Sentiment Classification using Document Embeddings trained with Cosine Similarity

Tan Thongtan et al,.

ACL 2019

AILAB 송지수

Sentiment Classification

- Identify, extract, quantify, and study affective states and subjective information

오늘 보고 5시 반에 시작이라는데?

Positive

Negative

What do we need for SC?

- **Text Representation** (Word, Sentence, Document, ...)
- Classifier (Based on:
  - Traditional Methods (SVM, Statistical, ...)
  - Deep Learning Language Model (BERT, CNN, RNN, ...))

Text Representation

- Mapping variable length texts into fixed length, Dense, real-valued vectors

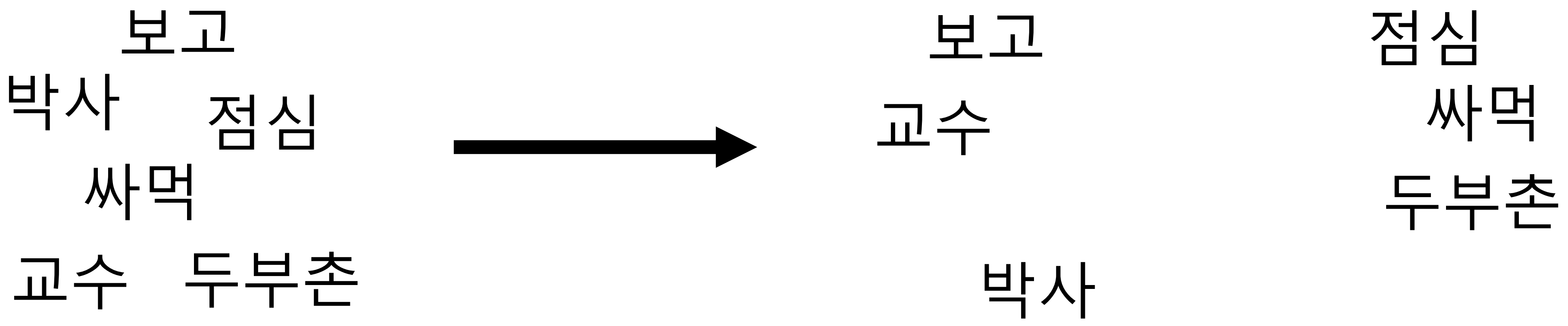
  -> To make texts as valid inputs to a classifier

보고 ⟶ [0.9528, 0.1589, -0.1862, 0.7554, ... , 0.1198]

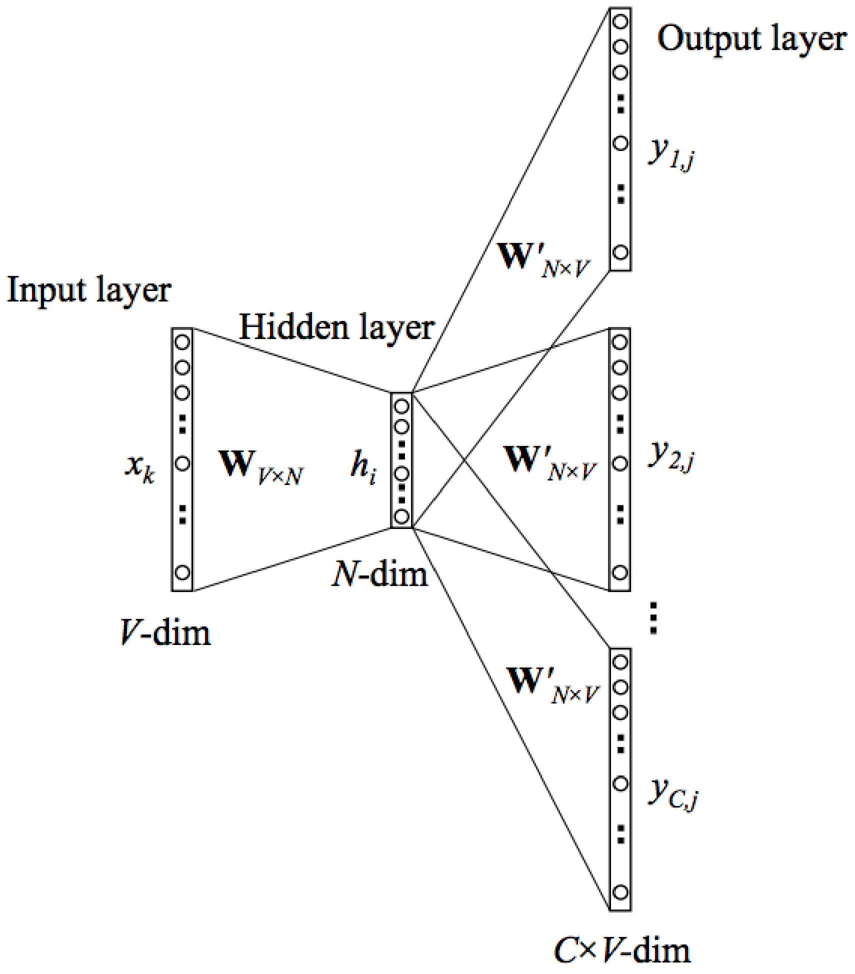
시작 ⟶ [-0.4981, 0.8631, 0.5519, 0.1891, ... , -0.8191]

Word2Vec

- Map each word to a real vector, whereby the dot product between two vectors represents the amount of similarity in meaning between the words they represent



Word2Vec



- Weight Matrix: W(V X N), W’(N X V)
  - V: Number of Words
  - N: Dimension of Word Vector

Word2Vec : Assume that V = 10,000, N = 300

1. Make a dictionary of all V

   (0: 보고, 1: 시간, 2: 점심, 3: 싸먹, 4: 두부촌, 5: 피자, …)

   then a word can be converted as a one-hot-vector

   (점심 -> [0, 0, 1, 0, 0, 0, …] / 보고 -> [1, 0, 0, 0, 0, 0, …])

