Word2vec

박준형
데이터인텔리전스 연구실

irish07@korea.ac.kr

Class Lab - Schedule & Assignment

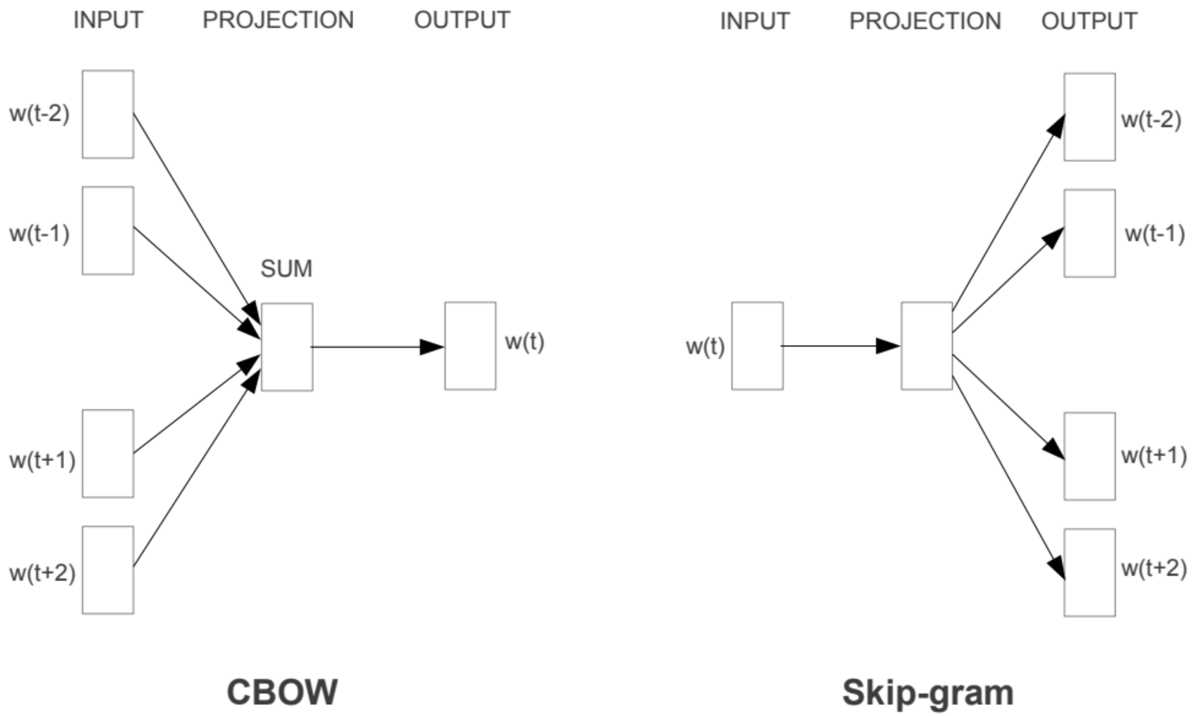
1. Neural Network Introduction (~10/14)

2. Skip-gram / CBOW (~10/30)
   (Basic) Softmax

3. Hierarchical Softmax / Negative sampling (~11/13)
   Subsampling

Class Lab - Schedule & Assignment

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



Class Lab - Schedule & Assignment

- T. Mikolov, I. Sutskever, K. Chen, G. Corrado, J. Dean, "Distributed Representations of Words and Phrases and their Compositionality", NIPS 2013

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

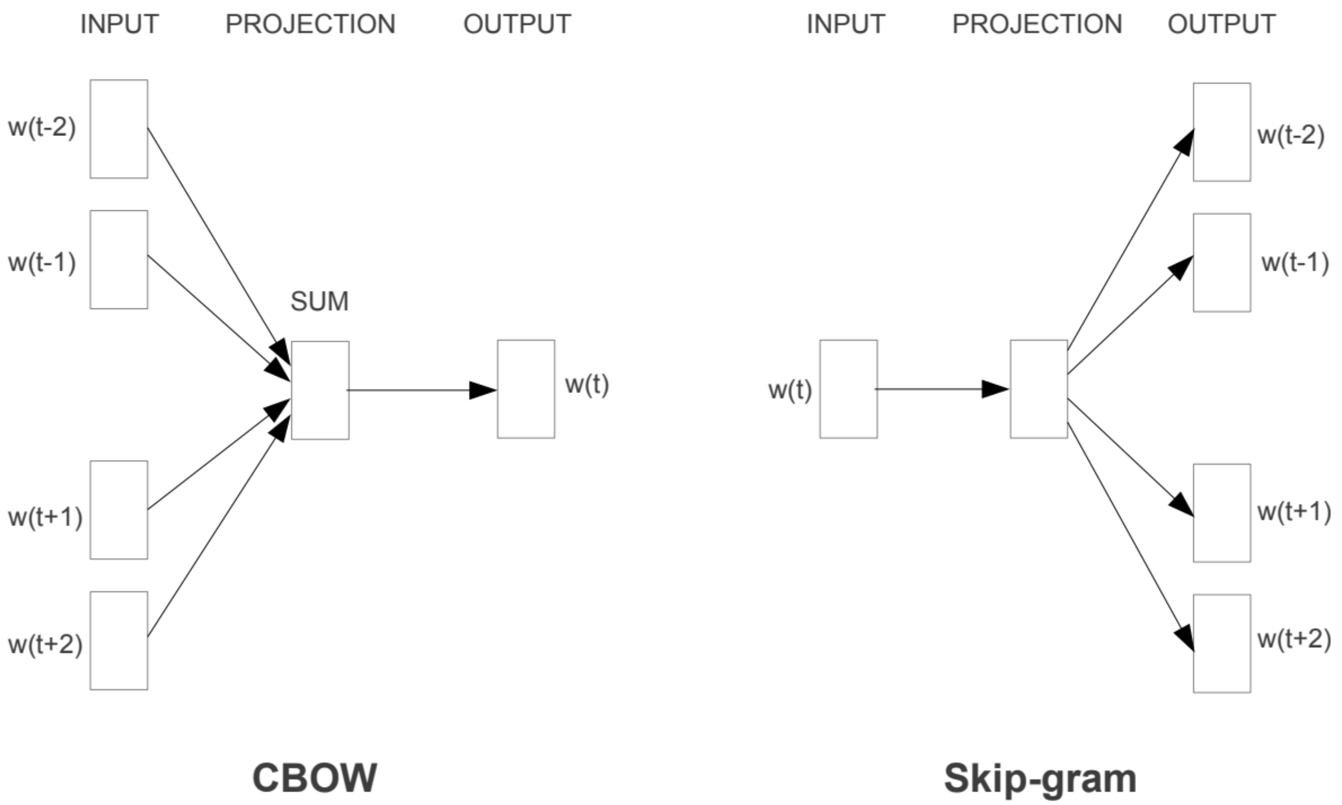
Word2Vec

1. Determine forms of input and output

2. Define loss function

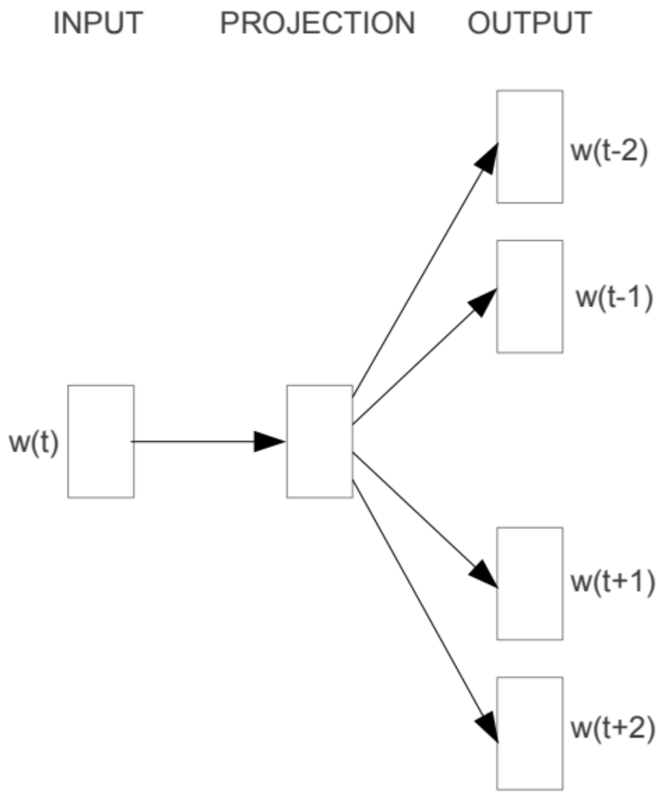
3. Training

Word2Vec


INPUT
PROJECTION
OUTPUT
w(t-2)
w(t-1)
SUM
w(t)
w(t+1)
w(t+2)
CBOW
INPUT
PROJECTION
OUTPUT
w(t)
w(t-2)
w(t-1)
w(t+1)
w(t+2)
Skip-gram

Word2Vec

Skip-gram



Skip-gram

Predict context words using a center word

I eat an **orange** every day.

context words　　context words

Predict the probability

???	???	???	**orange**	???	???

Word2Vec

Skip-gram

1. Word encoding

I eat an **orange** every day.

orange

Parameterize

Word vector

Word2Vec

Skip-gram

HW1

2. Predict

Word vector

Probabilities

- Each element represent probability of a word

Word2Vec

Skip-gram

3. Update

I eat an **orange** every day.

Answer: I

$p(\text{word}_1)$
$p(\text{word}_2)$
$p(\text{word}_3)$

...

$p(\text{word}_n)$

Probabilities

HW1

- Negative Log Likelihood Loss
- Backpropagation
- Stochastic Gradient Descent

Weight Matrix

Word2Vec

Skip-gram

3. Update

I eat an **orange** every day.

Answer: eat

$p(\text{word}_1)$
$p(\text{word}_2)$
$p(\text{word}_3)$

…

$p(\text{word}_n)$

Probabilities

HW1

- Negative Log Likelihood Loss
- Backpropagation
- Stochastic Gradient Descent

Weight Matrix

Word2Vec

Skip-gram

3. Update

I eat an **orange** every day.

Answer: an

$p(\text{word}_1)$
$p(\text{word}_2)$
$p(\text{word}_3)$

...

$p(\text{word}_n)$

Probabilities

HW1

- Negative Log Likelihood Loss
- Backpropagation
- Stochastic Gradient Descent

Weight Matrix

Word2Vec

Skip-gram

3. Update

I eat an **orange** every day.

Answer: every

$p(\text{word}_1)$
$p(\text{word}_2)$
$p(\text{word}_3)$

...

$p(\text{word}_n)$

Probabilities

HW1

- Negative Log Likelihood Loss
- Backpropagation
- Stochastic Gradient Descent

Weight Matrix

Word2Vec

Skip-gram

3. Update

I eat an **orange** every day.

Answer: day

$p(\text{word}_1)$
$p(\text{word}_2)$
$p(\text{word}_3)$

...

$p(\text{word}_n)$

Probabilities

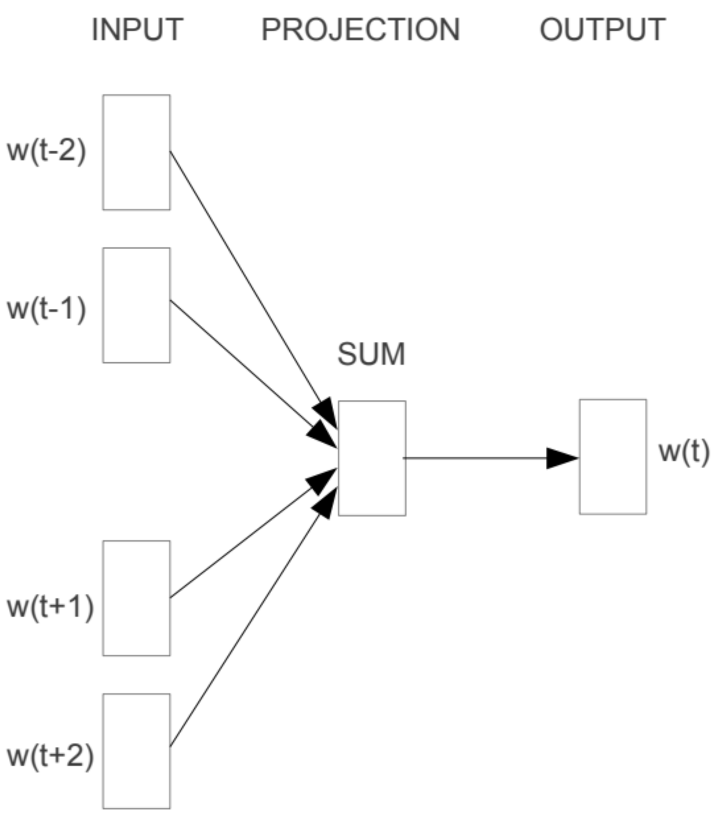
HW1

- Negative Log Likelihood Loss
- Backpropagation
- Stochastic Gradient Descent

Weight Matrix

Word2Vec

Continuous Bag of Words

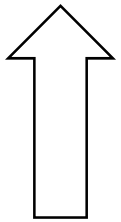


CBOW

How frequent the center word occurs in some context?

