Preprocessing Pipeline

hreeee@yonsei.ac.kr
강사 백혜림

Subword Segmentation

단어보다 더 작은 의미 단위: Subword

- 많은 언어들에서, 단어는 더 작은 의미 단위들이 모여 구성됨

언어	단어	조합
영어	Concentrate	con(=together) + centr(=center) + ate(=make)
한국어	집중(集中)	集(모을 집) + 中(가운데 중)

- 따라서 이러한 작은 의미 단위로 분절할 수 있다면 좋을 것

- 하지만 이를 위해선 언어별 subword 사전이 존재해야 할 것

단어보다더작은의미단위: Subword

- 기계에 많은 단어를 학습하면 세상의 모든 단어를 알 수 있을까?
- 만약, 기계가 모르는 단어가 등장하면 그 단어를 단어 집합에 없는 단어란 의미에서 OOV(Out-of-Vocabulary) 또는 UNK(Unknown Token)이라 표현한다.
- 모르는 단어로 인해 문제를 푸는 것이 까다로워지는 상황을 OOV문제라 한다.

단어보다더작은의미단위: Subword

- 서브워드 분리(Subword segmenation)작업은 하나의 단어는 더 작은 단위의 의미있는 여러 서브워드들 (ex) birthplace = birth + place)의 조합으로 구성된 경우가 많기 때문에 하나의 단어를 여러 Subword 로 분리해서 단어를 인코딩 및 임베딩하겠다는 의도를 가진 전처리 작업
- Preview와 predict를 보면 접두어인 pre가 공통이다.
- 컴퓨터도 이 두 단어를 따로 볼게 아니라, pre+view와 pre+dict로 본다면 학습을 더 잘 할 수 있겠죠?

단어보다더작은의미단위: Subword

- 이를 통해 OOV나 희귀단어, 신조어 같은 문제들을 완화
- 실제로 언어의 특성에 따라 영어권 언어나 한국어는 서브워드 분리를 시도했을 때, 어느정도 의미있는 단위로 나누는 것이 가능
- 이런 작업을 하는 토크나이저를 **서브워드 토크나이저(Subword tokenizer)로** 명명

단어보다더작은의미단위: Subword

- OOV문제를 완화하는 대표적인 서브워드 분리 알고리즘인 BPE(Byte Pair Encoding)알고리즘을 소개
- 실무에 사용할 수 있도록 구현한 센텐스피스(Sentencepiece)를 소개

Byte Pair Encoding (BPE) 알고리즘

- 압축 알고리즘을 활용하여 subword segmentation을 적용 [Sennrich at el., 2015]

- 학습 코퍼스를 활용하여 BPE 모델을 학습 후, 학습/테스트 코퍼스에 적용

- 장점:
  - 희소성을 통계에 기반하여 효과적으로 낮출 수 있다.
  - 언어별 특성에 대한 정보 없이, 더 작은 의미 단위로 분절 할 수 있다.
  - OoV를 없앨 수 있다. (seen character로만 구성될 경우)

- 단점:
  - 학습 데이터 별로 BPE 모델도 생성됨

BPE Training & Applying

- Training
  - Ⓒ 단어 사전 생성 (빈도 포함)
  - Ⓒ Character 단위로 분절 후, pair 별 빈도 카운트
  - Ⓒ 최빈도 pair를 골라, merge 수행
  - ④ Pair 별 빈도 카운트 업데이트
  - ⑤ 3 번 과정 반복

- Applying
  - Ⓒ 각 단어를 character 단위로 분절
  - Ⓒ 단어 내에서 '학습 과정에서 merge에 활용된 pair의 순서대로' merge 수행

Training Example

vocab {'l o w </w>': 5, 'l o w e r </w>': 2, 'n e w e s t </w>': 6, 'w i d e s t </w>': 3}
pairs (l, o): 7, (o, w): 7, (w, </w>): 5, (w, e): 8, (e, r): 2, (r, </w>): 2, (n, e): 6, (e, w): 6, (e, s): 9, (s, t): 9, (t, </w>): 9, (w, i): 3, (i, d): 3, (d, e): 3
best pair ('e', 's') 9

vocab {'l o w </w>': 5, 'l o w e r </w>': 2, 'n e w **es** t </w>': 6, 'w i d **es** t </w>': 3}
pairs (l, o): 7, (o, w): 7, (w, </w>): 5, (w, e): 2, (e, r): 2, (r, </w>): 2, (n, e): 6, (e, w): 6, (w, es): 6, (es, t): 9, (t, </w>): 9, (w, i): 3, (i, d): 3, (d, es): 3
best pair ('es', 't') 9

