Text Representation

Jehyuk Lee

Department of AI, Big Data & Management

Kookmin University

Contents

- **Statistical Text Representation**
  - Word Representation
  - Document Representation

- **NN based Text Representation**
  - **Word2Vec**
  - **Doc2Vec**

1. Statistical Text Representation

Text를 벡터로 표현하기

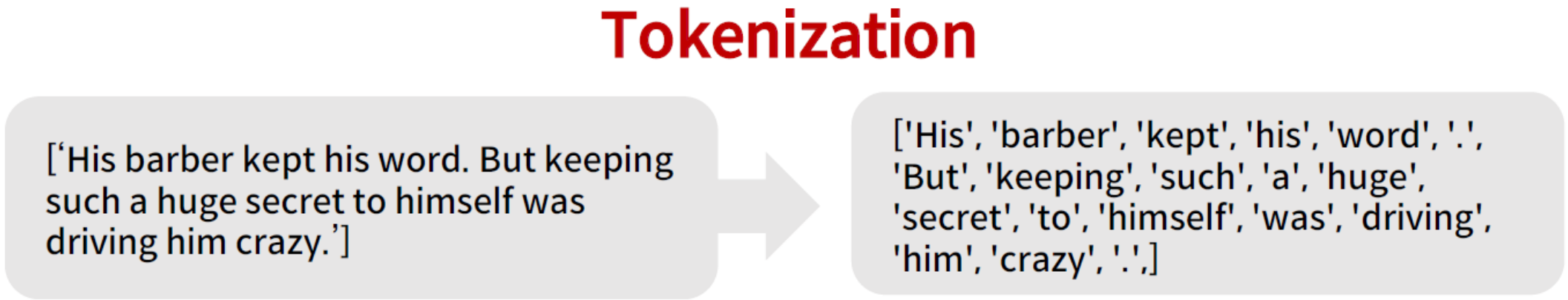
- **우리는 Text를 vector형태로 표현해야 합니다.**
  - 컴퓨터는 숫자 밖에 모르는 바보
  - 특히, Machine Learning task를 수행하기 위해서는 input을 vector형태로 바꿔야 함
    - 많은 ML 알고리즘들은 input값으로 벡터를 사용
  - 어떻게 표현해야 하는가?
    - 단어는?
    - 문서는?

Text를 벡터로 표현하기

- **우리는 Text를 vector형태로 표현해야 합니다.**
  - 컴퓨터는 숫자 밖에 모르는 바보
  - 특히, Machine Learning task를 수행하기 위해서는 input을 vector형태로 바꿔야 함
    - 많은 ML 알고리즘들은 input값으로 벡터를 사용
  - 어떻게 표현해야 하는가?
    - 단어는
      - 인공 신경망 미사용: One-hot encoding
      - 인공 신경망 사용: Word Embedding(Word2Vec, Glove, FastText 등)
    - 문서는
      - 인공 신경망 미사용: Document Term Matrix(DTM), TF-IDF
      - 인공 신경망 사용: Doc2Vec, Sent2Vec 등

Word Representation

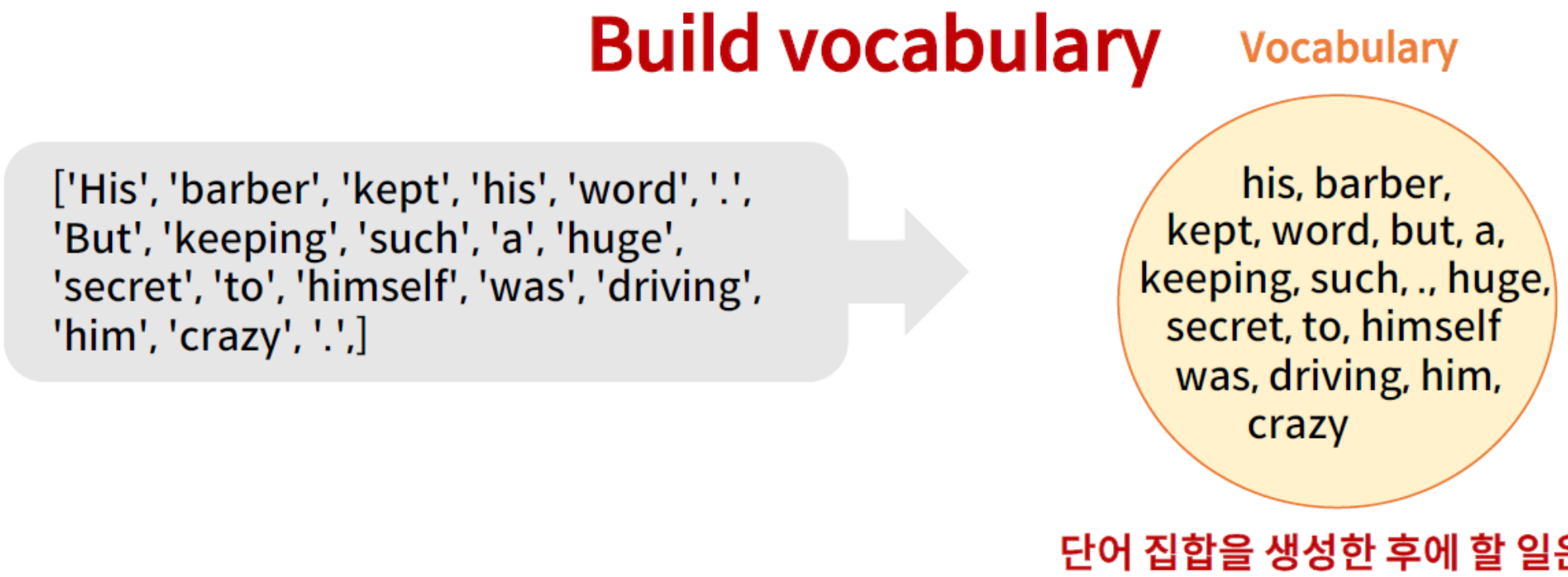
- **Text preprocessing(복습)**
  - **Tokenization**



(그림 출처: 딥 러닝을 이용한 자연어 처리 입문, 안상준, 유원준 저, Wikibooks)

Word Representation

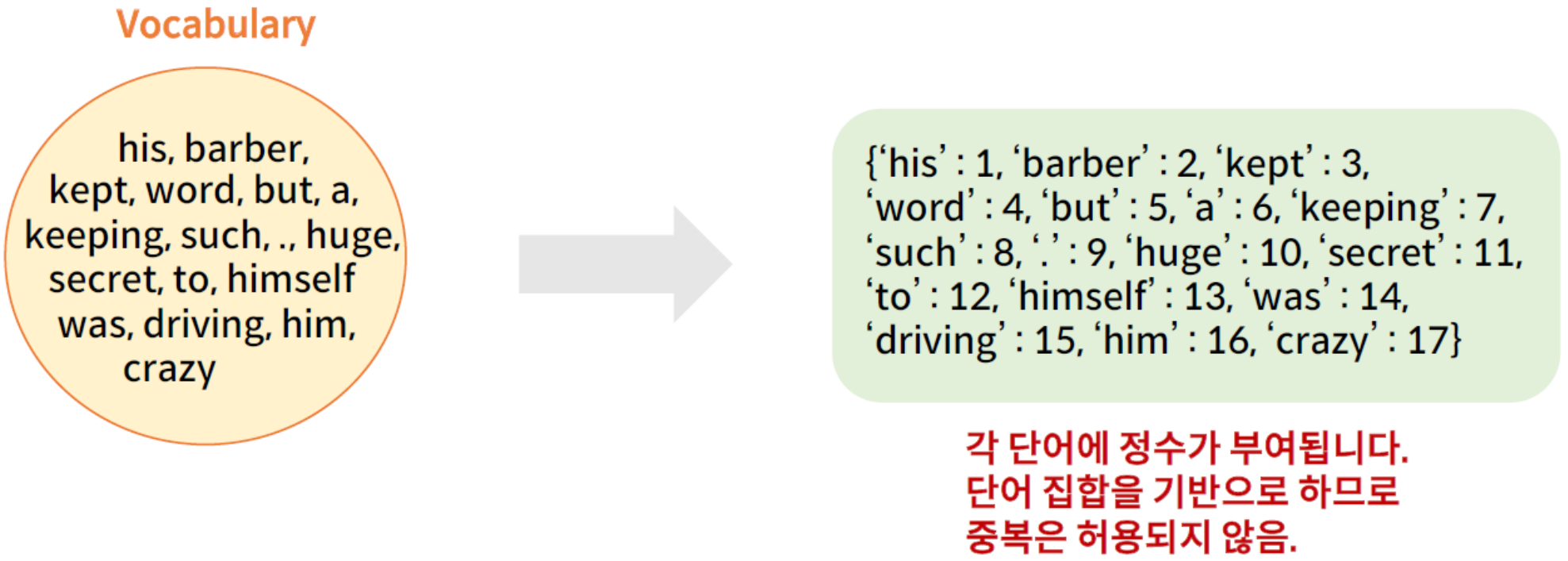
- **Text preprocessing(복습)**
  - Vocabulary set생성