I eat an **orange** every day.

center word



Predict the probability

I eat an **???** every day.

Word2Vec

Continuous Bag of Words

1. Context encoding

I eat an **???** every day.

I
eat
an
every
day

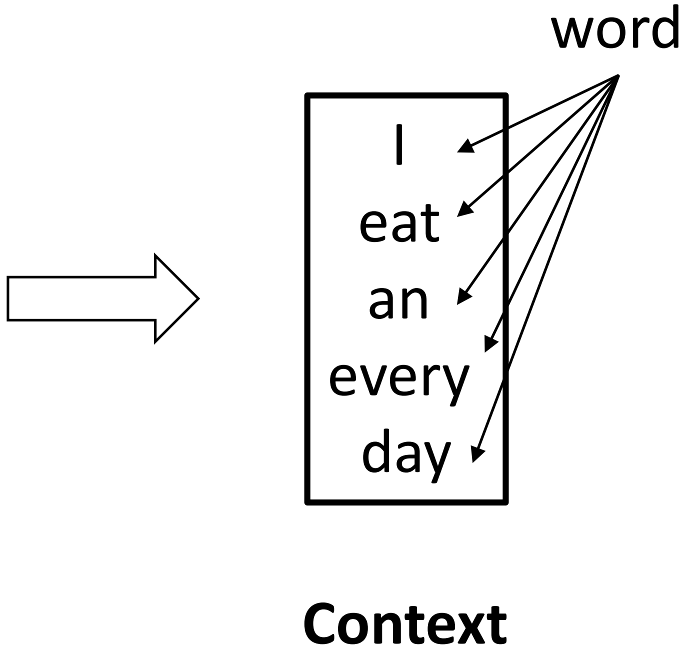
Context

Word2Vec

Continuous Bag of Words

1. Context encoding

I eat an **???** every day.

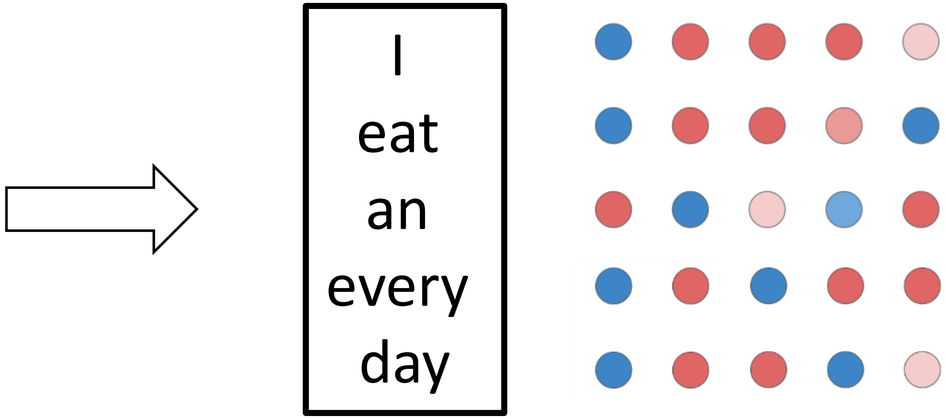


Word2Vec

Continuous Bag of Words

1. Context encoding

I eat an **???** every day.



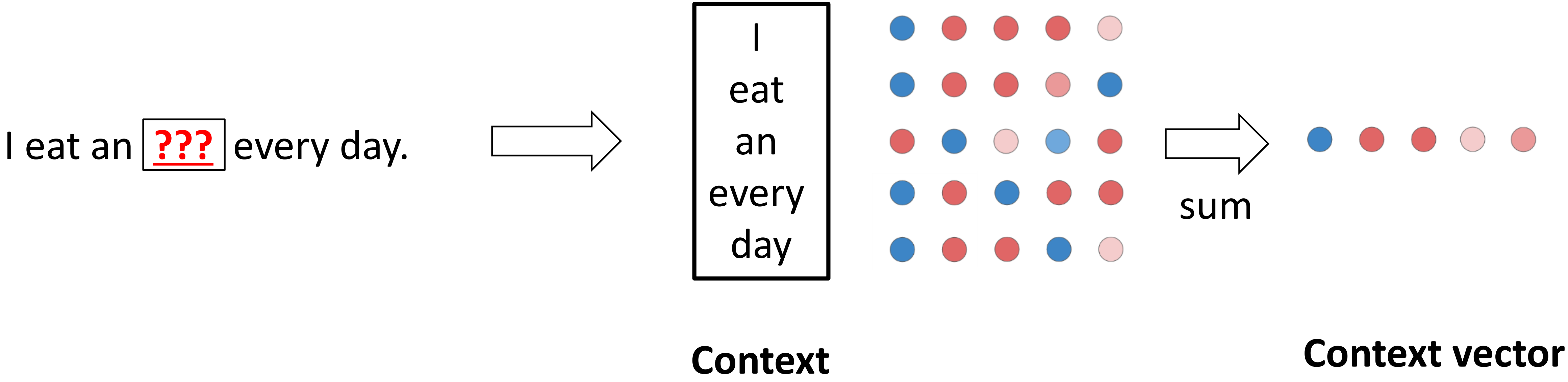
Parameterize

Context

Word2Vec

Continuous Bag of Words

1. Context encoding



Word2Vec

Continuous Bag of Words

2. Prediction

HW1

Context vector

Probabilities

- Each element represent probability of a word

Word2Vec

Continuous Bag of Words

3. Update

I eat an $\boxed{???}$ every day.

Answer: orange

$p(\text{word}_1)$
$p(\text{word}_2)$
$p(\text{word}_3)$

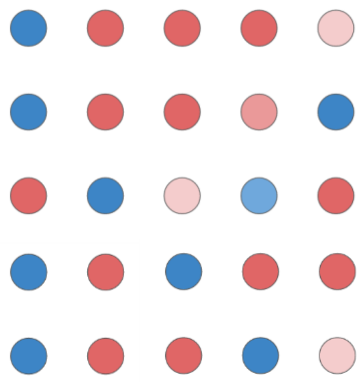
...

$p(\text{word}_n)$

Probabilities

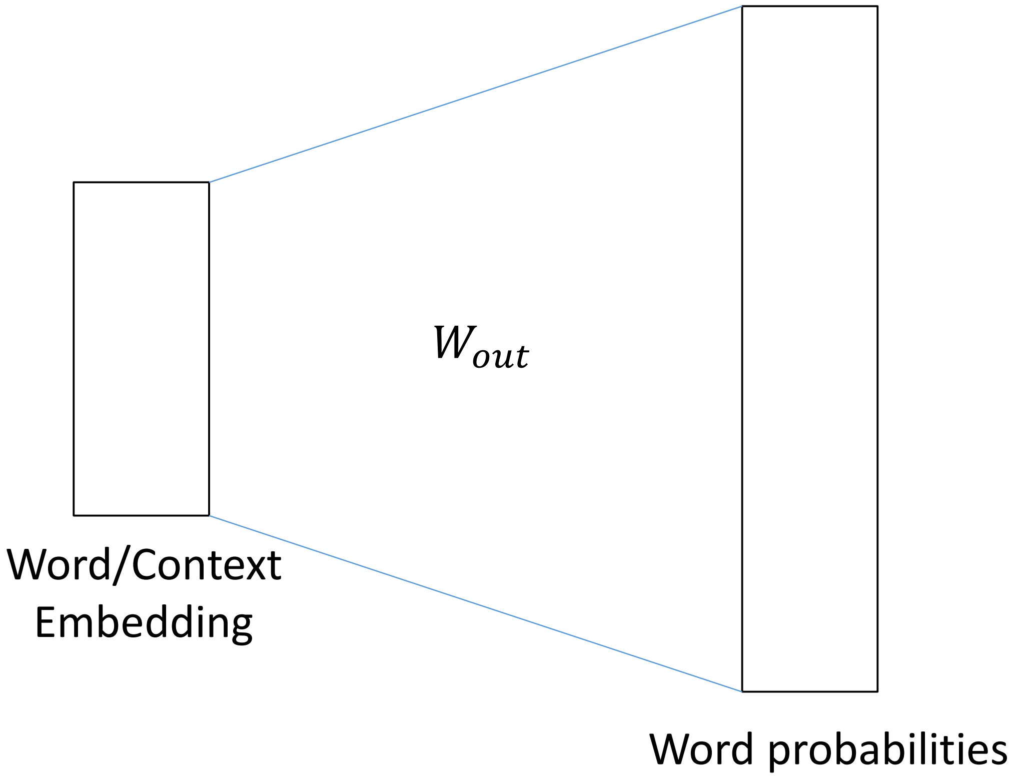
HW1

- Negative Log Likelihood Loss
- Backpropagation
- Stochastic Gradient Descent

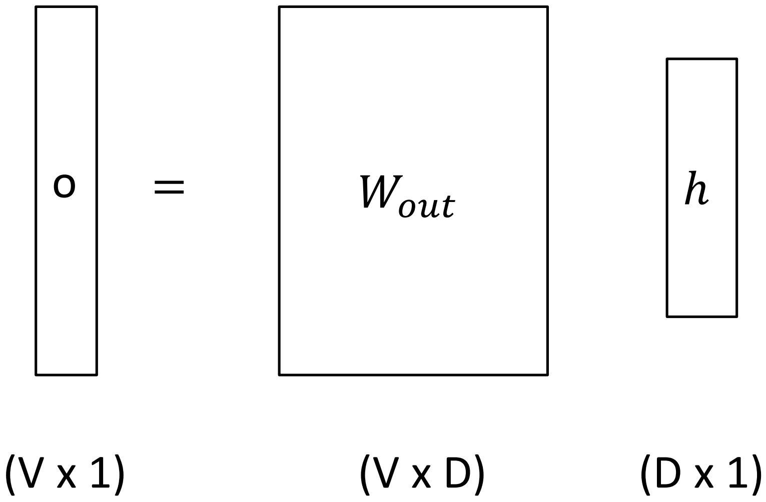


Weight Matrix

Word2Vec

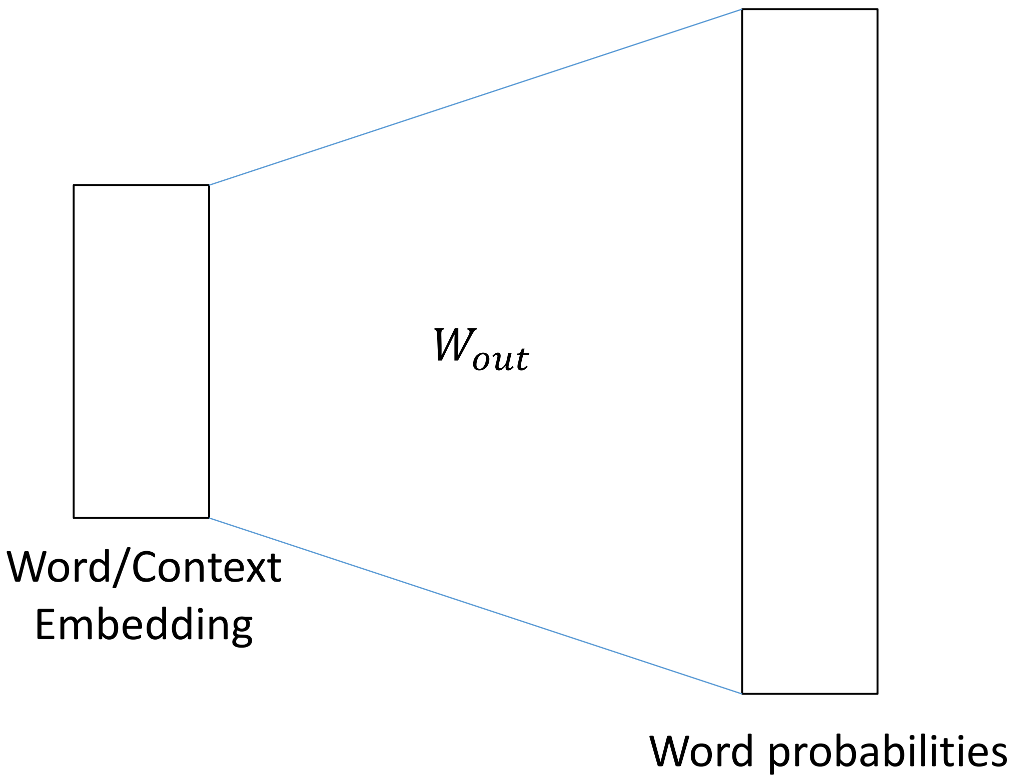


$$y = softmax(W_{out}h)$$



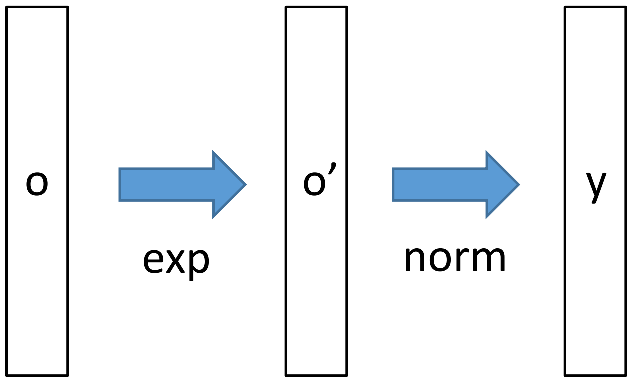
V: vocabulary size
h: embedding dimension

