^\top \mathbf{u}_i + \beta_i + \gamma_j - \log(x_{ij}) \right)^2$$

GloVe — aclanthology.org/D14-1162.pdf

- Global Vectors, uses ratios of probabilities from the word co-occurrence matrix
- not a neural network, bilinear model, scalable fast, not the best evaluation
- more optimisation functions?

$$\arg\min_{\mathbf{C},\mathbf{U}} \sum_{i\in\mathcal{V}} \sum_{j\in\mathcal{V}} f(x_{ij})\left(\mathbf{c}_j^\top \mathbf{u}_i + \beta_i + \gamma_j - \log(x_{ij})\right)^2$$

How do **word2vec** and GloVe deal with unknown words?

Other static word representation models

GloVe — aclanthology.org/D14-1162.pdf

- Global Vectors, uses ratios of probabilities from the word co-occurrence matrix
- not a neural network, bilinear model, scalable fast, not the best evaluation
- more optimisation functions?

$$\arg\min_{\mathbf{C},\mathbf{U}} \sum_{i \in \mathscr{V}} \sum_{j \in \mathscr{V}} f(x_{ij}) \left(\mathbf{c}_j^\top \mathbf{u}_i + \beta_i + \gamma_j - \log(x_{ij}) \right)^2$$

How do **word2vec** and GloVe deal with unknown words?

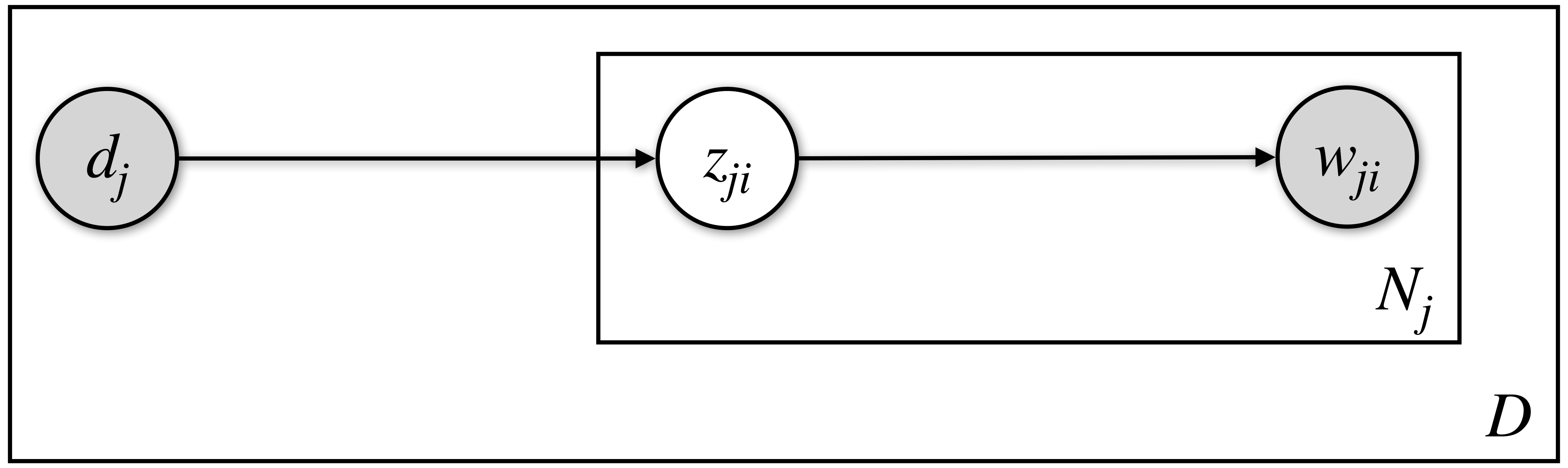
fasttext — aclanthology.org/Q17-1010.pdf