Word2Vec : Assume that V = 10,000, N = 300

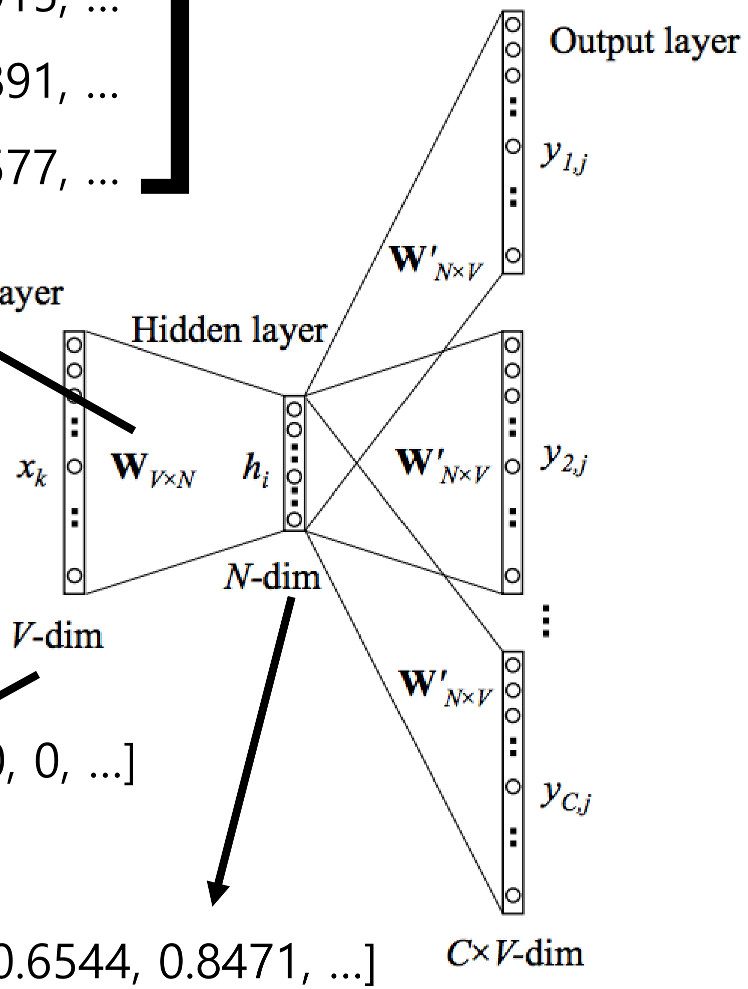
2. Get a word vector by lookup table (W)

점심 $\rightarrow$ $\underbrace{[0, 0, 1, 0, 0, 0, \ldots]}_{10{,}000}$

$$[0, 0, 1, 0, 0, 0, \ldots] \times \underbrace{\begin{bmatrix} 0.1895, 0.6841, 0.8611, -0.4915, \ldots \\ 0.9813, -0.3241, 0.1476, 0.2891, \ldots \\ 0.6544, 0.8471, 0.3695, -0.6577, \ldots \end{bmatrix}}_{300} = [0.6544, 0.8471, \ldots]$$

Word2Vec : Assume that V = 10,000, N = 300

$$\begin{bmatrix} 0.1895, 0.6841, 0.8611, -0.4915, \dots \\ 0.9813, -0.3241, 0.1476, 0.2891, \dots \\ 0.6544, 0.8471, 0.3695, -0.6577, \dots \end{bmatrix}$$



[0, 0, 1, 0, 0, 0, ...]

[0.6544, 0.8471, ...]

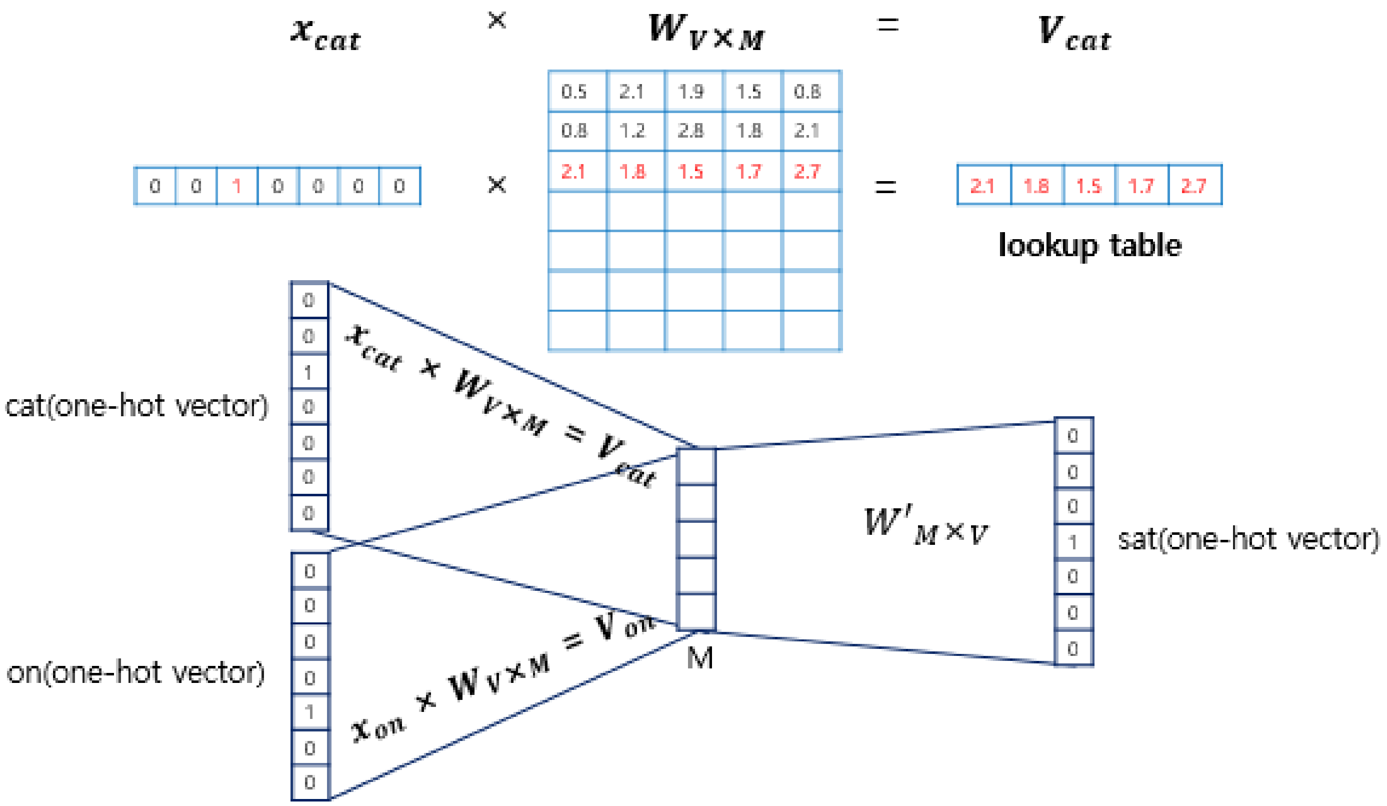
Word2Vec : Assume that V = 10,000, N = 300

3. Skip-Gram or CBOW (Continuous Bag-of-Words)

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)

