Convolutional Neural Networks for Sentence Classification

Daeung Kim

April, 05, 2018

Dongguk University

Artificial Intelligence Laboratory

dukim@dongguk.edu

Contents

0. Basic NLP
1. Introduction
2. Model
3. Data and Experimental Setup
4. Results and Discussion
5. Conclusion

Definition of NLP

Tokenizing

Embedding

Definition of NLP

How to program computers
to **process** and **analyze**
of natural language data.

0. Basic NLP

Tokenizing

Time is not gold, but it is yourself.

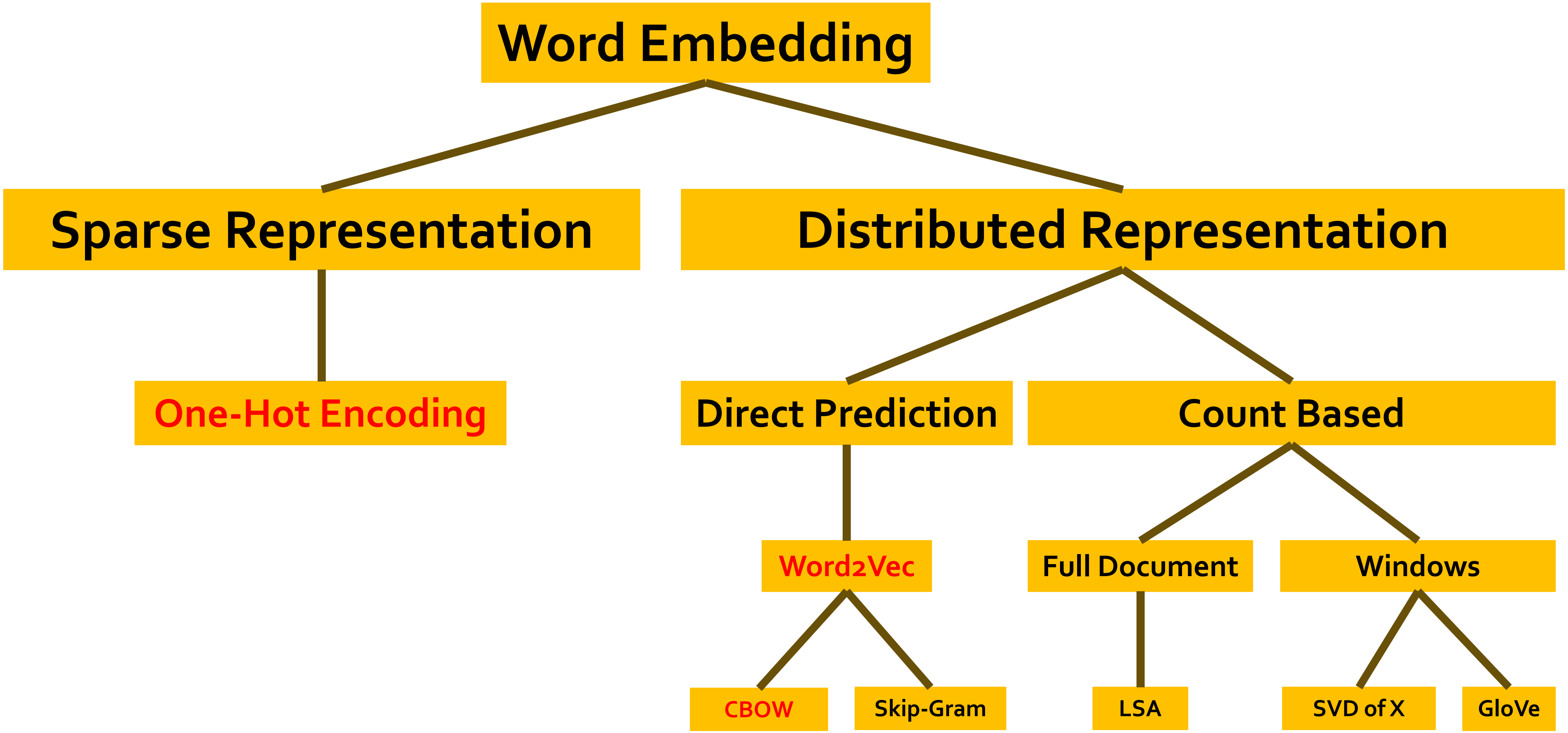
- Alphabet : “T”, “i”, “m”, “e”, “s”, “n”, “o”, “t”, “g”, “l”, “d”, “b”, “u”, “f”
- Word : “Time”, “is”, “not”, “gold”, “but”, “it”, “yourself”
- Sentence : “Time is not gold, but it is yourself.”

0. Basic NLP

Word Embedding

- **Word Representation** to make machine understand Natural Language.
- **Vector representations** of a particular word

0. Basic NLP


Word Embedding
Sparse Representation
Distributed Representation
One-Hot Encoding
Direct Prediction
Count Based
Word2Vec
Full Document
Windows
CBOW
Skip-Gram
LSA
SVD of X
GloVe

0. Basic NLP

One-hot-encoding

- "Time" : [1, 0, 0, 0, 0, 0, 0]
- "is" : [0, 1, 0, 0, 0, 0, 0]
- "not" : [0, 0, 1, 0, 0, 0, 0]
- "gold" : [0, 0, 0, 1, 0, 0, 0]
- "but" : [0, 0, 0, 0, 1, 0, 0]
- "it" : [0, 0, 0, 0, 0, 1, 0]
- "yourself" : [0, 0, 0, 0, 0, 0, 1]

