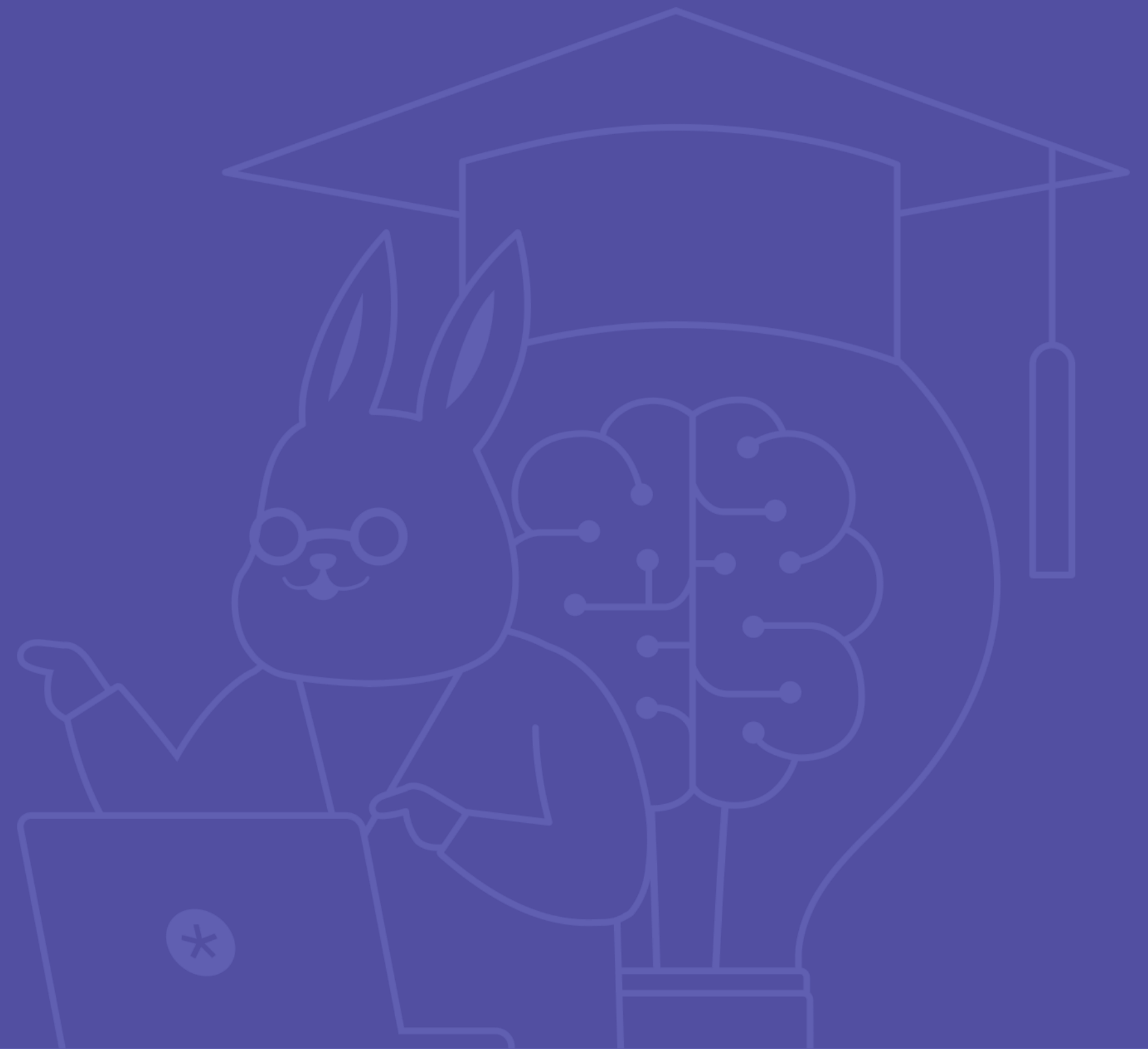


# 자연어 처리를 위한 딥러닝1

## 1장 Document Representation

JunnG\_T



# Contents

- 01. Text Overview
- 02. Count-based Representations
- 03. Document Similarity
- 04. DL-based Representations

01

# Text Overview



# 01 Text Overview

## ✓ What?

자연어처리의 전반적인 흐름 및 중요한 개념

각 과정마다 주로 사용되는 기법, 툴에 대한 소개

NLP가 무엇인지에 대해서 감을 잡자!

# 01 Text Overview

## ✓ What is NLP?

- 자연어(Natural Language)

우리가 일상생활에서 사용하는 언어

- 자연어처리(Natural Language Processing)

자연어의 의미를 분석하여 컴퓨터가 처리할 수 있도록 하는 일!

 인공지능의 주요 연구 분야!

# 01 Text Overview

## ✓ Why NLP?

자연어 이해 및 자연어 처리는 인공지능 분야에 있어서 필수적

빅데이터에서 주목받고 있는 것은 ‘비정형 데이터’

비정형 데이터 중 상당 부분이 텍스트 데이터

텍스트 데이터는 인간에 대한 정보를 많이 담고 있음

# 01 Text Overview

## ✓ Where NLP is used?

감성 분석, 주제분석

맞춤법 검사

번역, 질의응답

음성인식 스피커

# 01 Text Overview

## ✓ NLP Process

**1. Data Collection**  
(Crawling)

**3. Embedding**  
(Word2Vec, GloVe, FastText, BERT...)

**5. Modeling**  
(RNN, LSTM...)

**2. Tokenizing**  
(Konlpy, Mecab, Khaii...)

**4. Similarity**  
(Euclidean, Cosine, Jaccard...)

*/\* elice \*/*



# 01 Text Overview

## ✓ Step 1. Data Collection

Crwaling

The screenshot shows the Wikipedia page for '빅 데이터' (Big Data). A red box highlights the definition: '빅 데이터(영어: big data)란 기존 데이터베이스 관리도구의 능력을 넘어서는 대량(수십 테라바이트)의 정형 또는 심지어 데이터베이스 형태가 아닌 비정형의 데이터 집합조차 포함한[1] 데이터로부터 가치를 추출하고 결과를 분석하는 기술[2]이다.' The Chrome DevTools overlay shows the DOM tree with the selected element being the first heading, and the CSS styles pane showing the default font-family of 'sans-serif'.

로그인하지 않음 토론 기여 계정 만들기 로그인

문서 토론 읽기 편집 역사 보기 위키백과 검색

2019년 1차 인문학 에디터톤이 8월 31일에 열립니다. [숨기기]

## 빅 데이터

위키백과, 우리 모두의 백과사전.

**빅 데이터**(영어: big data)란 기존 데이터베이스 관리도구의 능력을 넘어서는 대량(수십 테라바이트)의 정형 또는 심지어 데이터베이스 형태가 아닌 비정형의 데이터 집합조차 포함한[1] 데이터로부터 가치를 추출하고 결과를 분석하는 기술[2]이다.

다양한 종류의 대규모 데이터에 대한 생성, 수집, 분석, 표현을 그 특징으로 하는 빅 데이터 기술의 발전은 다변화된 현대 사회를 더욱 정확하게 예측하여 효율적으로 작동케 하고 개인화된 현대 사회 구성원마다 맞춤형 정보를 제공, 관리, 분석 가능케 하며 과거에는 불가능했던 기술을 실현시키기도 한다.

이같이 빅 데이터는 정치, 사회, 경제, 문화, 과학 기술 등 전 영역에 걸쳐서 사회와 인류에게 가치있는 정보를 제공할 수 있는 가능성을 제시하며 그 중요성이 부각되고 있다.

위키백과의 편집 현황의 시각화 자료(IBM 작성). 수 테라바이트의 용량을 지닌 위키백과의 텍스트 및 이미지 자료는 빅 데이터의 고전적 사례에 속한다.

Filter :hov .cls +

element.style { }

.mw-body load.php?la...in=vector:1

h1:lang(ja), .mw-body-content

h1:lang(ja), .mw-body-content

h2:lang(ja), .mw-body h1:lang(he), .mw-body-content h1:lang(he), .mw-body-content h2:lang(he), .mw-body h1:lang(ko), .mw-body-content h1:lang(ko), .mw-body-content h2:lang(ko) {

font-family: sans-serif;

.mw-body load.php?la...in=vector:1

.firstHeading {

overflow: visible;

margin -

border -

padding -

577.667 x 37.333

1

7.200

Filter Show all

background-attachment scroll

background-clip border-box

background-color

Console What's New

Highlights from the Chrome 76 update

Autocomplete with CSS keyword values

Typing a keyword value like "bold" in the Styles pane now autocompletes to "font-weight: bold".

A new UI for network settings

The "Use large request rows", "Group by frame", "Show overview", and "Capture screenshots" options have moved to the new Network Settings pane.

p

Clear (41) Toggle Position XPath

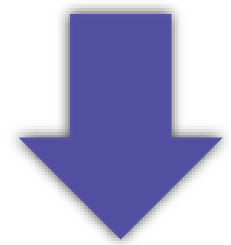
? X

# 01 Text Overview

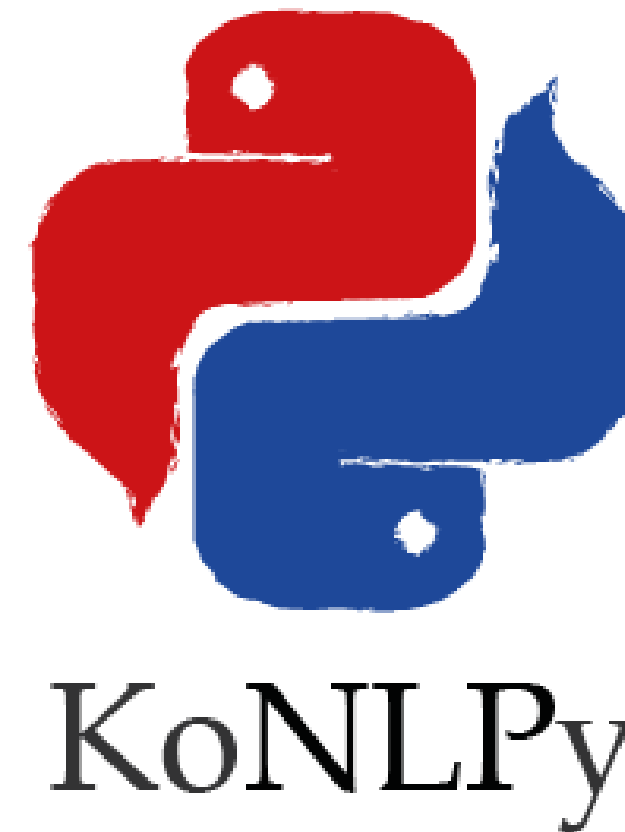
## ✔ Step 2. Tokenizing(Konlpy, Mecab, Khaii...)

“ 문장을 의미가 있는 단위로 나눔. ”

EX) 사과가 상했다



‘사과/N’, ‘가/J’, ‘상하/V’, ‘쌔/E’, ‘다/E’



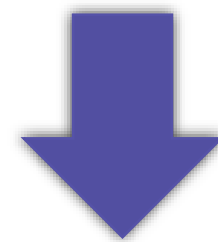


# 01 Text Overview

## ✓ Step 3. Embedding

“토큰을 컴퓨터가 알아들을 수 있도록 숫자로 바꿈”

사과가 상했다



‘사과’: [0.1234, 0.1234] ‘가’: [0.5678, 0.1234] ‘상하’: [0.7890, 0.1567] ‘ㅅ’: [0.9021, 0.4321] ‘다’: [0.0876, 0.3579]

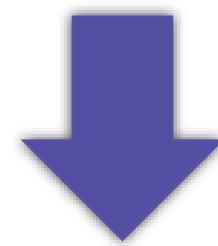
`/* elice */`

# 01 Text Overview

## ✓ Step 4. Similarity

“단어 또는 문장간의 유사한 정도를 구한다!”

‘사과’: [0.9021, 0.4321] ‘가’: [0.5678, 0.1234] ‘상하’: [0.3456, 0.1764] ‘쓰’: [0.1234, 0.1234] ‘다’: [0.0876, 0.3579]



‘사과’: [0.9021, 0.4321] ‘상하’: [0.3456, 0.1764]

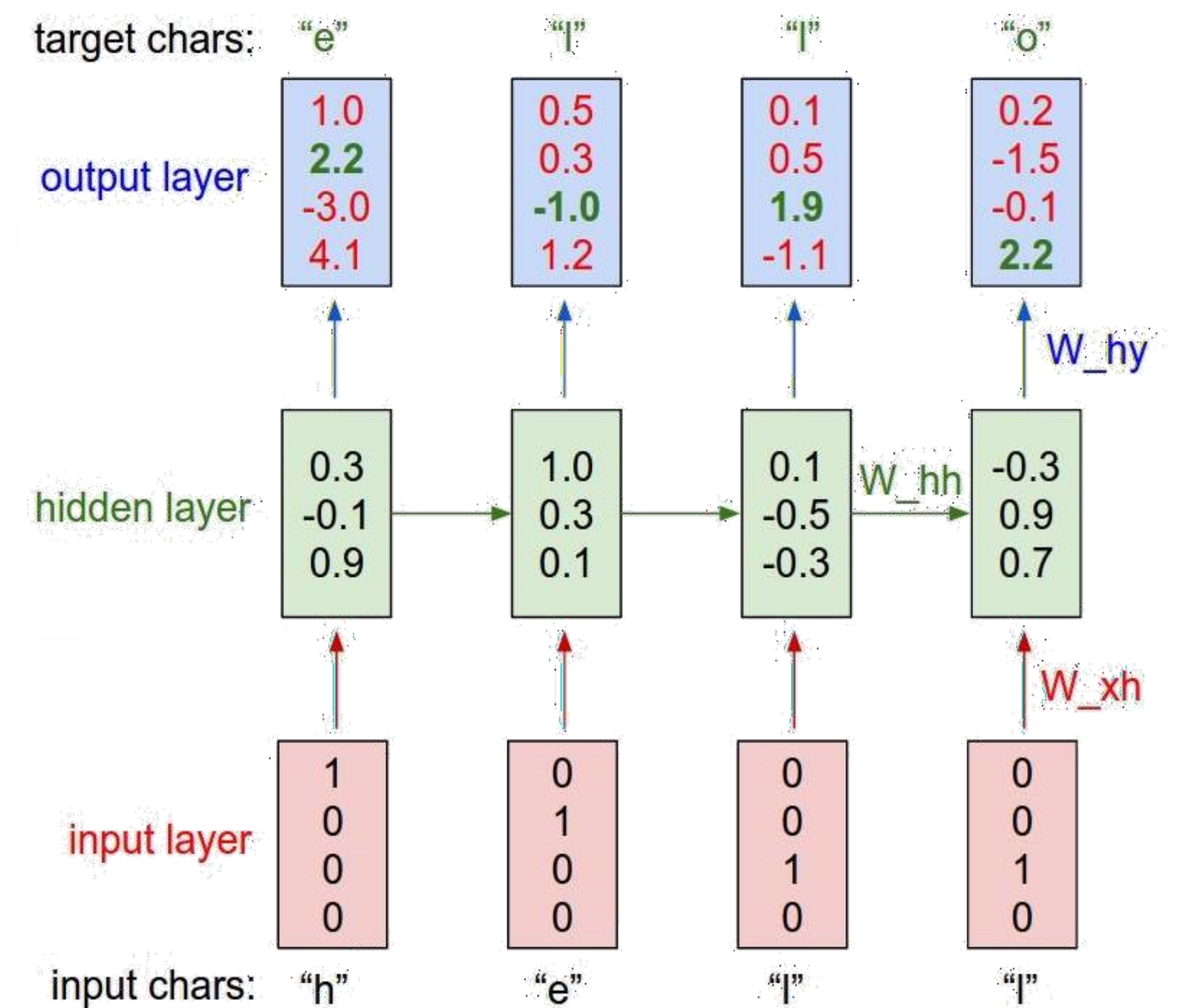
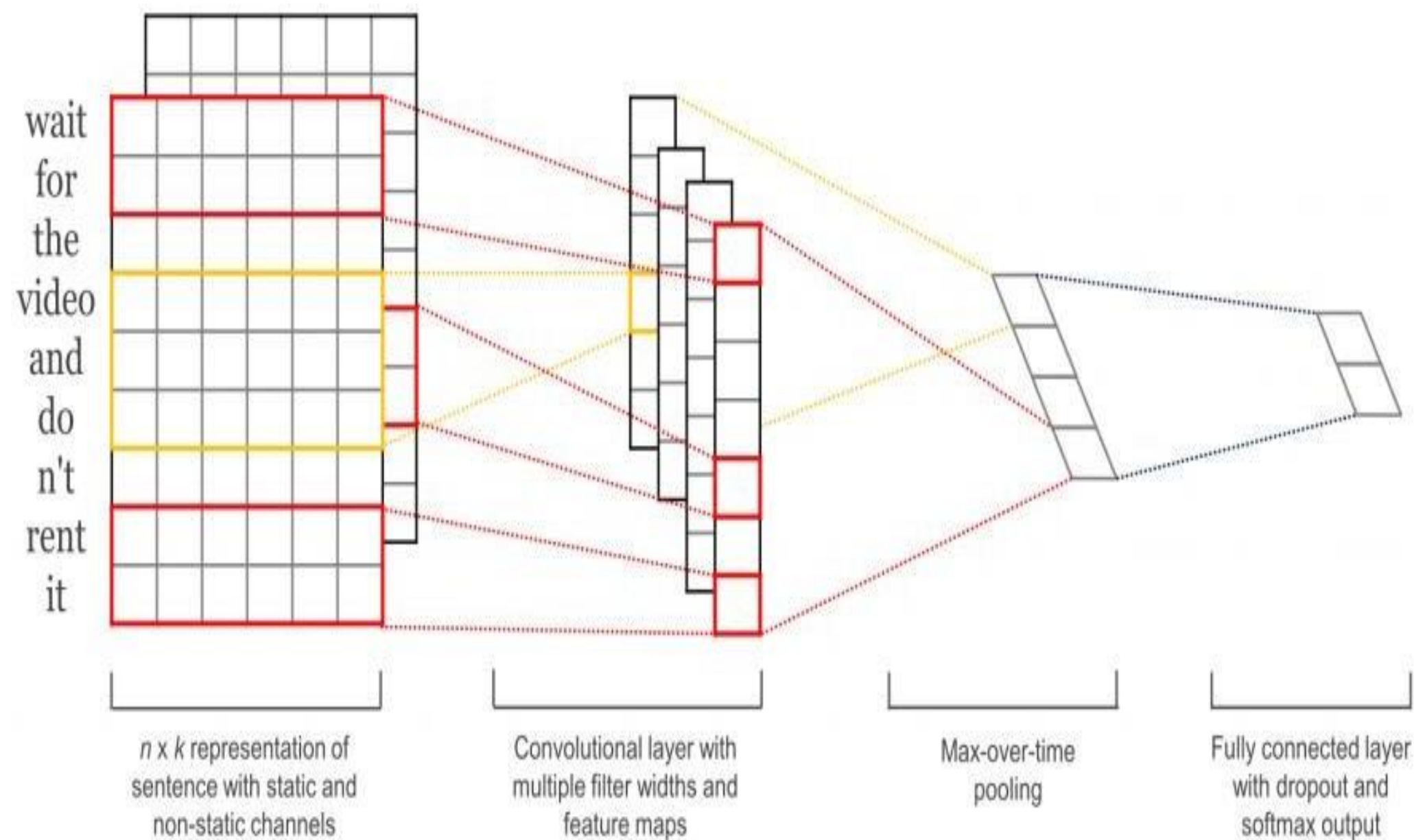
코사인 유사도에 따르면, 이 두 토큰은 유사하다고 판단할 수 있다.