(그림 출처: 딥 러닝을 이용한 자연어 처리 입문, 안상준, 유원준 저, Wikibooks)

Word Representation

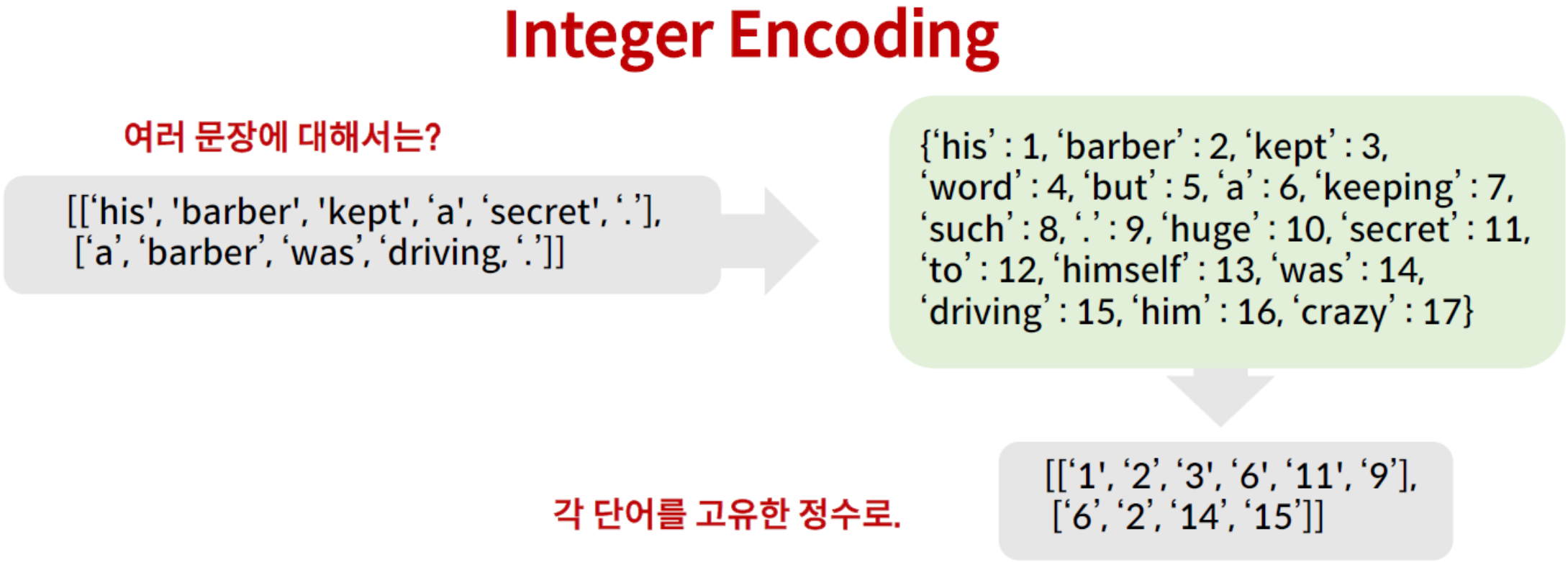
- **Text preprocessing(복습)**
  - Integer Encoding
    - Vocabulary set에 고유한 정수를 mapping



(그림 출처: 딥 러닝을 이용한 자연어 처리 입문, 안상준, 유원준 저, Wikibooks)

Word Representation

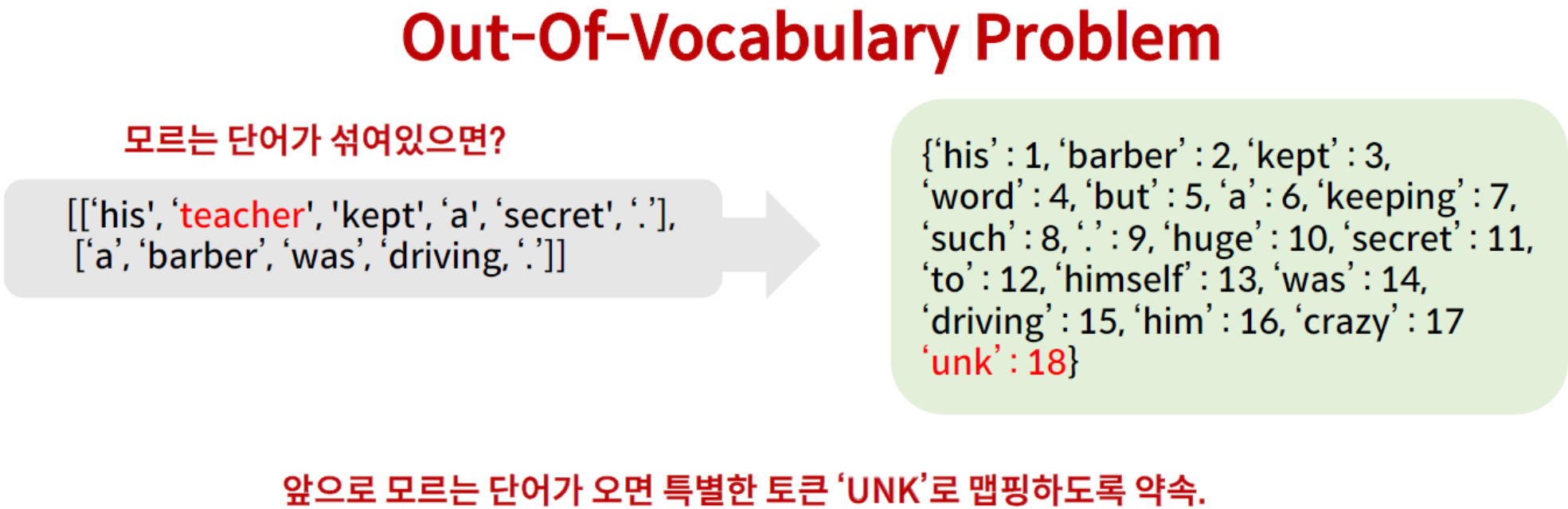
- **Text preprocessing(복습)**
  - Integer Encoding
    - 새로운 문장이 들어오면 integer sequence로 변환



