Word2vec

(Hierarchical Softmax, Negative Sampling, Subsampling)

Dohyun Kim
dhkim1028@korea.ac.kr
Data Intelligence Lab, Korea University
2020.05.18.

Class Lab - Schedule & Assignment

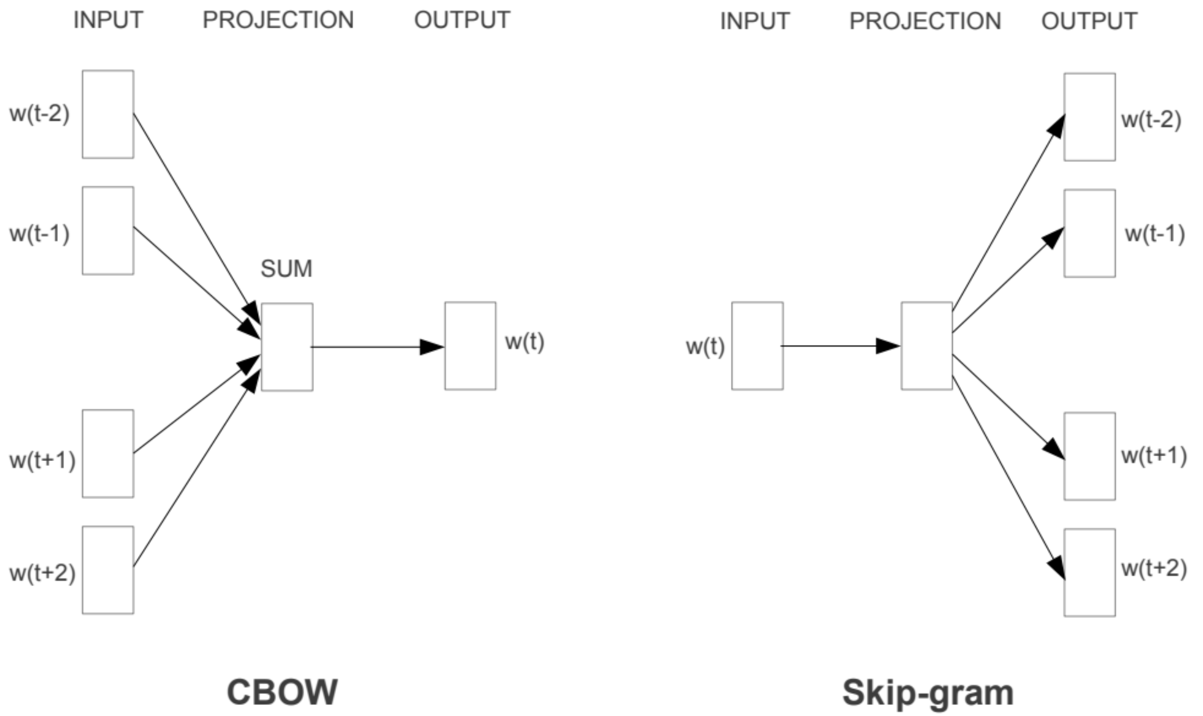
1. Skip-gram/CBOW with (Basic) Softmax (~5/20)

2. Skip-gram/CBOW with Hierarchical Softmax, Negative sampling, Subsampling (~6/7)

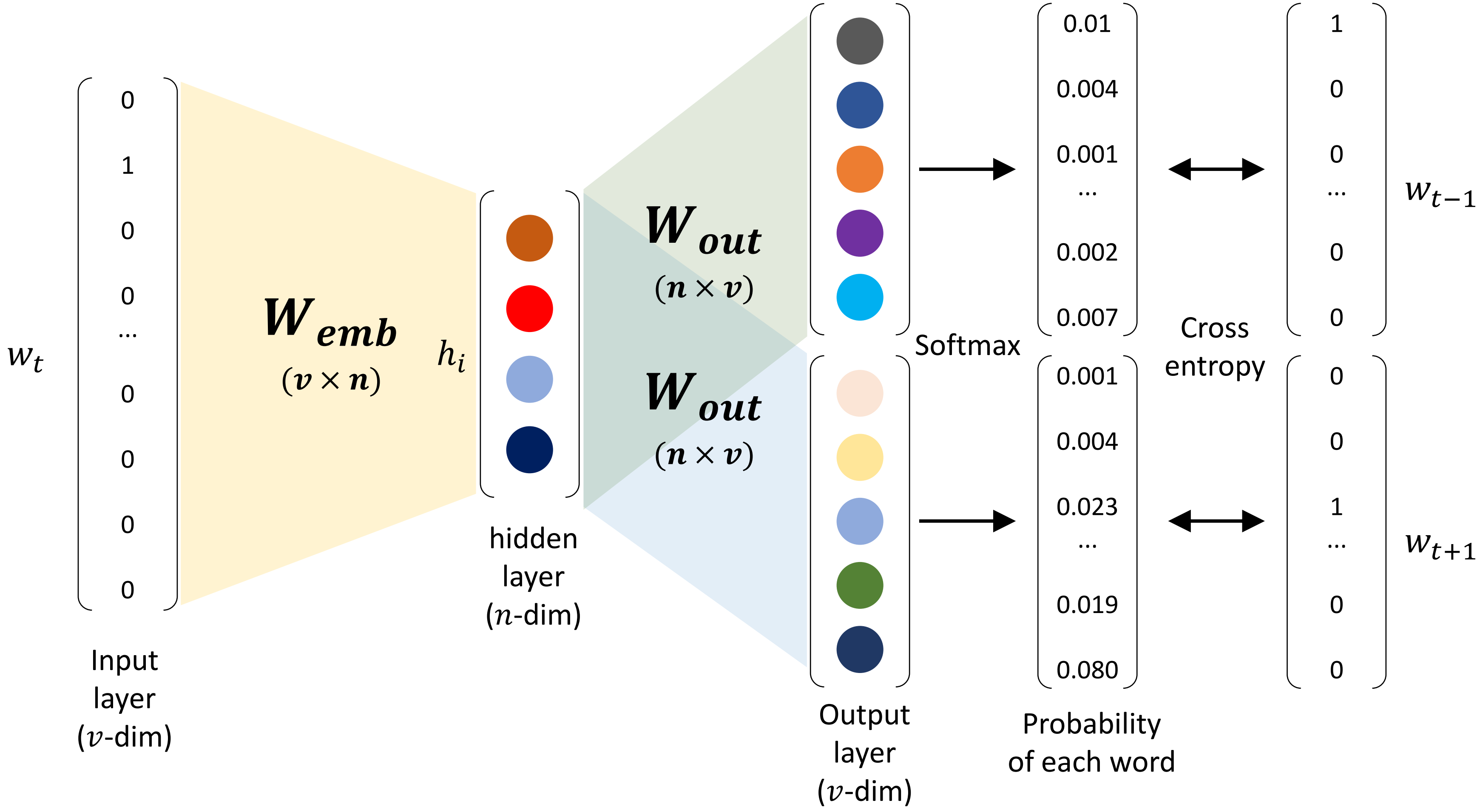
3. Fasttext / CNN(Yoon Kim) / RNN + Attention (~6/28)

Class Lab - Schedule & Assignment

- T. Mikolov, K. Chen, G. Corrado, J. Dean, “Efficient Estimation of Word Representations in Vector Space”, ICLR 2013



Word2Vec

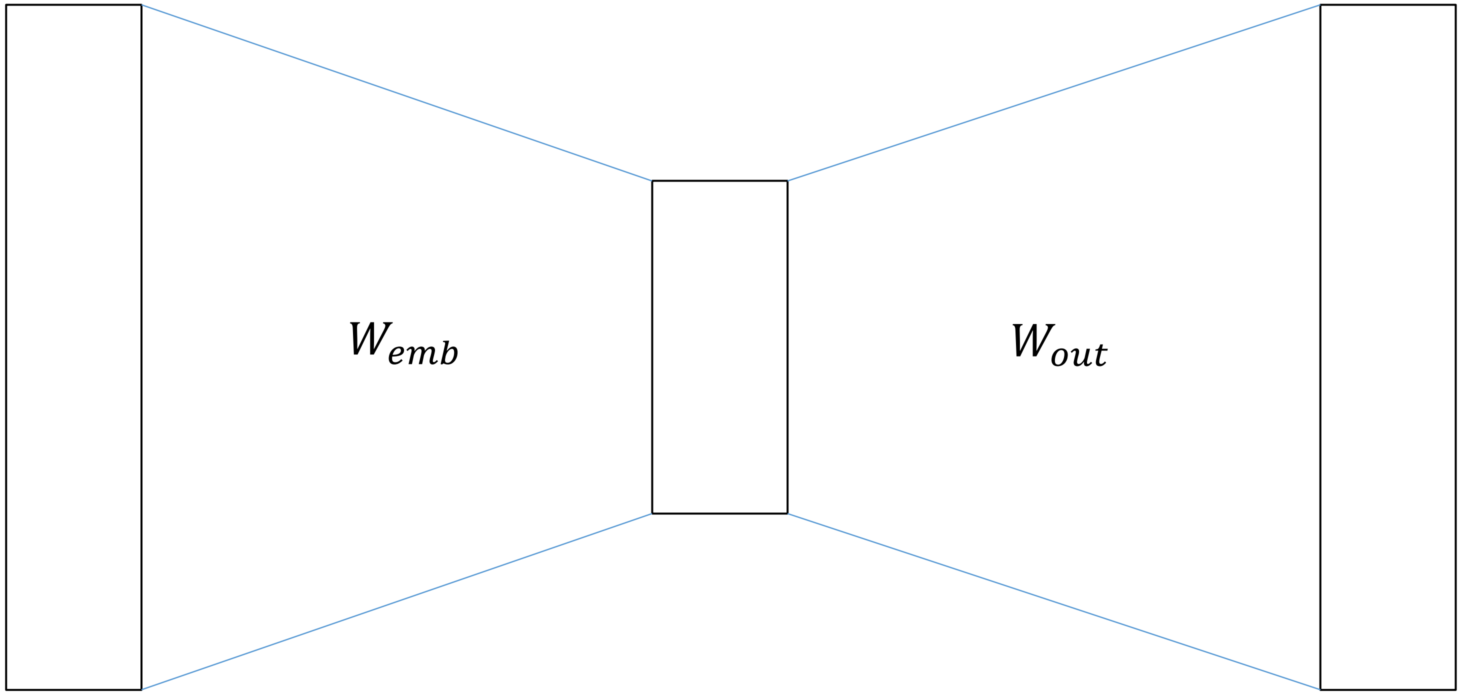


w_t
0
1
0
0
…
0
0
0
0
Input
layer
(v-dim)
W_{emb}
$(v \times n)$
h_i
hidden
layer
(n-dim)
W_{out}
$(n \times v)$
W_{out}
$(n \times v)$
Output
layer
(v-dim)
Softmax
0.01
0.004
0.001
…
0.002
0.007
0.001
0.004
0.023
…
0.019
0.080
Probability
of each word
Cross
entropy
1
0
0
…
0
0
0
0
1
…
0
0
w_{t-1}
w_{t+1}