*/\* elice \*/*

# 01 Text Overview

## ✓ Step 5. Modeling

“앞에서 한 준비로 생성, 요약, 추천 등 다양한 task를 수행한다!”



/\* elice \*/

# 01 Text Overview

## ✓ NLP Process

**1. Data Collection**  
(Crawling)

**3. Embedding**  
(Word2Vec, GloVe, FastText, BERT...)

**5. Modeling**  
(RNN, LSTM...)

**2. Tokenizing**  
(Konlpy, Mecab, Khaii...)

**4. Similarity**  
(Euclidean, Cosine, Jaccard...)

# 01 Text Overview

## ✓ Tokenizing(Konlpy, Mecab, Khaii...)

- Q1) Tokenizing, 왜 하나요?
- A1) 자연어 처리를 위한 의미단위를 만들기 위해서
- Q2) What is “token”?
- A2) 의미를 가지는 요소 ex) 자소/음소, 형태소, 단어, 문장, 문서(영어의 경우 흔히 단어, 한국어의 경우 흔히 형태소 단위로 토큰화)
- Q3) What is “형태소 분석(POS-tagging)”?
- A3) 원시말뭉치를 형태소 단위로 쪼개고 각 형태소에 품사 정보를 부착하는 작업

# 01 Text Overview

## ✔ Tokenizing(Konlpy, Mecab, Khaii...)

Hannanum	Kkma	Komoran	Twitter	Mecab	Khaii
아버지가방 에들어가/N	아버지/NNG	아버지가방에 들어가신다/NNP	아버지 /Noun	아버지/NNG	아버지/NNG
이/J	가방/NNG		가방/Noun	가/JKS	가/JKS
시나다/E	에/JKM		에/Josa	방/NNG	방/NNG
	들어가/VV		들어가신 /Verb	에/JKB	에/JKB
	시/EPH		다/Eomi	들어가/VV	들어가/VV
	s다/EFN			신다/EP+EC	시/EP
					다/EC

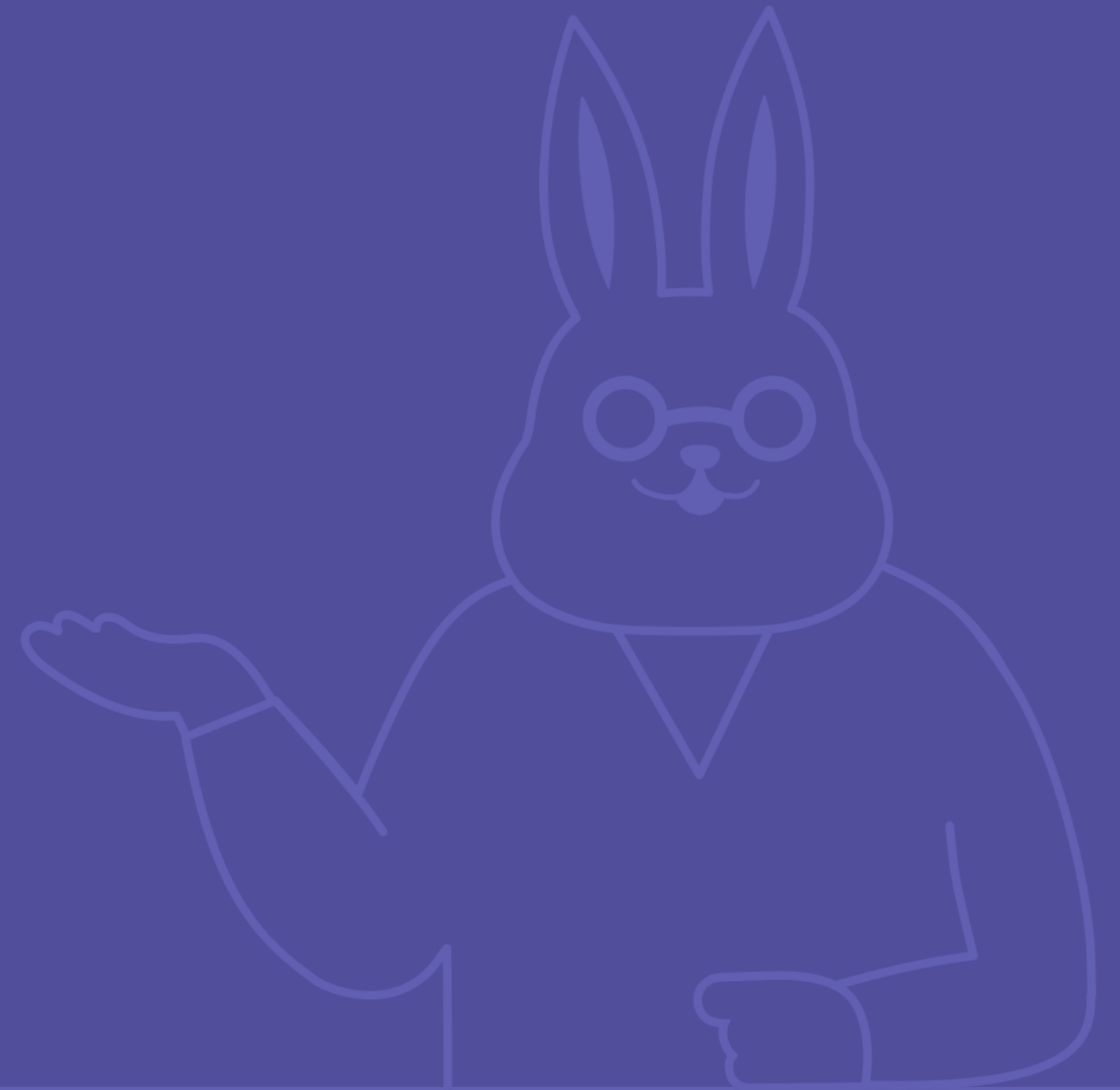
- 사용할 데이터의 특성(띄어쓰기 유무 등)이나 개발 환경(Python, Java)에 따라서 적합한 형태소 분석기를 고려해야함
- 연산 속도가 중요하다면 mecab을 최우선으로 고려해야하며, 심지어 분석 품질도 상위권으로 보여짐
- 자소 분리나 오탈자에 대해서도 어느 정도 분석 품질이 보장되어야 한다면 KOMORAN 사용을 고려
- 한나눔과 khaii는 일부 케이스에 대한 분석 품질, 꼬꼬마는 분석 시간에서 약간 아쉬운 점이 보임

/\* elice \*/



02

# Count-based Representations



## 02 Count-based Representations

### ✓ NLP Process

**1. Data Collection**  
(Crawling)

**3. Embedding**  
(Word2Vec, GloVe, FastText, BERT...)


**5. Modeling**  
(RNN, LSTM...)

**2. Tokenizing**  
(Konlpy, Mecab, Khaii...)

**4. Similarity**  
(Euclidean, Cosine, Jaccard...)

## 02 Count-based Representations

## ✔ NLP Overview



Cornell University

Library

We gratefully acknowledge support from


the Simons Foundation

and member institution

arXiv.org > search

Search or Article ID

All papers



[\(help\)](#) | [Advanced search](#)

arXiv.org Search Results

Back to Search form

|

Next 25 results

The URL for this search is [http://arxiv.org:443/find/all/1/all:+EXACT+text\\_mining/0/1/0/all/0/1](http://arxiv.org:443/find/all/1/all:+EXACT+text_mining/0/1/0/all/0/1)

Showing results 1 through 25 (of 168 total) for all:"text mining"

1. [arXiv:1703.05692](#) [pdf]

OncoScore: a novel, Internet-based tool to assess the oncogenic potential of genes

Rocco Piazza, Daniele Ramazzotti, Roberta Spinelli, Alessandra Pirola, Luca De Sano, Pierangelo Ferrari, Vera Magistroni, Nicoletta Cordani, Nitesh Sharma, Carlo Gambacorti-Passerini

Subjects: Genomics (q-bio.GN); Quantitative Methods (q-bio.QM)

2. [arXiv:1703.04213](#) [pdf, other]

MetaPAD: Meta Pattern Discovery from Massive Text Corpora

Meng Jiang, Jingbo Shang, Taylor Cassidy, Xiang Ren, Lance M. Kaplan, Timothy P. Hanratty, Jiawei Han

Comments: 9 pages

Subjects: Computation and Language (cs.CL)

3. [arXiv:1703.02819](#) [pdf, other]

Introduction to Formal Concept Analysis and Its Applications in Information Retrieval and Related Fields

Dmitry I. Ignatov

Journal-ref: RuSSIR 2014, Nizhny Novgorod, Russia, CCIS vol. 505, Springer 42-141

Subjects: Information Retrieval (cs.IR); Artificial Intelligence (cs.AI); Computation and Language (cs.CL); Discrete Mathematics (cs.DM); Machine Learning (stat.ML)

4. [arXiv:1702.07117](#) [pdf, other]

LTSG: Latent Topical Skip-Gram for Mutually Learning Topic Model and Vector Representations

Jarvan Law, Hankui Zhuo, Junhua He, Erhu Rong (Dept. of Computer Science, Sun Yat-Sen University, GuangZhou, China.)

Subjects: Computation and Language (cs.CL)

5. [arXiv:1702.03519](#) [pdf, ps, other]

A Technical Report: Entity Extraction using Both Character-based and Token-based Similarity

Zeyi Wen, Dong Deng, Rui Zhang, Kotagiri Ramamohanarao

Comments: 12 pages, 6 figures, technical report

Subjects: Databases (cs.DB)

The complicated, evolving landscape of cancer  
Mining textual patterns in news, tweets, papers, and  
This paper is a tutorial on Formal Concept Analysis (FCA) and its applications. FCA is an applied branch of Lattice Theory, a mathematical discipline which enables formalisation of concepts as basic units of human thinking and analysing data in the object-attribute form. Originated in early 80s, during the last three decades, it became a popular human-centred tool for knowledge representation and data analysis with numerous applications. Since the tutorial was specially prepared for RuSSIR 2014, the covered FCA topics include Information Retrieval with a focus on visualisation aspects, Machine Learning, Data Mining and Knowledge Discovery, Text Mining and several others.

pattern quality assessment function, which avoids costly dependency parsing and generates high-quality patterns; (2) it identifies and groups synonymous meta patterns from multiple facets---their types, contexts, and extractions; and (3) it examines type distributions of entities in the instances extracted by each group of patterns, and looks for appropriate type levels to make discovered patterns precise. Experiments demonstrate that our proposed framework discovers high-quality typed textual patterns efficiently from different genres of massive corpora and facilitates information extraction.

the complex evolving landscape of cancer mutations pose a  
formine textual pattern in news tweet paper and mani  
list other  
priority text  
to a decision  
too the  
based on  
curve and  
curated data  
oncology  
oncology  
priority o

this paper is a tutorial on formal concept analysis (fca) and  
its application fca is an applied branch of lattice theory a ma  
thematical discipline which enables formalisation of concepts as  
basic unit of human thought and analysis data in the  
object attribute form origin in early 1980s during the last three  
decades it became a popular human-centred tool for  
knowledge representation and data analysis with numerous applic  
since the tutorial was specially prepared for Russia the covered  
fca topics include information retrieval with a focus on visualisation  
aspect machine learning data mining and knowledge discovery  
text mining and several other

# 1. Data Collection (Crawling)

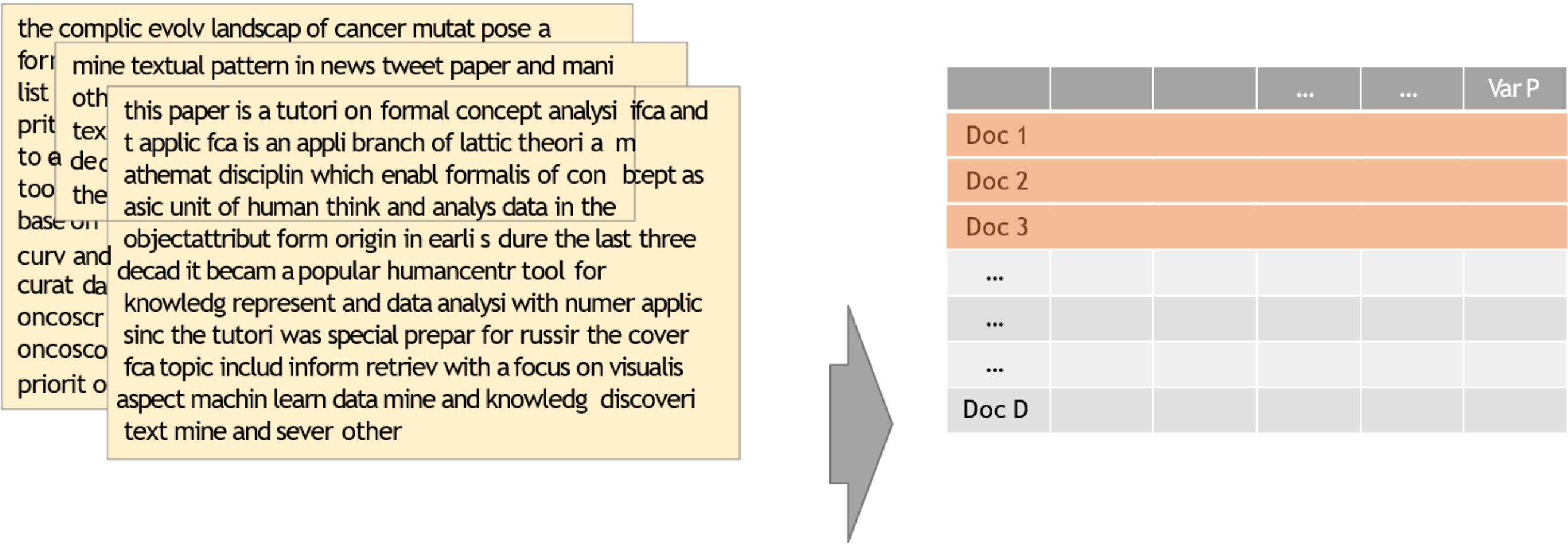
## 1.5 Preprocessing (Stopwording)

```
/* elice */
```

# 02 Count-based Representations

## ✓ NLP Overview

### 비정형 데이터를 정형데이터로 바꾸는 작업



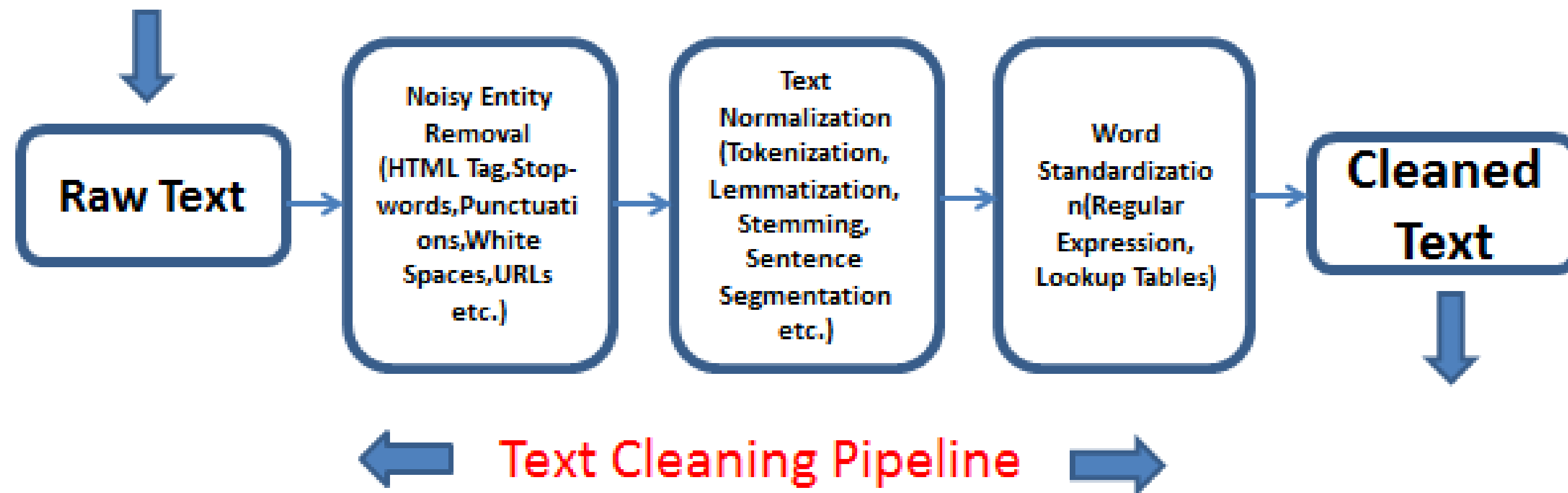
### 2. Tokenizing & 3. Embedding (Konlpy, Mecab, Khaii...)



## 02 Count-based Representations

### ✓ Text Preprocessing

- 1. 특수문자 : 특수문자의 경우 중요한 의미를 내보하지 않아 삭제하는 경우가 많다.
- 2. 숫자 : 숫자의 경우 필요없다 생각할 수 있겠지만 특정 도메인에서는 매우 중요할 수 있다.
- 3. 조사,접속사와 같은 문장에서 필요없는 것들 제거하면 좋을 수 있다.