(그림 출처: 딥 러닝을 이용한 자연어 처리 입문, 안상준, 유원준 저, Wikibooks)

Word Representation

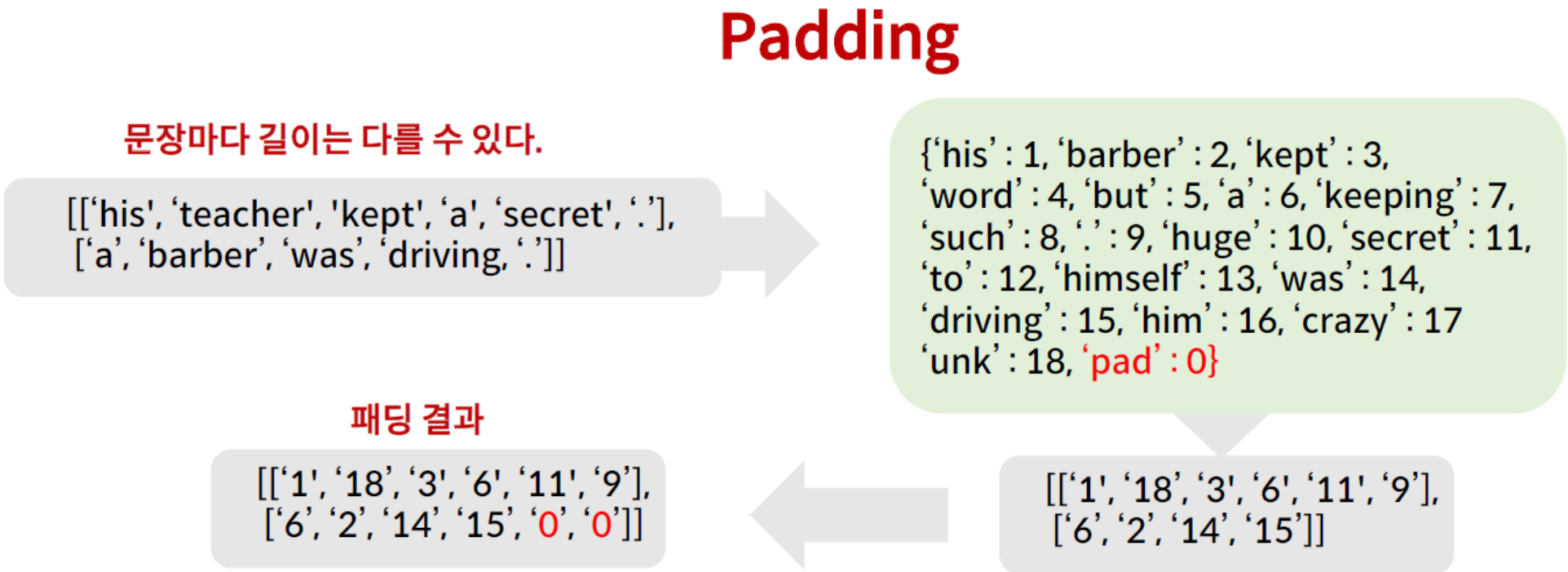
- **Text preprocessing(복습)**
  - Integer Encoding
    - OOV(Out-of-Vocabulary): Vocabulary set에 없는 단어. 하나의 토큰으로 mapping



(그림 출처: 딥 러닝을 이용한 자연어 처리 입문, 안상준, 유원준 저, Wikibooks)

Word Representation

- **Text preprocessing(복습)**
  - Padding
    - 여러 개의 문장을 병렬적으로 처리할 때 필요. 문장의 길이를 맞춰주자!



(그림 출처: 딥 러닝을 이용한 자연어 처리 입문, 안상준, 유원준 저, Wikibooks)

Word Representation

- **One-hot encoding**
  - 전체 vocabulary set의 크기를 차원으로 하는 vector로 표현
    - Ex) 전체 vocabulary set에 10개의 단어 → 10차원의 vector로 표현
  - 각 단어에 고유 integer index를 부여. 해당 index의 값은 1, 나머지는 0인 벡터
  - 한계
    - 유사한 단어 벡터들 간에 유의미한 유사도를 잘 측정할 수 없음
      - 예시) dog와 puppy는 유사한 단어. dog=[1,0,0,0,0], puppy=[0,1,0,0,0] → 유사도는 0
    - 벡터의 크기가 vocabulary set 크기
      - Sparse vector
      - 많은 양의 저장 공간과 높은 계산 복잡도가 필요

Word Representation

- **One-hot encoding**

입력된 문장

['his', 'teacher', 'kept', 'a', 'secret', '.']

{'his' : 1, 'barber' : 2, 'kept' : 3,
'word' : 4, 'but' : 5, 'a' : 6, 'keeping' : 7,
'such' : 8, '.' : 9, 'huge' : 10, 'secret' : 11,
'to' : 12, 'himself' : 13, 'was' : 14,
'driving' : 15, 'him' : 16, 'crazy' : 17
'unk' : 18, 'pad' : 0}

['1', '18', '3', '6', '11', '9']

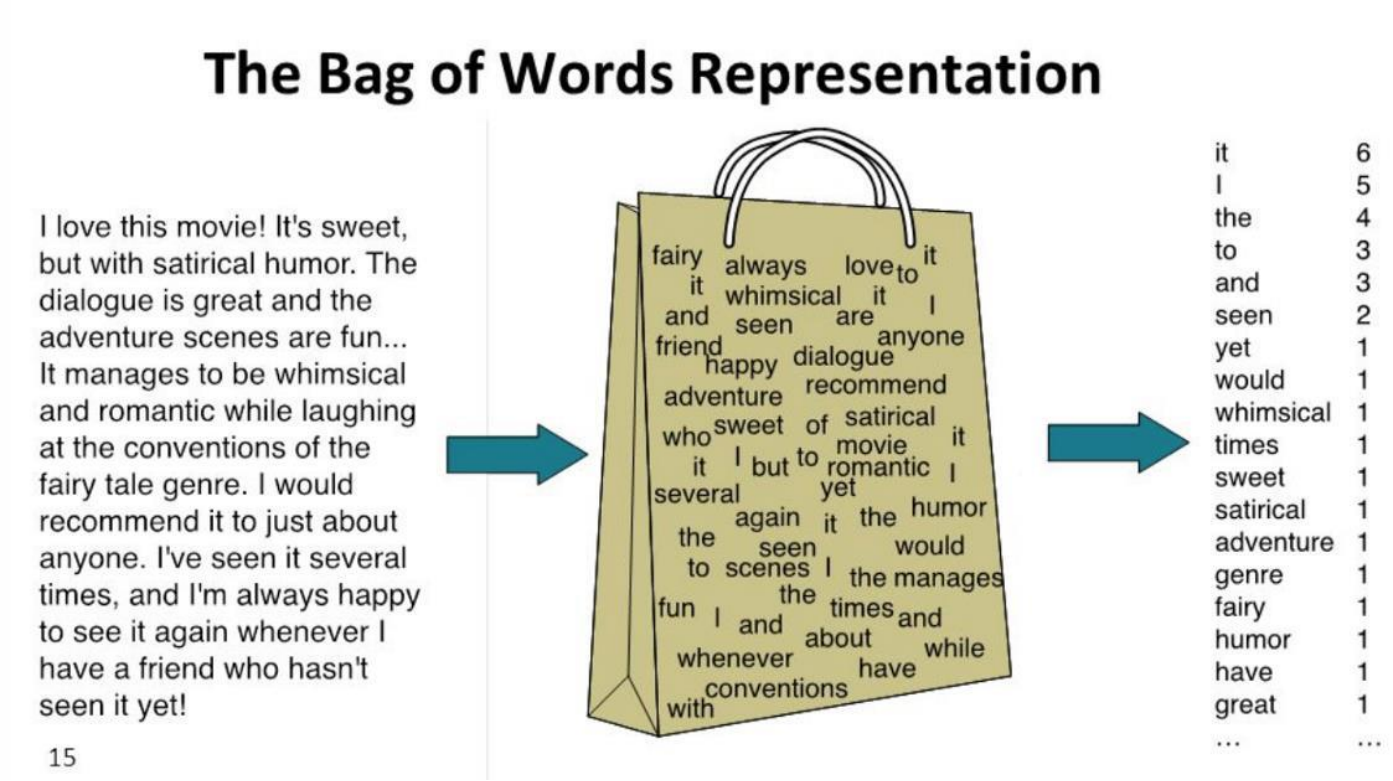
원-핫 인코딩.

