Word Embeddings

Boris Zubarev

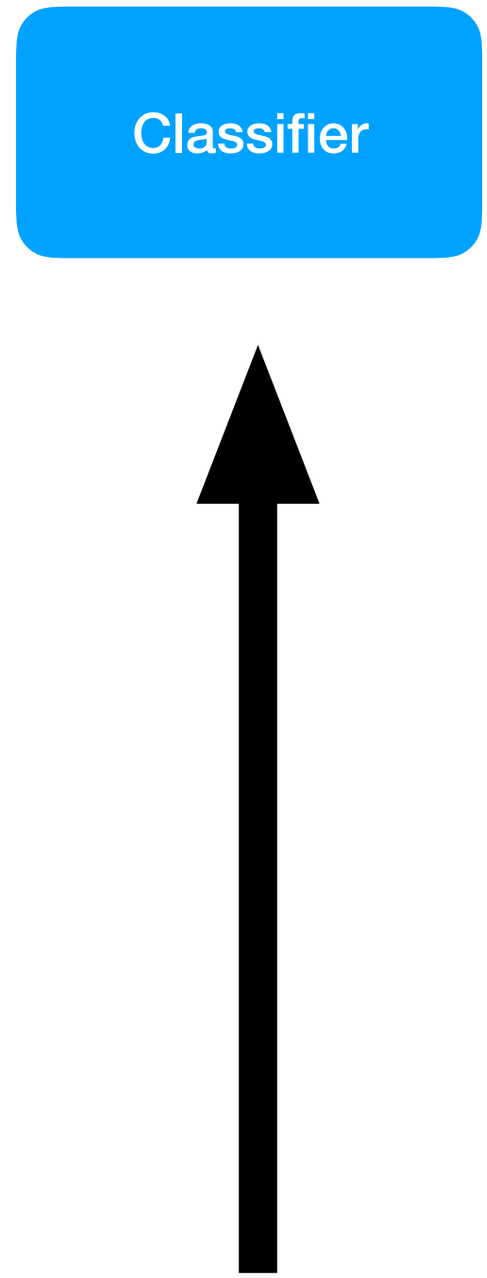
@bobazooba

Training Pipeline

Training Pipeline

The iPhone X is the huge leap forward

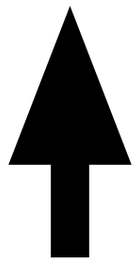
Training Pipeline



The iPhone X is the huge leap forward

Training Pipeline

Classifier



Featuring

The iPhone X is the huge leap forward

Training Pipeline

Classifier

Log Reg CNN

MLP Transformer

RNN etc

Featuring

The iPhone X is the huge leap forward

Training Pipeline

Classifier

Log Reg

MLP

RNN

CNN

Transformer

etc

Featuring

The iPhone X is the huge leap forward

Training Pipeline

Classifier

Log Reg

MLP

RNN

CNN

Transformer

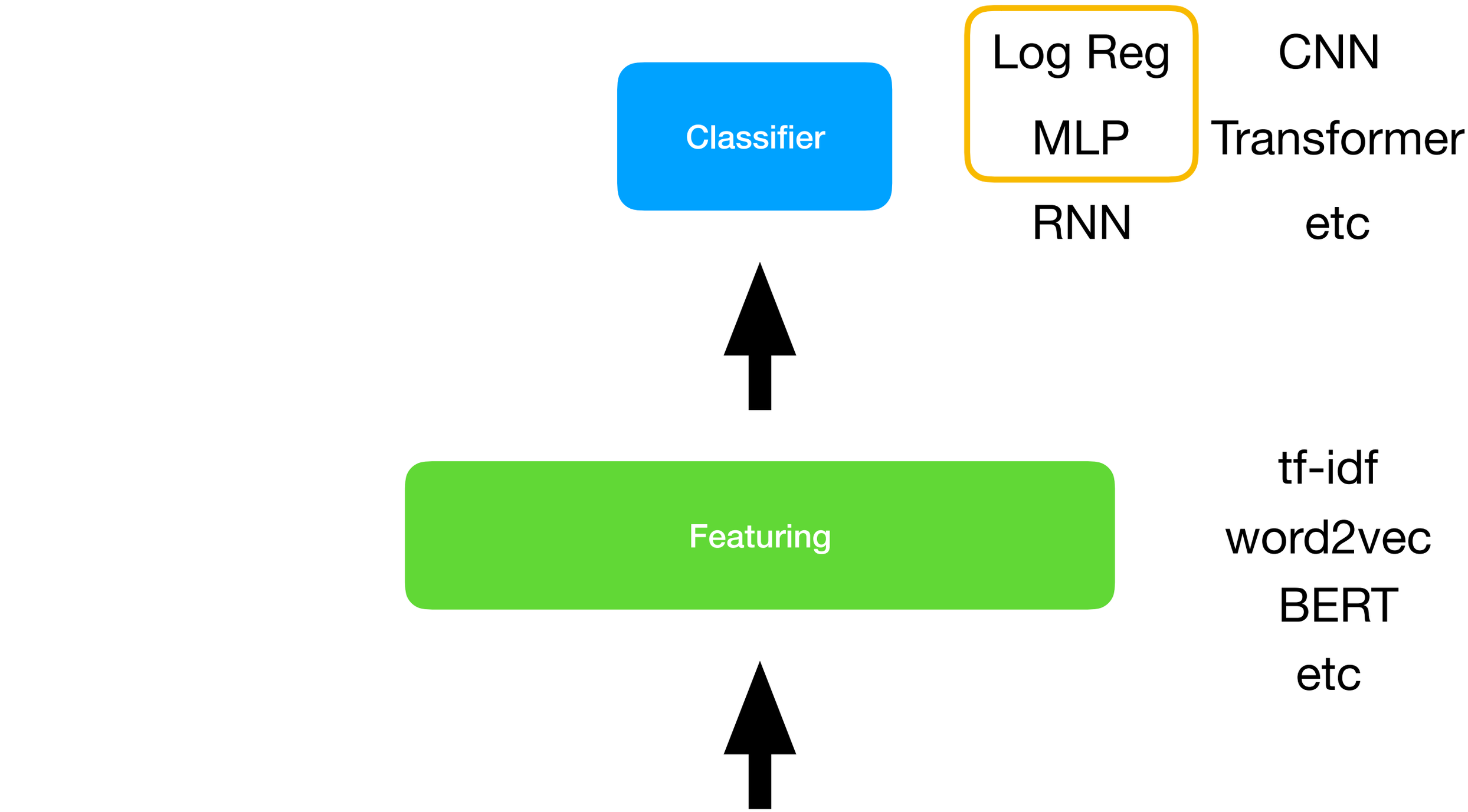
etc

Featuring

?

The iPhone X is the huge leap forward

Training Pipeline



Training Pipeline

Classifier

Log Reg
MLP
RNN

CNN
Transformer
etc

Featuring

tf-idf
word2vec
BERT
etc

The iPhone X is the huge leap forward

One Hot Encoding

Bag of words

motel = [0 0 0 0 0 0 0 0 0 0 1 0 0 0]

hotel = [0 0 0 0 0 0 1 0 0 0 0 0 0]

One Hot Encoding

Bag of words

motel = [0 0 0 0 0 0 0 0 0 0 1 0 0 0]

hotel = [0 0 0 0 0 0 1 0 0 0 0 0 0]

Ortogonal vectors

Dimension = len(vocabulary)

Similarity

Similarity

$$\text{similarity} = \cos(\theta) = \frac{\mathbf{A} \cdot \mathbf{B}}{\|\mathbf{A}\|\|\mathbf{B}\|} = \frac{\sum_{i=1}^{n} A_i B_i}{\sqrt{\sum_{i=1}^{n} A_i^2} \sqrt{\sum_{i=1}^{n} B_i^2}}$$

Similarity

Dot Product

$$\text{similarity} = \cos(\theta) = \frac{\mathbf{A} \cdot \mathbf{B}}{\|\mathbf{A}\|\|\mathbf{B}\|} = \frac{\sum_{i=1}^{n} A_i B_i}{\sqrt{\sum_{i=1}^{n} A_i^2} \sqrt{\sum_{i=1}^{n} B_i^2}}$$

Similarity

Dot Product

$$\text{similarity} = \cos(\theta) = \frac{\mathbf{A} \cdot \mathbf{B}}{\|\mathbf{A}\|\|\mathbf{B}\|} = \frac{\sum_{i=1}^{n} A_i B_i}{\sqrt{\sum_{i=1}^{n} A_i^2} \sqrt{\sum_{i=1}^{n} B_i^2}}$$

[0, 1]

Vector Norms

TF-IDF

$$w_{x,y} = tf_{x,y} \times \log\left(\frac{N}{df_x}\right)$$

TF-IDF

Term x within document y

$tf_{x,y}$ = frequency of x in y

df_x = number of documents containing x

N = total number of documents

TF-IDF

$$w_{x,y} = tf_{x,y} \times \log\left(\frac{N}{df_x}\right)$$

TF-IDF

Term *x* within document *y*

$tf_{x,y}$ = frequency of *x* in *y*

df_x = number of documents containing *x*

N = total number of documents

text1	0	0	0	0	0.47	0	0.23	0
text2	0	0.68	0	0	0.32	0	0	0
text3	0.11	0	0.19	0	0	0	0	0

Co-occurrence Vectors

«You shall know a word by the company it keeps» — Firth, 1957

Corpus sentences

He also found five fish swimming in murky water in an old **bathtub**.

We do abhor dust and dirt, and stains on the **bathtub**, and any kind of filth.

Above At the far end of the garden room a **bathtub** has been planted with herbs for the winter.

They had been drinking Cisco, a fruity, wine-based fluid that smells and tastes like a mixture of cough syrup and **bathtub** gin.

Science finds that a surface tension on the water can draw the boats together, like toy boats in a **bathtub**.

In fact, the godfather of gloom comes up with a plot that takes in Windsor Davies (the ghost of sitcoms past), a **bathtub** and a big box of concentrated jelly.

'I'll tell him,' said the Dean from the bathroom above the sound of bathwater falling from a great height into the ample Edwardian **bathtub**.

Co-occurrence counts

the	12
a	9
of	7
and	6
in	5
…	…
like	2
water	2
boat	2
from	2
stain	1
toy	1
god-father	1
Cisco	1
…	…

vector

$$\begin{pmatrix} 12 \\ 9 \\ 7 \\ 6 \\ 5 \\ \vdots \\ 2 \\ 2 \\ 2 \\ 2 \\ 1 \\ 1 \\ 1 \\ 1 \\ \vdots \end{pmatrix}$$

Co-occurrence Vectors

«You shall know a word by the company it keeps» — Firth, 1957

Corpus sentences

He also found five fish swimming in murky water in an old **bathtub**.

We do abhor dust and dirt, and stains on the **bathtub**, and any kind of filth.

Above At the far end of the garden room a **bathtub** has been planted with herbs for the winter.

They had been drinking Cisco, a fruity, wine-based fluid that smells and tastes like a mixture of cough syrup and **bathtub** gin.

Science finds that a surface tension on the water can draw the boats together, like toy boats in a **bathtub**.

In fact, the godfather of gloom comes up with a plot that takes in Windsor Davies (the ghost of sitcoms past), a **bathtub** and a big box of concentrated jelly.

'I'll tell him,' said the Dean from the bathroom above the sound of bathwater falling from a great height into the ample Edwardian **bathtub**.

Co-occurrence counts

the	12
a	9
of	7
and	6
in	5
…	…
like	2
water	2
boat	2
from	2
stain	1
toy	1
god-father	1
Cisco	1
…	…

vector

$$\begin{pmatrix} 12 \\ 9 \\ 7 \\ 6 \\ 5 \\ \vdots \\ 2 \\ 2 \\ 2 \\ 2 \\ 1 \\ 1 \\ 1 \\ 1 \\ \vdots \end{pmatrix}$$

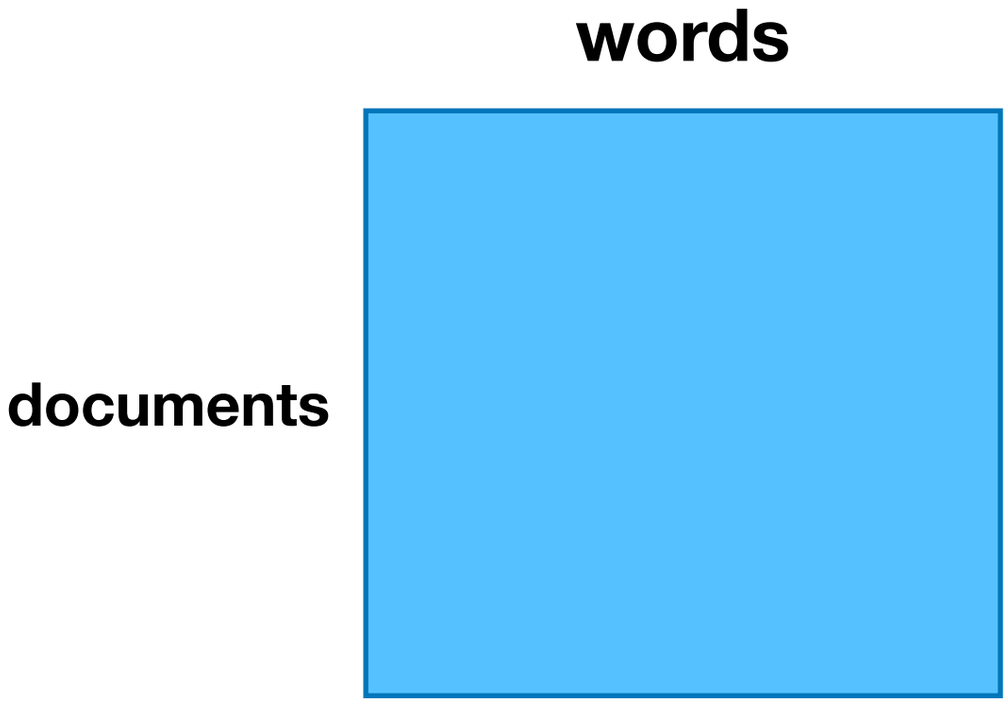
Dimensionality reduction

small vector

$$\begin{pmatrix} \vdots \\ \vdots \\ \vdots \\ \vdots \\ \vdots \\ \vdots \\ \vdots \\ \vdots \\ \vdots \\ \vdots \end{pmatrix}$$