vocab {'l o w </w>': 5, 'l o w e r </w>': 2, 'n e w **est** </w>': 6, 'w i d **est** </w>': 3}
pairs (l, o): 7, (o, w): 7, (w, </w>): 5, (w, e): 2, (e, r): 2, (r, </w>): 2, (n, e): 6, (e, w): 6, (w, est): 6, (est, </w>): 9, (w, i): 3, (i, d): 3, (d, est): 3
best pair ('est', '</w>') 9

vocab {'l o w </w>': 5, 'l o w e r </w>': 2, 'n e w **est</w>**': 6, 'w i d **est</w>**': 3}
pairs (l, o): 7, (o, w): 7, (w, </w>): 5, (w, e): 2, (e, r): 2, (r, </w>): 2, (n, e): 6, (e, w): 6, (w, est</w>): 6, (w, i): 3, (i, d): 3, (d, est</w>): 3
best pair ('l', 'o') 7

vocab {'**lo** w </w>': 5, '**lo** w e r </w>': 2, 'n e w est</w>': 6, 'w i d est</w>': 3}
pairs (lo, w): 7, (w, </w>): 5, (w, e): 2, (e, r): 2, (r, </w>): 2, (n, e): 6, (e, w): 6, (w, est</w>): 6, (w, i): 3, (i, d): 3, (d, est</w>): 3
best pair ('lo', 'w') 7

Training Example

vocab {'**low** </w>': 5, '**low** e r </w>': 2, 'n e w est</w>': 6, 'w i d est</w>': 3}
pairs (low, </w>): 5, (low, e): 2, (e, r): 2, (r, </w>): 2, (n, e): 6, (e, w): 6, (w, est</w>): 6, (w, i): 3, (i, d): 3, (d, est</w>): 3
best pair ('n', 'e') 6

vocab {'low </w>': 5, 'low e r </w>': 2, '**ne** w est</w>': 6, 'w i d est</w>': 3}
pairs (low, </w>): 5, (low, e): 2, (e, r): 2, (r, </w>): 2, (ne, w): 6, (w, est</w>): 6, (w, i): 3, (i, d): 3, (d, est</w>): 3
best pair ('ne', 'w') 6

vocab {'low </w>': 5, 'low e r </w>': 2, '**new** est</w>': 6, 'w i d est</w>': 3}
pairs (low, </w>): 5, (low, e): 2, (e, r): 2, (r, </w>): 2, (new, est</w>): 6, (w, i): 3, (i, d): 3, (d, est</w>): 3
best pair ('new', 'est</w>') 6

vocab {'low </w>': 5, 'low e r </w>': 2, '**newest</w>**': 6, 'w i d est</w>': 3}
pairs (low, </w>): 5, (low, e): 2, (e, r): 2, (r, </w>): 2, (w, i): 3, (i, d): 3, (d, est</w>): 3
best pair ('low', '</w>') 5

vocab {'**low</w>**': 5, 'low e r </w>': 2, 'newest</w>': 6, 'w i d est</w>': 3}
pairs (low, e): 2, (e, r): 2, (r, </w>): 2, (w, i): 3, (i, d): 3, (d, est</w>): 3
best pair ('w', 'i') 3

vocab {'low</w>': 5, 'low e r </w>': 2, 'newest</w>': 6, '**wi** d est</w>': 3}

Segmentation Example

lastest news

1) l a t e s t </w> n e w s </w>
2) l a t es t </w> n e w s </w>
3) l a t est </w> n e w s </w>
4) l a t est</w> n e w s </w>
5) l a t est</w> ne w s </w>
6) l a t est</w> new s </w>

applicable pairs in order

1) ('e', 's')
2) ('es', 't')
3) ('est', '</w>')
4) ('l', 'o')
5) ('lo', 'w')
6) ('n', 'e')
7) ('ne', 'w')
8) ('new', 'est</w>')
9) ('low', '</w>')
10) ('w', 'i')

Segmentation Example

코랩으로 실습을 해봅시다!

Subword Segmentation Modules

- Subword-nmt
  - https://github.com/rsennrich/subword-nmt
- WordPiece
  - Upgrade BPE version. Currently unavailable…?
- SentencePiece
  - https://github.com/google/sentencepiece

Wordpiece Model

- WordPiece Model은 BPE의 변형 알고리즘. 이하 WPM
- WPM은 BPE가 빈도수에 기반하여 가장 많이 등장한 쌍을 병합하는 것과 달리, 병합되었을 때, 코퍼스의 우도를 가장 높이는 쌍을 병합함.