0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0

(그림 출처: 딥 러닝을 이용한 자연어 처리 입문, 안상준, 유원준 저, Wikibooks)

Document Representation

- **Bag-of-words (BoW)**
  - 문서를 단어의 빈도수로 나타내는 방식
  - 단어의 순서를 고려하지 않음



(그림 출처: https://slideplayer.com/slide/7073400/)

Document Representation

- **Document Term Matrix (DTM)**
  - 여러 개의 문서를 BoW 형태로 나타낸 matrix

문서1 : 먹고 싶은 사과
문서2 : 먹고 싶은 바나나
문서3 : 길고 노란 바나나 바나나
문서4 : 저는 과일이 좋아요

Integer encoding

과일이	0
길고	1
노란	2
먹고	3
바나나	4
사과	5
싶은	6
저는	7
좋아요	8

Bag of Words 가설 기반

0 1 2 3 4 5 6 7 8

-	과일이	길고	노란	먹고	바나나	사과	싶은	저는	좋아요
문서1	0	0	0	1	0	1	1	0	0
문서2	0	0	0	1	1	0	1	0	0
문서3	0	1	1	0	2	0	0	0	0
문서4	1	0	0	0	0	0	0	1	1

Vocabulary size

(그림 출처: 딥 러닝을 이용한 자연어 처리 입문, 안상준, 유원준 저, Wikibooks)

Document Representation

- **Document Term Matrix (DTM)**
  - DTM의 한계
    - Sparse Matrix
      - too large dimension with zero value
      - 많은 양의 저장 공간과 높은 계산 복잡도가 필요
    - 단순 빈도 수 기반 접근
      - 불용어(stopwords)는 많은 문서에서 자주 발생함 (I, you, at, on 등)
      - 반면, 중요한 단어는 일부 문서에서만 자주 발생함
      - 이런 불용어와 중요한 단어에 가중치를 주는 방법이 필요 → **TF-IDF**

Document Representation

- **Term Frequency – Inverse Document Frequency (TF-IDF)**
  - DTM내의 단어들에 대해, 각 단어의 중요도에 따라 가중치를 부여하여 표현하는 방식
    - 중요도와 가중치는 어떻게 주는지?
  - TF-IDF를 산출하기 위해 다음과 같은 term들이 필요합니다.
    - tf(d,t)
      - Term Frequency
      - 문서 d에서 단어 t가 등장하는 횟수 → 사실 DTM과 동일함
    - df(t)
      - Document Frequency
      - 단어 t가 등장하는 문서의 수
    - idf(d,t)
      - Inverse Document Frequency
      - df(t)에 반비례 하는 수
      - 정의: $\ln\left(\frac{n}{1+df(t)}\right)$, (n: 문서의 수)

Document Representation

- **Term Frequency – Inverse Document Frequency (TF-IDF)**
  - idf(d,t)에서 굳이 log term을 사용하는 이유
    - 문서 수가 많아질수록 idf term이 기하급수적으로 커짐
    - 특히 희귀 단어들에 대해 지나치게 가중치가 지나치게 커짐 → 이를 방지하고자 log사용

log의 밑으로 e대신 10 사용

$$idf(d,t) = log(n/df(t))$$
$$n = 1,000,000$$

단어 t	$df(t)$	$idf(d,t)$
word1	1	6
word2	100	4
word3	1,000	3
word4	10,000	2
word5	100,000	1
word6	1,000,000	0

로그 사용

$$idf(d,t) = n/df(t)$$
$$n = 1,000,000$$

단어 t	$df(t)$	$idf(d,t)$
word1	1	1,000,000
word2	100	10,000
word3	1,000	1,000
word4	10,000	100
word5	100,000	10
word6	1,000,000	1

로그 미사용

log를 사용하지 않으면 idf 값이 지나치게 커짐

(그림 출처: 딥 러닝을 이용한 자연어 처리 입문, 안상준, 유원준 저, Wikibooks)

Document Representation

- **Term Frequency – Inverse Document Frequency (TF-IDF)**
  - 중요도
    - 모든 문서에서 나타난 단어(불용어 등)는 중요도가 낮음
      - df(t)가 높음 → idf(t)가 낮음
    - 특정 문서에서만 자주 나타난 단어는 중요도가 높음
      - df(t)가 낮음 → idf(t)가 높음

Document Representation

- **Term Frequency – Inverse Document Frequency (TF-IDF)**
  - TF-IDF는 tf와 idf를 곱한 값
  - DTM내의 단어들에 대해, 각 단어의 중요도에 따라 가중치를 부여하여 표현하는 방식
    - 중요도
      - 모든 문서에서 나타난 단어(불용어 등)는 중요도가 낮음 → 낮은 idf
      - 특정 문서에서만 자주 나타난 단어는 중요도가 높음 → 높은 idf
    - 가중치: idf
    - 즉, tf값에 idf라는 가중치를 곱해서 중요도에 따라 가중치를 부여하여 표현할 수 있음

Document Representation

- **Term Frequency – Inverse Document Frequency (TF-IDF)**
  - 예시

$$idf(t) = \ln\left(\frac{n}{1+df(t)}\right)$$

훈련 데이터

문서1 : 먹고 싶은 사과
문서2 : 먹고 싶은 바나나
문서3 : 길고 노란 바나나 바나나
문서4 : 저는 과일이 좋아요

DTM

-	과일이	길고	노란	먹고	바나나	사과	싶은	저는	좋아요
문서1	0	0	0	1	0	1	1	0	0
문서2	0	0	0	1	1	0	1	0	0
문서3	0	1	1	0	2	0	0	0	0
문서4	1	0	0	0	0	0	0	1	1

IDF

단어	IDF(역 문서 빈도)
과일이	ln(4/(1+1)) = 0.693147
길고	ln(4/(1+1)) = 0.693147
노란	ln(4/(1+1)) = 0.693147
먹고	ln(4/(2+1)) = 0.287682
바나나	ln(4/(2+1)) = 0.287682
사과	ln(4/(1+1)) = 0.693147
싶은	ln(4/(2+1)) = 0.287682
저는	ln(4/(1+1)) = 0.693147
좋아요	ln(4/(1+1)) = 0.693147

'과일이'라는 단어는 한 개의 문서(문서 4)에 등장
→ idf('과일이') = ln(4/(1+1)) = ln2 = 0.6931

'먹고'라는 단어는 두 개의 문서(문서 1,2)에 등장
→ idf('먹고') = ln(4/(2+1)) = ln(4/3) = 0.2876

'바나나'라는 단어는 두 개의 문서(문서 2,3)에 등장
(문서 3에 2번 등장했지만, 이에 무관하게 '바나나'라는 단어가 등장한 문서의 수는 2개)

→ Idf('바나나') = ln(4/(2+1)) = ln(4/3) = 0.2876

(그림 출처: 딥 러닝을 이용한 자연어 처리 입문, 안상준, 유원준 저, Wikibooks)