One-hot-encoding

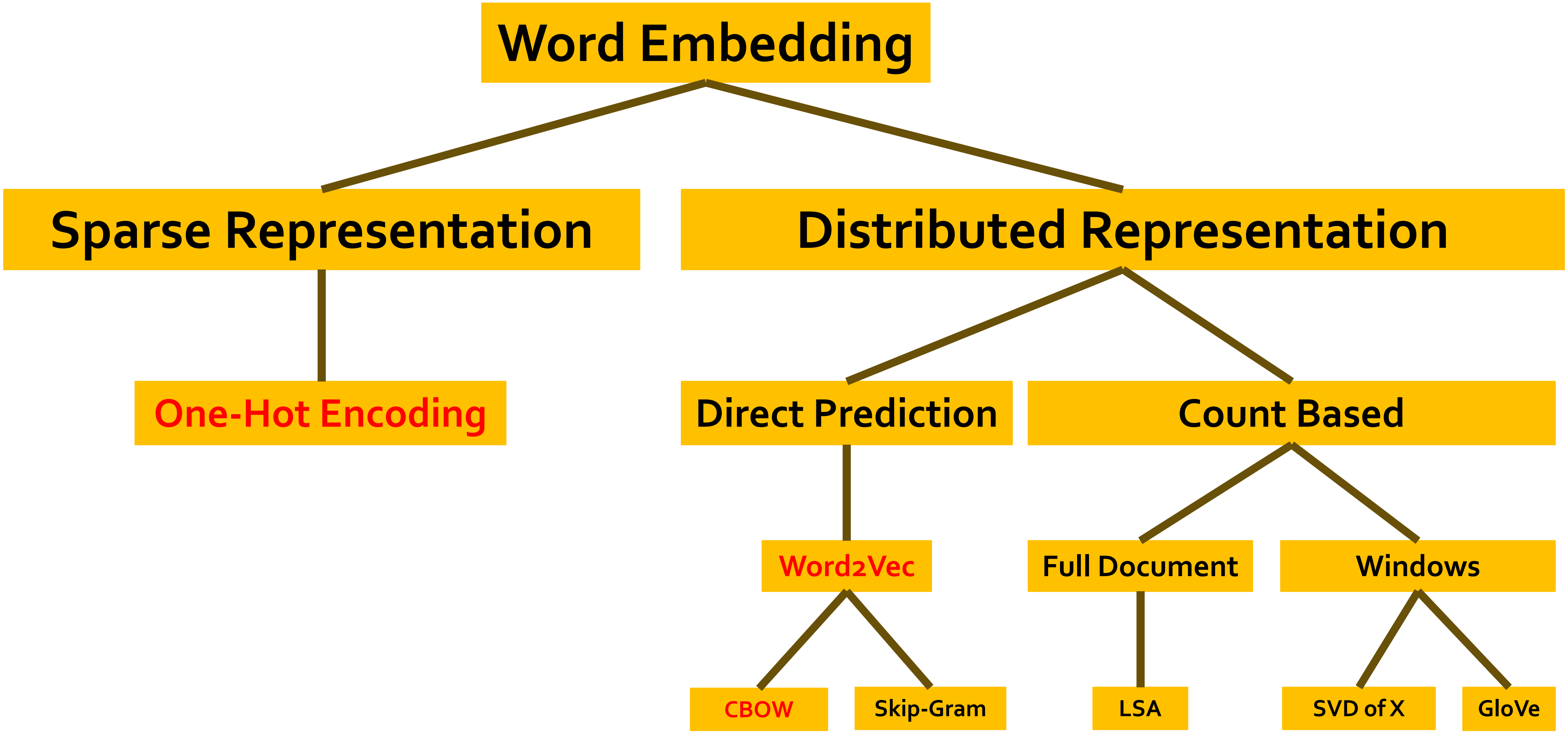
- "Time" : [1, 0, 0, 0, 0, 0, 0]
- "is" : [0, 1, 0, 0, 0, 0, 0]
- "not" : [0, 0, 1, 0, 0, 0, 0]
- "gold" : [0, 0, 0, 1, 0, 0, 0]
- "but" : [0, 0, 0, 0, 1, 0, 0]
- "it" : [0, 0, 0, 0, 0, 1, 0]
- "yourself" : [0, 0, 0, 0, 0, 0, 1]

Sparse

&

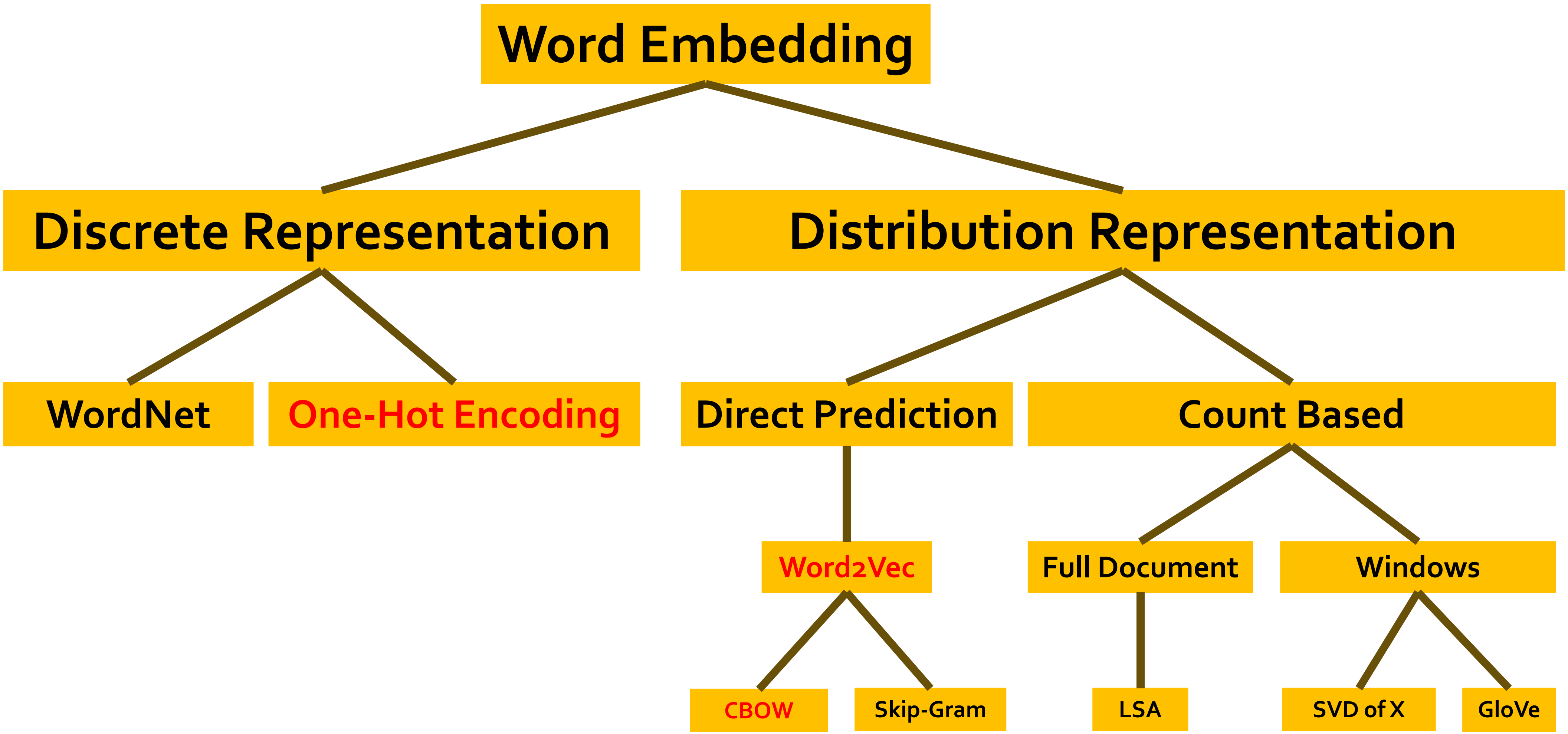
It can't represent mean of words.