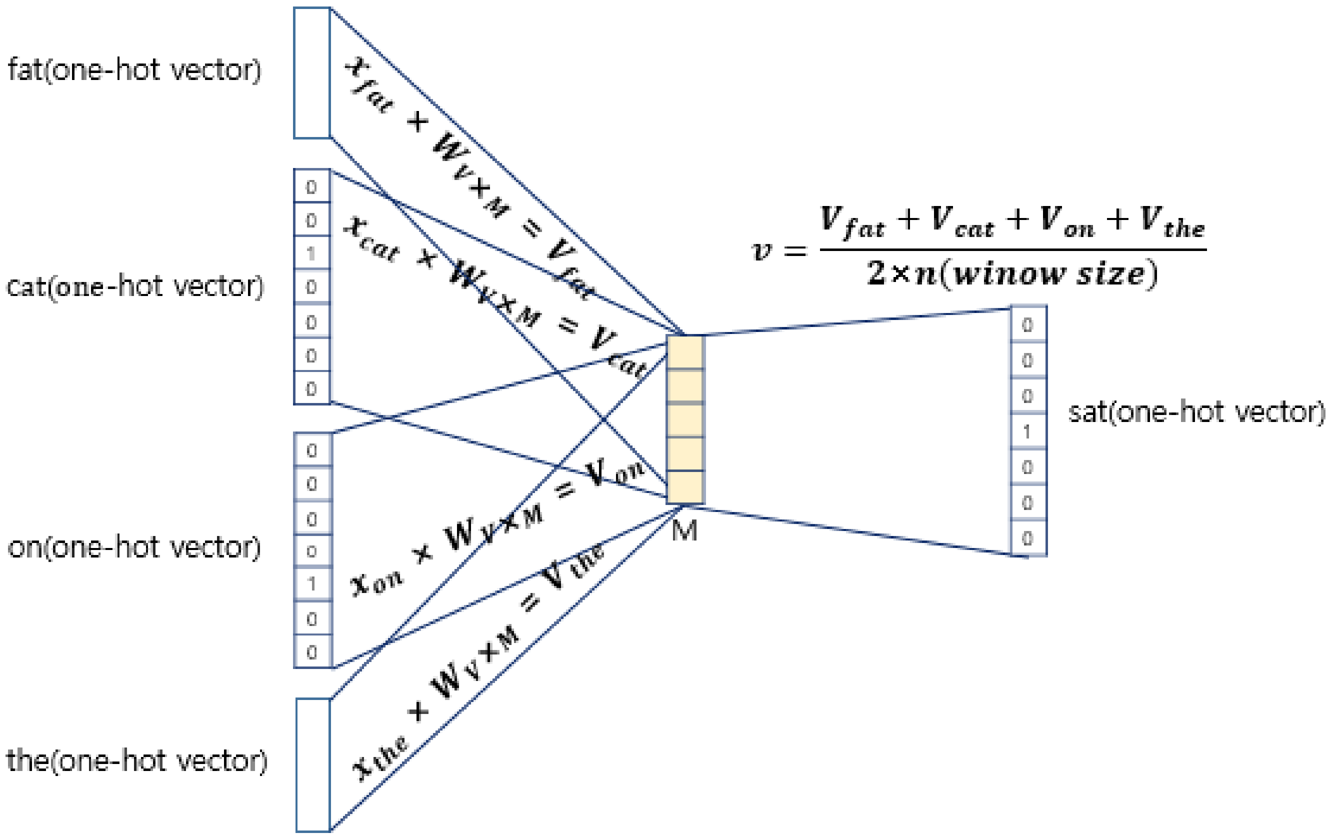
Word2Vec : Assume that V = 10,000, N = 300

3. CBOW (Continuous Bag-of-Words)



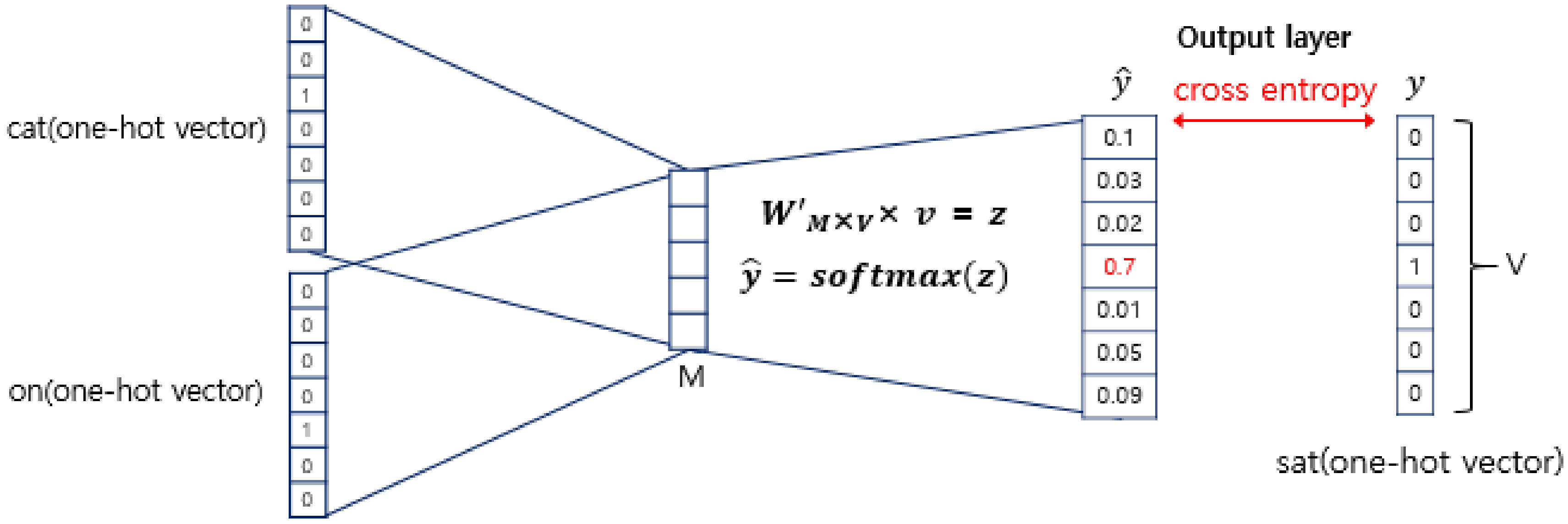
Word2Vec : Assume that V = 10,000, N = 300