# 02 Count-based Representations

## ✓ Stop Wording

- 중요한 의미를 포함하지 않은 Word
- 매우 문법적인 표현인 단어(조사, 접속사)
- 주로 이러한 Word를 삭제하면 성능이 개선된다.(Feature Selection 관점)

Korean Stopwords		
아	어찌됐든	하기보다는
휴	그위에	차라리
아이구	게다가	하는 편이 낫다
아이구	점에서 보아	흐흐
아이고	비추어 보아	놀라다
어	고려하면	상대적으로 말하
나	하게될것이다	자면
우리	일것이다	마치
저희	비교적	아니라면
따라	좀	셋
의해	보다더	그렇지 않으면
을	비하면	그렇지 않다면
를	시키다	안 그러면
에	하게하다	아니었다면
의	할만하다	하든지
가	의해서	아니면
으로	연이서	이라면
로	이어서	좋아
에게	잇따라	알았어
뿐이다	뒤따라	하는것도
의거하여	뒤이어	그만이다
근거하여	결국	어쩔수 없다
입각하여	의지하여	하나
기준으로	기대여	일
예하면	통하여	일반적으로
예를 들면	자마자	일단
예를 들자면	더욱더	한편으로는
저	불구하고	오자마자
소인	얼마든지	이렇게되면
소생	마음대로	이와같다면

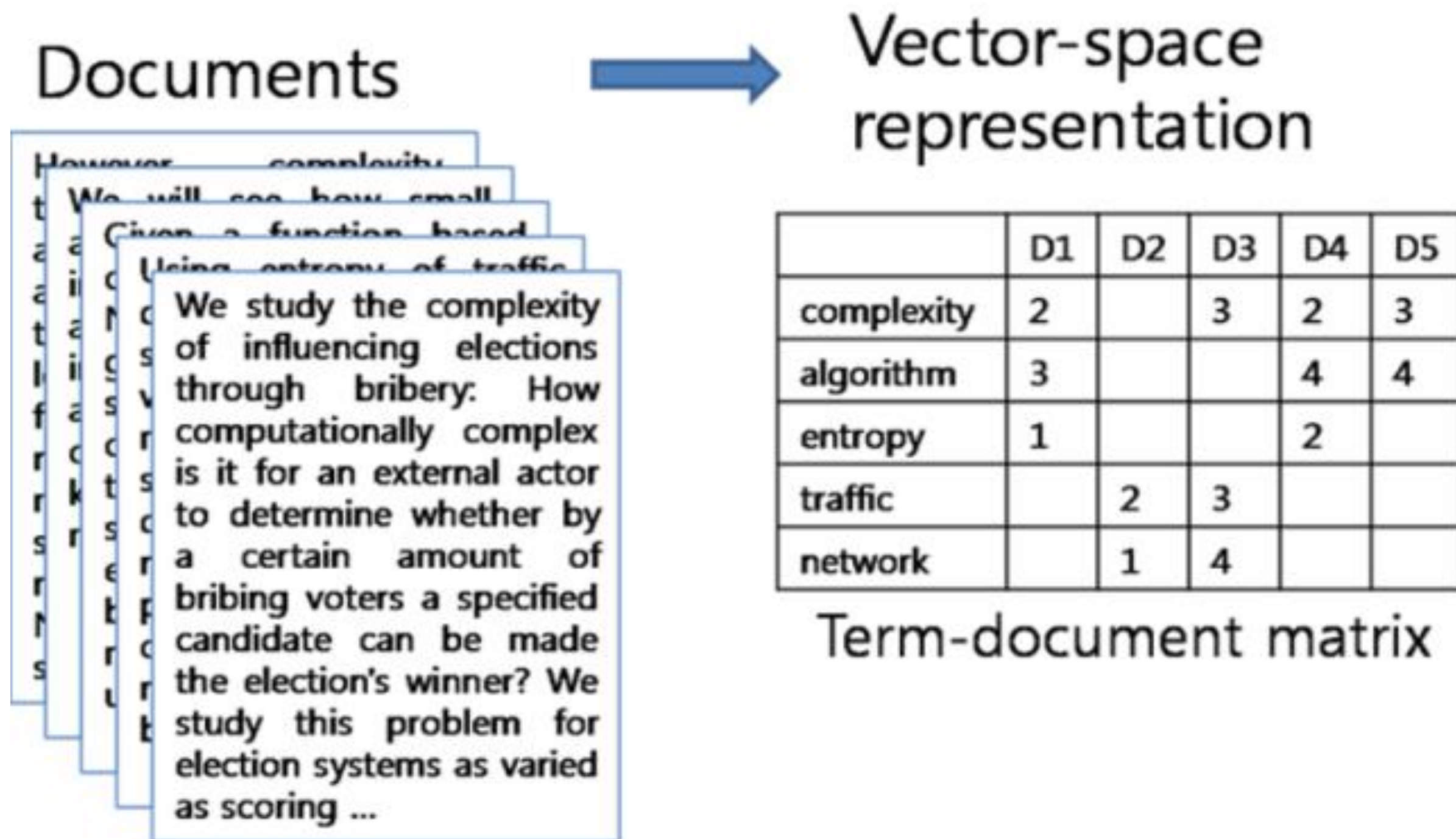
지말고	곧	한마디	불문하고	픽	그리하여
하지마	즉시	한항목	향하여	펄렁	여부
하지마라	바로	근거로	향해서	동안	하기보다는
다른	당장	하기에	향하다	이래	하느니
물론	하자마자	아울러	쪽으로	하고있었다	하면 할수록
또한	밖에 안된다	하지 않도록	틈타	이었다	운운
그리고	하면된다	않기 위해서	이용하여	에서	이러이러하다
비길수 없다	그래	이르기까지	타다	로부터	하구나
해서는 안된	그렇지	이 되다	오르다	까지	하도다
다	요컨대	로 인하여	제외하고	예하면	다시말하면
뿐만 아니라	다시 말하자면	까닭으로	이 외에	했어요	다음으로
만이 아니다	바꿔 말하면	이유만으로	이 밖에	해요	에 있다
만은 아니다	즉	이로 인하여	하여야	함께	에 달려 있다
막론하고	구체적으로	그래서	비로소	같이	우리
관계없이	말하자면	이 때문에	한다면 몰라	더불어	우리들
그치지 않다	시작하여	그러므로	도	마저	오히려
그러나	시초에	그런 까닭에			
그런데	이상	알 수 있다			
하지만	허	결론을 낼 수 있			

/\* elice \*/

## 02 Count-based Representations

### ✓ Document Representation

- 어떻게 Document를 정형데이터로 표현할 것인가?
- 머신러닝 알고리즘을 사용하기 위해서 비정형 데이터를 정형데이터로 바꾸는 작업이 필요함.

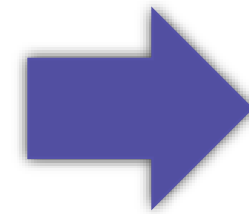


## 02 Count-based Representations

### ✓ Bag-of-words

- Document를 표현하는 가장 쉬운 방법으로 텍스트가 word의 순서를 고려하지 않는 Vector로 표현된다.
- 또한 표현된 Vector는 discrete space에 존재함.

Review 1: This movie is very scary and long  
Review 2: This movie is not scary and is slow  
Review 3: This movie is spooky and good



	1 This	2 movie	3 is	4 very	5 scary	6 and	7 long	8 not	9 slow	10 spooky	11 good	Length of the review(in words)
Review 1	1	1	1	1	1	1	1	0	0	0	0	7
Review 2	1	1	2	0	0	1	1	0	1	0	0	8
Review 3	1	1	1	0	0	0	1	0	0	1	1	6

/\* elice \*/



## 02 Count-based Representations

### ✓ Bag-of-words : Term-Document Matrix

- Document를 표현하는 가장 쉬운 방법으로 텍스트가 word의 순서를 고려하지 않는 Vector로 표현된다.
- Binary / Frequency representation 방법 2가지가 존재한다.

S1: John likes to watch movies. Mary likes too.

S2: John also likes to watch football game.

Binary  
representation

Word	S1	S2
John	1	1
Likes	1	1
To	1	1
Watch	1	1
Movies	1	0
Also	0	1
Football	0	1
Games	0	1
Mary	1	0
too	1	0

Frequency  
representation

Word	S1	S2
John	1	1
Likes	2	1
To	1	1
Watch	1	1
Movies	1	0
Also	0	1
Football	0	1
Games	0	1
Mary	1	0
too	1	0

/\* elice \*/

## 02 Count-based Representations

### ✓ Bag-of-words : Term-Document Matrix

Binary representation

Word	S1	S2
John	1	1
Likes	1	1
To	1	1
Watch	1	1
Movies	1	0
Also	0	1
Football	0	1
Games	0	1
Mary	1	0
too	1	0

Frequency representation

Word	S1	S2
John	1	1
Likes	2	1
To	1	1
Watch	1	1
Movies	1	0
Also	0	1
Football	0	1
Games	0	1
Mary	1	0
too	1	0

Document Embedding Vector

Word Embedding Vector

/\* elice \*/

## 02 Count-based Representations

### ✓ Bag-of-words : Term-Document Matrix

- Text의 내용이 Word Frequency에 의해 표현된다.
- 임베딩의 결과가 word의 순서를 고려하지 않은채 표현된다.  
ex) John is quicker than Mary = Mary is quicker than John in BOW representation
- 임베딩의 결과만 놓고 봤을 때 Raw Text를 정확히 알 수 없다.

Original text

John loves Mary

BOW

(..., John, ..., loves, ..., Mary, ...)  
(..., 1, ..., 1, ..., 1, ...)

Reconstruction

John loves Mary?  
Mary loves John?

/\* elice \*/

## 02 Count-based Representations

### ✓ Bag-of-words : Term Frequency – Inverse Document Frequency

#### ① Term frequency $tf_{t,d}$

- Document  $d$ 의 단어  $t$ 의 빈도  $\rightarrow$  빈도가 높을수록 중요(Weighting이 높아짐)

	Antony and Cleopatra	Julius Caesar	The Tempest	Hamlet	Othello	Macbeth
Antony	157	73	0	0	0	0
Brutus	4	157	0	1	0	0
Caesar	232	227	0	2	1	1
Calpurnia	0	10	0	0	0	0
Cleopatra	57	0	0	0	0	0
mercy	2	0	3	5	5	1
worser	2	0	1	1	1	0

/\* elice \*/

## 02 Count-based Representations

### ✓ Bag-of-words : Term Frequency – Inverse Document Frequency

#### ② Document frequency $df_t$

- 단어  $t$ 가 등장하는 Documents의 개수
- 전체 문서에서 희소하게 등장하는 단어가 매우 중요한 단어일 것이다.  
(Weighting이 높아짐)
- 즉, 희소하게 등장하는 단어일 수 록 높은 Weight를 준다.
- Q)  $df_t$ 가 높을 수록 중요할까요?

## 02 Count-based Representations

### ✓ Bag-of-words : Term Frequency – Inverse Document Frequency

#### ③ Inverse document frequency $idf_t$

- $idf_t = \log_{10}(N/df_t)$ , where  $N$  : The number of documents
- 여기서 Log는 Numerical Error를 피하기 위해 사용

IDF example with  $N = 1$  million

term	$df_t$	$idf_t$
calpurnia	1	6
animal	100	4
sunday	1,000	3
fly	10,000	2
under	100,000	1
the	1,000,000	0

/\* elice \*/

## 02 Count-based Representations

### ✓ Bag-of-words : Term Frequency – Inverse Document Frequency

④ TF-IDF : TF-IDF는 TF와 IDF의 곱으로 표현되는 단어의 가중치로 표현된 Embedding 방법이다.

$$TF - IDF(w) = tf(w) \times \log\left(\frac{N}{df(w)}\right)$$

한 Document에서 빈번하게 등장하는 단어일수록 커짐

전체 Document에서 희소하게 등장하는 단어일수록 커짐

단순 Frequency보다 훨씬 정보를 풍부하게 담고있다.

/\* elice \*/

# 02 Count-based Representations

## ✓ Bag-of-words : Term Frequency – Inverse Document Frequency

	Antony and Cleopatra	Julius Caesar	The Tempest	Hamlet	Othello	Macbeth
Antony	157	73	0	0	0	0
Brutus	4	157	0	1	0	0
Caesar	232	227	0	2	1	1
Calpurnia	0	10	0	0	0	0
Cleopatra	57	0	0	0	0	0
mercy	2	0	3	5	5	1
worser	2	0	1	1	1	0

	Antony and Cleopatra	Julius Caesar	The Tempest	Hamlet	Othello	Macbeth
Antony	1.5	1.5	1.5	1.5	1.5	1.5
Brutus	2.3	2.3	2.3	2.3	2.3	2.3
Caesar	0.2	0.2	0.2	0.2	0.2	0.2
Calpurnia	0	0	0	0	0	0
Cleopatra	1	1	1	1	1	1
mercy	5	5	5	5	5	5
worser	3	3	3	3	3	3

X  
Element Wise!!!!

/\* elice \*/



## 02 Count-based Representations

### ✓ Bag-of-words : Term Frequency – Inverse Document Frequency

각각의 Document는 TF-IDF 가중치를 갖는 real-valued vector로 표현된다.

$|V|$ -dimensional vector space

- 각 단어들이 각 Feature가 된다.
- Document는 공간에서의 하나의 점으로 표시됨(Word도 마찬가지로)-> Embedding
- 그러나, unique한 단어의 갯수가 Feature의 갯수가 되므로 매우 차원이 높아진다.
- 0의 값이 매우 많이 존재한다.

# 02 Count-based Representations

## ✓ Bag-of-words : Term Frequency – Inverse Document Frequency

	Antony and Cleopatra	Julius Caesar	The Tempest	Hamlet	Othello	Macbeth
Antony	5.25	3.18	0	0	0	0
Brutus	1.21	6.1	0	1	0	0
Caesar	8.59	2.54	0	1.51	0.25	0.35
Calpurnia	0	1.54	0	0	0	0
Cleopatra	2.85	0	0	0	0	0
mercy	1.51	0	1.9	0.12	5.25	0.88
worser	1.37	0	0.11	4.15	0.25	1.95

Document Embedding Vector

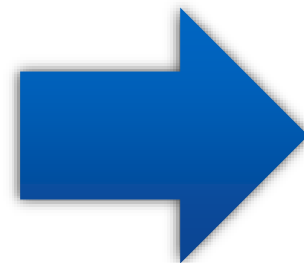
Word Embedding Vector

## 02 Count-based Representations

### ✓ Bag-of-words : Term Frequency – Inverse Document Frequency

- Q1: Which term is the **most** important for the document 1?
- Q2: Which term is the **least** important for the document 1?

	Doc1	Doc2	Doc3
Term1	5	0	0
Term2	1	0	0
Term3	5	5	5
Term4	3	3	3
Term5	3	0	1



	Doc1	TF	DF	IDF	TF-IDF
Term1		5	1	$\text{Log}3$	$5\text{log}3$
Term2		1	1	$\text{Log}3$	$1\text{log}3$
Term3		5	3	$\text{Log}1$	0
Term4		3	3	$\text{Log}1$	0
Term5		3	2	$\text{Log}(3/2)$	$3\text{log}(3/2)$

Term 1 > Term 5 > Term 2 > Term 3 = Term 4 */\* elice \*/*

## 02 Count-based Representations

### ✓ N-Gram

- 전에 등장한 N-1개의 단어를 사용하여 다음 단어를 예측

$$P(w_n | w_{n-1}, w_{n-2}, \dots, w_1) = \frac{P(w_n, w_{n-1}, w_{n-2}, \dots, w_1)}{P(w_{n-1}, w_{n-2}, \dots, w_1)}$$

EX) 나는 너를 \_\_\_\_\_ 1. 좋아해 / 2. 싫어해 / 3. 배고파 / 4. 졸려

		2 <sup>nd</sup> word							
1 <sup>st</sup> word		i	want	to	eat	chinese	food	lunch	spend
	i	5	827	0	9	0	0	0	2
	want	2	0	608	1	6	6	5	1
	to	2	0	4	686	2	0	6	211
	eat	0	0	2	0	16	2	42	0
	chinese	1	0	0	0	0	82	1	0
	food	15	0	15	0	1	4	0	0
	lunch	2	0	0	0	0	1	0	0
	spend	1	0	1	0	0	0	0	0

/\* elice \*/



# 실습 - 국민청원 Project

## 국민청원 추천 시스템 구현 및 효율화

TOBIGS  
김수지 서석현 이준걸  
임소정 정민호 황이은

03

# Document Similarity



## 03 Document Similarity

### ✓ NLP Process

**1. Data Collection**  
(Crawling)

**3. Embedding**  
(Word2Vec, GloVe, FastText, BERT...)

**5. Modeling**  
(RNN, LSTM...)

**2. Tokenizing**  
(Konlpy, Mecab, Khaii...)

**4. Similarity**  
(Euclidean, Cosine, Jaccard...)

*/\* elice \*/*

# 03 Document Similarity

## ✓ Document Similarity

- Document를 한 벡터공간으로 Embedding을 시켜봤음.

Binary representation

Word	S1	S2
John	1	1
Likes	1	1
To	1	1
Watch	1	1
Movies	1	0
Also	0	1
Football	0	1
Games	0	1
Mary	1	0
too	1	0

Frequency representation

Word	S1	S2
John	1	1
Likes	2	1
To	1	1
Watch	1	1
Movies	1	0
Also	0	1
Football	0	1
Games	0	1
Mary	1	0
too	1	0

Document Embedding Vector

Word Embedding Vector

TDM

	Antony and Cleopatra	Julius Caesar	The Tempest	Hamlet	Othello	Macbeth
Antony	5.25	3.18	0	0	0	0
Brutus	1.21	6.1	0	1	0	0
Caesar	8.59	2.54	0	1.51	0.25	0.35
Calpurnia	0	1.54	0	0	0	0
Cleopatra	2.85	0	0	0	0	0
mercy	1.51	0	1.9	0.12	5.25	0.88
worser	1.37	0	0.11	4.15	0.25	1.95

Document Embedding Vector

Word Embedding Vector

TF-IDF /\* elice \*/



## 03 Document Similarity

### ✓ Document Similarity

- 두 Document가 유사하다는 것의 정의가 필요하다.
  - d1: ant, ant, bee
  - d2: dog, bee, dog, hog, dog, ant, dog
  - d3: cat, gnu, dog, eel, fox
- 개인적으로 자연어처리에서 제일 중요하다고 생각 - Simple is Best
- 활용범위가 굉장히 다양함. Ex) 문서 찾기, 챗봇, text 생성 등 모두 가능

## 03 Document Similarity

### ✓ Document Similarity의 두가지 관점

- Term Similarity

- ◆ 두 Document는 같은 단어를 공유하는 것이 많을수록 유사할 것이다.
- ◆ 두 문서간 유사도의 식을 만들기위해 고려해야 하는 요소들
  1. 문서의 길이
  2. 두 문서에서 같이 등장하는 단어
  3. 두 문서 중 하나만 등장하거나 등장하지 않는 단어
  4. 각각의 단어가 얼마나 등장하는지

- Vector Space Model

- ◆ Embedding된 Document의 벡터를 활용하여 유사도를 계산 -> Correlation, Cos-similarity

/\* elice \*/

## 03 Document Similarity

### ✓ Document Similarity: Term Similarity

- Term Similarity

- ◆ 두 문서간 유사도의 식을 만들기위해 고려해야 하는 요소들

1.  $x_{ik}$ : Document i에서 Term k의 Frequency
2.  $a_{ij}$ : Document i와 Document j 에 동시에 등장하는 Term 개수
3.  $b_{ij}$ : Document i와 Document j 中 i에만 등장하는 Term 개수
4.  $c_{ij}$ : Document i와 Document j 中 j에만 등장하는 Term 개수
5.  $d_{ij}$ : Document i와 Document j 모두 등장하지 않는 Term 개수

## 03 Document Similarity

### ✓ Document Similarity: Term Similarity

- Term Similarity

- ◆ 다음과 같은 Term Similarity가 존재함.