0. Basic NLP



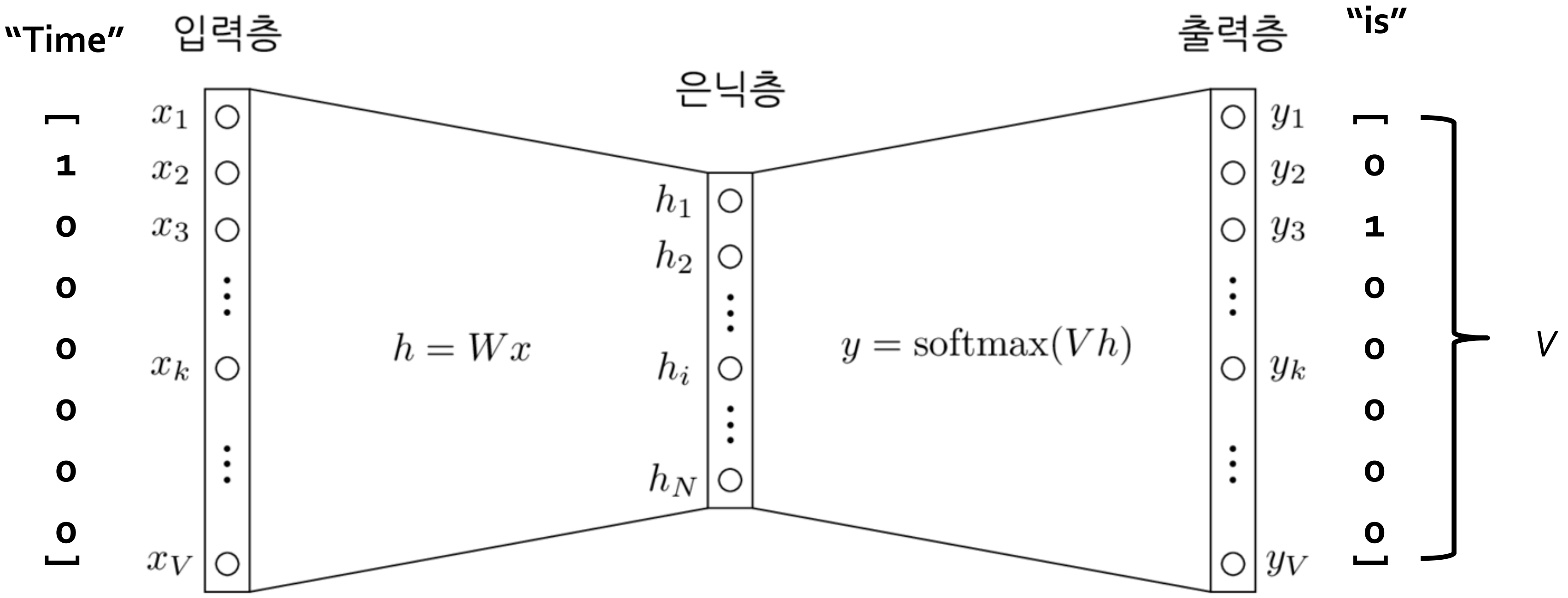
Word Embedding
Sparse Representation
Distributed Representation
One-Hot Encoding
Direct Prediction
Count Based
Word2Vec
Full Document
Windows
CBOW
Skip-Gram
LSA
SVD of X
GloVe

0. Basic NLP


Word Embedding
Discrete Representation
Distribution Representation
WordNet
One-Hot Encoding
Direct Prediction
Count Based
Word2Vec
Full Document
Windows
CBOW
Skip-Gram
LSA
SVD of X
GloVe

0. Basic NLP

Word2Vec : (0) Basic

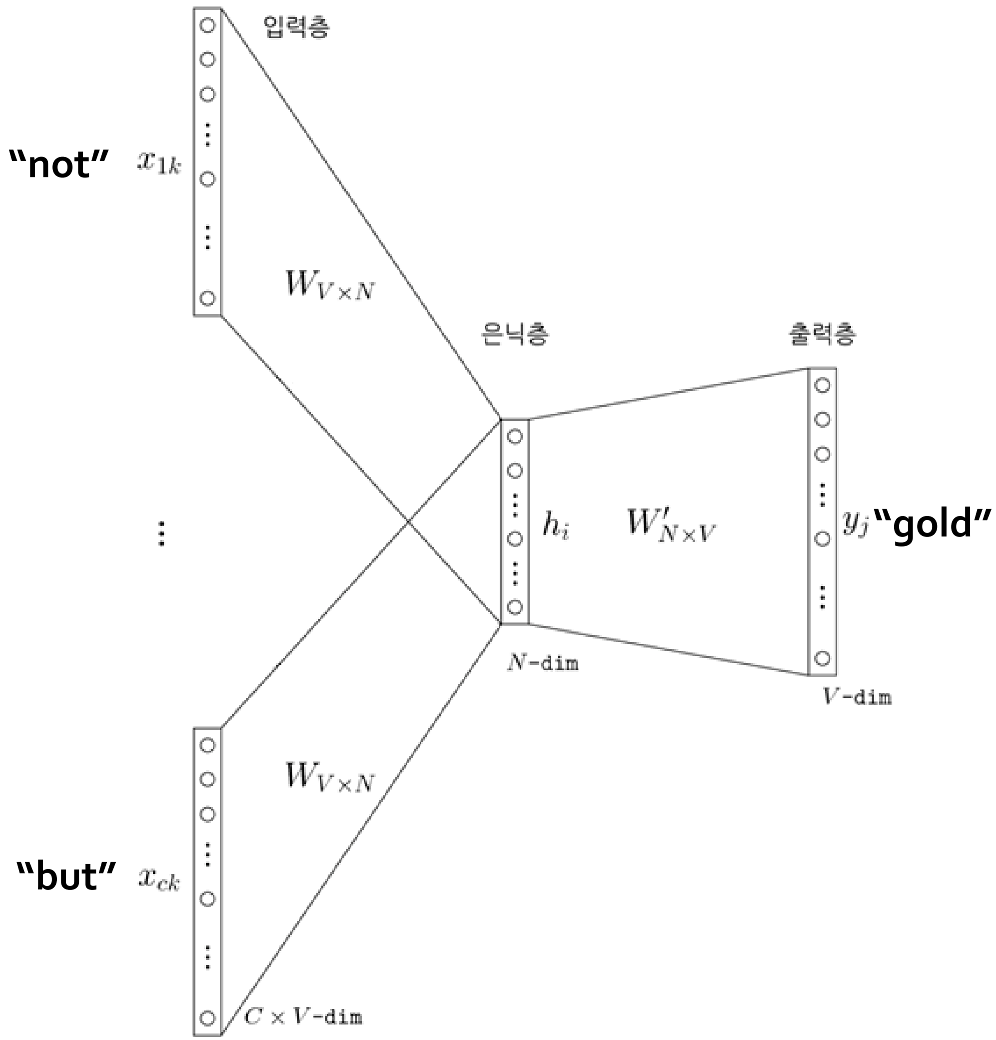


0. Basic NLP

Word2Vec : (1) CBOW

Time is not ____ but it is yourself

W = 1 (window size)