Word2Vec



$$y = softmax(W_{out}h)$$

$$\sigma(\mathbf{z})_j = \frac{e^{z_j}}{\sum_{k=1}^{K} e^{z_k}} \quad \text{for } j = 1, \ldots, K.$$



*Xin Rong. word2vec Parameter Learning Explained

Word2Vec

$$o = W_{out}h$$

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

$$L = NLL(y,t) = -\log(y_t)$$

$$\frac{\partial L}{\partial h} = W_{out}\ e$$

$$\frac{\partial L}{\partial W_{out}} = he^T$$

CBOW

$$h = w_a + w_b + w_c + w_d$$

$$\frac{\partial h}{\partial w_a}, \frac{\partial h}{\partial w_b}, \frac{\partial h}{\partial w_c}, \frac{\partial h}{\partial w_d} = 1$$

$$w_a = w_a - \eta \frac{\partial L}{\partial w_a}$$

$$w_b = w_b - \eta \frac{\partial L}{\partial w_b}$$

$$w_c = w_c - \eta \frac{\partial L}{\partial w_c}$$

$$w_d = w_d - \eta \frac{\partial L}{\partial w_d}$$

$$W_{out} = W_{out} - \eta \frac{\partial L}{\partial W_{out}}$$

Skip-gram

$$h = w_k$$

$$w_k = w_k - \eta \frac{\partial L}{\partial h}$$

$$W_{out} = W_{out} - \eta \frac{\partial L}{\partial W_{out}}$$

Word2Vec

CBOW vs Skip-gram

Model	Vector Dimensionality	Training words	Accuracy [%]			Training time [days]
			Semantic	Syntactic	Total	
3 epoch CBOW	300	783M	15.5	53.1	36.1	1
3 epoch Skip-gram	300	783M	50.0	55.9	53.3	3
1 epoch CBOW	300	783M	13.8	49.9	33.6	0.3
1 epoch CBOW	300	1.6B	16.1	52.6	36.1	0.6
1 epoch CBOW	600	783M	15.4	53.3	36.2	0.7
1 epoch Skip-gram	300	783M	45.6	52.2	49.2	1
1 epoch Skip-gram	300	1.6B	52.2	55.1	53.8	2
1 epoch Skip-gram	600	783M	56.7	54.5	55.5	2.5

Word2Vec

Better and Faster

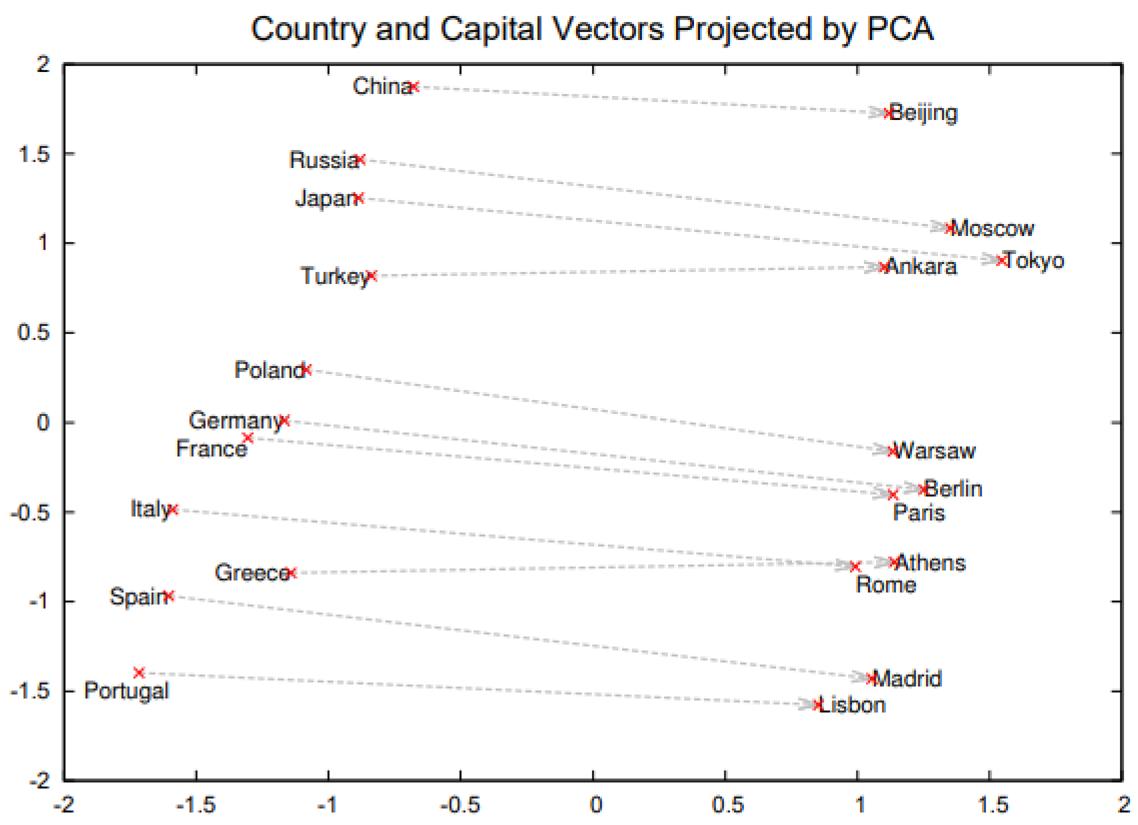
Model Architecture	Semantic-Syntactic Word Relationship test set		MSR Word Relatedness Test Set [20]
	Semantic Accuracy [%]	Syntactic Accuracy [%]	
RNNLM	9	36	35
NNLM	23	53	47
CBOW	24	64	61
Skip-gram	55	59	56

Model	Vector Dimensionality	Training words	Accuracy [%]			Training time [days x CPU cores]
			Semantic	Syntactic	Total	
NNLM	100	6B	34.2	64.5	50.8	14 x 180
CBOW	1000	6B	57.3	68.9	63.7	2 x 140
Skip-gram	1000	6B	66.1	65.1	65.6	2.5 x 125

Word2Vec

Additive Compositionality

vec("Paris") - vec("France")
= vec("Berlin") - vec("Germany")



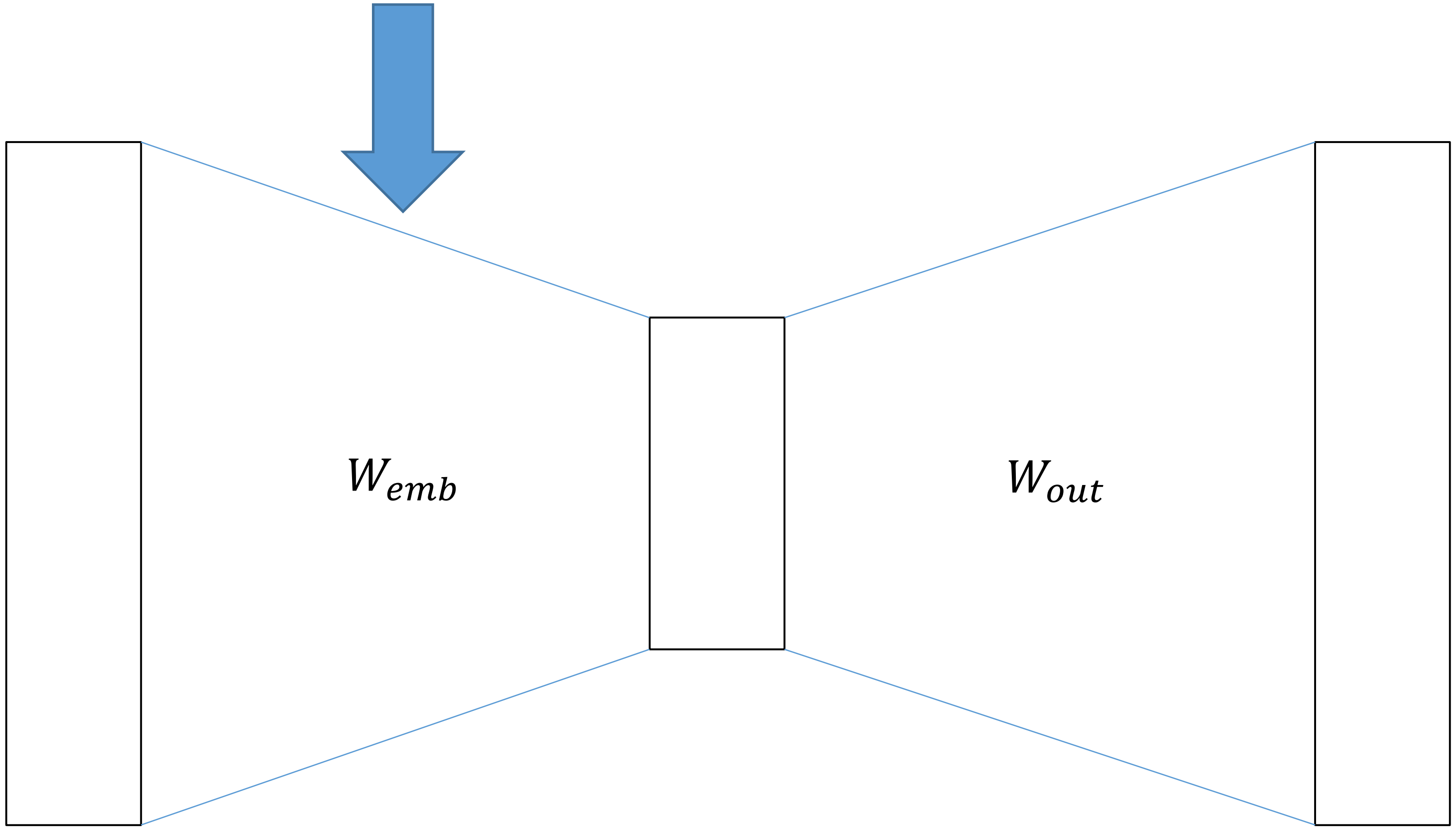
Word2Vec

- Word2vec is very slow…

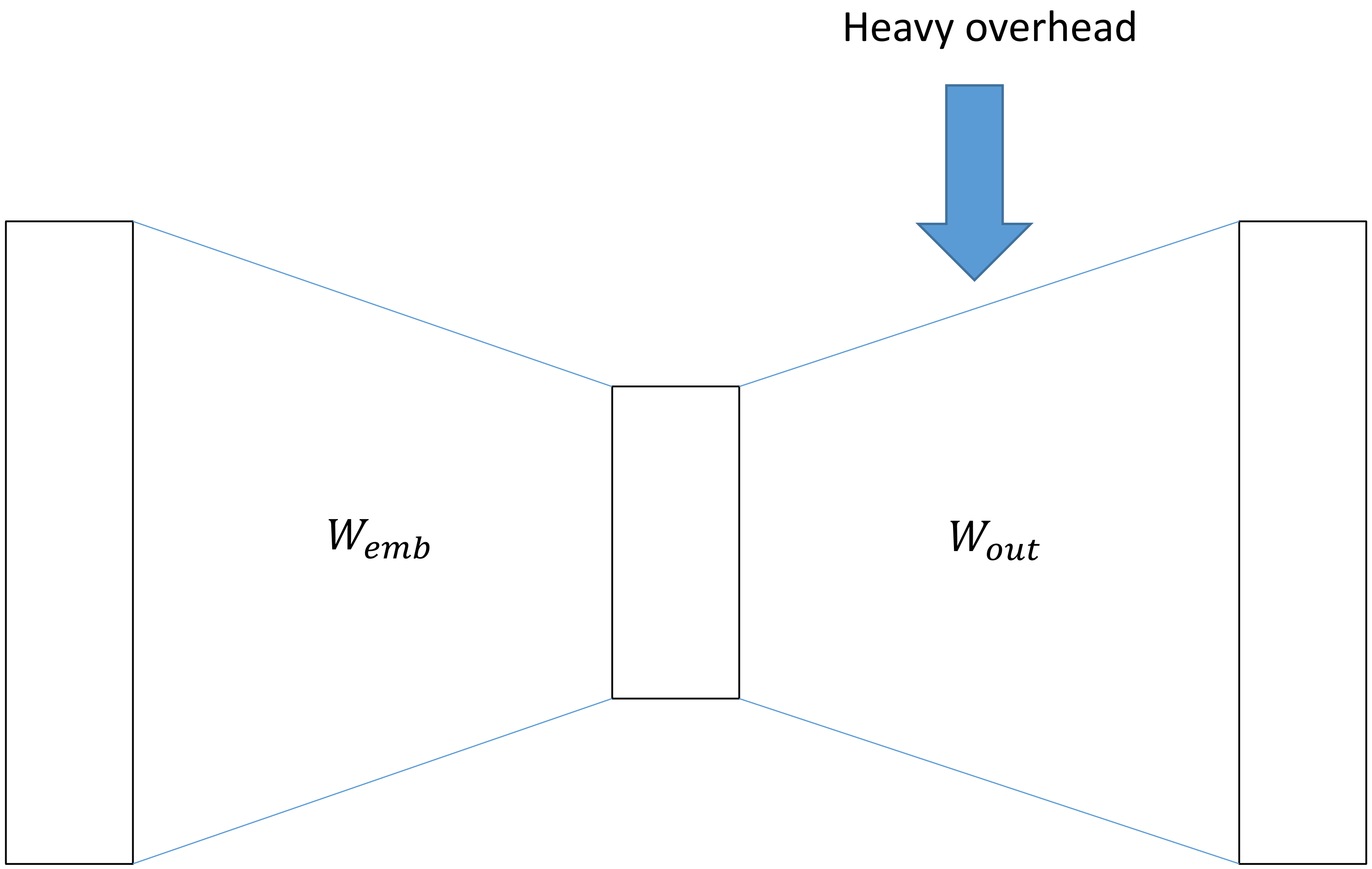
Why?