Document Representation

- **Term Frequency – Inverse Document Frequency (TF-IDF)**
  - 예시

훈련 데이터

문서1 : 먹고 싶은 사과
문서2 : 먹고 싶은 바나나
문서3 : 길고 노란 바나나 바나나
문서4 : 저는 과일이 좋아요

DTM

-	과일이	길고	노란	먹고	바나나	사과	싶은	저는	좋아요
문서1	0	0	0	1	0	1	1	0	0
문서2	0	0	0	1	1	0	1	0	0
문서3	0	1	1	0	2	0	0	0	0
문서4	1	0	0	0	0	0	0	1	1

×

IDF

단어	IDF(역 문서 빈도)
과일이	ln(4/(1+1)) = 0.693147
길고	ln(4/(1+1)) = 0.693147
노란	ln(4/(1+1)) = 0.693147
먹고	ln(4/(2+1)) = 0.287682
바나나	ln(4/(2+1)) = 0.287682
사과	ln(4/(1+1)) = 0.693147
싶은	ln(4/(2+1)) = 0.287682
저는	ln(4/(1+1)) = 0.693147
좋아요	ln(4/(1+1)) = 0.693147

=

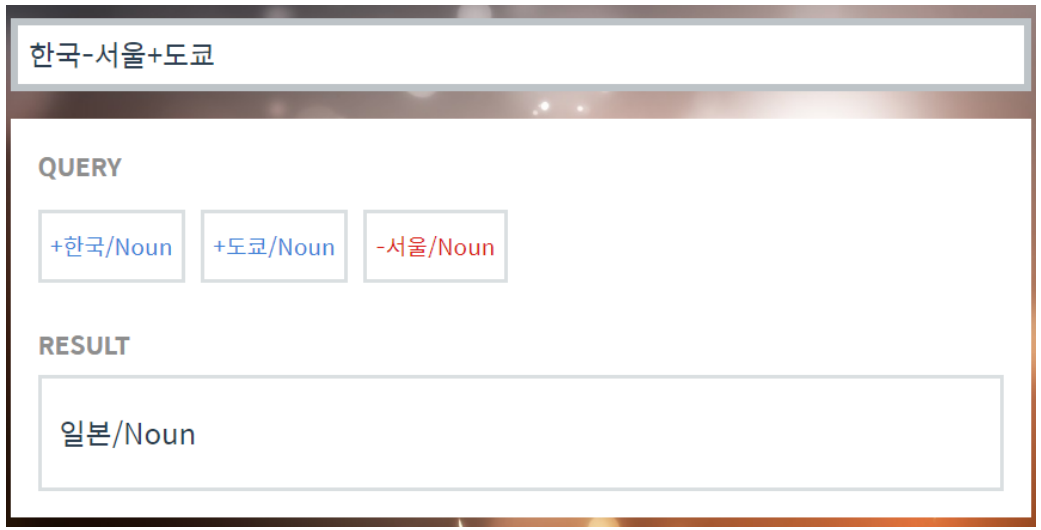
-	과일이	길고	노란	먹고	바나나	사과	싶은	저는	좋아요
문서1	0	0	0	0.287682	0	0.693147	0.287682	0	0
문서2	0	0	0	0.287682	0.287682	0	0.287682	0	0
문서3	0	0.693147	0.693147	0	0.575364	0	0	0	0
문서4	0.693147	0	0	0	0	0	0	0.693147	0.693147

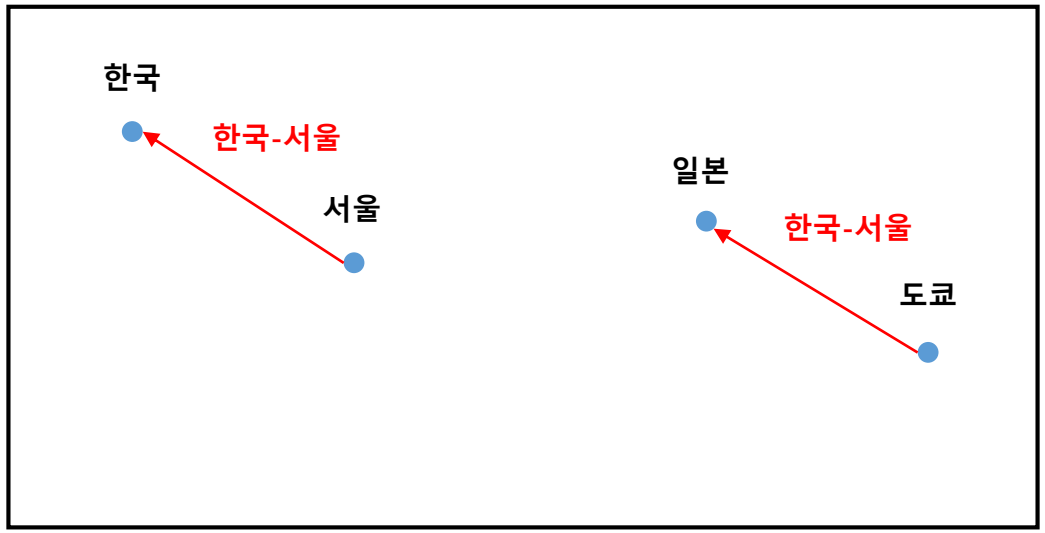
(그림 출처: 딥 러닝을 이용한 자연어 처리 입문, 안상준, 유원준 저, Wikibooks)

2. NN-based Text Representation

Word2Vec

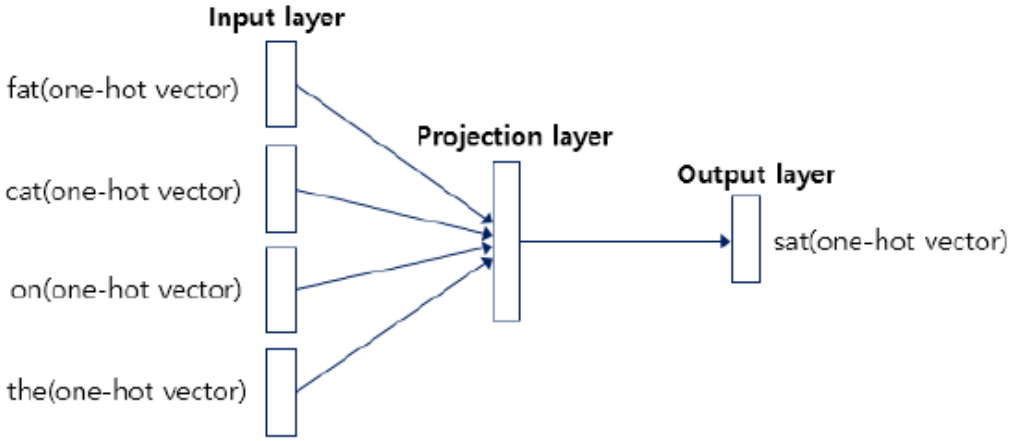
- **단어 벡터 간 유의미한 유사도를 반영할 수 있는 벡터 표현 방법**
  - https://word2vec.kr/search/
    - 한국어 단어에 대해서 벡터 연산을 시도해 볼 수 있음
    - 단어 벡터가 단어 벡터간 유사도를 반영한 값을 가지고 있기 때문에 가능함