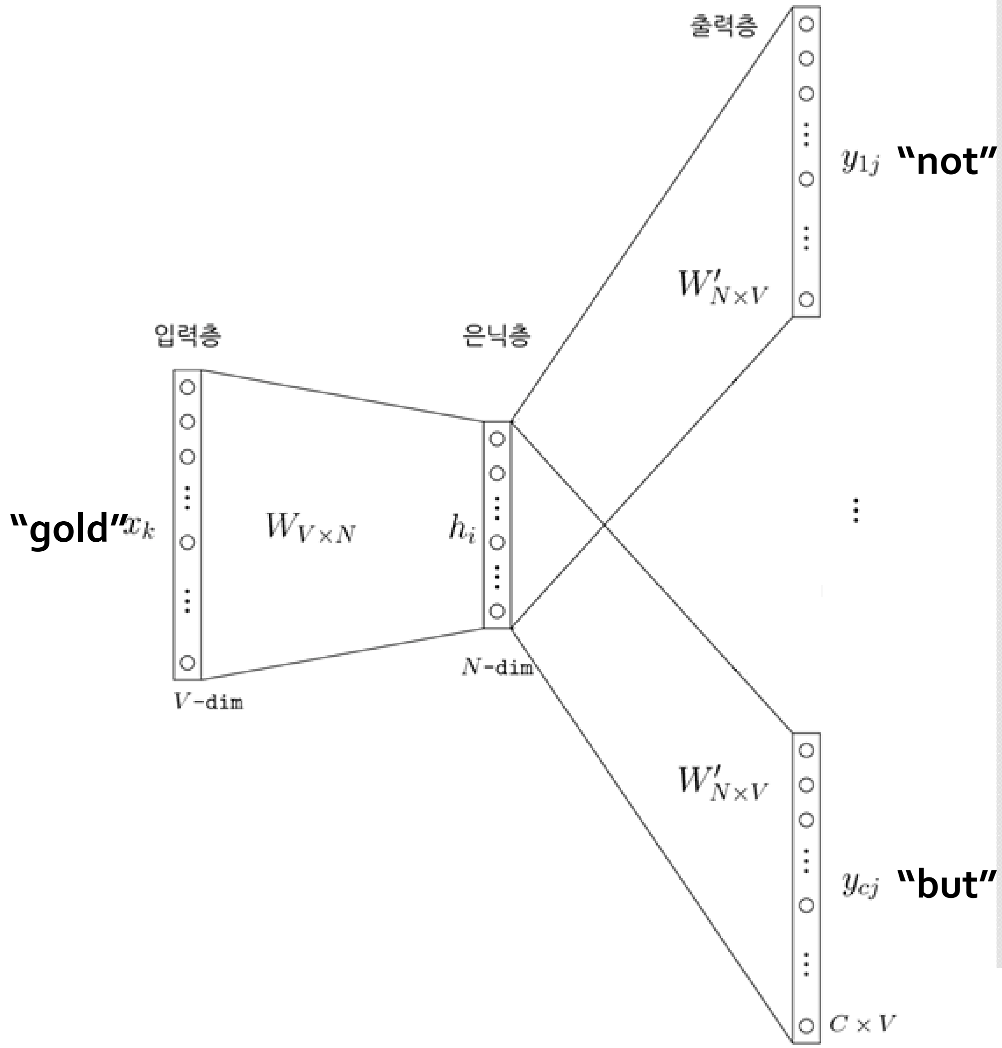


0. Basic NLP

Word2Vec : (2) Skip-gram

Time is ___ gold ___ it is yourself

W = 1 (window size)

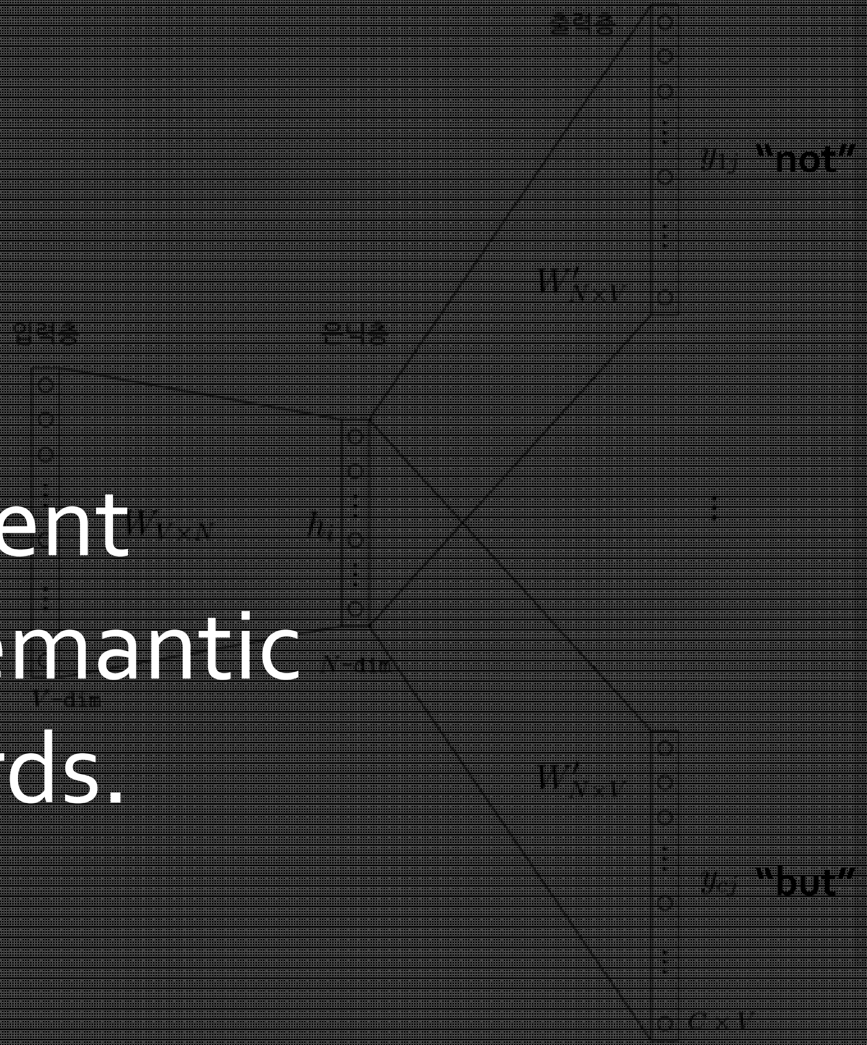


0. Basic NLP

Word2Vec : (2) Skip-gram

Time is ___ gold ___ it is

W = 1 (window size)