Word2Vec

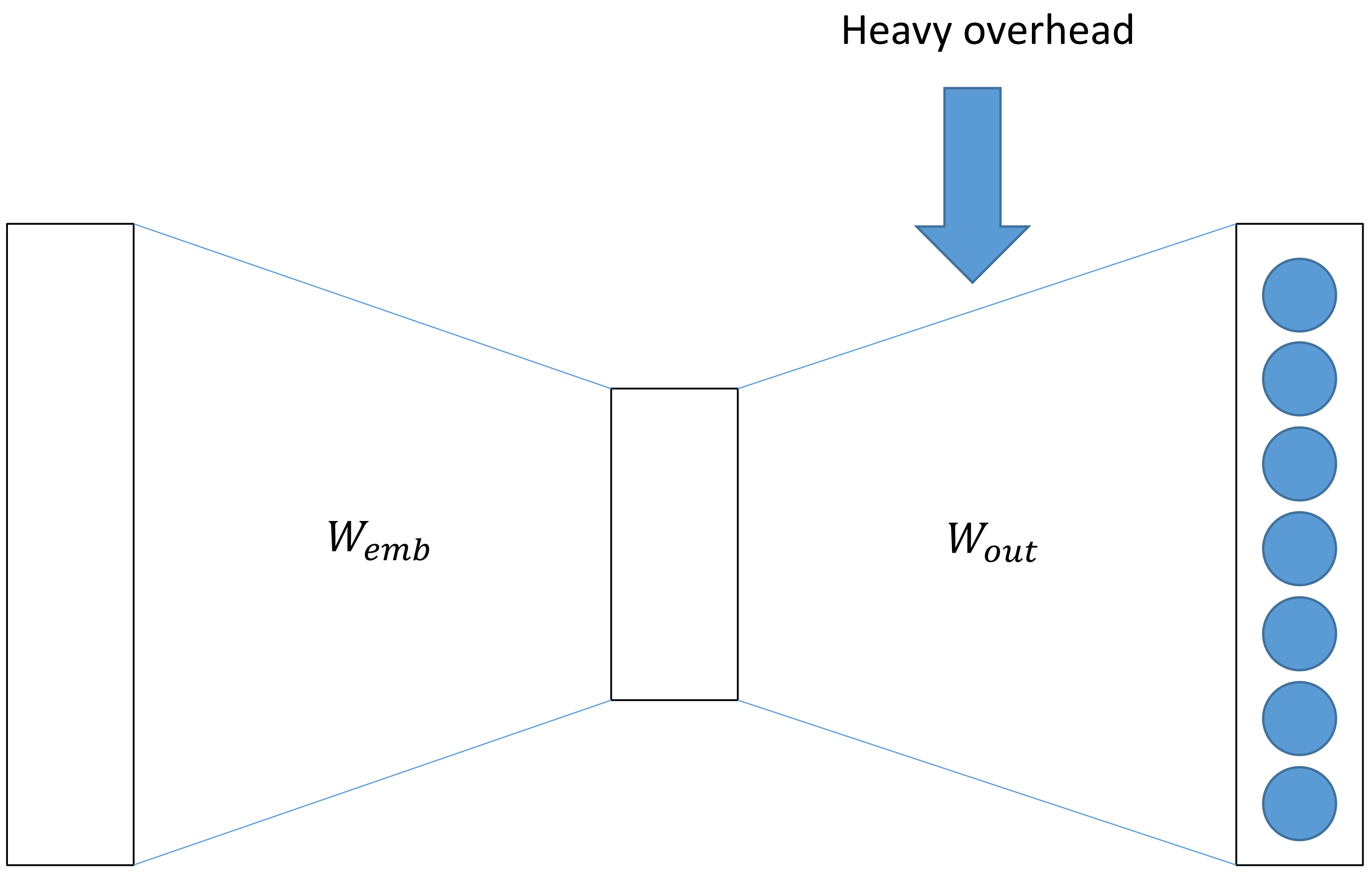
No overhead(just load a vector instead of matrix multiplication)



Word2Vec



Word2Vec

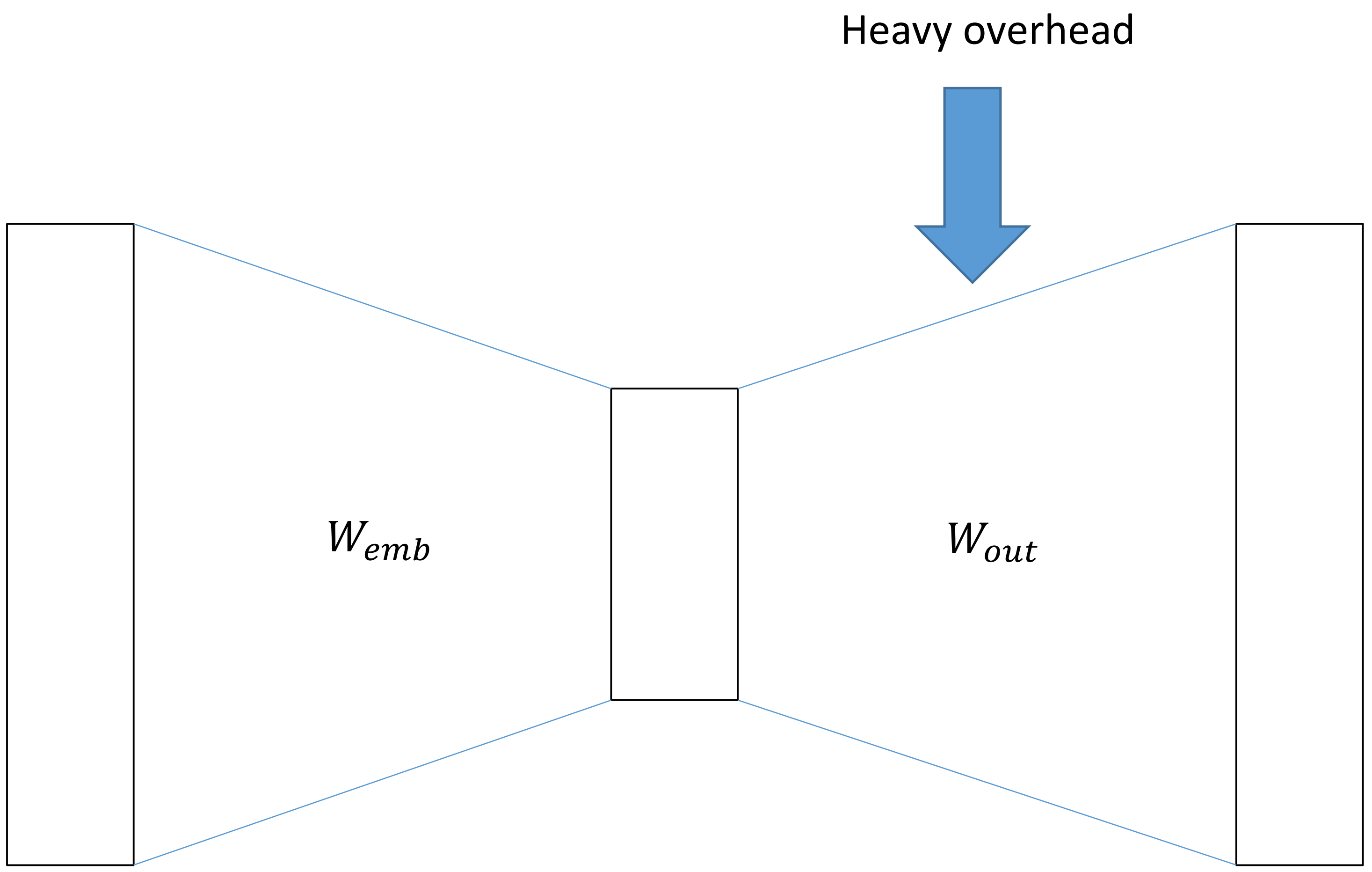


The reason is…

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

Softmax function needs all values of the output vector

Word2Vec



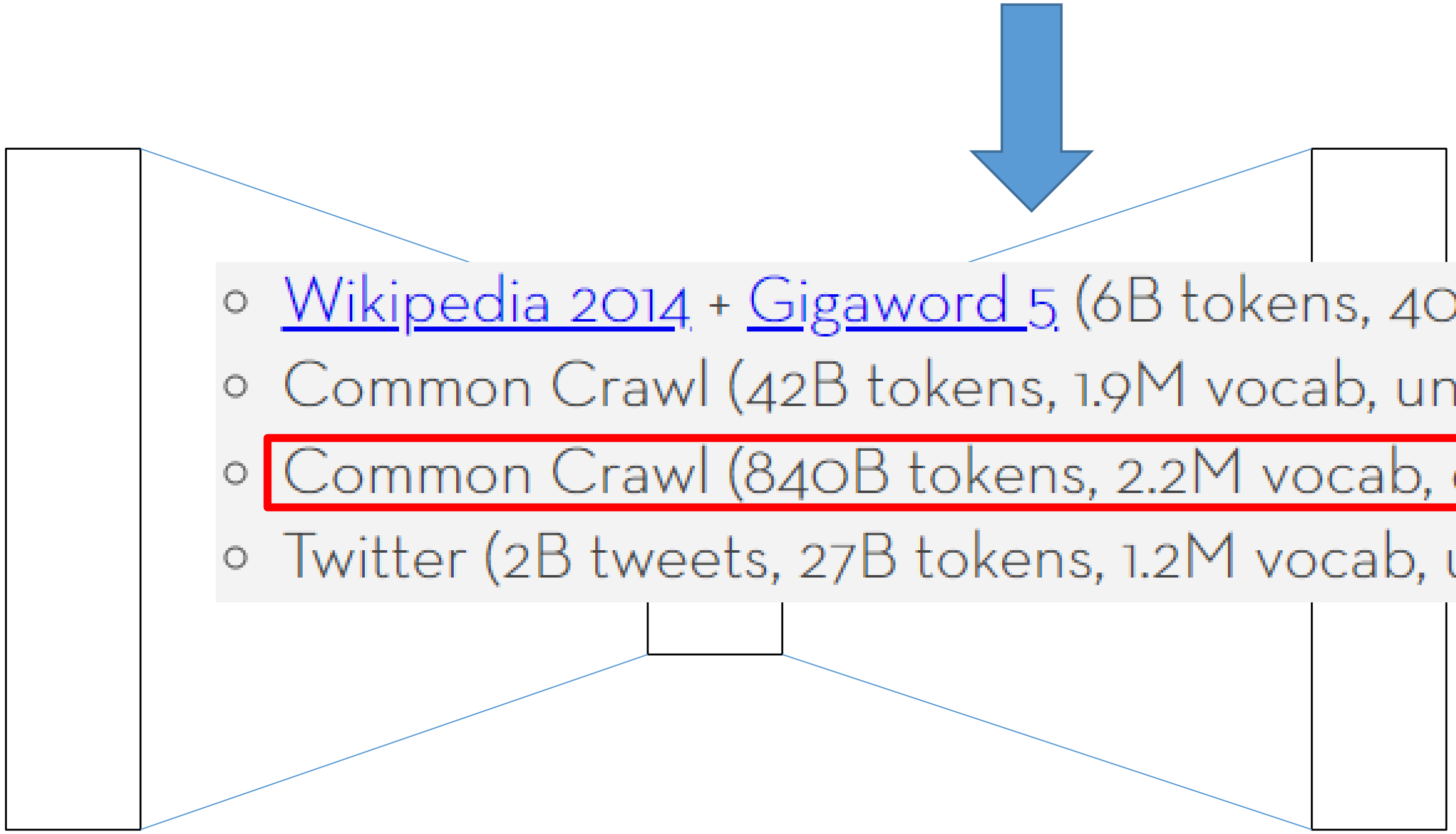
The reason is…

Output dimension : V
Feature dimension : D

Complexity : O(V x D)