WPM을 수행하기 이전의 문장: Jet makers feud over seat width with big orders at stake
WPM을 수행한 결과(wordpieces): _J et _makers _fe ud _over _seat _width _with _big _orders _at _stake

이 알고리즘은 최신 딥 러닝 모델 BERT를 훈련하기 위해서 사용되기도 하였습니다.

OoV가 미치는 영향

- 입력 데이터에 OoV가 발생할 경우, <UNK> 토큰으로 치환하여 모델에 입력
  - e.g. 나는 학교에 가서 밥을 먹었다. → 나 는 <UNK> 에 가 서 <UNK> 을 먹 었 다 .
- 특히, 이전 단어들을 기반으로 다음 단어를 예측하는 task에서 치명적
  - e.g. Natural Language Generation
- 어쨌든 모르는 단어지만, 알고있는 subword들을 통해 의미를 유추해볼 수 있음
  - e.g. 버카충

Summary

- BPE 압축 알고리즘을 통해 통계적으로 더 작은 의미 단위(subword)로 분절 수행
- BPE를 통해 OoV를 없앨 수 있으며, 이는 성능상 매우 큰 이점으로 작용
- 한국어의 경우
  - 띄어쓰기가 제멋대로인 경우가 많으므로,
    normalization 없이 바로 subword segmentation을 적용하는 것은 위험
  - 따라서 형태소 분석기를 통한 tokenization을 진행한 이후,
    subword segmentaion을 적용하는 것을 권장

SubwordTextEncoder

IMDB 리뷰 토큰화하기

Appendix:
Align Parallel Corpus

Why we need

- Parallel Corpus?

English	Korean
I love to go to school.	나는 학교에 가는 것을 좋아한다.
I am a doctor.	나는 의사 입니다.