1. Common Features Model : 같이 등장하는 단어( $a_{ij}$ )가 중요하다라는 관점
2. Ratio Model : 같이 등장하는 단어와 하나에서만 존재하는 단어의 비
3. Simple Matching Coefficient : 한 Document가 다른 것에 비해 이질적인 단어를 많이 갖고 있으면 덜 유사하다라는 관점
4. Jaccard Similarity : 전체 중 같이 등장하는 단어의 개수의 Ratio
5. Overlap Similarity : Jaccard Similarity의 단점을 보완한 유사도

## 03 Document Similarity

### ✓ Term Similarity: Example

## TDM

Freq.	D1	D2	D3
T1	3	0	2
T2	0	0	1
T3	5	3	0
T4	0	2	1
T5	0	1	2

Binary	D1	D2	D3
T1	1	0	1
T2	0	0	1
T3	1	1	0
T4	0	1	1
T5	0	1	1

$a_{12}$ : Document 1와 Document 2 에 동시에 등장하는 Term 개수 = 1

$b_{12}$ : Document 1와 Document 2 中 1에만 등장하는 Term 개수 = 1

$c_{12}$ : Document 1와 Document 2 中 2에만 등장하는 Term 개수 = 2

$d_{12}$ : Document 1와 Document 2 모두 등장하지 않는 Term 개수 = 1

/\* elice \*/

## 03 Document Similarity

### ✓ Common Features Model - 같이 등장하는 단어가 중요하다라는 관점

Freq.	D1	D2	D3
T1	3	0	2
T2	0	0	1
T3	5	3	0
T4	0	2	1
T5	0	1	2

Binary	D1	D2	D3
T1	1	0	1
T2	0	0	1
T3	1	1	0
T4	0	1	1
T5	0	1	1

$$\blacklozenge S_{ij}^{common} = \frac{a_{ij}}{a_{ij}+b_{ij}+c_{ij}+d_{ij}}$$

$$ex) S_{12}^{common} = \frac{a_{12}}{a_{12}+b_{12}+c_{12}+d_{12}} = \frac{1}{1+1+2+1}$$



Common	D1	D2	D3
D1	-	1/5	1/5
D2		-	2/5
D3			-

## 03 Document Similarity

### ✔ Ratio Model - 같이 등장하는 단어와 하나에서만 존재하는 단어의 비

Freq.	D1	D2	D3
T1	3	0	2
T2	0	0	1
T3	5	3	0
T4	0	2	1
T5	0	1	2

Binary	D1	D2	D3
T1	1	0	1
T2	0	0	1
T3	1	1	0
T4	0	1	1
T5	0	1	1

$$\blacklozenge S_{ij}^{ratio} = \frac{a_{ij}}{a_{ij}+b_{ij}+c_{ij}}$$

$$ex) S_{12}^{ratio} = \frac{a_{12}}{a_{12}+b_{12}+c_{12}} = \frac{1}{1+1+2}$$



Ratio	D1	D2	D3
D1	-	1/4	1/5
D2		-	2/5
D3			-

## 03 Document Similarity

- ✓ Simple Matching Coefficient - 한 Document가 다른 것에 비해 이질적인 단어를 많이 갖고 있으면 덜 유사하다는 관점

Freq.	D1	D2	D3
T1	3	0	2
T2	0	0	1
T3	5	3	0
T4	0	2	1
T5	0	1	2

Binary	D1	D2	D3
T1	1	0	1
T2	0	0	1
T3	1	1	0
T4	0	1	1
T5	0	1	1

$$\blacklozenge S_{ij}^{smc} = \frac{a_{ij} + d_{ij}}{a_{ij} + b_{ij} + c_{ij} + d_{ij}}$$

$$ex) S_{12}^{smc} = \frac{a_{12} + d_{12}}{a_{12} + b_{12} + c_{12} + d_{12}} = \frac{1 + 1}{1 + 1 + 2 + 1}$$



SMC	D1	D2	D3
D1	-	2/5	1/5
D2		-	2/5
D3			-



## 03 Document Similarity

### ✓ Jaccard Similarity - 전체 중 같이 등장하는 단어의 개수의 Ratio

Freq.	D1	D2	D3
T1	3	0	2
T2	0	0	1
T3	5	3	0
T4	0	2	1
T5	0	1	2

Binary	D1	D2	D3
T1	1	0	1
T2	0	0	1
T3	1	1	0
T4	0	1	1
T5	0	1	1

$$\blacklozenge S_{ij}^{jaccard} = \frac{\sum_k \min(x_{ik}, x_{jk})}{\sum_k \max(x_{ik}, x_{jk})}$$

$$ex) S_{12}^{jaccard} = \frac{0+0+3+0+0}{3+0+5+2+1} = \frac{3}{11}$$



Jaccard	D1	D2	D3
D1	-	3/11	2/12
D2		-	2/9
D3			-

/\* elice \*/

## 03 Document Similarity

### ✓ Overlap Similarity - Jaccard Similarity의 단점을 보완한 유사도

Freq.	D1	D2	D3
T1	3	0	2
T2	0	0	1
T3	5	3	0
T4	0	2	1
T5	0	1	2

Binary	D1	D2	D3
T1	1	0	1
T2	0	0	1
T3	1	1	0
T4	0	1	1
T5	0	1	1

$$\blacklozenge S_{ij}^{overlap} = \frac{\sum_k \min(x_{ik}, x_{jk})}{\min(\sum_k x_{ik}, \sum_k x_{jk})}$$

$$ex) S_{12}^{overlap} = \frac{0+0+3+0+0}{\min(8, 6)} = \frac{3}{6}$$



Overlap	D1	D2	D3
D1	-	3/6	2/6
D2		-	2/6
D3			-

/\* elice \*/

## 03 Document Similarity

### ✔ Document Similarity: Vector Space Model - Cos Similarity

- Embedding된 Document의 벡터를 활용하여 유사도를 계산

Freq.	D1	D2	D3
T1	3	0	2
T2	0	0	1
T3	5	3	0
T4	0	2	1
T5	0	1	2

Binary	D1	D2	D3
T1	1	0	1
T2	0	0	1
T3	1	1	0
T4	0	1	1
T5	0	1	1

$$\text{Cos similarity}(A,B) = \frac{A \cdot B}{\|A\| \times \|B\|} = \frac{\sum_{i=1}^n A_i \times B_i}{\sqrt{\sum_{i=1}^n A_i^2} \times \sqrt{\sum_{i=1}^n B_i^2}} = \frac{15}{\sqrt{9 + 25} * \sqrt{9 + 4 + 1}}$$

D1, D2

/\* elice \*/



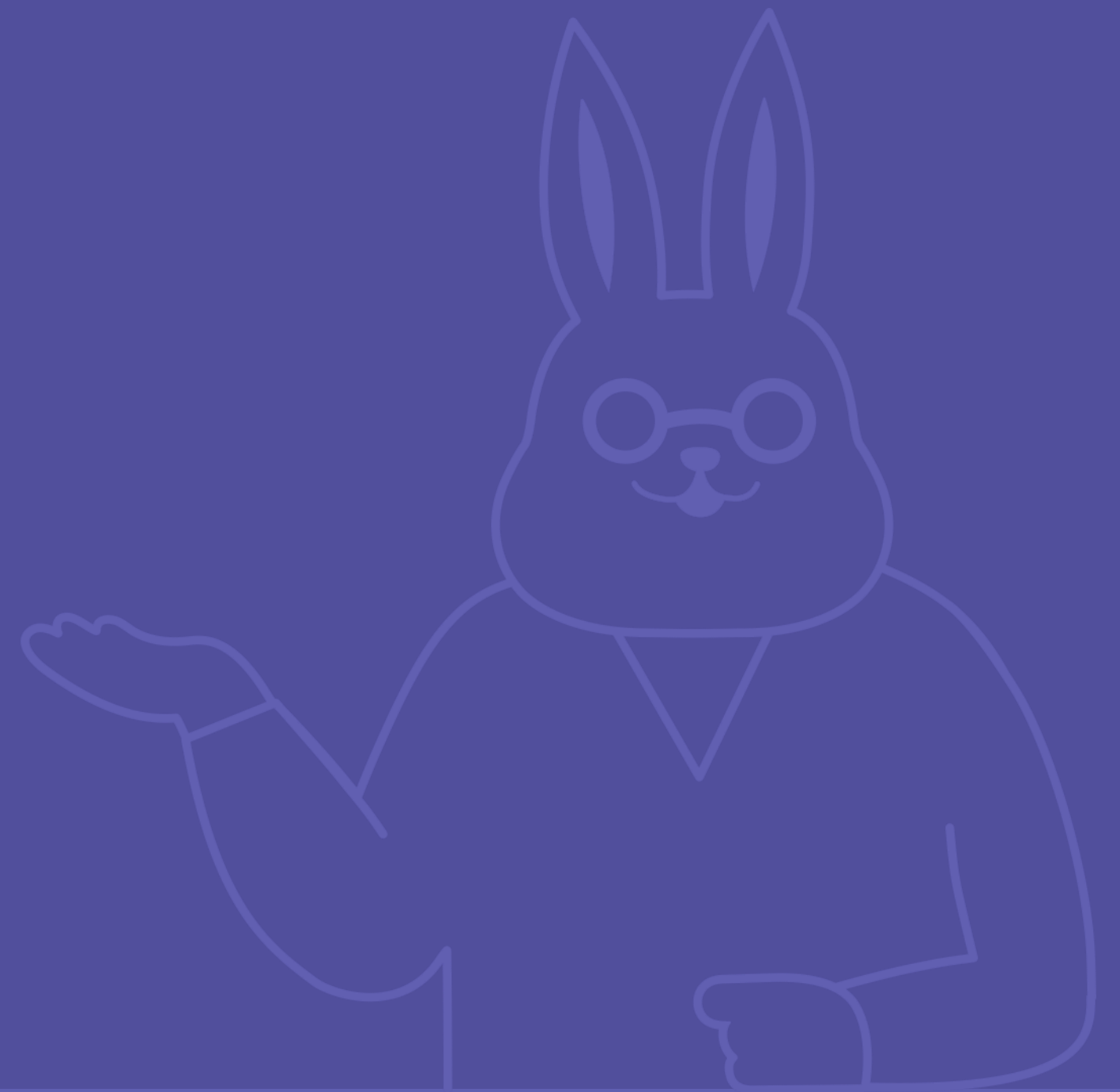
# 실습 - 국민청원 Project

## 국민청원 추천 시스템 구현 및 효율화

TOBIGS  
김수지 서석현 이준걸  
임소정 정민호 황이은

04

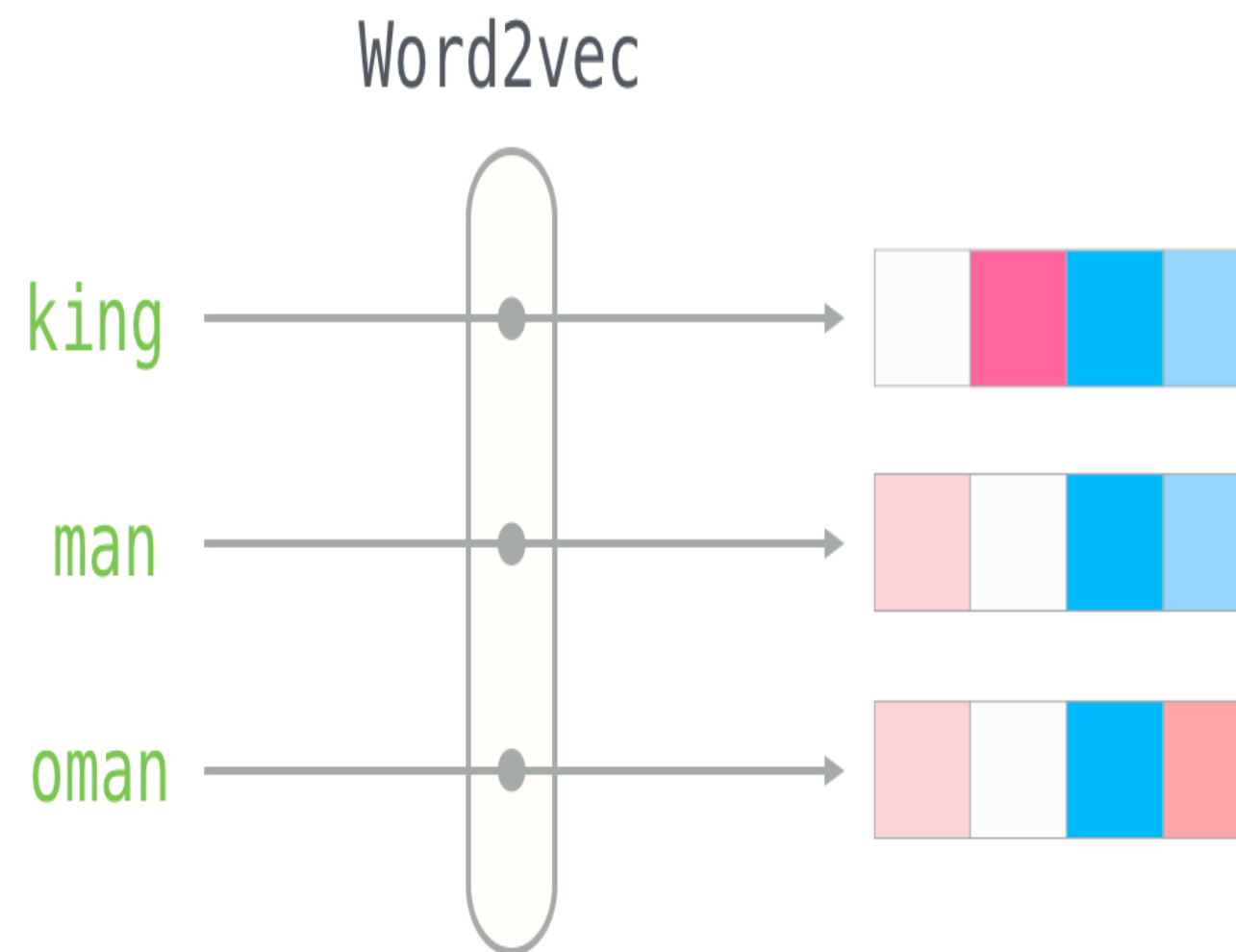
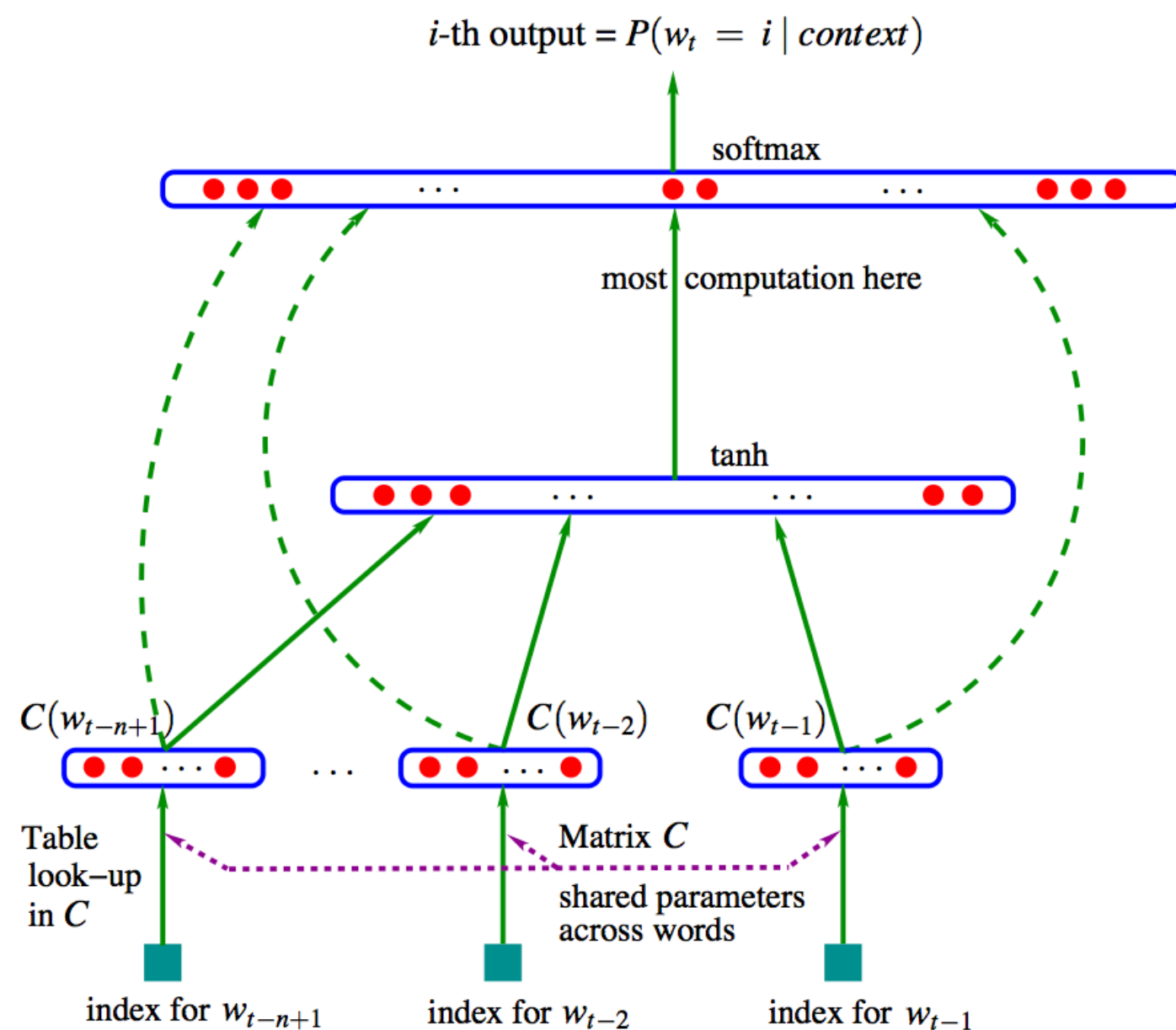
# DL-based Representations



## 04 DL-based Representations

### ✓ Word Embedding

- 워드 임베딩은 단어들을 의미상으로 유사한 단어가 벡터공간에 가까이 있도록 Mapping 시키는 작업을 뜻한다.