Word2Vec

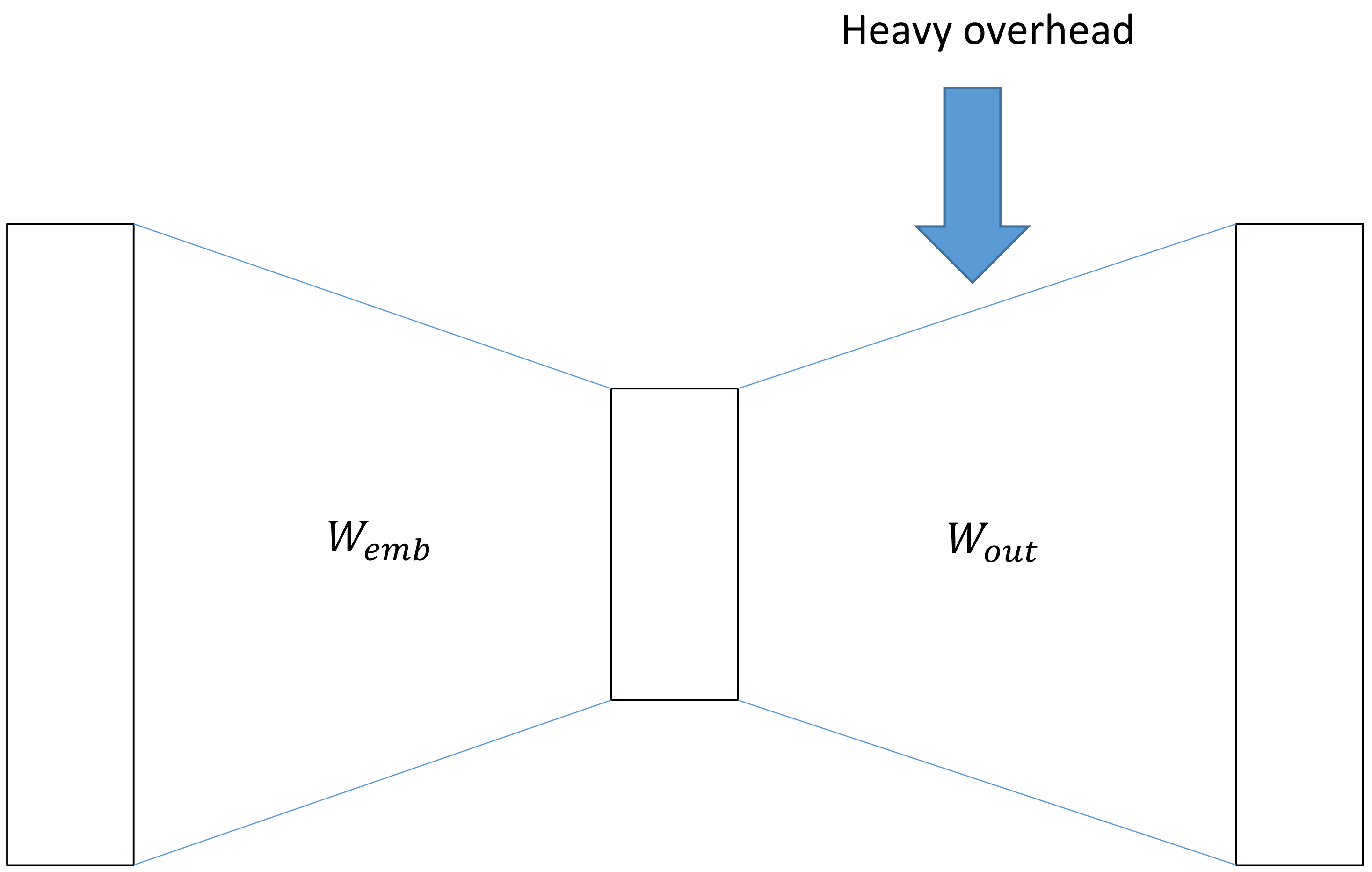
Heavy overhead



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

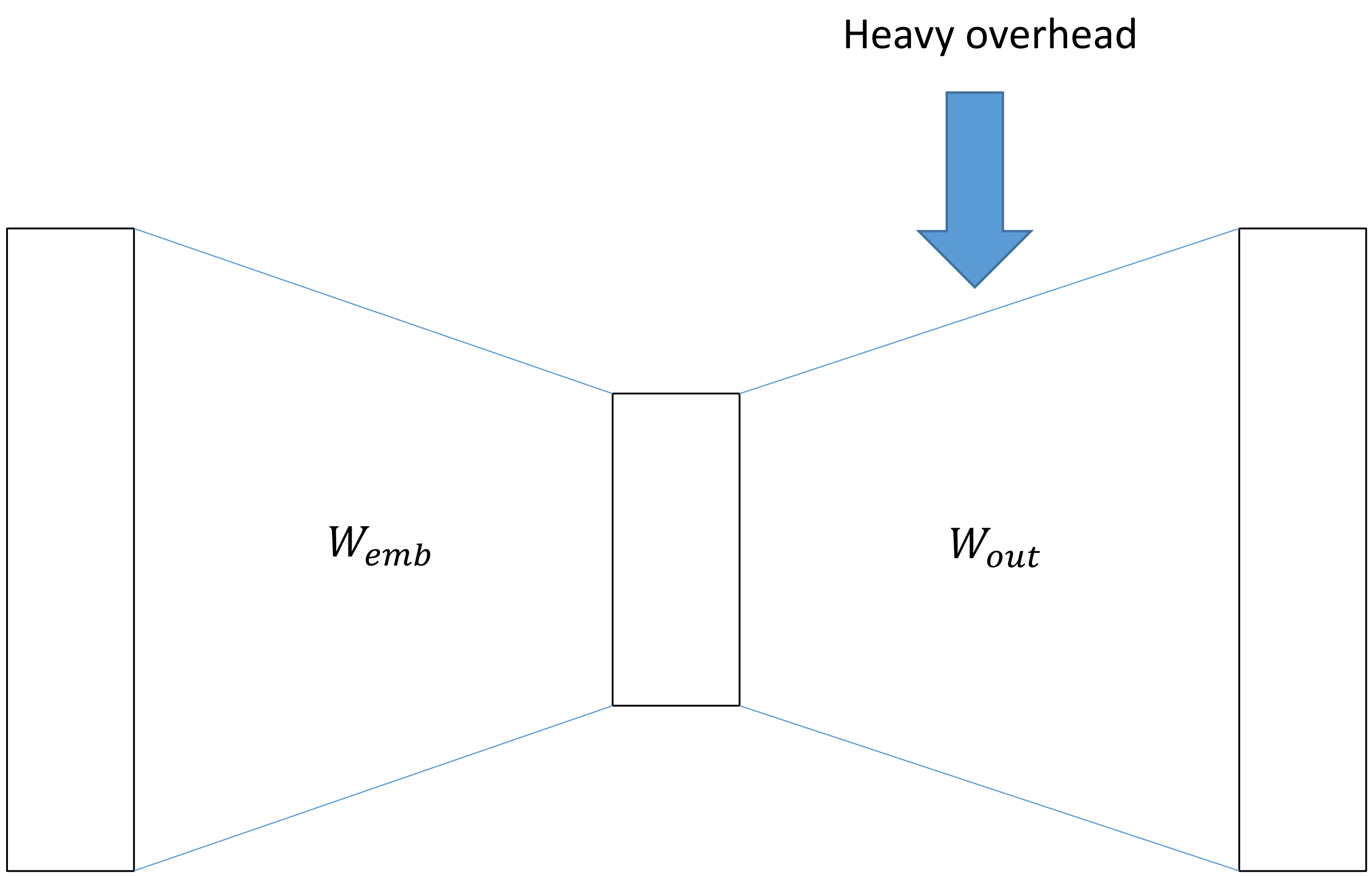


With 840B dataset

Output dimension : 2.2M
Feature dimension : 300

W_{out} : (2.2M, 300)