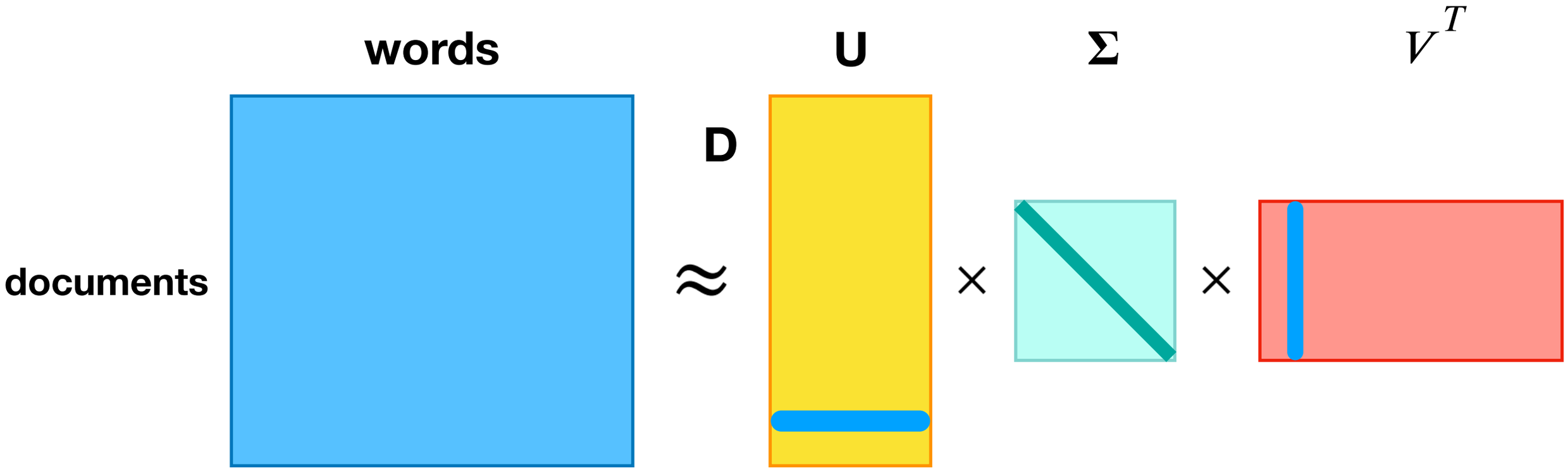
Co-occurrence Matrix

$$X \approx \hat{X} = U \Sigma V^T$$



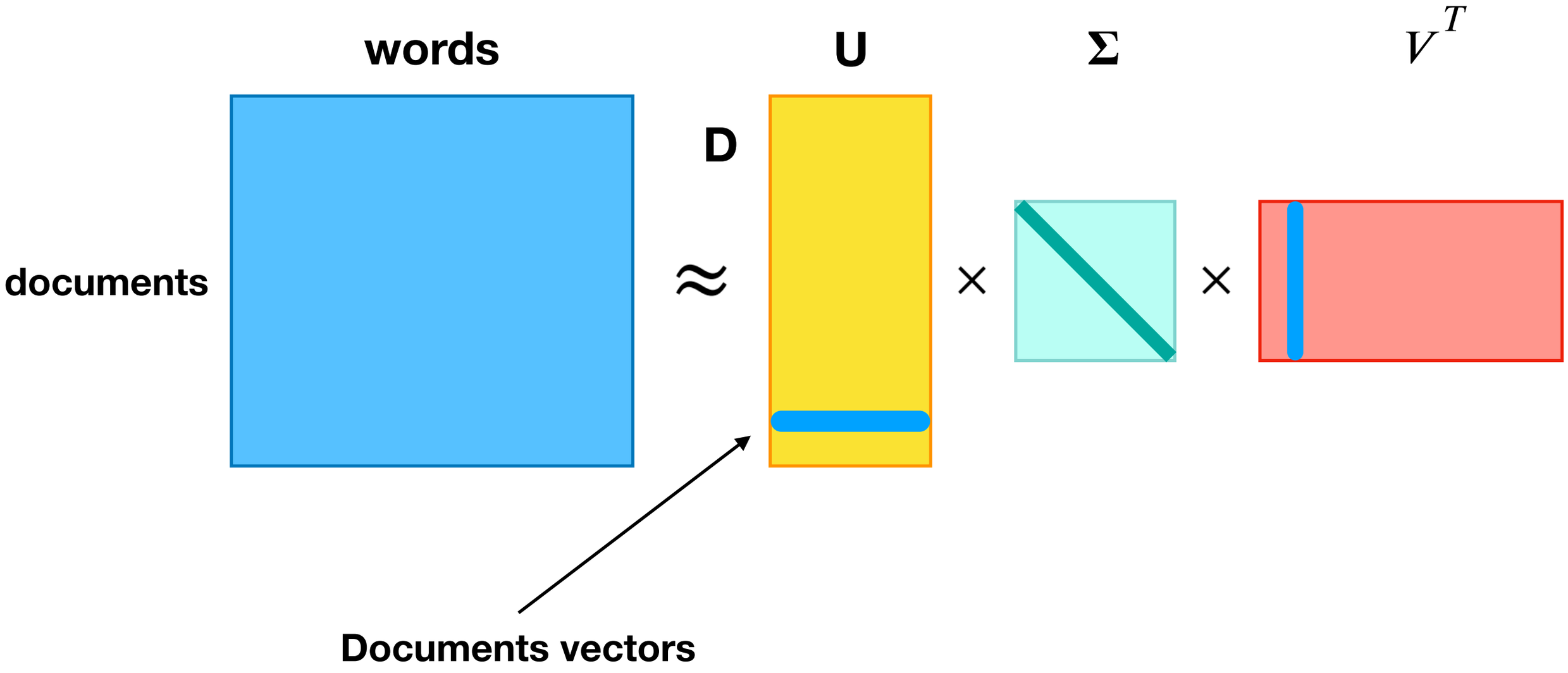
Co-occurrence Matrix

$$X \approx \hat{X} = U \Sigma V^T$$



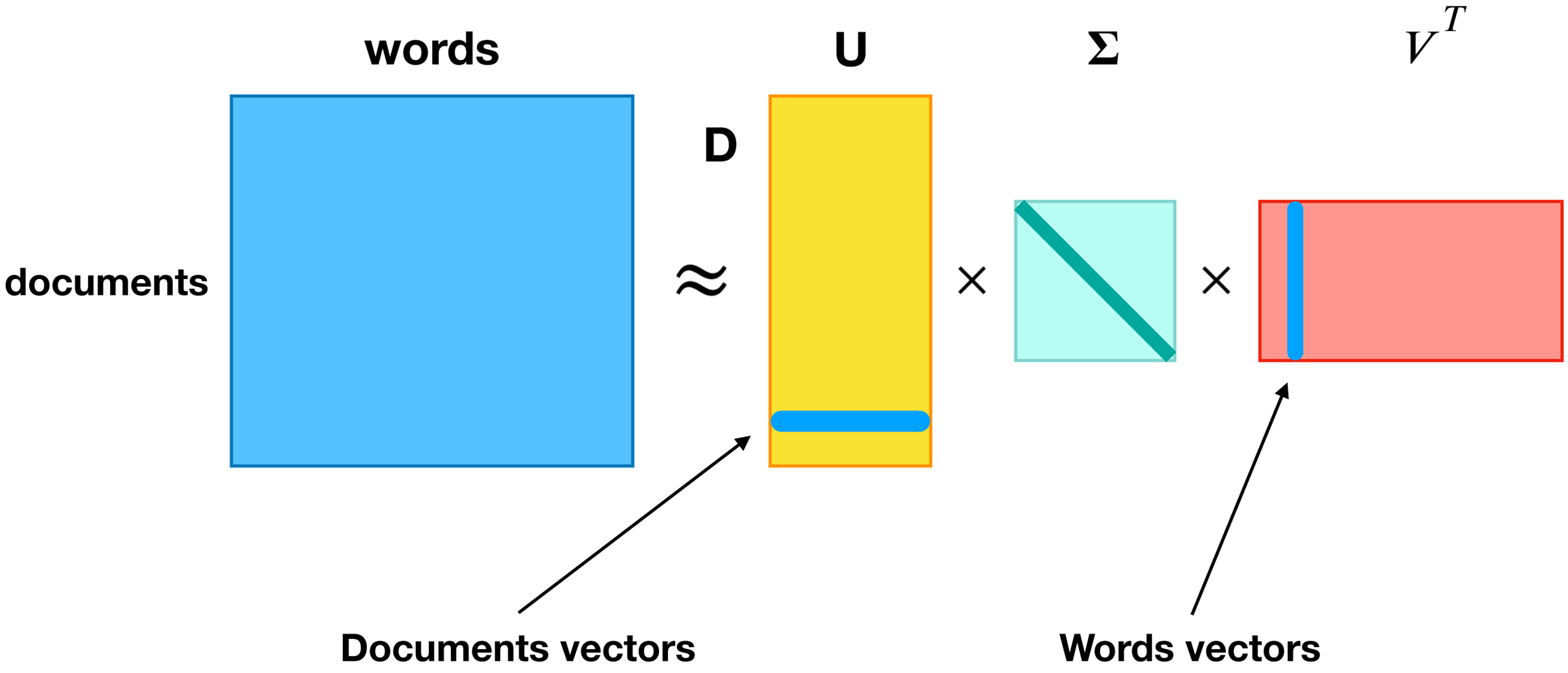
Co-occurrence Matrix

$$X \approx \hat{X} = U \Sigma V^T$$



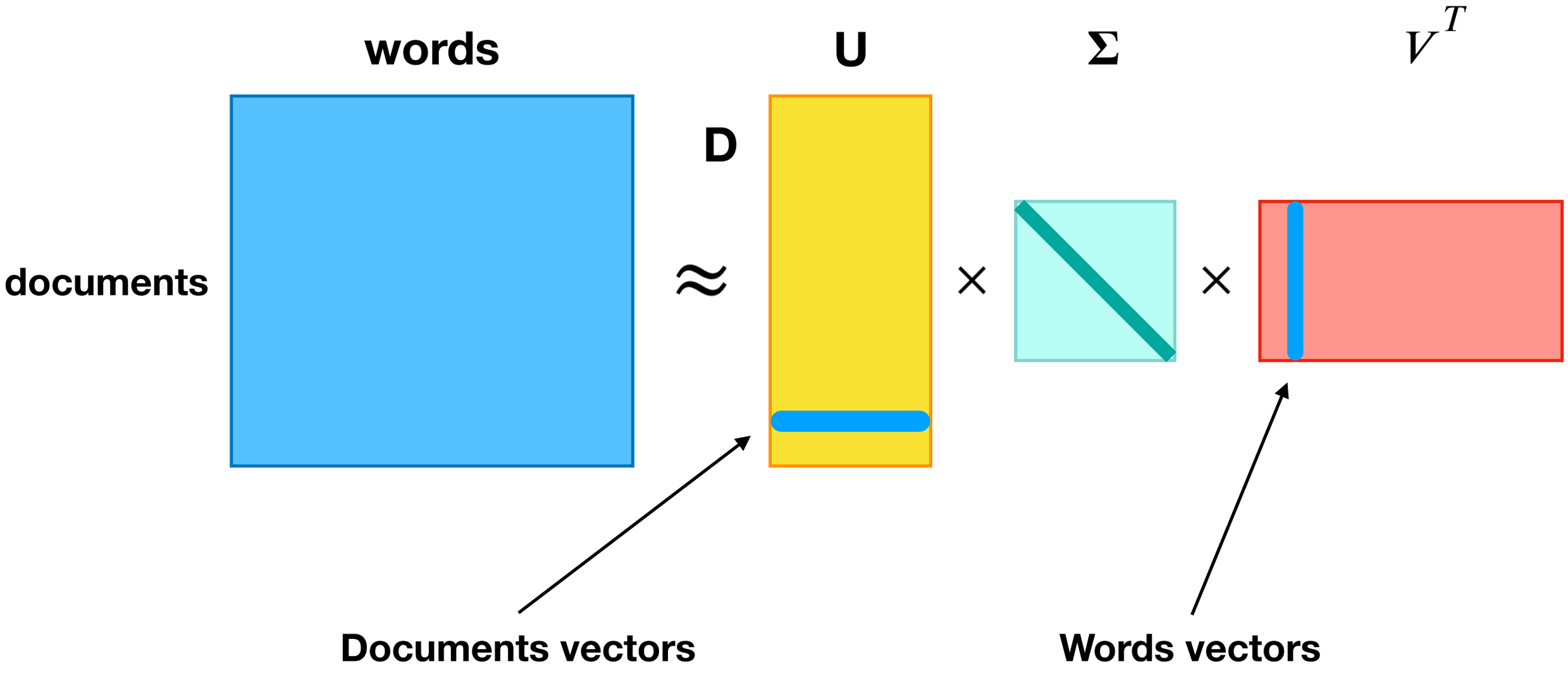
Co-occurrence Matrix

$$X \approx \hat{X} = U \Sigma V^T$$



Co-occurrence Matrix

$$X \approx \hat{X} = U \Sigma V^T$$



Computational expensive

Co-occurrence Matrix

words

words

№	Словосочетание	Документы	Частота
1	и не	22732	201352
2	и в	27048	193983
3	потому что	14926	117401
4	я не	10675	113767
5	у меня	9734	97102
6	может быть	16086	96065
7	то что	17195	95251
8	что он	11786	92743
9	не было	13196	92729
10	в том	21604	89842

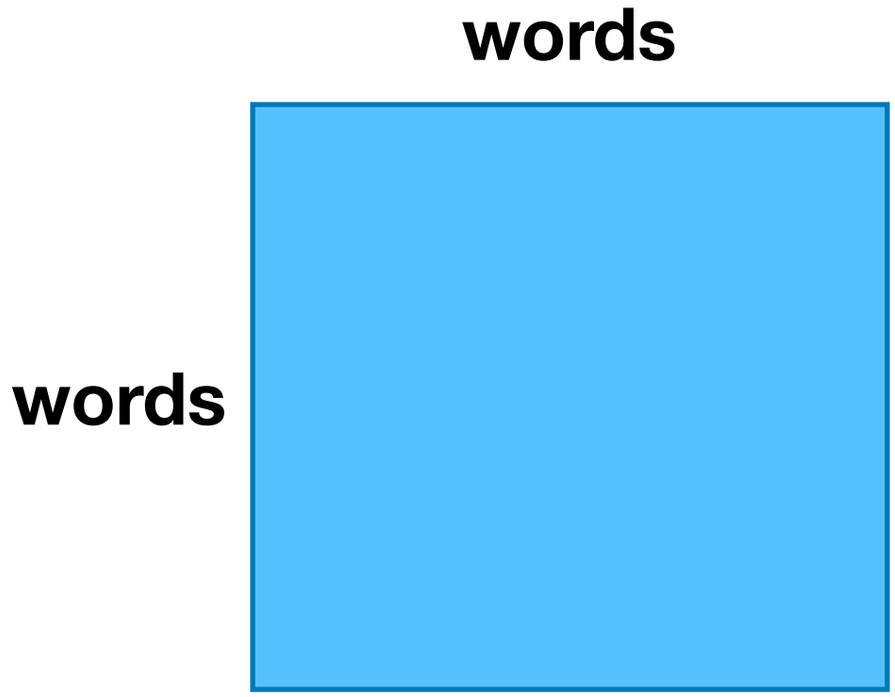
Co-occurrence Matrix

words

words

№	Словосочетание	Документы	Частота
1	о том что	11793	37235
2	в том что	12321	36961
3	до сих пор	8947	24284
4	и т д	6916	22828
5	для того чтобы	7661	19722
6	в том числе	9206	19309
7	в то время	6072	19037
8	в это время	4456	16541
9	в то же	6130	16088

Co-occurrence Matrix

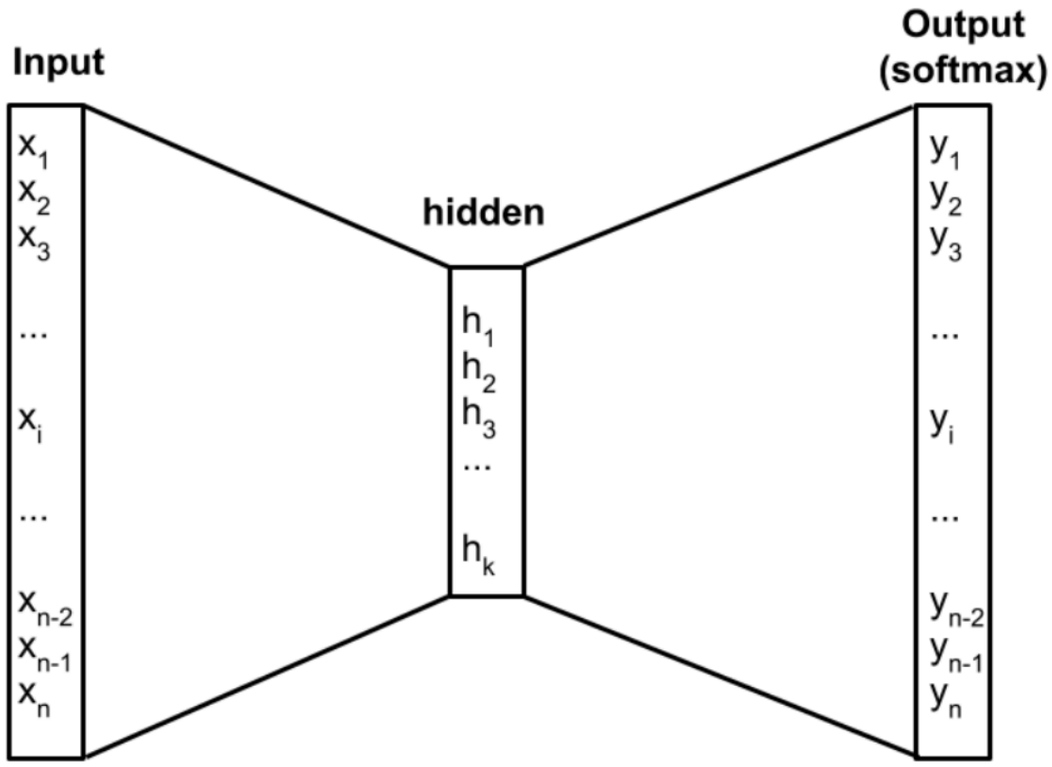


$$\text{pmi}(x;y) \equiv \log \frac{p(x,y)}{p(x)p(y)} = \log \frac{p(x|y)}{p(x)} = \log \frac{p(y|x)}{p(y)}$$

ppmi = max(pmi, 0)