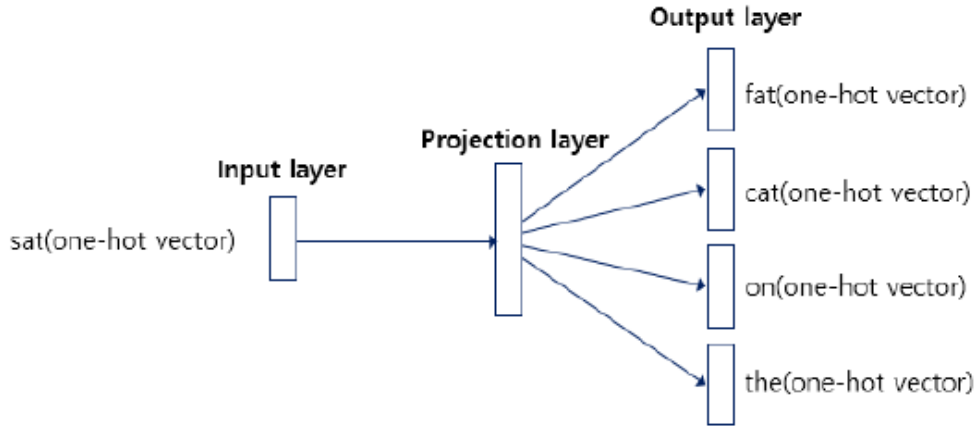
Word2Vec

- **Word2Vec에는 크게 2가지 학습 방법이 존재**
  - **Continuous Bag-of-Words(CBOW)**
    - 주변 단어로부터 중심 단어를 예측하는 task에서 단어의 vector representation학습
  - **Skip-gram**
    - 중심 단어로부터 주변 단어를 예측하는 task에서 단어의 vector representation 학습



CBoW

주변 단어들로부터 중심 단어를 예측



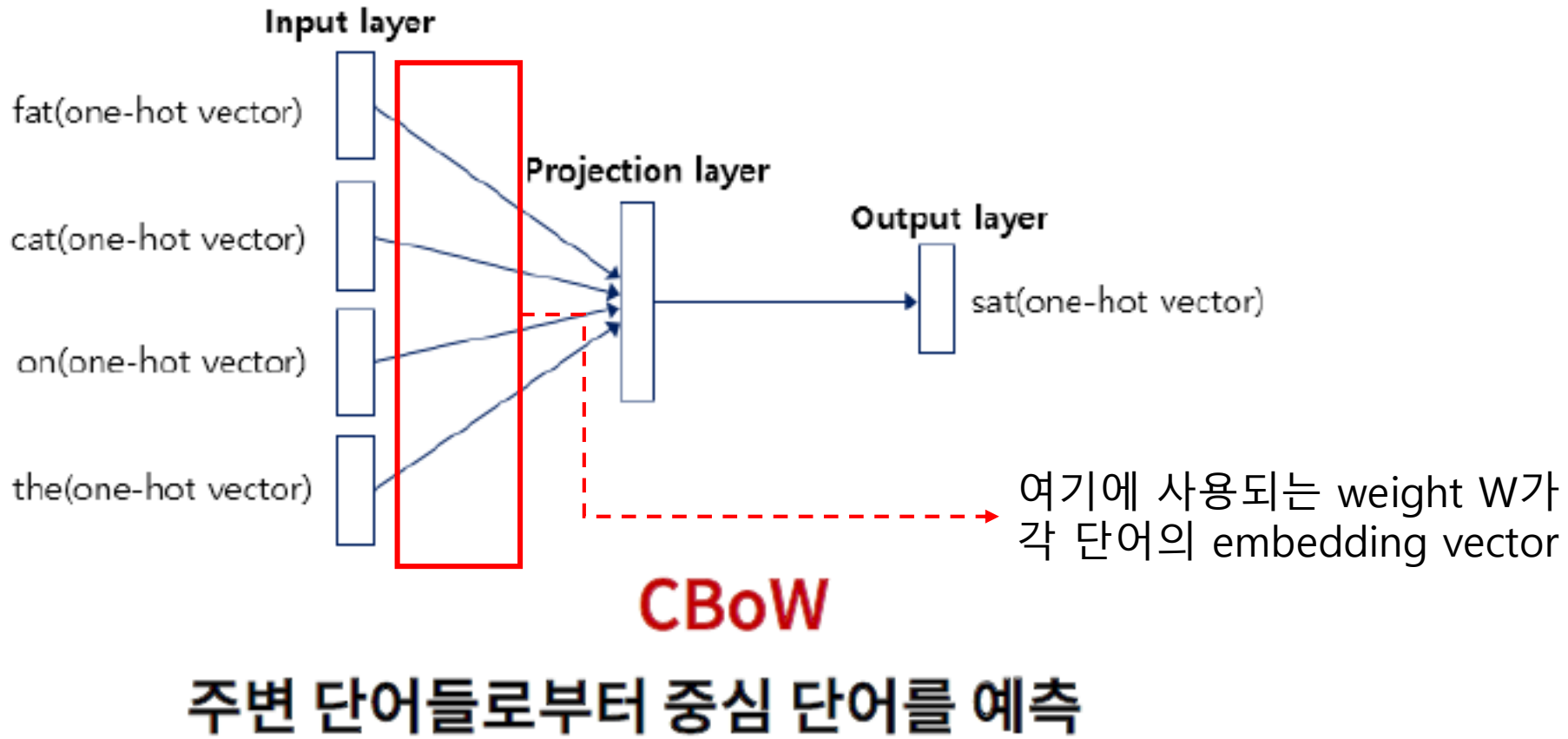
Skip-gram

중심 단어들로부터 주변 단어를 예측

(그림 출처: 딥 러닝을 이용한 자연어 처리 입문, 안상준, 유원준 저, Wikibooks)