Word2Vec

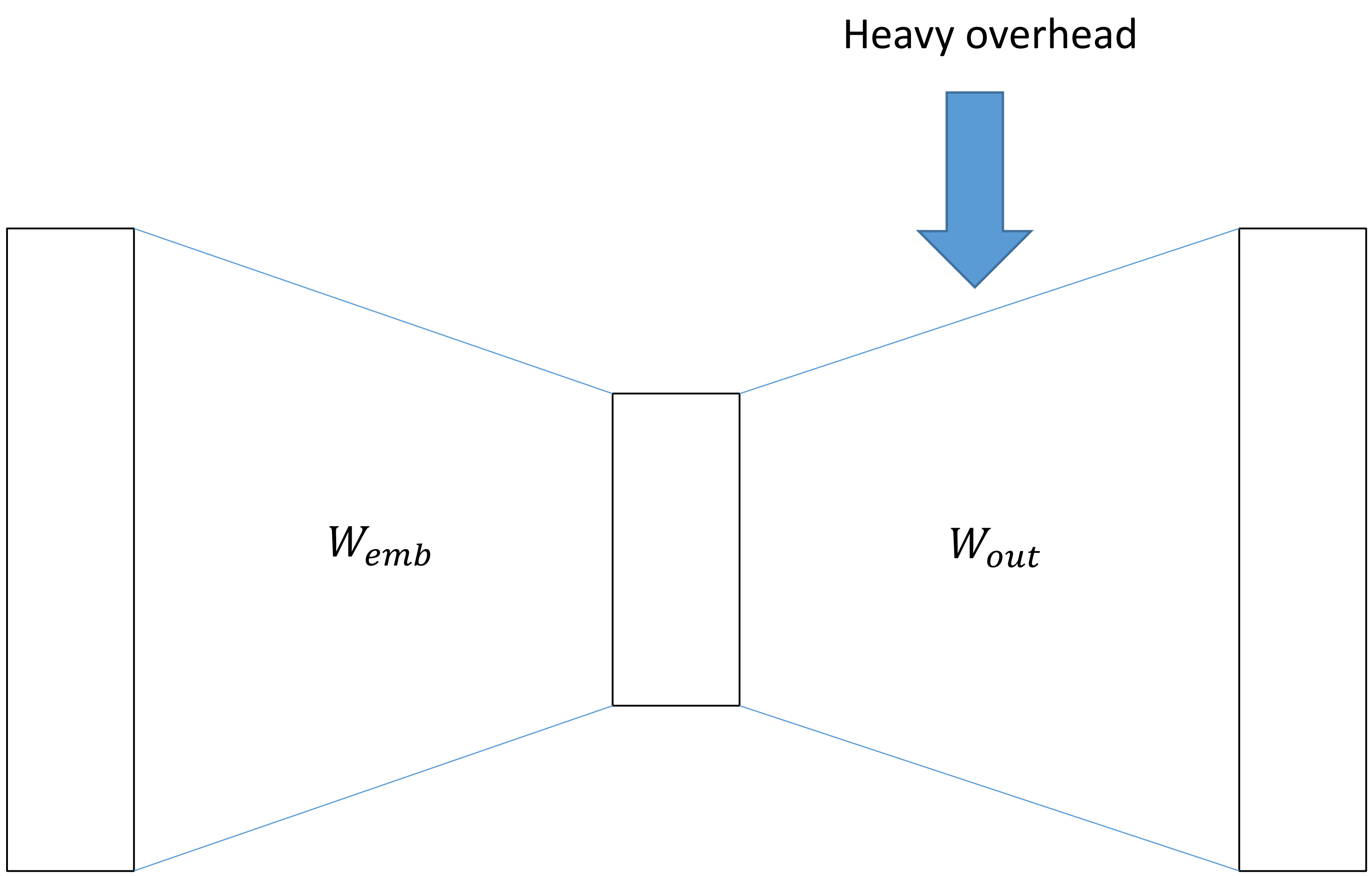


With 840B dataset

Output dimension : 2.2M
Feature dimension : 300

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

Word2Vec



With 840B dataset

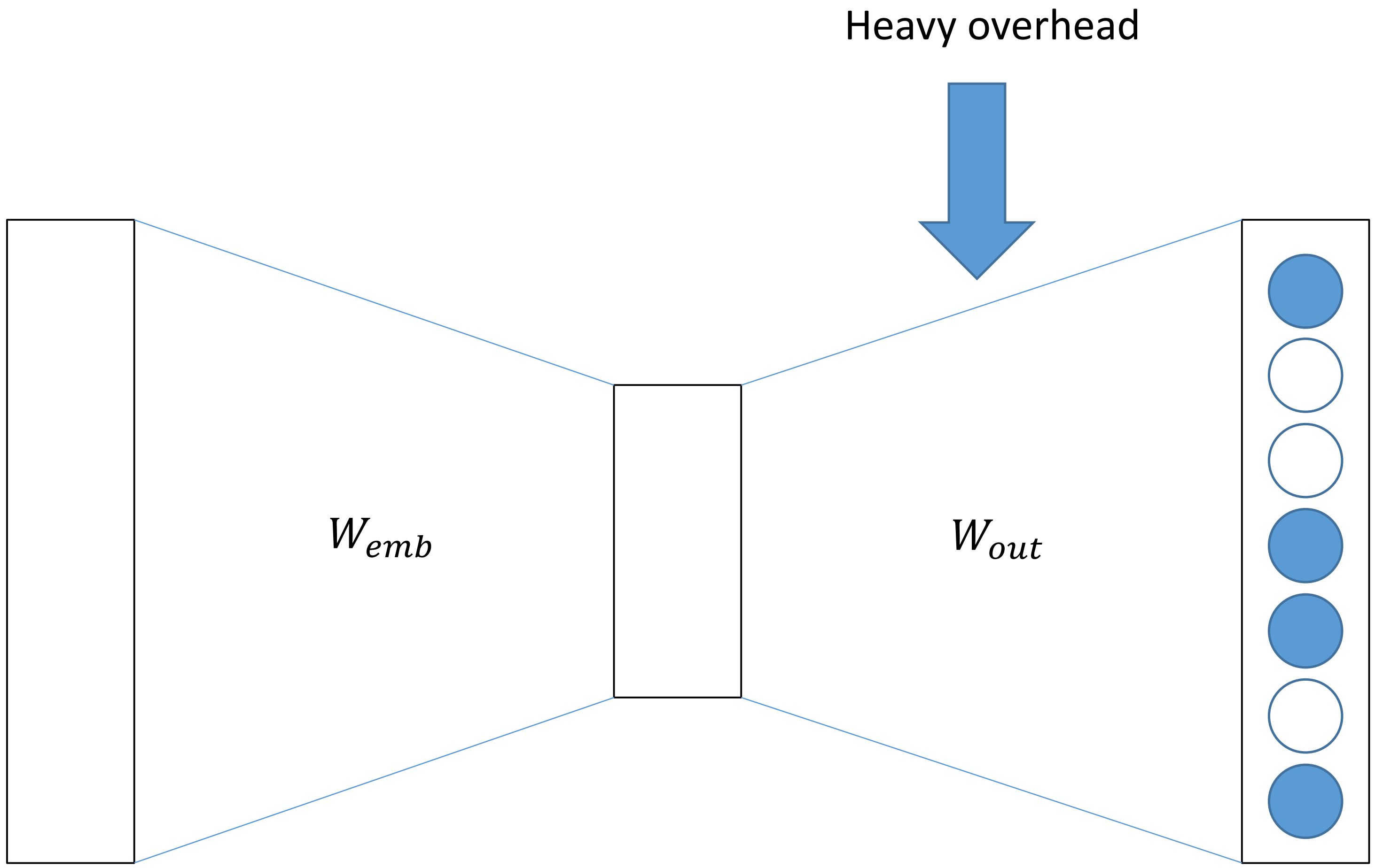
Output dimension : 2.2M
Feature dimension : 300

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

Hierarchical Softmax

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

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

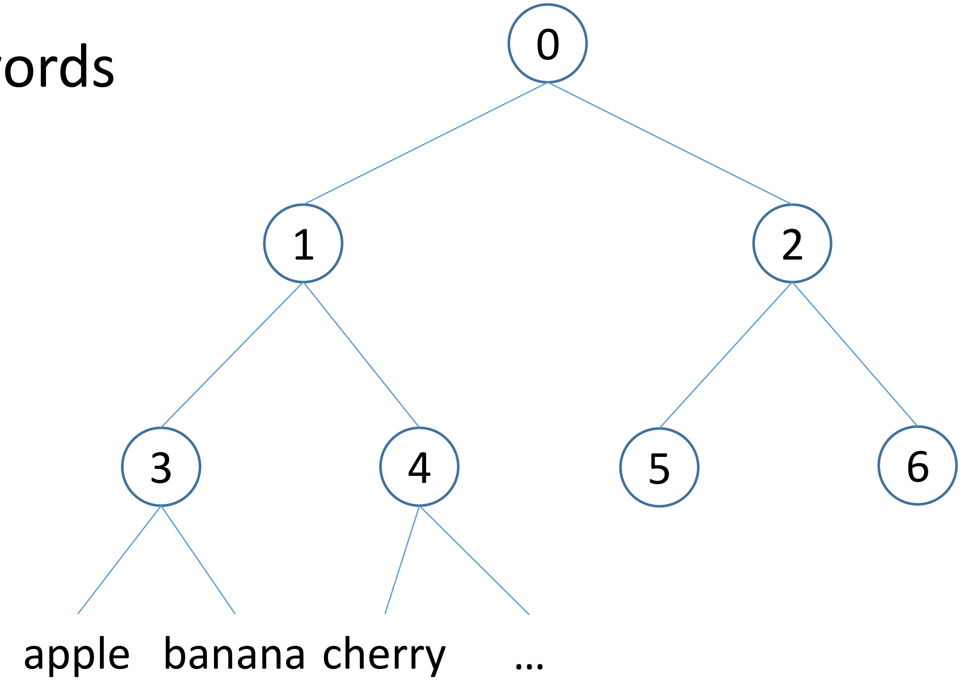
Word2Vec

Hierarchical Softmax

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

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

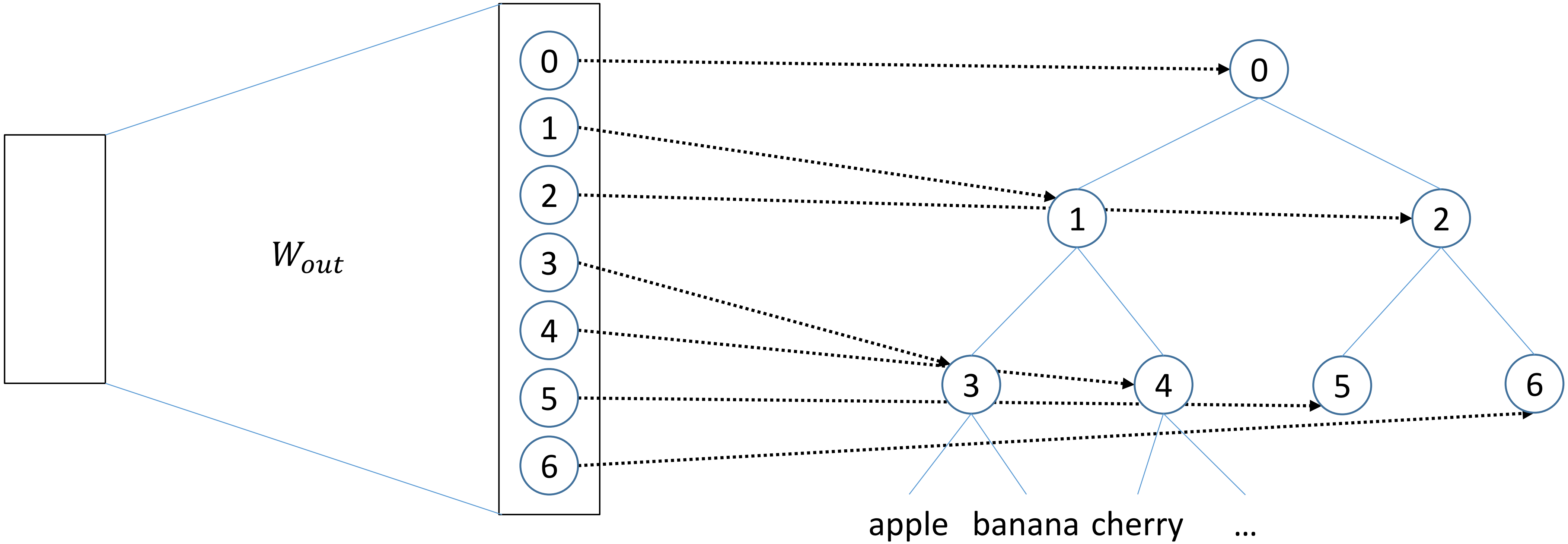
Suppose that 0 is the left and 1 is the right