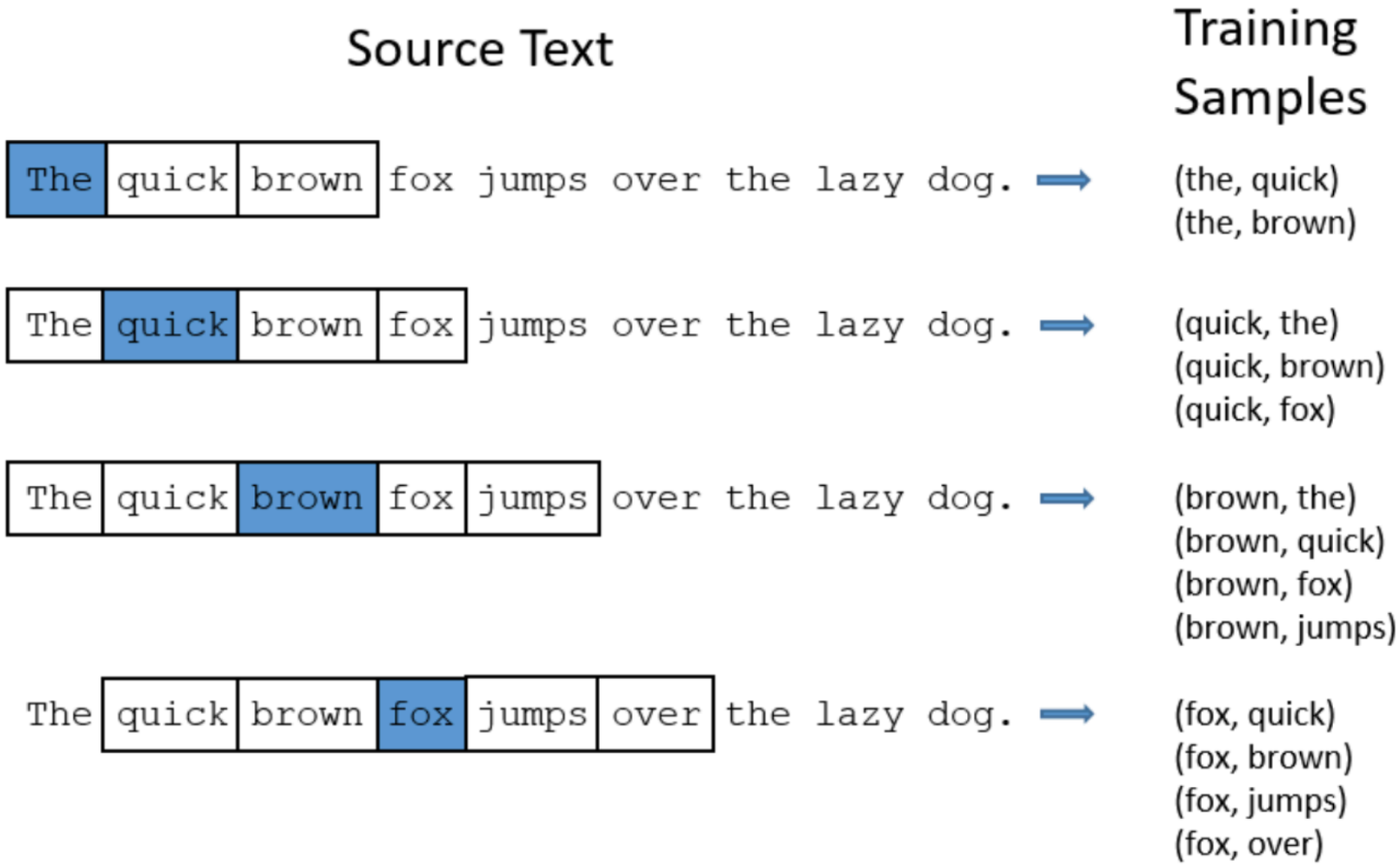
Word2Vec

Word2Vec

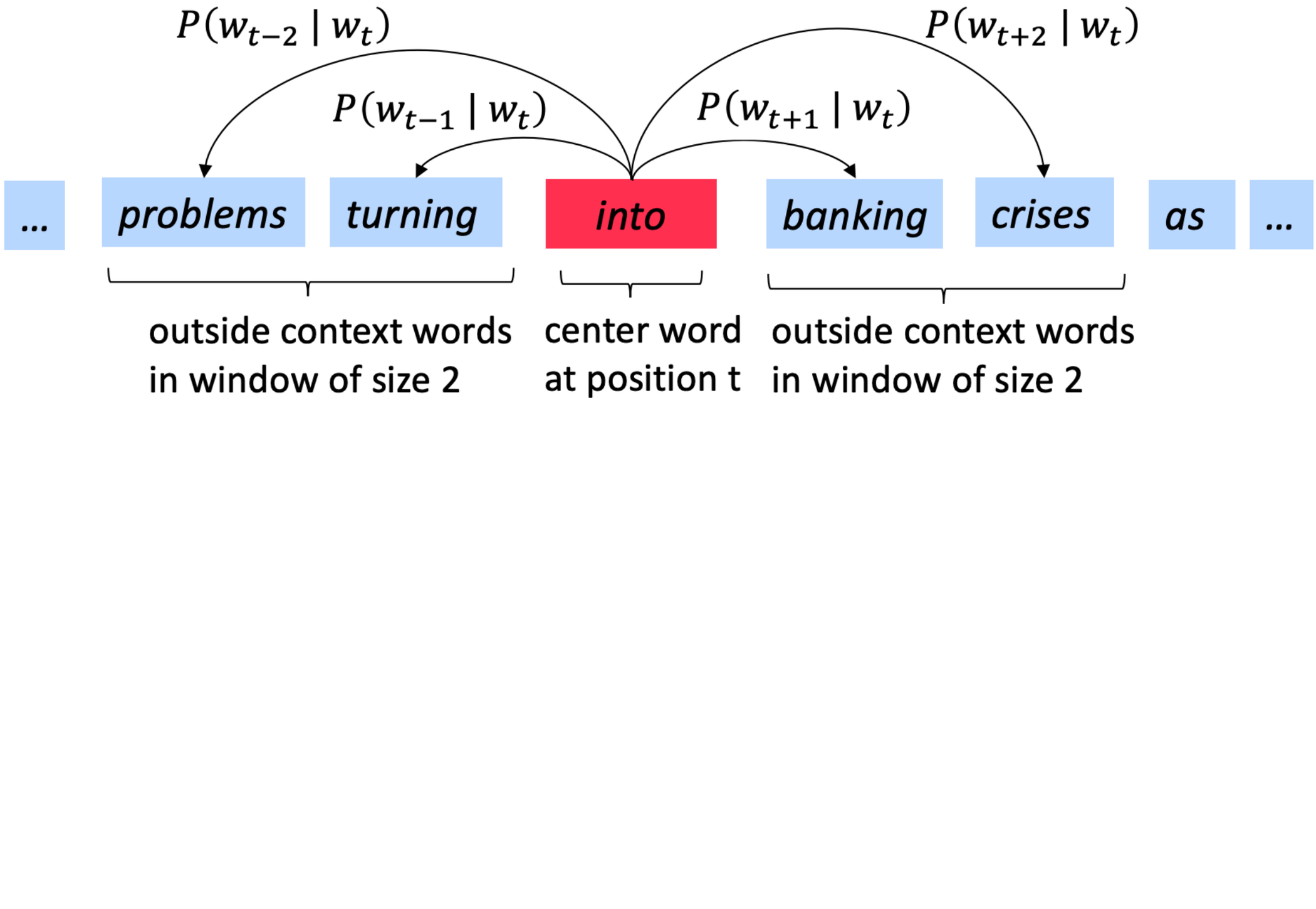


$$\frac{1}{T}\sum_{t=1}^{T}\sum_{-c\leq j\leq c, j\neq 0}\log p(w_{t+j}|w_t)$$

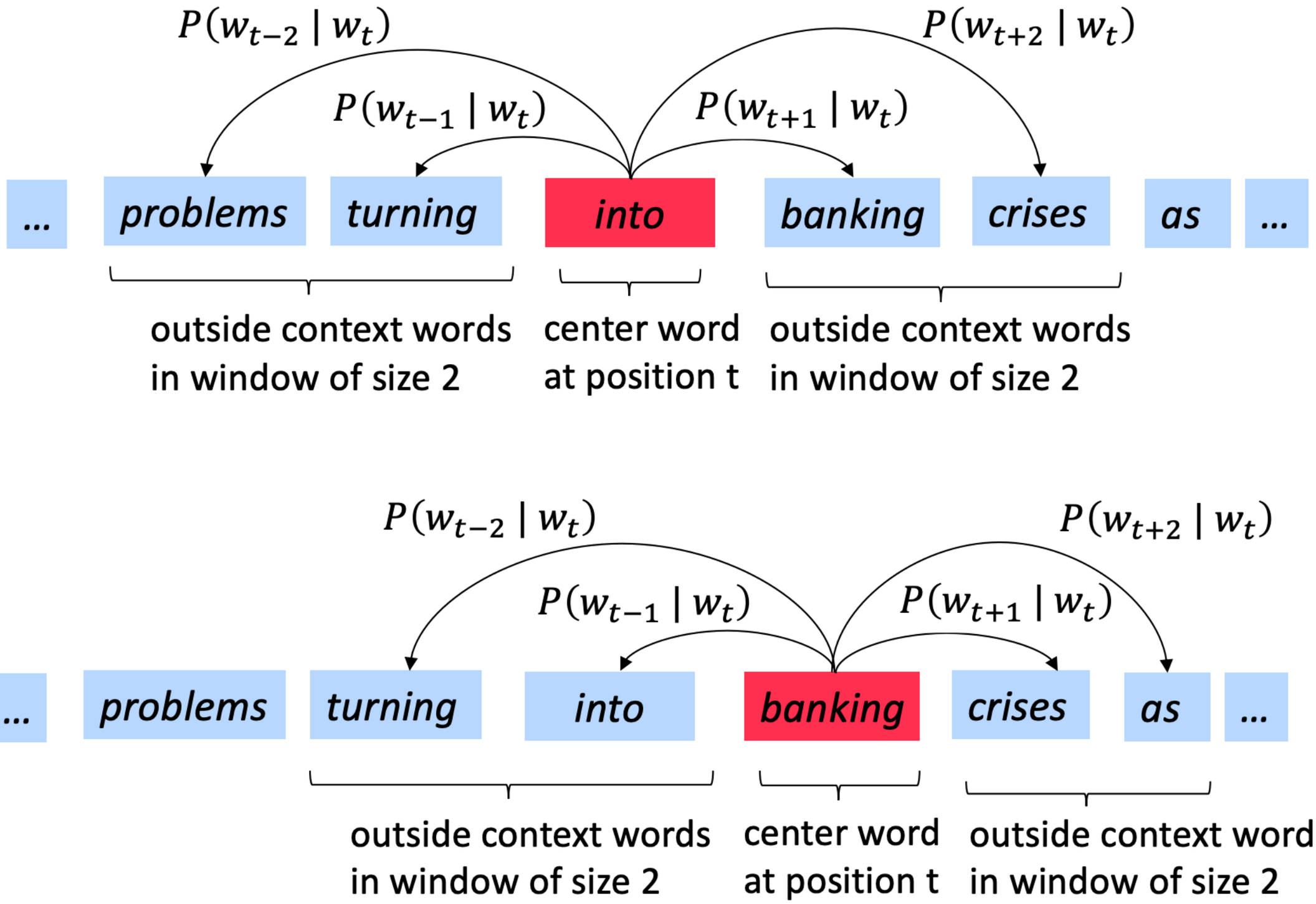
Word2Vec


Source Text
Training Samples
The quick brown fox jumps over the lazy dog.
(the, quick)
(the, brown)
The quick brown fox jumps over the lazy dog.
(quick, the)
(quick, brown)
(quick, fox)
The quick brown fox jumps over the lazy dog.
(brown, the)
(brown, quick)
(brown, fox)
(brown, jumps)
The quick brown fox jumps over the lazy dog.
(fox, quick)
(fox, brown)
(fox, jumps)
(fox, over)

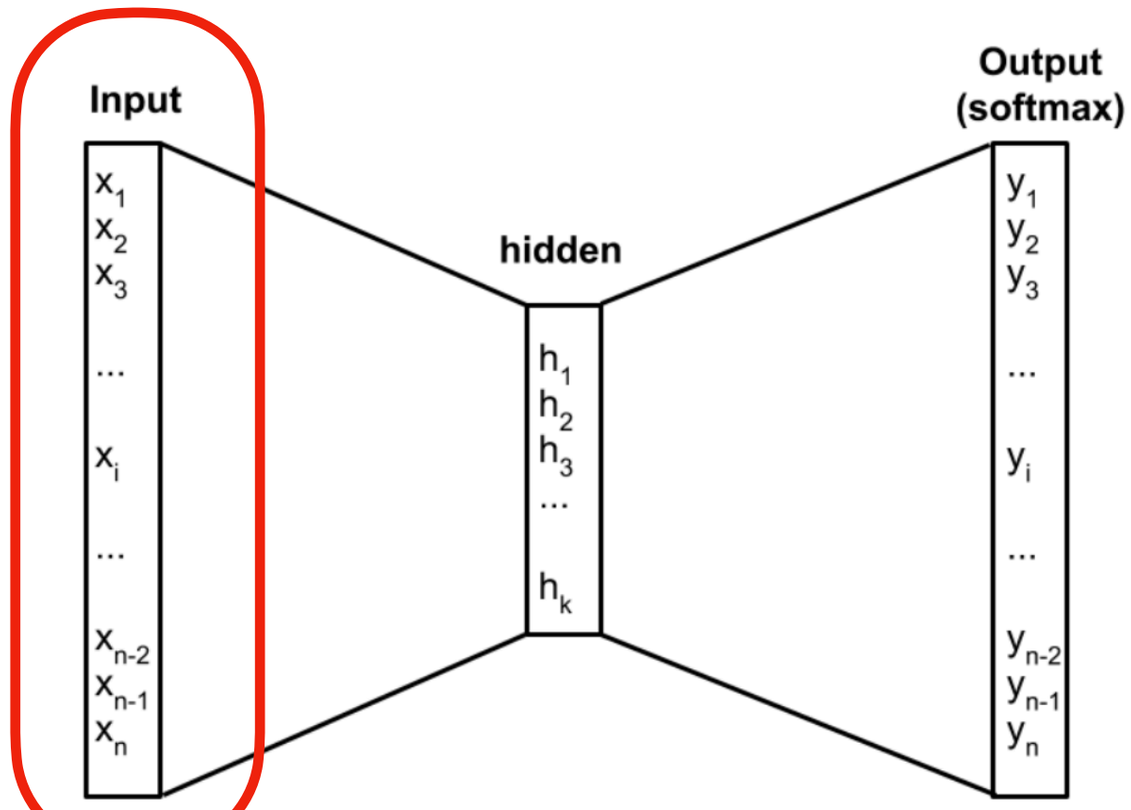
Word2Vec


$P(w_{t-2} \mid w_t)$
$P(w_{t-1} \mid w_t)$
$P(w_{t+1} \mid w_t)$
$P(w_{t+2} \mid w_t)$
...
problems
turning
into
banking
crises
as
...
outside context words in window of size 2
center word at position t
outside context words in window of size 2

Word2Vec

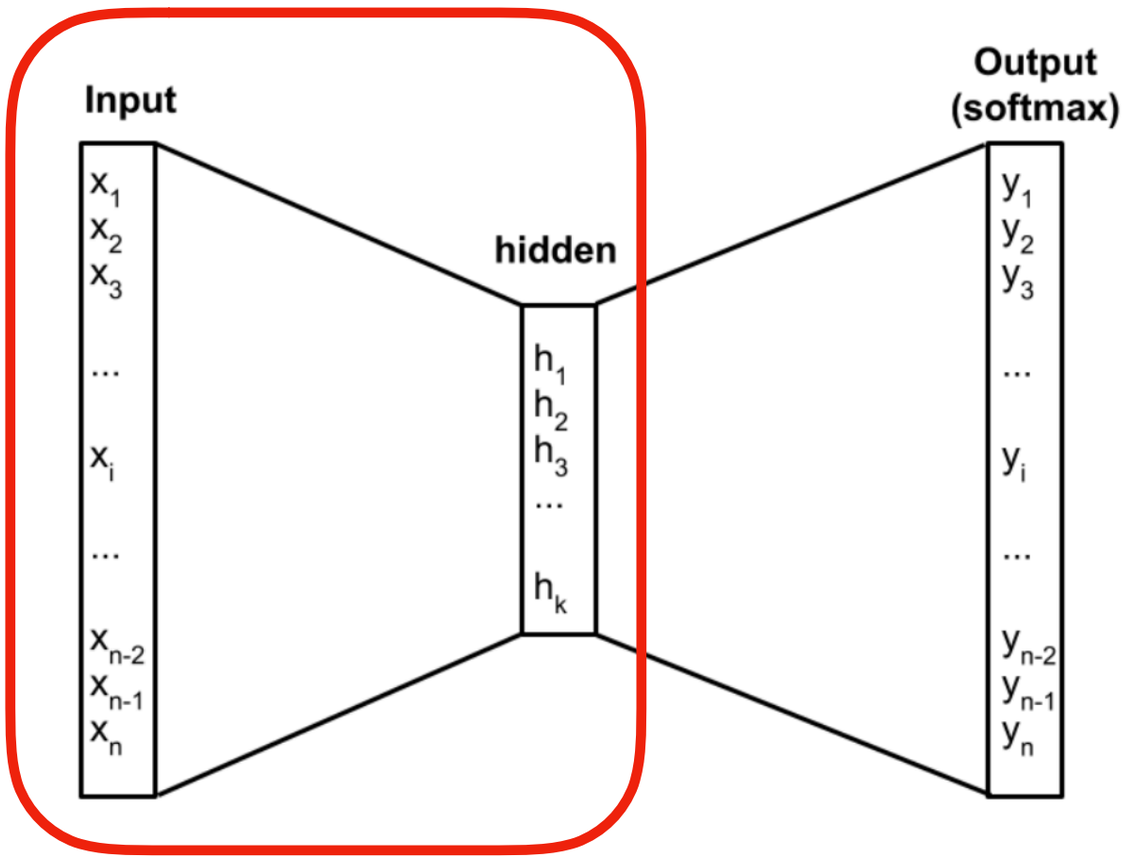

$P(w_{t-2} \mid w_t)$
$P(w_{t-1} \mid w_t)$
$P(w_{t+1} \mid w_t)$
$P(w_{t+2} \mid w_t)$
...
problems
turning
into
banking
crises
as
...
outside context words in window of size 2
center word at position t
outside context words in window of size 2
$P(w_{t-2} \mid w_t)$
$P(w_{t-1} \mid w_t)$
$P(w_{t+1} \mid w_t)$
$P(w_{t+2} \mid w_t)$
...
problems
turning
into
banking
crises
as
...
outside context words in window of size 2
center word at position t
outside context words in window of size 2