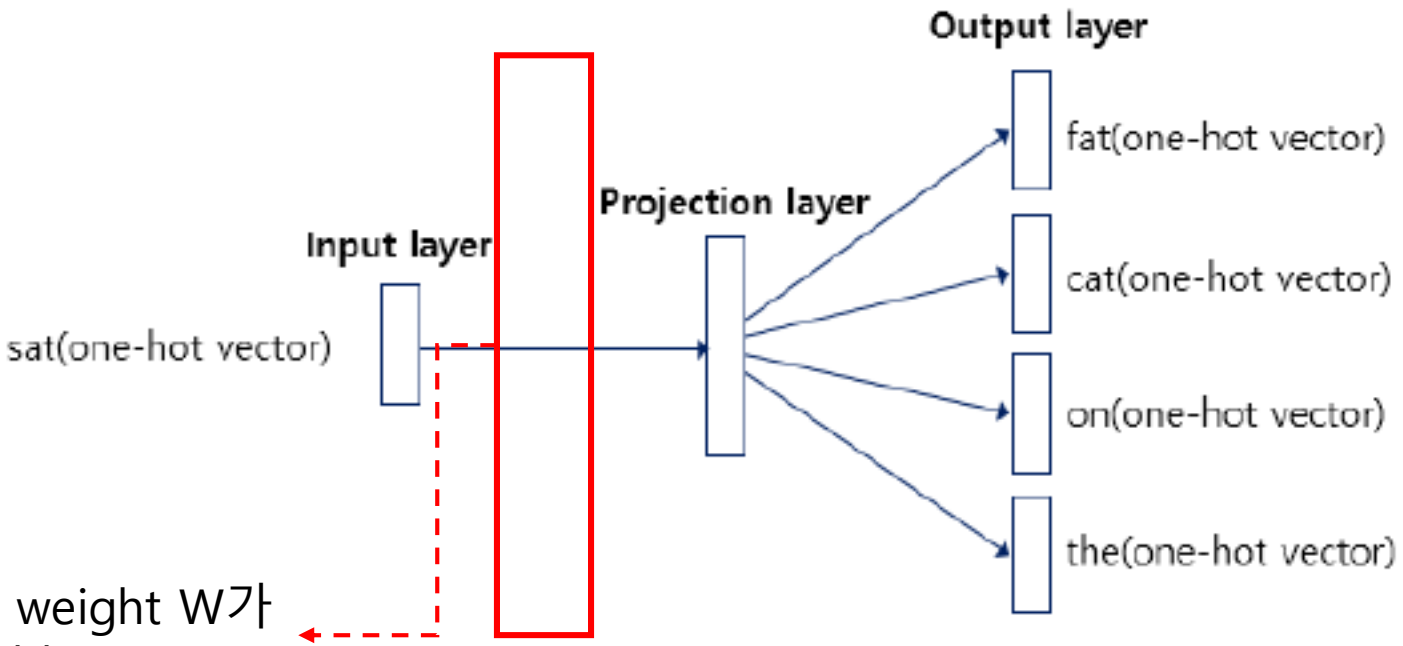
Word2Vec(1): CBOW

- 주변 단어에서 중심 단어를 예측하는 task에서 임베딩 벡터 학습



Word2Vec(2): Skip-gram

- 중심 단어에서 주변 단어들을 예측하는 task에서 임베딩 벡터 학습



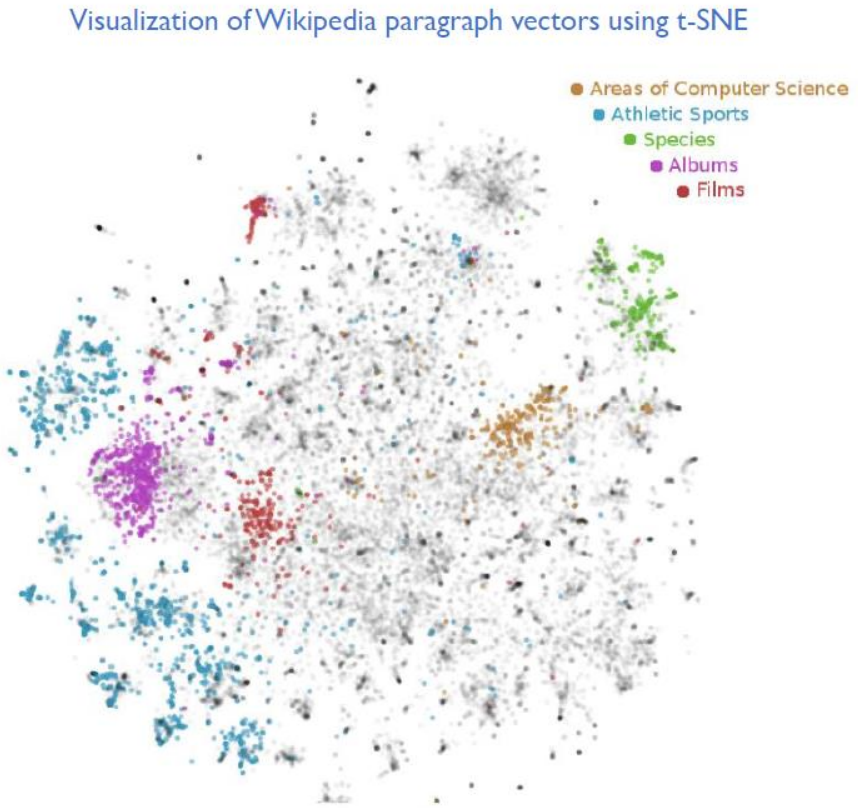
여기에 사용되는 weight W가
각 단어의 embedding vector

Skip-gram

중심 단어들로부터 주변 단어를 예측

Represent documents in vector?

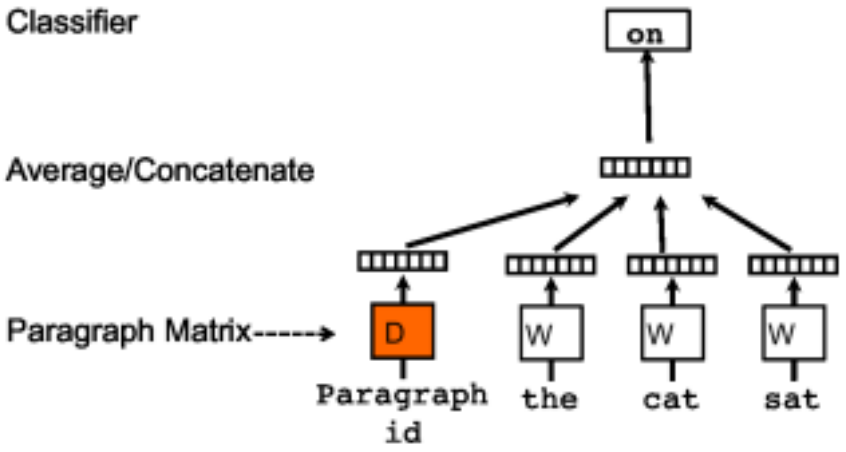
- **Embedding documents in vector space?**
  - 단어들은 word2vec 등의 방법으로 임베딩 했는데,
  - Documents, sentences, phrases도 임베딩 할 수 있지 않을까?



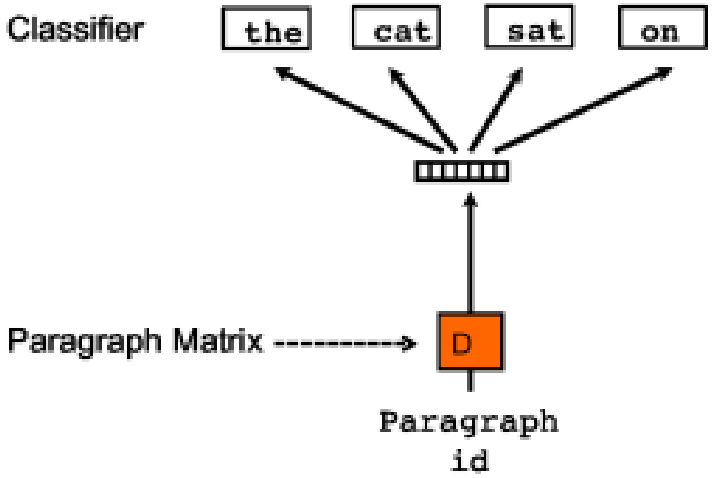
(그림 출처: Dai, A. M., Olah, C., & Le, Q. V. (2015). Document embedding with paragraph vectors. arXiv preprint arXiv:1507.07998.)

Doc2Vec

- **Documents들도 비슷한 방식으로 임베딩 해보자!**
  - Word 벡터들과 함께 임베딩 해보자



PV-DM



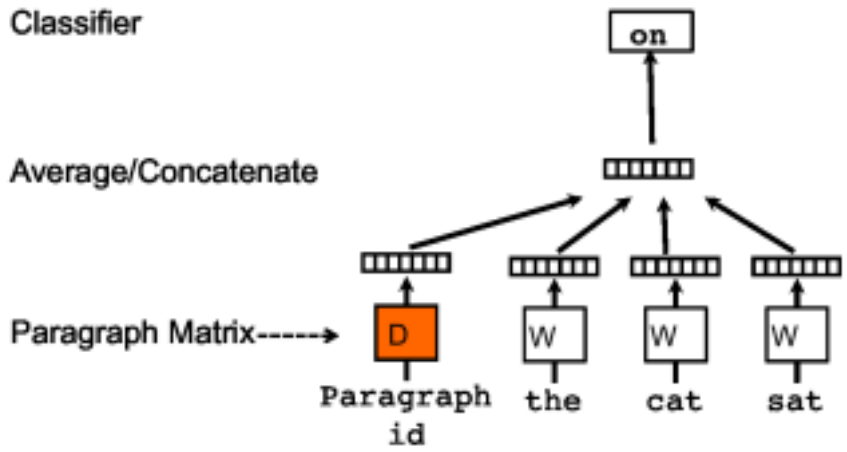
PV-DBOW

(그림 출처: Le, Q., & Mikolov, T. (2014, June). Distributed representations of sentences and documents. In International conference on machine learning (pp. 1188-1196). PMLR.)