Word2Vec

Hierarchical Softmax

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



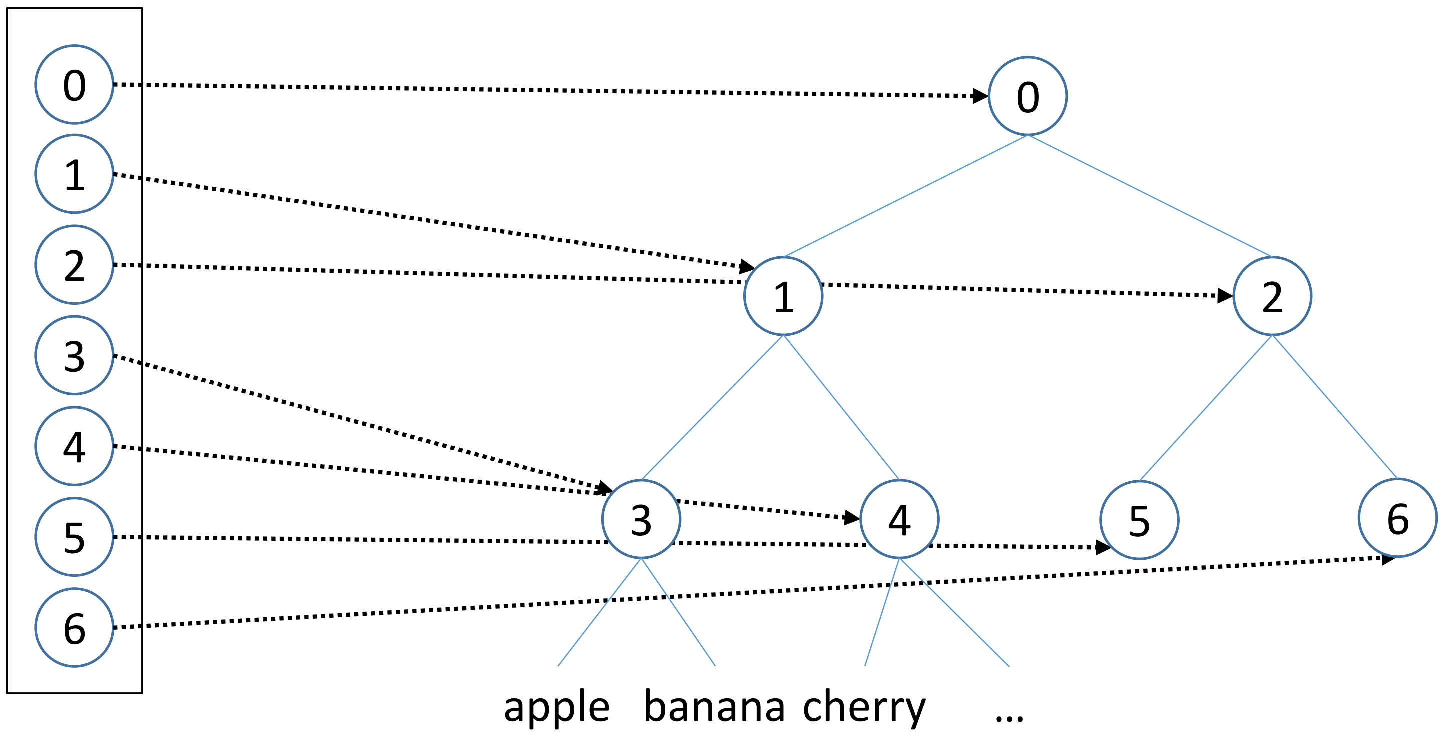
Word2Vec

Hierarchical Softmax

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

sigmoid activation function instead of softmax

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

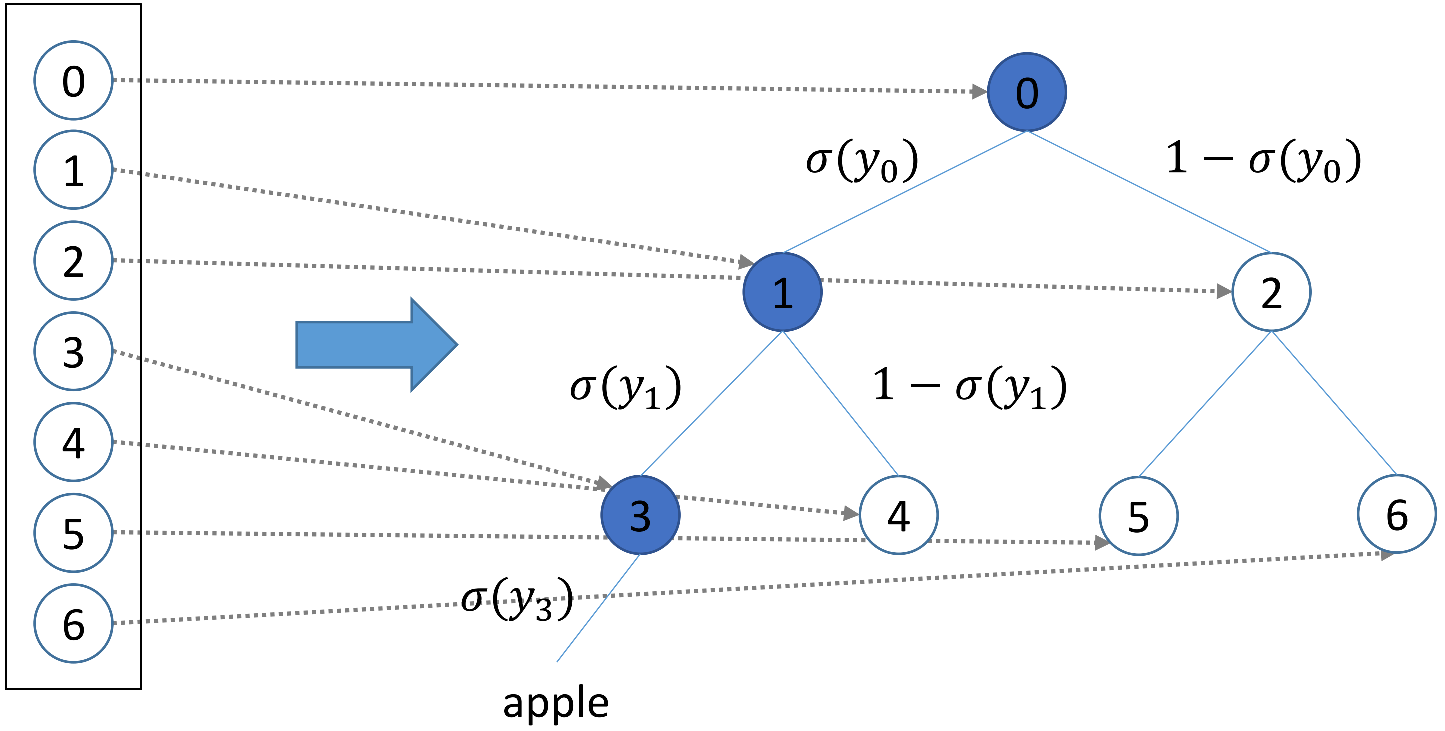


Word2Vec

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



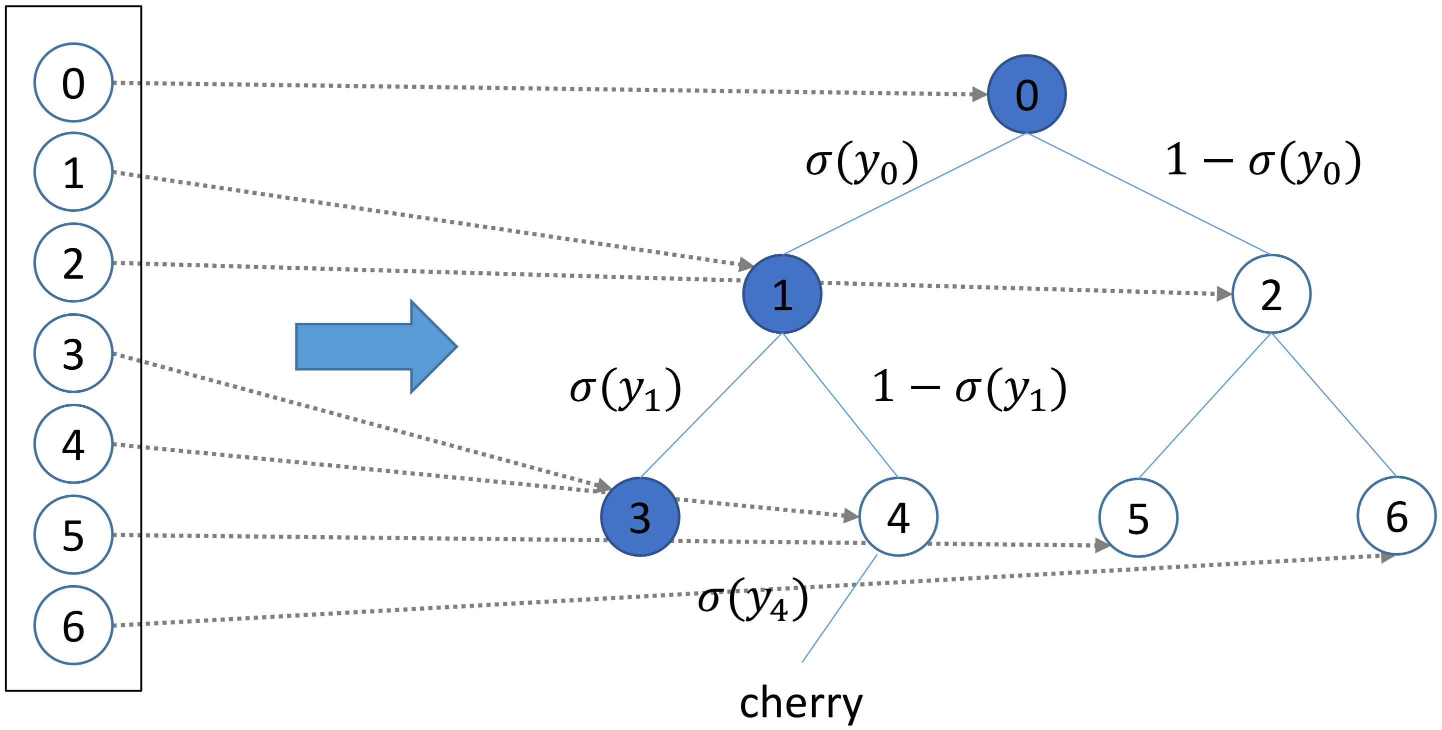
$$p(apple) = \sigma(y_0)\,\sigma(y_1)\,\sigma(y_3)$$

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



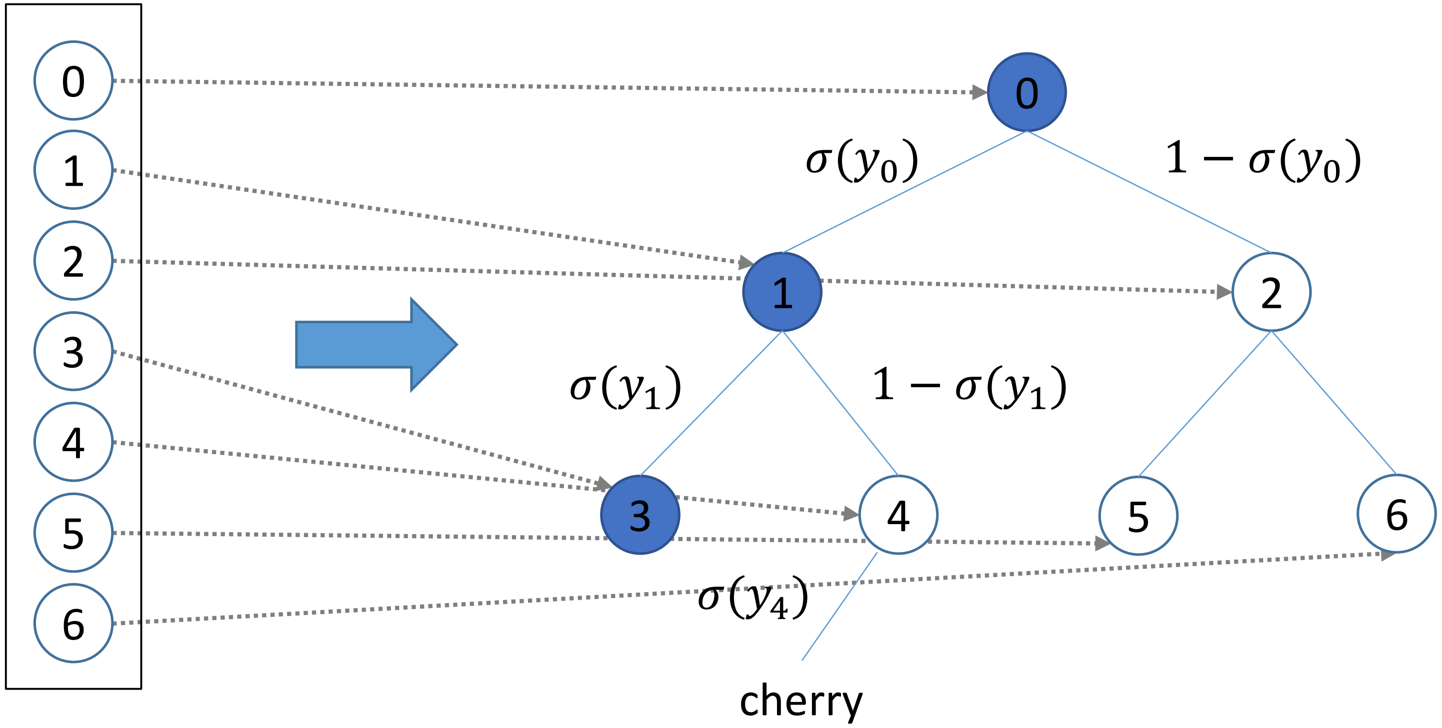
$$p(cherry) = \sigma(y_0)\,(1 - \sigma(y_1))\,\sigma(y_4)$$