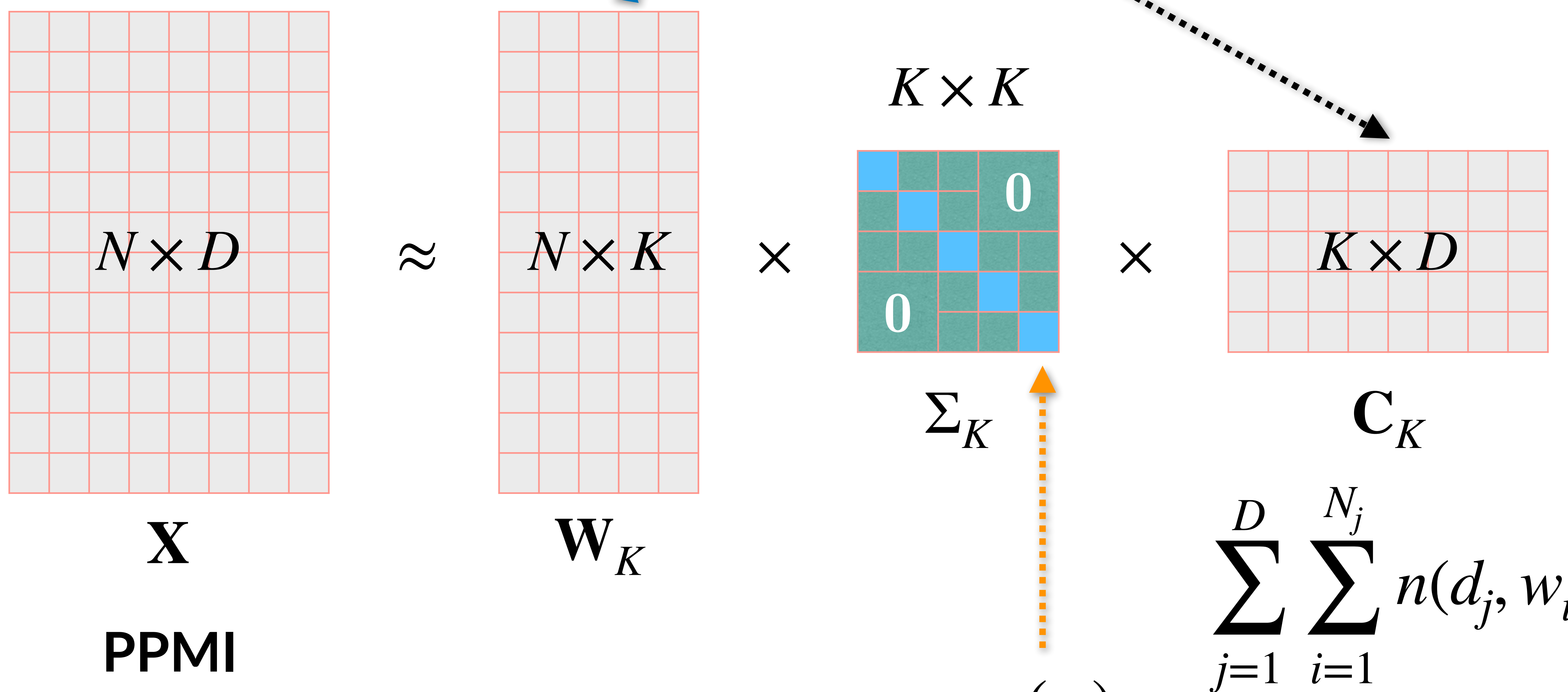
- deals with unknown words
- a word is represented by itself plus sub-word n-grams

  e.g. “steely” $\Longrightarrow$ <steely>, <st, ste, tee, eel, ely, ly> by setting n-gram length to 3
- < > special word boundaries to distinguish prefix / suffix, train with skip-gram
- known words are represented by the sum of all their sub-word embeddings
- unknown words are represented by the sum of embeddings of sub-words

Connection between SVD and topic models

$$p(\mathbf{d}, \mathbf{W}) = \prod_{j=1}^{D} p(d_j) \prod_{i=1}^{N_j} \sum_{k=1}^{K} \underline{p(z_{ji} = k \mid d_j)}\; \underline{p(w_{ji} \mid z_{ji} = k)}$$





$$p(z_k) = \frac{\sum_{j=1}^{D} \sum_{i=1}^{N_j} n(d_j, w_i)\, p(z_k \mid d_j, w_i)}{\sum_{j=1}^{D} \sum_{i=1}^{N_j} n(d_j, w_i)}$$

pLSA
Probabilistic
Latent
Semantic
Analysis

Next lecture *with me*

- Friday, February 2
- Recurrent Neural Networks (*for NLP*)