Dense

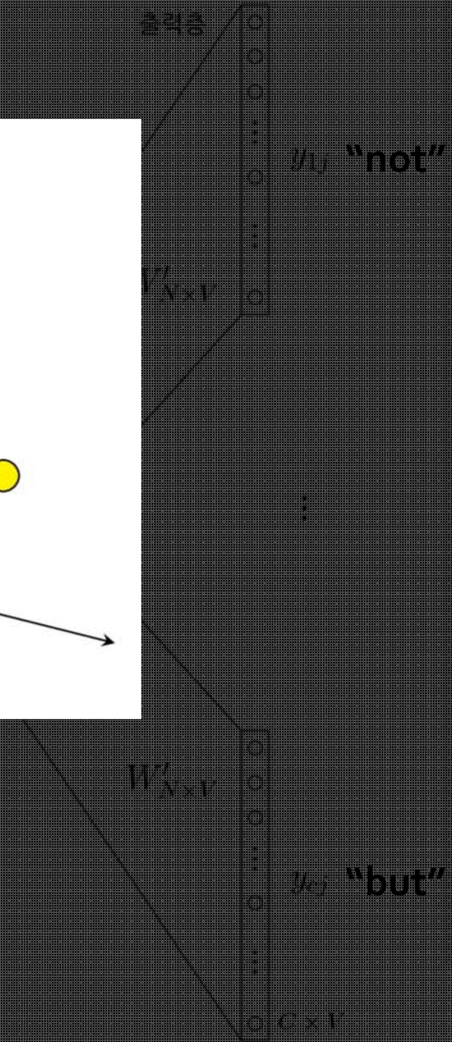
&

It can represent

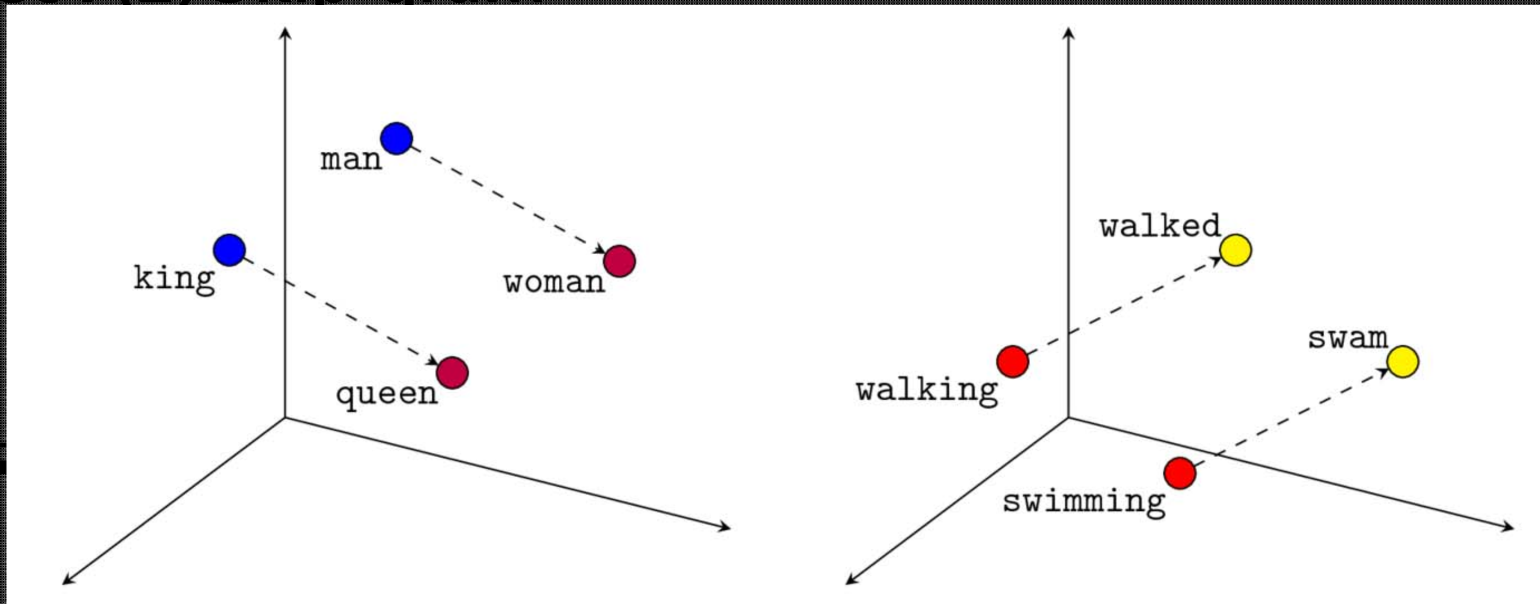
Semantic and Semantic

mean of words.

Word2Vec : (2) Skip-gram



Time is _



W = 1 (window size)

Semantic & Synthetic mean of words.

1. Introduction

Background of this paper

(1) Much of NLP work with deep-learning is based on

Word Embedding represented by Neural Language Model

(2) **CNN** have been shown to be **effective** for **NLP** too.

(3) In Image Classification, **pre-trained** **feature extractors perform well** on a variety of tasks (Razavian et al. , 2014).

1. Introduction

Background of this paper

What if

we design the NLP model

with CNN

and Pre-trained word vector?