3. CBOW (Continuous Bag-of-Words)



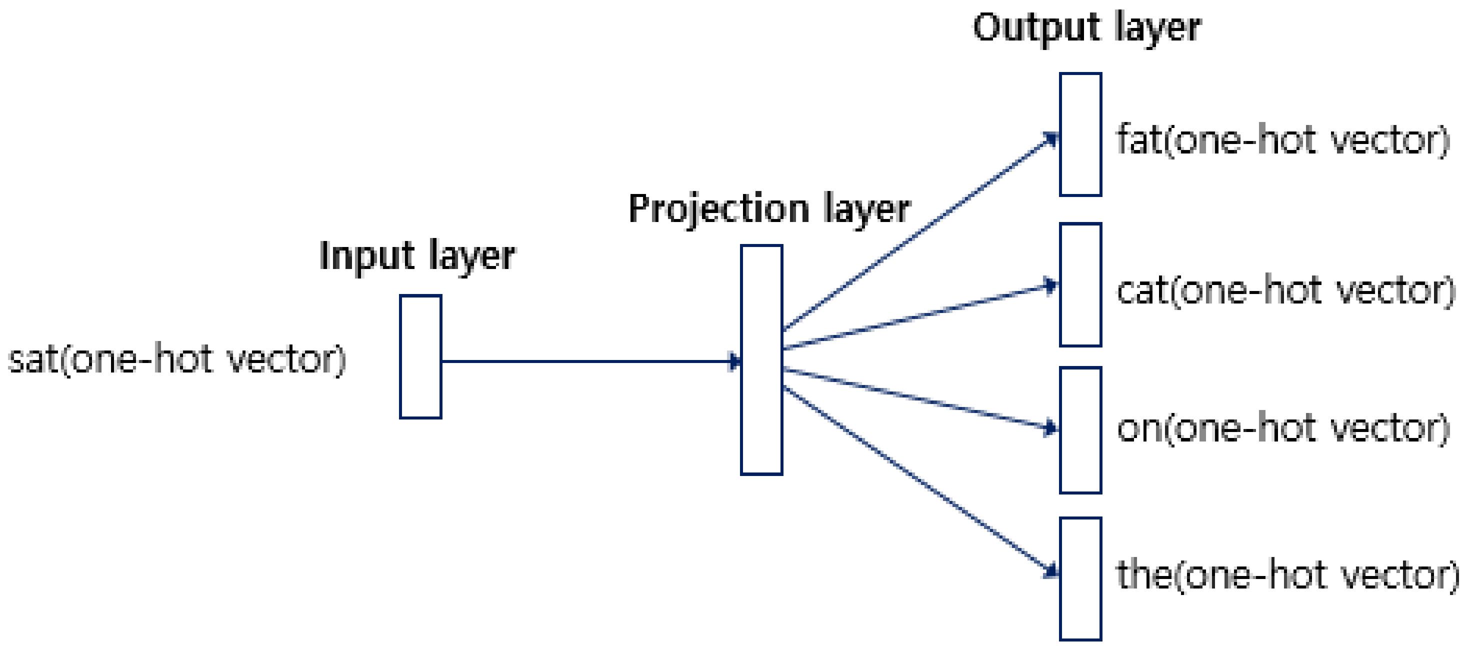
Word2Vec : Assume that V = 10,000, N = 300

3. CBOW (Continuous Bag-of-Words)



Word2Vec : Assume that V = 10,000, N = 300

3. Skip-Gram



Word2Vec : Assume that V = 10,000, N = 300

4. Learning (in Skip-Gram)

$$P(o|c) = \frac{exp(u_o^T v_c)}{\sum_{w=1}^{W} exp(u_w^T v_c)}$$

v: row vector of W

u: column vector of W’

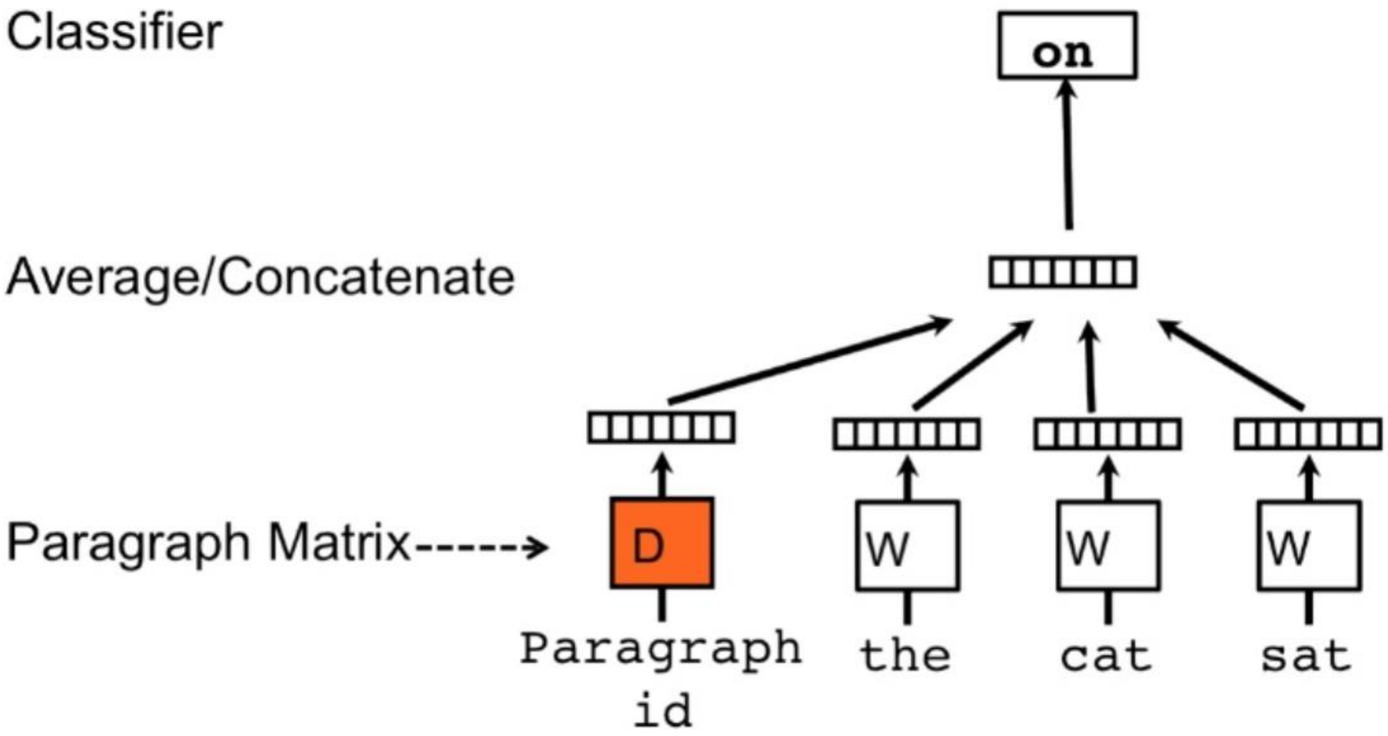
- To maximize P(o|c):
  - Numerator has to be a higher value
  - Denominator has to be a lower value

Paragraph Vector

- PV-DM vs PV-DBOW
  1. PV-DM (Distributed Memory Model)
     - A paragraph vector is additionally averaged or concatenated along with the context and that whole is used to predict the next word
  2. PV-DBOW (Distributed Bag of Words)
     - A paragraph vector alone is used and trained to predict the words in the paragraph

Paragraph Vector

- PV-DM



Paragraph Vector

- PV-DM

오늘 보고 5시 반에 시작이라는데?