Word2Vec

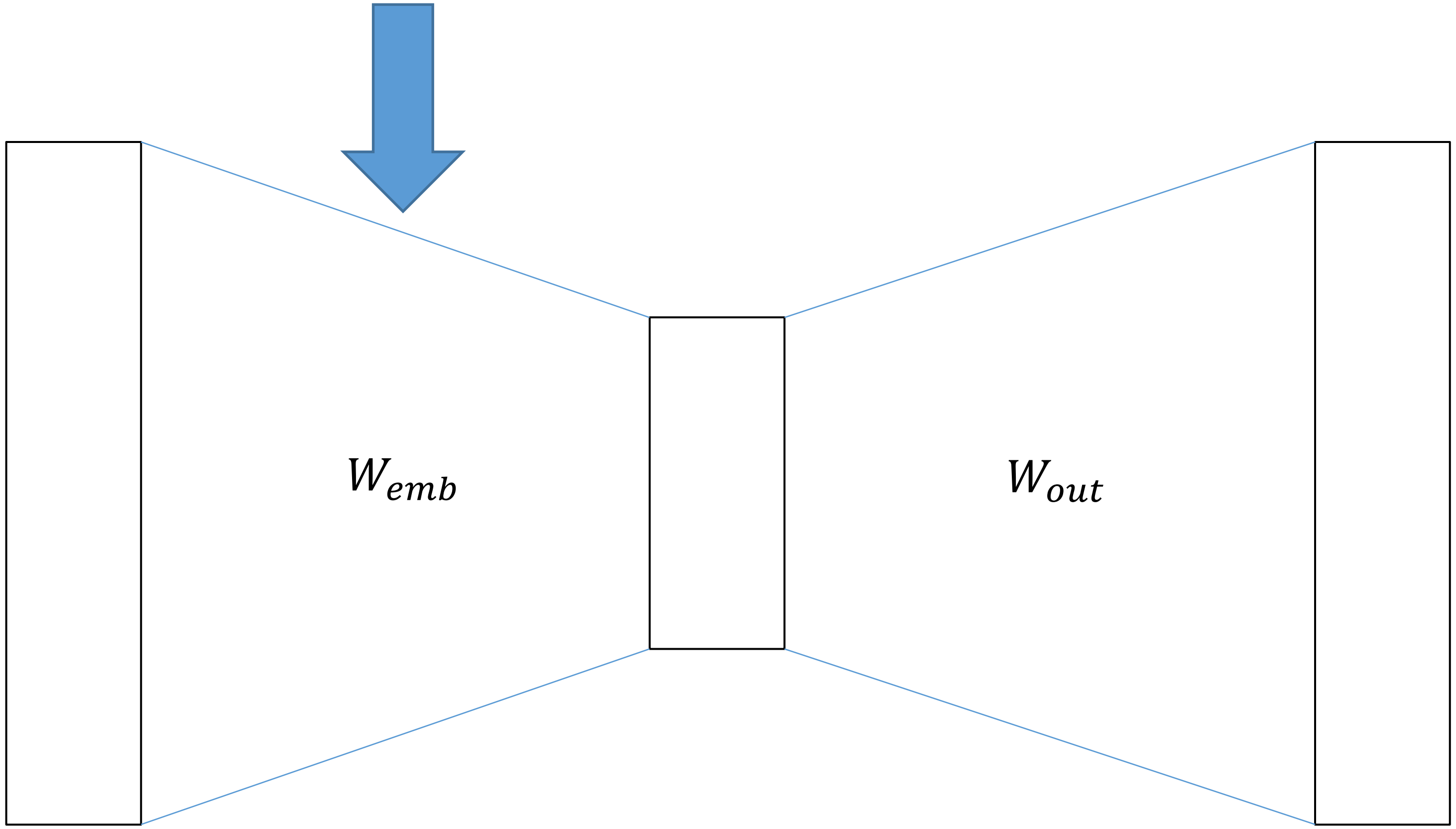
- Word2vec is very slow…

Why?

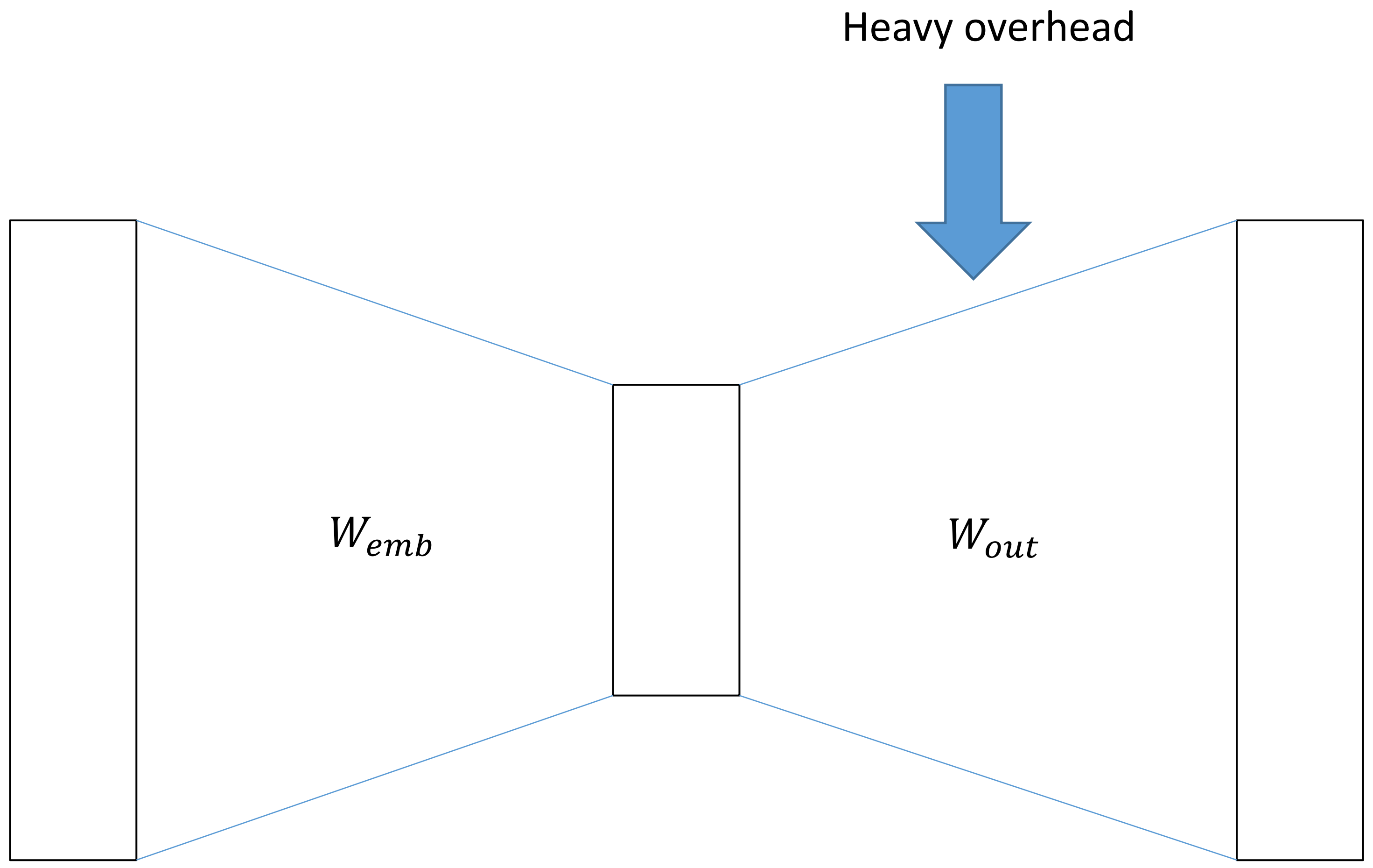


Word2Vec

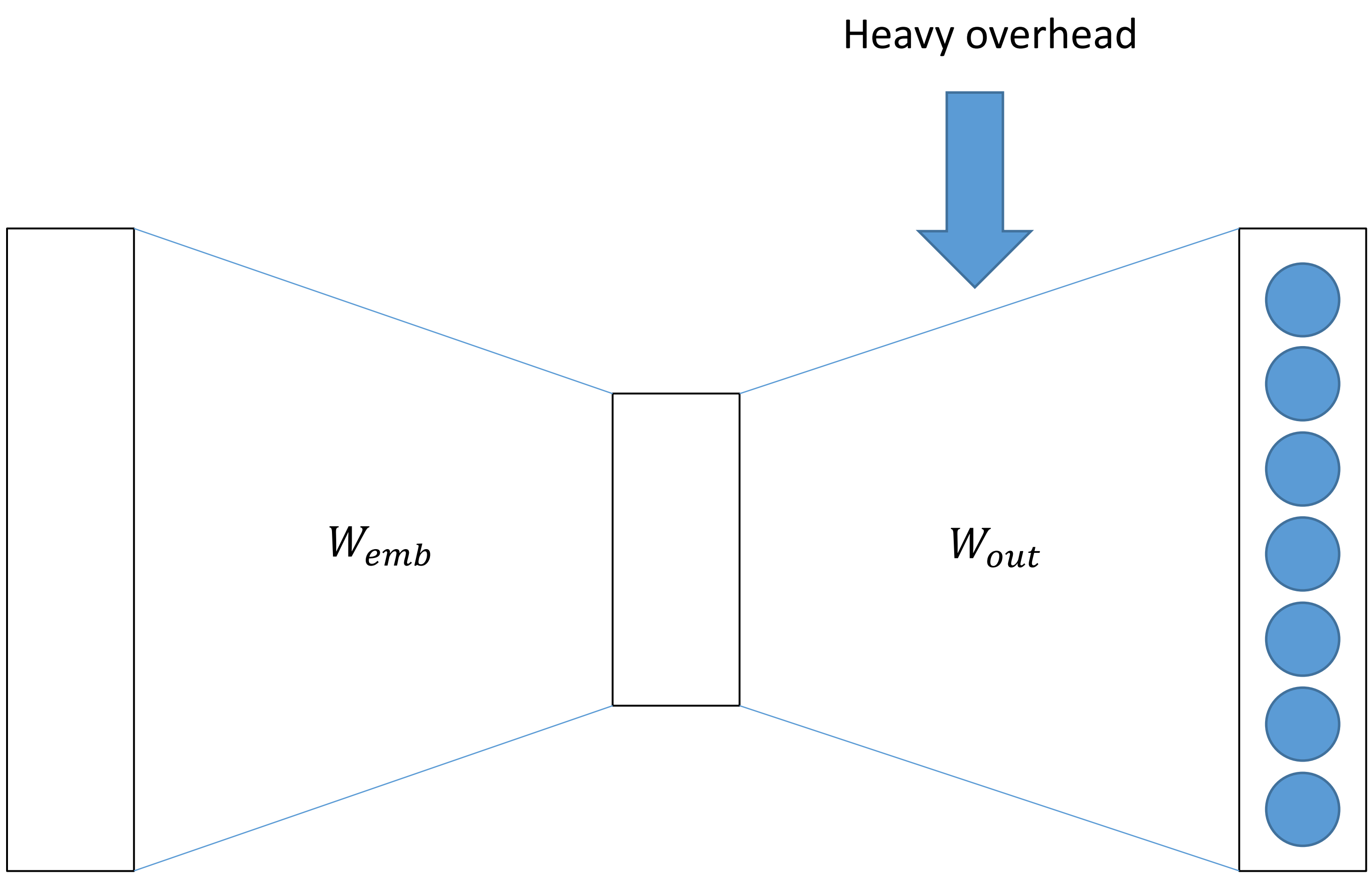
No overhead(just load a vector instead of matrix multiplication)