- 주요 수집 대상
  - 뉴스 기사, 드라마/영화 자막

- 대부분의 경우, 문서 단위의 matching은 되어 있지만, 문장 단위는 되어 있지 않음

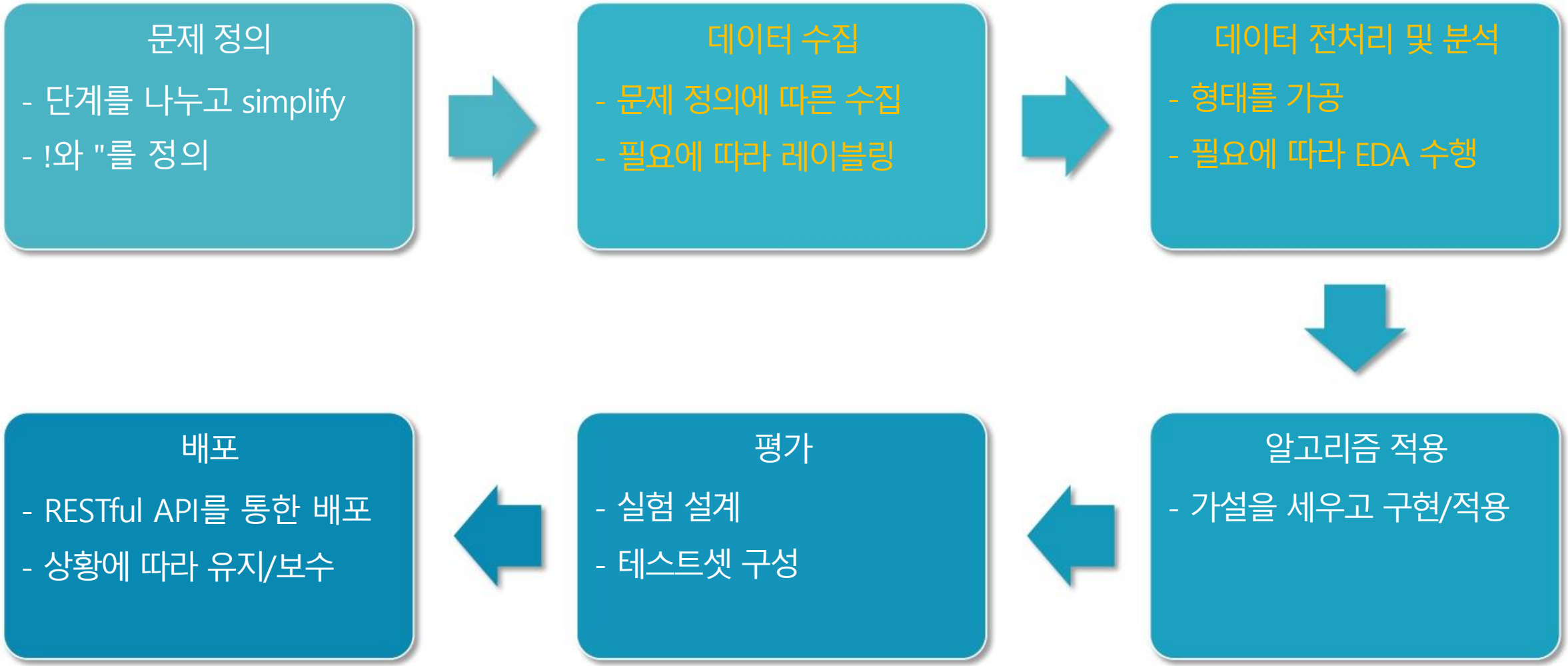
TIP: 전처리의 중요성

전처리의 중요성

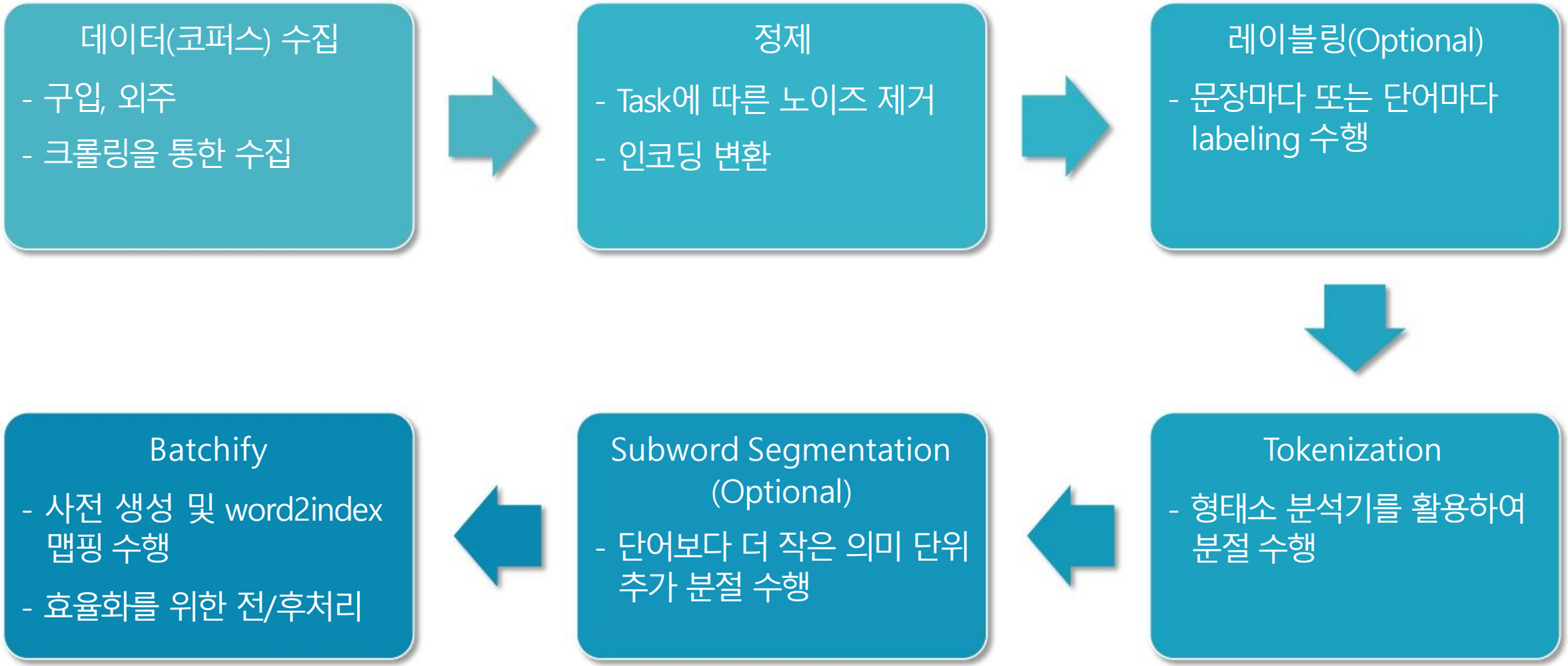
- 오픈 소스 문화의 확산
- 데이터가 더 큰 자산
- 반복되고 지치는 업무지만, 소홀히 하면 안됨
  - 이상적인 팀 구성