Paragraph dictionary

Id	Paragraph
0	오늘 보고 5시 반에 시작이라는데?

Word dictionary

Id	Word
0	오늘
1	보고
2	5시
3	반
4	에
5	시작
6	이라는데

Paragraph Vector

- PV-DM

오늘 보고 5시 반에 시작이라는데?

Paragraph dictionary

Id	Paragraph
0	[0.3188, 0.6849, 0.6351, …]

Word dictionary

Id	Word
0	[0.6584, …]
1	[0.1387, …]
2	[0.9683, …]
3	[0.2279, …]
4	[0.2499, …]
5	[0.3766, …]
6	[0.7725, …]

Paragraph Vector

- PV-DM

오늘 보고 5시 반에 시작이라는데?

Step	Input	Label
0	[오늘 보고 5시 반에 시작이라는데?, 오늘, 보고, 5시]	반
1	[오늘 보고 5시 반에 시작이라는데?, 보고, 5시, 반]	에
2	[오늘 보고 5시 반에 시작이라는데?, 5시, 반, 에]	시작

Step	Input	Label
0	[d0, w0, w1, w2]	w3
1	[d0, w1, w2, w3]	w4
2	[d0, w2, w3, w4]	w5

Paragraph Vector

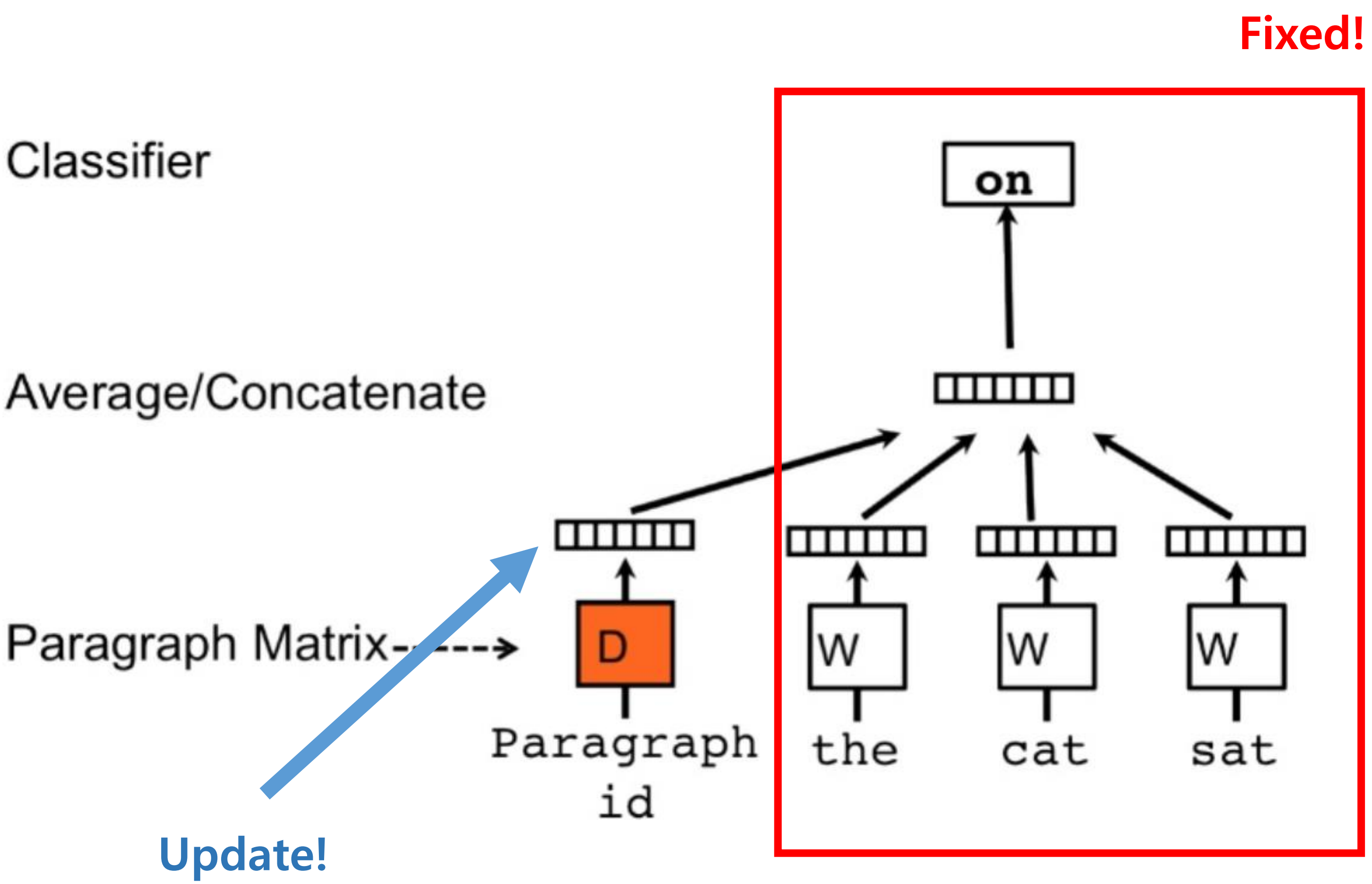
- PV-DM

오늘 보고 5시 반에 시작이라는데?

Step	Input	Label
0	[[0.3188, 0.6849, 0.6351, ...], [0.6584, ...], [0.1387, ...], [0.9683, ...]]	[0, 0, 0, 1, 0, 0, ...]
1	[[0.3188, 0.6849, 0.6351, ...], [0.1387, ...], [0.9683, ...], [0.2279, ...]]	[0, 0, 0, 0, 1, 0, ...]
2	[[0.3188, 0.6849, 0.6351, ...], [0.9683, ...], [0.2279, ...], [0.2499, ...]]	[0, 0, 0, 0, 0, 0, ...]