Word2Vec



Word2Vec

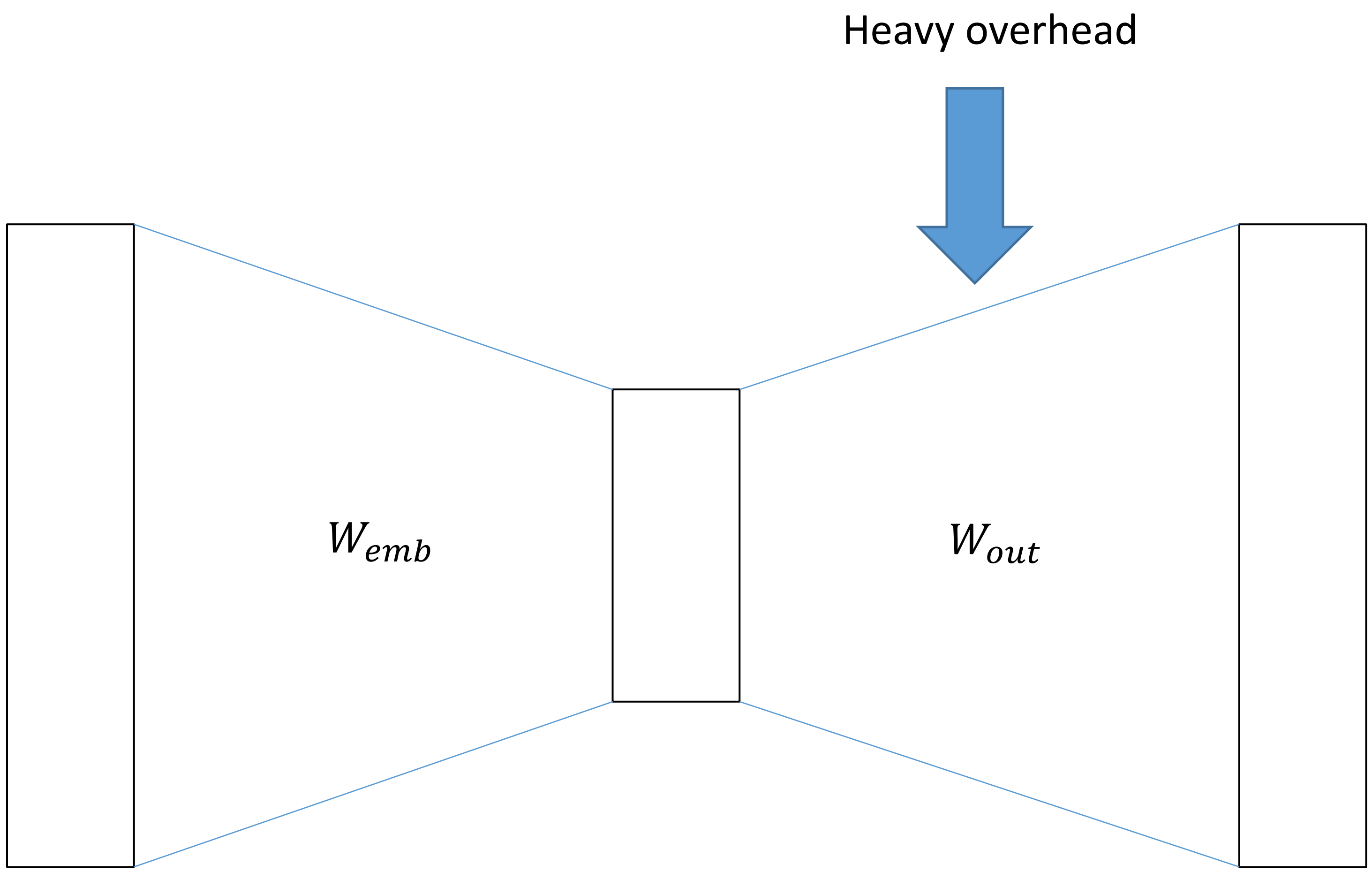


The reason is…

$$y = softmax(o) = \frac{e^o}{\sum_k e^k}$$

Softmax function needs all values of the output vector

Word2Vec



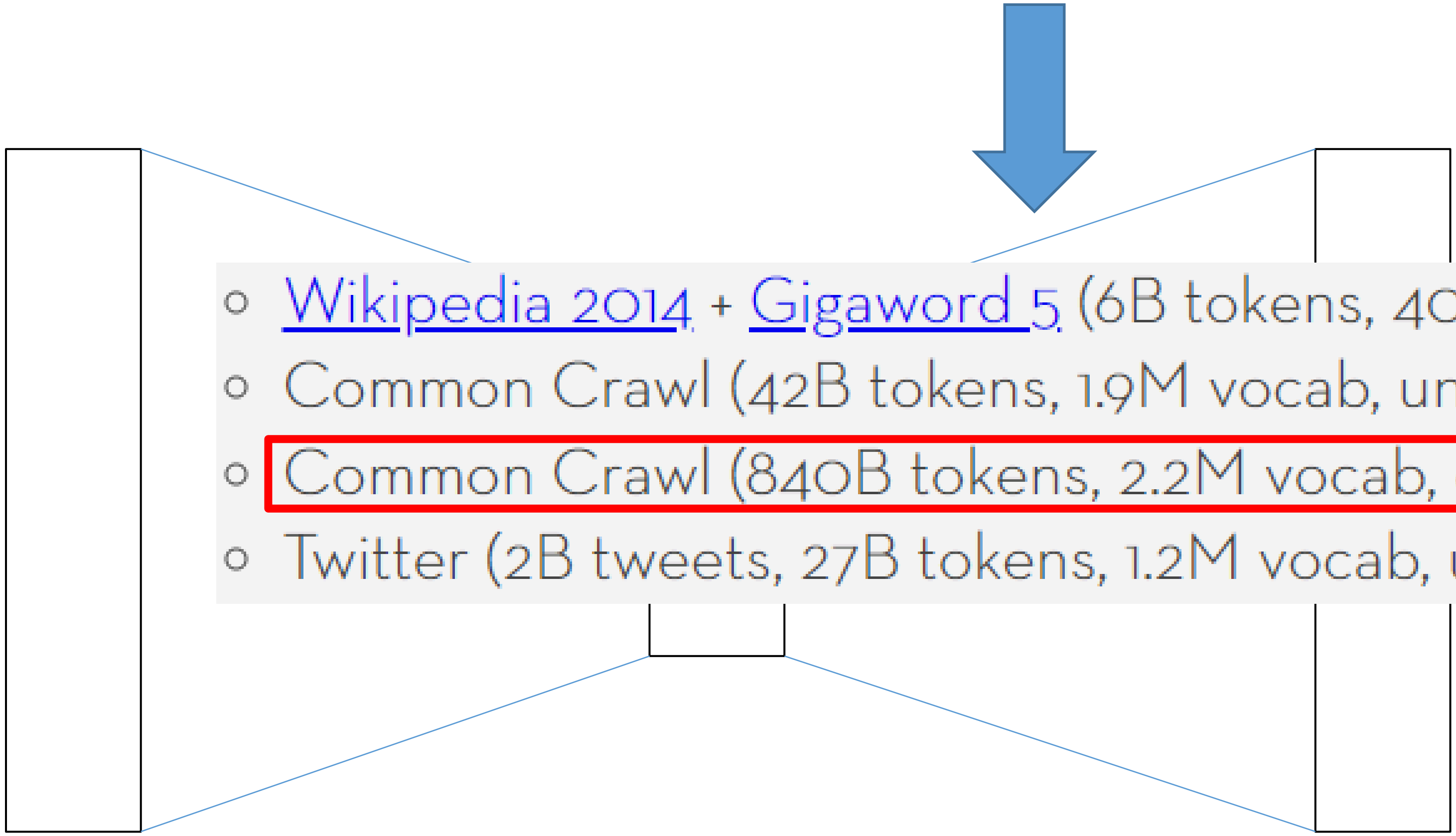
The reason is…

Output dimension : V
Feature dimension : D

Complexity : O(V x D)

Word2Vec

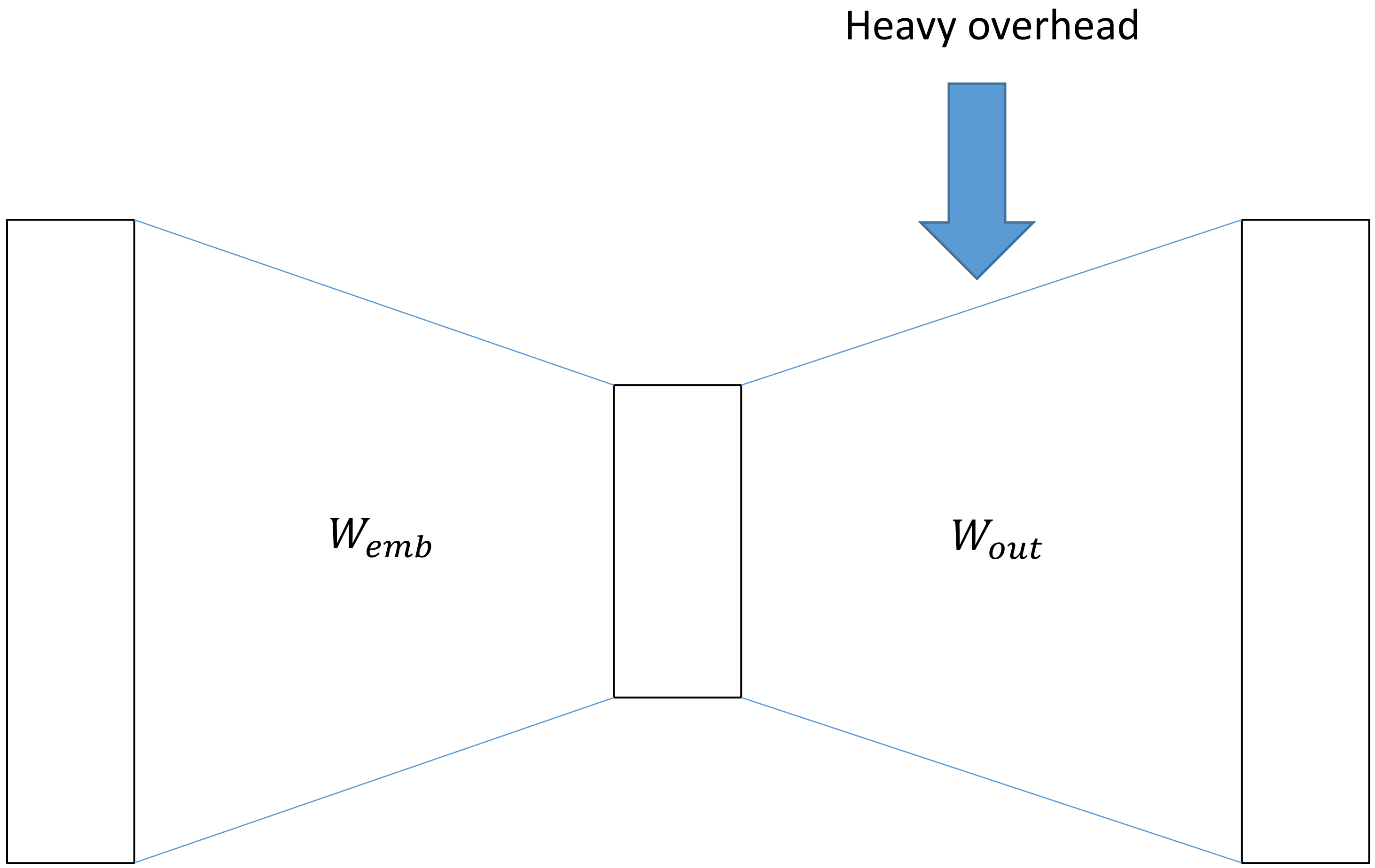
Heavy overhead



The reason is…

- [Wikipedia 2014](#) + [Gigaword 5](#) (6B tokens, 400K vocab, uncased,
- Common Crawl (42B tokens, 1.9M vocab, uncased, 300d vectors
- Common Crawl (840B tokens, 2.2M vocab, cased, 300d vectors
- Twitter (2B tweets, 27B tokens, 1.2M vocab, uncased, 25d, 50d, 1