Preprocessing Summary

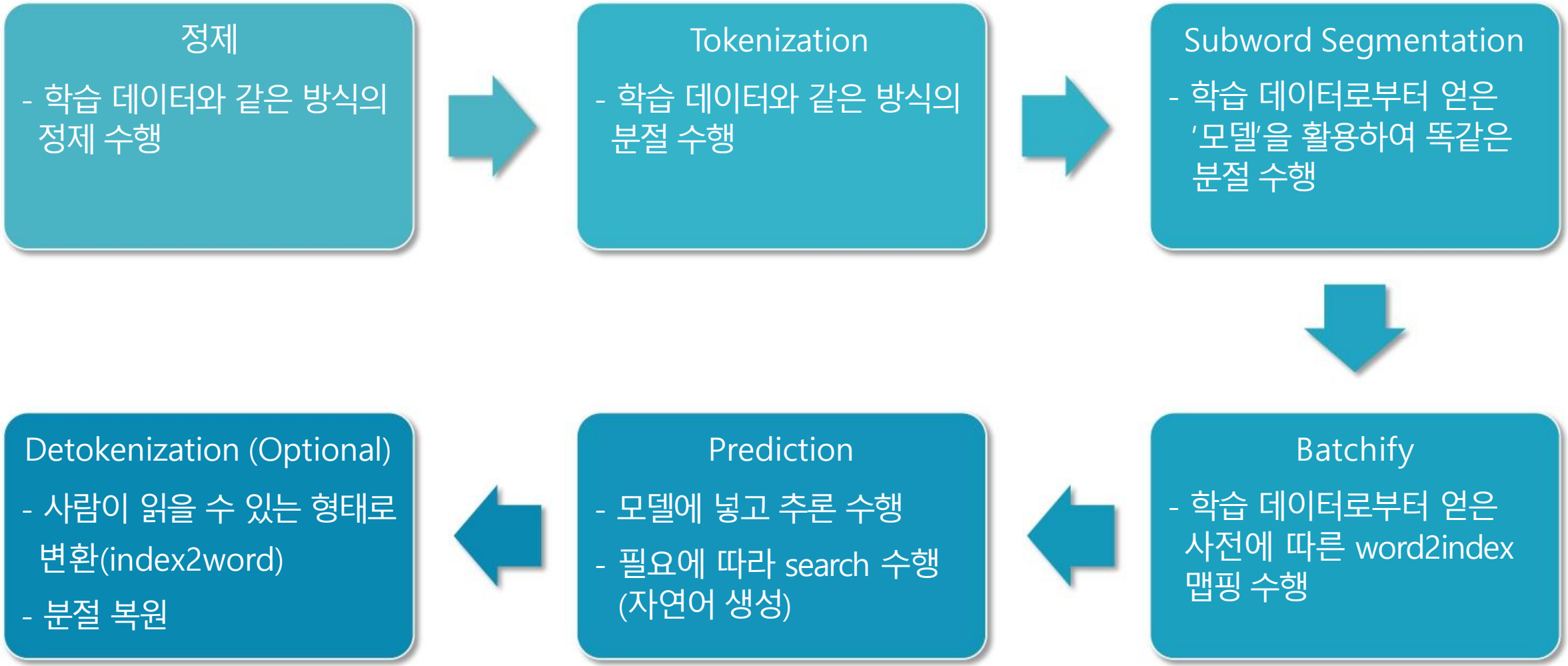
NLP Project Workflow



Preprocessing Workflow



Service Pipeline



Summary

- 정제
  - Task와 언어 및 도메인에 따른 특성
    - 풀고자 하는 문제의 특성에 따라 전처리 전략이 다름
  - 끝이 없는 과정
    - 노력과 품질 사이의 trade-off
    - Sweet spot을 찾아야함

- 분절
  - 한국어의 경우 띄어쓰기 normalization을 위해 형태소 분석기 활용이 필요
  - Subword segmentation을 통해 좀 더 잘게 분절 할 수 있음

- 모두 비슷한 알고리즘을 사용하고 있으므로, 결국 데이터의 양과 품질이 좌우함
  - 따라서 전처리 과정을 경시해서는 안됨

Word Feature Vectors: Traditional Methods

TF-IDF

- 텍스트 마이닝(Text Mining)에서 중요하게 사용
- 어떤 단어 w가 문서 d 내에서 얼마나 중요한지 나타내는 수치
- TF(Term Frequency)
  - 단어의 문서 내에 출현한 횟수
  - 숫자가 클수록 문서 내에서 중요한 단어
  - 하지만, **'the'와 같은 단어도 TF값이 매우 클 것**
- IDF(Inverse Document Frequency)
  - 그 단어가 출현한 문서의 숫자의 역수(inverse)
  - 값이 클수록 'the'와 같이 일반적으로 많이 쓰이는 단어

$$\text{TF-IDF}(w, d) = \frac{\text{TF}(w, d)}{\text{DF}(w)}$$

TF-IDF를 feature로 사용할 수 없을까?