2. Model

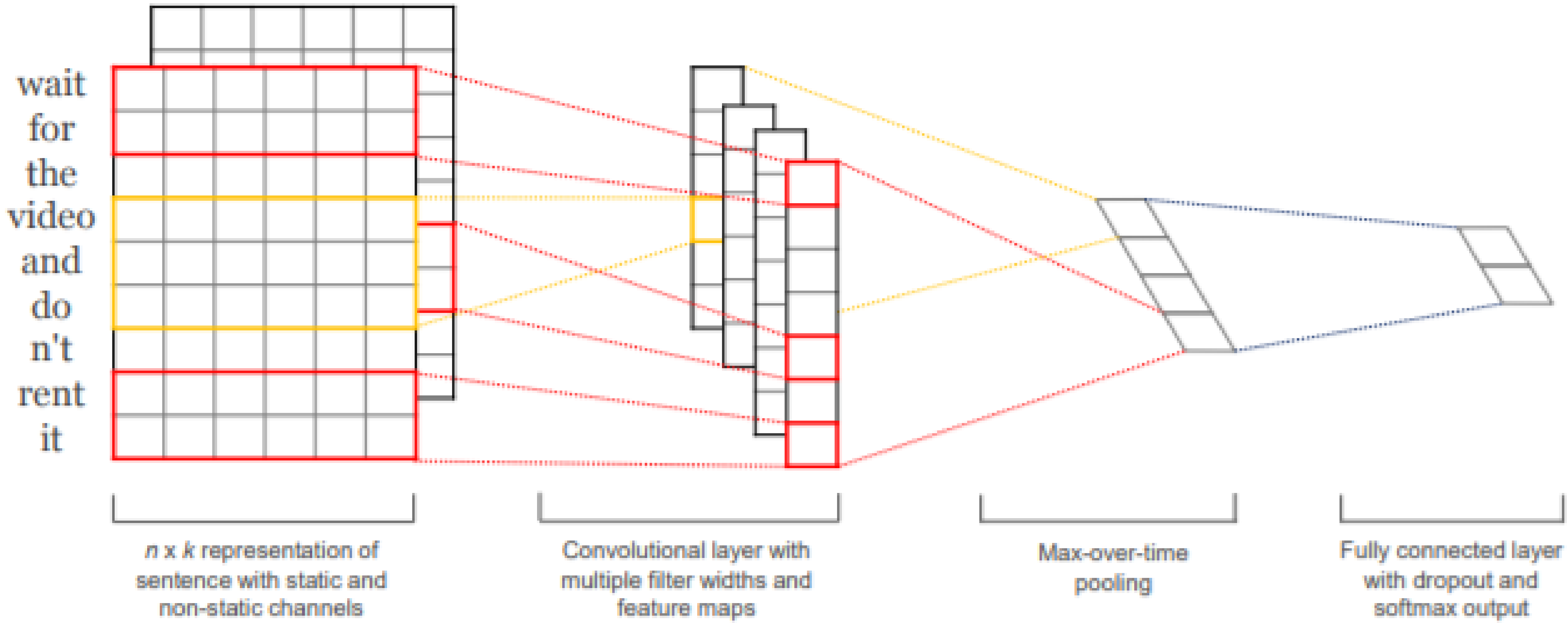
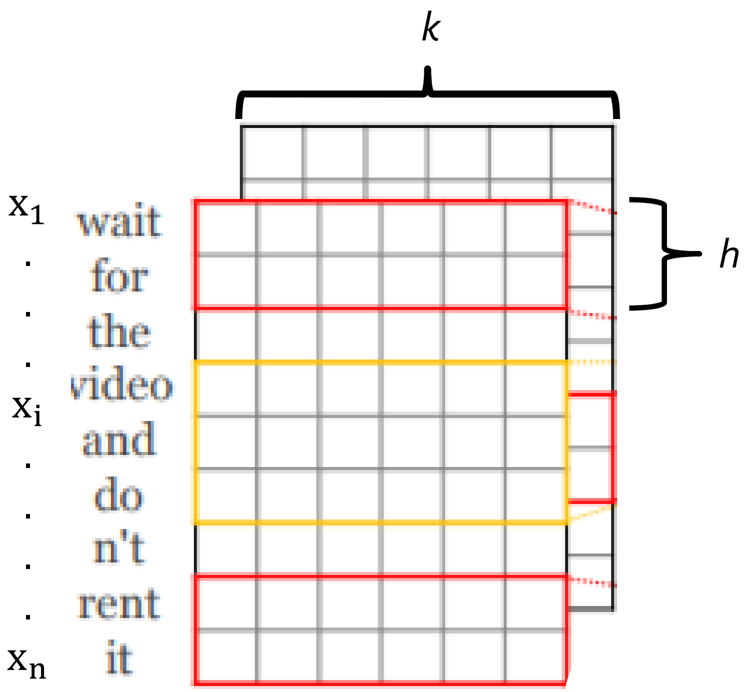


Figure 1: Model architecture with two channels for an example sentence.

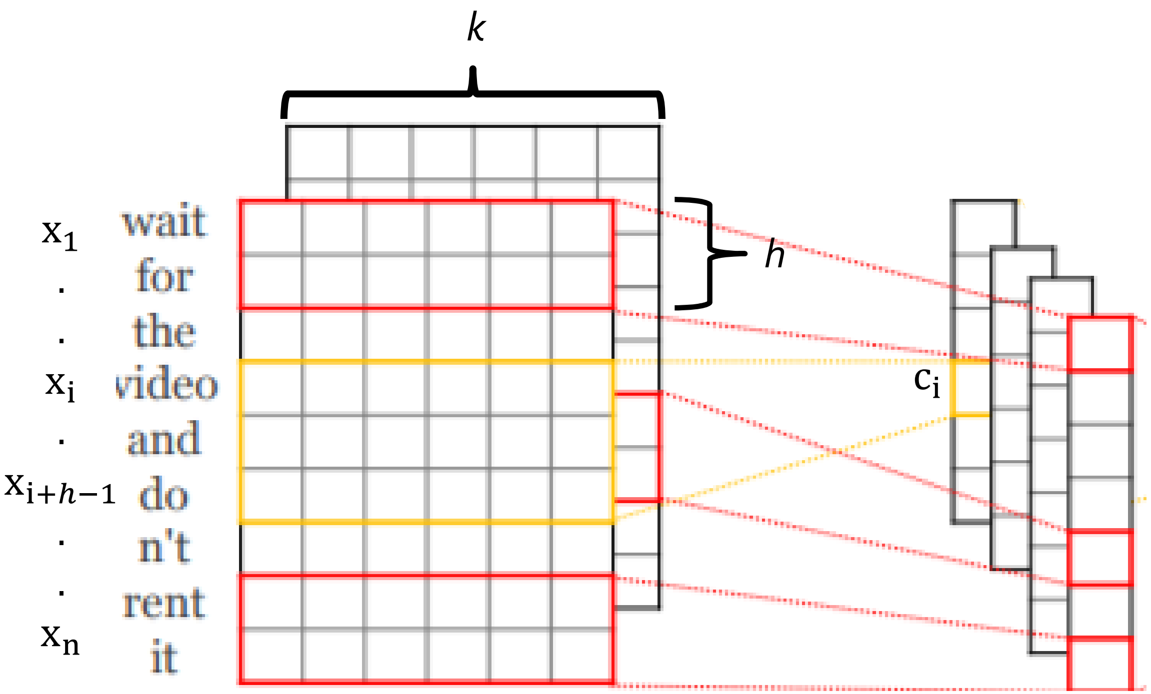
2. Model

Model Architecture
(1) Representation of Sentence



2. Model

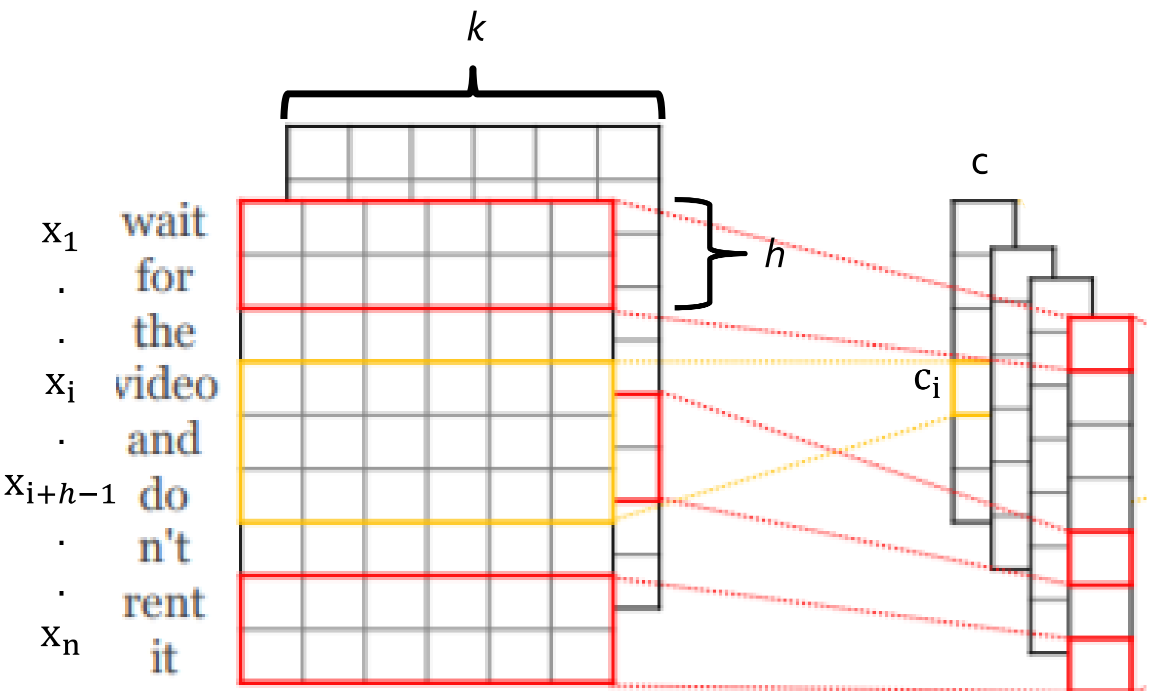
Model Architecture
(2) Convolution Operation