Word2Vec



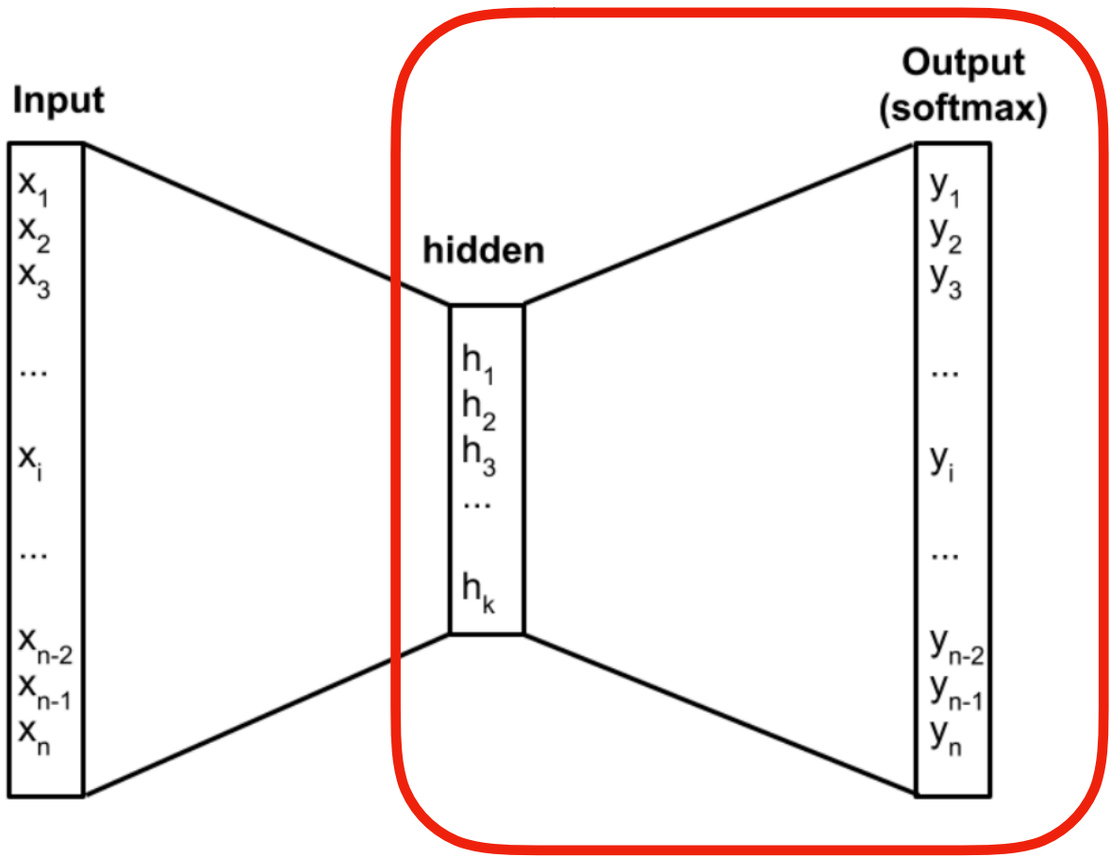
$$\theta = \begin{bmatrix} v_{aardvark} \\ v_a \\ \vdots \\ v_{zebra} \\ u_{aardvark} \\ u_a \\ \vdots \\ u_{zebra} \end{bmatrix} \in \mathbb{R}^{2dV}$$

Word2Vec



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

Word2Vec



Output weights for “car”

Word vector for “ants”

300 features

×

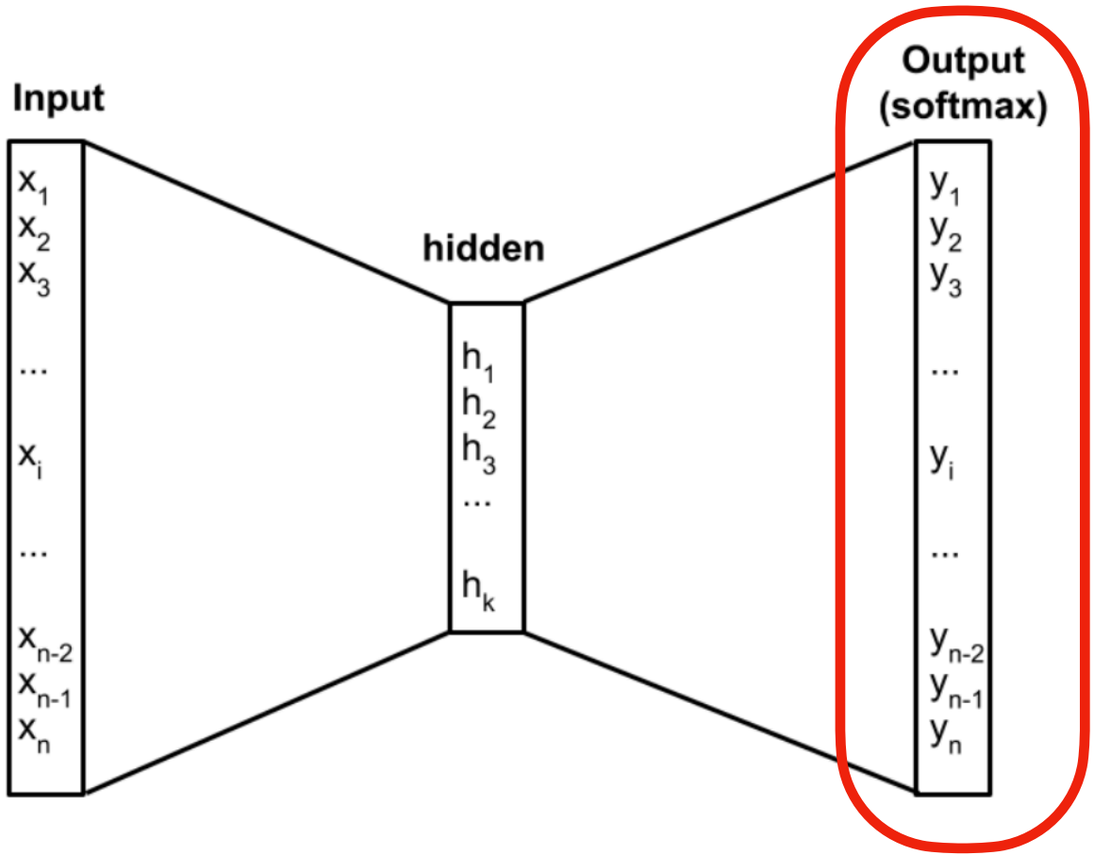
300 features

softmax

$$\frac{e^x}{\sum e^x}$$

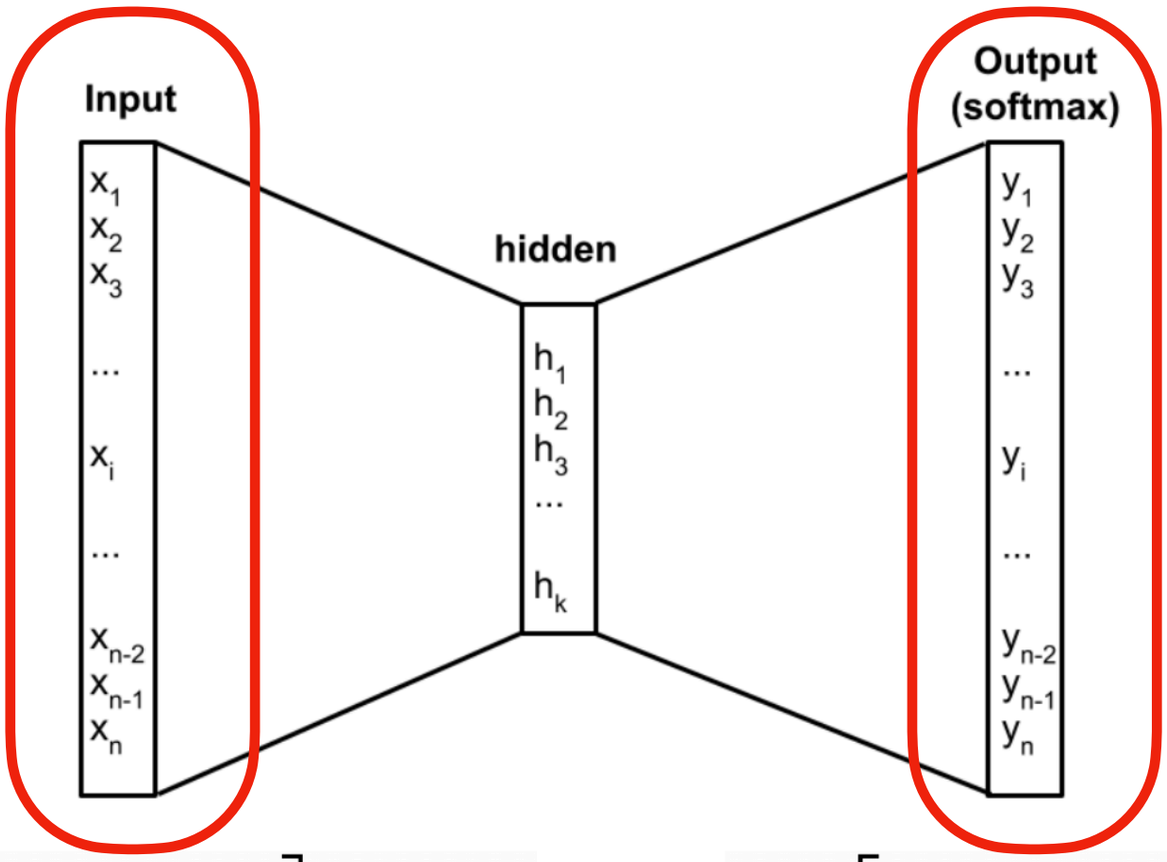
= Probability that if you randomly pick a word nearby “ants”, that it is “car”

Word2Vec



$$p(w_O|w_I) = \frac{\exp\left({v'_{w_O}}^\top v_{w_I}\right)}{\sum_{w=1}^{W}\exp\left({v'_w}^\top v_{w_I}\right)}$$

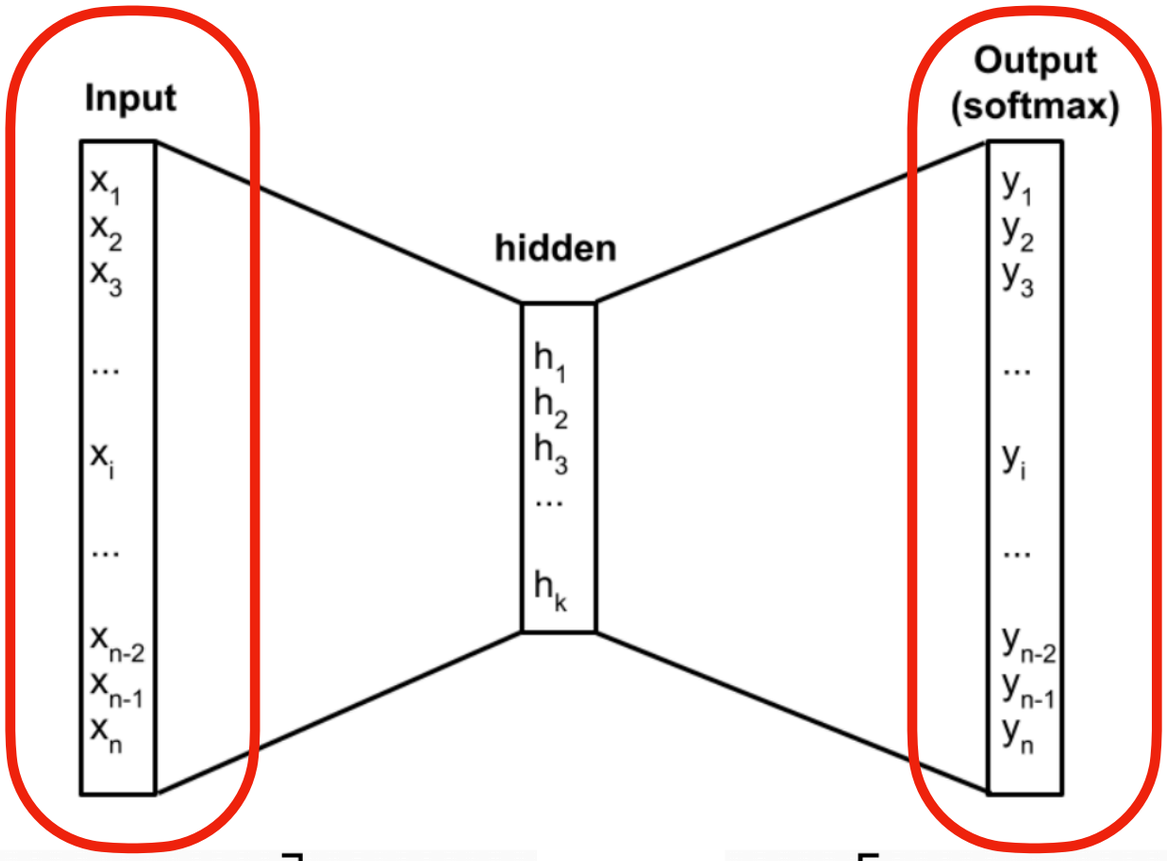
Word2Vec



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

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

Word2Vec



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

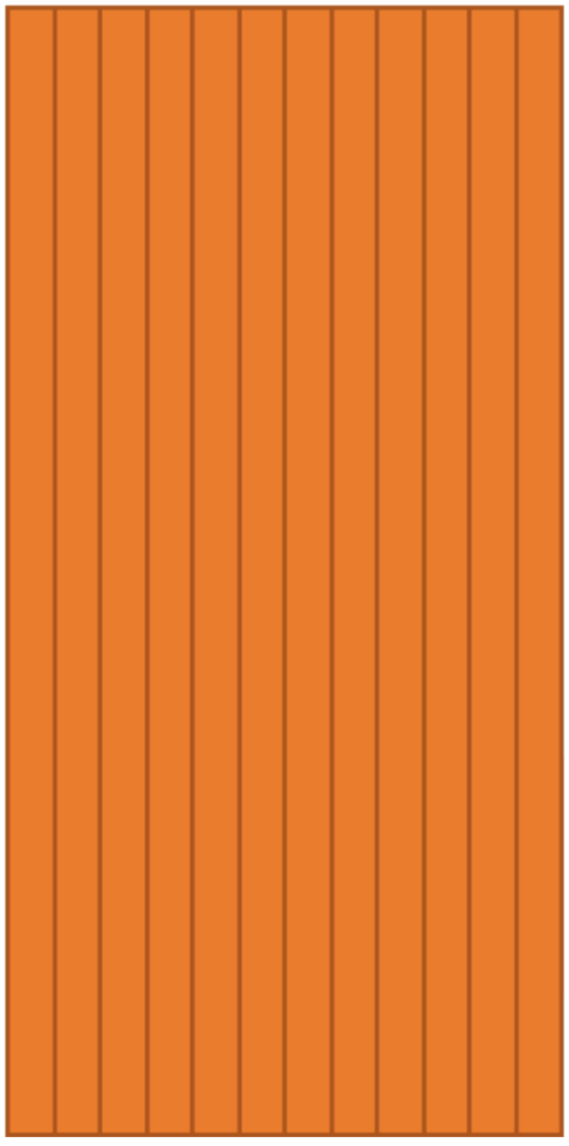
Same words

Different vectors

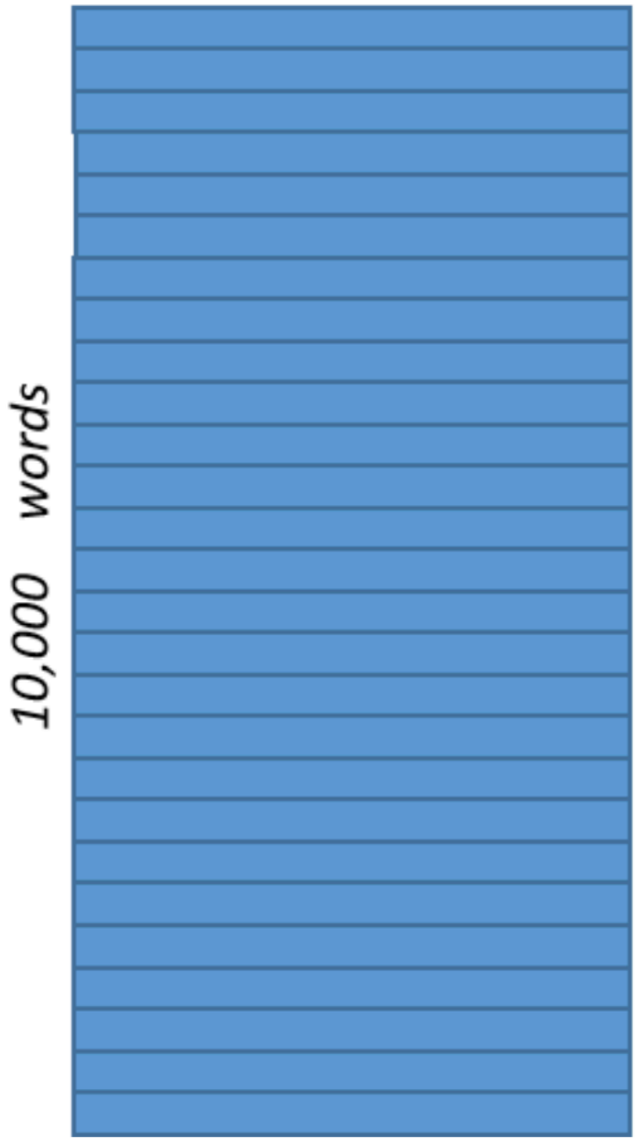
$$\theta = \begin{bmatrix} v_{aardvark} \\ v_a \\ \vdots \\ v_{zebra} \\ u_{aardvark} \\ u_a \\ \vdots \\ u_{zebra} \end{bmatrix} \in \mathbb{R}^{2dV}$$