- TF-IDF는 문서에서 해당 단어가 얼마나 중요한지 수치화
- 중요한 문서가 비슷한 단어들은 비슷한 의미를 지닐까?
- 각 문서에서의 중요도를 feature로 삼아서 vector를 만든다면?

TF-IDF Matrix

- 단어의 각 문서(문장, 주제)별 TF-IDF 수치를 vector화
  - Row: 단어
  - Column: 문서

예제: 각 단어별 주제에 대한 TF-IDF 수치

단어	정치	경제	사회	생활	세계	연예	스포츠
문재인	높음	높음	높음	낮음	중간	낮음	낮음
BTS	낮음	낮음	낮음	낮음	높음	높음	낮음
류현진	낮음	낮음	낮음	낮음	높음	중간	높음
날씨	낮음	높음	중간	높음	낮음	낮음	높음
주식	높음	높음	중간	낮음	높음	낮음	낮음
버핏	낮음	높음	낮음	낮음	높음	낮음	낮음

Based on Context Window (Co-occurrence)

- 함께 나타나는 단어들을 활용

- 가정:
  - 의미가 비슷한 단어라면 **쓰임새가 비슷**할 것
  - 쓰임새가 비슷하기 때문에, 비슷한 문장 안에서 **비슷한 역할**로 사용될 것
  - **따라서 함께 나타나는 단어들이 유사**할 것

- Context Window를 사용하여 windowing을 실행
  - window의 크기라는 hyper-parameter 추가
  - **적절한 window 크기**를 정하는 것이 중요

Example

각 단어별 context window 내에 함께 나타난 빈도

	문재인	박근혜	이명박	BTS	싸이	방탄	주식	KOSPI	양적완화
문재인		높음	높음	낮음	낮음	낮음	높음	높음	낮음
박근혜	높음		높음	낮음	중간	낮음	높음	높음	낮음
이명박	높음	높음		낮음	낮음	낮음	높음	높음	중간
BTS	낮음	낮음	낮음		높음	높음	중간	낮음	낮음
싸이	낮음	낮음	낮음	높음		높음	중간	낮음	낮음
방탄	낮음	낮음	낮음	높음	높음		낮음	낮음	낮음
주식	높음	높음	높음	중간	중간	낮음		높음	높음
KOSPI	높음	높음	높음	낮음	낮음	낮음	높음		높음
양적완화	낮음	낮음	중간	낮음	낮음	낮음	높음	높음	

주요 단어만 feature로 활용하는 것도 한 방법

Summary

- Thesaurus 기반 방식에 비해 코퍼스(or 도메인) 특화된 표현 가능
- 여전히 sparse한 vector로 표현됨
  - PCA를 통해 차원 축소를 하는 것도 한 방법

Similarity Metrics

Manhattan Distance (L1distance)

$$\mathrm{d_{L1}}(w, v) = \sum_{i=1}^{d} |w_i - v_i|, \text{ where } w, v \in \mathbb{R}^d.$$

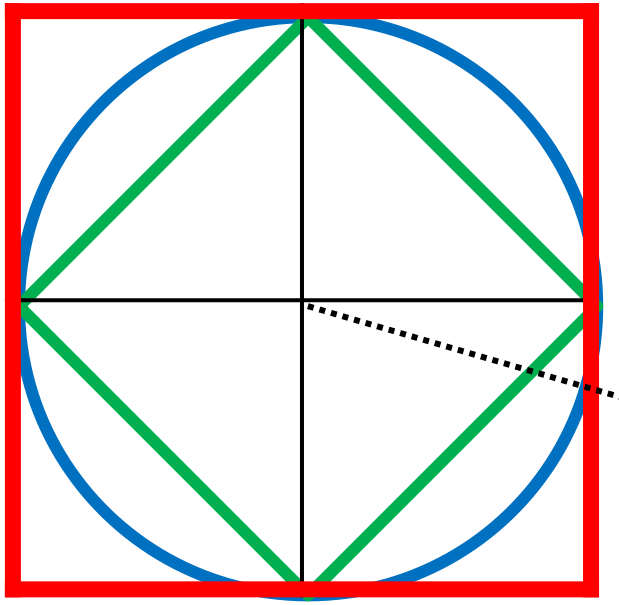
Euclidean Distance (L2 distance)

$$\mathrm{d_{L2}}(w, v) = \sqrt{\sum_{i=1}^{d} (w_i - v_i)^2}, \text{ where } w, v \in \mathbb{R}^d.$$

Infinity Norm

$$d_\infty(w, v) = \max(|w_1 - v_1|, |w_2 - v_2|, \cdots, |w_d - v_d|), \text{ where } w, v \in \mathbb{R}^d$$