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

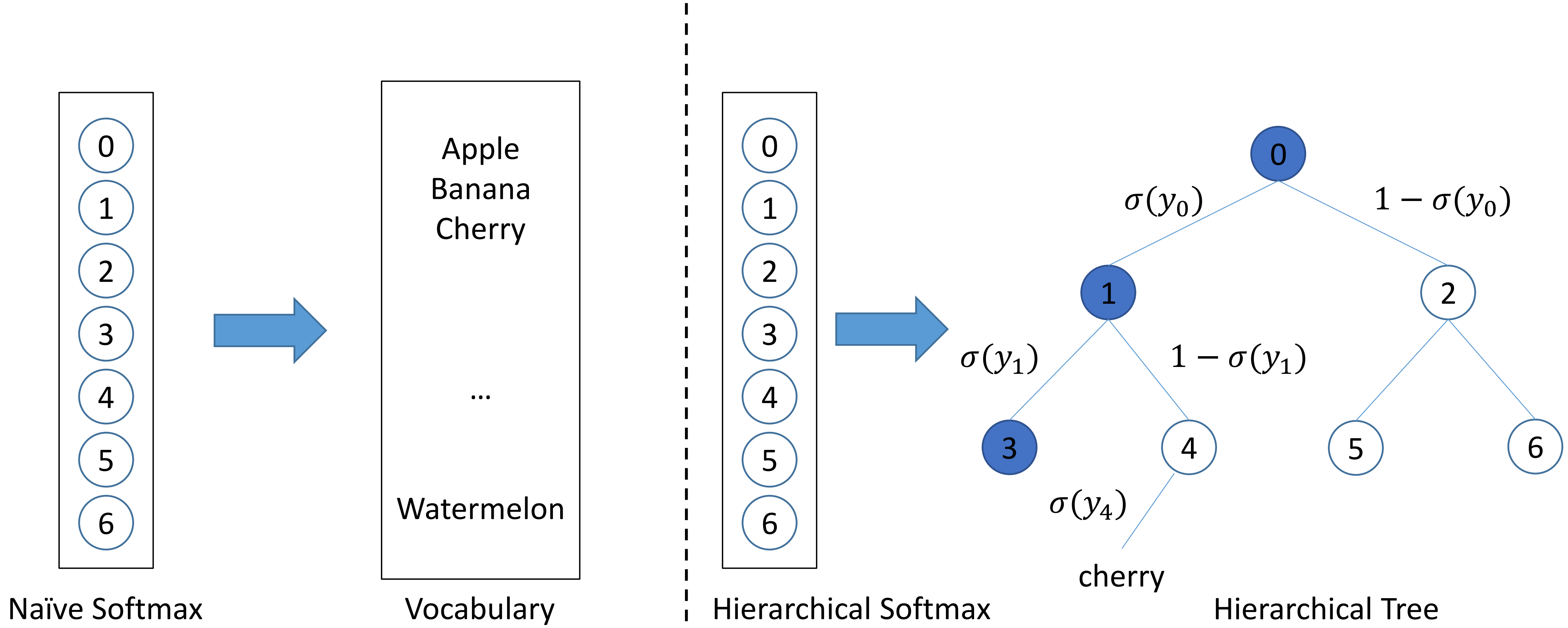


$p(cherry) = \sigma(y_0)\,(1-\sigma(y_1))\,\sigma(y_4)$

Minimize $-\log p(cherry)$

Word2Vec

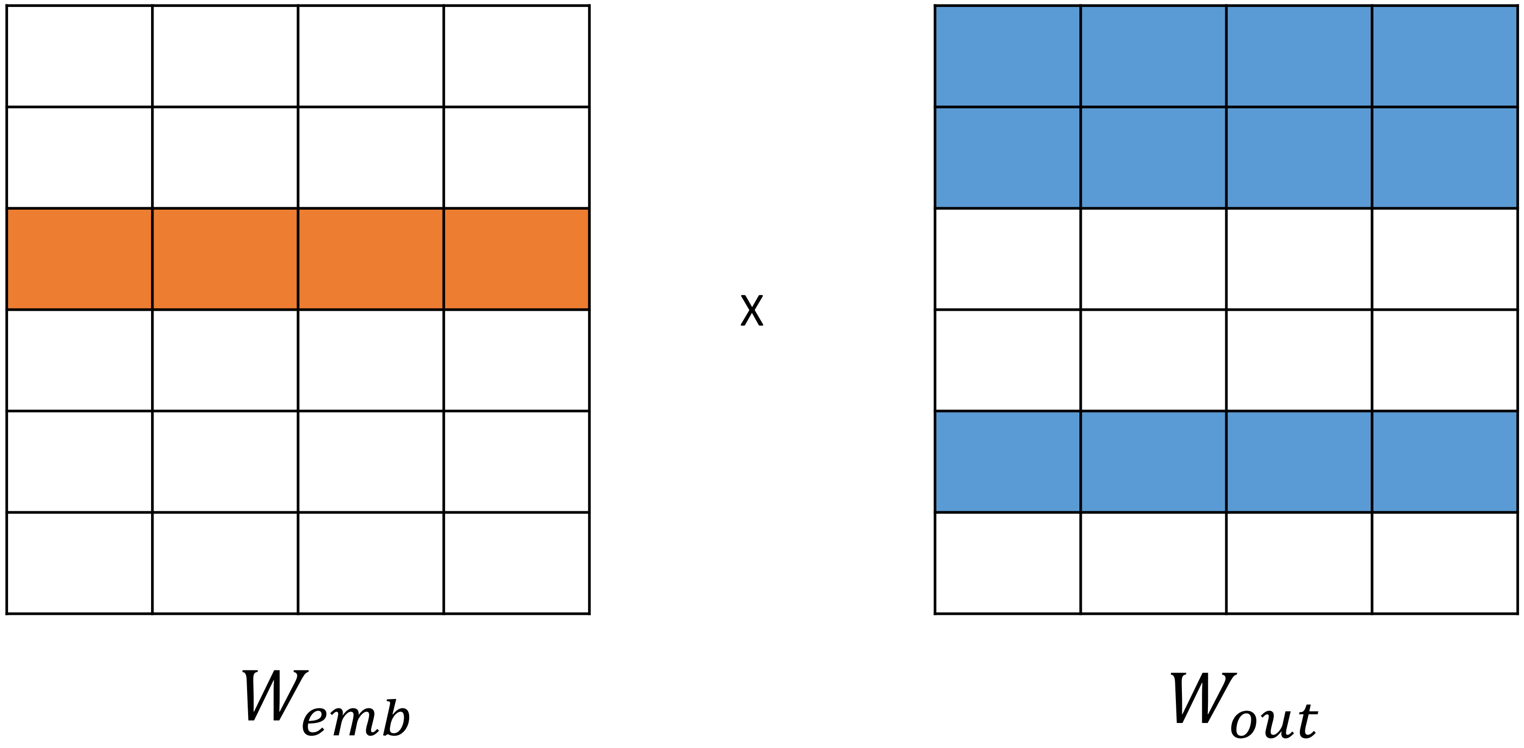
Hierarchical Softmax



Word2Vec

Hierarchical Softmax

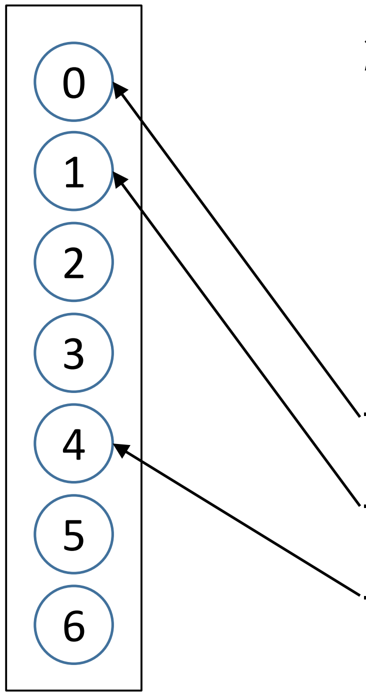
6. Weights connected to the activated nodes are updated



Word2Vec

Hierarchical Softmax

6. Weights connected to the activated nodes are updated



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

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

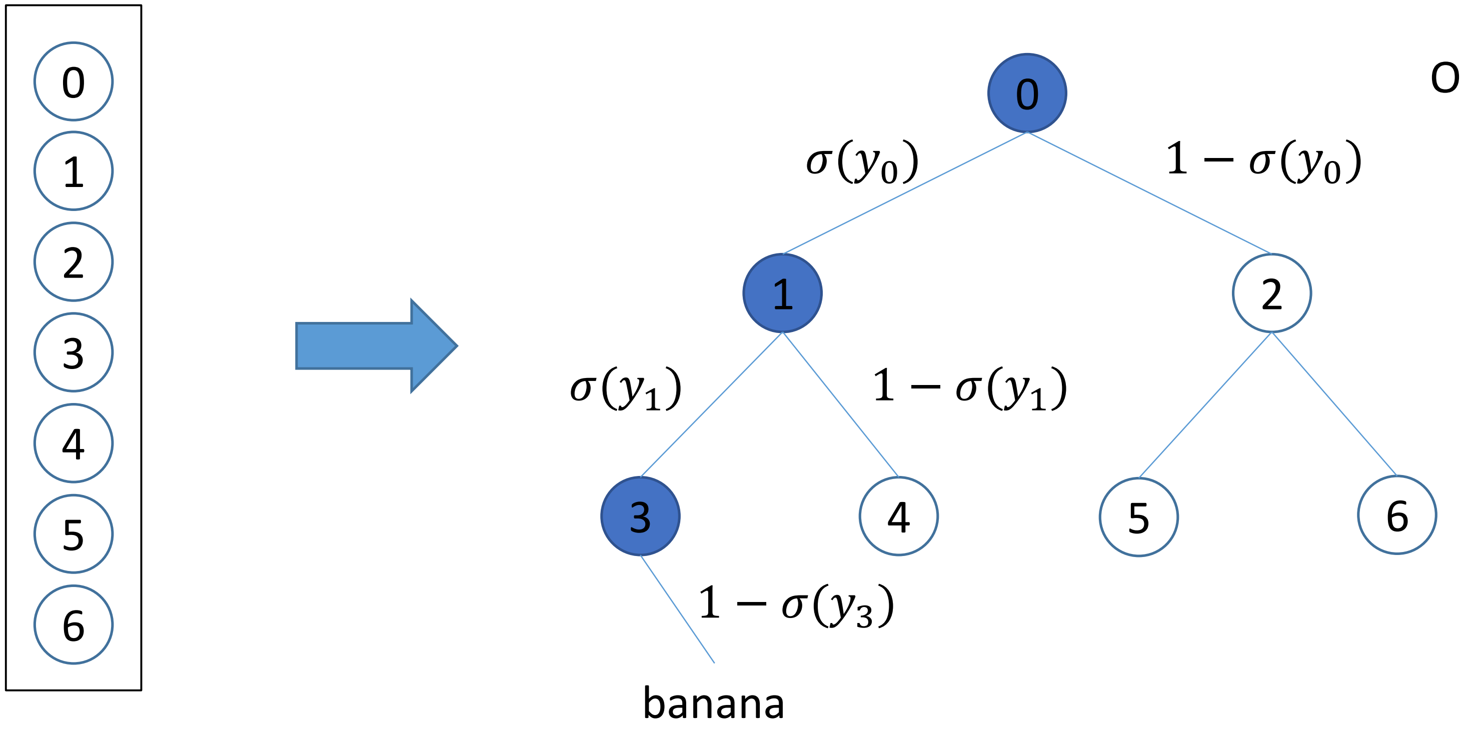
$$\frac{\partial L}{\partial y_0} = \sigma(y_0) - 1$$

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

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

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

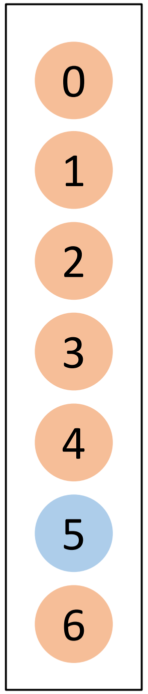
Word2Vec

- Word2vec is still slow…

Model	Vector Dimensionality	Training words	Accuracy [%]			Training time [days]
			Semantic	Syntactic	Total	
3 epoch CBOW	300	783M	15.5	53.1	36.1	1
3 epoch Skip-gram	300	783M	50.0	55.9	53.3	3
1 epoch CBOW	300	783M	13.8	49.9	33.6	0.3
1 epoch CBOW	300	1.6B	16.1	52.6	36.1	0.6
1 epoch CBOW	600	783M	15.4	53.3	36.2	0.7
1 epoch Skip-gram	300	783M	45.6	52.2	49.2	1
1 epoch Skip-gram	300	1.6B	52.2	55.1	53.8	2
1 epoch Skip-gram	600	783M	56.7	54.5	55.5	2.5

Word2Vec

Negative Sampling



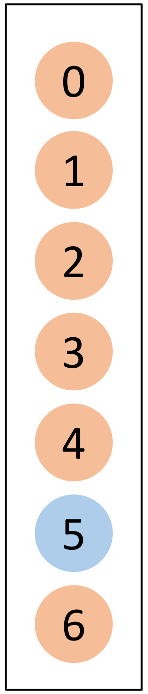
1 of positive sample

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

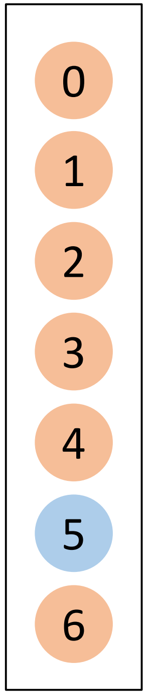
1?
5-10?
Half of the negatives?

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

Frequency^(3/4)

Word2Vec

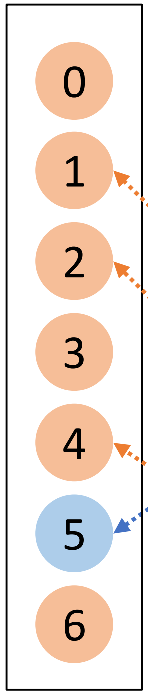
Negative Sampling

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

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

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

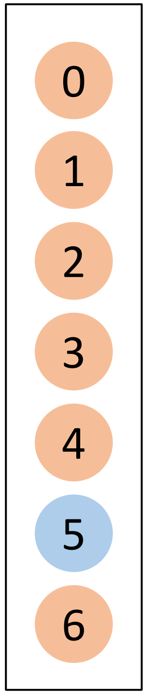
$$\frac{\partial L}{\partial y_1} = \sigma(y_1)$$

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

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

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

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

Word2Vec

Subsampling

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 3

- 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 3

- Activated Weight Matrix

```
if mode=="CBOW":
    if NS==0:
        #Only use the activated rows of the weight matrix
        #activated should be torch.tensor(K,) so that activated W_out has the form of torch.tensor(K, D)
        activated = None
        L, G_in, G_out = CBOW_HS(inputs, codes[output], W_in, W_out[activated])
        W_in[inputs] -= learning_rate*G_in
        W_out[activated] -= learning_rate*G_out
    else:
        #Only use the activated rows of the weight matrix
```

Recommend to use a portion of W_out for the computational efficiency

Assignment 3

- 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 3

- Negative Sampling
  - Use a table instead of random sampling functions

size: 20

0	1	1	1	2	2	2	2	2	2	3	3	3	3	3	3	3	4	4	4

Probabilities

word0: 0.05
word1: 0.15
word2: 0.3
word3: 0.35
word4: 0.15

- Generate a random integer x [0,19]
- Pick the xth element

Assignment 3

- 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

Note: text8 only includes lower cases

[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 3

- Word2Vec Exper

Analogical reasoning task

```
: capital-common-countries
Athens Greece Baghdad Iraq
Athens Greece Bangkok Thailand
Athens Greece Beijing China
Athens Greece Berlin Germany
Athens Greece Bern Switzerland
Athens Greece Cairo Egypt
Athens Greece Canberra Australia
Athens Greece Hanoi Vietnam
Athens Greece Havana Cuba
Athens Greece Helsinki Finland
Athens Greece Islamabad Pakistan
Athens Greece Kabul Afghanistan
Athens Greece London England
Athens Greece Madrid Spain
Athens Greece Moscow Russia
Athens Greece Oslo Norway
Athens Greece Ottawa Canada
Athens Greece Paris France
Athens Greece Rome Italy
Athens Greece Stockholm Sweden
Athens Greece Tehran Iran
Athens Greece Tokyo Japan
Baghdad Iraq Bangkok Thailand
Baghdad Iraq Beijing China
Baghdad Iraq Berlin Germany
Baghdad Iraq Bern Switzerland
Baghdad Iraq Cairo Egypt
Baghdad Iraq Canberra Australia
Baghdad Iraq Hanoi Vietnam
```

“ ?

vec(x) =vec c(“France”)

Fi rity

N ses

[1] http://code.google.com/p/word2vec/source/brows

[2] Tomas Mikolov et al. Distributed Representations of Words and Phrases and their Compositionality, 2013

Assignment 3

- Word2Vec Experiment

Analogical reasoning task[1][2]

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

Corpus : text8, 1B tokens corpus(optional)

[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

Submission 3

- Due Date : ~11/13(수) 23:59
- Submission : Submission : Online submission on blackboard
- word2vec.py + Report with analysis of word analogy task(.docx / .hwp)

- You must implement the components yourself!
- File name : StudentID_Name.zip

Q&A

- Data intelligence lab.
- irish07@korea.ac.kr (박준형)
- saga9017@korea.ac.kr (이욱진)