Word2Vec

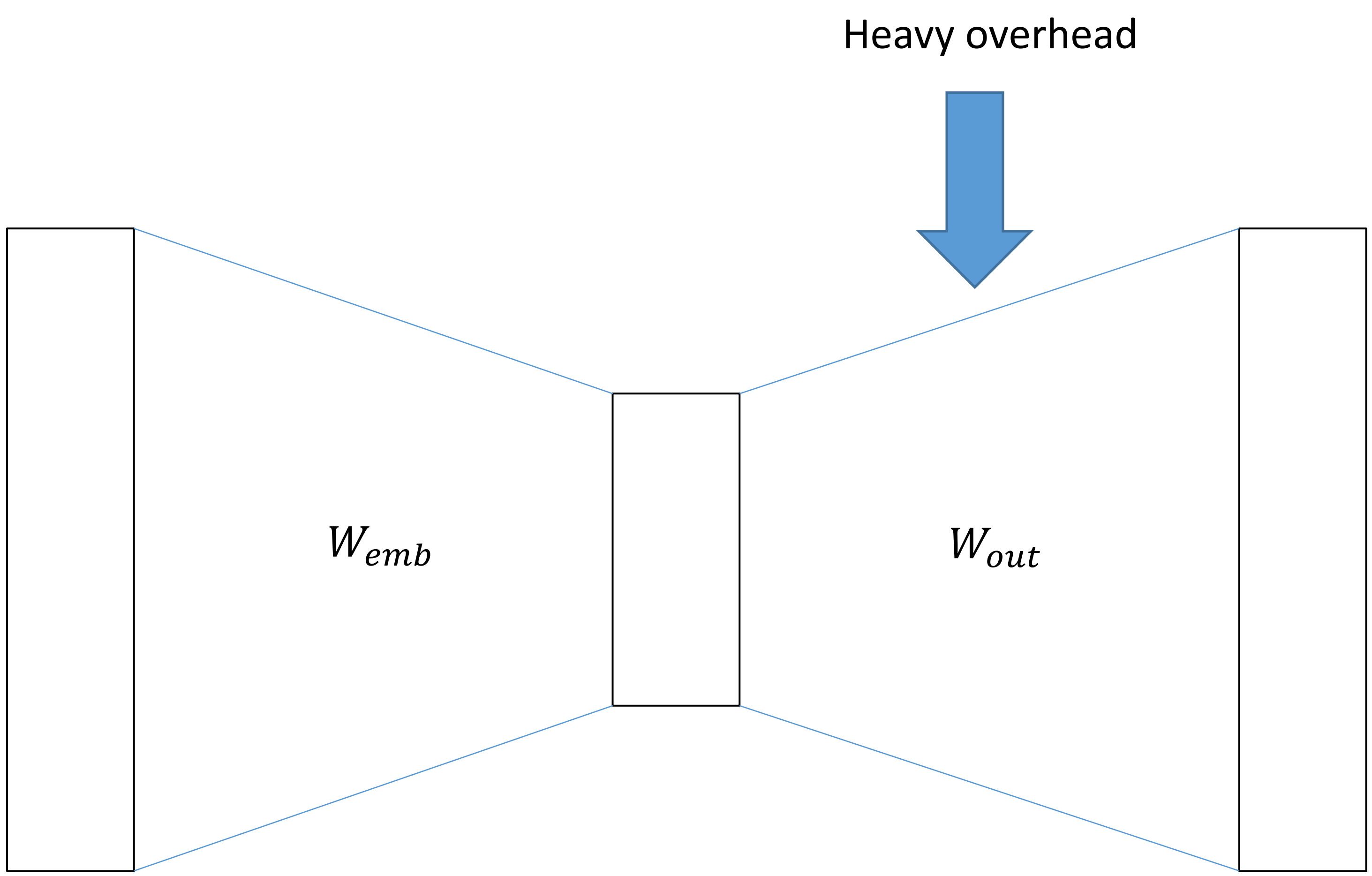


With 840B dataset

Output dimension : 2.2M
Feature dimension : 300

W_{out} : (2.2M, 300)

Word2Vec

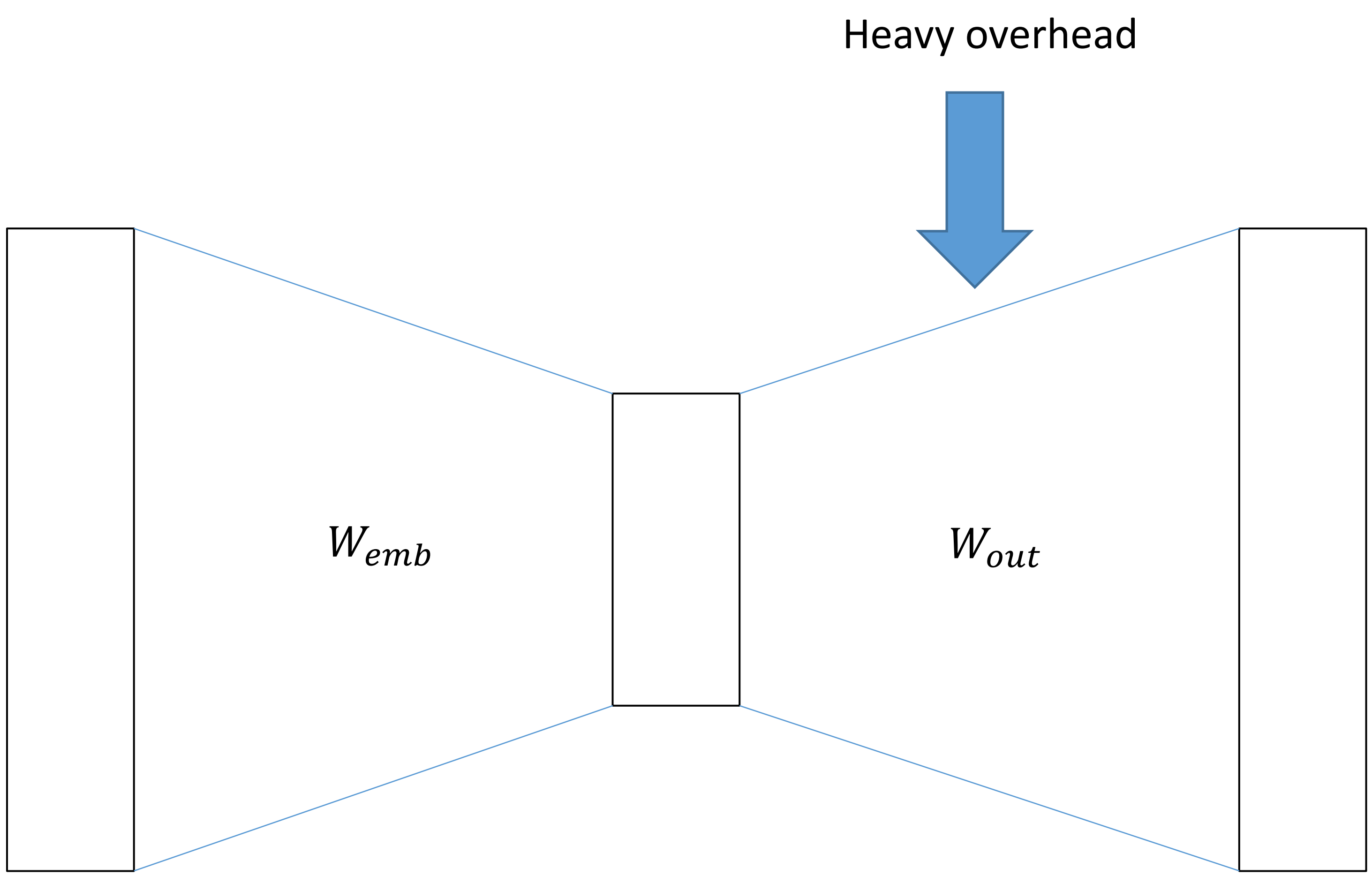


With 840B dataset

Output dimension : 2.2M
Feature dimension : 300

660M operations to calculate
$y = softmax({W_{out}}^T W_{emb}[k])$

Word2Vec



With 840B dataset

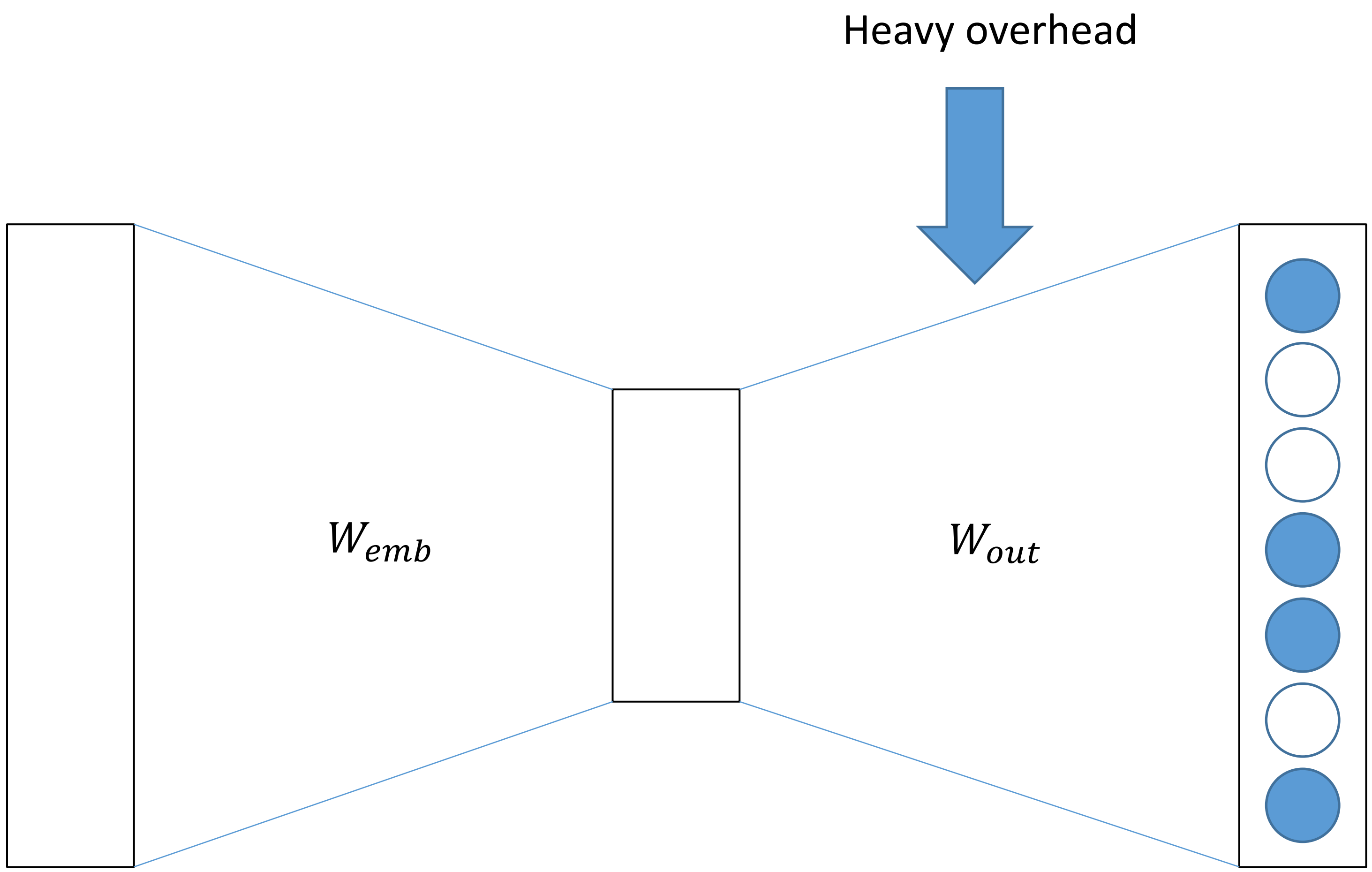
Output dimension : 2.2M
Feature dimension : 300

840B tokens with window size 5

10 training pairs each word

660M x 8.4 trillion operations an epoch

Word2Vec



The reason is…

Output dimension : V
Feature dimension : D

Complexity : O(V x D)