Doc2Vec

document를 word2vec와 비슷하게 임베딩하는 방법

- **Paragraph Vector-model: Distributed Memory(PV-DM) model**
  - NNLM과 거의 비슷한 구조인데, paragraph 함께 input으로 넣어서 학습
    - NNLM: 이전 n개의 단어로부터 그 다음 1개의 단어를 예측 (by FFNN)
  - **Training Phase**
    - 학습 corpus의 Paragraph vector, word vector, classifier가 함께 학습됨
  - **Inference Phase**
    - Word vector와 classifier는 이미 학습된 상태
    - 새로운 paragraph가 들어왔을 때
      - 이에 대한 document vector는 아직 없음
      - Document vector도 gradient descent로 얻게 됨



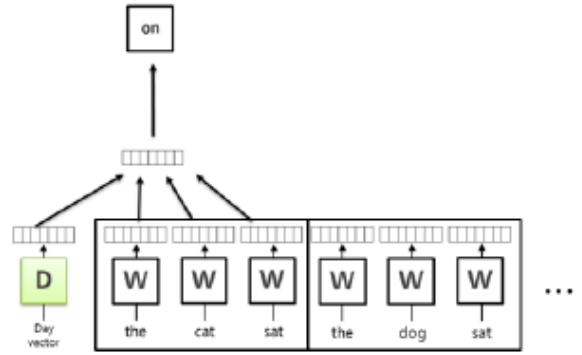
PV-DM

(그림 출처: Le, Q., & Mikolov, T. (2014, June). Distributed representations of sentences and documents. In International conference on machine learning (pp. 1188-1196). PMLR.)

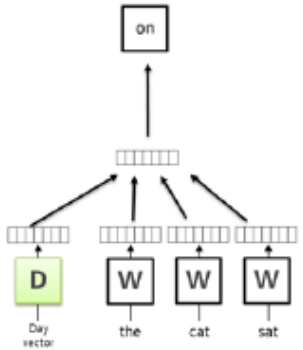
More things to embed?

- **Day embedding in News corpus**

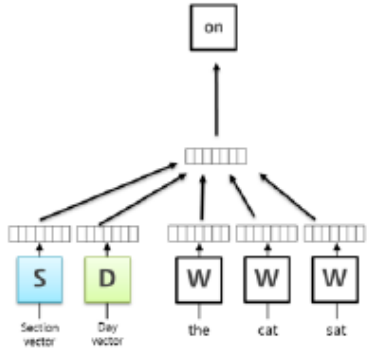
Approach 01: 하루 동안의 기사제목들을 병합 후 Day 태그



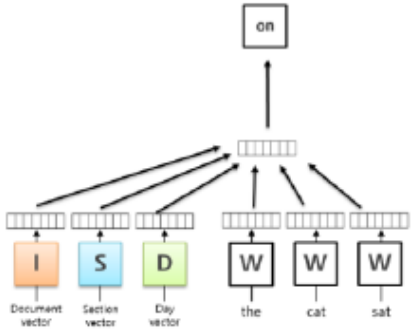
Approach 02: 각 뉴스기사 제목에 Day 태그



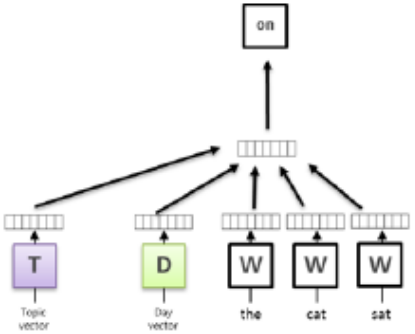
Approach 03: 각 뉴스기사제목에 Day, Section 태그



Approach 04: 각 뉴스기사제목에 Day, Section, IDX 태그



Approach 05: 각 뉴스기사제목에 Day, Topic 태그



(그림 출처: https://github.com/pilsung-kang/Text-Analytics/blob/master/05%20Text%20Representation%20II%20-%20Distributed%20Representation/05_Text%20Representation%20II%20-%20Distributed%20Representation_FastText.pdf)

More things to embed?

- **Live2Vec: In Afreeca TV**

[그림10] 문장과 Live 방송의 추론의 예시

(그림 출처: https://brunch.co.kr/@goodvc78/16)

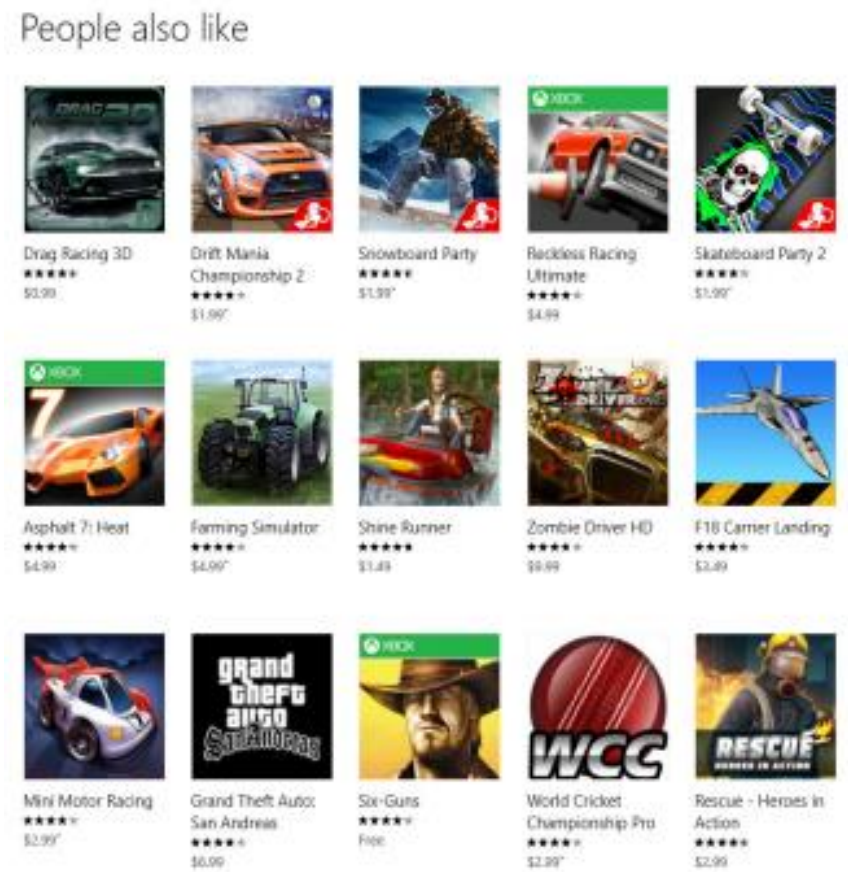
More things to embed?

- **Song2Vec: For recommendation**
  - Playlist를 sentence로 각 song들을 word들로 간주하여 embedding

(그림 출처: https://velog.io/@mstar228/Song2Vec)

More things to embed?

- **Item2Vec: For recommendation**
  - 하나의 'session' (not user)에서 조회한 상품들을 word로 간주하여 embedding



(그림 출처: Barkan, O., & Koenigstein, N. (2016, September). Item2vec: neural item embedding for collaborative filtering. In 2016 IEEE 26th International Workshop on Machine Learning for Signal Processing (MLSP) (pp. 1-6). IEEE.)

End of the documents