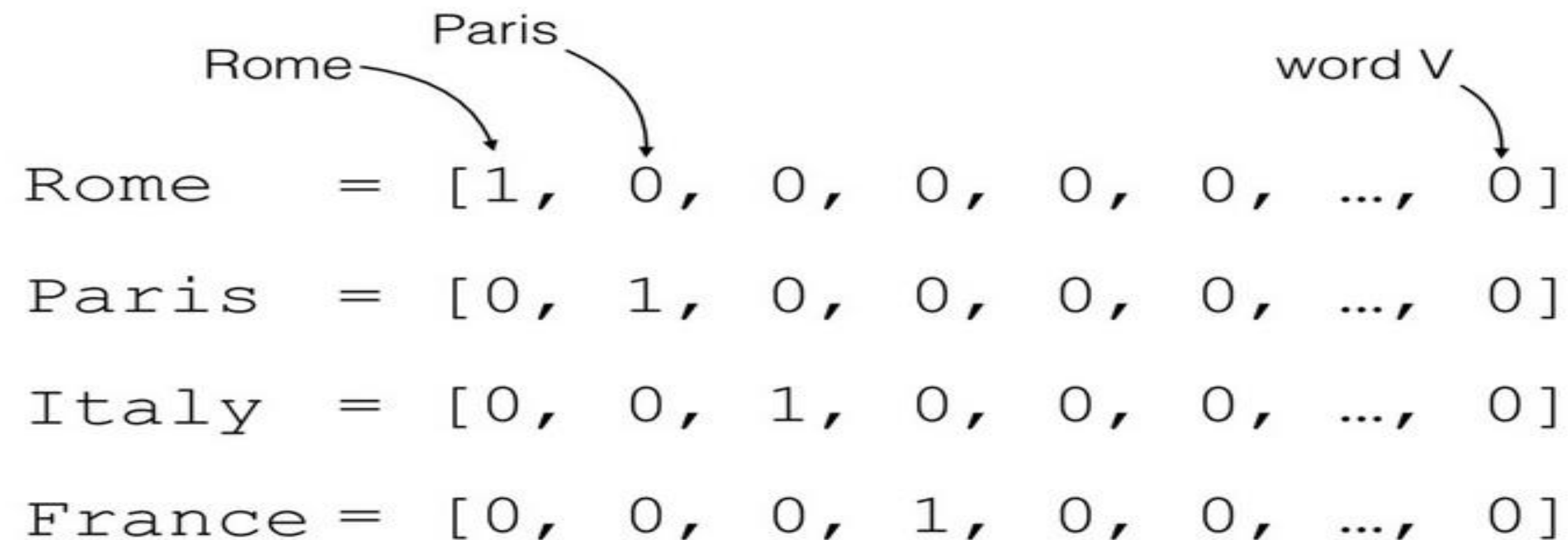


/\* elice \*/

## 04 DL-based Representations

### ✓ Word Embedding : one-hot vector

- 가장 간단하고 쉬운 Embedding 방법



- Vector로 표현은 되나 두 단어간의 유사성이 보존되지 않는다.

$$(w^{Rome})^T w^{Italy} = 0$$

$$(w^{Paris})^T w^{France} = 0$$

/\* elice \*/

## 04 DL-based Representations

### ✓ Word Embedding : Distributed Representation

- 특정 함수를 통해 우리가 원하는 차원으로 단어의 벡터를 Embedding 시키는 방법.
- 또한 이 함수에 의해 기존의 one-hot과 달리 벡터가 갖는 value가 Dense해짐.

$$W : \text{words} \rightarrow \mathbb{R}^n$$

$$w^{Rome} = (0.2, -0.4, 0.7, \dots)$$

$$w^{France} = (0.0, 0.6, -0.1, \dots)$$

- 이러한 특정 함수를 Neural Network를 통해 만들어보자.



## 04 DL-based Representations

### ✓ Word Embedding : Distributed Representation

- Only Embedding task Neural Network
  - NNLM, Word2Vec, Glove, Fasttext



Feature  
generating  
(Embedding)

- Multi-task Neural Network
  - Sentimental Analysis Model, BERT



Feature  
generating  
(Embedding)

+

Other Task

## 04 DL-based Representations

### ✓ ① Neural Network Language Model(NNLM)

- 딥러닝을 사용하여 단어 사전에 있는 각 단어를 Distributed representation으로 Embedding시킨 최초 모델
- Language-Model의 아이디어를 차용하여 목적함수에 이용
- 결과물로 Vocab size X embedding size Matrix 즉, Embedding Matrix를 얻을 수 있다.

## 04 DL-based Representations

### ✓ Count Based Language Model

$$p(w_1, \dots, w_T) = \prod_{t=1}^T p(w_t | w_1, \dots, w_{t-1})$$

Markov Assumption

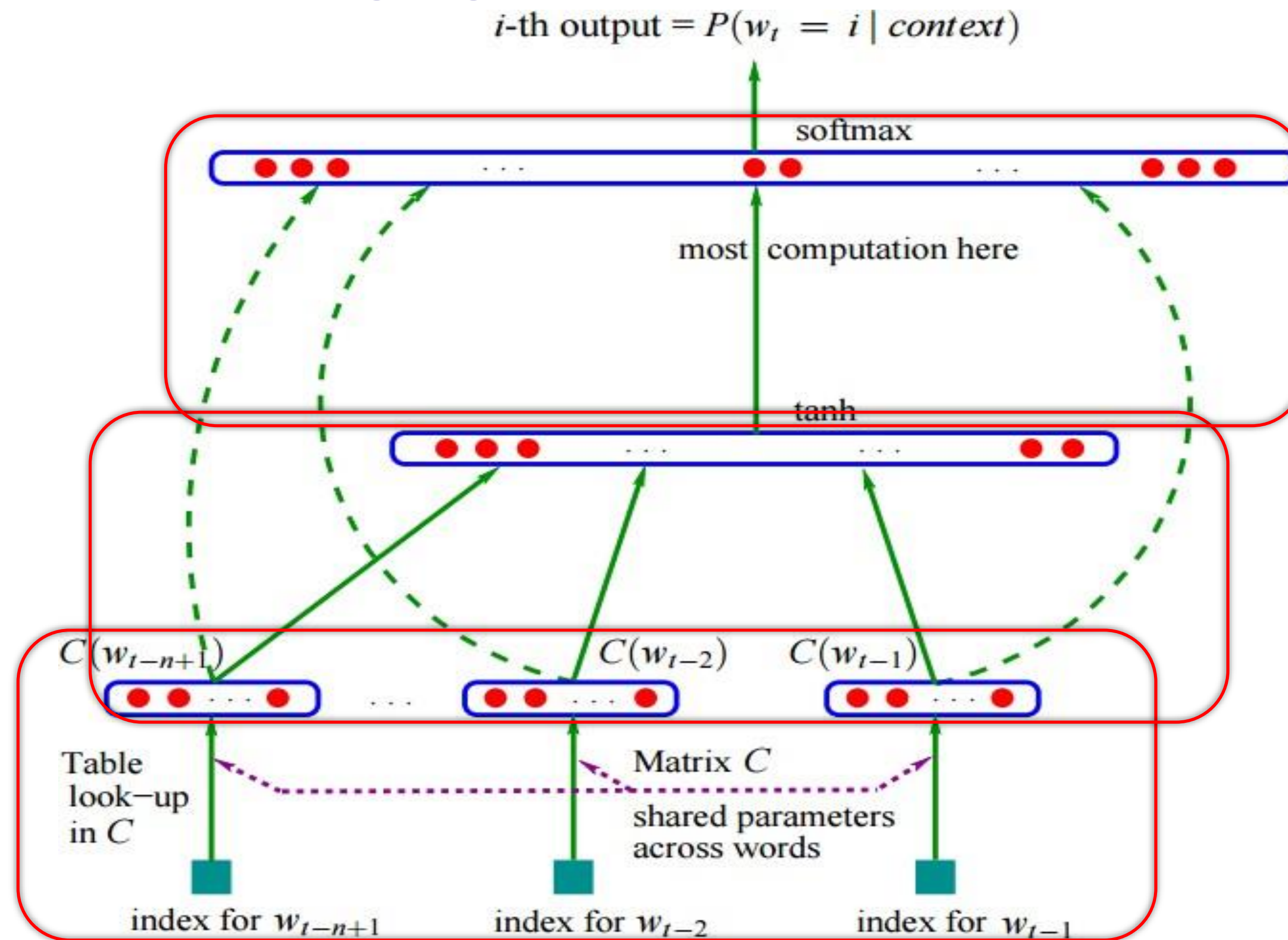


$$p(w_t | w_1, \dots, w_t) \approx p(w_t | w_{t-n}, \dots, w_{t-1})$$

(n-gram Language Model)

## 04 DL-based Representations

### ✓ ① Neural Network Language Model(NNLM)



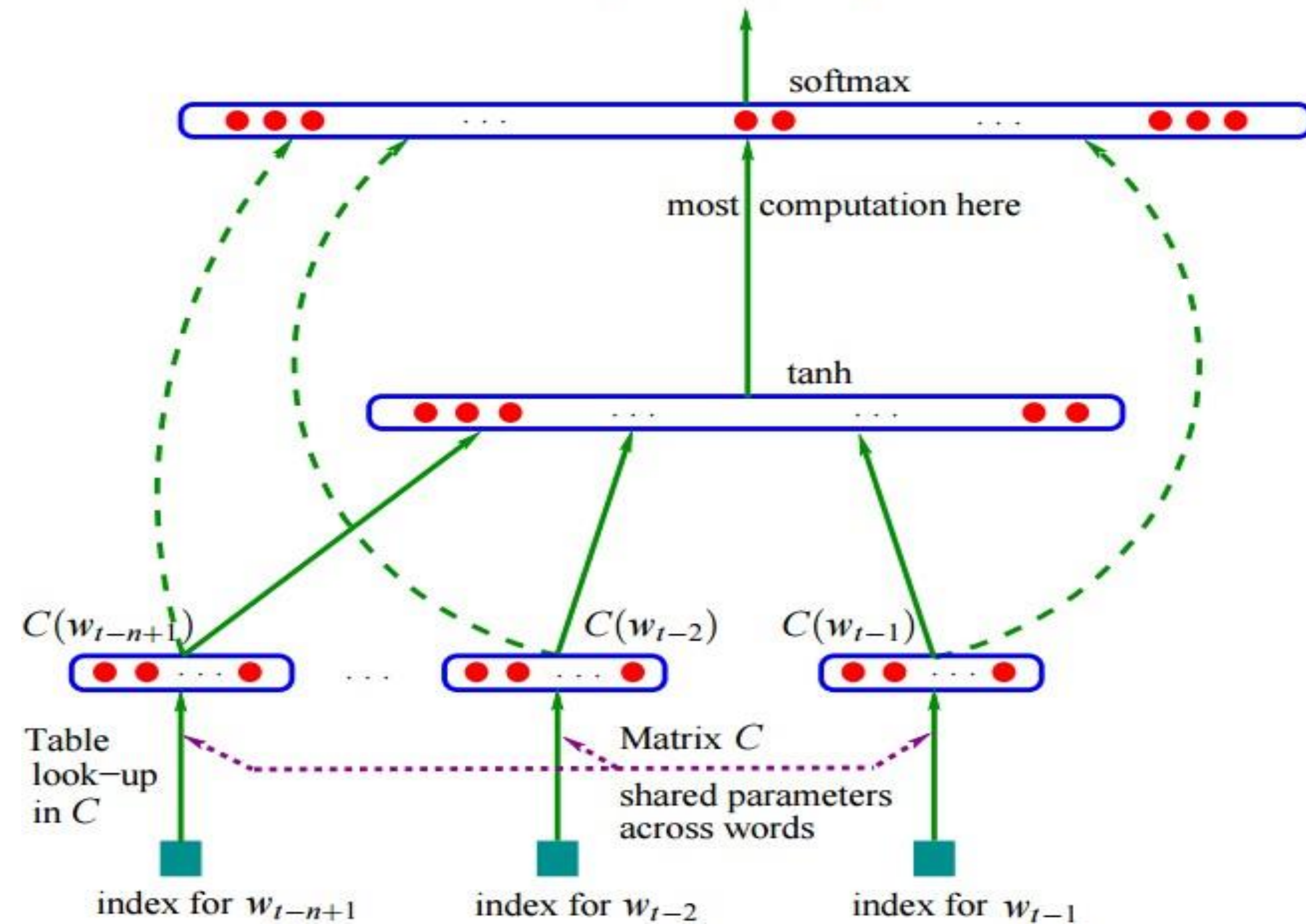
`/* elice */`



## 04 DL-based Representations

### ✓ ① Neural Network Language Model(NNLM)

$i$ -th output =  $P(w_t = i \mid \text{context})$



$$y = b + Wx + U \tanh(d + Hx)$$

$$U \tanh(d + Hx)$$

$$d + Hx$$

$$x = \text{flatten}(WC)$$

$$WC$$

`/* elice */`

# 04 DL-based Representations

## ① Neural Network Language Model(NNLM)

$i$ -th output =  $P(w_t = i \mid context)$

$$\frac{\exp(y_{w_t})}{\sum_i \exp(y_i)}$$

$$y = b + Wx + U \tanh(d + Hx)$$

$$U \tanh(d+Hx)$$

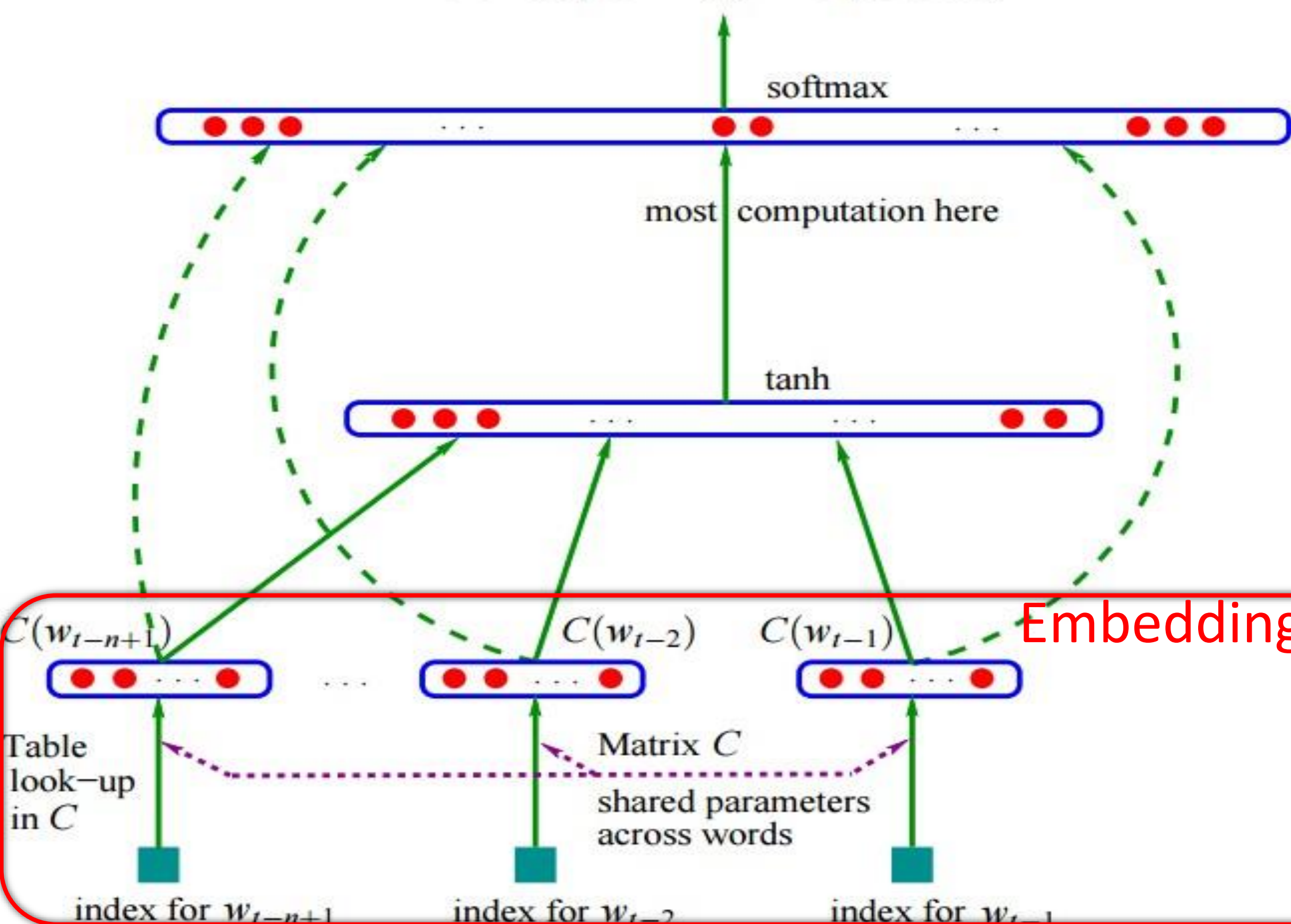
$$d+Hx$$

Embedding Layer

$$x = \text{flatten}(WC)$$

$$WC$$

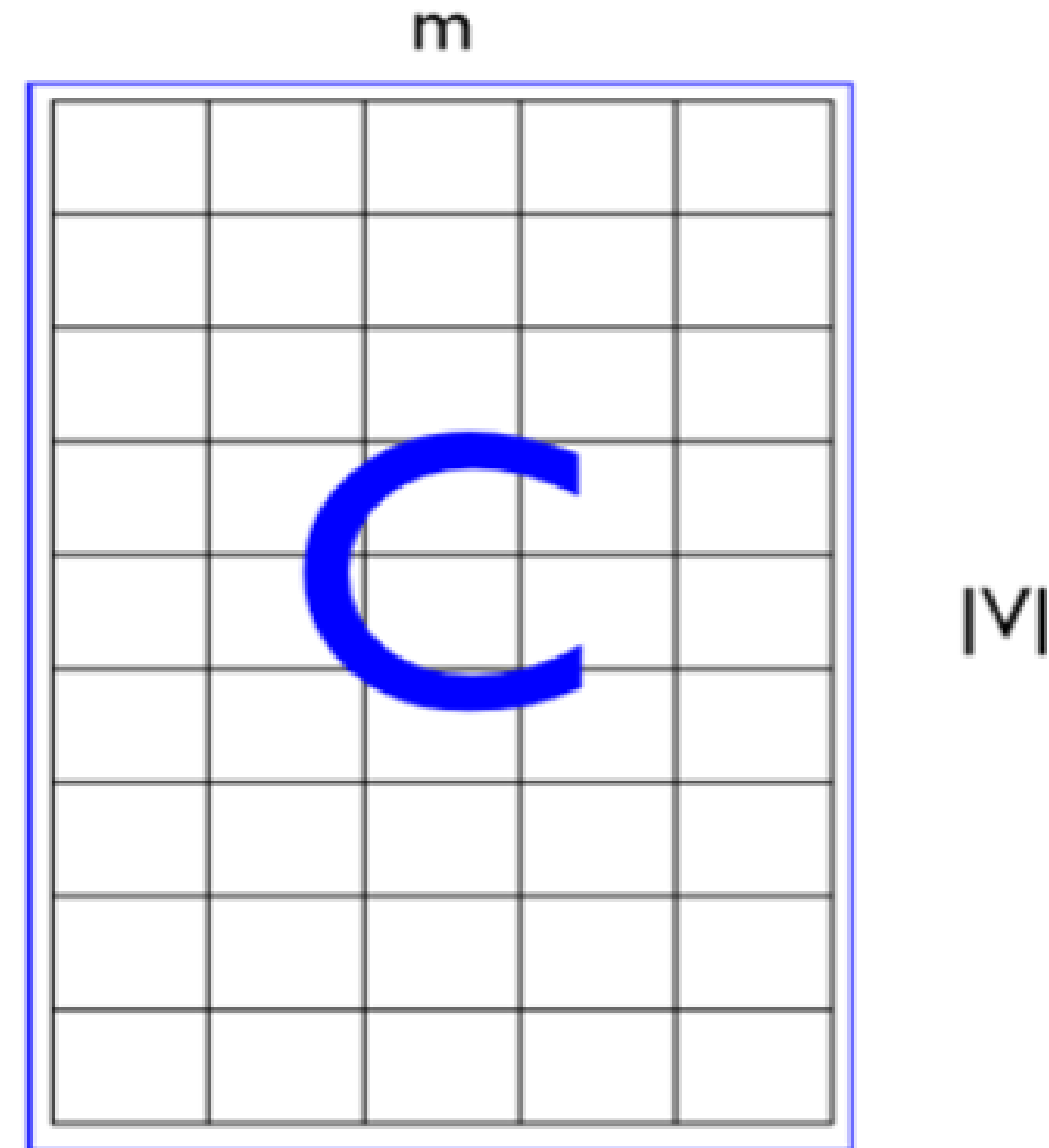
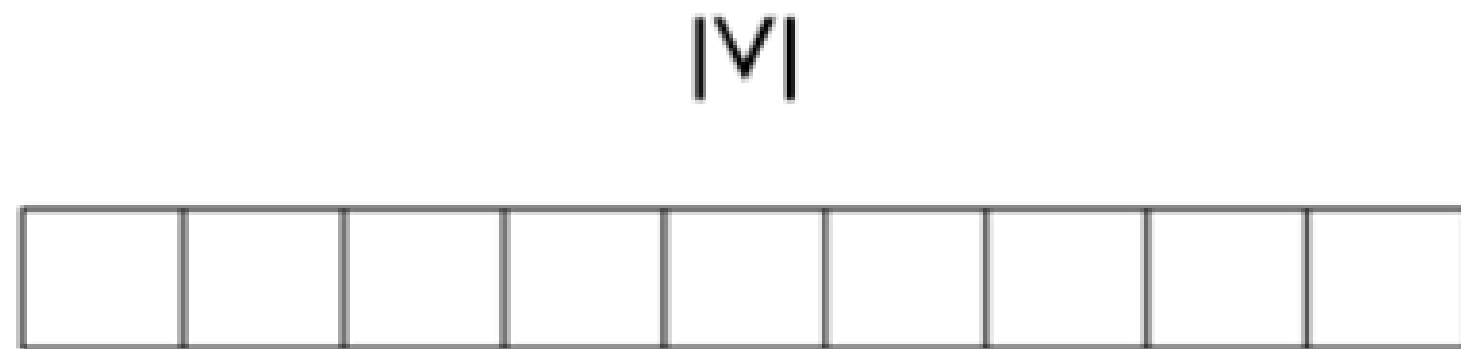
/\* elice \*/