The idea is…

Use a portion of the output vector

Word2vec

1. Hierarchical Softmax

2. Negative Sampling

3. Subsampling

Word2Vec

Hierarchical Softmax

1. Give every word a binary code (Huffman coding recommended)

ex) apple : 000
banana : 001
cherry : 010
…

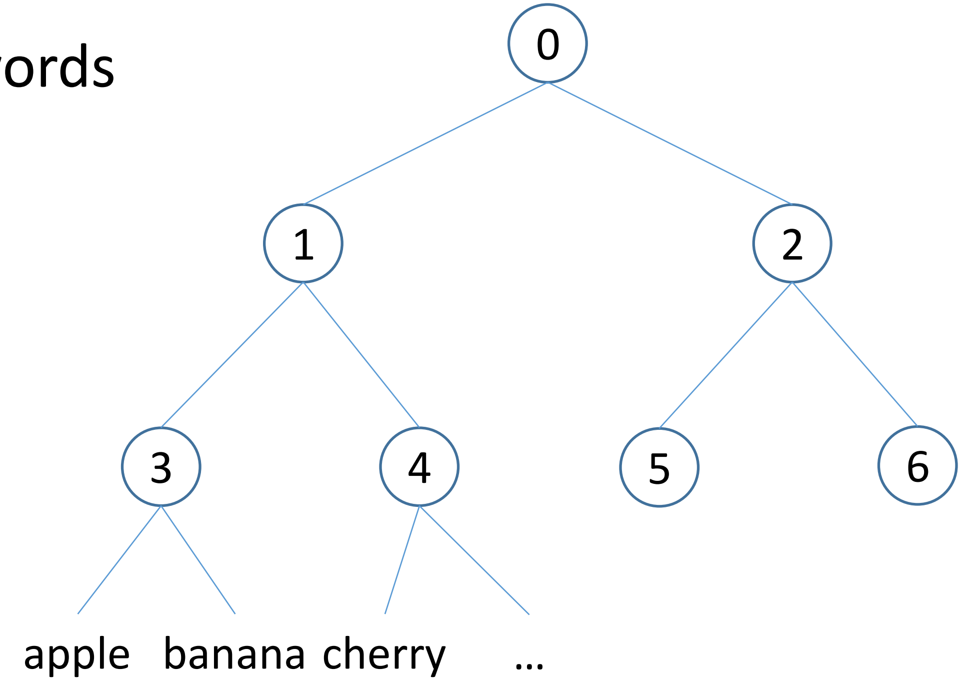
Word2Vec

Hierarchical Softmax

2. Make a binary tree whose leaf nodes are the words

ex) apple : 000
banana : 001
cherry : 010
…

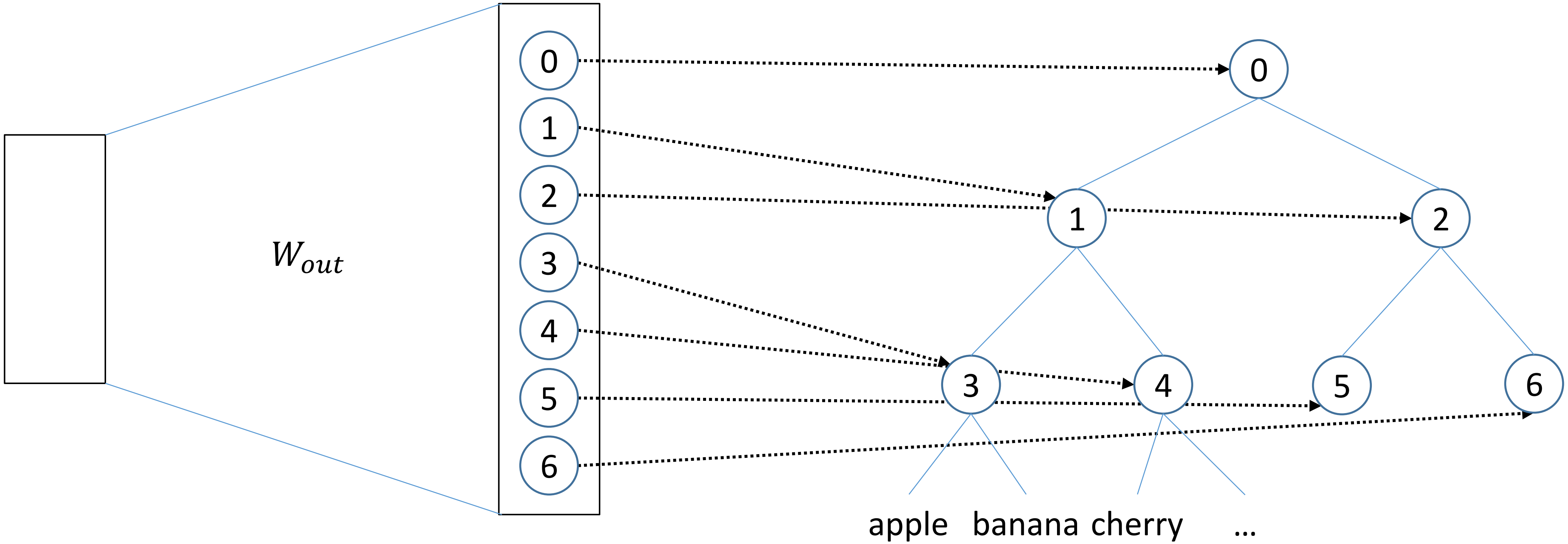
Suppose that 0 is the left and 1 is the right



Word2Vec

Hierarchical Softmax

3. Predict probability of “each non-leaf node”



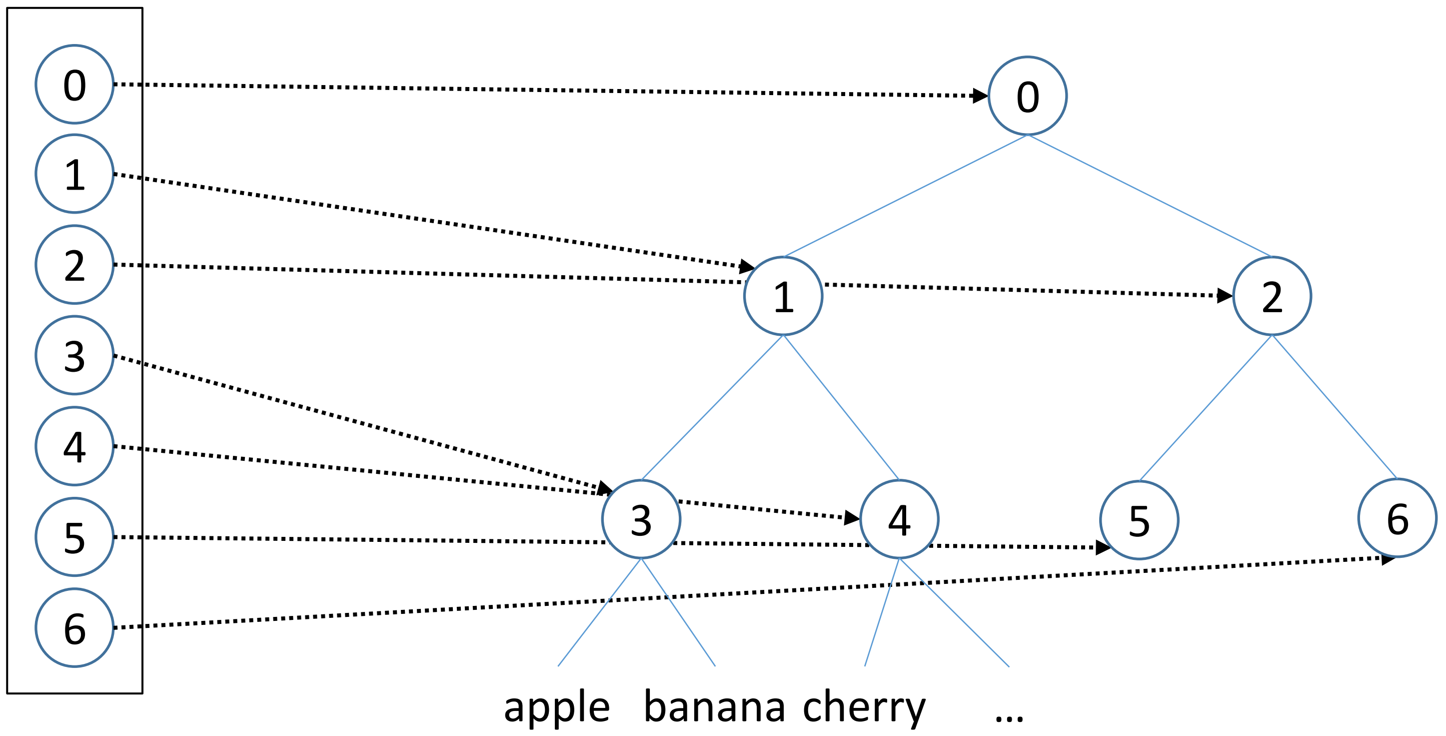
Word2Vec

Hierarchical Softmax

3. Predict probability of "each non-leaf node"

sigmoid activation function instead of softmax

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

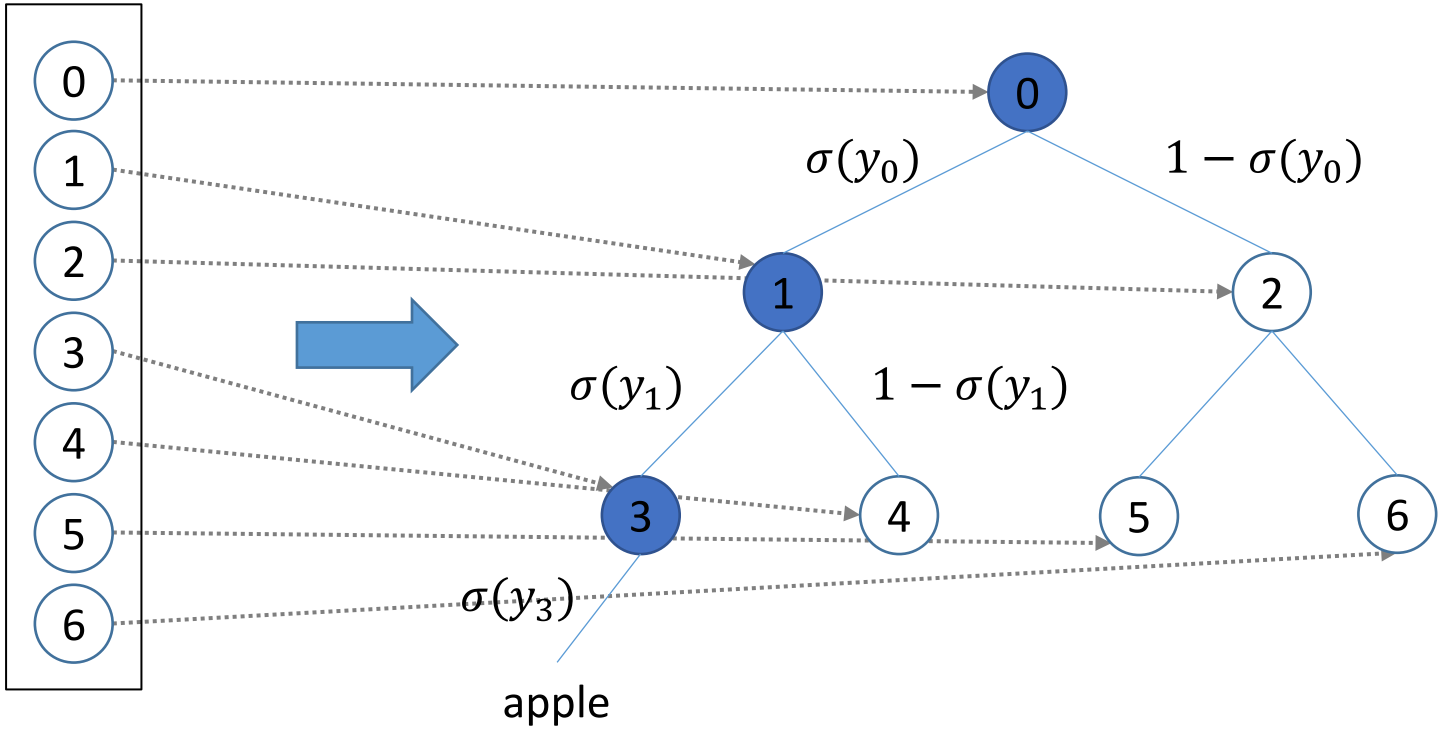


Word2Vec

Hierarchical Softmax

$$p(w|w_I) = \prod_{j=1}^{L(w)-1} \sigma\left([\![n(w, j+1) = \mathrm{ch}(n(w,j))]\!] \cdot {v'_{n(w,j)}}^\top v_{w_I} \right)$$