L1, L2 andInfinity



Green	L1 Norm
Blue	L2 Norm
Red	Infinity Norm

Cosine Similarity

$$\text{sim}_{\cos}(w, v) = \overbrace{\frac{w \cdot v}{|w||v|}}^{\text{dot product}} = \overbrace{\frac{w}{|w|}}^{\text{unit vector}} \cdot \frac{v}{|v|}$$

$$= \frac{\sum_{i=1}^{d} w_i v_i}{\sqrt{\sum_{i=1}^{d} w_i^2} \sqrt{\sum_{i=1}^{d} v_i^2}}$$

$$\text{where } w, v \in \mathbb{R}^d$$

Summary

- L1, L2 Norm과 Infinity Norm은 강조하고자 하는 것에 따라 사용

- Cosine Similarity는 벡터의 방향을 중요시 함
  - Feature vector의 각 차원의 상대적인 크기가 중요할 때 사용

Word Embedding

Word Embedding

• 자연어를 컴퓨터가 이해하고 효율적으로 처리하게 하기 위해서는 컴퓨터가 이해할 수 있도록 자연어를 적절히 변환할 필요가 있다.

•단어를 표현하는 방법에 따라서 자연어 처리의 성능이 크게 달라지기 때문에 이에 대한 많은 연구가 있었고 여러가지 방버들이 있다!

- 단어의 의미를 벡터화시킬 수 있는 Word2Vec과 Glove가 많이 사용
- 전통적 방법의 한계를 개선시킨 워드 임베딩(Word Embedding)방법론에 대해 배워보자!

희소표현 (Sparse Representation)

• 원-핫 인코딩을 통해서 나온 원-핫 벡터들은 표현하고자 하는 단어의 인덱스의 값만 1이고, 나머지 인덱스에는 전부 0으로 표현되는 벡터 표현 방법

- 벡터 또는 행렬(matrix)의 값이 대부분이 0으로 표현되는 방법을 희소 표현(sparse representation)

• 문제점

단어의 개수가 늘어나면 벡터의 차원이 한없이 커진다는 점

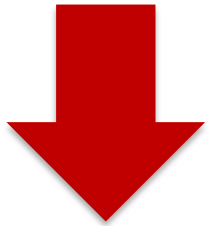
Ex) 강아지 = [0 0 0 0 1 0 0 0 0 0 0 0 ... 중략 ... 0] # 이 때 1 뒤의 0의 수는 9995개.

밀집표현 (Dense Representation)

- 밀집 표현은 벡터의 차원을 단어 집합의 크기로 상정하지 않습니다. 사용자가 설정한 값으로 모든 단어의 벡터 표현의 차원을 맞춥니다

- 이 과정에서 더 이상 0과 1만 가진 값이 아니라 실수값을 가지게 됩니다

Ex) 강아지 = [0 0 0 0 1 0 0 0 0 0 0 0 ... 중략 ... 0] # 이 때 1 뒤의 0의 수는 9995개.



Ex) 강아지 = [0.2 1.8 1.1 -2.1 1.1 2.8 ... 중략 ...] # 이 벡터의 차원은 128

워드 임베딩 (Word Embedding)