$$c_i = f(\mathrm{w} \cdot \mathrm{x}_{\mathrm{i:i+h-1}} + \mathrm{b})$$

2. Model

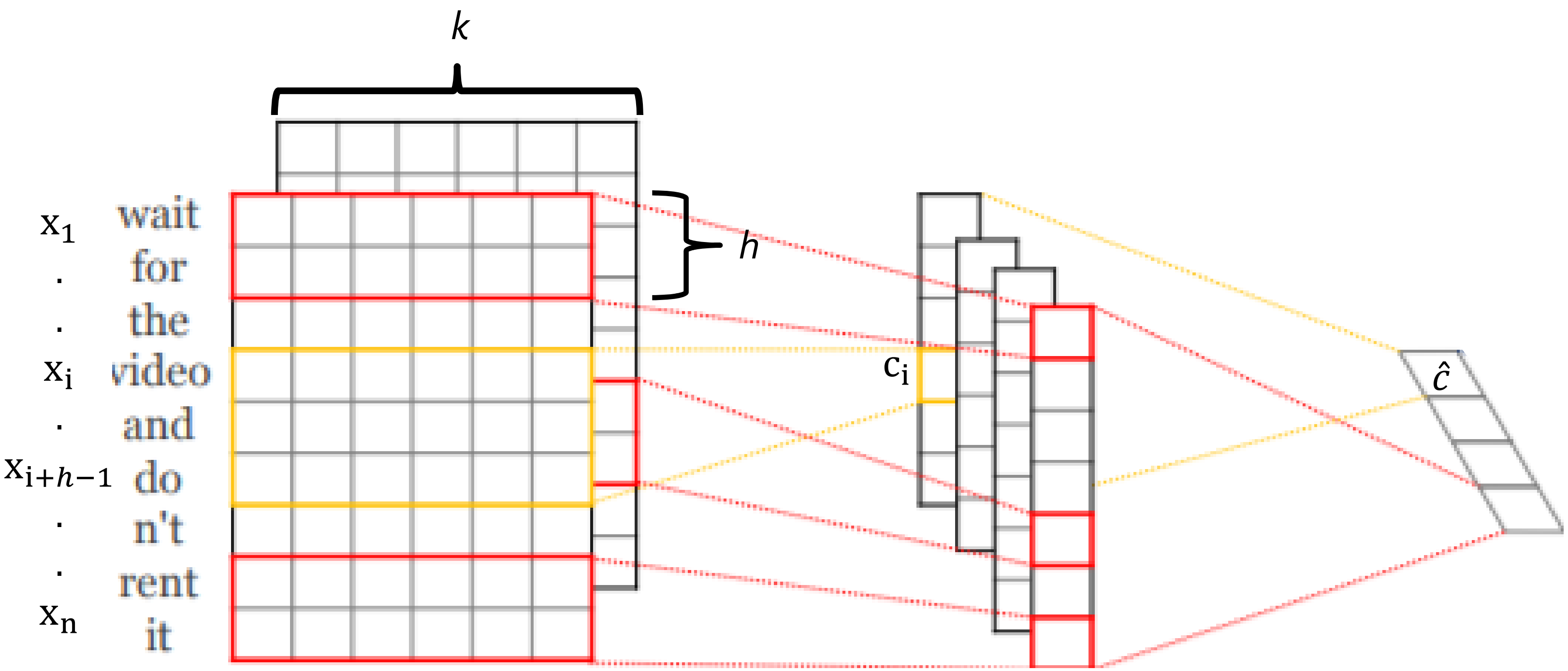
Model Architecture
(3) Feature map



$$c = [c_1, c_2, \dots c_i, \dots, c_{n-h+1}]$$

2. Model

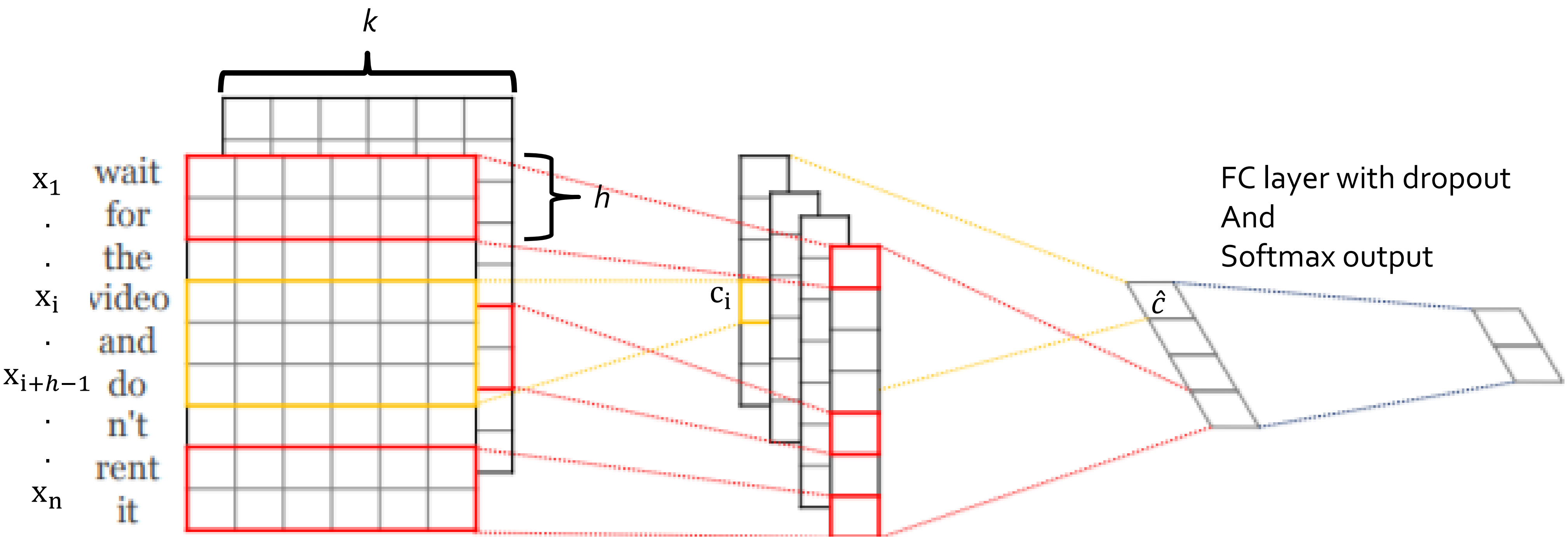
Model Architecture
(4) Max-over-time pooling



$$\hat{c} = \max\{c\}$$

2. Model

Model Architecture
(5) Dropout and Softmax output



2. Model

1. Regularization

1. Dropout

2. constraint on l_2 - norms of weight vectors

3. Data and Experimental setup

- MR : Movie reviews(pos / neg)
- SST-1 : vpos / pos / neu / neg / vneg
- SST-2 : Same as MR