4. The probability of a word is the product of nodes on the way



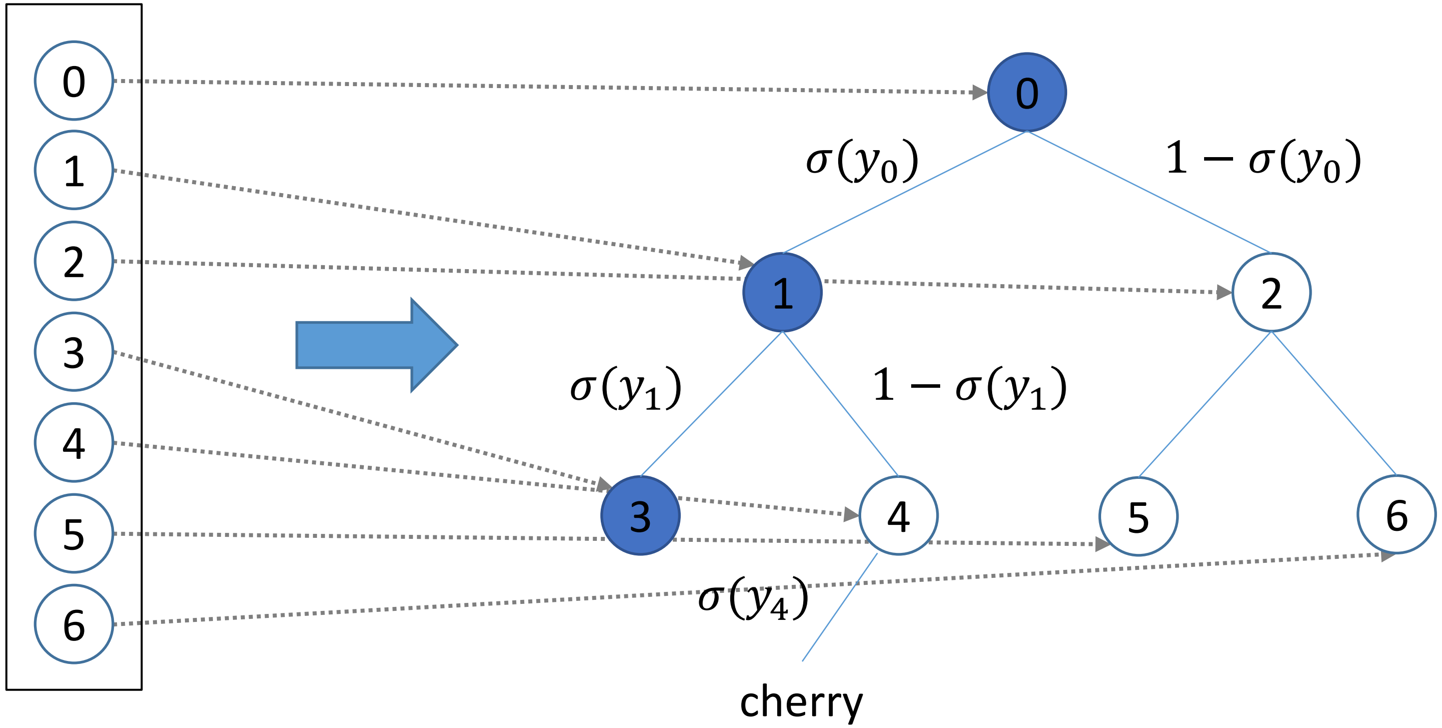
$$p(apple) = \sigma(y_0)\,\sigma(y_1)\,\sigma(y_3)$$

Word2Vec

Hierarchical Softmax

$$p(w|w_I) = \prod_{j=1}^{L(w)-1} \sigma\left([\![n(w, j+1) = \mathrm{ch}(n(w,j))]\!] \cdot {v'_{n(w,j)}}^{\top} v_{w_I} \right)$$

4. The probability of a word is the product of nodes on the way



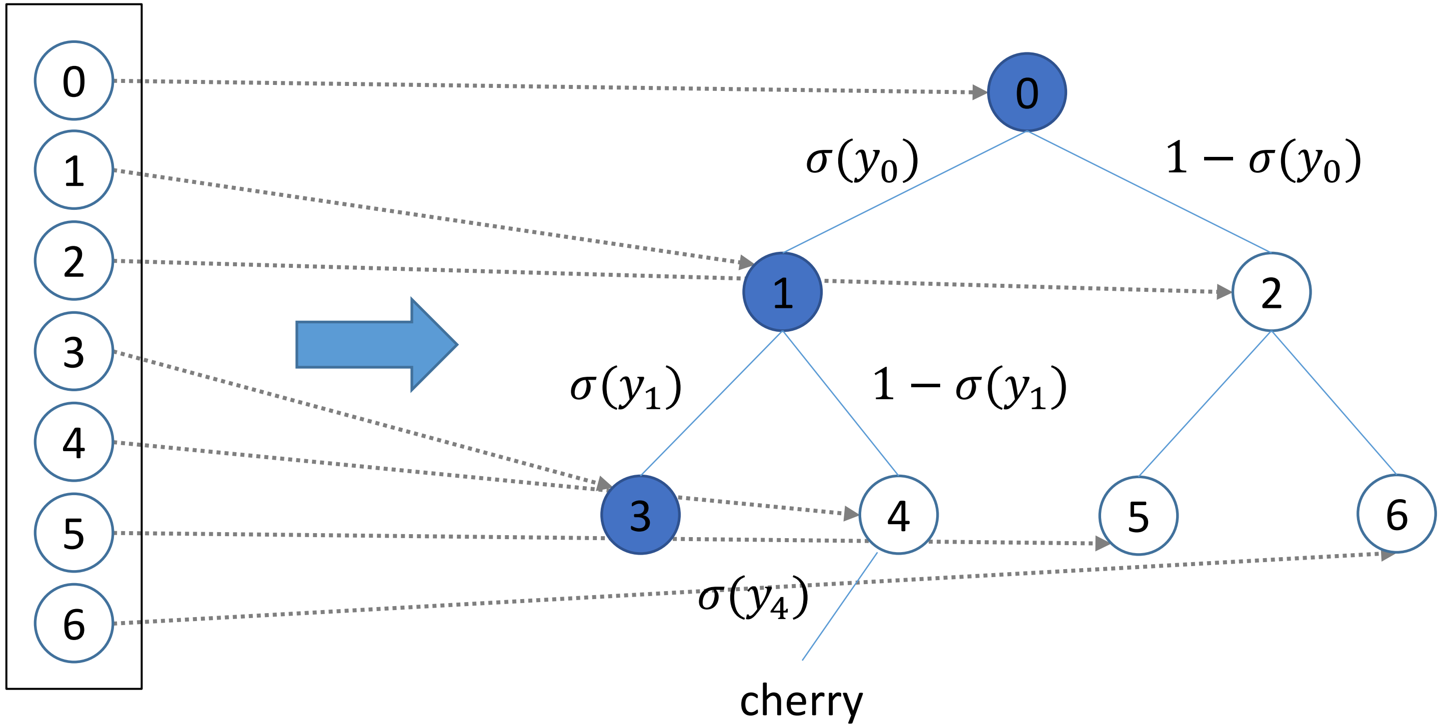
$$p(cherry) = \sigma(y_0)\,(1-\sigma(y_1))\,\sigma(y_4)$$

Word2Vec

Hierarchical Softmax

$$p(w|w_I) = \prod_{j=1}^{L(w)-1} \sigma\left([\![n(w, j+1) = \mathrm{ch}(n(w,j))]\!] \cdot {v'_{n(w,j)}}^{\top} v_{w_I} \right)$$