## 04 DL-based Representations

### ✓ Embedding Layer

$WC$



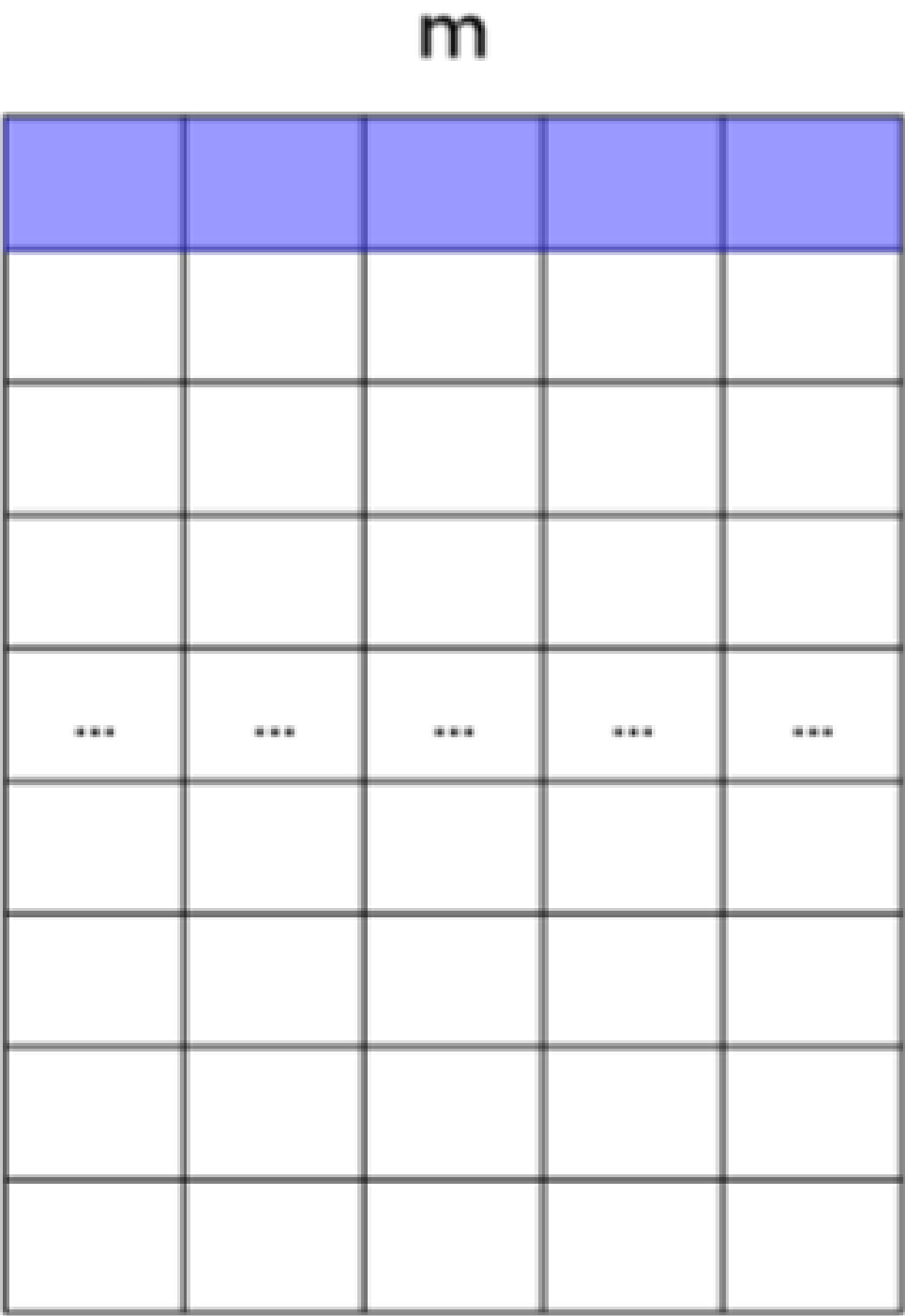
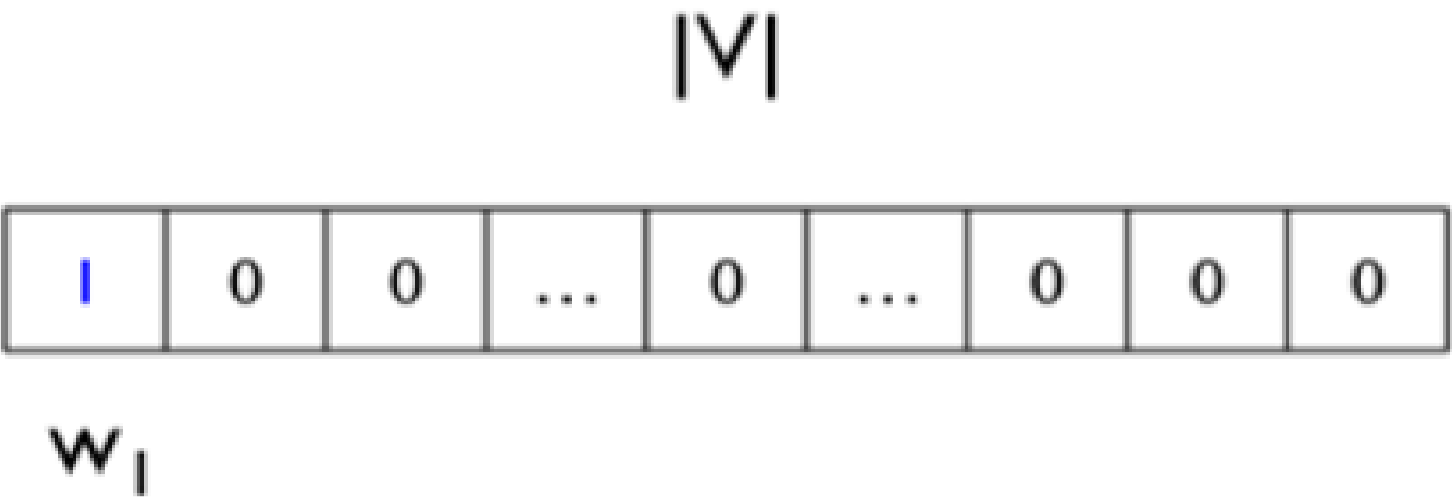
`/* elice */`



# 04 DL-based Representations

## ✓ Embedding Layer

$WC$



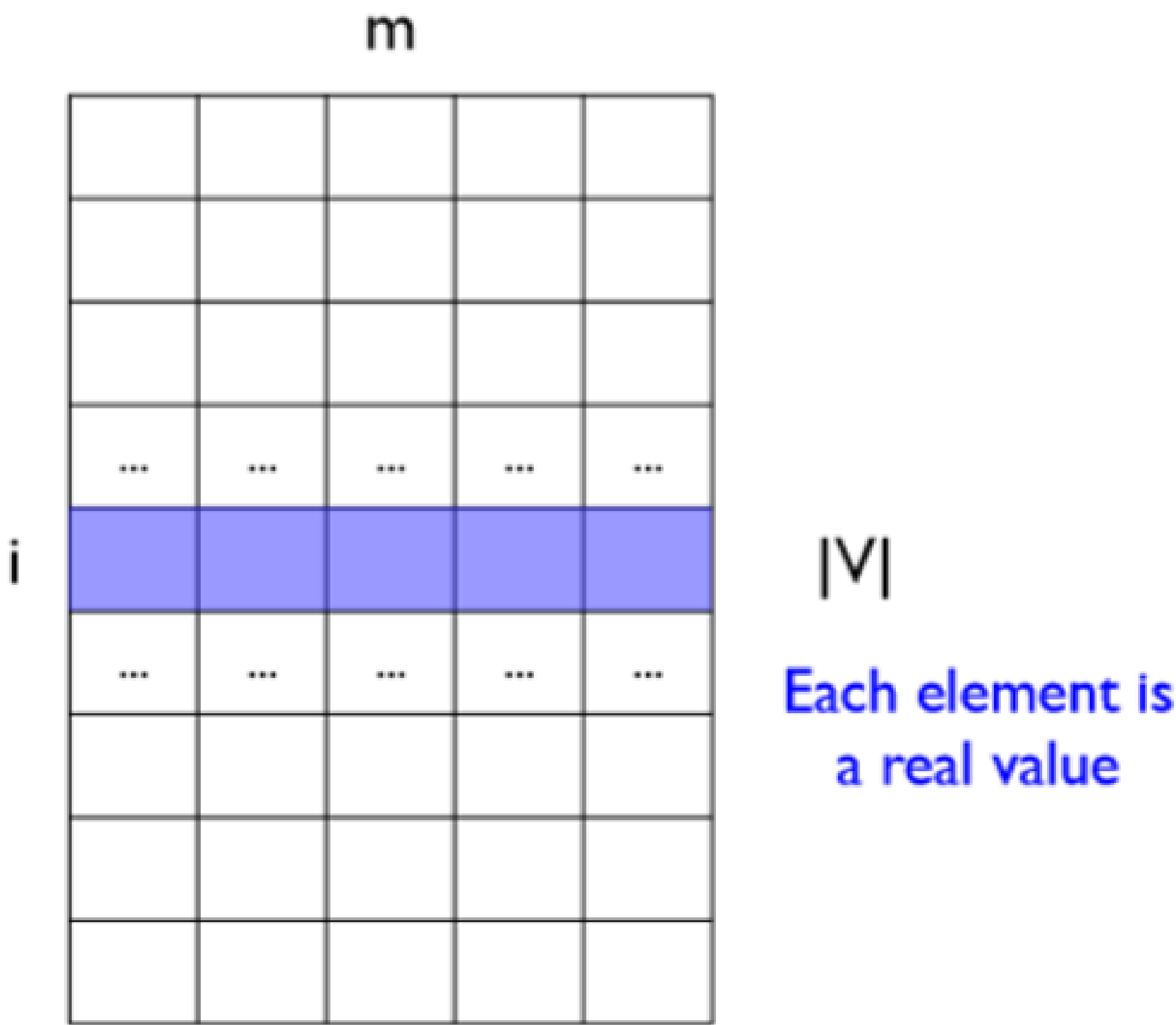
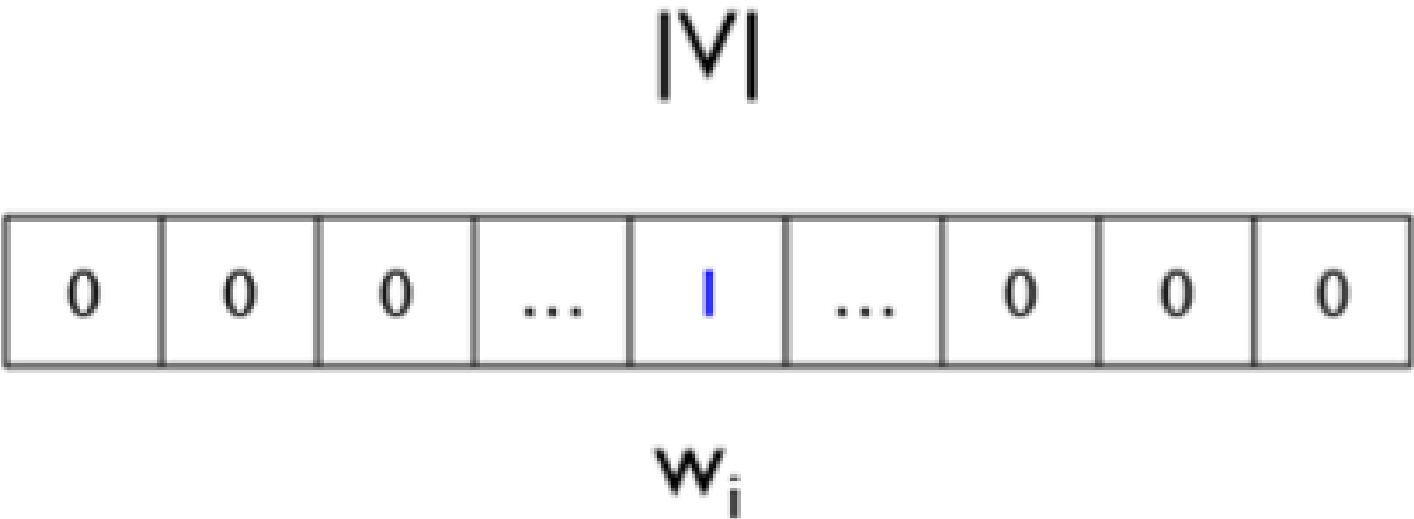
Each element is  
a real value