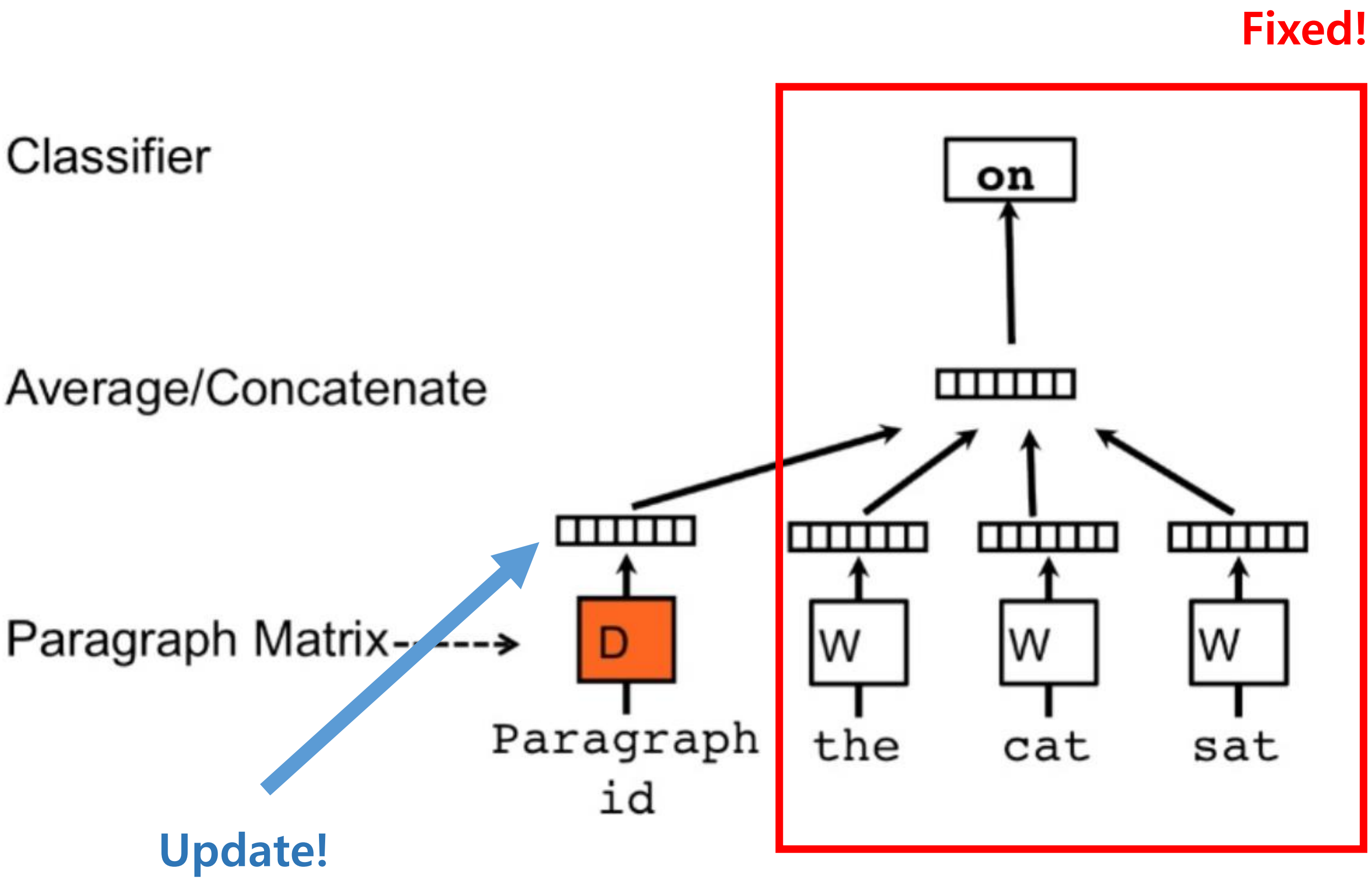
Paragraph Vector

- PV-DM



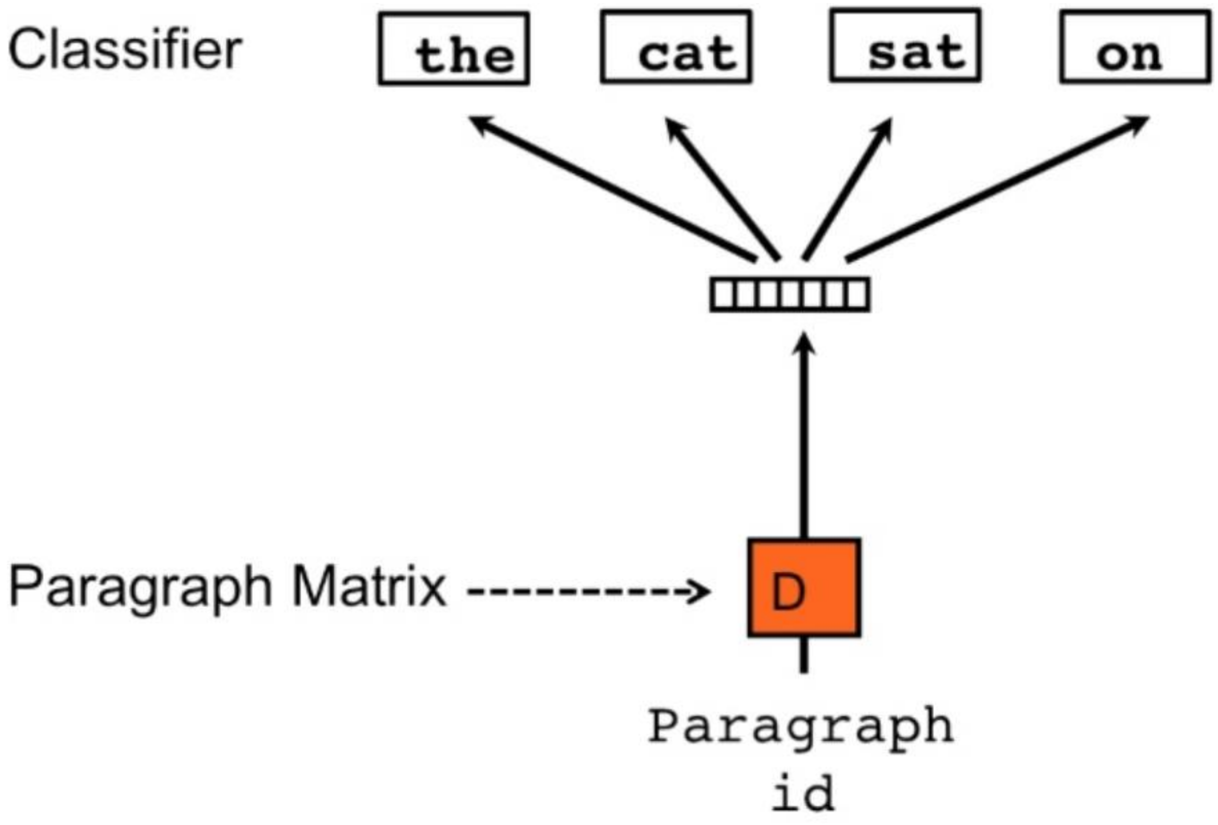
Paragraph Vector

- PV-DM



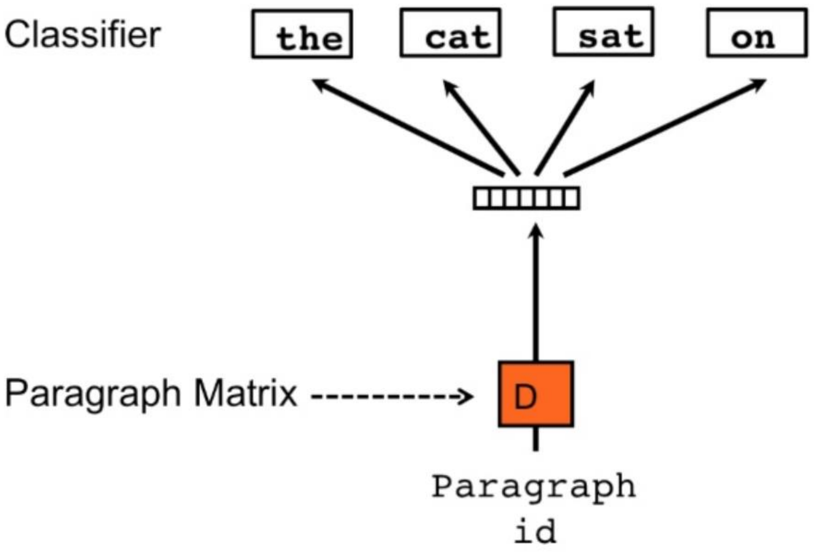
Paragraph Vector

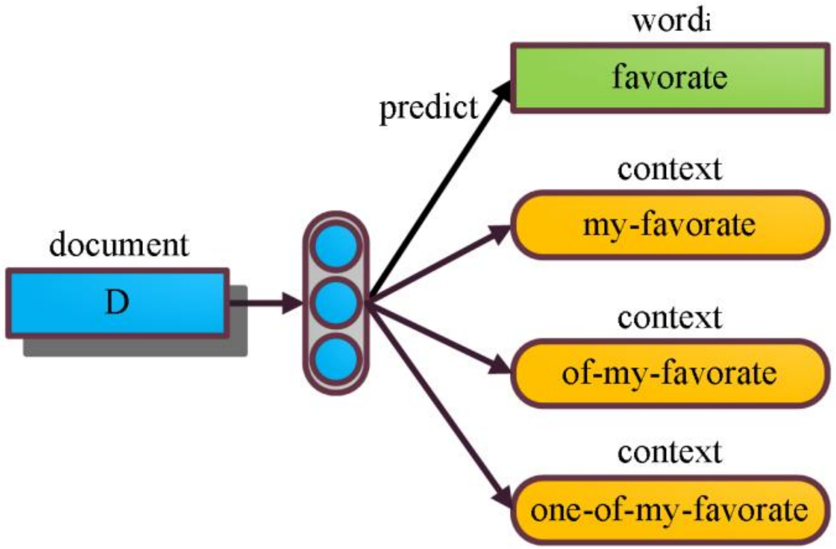
- PV-DBOW



Proposed Document Embeddings

- PV-DBOW and DV-ngram





Blended!

Proposed Document Embeddings

- Objective function

minimizes
$$\sum_{d \in D} \sum_{w_o \in W_d} -\log p(w_o | d)$$

d : Document

D : the set of all documents in the dataset

w_0 : n-gram

W_d : the set of all n-grams in the document d

Proposed Document Embeddings

- Objective function

$$p(w_o|d) = \frac{e^{\alpha \cos \theta_{w_o}}}{\sum_{w \in W} e^{\alpha \cos \theta_w}} = \mathrm{softmax}(\alpha \cos \theta_{w_o})$$

$$\cos \theta_w = \frac{\boldsymbol{v}_d^T \boldsymbol{v}_w}{||\boldsymbol{v}_d|| \, ||\boldsymbol{v}_w||}$$

v_d , v_w: vector representations of the d and the word/n-gram w

α : hyperparameter

W : the set of all n-grams in the vocabulary

Proposed Document Embeddings

- Dot Product vs Cosine Similarity

$$p(w_o|d) = \frac{e^{\boldsymbol{v}_d^T \boldsymbol{v}_{w_o}}}{\sum_{w \in W} e^{\boldsymbol{v}_d^T \boldsymbol{v}_w}}$$

Dot Product

Range : -inf ~ inf

$$p(w_o|d) = \frac{e^{\alpha \cos \theta_{w_o}}}{\sum_{w \in W} e^{\alpha \cos \theta_w}}$$