- Subj : Sub / Obj
- TREC : Classifying a Q into 6 Q types(person, location, etc.)
- CR : Customer Reviews of product (pos / neg)
- MPQA : Opinion polarity detection.

Data	c	l	N	$\|V\|$	$\|V_{pre}\|$	*Test*
MR	2	20	10662	18765	16448	CV
SST-1	5	18	11855	17836	16262	2210
SST-2	2	19	9613	16185	14838	1821
Subj	2	23	10000	21323	17913	CV
TREC	6	10	5952	9592	9125	500
CR	2	19	3775	5340	5046	CV
MPQA	2	3	10606	6246	6083	CV

3. Data and Experimental setup

1. Hyper-parameters and Training

- Activation function : ReLU
- Filter windows (h) : 3, 4, 5 with 100 feature maps each
- Dropout rate (p) : 0.5
- l_2 constraint (s) : 3
- Mini-batch size : 50

- Chosen via a grid search on the SST-2 dev set.

3. Data and Experimental setup

1. Hyper-parameters and Training

- Early Stopping
- 10-fold CV
- SGD over shuffled mini-batches
- Adadelta

3. Data and Experimental setup

2. Pre-trained Word Vectors : word2vec

- Word2vec
- 100 billion words from Google News
- 300 Dimension
- trained using the CBOW architecture

3. Data and Experimental setup

3. Model Variations

- CNN-rand
- CNN-static
- CNN-non-static
- CNN-multichannel

4. Result and Discussion

- Pre-trained vectors are
- 1) good
- 2) ‘universal’ feature extractors
- 3) can be utilized across datasets

Model	MR	SST-1	SST-2	Subj	TREC	CR	MPQA
CNN-rand	76.1	45.0	82.7	89.6	91.2	79.8	83.4
CNN-static	81.0	45.5	86.8	93.0	92.8	84.7	**89.6**
CNN-non-static	**81.5**	48.0	87.2	93.4	93.6	84.3	89.5
CNN-multichannel	81.1	47.4	**88.1**	93.2	92.2	**85.0**	89.4
RAE (Socher et al., 2011)	77.7	43.2	82.4	–	–	–	86.4
MV-RNN (Socher et al., 2012)	79.0	44.4	82.9	–	–	–	–
RNTN (Socher et al., 2013)	–	45.7	85.4	–	–	–	–
DCNN (Kalchbrenner et al., 2014)	–	48.5	86.8	–	93.0	–	–
Paragraph-Vec (Le and Mikolov, 2014)	–	**48.7**	87.8	–	–	–	–
CCAE (Hermann and Blunsom, 2013)	77.8	–	–	–	–	–	87.2
Sent-Parser (Dong et al., 2014)	79.5	–	–	–	–	–	86.3
NBSVM (Wang and Manning, 2012)	79.4	–	–	93.2	–	81.8	86.3
MNB (Wang and Manning, 2012)	79.0	–	–	**93.6**	–	80.0	86.3
G-Dropout (Wang and Manning, 2013)	79.0	–	–	93.4	–	82.1	86.1
F-Dropout (Wang and Manning, 2013)	79.1	–	–	**93.6**	–	81.9	86.3
Tree-CRF (Nakagawa et al., 2010)	77.3	–	–	–	–	81.4	86.1
CRF-PR (Yang and Cardie, 2014)	–	–	–	–	–	82.7	–
SVM_S (Silva et al., 2011)	–	–	–	–	**95.0**	–	–

4. Result and Discussion

1. Multichannel vs Single Channel Models

- What they expected : Prevent Overfitting
- But the results are mixed.
- Further work on regularizing the fine-tuning process is warranted.

4. Result and Discussion

2. Static vs. Non-static Representations

- Fine tuned on the SST2 dataset.

- In Pre-trained word2vec, bad ≈ good
- In Non-static channel, bad ≈ terrible

For the word not in the set

- "!" ≈ effusive expressions
- "," ≈ conjunctive

	Most Similar Words for	
	Static Channel	Non-static Channel
bad	*good* *terrible* *horrible* *lousy*	*terrible* *horrible* *lousy* *stupid*
good	*great* *bad* *terrific* *decent*	*nice* *decent* *solid* *terrific*
n't	*os* *ca* *ireland* *wo*	*not* *never* *nothing* *neither*
!	*2,500* *entire* *jez* *changer*	*2,500* *lush* *beautiful* *terrific*
,	*decasia* *abysmally* *demise* *valiant*	*but* *dragon* *a* *and*

4. Result and Discussion

3. Further Observations

- Achieved more accuracy than existing Max-TDNN model(37.4% -> 45.0%)
- Dropout proved to be such a good regularizer
- When randomly initializing words not in word2vec, we obtained slight improvements by sampling each dimension from U[-a, a] where have the same variance as the pre-trained ones.
- Word2vec trained on google news gave far superior performance than word2vec trained on Wikipedia.
- Adadelta gave similar results to Adagrad but required fewer epochs.

5. Conclusion

- Unsupervised pre-training of word vectors is important ingredient in deep learning for NLP.

6. Reference

- https://www.slideshare.net/keunbongkwak/gloveglobal-vectors-for-word-representation
- https://shuuki4.wordpress.com/2016/01/27/word2vec-%EA%B4%80%EB%A0%A8-%EC%9D%B4%EB%A1%A0-%EC%A0%95%EB%A6%AC/
- https://wikidocs.net/22660
- https://www.quantumdl.com/entry/1%EC%A3%BC%EC%B0%A8-Convolutional-Neural-Networks-for-Sentence-Classification
- https://dreamgonfly.github.io/machine/learning,/natural/language/processing/2017/08/16/word2vec_explained.html
- https://datascienceschool.net/view-notebook/6927b0906f884a67b0da9310d3a581ee/
- https://www.youtube.com/watch?v=mPxi1YgU9Zw
- https://github.com/dennybritz/cnn-text-classification-tf
- http://docs.likejazz.com/cnn-text-classification-tf/

- <텐서플로와 머신러닝으로 시작하는 자연어처리>, 위키북스, 전창욱 외 2명