Word2Vec

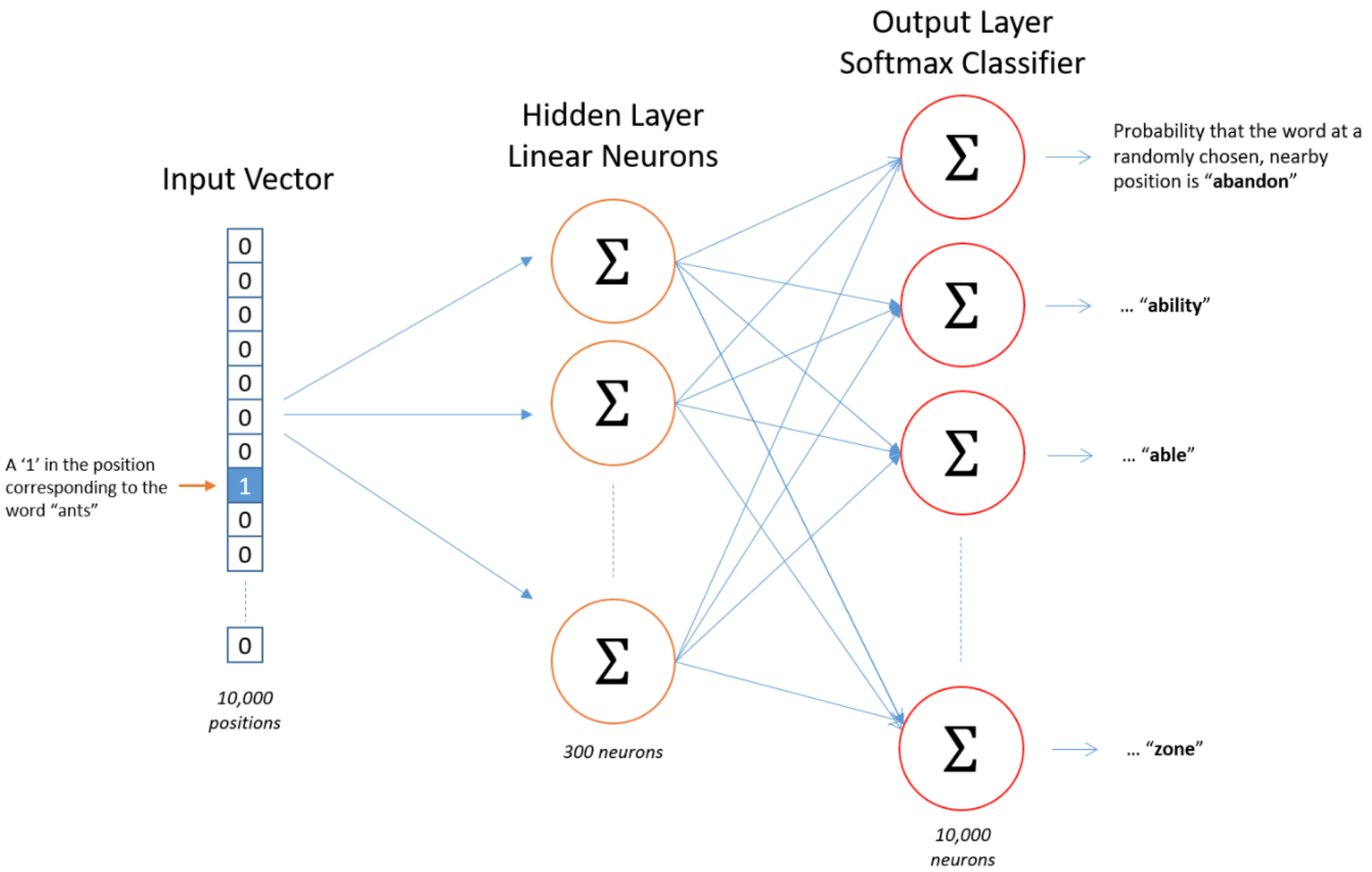
Hidden Layer
Weight Matrix



Word Vector
Lookup Table!



Word2Vec

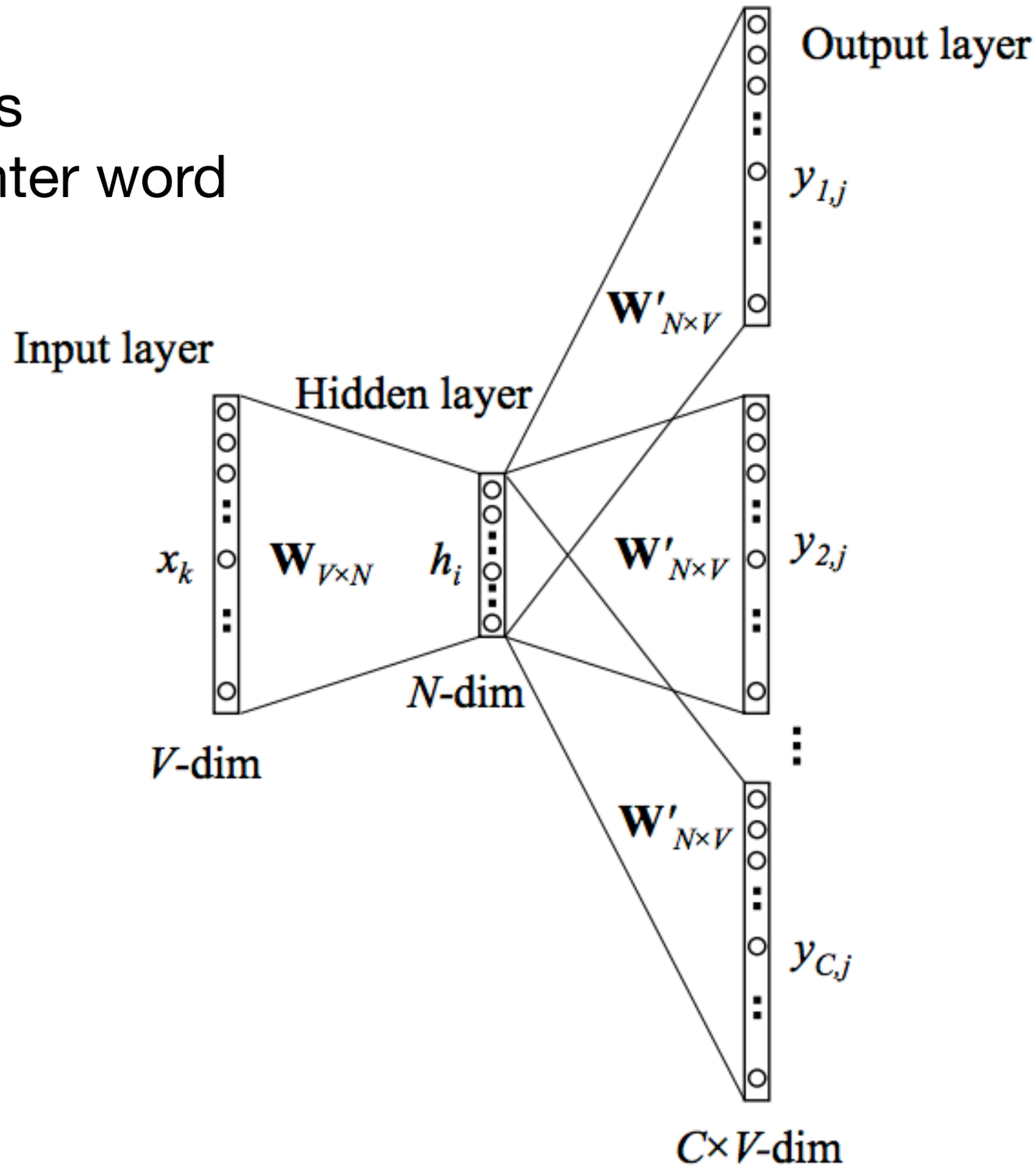

Output Layer
Softmax Classifier
Hidden Layer
Linear Neurons
Input Vector
Probability that the word at a randomly chosen, nearby position is "abandon"
... "ability"
... "able"
... "zone"
A '1' in the position corresponding to the word "ants"
0
0
0
0
0
0
1
0
0
0
10,000 positions
300 neurons
10,000 neurons

Word2Vec

Skipgrams

Predict context (”outside”) words
(position independent) given center word

Better for rare words

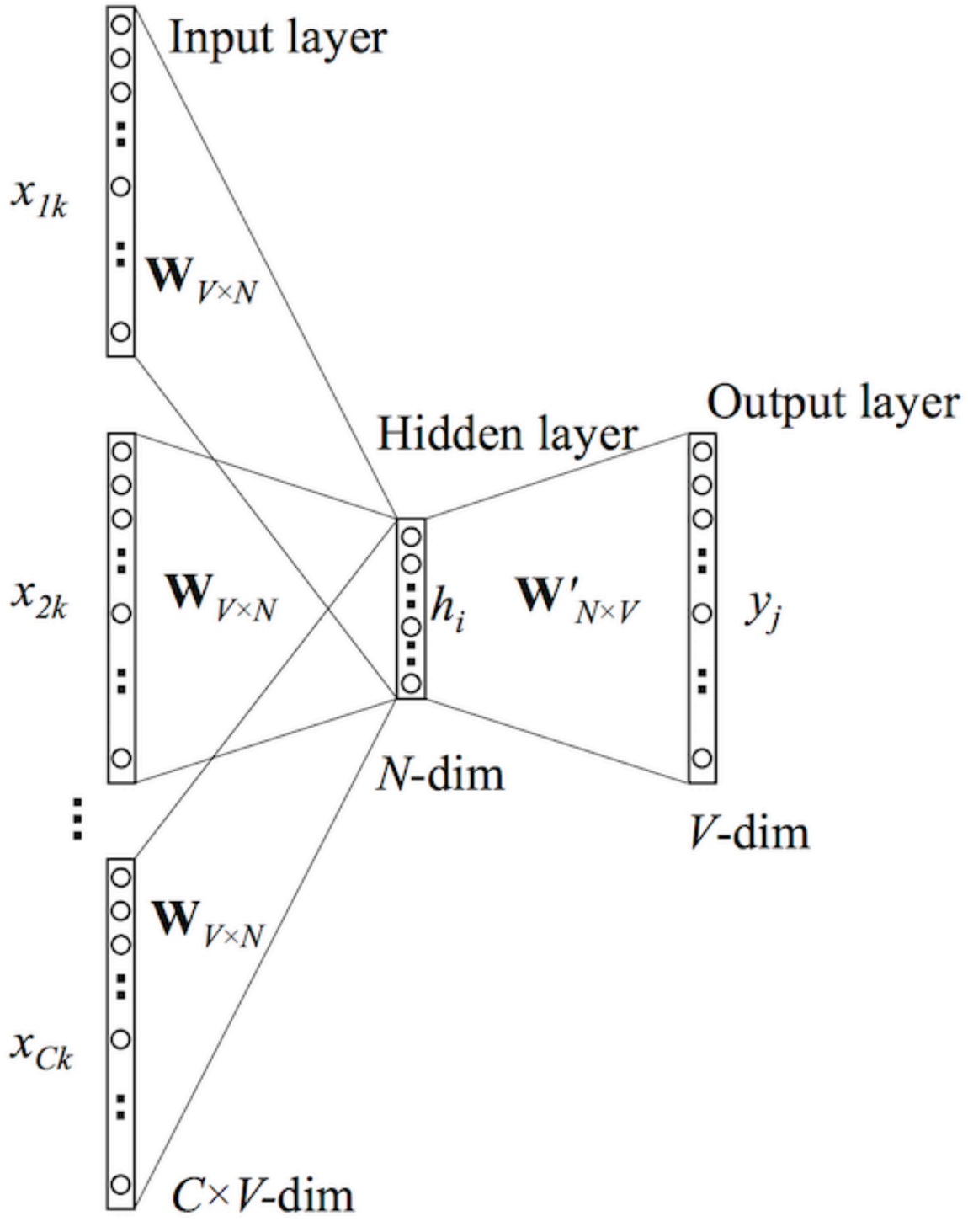


Word2Vec

CBOW

Predict center word from (bag of) context words

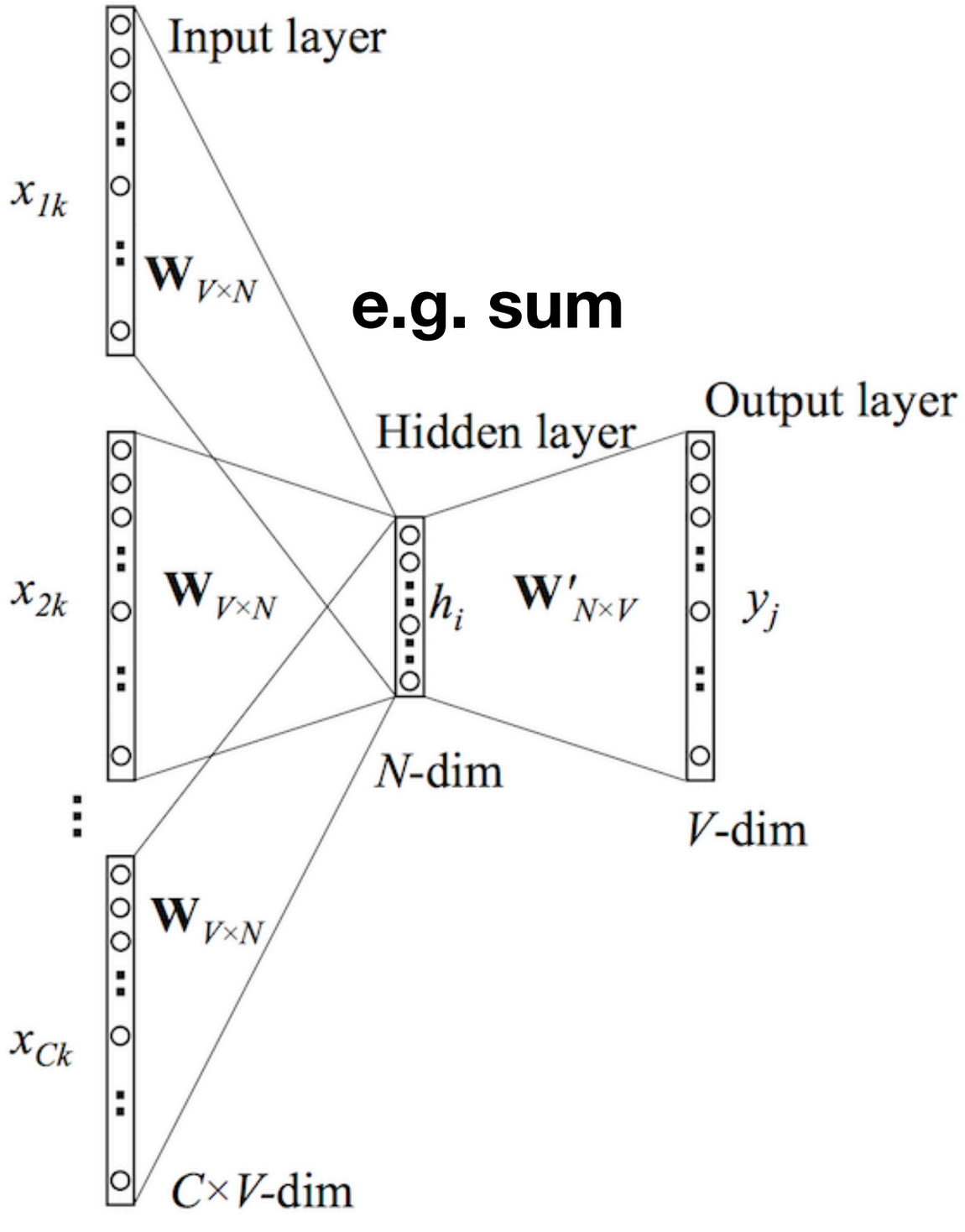
Faster



Word2Vec

CBOW

Predict center word from
(bag of) context words



Visualization

https://projector.tensorflow.org/

- BERT Embedding Projector

Visualization

Частотность слова

☑ Высокая ☑ Средняя ☐ Низкая

НКРЯ и Wikipedia

1. англия PROPN 0.58
2. европа PROPN 0.54
3. великобритания PROPN 0.52
4. страна NOUN 0.48
5. франция PROPN 0.47

Visualization

word2vec(king) - word2vec(man) + word2vec(woman) = ?