5. Maximize the probability by gradient descent on negative log likelihood

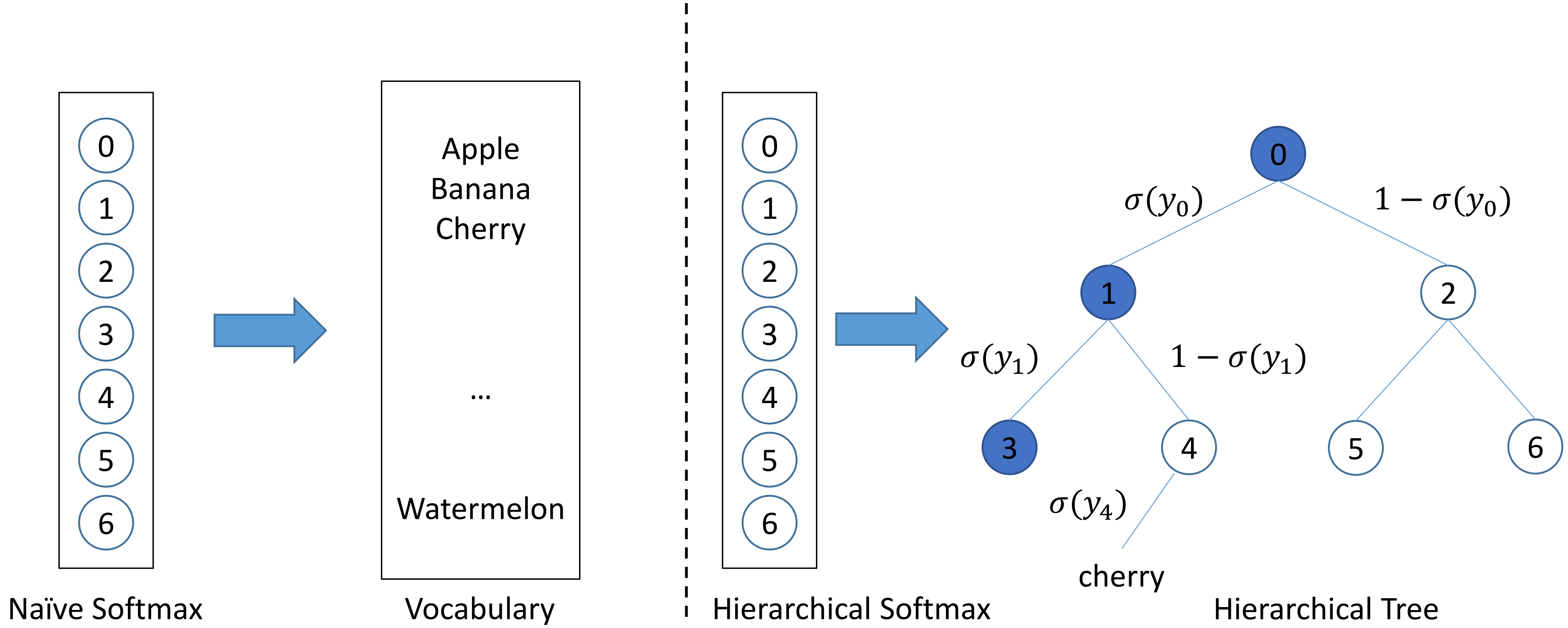


$p(cherry) = \ \sigma(y_0)\,(1-\sigma(y_1))\,\sigma(y_4)$

Minimize $-\log p(cherry)$

Word2Vec

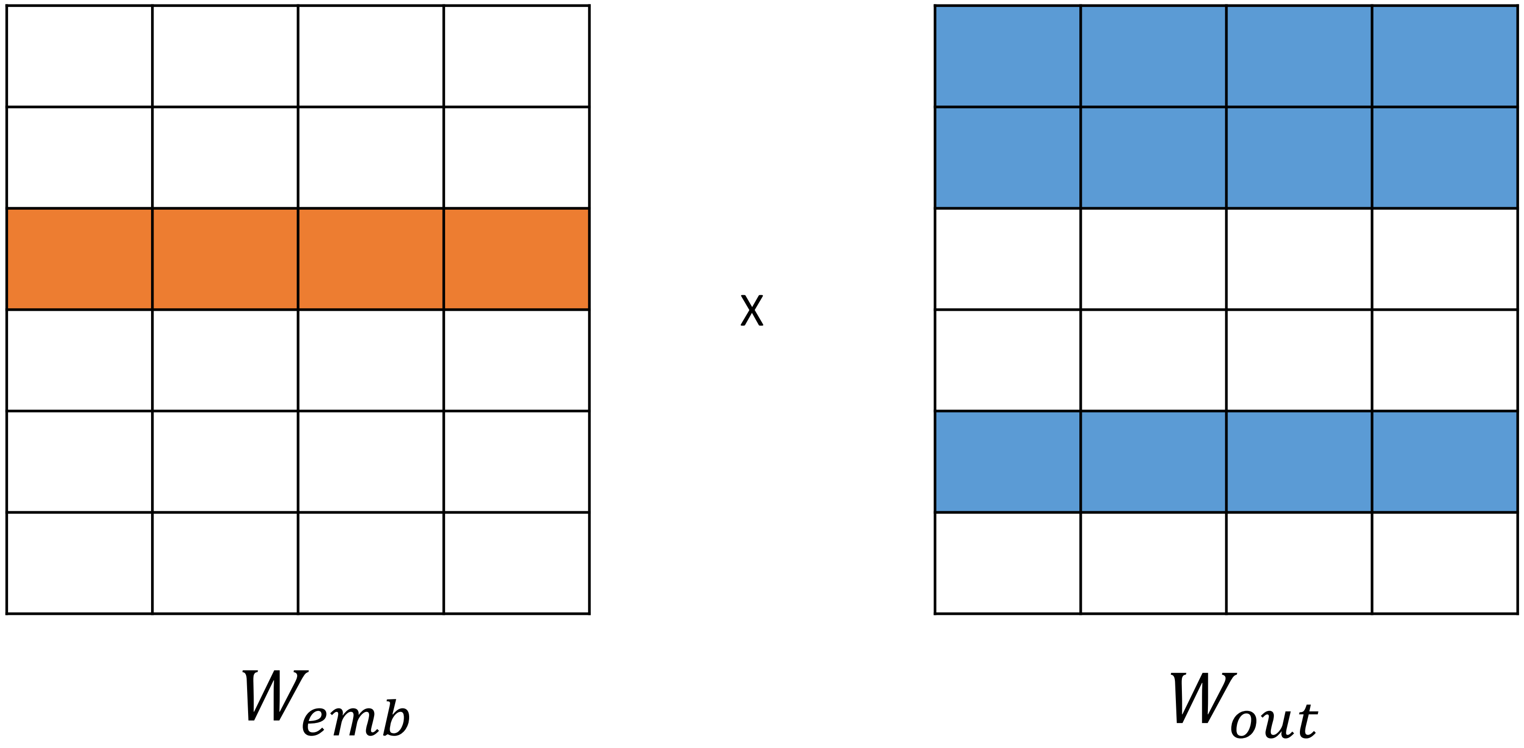
Hierarchical Softmax



Word2Vec

Hierarchical Softmax

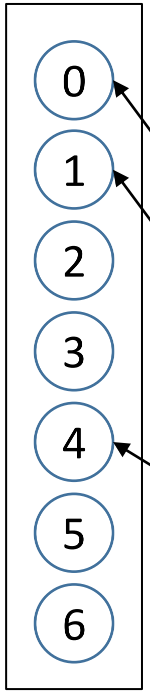
6. Weights connected to the activated nodes are updated



Word2Vec

Hierarchical Softmax

6. Weights connected to the activated nodes are updated



$$p(cherry) = \sigma(y_0)\,(1-\sigma(y_1))\,\sigma(y_4)$$

$$\begin{aligned} \mathrm{L} &= -\log p(cherry) \\ &= -\log\sigma(y_0) - \log\left(1-\sigma(y_1)\right) - \log\sigma(y_4) \end{aligned}$$

$$\frac{\partial L}{\partial y_0} = \sigma(y_0) - 1$$

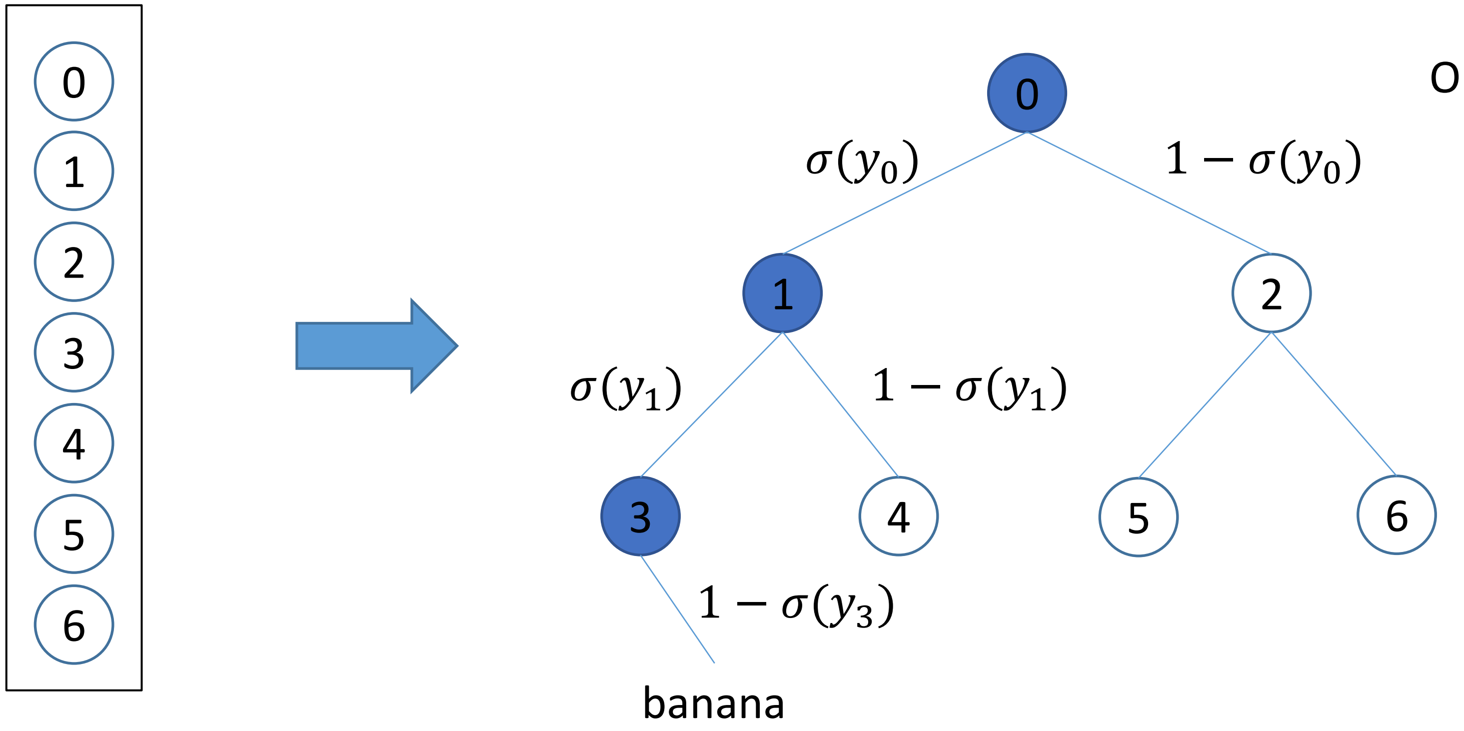
$$\frac{\partial L}{\partial y_1} = \sigma(y_1)$$