Cosine Similarity

Range : -1 ~ 1

Proposed Document Embeddings

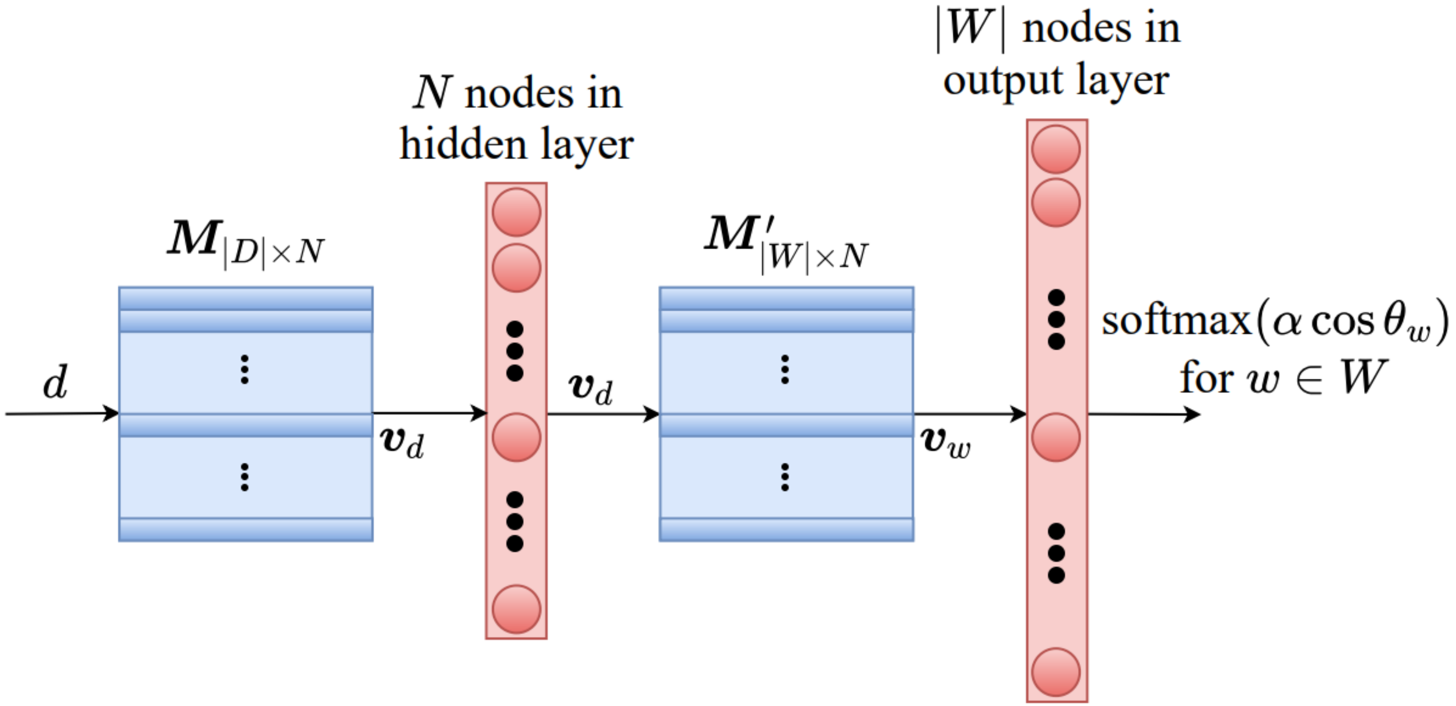
- Dot Product vs Cosine Similarity

$$p(w_o|d) = \frac{e^{\alpha \cos \theta_{w_o}}}{\sum_{w \in W} e^{\alpha \cos \theta_w}}$$

  - Using the cosine similarity term alone as an input to the softmax function may not be sufficient in modeling the conditional probability distribution → **added scaling parameter α**

Proposed Document Embeddings

- Proposed architecture



Experiments

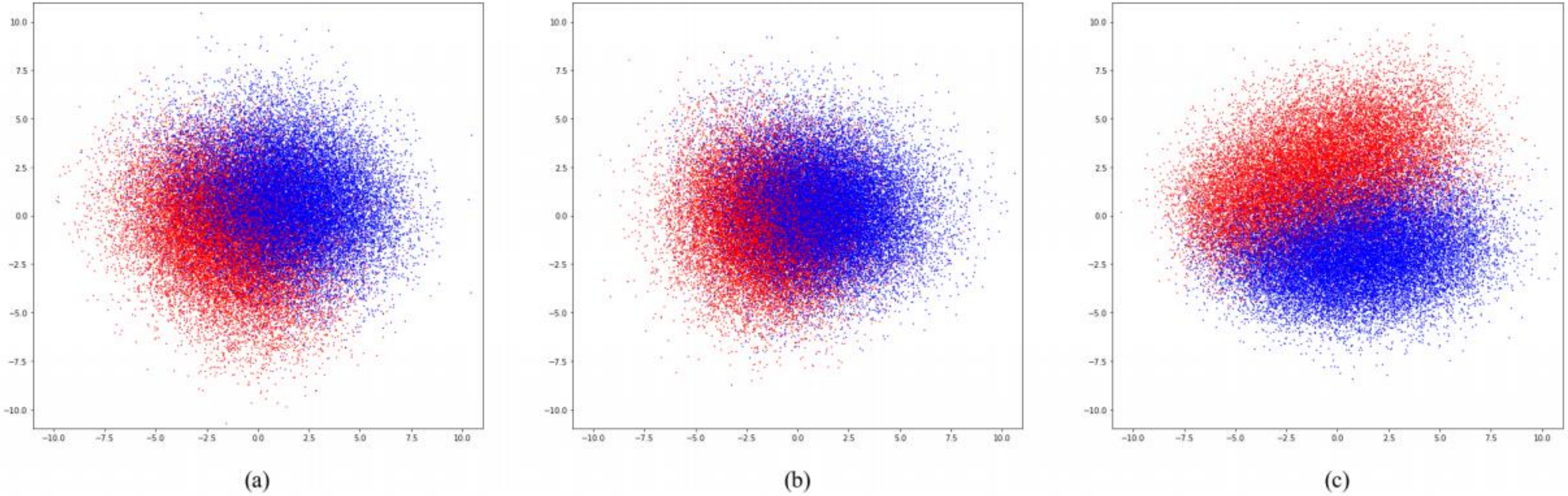


Figure 2: PCA visualization of embeddings trained with (a) dot product, (b) L2R dot product and (c) cos. similarity.

Experiments

Embedding Statistic	Dot Prod.	L2R Dot Prod.	Cos. Sim.
Same Mean Cos. Sim.	0.23	0.20	0.35
Diff. Mean Cos. Sim.	0.21	0.17	0.32
Mean Norm	8.91	6.30	5.35

Table 3: Embedding statistics.

Same (Classes) Mean Cos. Sim and Diff. (Classes) Mean Cos. Sim are both higher than embeddings using dot product

(Authors don't know why)

Experiments

Model	IMDB Dataset Accuracy (%)
NB-SVM Bigrams (Wang and Manning, 2012)	91.22
NB-SVM Trigrams (Mesnil et al., 2015)	91.87
DV-ngram (Li et al., 2016a)	92.14
Dot Product with L2 Regularization	**92.45**
Paragraph Vector (Le and Mikolov, 2014)	92.58
Document Vectors using Cosine Similarity	**93.13**
W-Neural-BON Ensemble (Li et al., 2016b)	93.51
TGNR Ensemble (Li et al., 2017)	93.51
TopicRNN (Dieng et al., 2017)	93.76
One-hot bi-LSTM (Johnson and Zhang, 2016)	94.06
Virtual Adversarial (Miyato et al., 2016)	94.09
BERT large finetune UDA (Xie et al., 2019)	95.80
NB-weighted-BON + DV-ngram	**96.95**
NB-weighted-BON + L2R Dot Product	**97.17**
NB-weighted-BON + Cosine Similarity	**97.42**

Table 4: Comparison with other models.