Visualization

Some vector close to queen

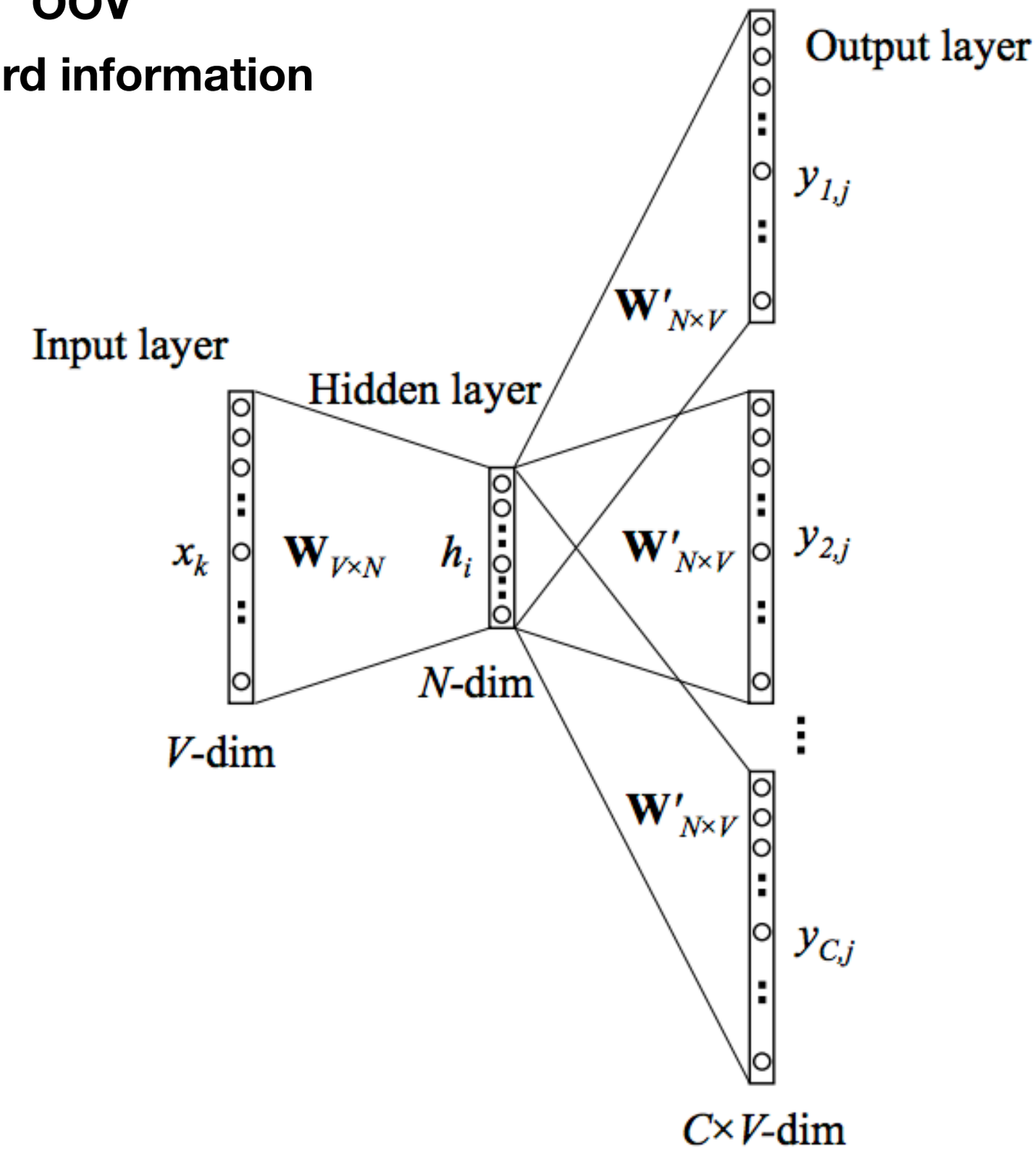
word2vec(king) - word2vec(man) + word2vec(woman) = word2vec (queen)

Word2Vec

Fasttext

OOV

Subword information

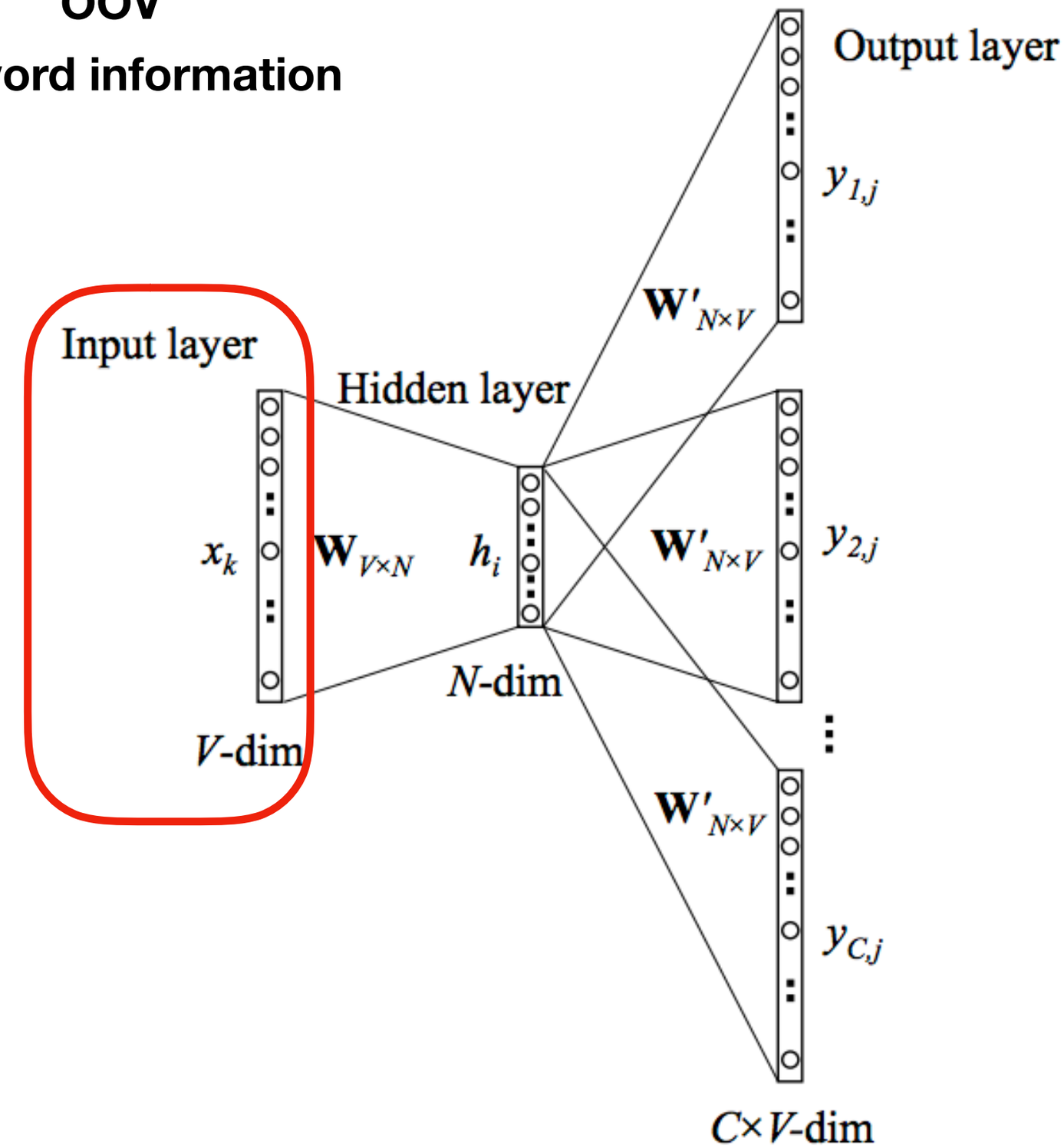


Word2Vec

Fasttext

OOV

Subword information



Word2Vec

Fasttext

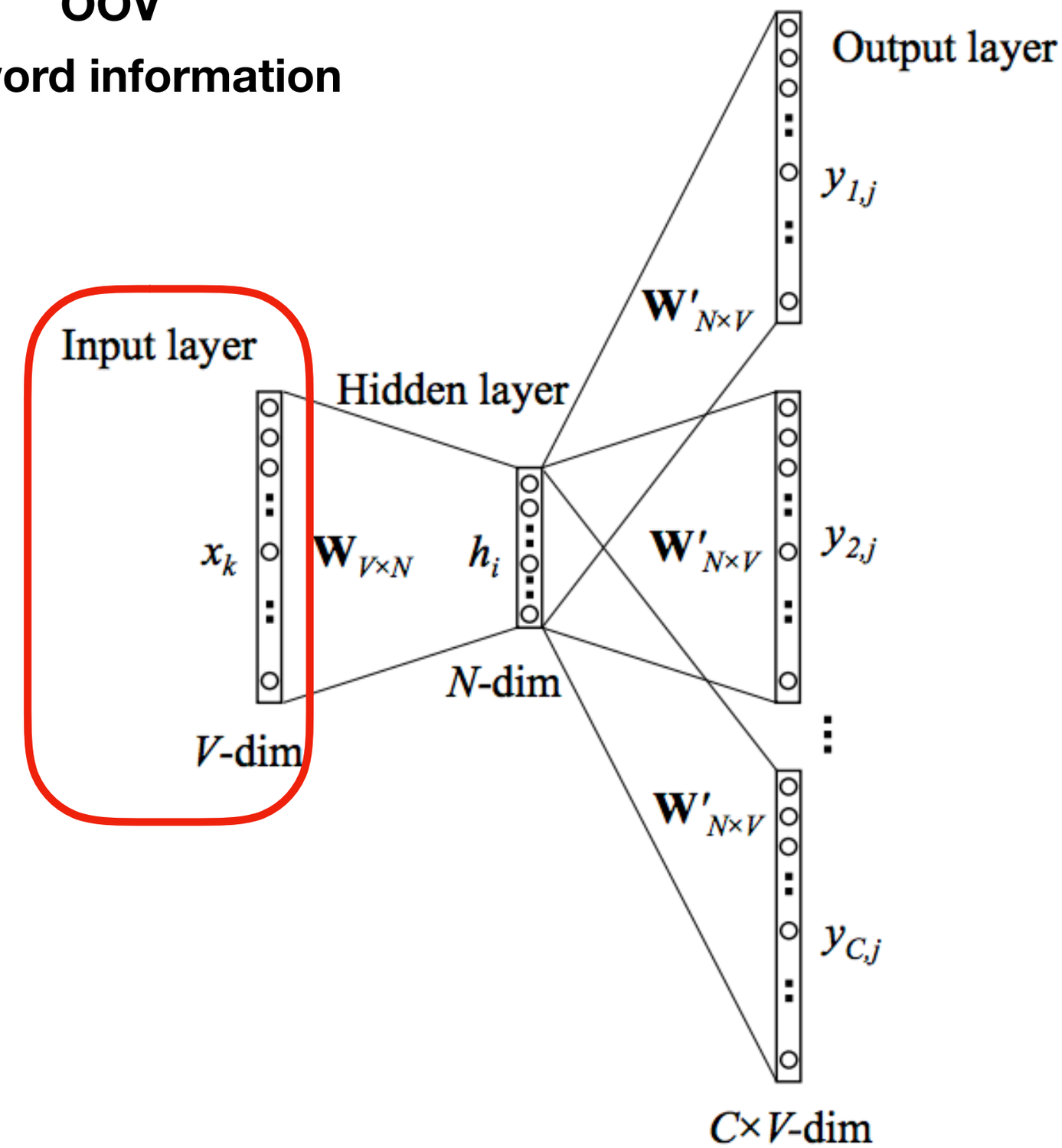
OOV

Subword information

where

=

<wh + whe + her + ere + re>



Word2Vec

Fasttext

OOV

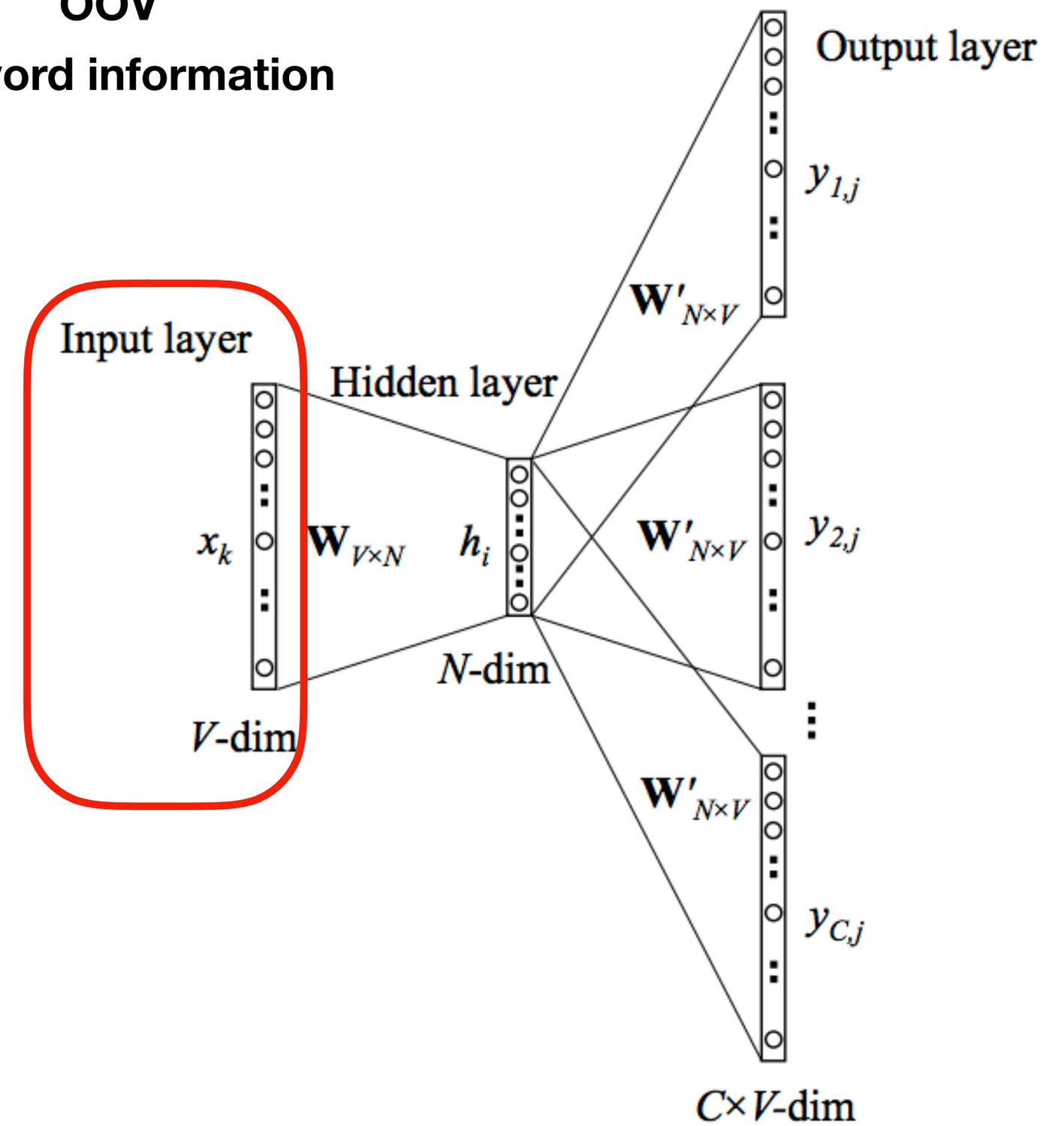
Subword information

where

=

<wh + whe + her + ere + re>

3 — 6 char n-gram length



Word2Vec

$$p(w_O|w_I) = \frac{\exp\left({v'_{w_O}}^\top v_{w_I}\right)}{\sum_{w=1}^{W} \exp\left({v'_w}^\top v_{w_I}\right)}$$

Word2Vec

$$p(w_O|w_I) = \frac{\exp\left({v'_{w_O}}^\top v_{w_I}\right)}{\sum_{w=1}^{W} \exp\left({v'_w}^\top v_{w_I}\right)}$$

Computational expensive

Word2Vec

- Hierarchical softmax

Word2Vec

- Hierarchical softmax
- Naive softmax
  - Subset of vocabulary

$$p(w_O|w_I) = \frac{\exp\left({v'_{w_O}}^\top v_{w_I}\right)}{\sum_{w=1}^{W} \exp\left({v'_w}^\top v_{w_I}\right)}$$

Word2Vec

- Hierarchical softmax
- Naive softmax
  - Subset of vocabulary

$$p(w_O|w_I) = \frac{\exp\left({v'_{w_O}}^\top v_{w_I}\right)}{\sum_{w=1}^{W} \exp\left({v'_w}^\top v_{w_I}\right)}$$

- Negative sampling
  - Binary classification

Word2Vec

Negative sampling

$$J_{neg-sample}(\boldsymbol{o}, \boldsymbol{v}_c, \boldsymbol{U}) = -\log(\sigma(\boldsymbol{u}_o^\top \boldsymbol{v}_c)) - \sum_{k=1}^{K} \log(\sigma(-\boldsymbol{u}_k^\top \boldsymbol{v}_c))$$

Word2Vec

Negative sampling

$$J_{neg-sample}(\boldsymbol{o}, \boldsymbol{v}_c, \boldsymbol{U}) = -\log(\sigma(\boldsymbol{u}_o^\top \boldsymbol{v}_c)) - \sum_{k=1}^{K} \log(\sigma(-\boldsymbol{u}_k^\top \boldsymbol{v}_c))$$

Sampling negatives

$$P(word)^{3/4}$$

Word2Vec

Negative sampling

$$J_{neg-sample}(\boldsymbol{o}, \boldsymbol{v}_c, \boldsymbol{U}) = -\log(\sigma(\boldsymbol{u}_o^\top \boldsymbol{v}_c)) - \sum_{k=1}^{K} \log(\sigma(-\boldsymbol{u}_k^\top \boldsymbol{v}_c))$$