$|V|$

# 04 DL-based Representations

## ✓ Embedding Layer

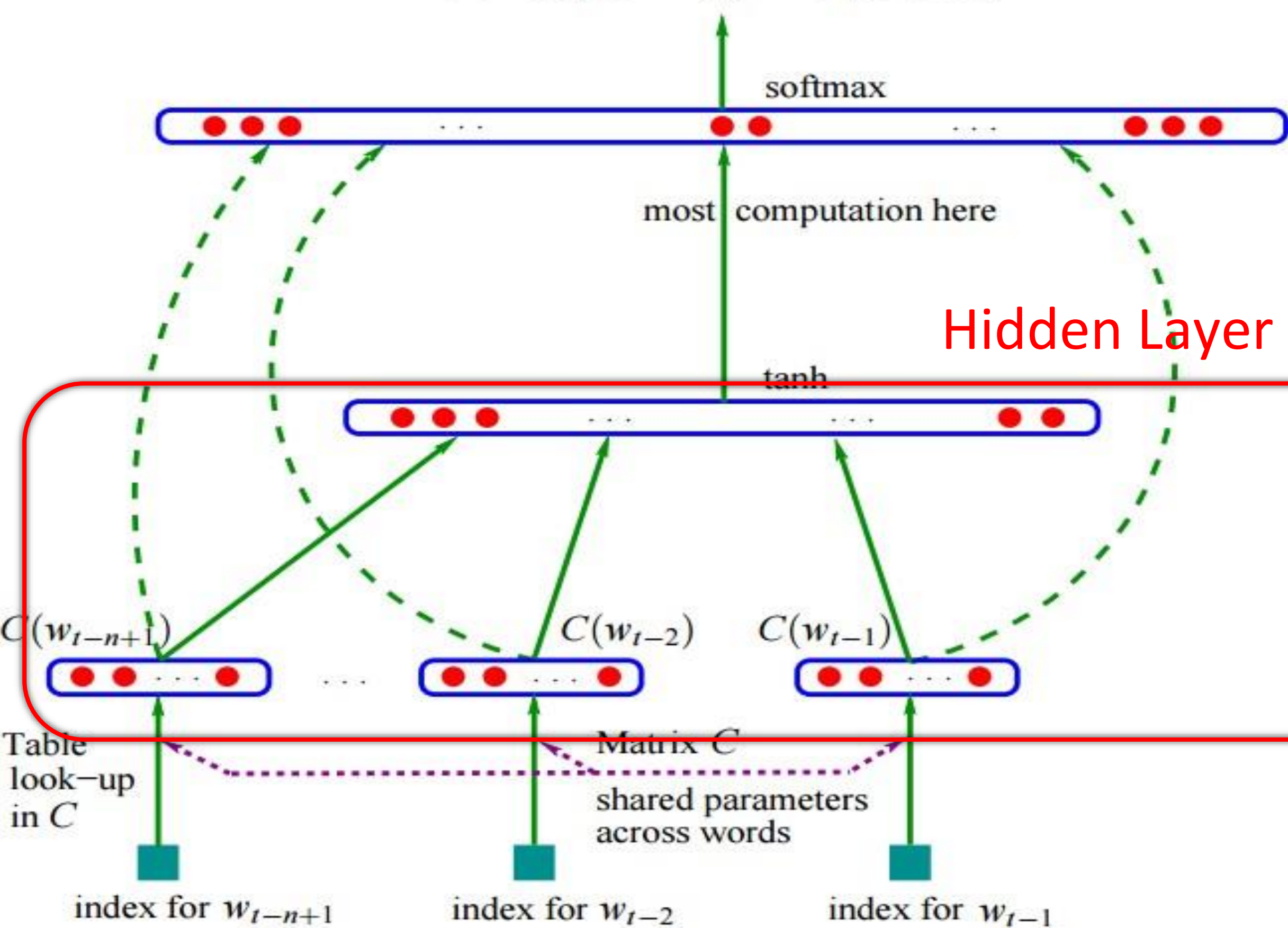
$WC$



# 04 DL-based Representations

## ① Neural Network Language Model(NNLM)

$i$ -th output =  $P(w_t = i \mid context)$



d : the Hidden layer biased d  
H : the Hidden layer weight

$\tanh(d+Hx)$

$d+Hx$

$x = \text{flatten}(WC)$

$WC$

`/* elice */`

## 04 DL-based Representations

### ① Neural Network Language Model(NNLM)

$i$ -th output =  $P(w_t = i \mid \text{context})$

$$\frac{\exp(y_{w_t})}{\sum_i \exp(y_i)}$$

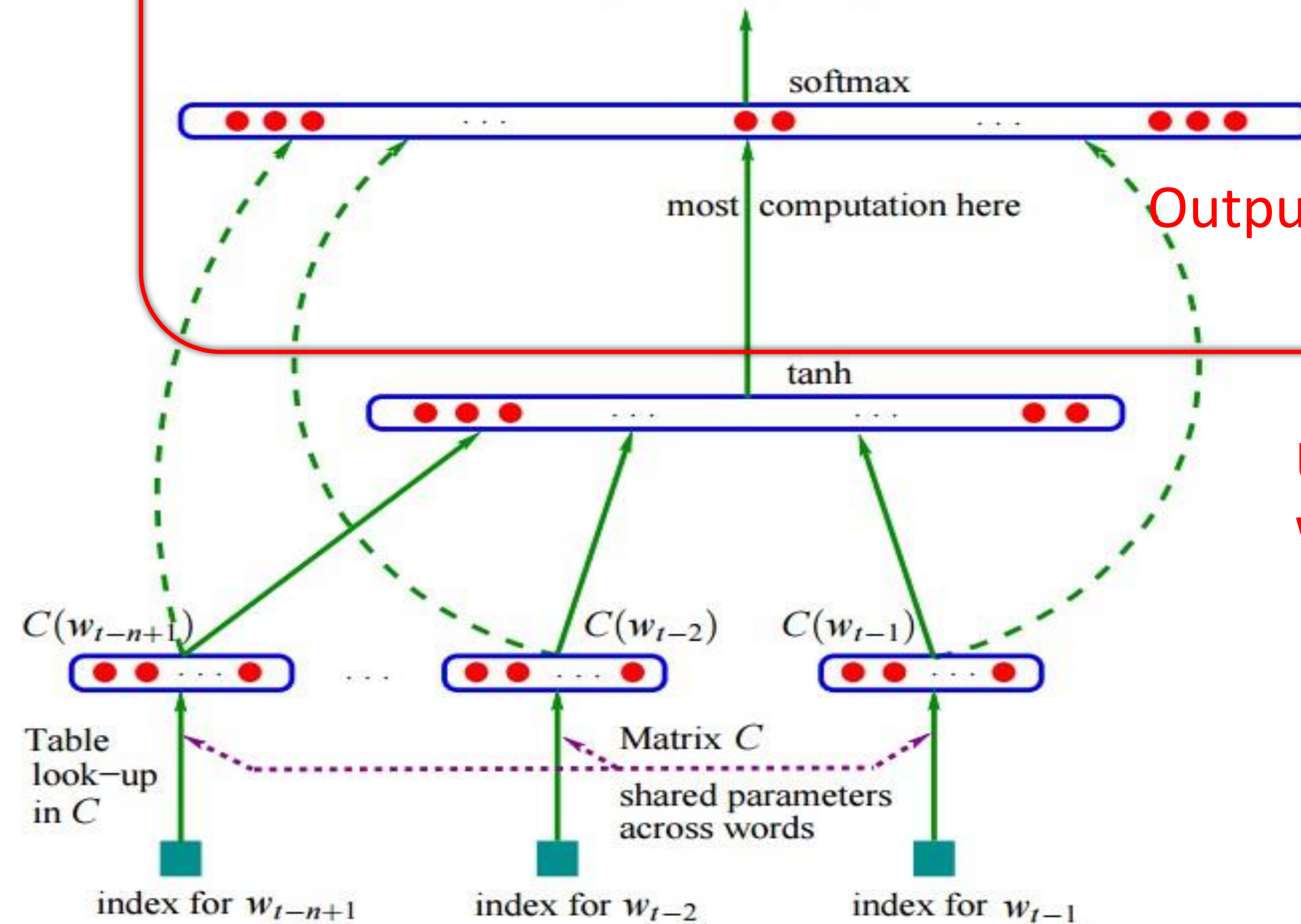
$$y = b + Wx + U \tanh(d + Hx)$$

Output Layer

$$U \tanh(d + Hx)$$

$U$  : the hidden-to-output weight  
 $W$  : input layer to output weight  
 $b$  : the output bias

/\* elice \*/



## 04 DL-based Representations

### ✓ NNLM

The neural network has one hidden layer beyond the word features mapping, and optionally, direct connections from the word feature to the output.

$$\hat{P}(w_t | w_{t-1}, \dots, w_{t-n+1}) = \frac{\exp(y_{w_t})}{\sum_i \exp(y_i)} \quad \longleftrightarrow \quad p(w_t | w_1, \dots, w_t) \approx p(w_t | w_{t-n}, \dots, w_{t-1})$$

$$y = b + Wx + U \cdot \tanh(d + Hx)$$

Neural Network (n-gram) Language Model

n-gram Language Model



## 04 DL-based Representations

### ✓ Word2Vec

- CBOW & Skip-gram VS N-gram

내가 어떻게 해야 그대를 잊을 수 있을까

Tokenizing

‘내’, ‘가’, ‘어떻게’, ‘해야’, ‘그대’, ‘를’,  
‘잊을’, ‘수’, ‘있을’, ‘까’

Window Size를 정하자

`/* elice */`

# 04 DL-based Representations

## ✔ Word2Vec

- CBOW & Skip-gram VS N-gram

‘내’, ‘가’, ‘어떻게’, ‘해야’, ‘그대’, ‘를’,  
‘있을’, ‘수’ ‘있을’, ‘까’

Center Word	Neighbor Words
‘어떻게’	‘내’, ‘가’
‘해야’	‘가’, ‘어떻게’
‘그대’	‘해야’, ‘해야’
‘를’	‘어떻게’, ‘그대’, ‘를’
‘있을’	를’, ‘그대’
‘수’	‘있을’, ‘를’
‘있을’	‘수’, ‘있을’
‘까’	‘있을’, ‘수’



# 04 DL-based Representations

## ✓ Word2Vec

- CBOW & Skip-gram VS N-gram

‘내’, ‘가’, ‘어떻게’, ‘해야’, ‘그대’, ‘를’,  
‘있을’, ‘수’ ‘있을’, ‘까 ’

Center Word	Neighbor Words
‘내’	‘가’, ‘어떻게’
‘가’	‘내’, ‘어떻게’, ‘해야’
‘어떻게’	‘내’, ‘가’, ‘해야’, ‘그대’
‘해야’	‘가’, ‘어떻게’, ‘그대’, ‘를’
‘그대’	‘어떻게’, ‘해야’, ‘를’, ‘있을’
‘를’	‘해야’, ‘그대’, ‘있을’, ‘수’
‘있을’	‘그대’, ‘를’, ‘수’, ‘있을’
‘수’	‘를’, ‘있을’, ‘있을’, ‘까’
‘있을 ’	‘있을’, ‘수’, ‘까’
‘까’	‘수’, ‘있을’

# 04 DL-based Representations

## ✓ Word2Vec

- CBOW & Skip-gram VS N-gram

center word    context words

I like playing football with my friends

I like playing football with my friends

I like playing football with my friends

I like playing football with my friends

I like playing football with my friends

I like playing football with my friends

I like playing football with my friends

center word	context words
[1,0,0,0,0,0,0]	[0,1,0,0,0,0,0] [0,0,1,0,0,0,0]
[0,1,0,0,0,0,0]	[1,0,0,0,0,0,0] [0,0,1,0,0,0,0] [0,0,0,1,0,0,0]
[0,0,1,0,0,0,0]	[1,0,0,0,0,0,0] [0,1,0,0,0,0,0] [0,0,0,1,0,0,0] [0,0,0,0,1,0,0]
[0,0,0,1,0,0,0]	[0,1,0,0,0,0,0] [0,0,1,0,0,0,0] [0,0,0,0,1,0,0] [0,0,0,0,0,1,0]
[0,0,0,0,1,0,0]	[0,0,1,0,0,0,0] [0,0,0,1,0,0,0] [0,0,0,0,0,1,0] [0,0,0,0,0,0,1]
[0,0,0,0,0,1,0]	[1,0,0,1,0,0,0] [0,0,0,0,1,0,0] [0,0,0,0,0,0,1]
[0,0,0,0,0,0,1]	[0,0,0,0,1,0,0] [0,0,0,0,0,1,0]

/\* elice \*/

# 04 DL-based Representations

## Word2Vec

- CBOW & Skip-gram VS N-gram
- ✓ 주변 단어(Context Words)로 중심단어(Center word) 예측하도록 학습

center word   context words

I like playing football with my friends

I like playing football with my friends

I like playing football with my friends

I like playing football with my friends

I like playing football with my friends

I like playing football with my friends

I like playing football with my friends

OUTPUT

center word	context words
[1,0,0,0,0,0,0]	[0,1,0,0,0,0,0] [0,0,1,0,0,0,0]
[0,1,0,0,0,0,0]	[1,0,0,0,0,0,0] [0,0,1,0,0,0,0] [0,0,0,1,0,0,0]
[0,0,1,0,0,0,0]	[1,0,0,0,0,0,0] [0,1,0,0,0,0,0] [0,0,0,1,0,0,0] [0,0,0,0,1,0,0]
[0,0,0,1,0,0,0]	[0,1,0,0,0,0,0] [0,0,1,0,0,0,0] [0,0,0,0,1,0,0] [0,0,0,0,0,1,0]
[0,0,0,0,1,0,0]	[0,0,1,0,0,0,0] [0,0,0,1,0,0,0] [0,0,0,0,0,1,0] [0,0,0,0,0,0,1]
[0,0,0,0,0,1,0]	[1,0,0,1,0,0,0] [0,0,0,0,1,0,0] [0,0,0,0,0,0,1]
[0,0,0,0,0,0,1]	[0,0,0,0,1,0,0] [0,0,0,0,0,1,0]

INPUT

/\* elice \*/

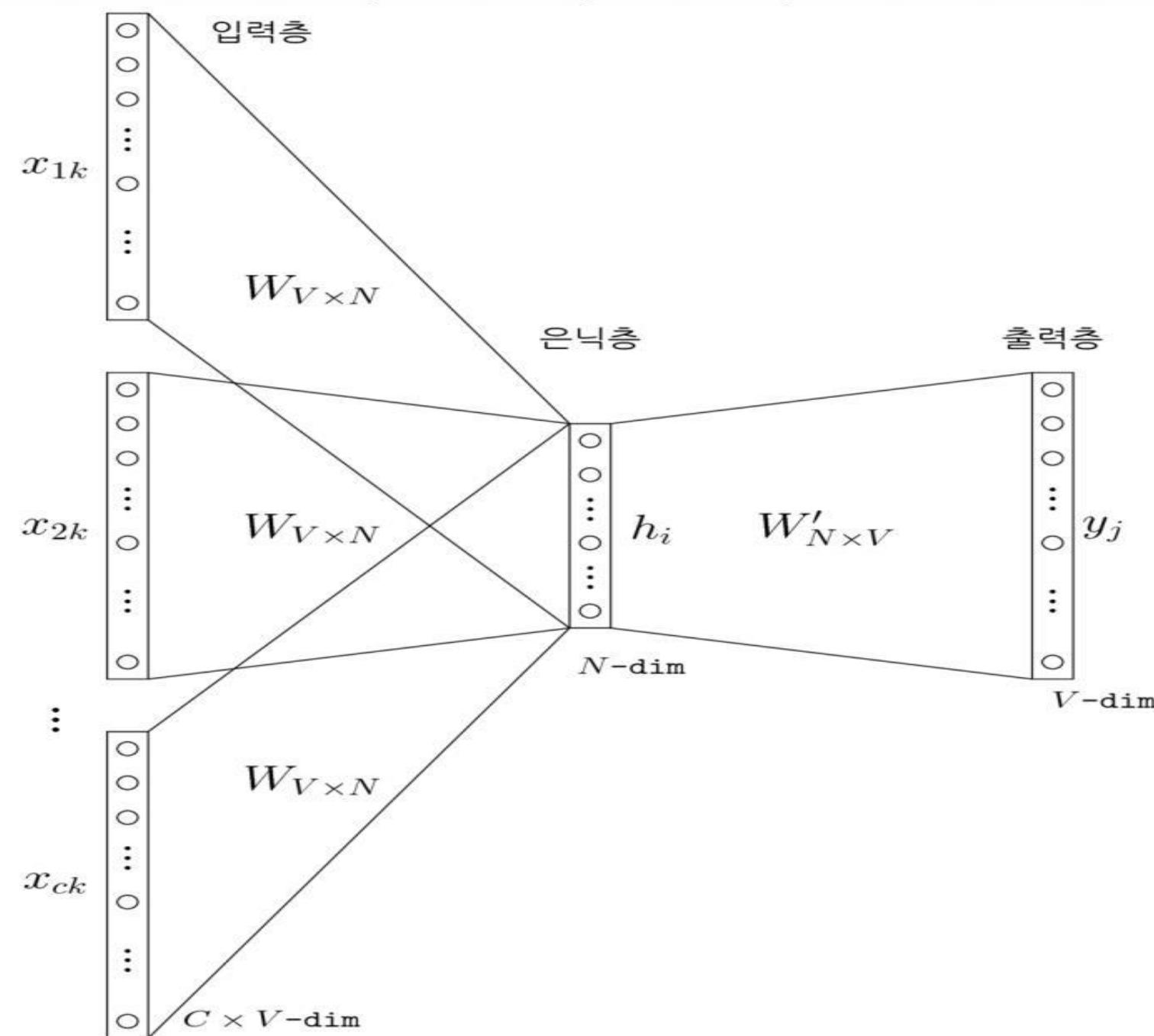
## 04 DL-based Representations

### ✓ Word2Vec

- CBOW & Skip-gram VS N-gram

✓ 주변 단어(Context Words)로 중심단어(Center word) 예측하도록 학습

INPUT : Context



OUTPUT : Center

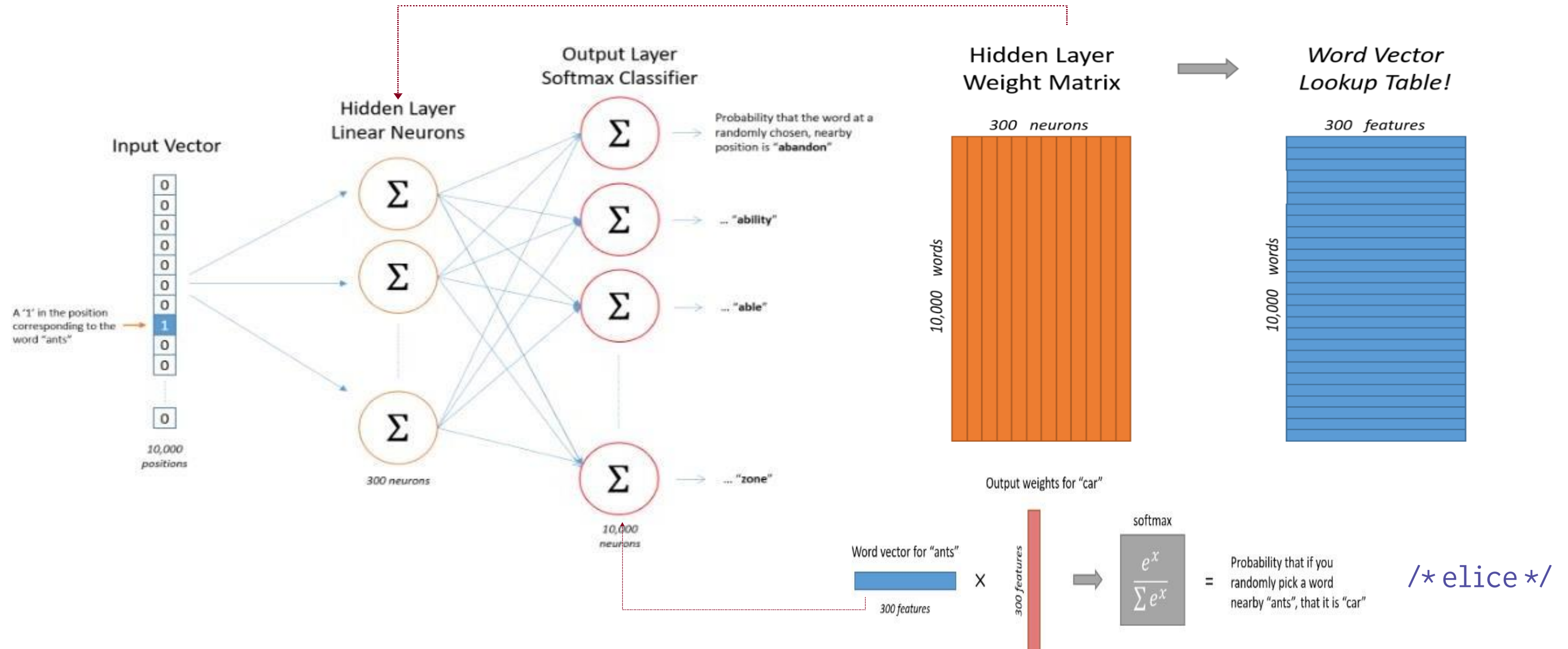
/\* elice \*/



# 04 DL-based Representations

## ✓ Word2Vec

- CBOW & Skip-gram VS N-gram



# 04 DL-based Representations

## Word2Vec

- CBOW & Skip-gram VS N-gram
- ✓ 중심단어(Center word)로 주변 단어(Context Words) 예측하도록 학습

center word   context words

I like playing football with my friends

I like playing football with my friends

I like playing football with my friends

I like playing football with my friends

I like playing football with my friends

I like playing football with my friends

I like playing football with my friends

INPUT

center word	context words
[1,0,0,0,0,0,0]	[0,1,0,0,0,0,0] [0,0,1,0,0,0,0]
[0,1,0,0,0,0,0]	[1,0,0,0,0,0,0] [0,0,1,0,0,0,0] [0,0,0,1,0,0,0]
[0,0,1,0,0,0,0]	[1,0,0,0,0,0,0] [0,1,0,0,0,0,0] [0,0,0,1,0,0,0] [0,0,0,0,1,0,0]
[0,0,0,1,0,0,0]	[0,1,0,0,0,0,0] [0,0,1,0,0,0,0] [0,0,0,0,1,0,0] [0,0,0,0,0,1,0]
[0,0,0,0,1,0,0]	[0,0,1,0,0,0,0] [0,0,0,1,0,0,0] [0,0,0,0,0,1,0] [0,0,0,0,0,0,1]
[0,0,0,0,0,1,0]	[1,0,0,1,0,0,0] [0,0,0,0,1,0,0] [0,0,0,0,0,0,1]
[0,0,0,0,0,0,1]	[0,0,0,0,1,0,0] [0,0,0,0,0,1,0]

OUTPUT

/\* elice \*/

## 04 DL-based Representations

### ✓ Word2Vec

- Learning parameters with Gradient Ascent

$$J(\theta) = \frac{1}{T} \sum_{t=1}^T \sum_{-m \leq j \leq m, j \neq 0} \log p(w_{t+j} | w_t) \quad p(o|c) = \frac{\exp(u_o^T v_c)}{\sum_{w=1}^W \exp(u_w^T v_c)}$$

- Compute the gradient

$$\begin{aligned} \frac{\partial}{\partial v_c} \log p(o|c) &= \frac{\partial}{\partial v_c} \log \frac{\exp(u_o^T v_c)}{\sum_{w=1}^W \exp(u_w^T v_c)} \\ &= \underbrace{\frac{\partial}{\partial v_c} u_o^T v_c}_A - \underbrace{\frac{\partial}{\partial v_c} \log \sum_{w=1}^W \exp(u_w^T v_c)}_B \end{aligned}$$

/\* elice \*/



## 04 DL-based Representations

### ✓ Word2Vec

- Learning parameters with Gradient Ascent

✓ For chunk **A**

$$\frac{\partial}{\partial v_c} u_o^T v_c = u_o$$

✓ For chunk **B**

$$\begin{aligned} & -\frac{\partial}{\partial v_c} \log \sum_{w=1}^W \exp(u_w^T v_c) \\ &= -\frac{1}{\sum_{w=1}^W \exp(u_w^T v_c)} \cdot \left( \sum_{w=1}^W \exp(u_w^T v_c) \cdot u_w \right) \\ &= -\sum_{w=1}^W \frac{\exp(u_w^T v_c)}{\sum_{w=1}^W \exp(u_w^T v_c)} \cdot u_w = -\sum_{w=1}^W P(w|c) \cdot u_w \end{aligned}$$

/\* elice \*/

## 04 DL-based Representations

### ✓ Word2Vec

- Learning parameters with Gradient Ascent

$$\frac{\partial}{\partial v_c} \log p(o|c) = u_o - \sum_{w=1}^W P(w|c) \cdot u_w$$

- Update the weight vector

$$v_c(t+1) = v_c(t) + \alpha \left( u_o - \sum_{w=1}^W P(w|c) \cdot u_w \right)$$

## 04 DL-based Representations

### ✓ Fasttext

- NNLM, Word2vec의 단점
  - ✓ morphology를 무시하고 각 단어를 다른 벡터로 Embedding 시킴.
  - ✓ Fasttext는 이러한 문제를 해결하기 위해 character n-grams을 고려한 Embedding을 고안함.
  - ✓ 따라서 단어를 n-gram vectors들의 평균/합으로 Embedding시킴.