$$\frac{\partial L}{\partial y_4} = \sigma(y_4) - 1$$

※ $\frac{\partial \log(\sigma(x))}{\partial x} = 1 - \sigma(x)$

Word2Vec

Hierarchical Softmax



On average, only $\log(V)$ nodes are activated

With 840B dataset

Output dimension : 2.2M
Feature dimension : 300

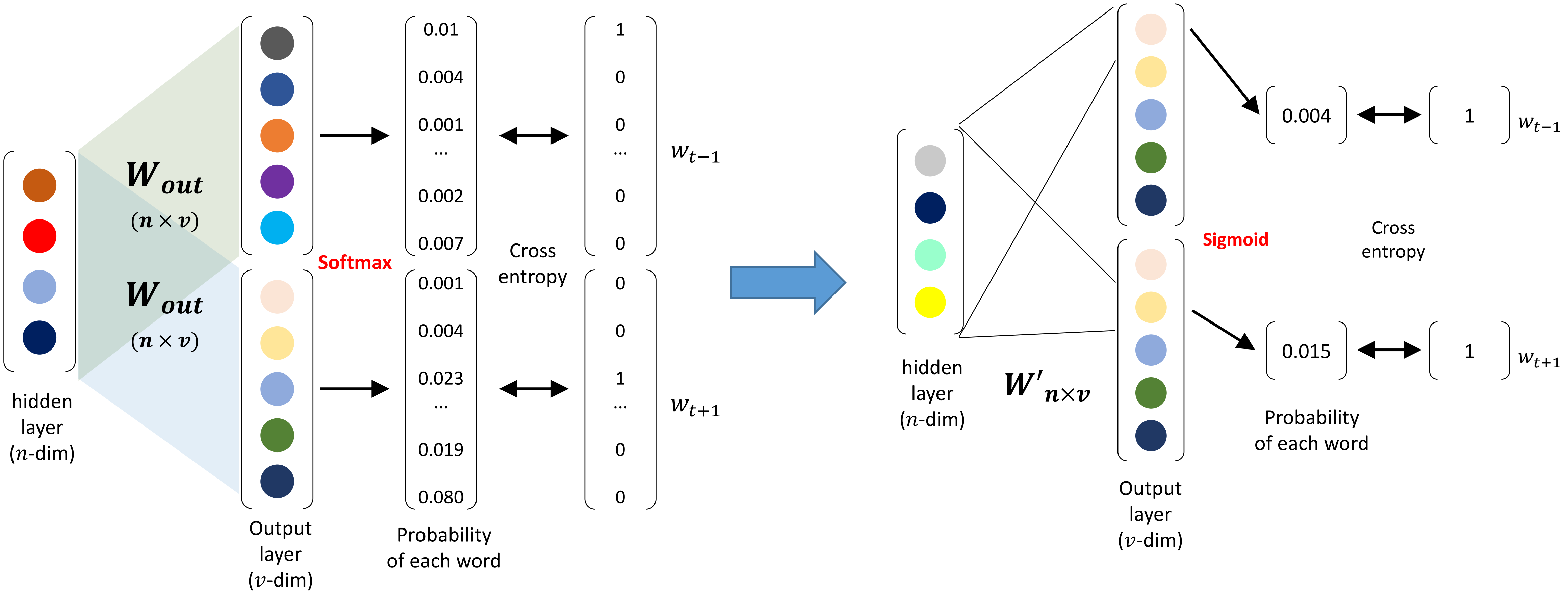
Average activated nodes : 21

6.3k operation to calculate
$y = softmax({W_{out}}^T W_{emb}[k])$

Basic softmax : 660M

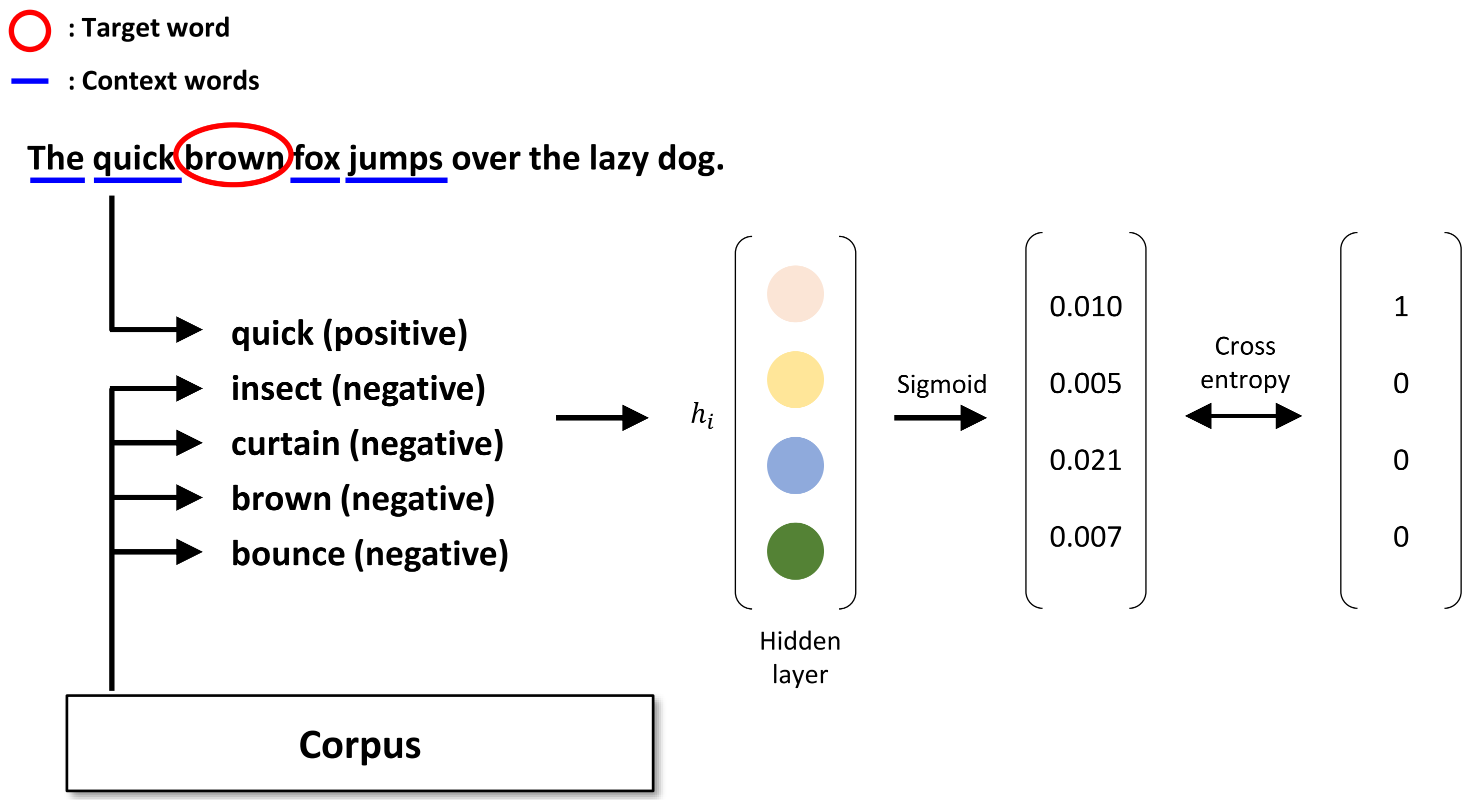
Word2Vec

Negative Sampling



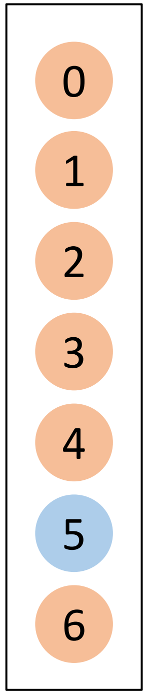
Word2Vec

Negative Sampling



Word2Vec

Negative Sampling



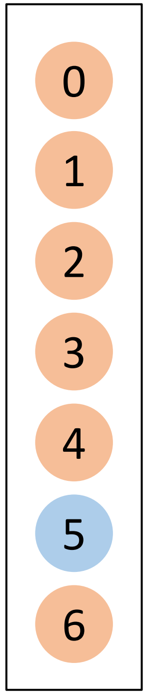
1 of positive sample

V-1 of negative samples

Approximate the softmax function
only using k negative samples

Word2Vec

Negative Sampling



Sigmoid output

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

k negative samples

How many samples

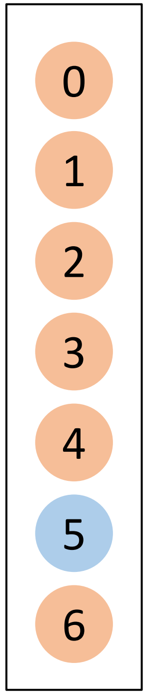
1?
5-10?
Half of the negatives?

How to sample

Uniformly?
Linearly?
With some heuristic function?

Word2Vec

Negative Sampling



Sigmoid output

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

k negative samples

How many samples

5~15 samples recommended
3~5 samples enough on big corpus

How to sample

$$P(w_i) = \frac{f(w_i)^{\frac{3}{4}}}{\sum_{j=0}^{n} f(w_i)^{\frac{3}{4}}}$$

$f(w_i)$ = frequency of the word

Word2Vec

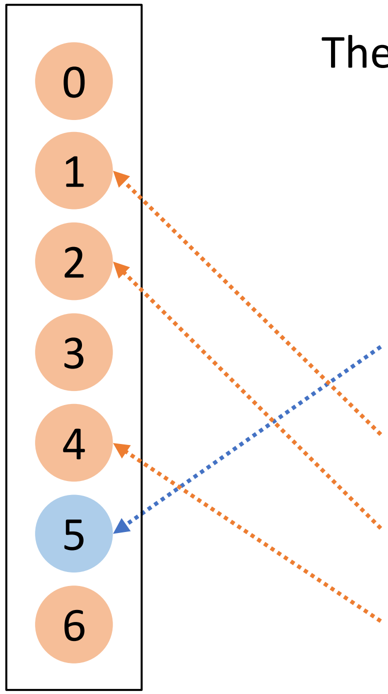
Negative Sampling

1 positive sample

5

k negative samples

1 2 4



Design loss function to maximize the positive and to minimize the negatives

L = -log(5) - log((1- 1)(1- 2)(1- 4))

Then the gradient descent algorithm optimizes the network

$$\mathrm{L} = -\log\sigma(y_5) - \log\left(1-\sigma(y_1)\right) - \log\left(1-\sigma(y_2)\right) - \log\left(1-\sigma(y_4)\right)$$

$$\frac{\partial L}{\partial y_5} = \sigma(y_5) - 1$$

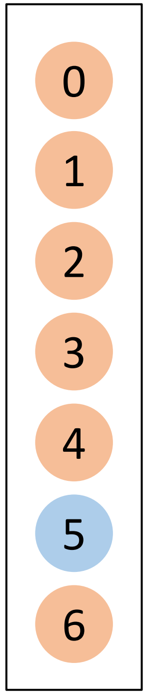
$$\frac{\partial L}{\partial y_1} = \sigma(y_1)$$

$$\frac{\partial L}{\partial y_2} = \sigma(y_2)$$

$$\frac{\partial L}{\partial y_4} = \sigma(y_4)$$

Word2Vec

Negative Sampling



Only k nodes are activated

With 840B dataset

Output dimension : 2.2M
Feature dimension : 300

Average activated nodes : 1 + 5

1.8k operation to calculate
$y = softmax({W_{out}}^T W_{emb}[k])$

Basic softmax : 660M
Hierarchical softmax : 6.3k

Word2Vec

Even faster but..

Method	Time [min]	Syntactic [%]	Semantic [%]	Total accuracy [%]
NEG-5	38	63	54	59
NEG-15	97	63	58	**61**
HS-Huffman	41	53	40	47
NCE-5	38	60	45	53

With 840B dataset

Window size : 5

Basic softmax : 660M x 8.4T
Hierarchical softmax : 6.3k x 8.4T
Negative Sampling : 1.8k x 8.4T

Word2Vec

Another idea is…

The orange is the fruit of the citrus species Citrus × sinensis in the family Rutaceae. It is also called sweet orange, to distinguish it from the related Citrus × aurantium, referred to as bitter orange. The sweet orange reproduces asexually varieties of sweet orange arise through mutations.

Highly frequent words are actually meaningful?

Word2Vec

Subsampling

~~The~~ orange is ~~the~~ fruit of ~~the~~ citrus species Citrus × sinensis in the family Rutaceae. It is also called sweet orange, to distinguish it from the related Citrus × aurantium, referred to as bitter orange. ~~The~~ sweet orange reproduces asexually varieties of sweet orange arise through mutations.

Discard frequent words with probability

$$P(w_i) = 1 - \sqrt{\frac{t}{f(w_i)}}$$

t = threshold

Word2Vec

Method	Time [min]	Syntactic [%]	Semantic [%]	Total accuracy [%]
NEG-5	38	63	54	59
NEG-15	97	63	58	**61**
HS-Huffman	41	53	40	47
NCE-5	38	60	45	53
The following results use 10^{-5} subsampling				
NEG-5	14	61	58	60
NEG-15	36	61	61	**61**
HS-Huffman	21	52	59	55

Assignment 5

- Word2Vec Implementation
  - Hierarchical Softmax
    - Assign binary code(Huffman coding)
    - Train with only weights connected to the activated nodes
    - Return : cost value and gradient of two word vectors
  - Negative Sampling
    - Frequency table
    - Random sampling during training
    - Return : cost value and gradient of two word vectors
  - Subsampling
    - Read(preprocess) corpus and make dictionary
    - Subsample corpus in every epoch

Assignment 5

- Activated Weight Matrix

```
if mode == "CBOW":
    if ns == 0:

        # Only use the activated rows of the weight matrix
        nodes = torch.cuda.LongTensor(ind2node[centerInd.item()][0])
        codes = torch.cuda.LongTensor(ind2node[centerInd.item()][1])
        L, G_emb, G_out = CBOW_HS(centerInd, contextInds, codes, W_emb, W_out[nodes])
```

```
W_emb[contextInds] -= lr * G_emb
W_out[nodes] -= lr * G_out
losses.append(L.item())
```

Recommend to use a portion of W_out for the computational efficiency

Assignment 5

- Hierarchical Softmax
  - Use given “huffman.py”
    - How to use
      - HuffmanCode().build(frequency)
      - Input: Dictionary(key: word, value: frequency)
      - Output: Dictionary(key: word, value: code), Dictionary(key: code, value: ID number)

Assignment 5

- Word2Vec Experiment

Analogical reasoning task[1][2]

"Germany" : "Berlin" :: "France" : ?

vec(x) =vec("Berlin") - vec("Germany") + vec("France")

Find the word x using cosine similarity

[1] http://code.google.com/p/word2vec/source/browse/trunk/questions-words.txt
[2] Tomas Mikolov et al. Distributed Representations of Words and Phrases and their Compositionality, 2013

Assignment 5

- Word2Vec Experiment
  - In this assignment, 9 types are used

Man-Woman	brother	sister	grandson	granddaughter
Adjective to adverb	apparent	apparently	rapid	rapidly
Opposite	possibly	impossibly	ethical	unethical
Comparative	great	greater	tough	tougher
Superlative	easy	easiest	lucky	luckiest
Present Participle	think	thinking	read	reading
Past tense	walking	walked	swimming	swam
Plural nouns	mouse	mice	dollar	dollars
Plural verbs	work	works	speak	speaks

Total 36 questions

work :: works = speak :: speaks

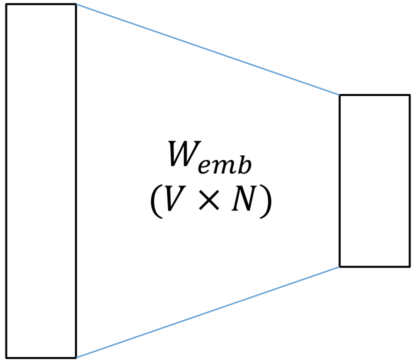
- works – work + speak
- work – works + speaks
- speaks – speak + work
- speak – speaks + works

Report top 5 accuracy
Any of 5 predictions is correct -> correct
None of 5 predictions is correct -> wrong

Assignment 5

- Analogy task

work::works = speak:speaks
works-work+speak= ??



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

$$v_{works} - v_{work} + v_{speak} = v_i$$

$$w_{emb} \times v_i = w_{sim}$$

find top-5 values in w_{sim}.

※ Regularize the W_{emb} .

Assignment 5

- Word2Vec Experiment

Analogical reasoning task[1][2]

- CBOW or Skip-gram
- Hierarchical Softmax or Negative Sampling
- Subsampling or not

[1] http://code.google.com/p/word2vec/source/browse/trunk/questions-words.txt
[2] Tomas Mikolov et al. Distributed Representations of Words and Phrases and their Compositionality, 2013

Assignment 5

- Evaluation report

Word2vec Evaluation Report										
	Hierarchical Softmax	Negative Sampling	# of negative samples	Subsampling	Learning rate	Learning rate decay(O/X)	dimension	iteration	training time	Accuracy
setting #1	X	X	-	X			300			
setting #2	X	X	-	O			300			
setting #3	X	O		X			300			
setting #4	X	O		O			300			
setting #5	O	X	-	X			300			
setting #6	O	X	-	O			300			

[결과 정리]

Submission

- Due Date : ~6/7(Sun) 23:59
- Submission : Online submission on blackboard
- word2vec.py + Evaluation Report (analysis of word analogy task)
- Report should include:

  1. Whole implementation the code
  2. Evaluation Report
- You must implement the components yourself!
- You must specify each member's contribution (role) in this assignment.
- File name : StudentID_Name.zip

Q & A

조교 김도현 : dhkim1028@korea.ac.kr