Sampling negatives

$P(word)^{3/4}$

e.g. K = 5

Word2Vec

Negative sampling

$$J_{neg-sample}(\boldsymbol{o}, \boldsymbol{v}_c, \boldsymbol{U}) = -\log(\sigma(\boldsymbol{u}_o^\top \boldsymbol{v}_c)) - \sum_{k=1}^{K} \log(\sigma(-\boldsymbol{u}_k^\top \boldsymbol{v}_c))$$

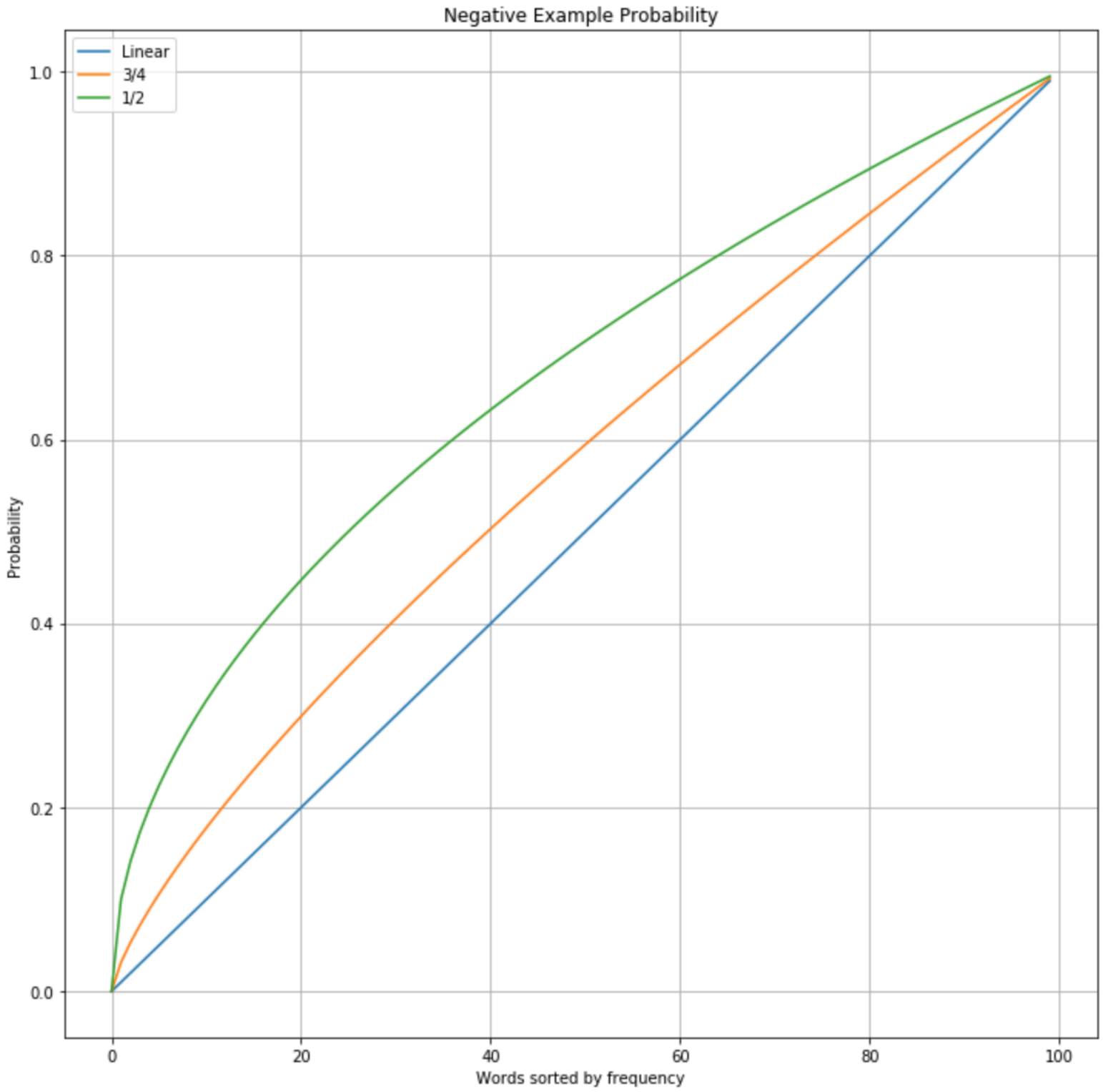
Sampling negatives

e.g. K = 5

P(word)$^{3/4}$

Increase rare word probability

Word2Vec


Negative Example Probability
Linear
3/4
1/2
1.0
0.8
0.6
0.4
0.2
0.0
Probability
0
20
40
60
80
100
Words sorted by frequency

Word2Vec

Negative sampling

$$J_{neg-sample}(\boldsymbol{o}, \boldsymbol{v}_c, \boldsymbol{U}) = -\log(\sigma(\boldsymbol{u}_o^\top \boldsymbol{v}_c)) - \sum_{k=1}^{K} \log(\sigma(-\boldsymbol{u}_k^\top \boldsymbol{v}_c))$$

Sampling negatives

$$P(word)^{3/4}$$

Word2Vec

Negative sampling

$$J_{neg-sample}(\boldsymbol{o}, \boldsymbol{v}_c, \boldsymbol{U}) = -\log(\sigma(\boldsymbol{u}_o^\top \boldsymbol{v}_c)) - \sum_{k=1}^{K} \log(\sigma(-\boldsymbol{u}_k^\top \boldsymbol{v}_c))$$

Sampling negatives

$$\text{P(word)}^{3/4}$$

Subsampling frequent words

$$P(w_i) = \frac{10^{-3}}{p_i}\left(\sqrt{10^3 p_i} + 1\right)$$

Word2Vec

Negative sampling

$$J_{neg-sample}(\boldsymbol{o}, \boldsymbol{v}_c, \boldsymbol{U}) = -\log(\sigma(\boldsymbol{u}_o^\top \boldsymbol{v}_c)) - \sum_{k=1}^{K} \log(\sigma(-\boldsymbol{u}_k^\top \boldsymbol{v}_c))$$

Sampling negatives

$$P(\text{word})^{3/4}$$

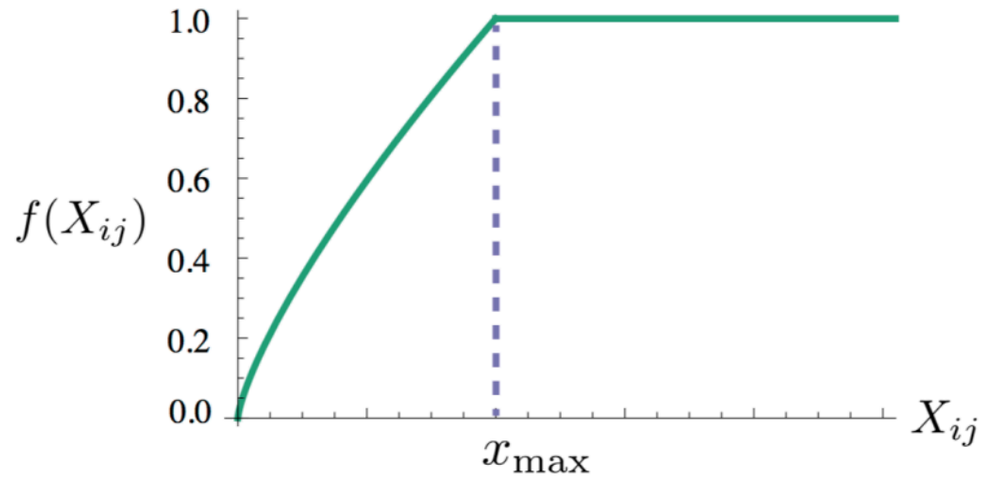
Subsampling frequent words

$$P(w_i) = \frac{10^{-3}}{p_i}\left(\sqrt{10^3 p_i} + 1\right)$$

Removing pairs

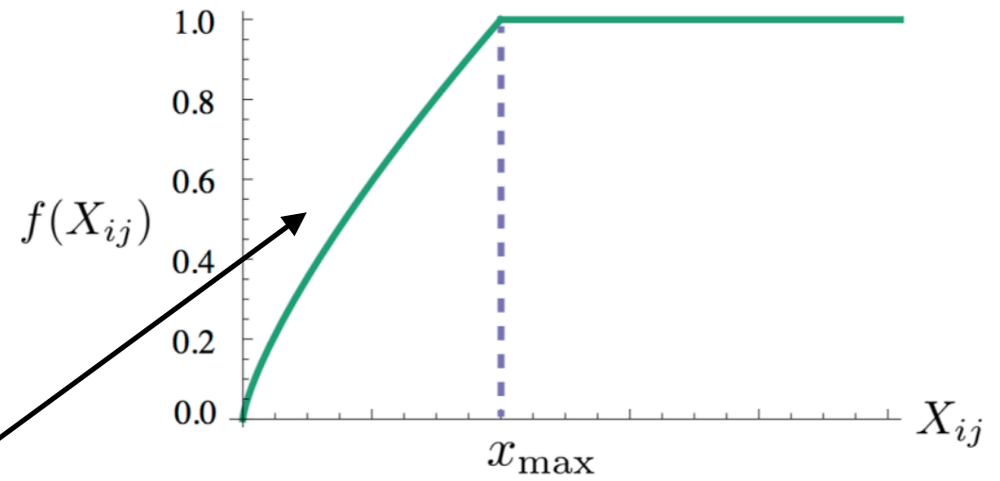
GloVe

$$J(\theta) = \frac{1}{2} \sum_{i,j=1}^{W} f(P_{ij})(u_i^T v_j - \log P_{ij})^2$$



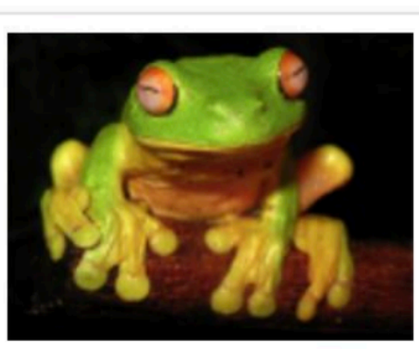
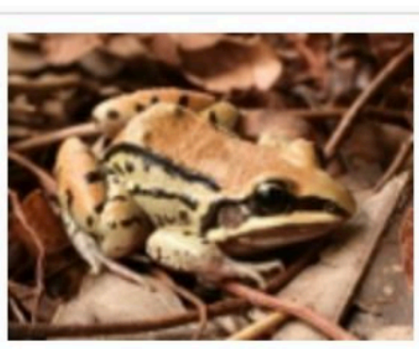
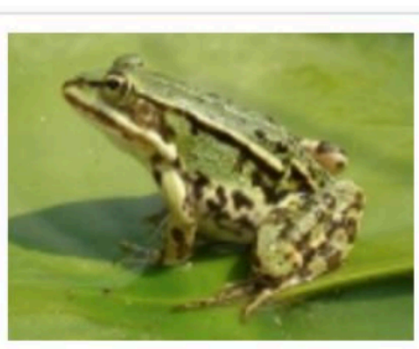
GloVe

$$J(\theta) = \frac{1}{2} \sum_{i,j=1}^{W} f(P_{ij})(u_i^T v_j - \log P_{ij})^2$$



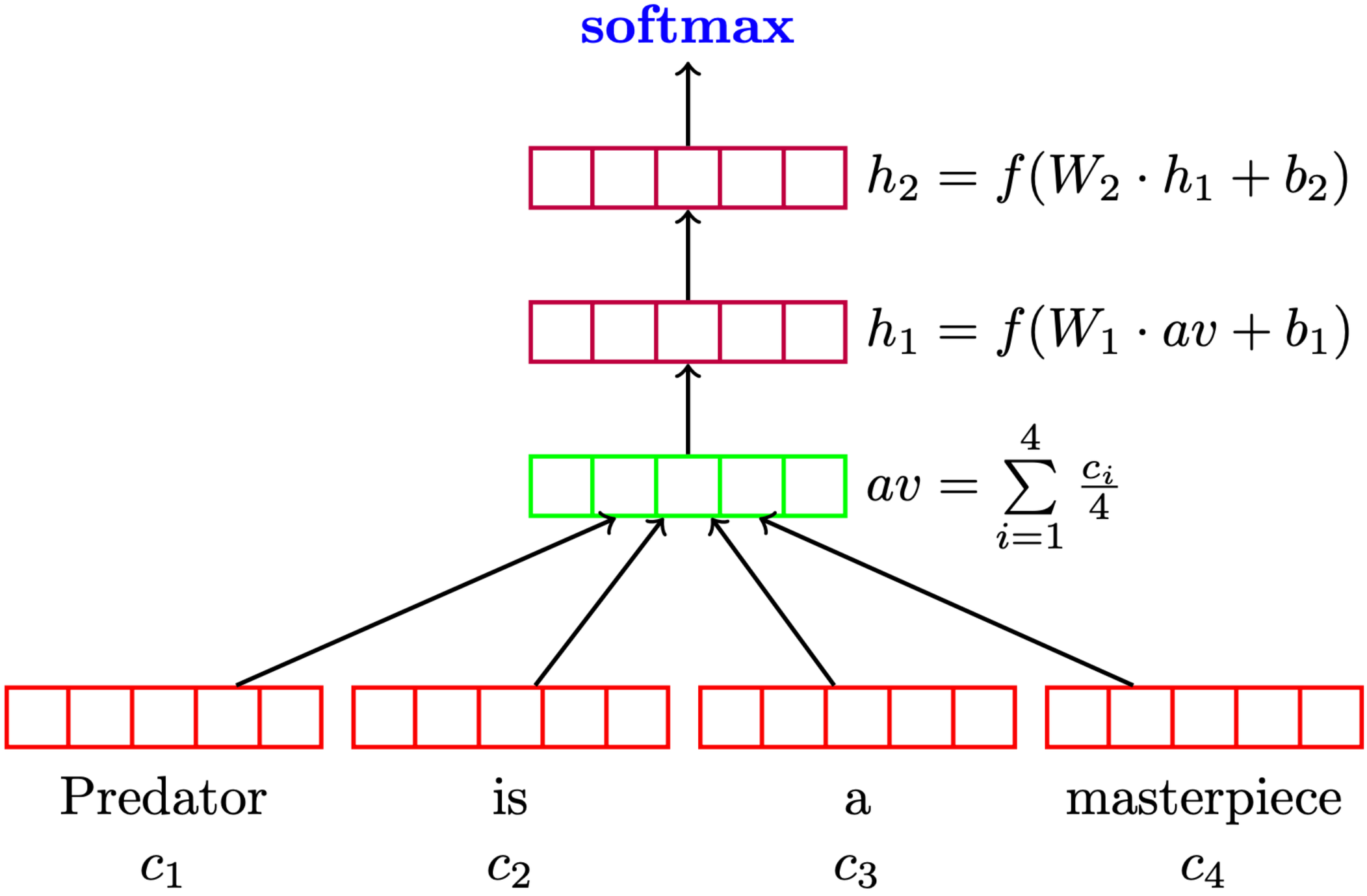
Less influence of rare words

GloVe

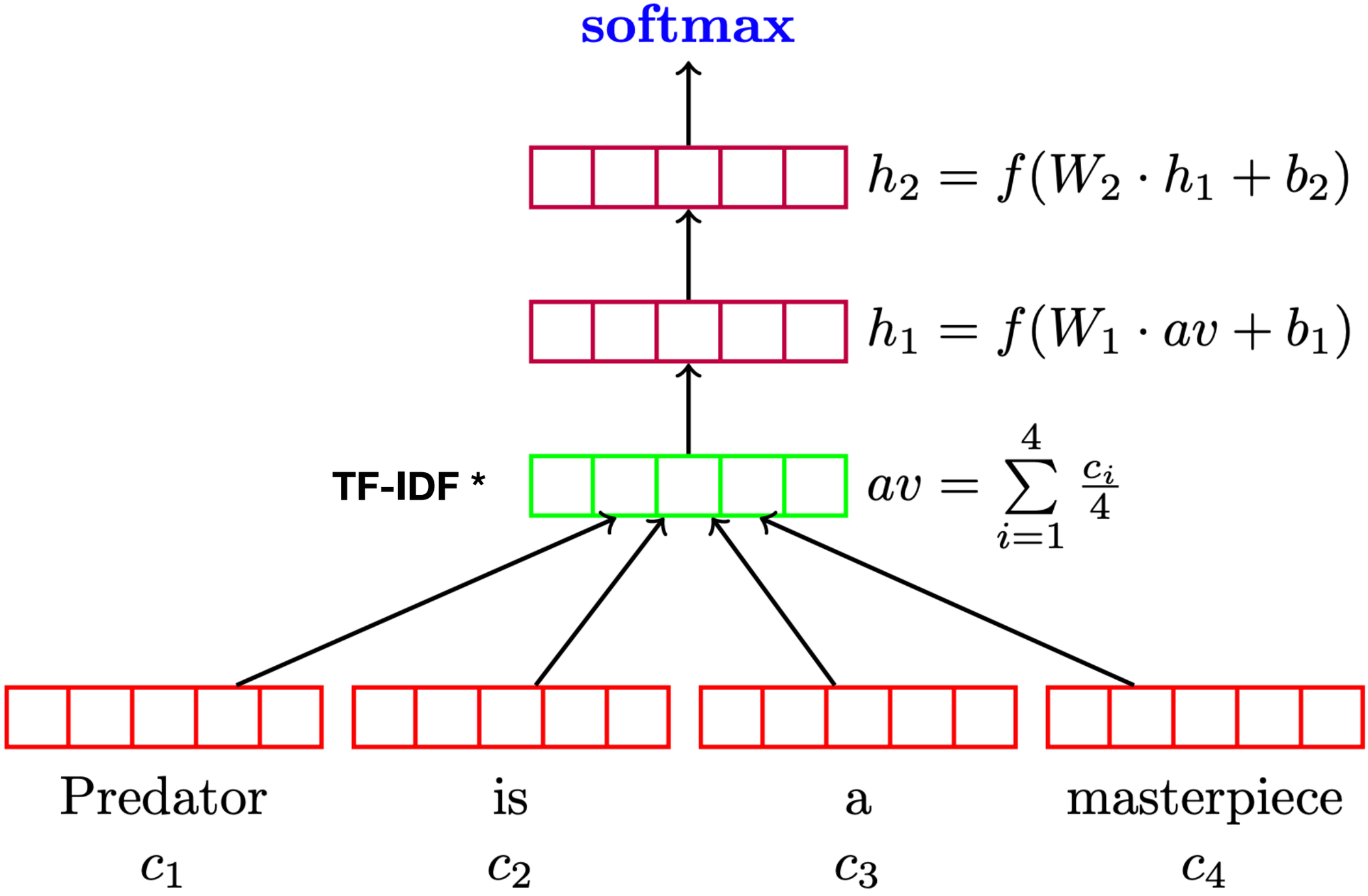
nearest neighbors of *frog*	Litoria	Leptodactylidae	Rana	Eleutherodactylus
Pictures				

How to choose embeddings?