## 04 DL-based Representations

### ✓ Fasttext

- 단어가 아닌 단어 내부의 n-gram이 최소 단위!



## 04 DL-based Representations

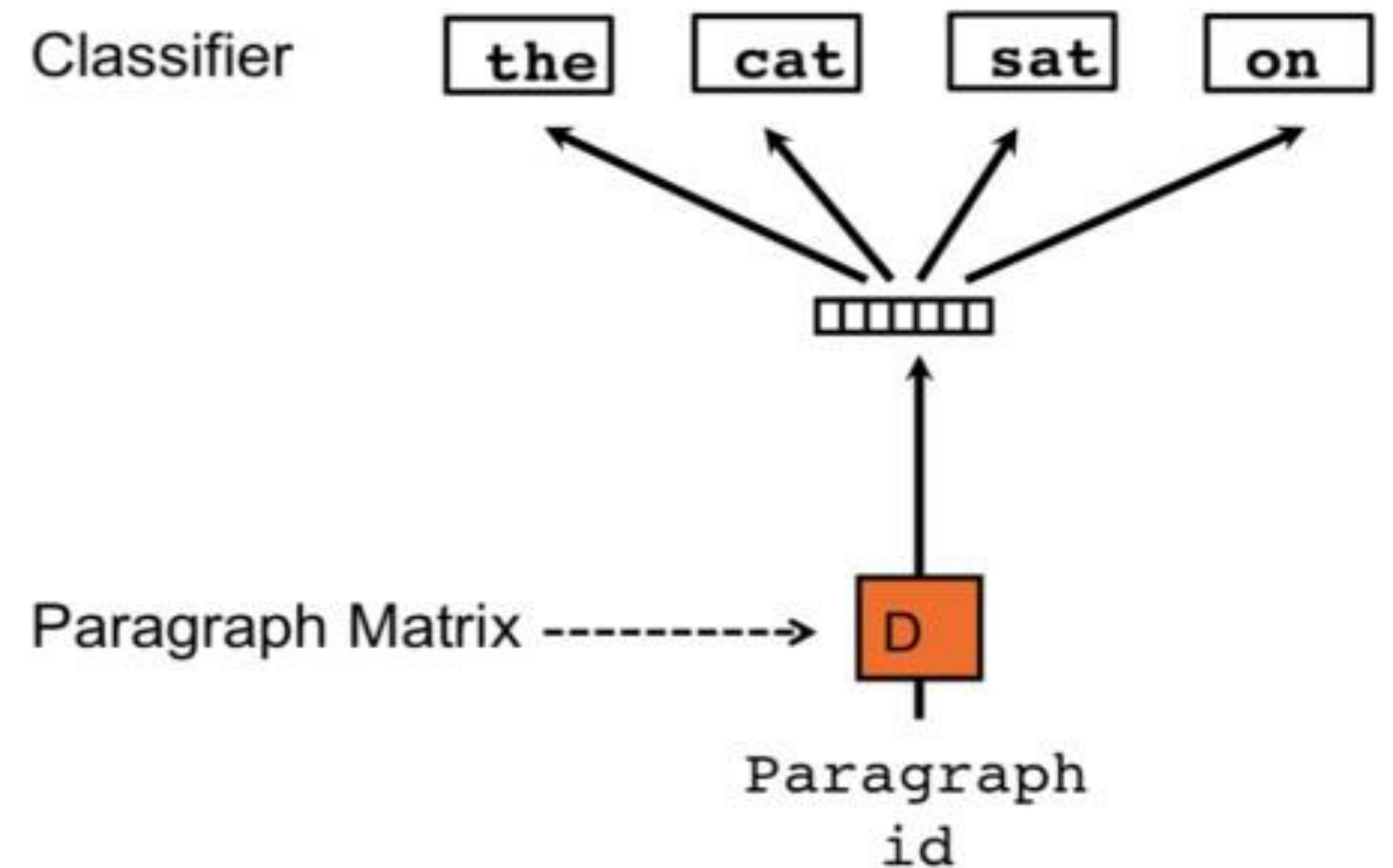
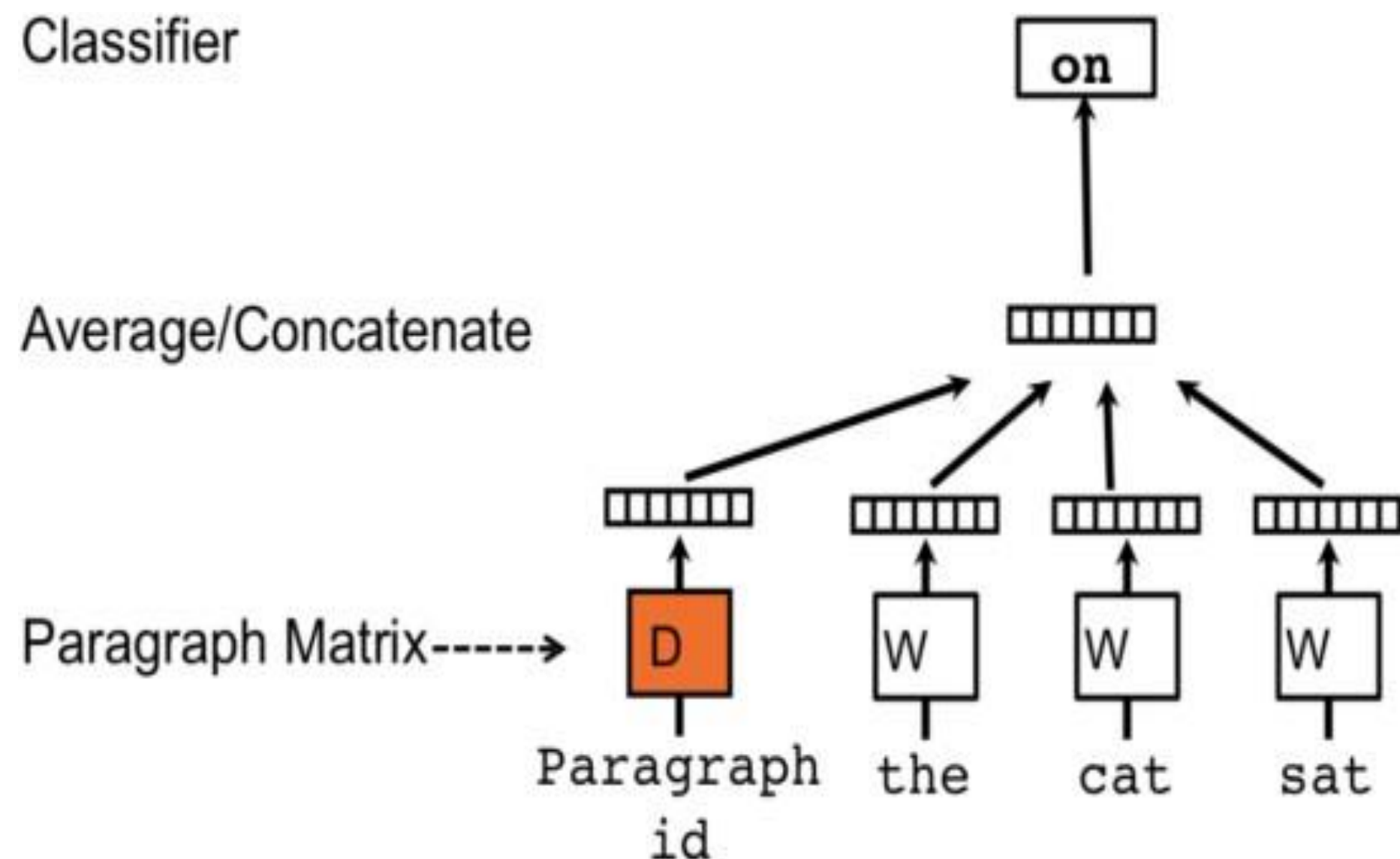
### ✓ Fasttext

- train corpus에 존재하지 않았던 단어의 embedding이 가능함 (ex) 'disaster' / 'disastrous'
- 희소한 단어에 대해 더 좋은 embedding이 가능함

## 04 DL-based Representations

### ✓ Doc2Vec

- Word2Vec 에 이어 2014년 구글 연구팀이 발표한 문서 임베딩 모델
- 타겟 단어와 이전 단어 k 개가 주어졌을 때, 이전 단어들 + 해당 문서의 아이디로 타겟 단어를 예측하는 과정에서 문맥이 비슷한 문서 벡터와 단어 벡터가 유사하게(코사인 유사도) 임베딩됨.





## 04 DL-based Representations

### ✓ Doc2Vec(PV-DM)

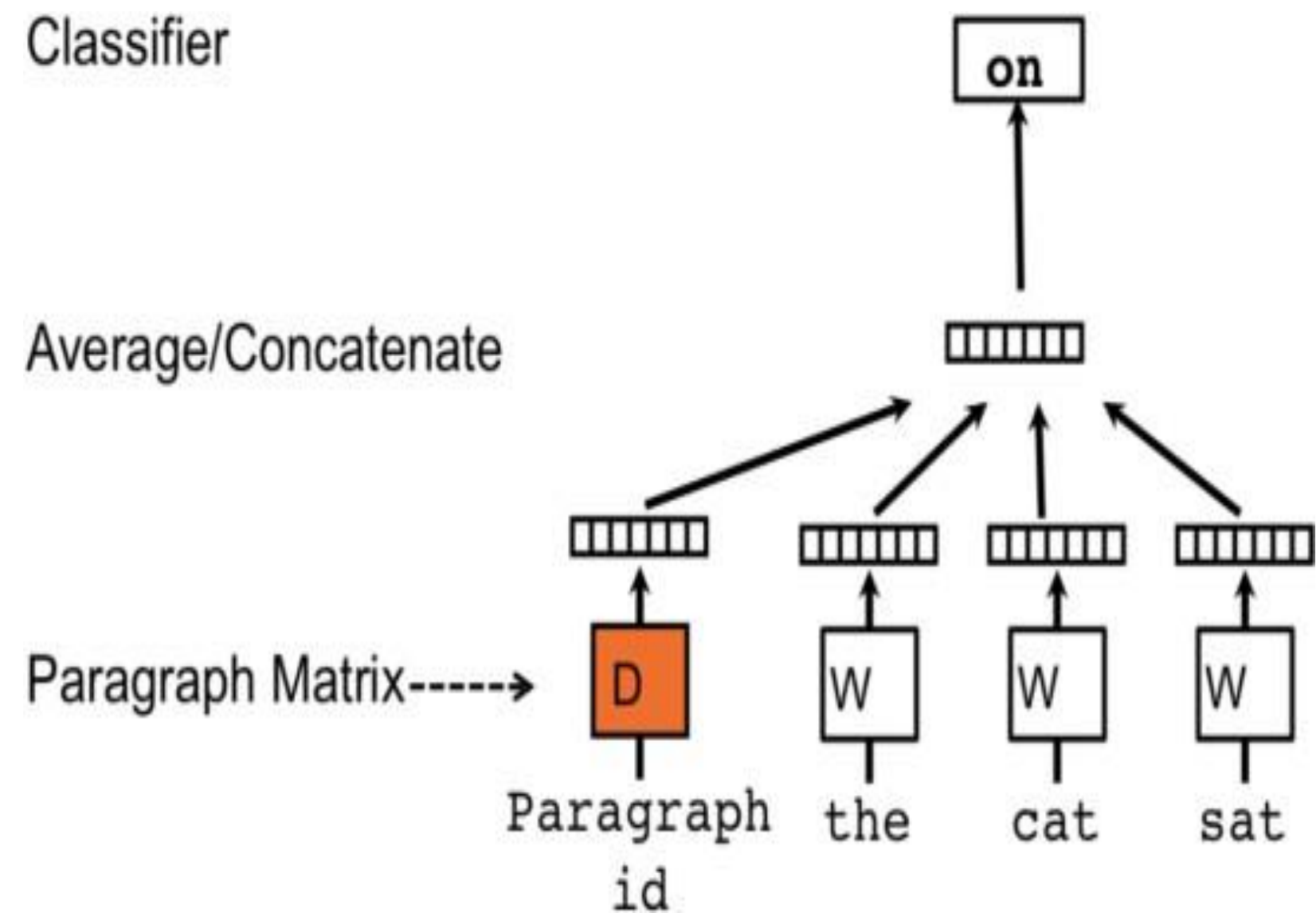
- paragraph\_1 이라는 문서에서 the cat sat on the mat 라는 문장이 있을 때, 다음과 같이 학습 데이터가 구성된다.
- Word도 Embedding이 됨.

윈도우 사이즈 :  $k = 3$

[paragraph\_1, the, cat, sat] - on

[paragraph\_1, cat, sat, on] - the

[paragraph\_1, sat, on, the] - mat

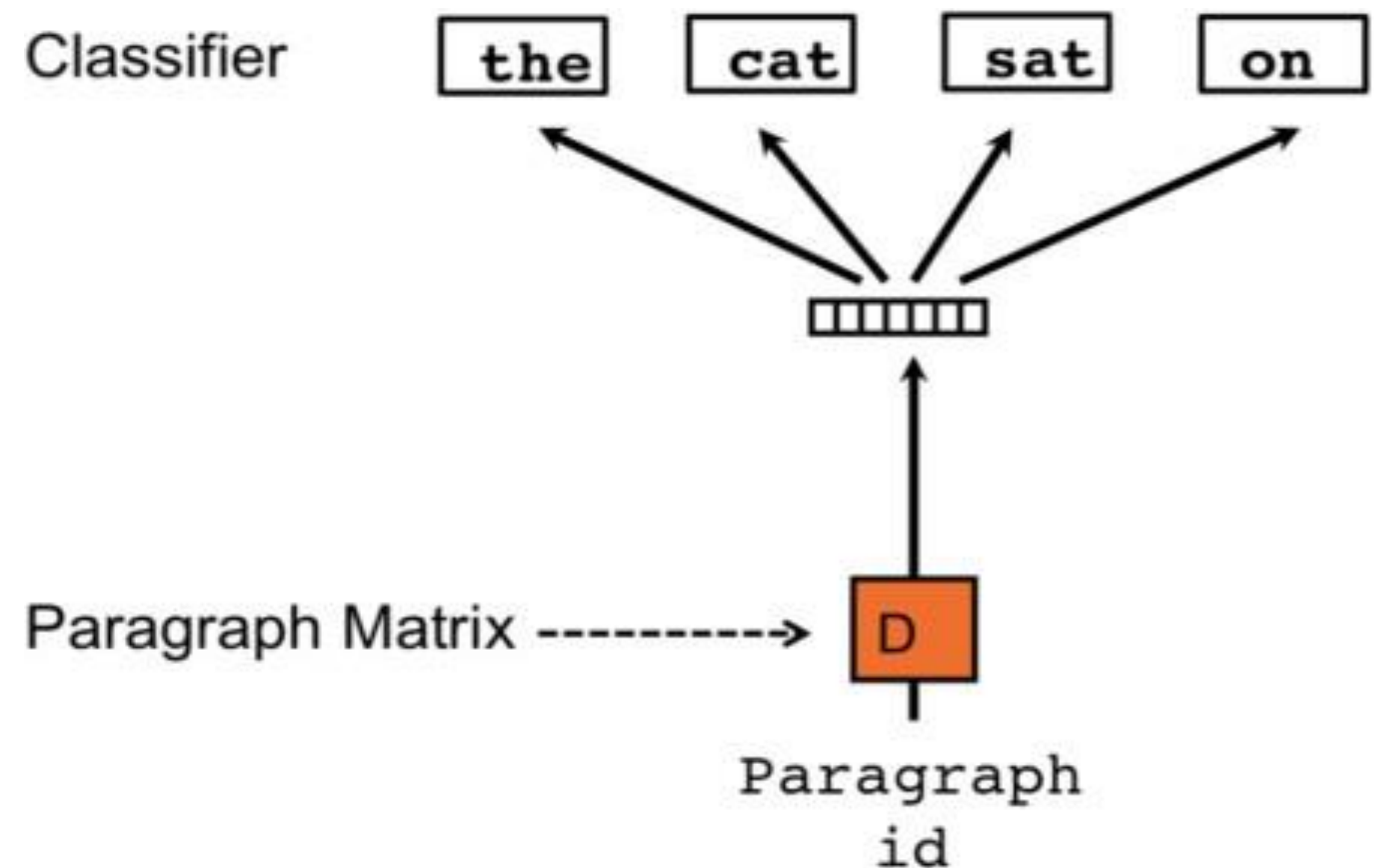


/\* elice \*/

## 04 DL-based Representations

### ✓ Doc2Vec(PV-DBOW)

- 문장에 존재하는 단어들의 문맥은 무시하고 오로지 idx만 투입돼 랜덤하게 뽑힌 word를 예측하도록 학습이 됨.
- Word Embedding이 안됨.
- 주로 PV-DM을 쓰나 주 방법을 혼용해서 쓰기도함





# 실습 - 국민청원 Project

## 국민청원 추천 시스템 구현 및 효율화

TOBIGS  
김수지 서석현 이준걸  
임소정 정민호 황이은

# Credit

/\* elice \*/

코스 매니저

임승연

콘텐츠 제작자

임승연

강사

이준걸

감수자

김수인

디자인

박주연

# Contact

TEL

010-2014-6910

WEB

<https://elice.io>

E-MAIL

[i2326@naver.com](mailto:i2326@naver.com)