DAN


softmax
$h_2 = f(W_2 \cdot h_1 + b_2)$
$h_1 = f(W_1 \cdot av + b_1)$
$av = \sum_{i=1}^{4} \frac{c_i}{4}$
Predator
is
a
masterpiece
c_1
c_2
c_3
c_4

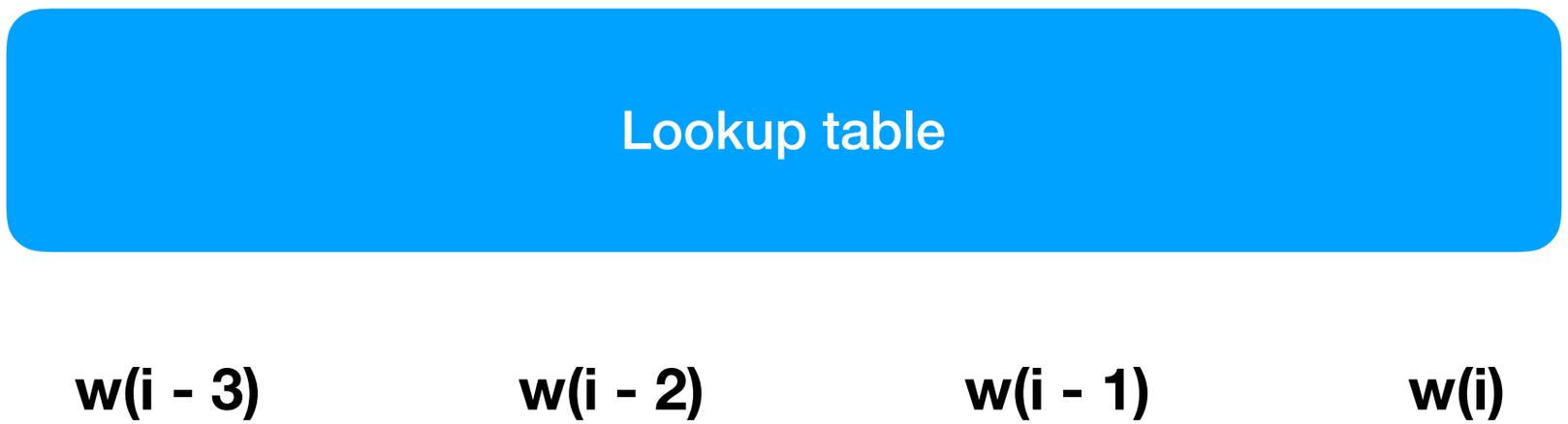
DAN

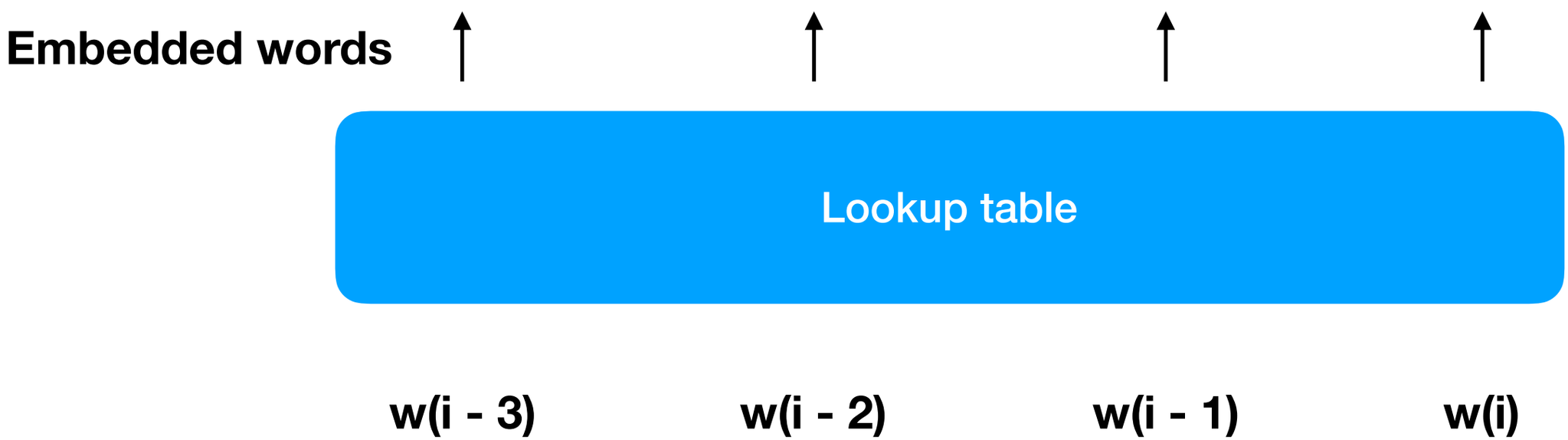

softmax
$h_2 = f(W_2 \cdot h_1 + b_2)$
$h_1 = f(W_1 \cdot av + b_1)$
TF-IDF *
$av = \sum_{i=1}^{4} \frac{c_i}{4}$
Predator
is
a
masterpiece
c_1
c_2
c_3
c_4

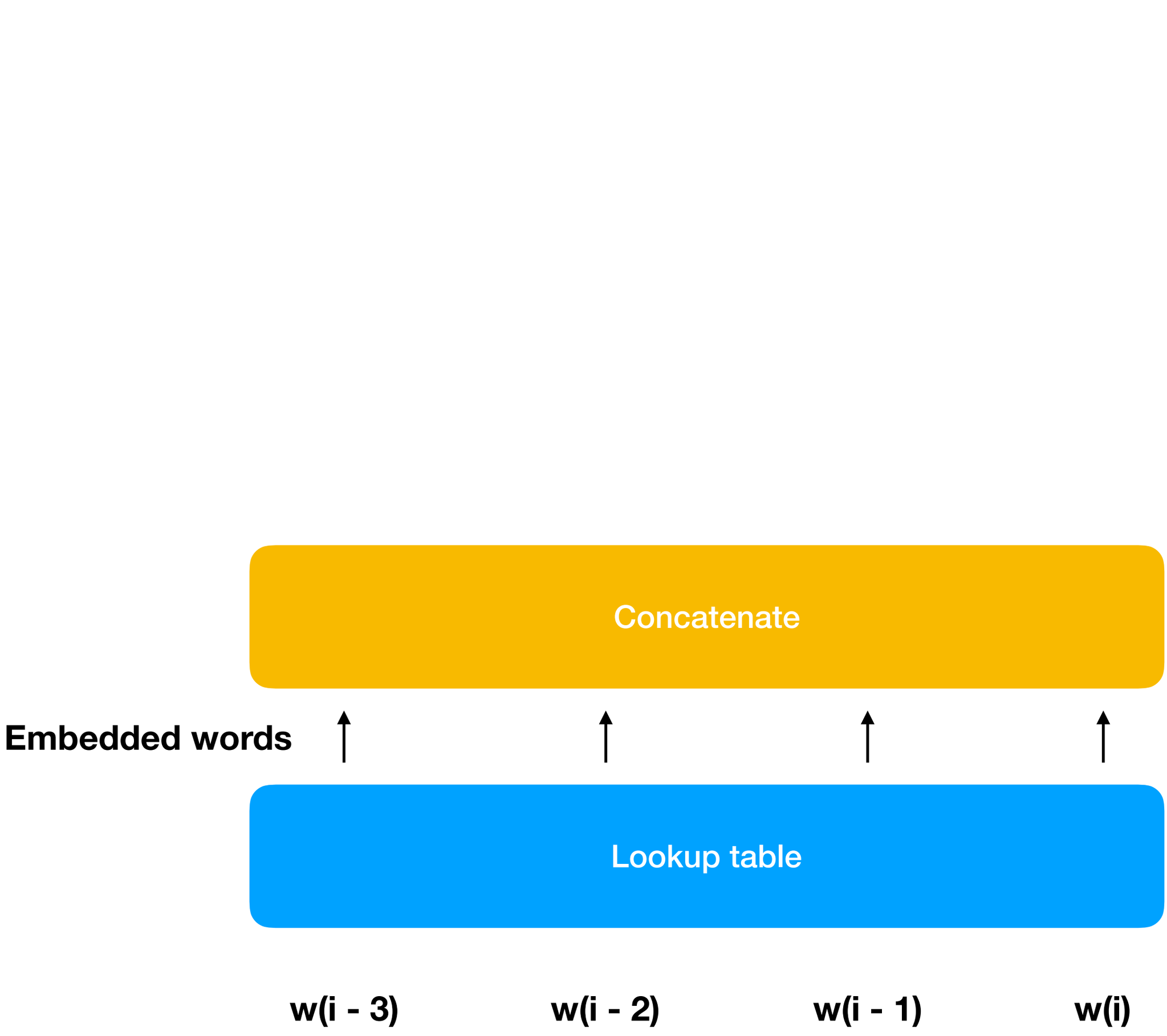
Language modeling

With CBOW

w(i - 3) w(i - 2) w(i - 1) w(i)


Lookup table
w(i - 3)
w(i - 2)
w(i - 1)
w(i)


Embedded words
Lookup table
w(i - 3)
w(i - 2)
w(i - 1)
w(i)


Concatenate
Embedded words
Lookup table
w(i - 3)
w(i - 2)
w(i - 1)
w(i)

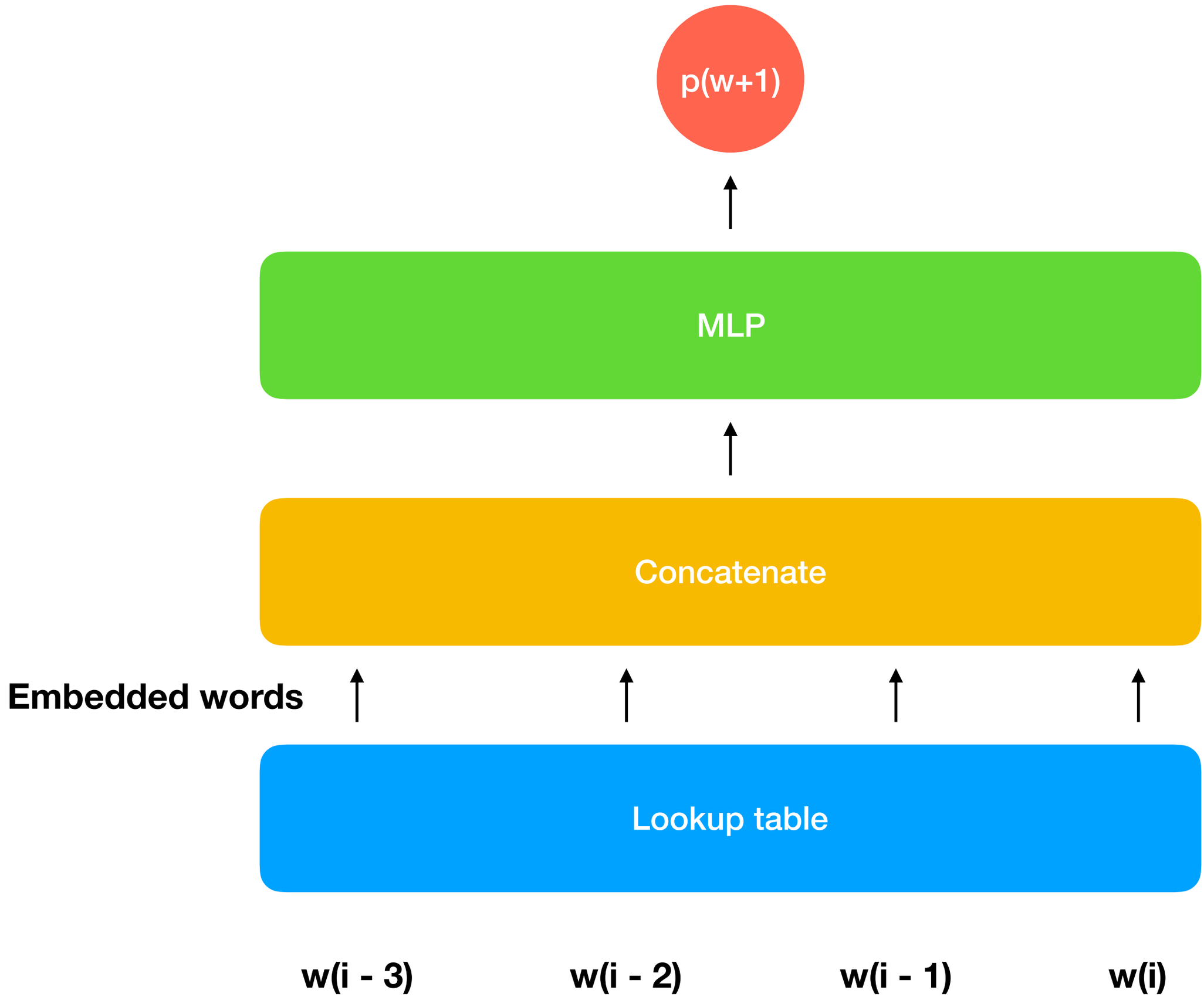
MLP

Concatenate

Embedded words

Lookup table

w(i - 3) w(i - 2) w(i - 1) w(i)


p(w+1)
MLP
Concatenate
Embedded words
Lookup table
w(i - 3)
w(i - 2)
w(i - 1)
w(i)

PyTorch

In [1]:
```
import torch

from modelling import layers, models, templates
from tools import data, wrappers
```

In [2]:
```
SEQUENCE_LENGTH = 32
EMBEDDING_DIM = 512
TRANSFORMER_LAYERS = 4

BATCH_SIZE = 16
```

In [3]:
```
dataset = data.TransformerLanguageModelData(tokenizer_path='bpe.model',
                                            max_sequence_length=SEQUENCE_LENGTH,
                                            bos_index=2,
                                            eos_index=3,
                                            batch_size=BATCH_SIZE)
```

In [4]:
```
dataset.load(file_path='testing_texts.txt', verbose=True)
```

```
Loading: 100%|██████████| 96951/96951 [00:01<00:00, 76085.59it/s]
```

In [5]:
```
loss = layers.TransformerLanguageModelLoss()
```

In [6]:
```
model = templates.TransformerLanguageModelTemplate(
    embedding_layer=models.Embedding(vocab_size=dataset.tokenizer.vocab_size(), embedding_dim=EMBEDDING_DIM),
    transformers=torch.nn.ModuleList([
        models.TransformerEncoder(
            layers.MultiheadAttention(embedding_dim=EMBEDDING_DIM, num_heads=512//64),
            model_dim=EMBEDDING_DIM,
            inner_dim=EMBEDDING_DIM*4) for _ in range(TRANSFORMER_LAYERS)]),
    sequence_length=SEQUENCE_LENGTH
)
```

In [7]:
```
optimizer = torch.optim.Adam(params=model.parameters(), betas=(0.9, 0.98), eps=1e-9)
```

In [8]:
```
scheduler = layers.NoamOpt(model_size=512, factor=1., warmup=3000, optimizer=optimizer)
```

In [9]:
```
gpt = wrappers.LanguageModelWrapper(model=model, loss=loss, data=dataset, optimizer=optimizer, scheduler=scheduler)
```

In [14]:
```
gpt.train()
```

```
Train Epoch 1: 100%|██████████| 6400/6400 [04:00<00:00, 26.20it/s, loss=9.02, moving_average=8.51]
Validation Epoch 1: 976it [00:36, 28.87it/s, loss=8.26]
Train Epoch 2:   0%|          | 0/6400 [00:00<?, ?it/s]
```

```
Epoch: 1 | Train Loss: 9.016 | Validation Loss: 8.259
```

Thanks for your Attention!

Boris Zubarev

@bobazooba