- 단어를 밀집 벡터(dense vector)의 형태로 표현하는 방법을 **워드 임베딩 (word embedding)**

- 이 밀집 벡터를 워드 임베딩 과정을 통해 나온 결과라고 하여 **임베딩 벡터(embedding vector)**

- 워드 임베딩 방법론으로는 LSA, Word2Vec, FastText, Glove 등이 있다

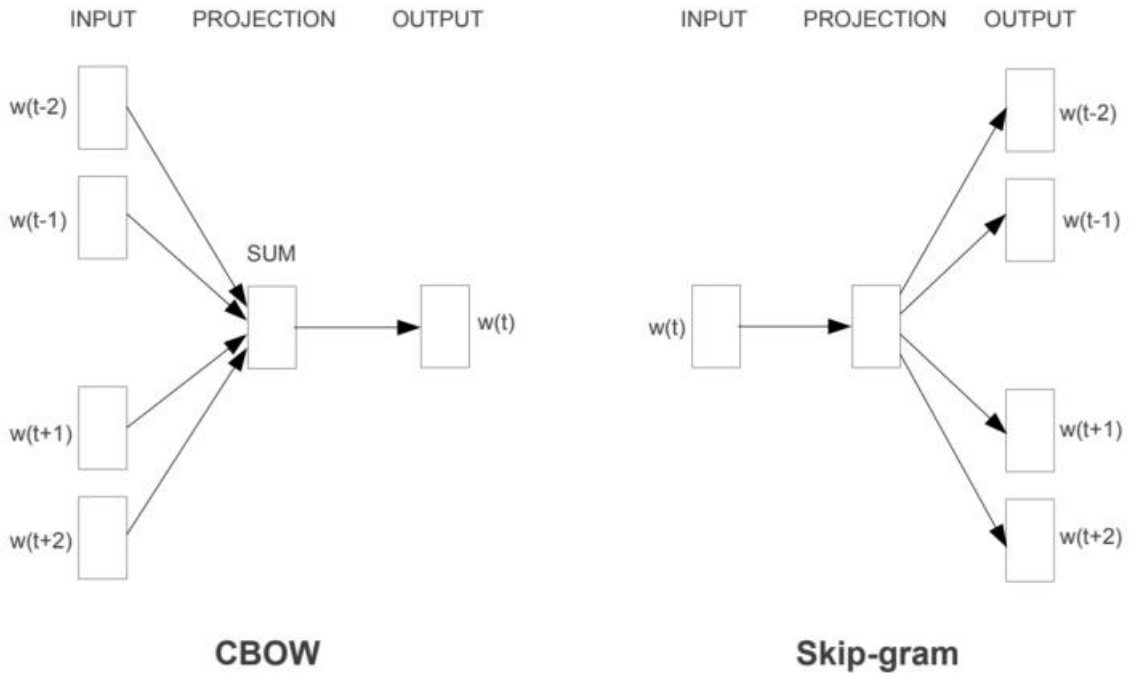
-	원-핫 벡터	임베딩 벡터
차원	고차원(단어 집합의 크기)	저차원
다른 표현	희소 벡터의 일종	밀집 벡터의 일종
표현 방법	수동	훈련 데이터로부터 학습함
값의 타입	1과 0	실수

Word2Vec

Word2Vec [Mikolov et al.2013]

- Objective:
  - 주변(context window)에 **같은 단어가 나타나는 단어일 수록** 비슷한 벡터 값을 가져야 한다.

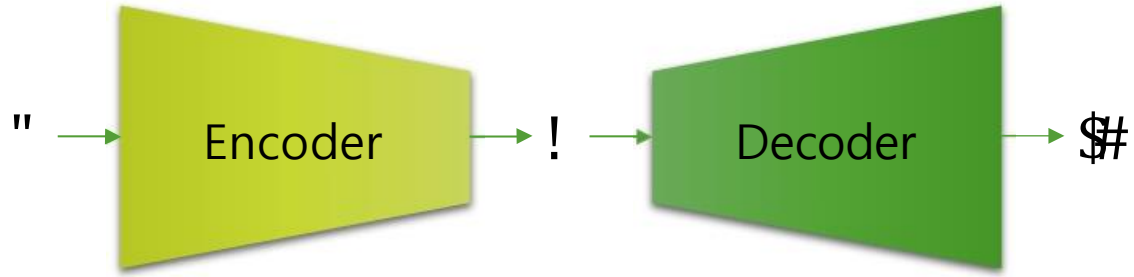
- 문장의 문맥에 따라 정해지는 것이 아님
  - context window의 사이즈에 따라 embedding의 성격이 바뀔 수 있다.



Skip-gram: Basic Concept

- 기본 전략
  - 주변 단어를 예측하도록 하는 과정에서 적절한 단어의 임베딩(정보의 압축)을 할 수 있다.
  - Non-linear activation func.이 없음

- 기본적인 개념은 오토인코더와 굉장히 비슷함
  - y를 성공적으로 예측하기 위해 필요한 정보를 선택/압축



Skip-gram Example (|W|=5)

예식장	은	용궁	예식장	주례	는	문어	아저씨	.	
피아노	는	오징어	예물	은	조개껍데기	.			

Skip-gram Example (|W|=5)

예식장	은	용궁	예식장	주례	는	문어	아저씨	.	
피아노	는	오징어	예물	은	조개껍데기	.			

Skip-gram Example (|W|=5)

예식장	은	용궁	예식장	주례	는	문어	아저씨	.	
피아노	는	오징어	예물	은	조개껍데기	.			

Skip-gram Example (|W|=5)

예식장	은	용궁	예식장	주례	는	문어	아저씨	.	
피아노	는	오징어	예물	은	조개껍데기	.			

Skip-gram Example (|W|=5)

예식장	은	용궁	예식장	주례	는	문어	아저씨	.	
피아노	는	오징어	예물	은	조개껍데기	.			

Skip-gram Example (|W|=5)

예식장	은	용궁	예식장	주례	는	문어	아저씨	.	
피아노	는	오징어	예물	은	조개껍데기	.			

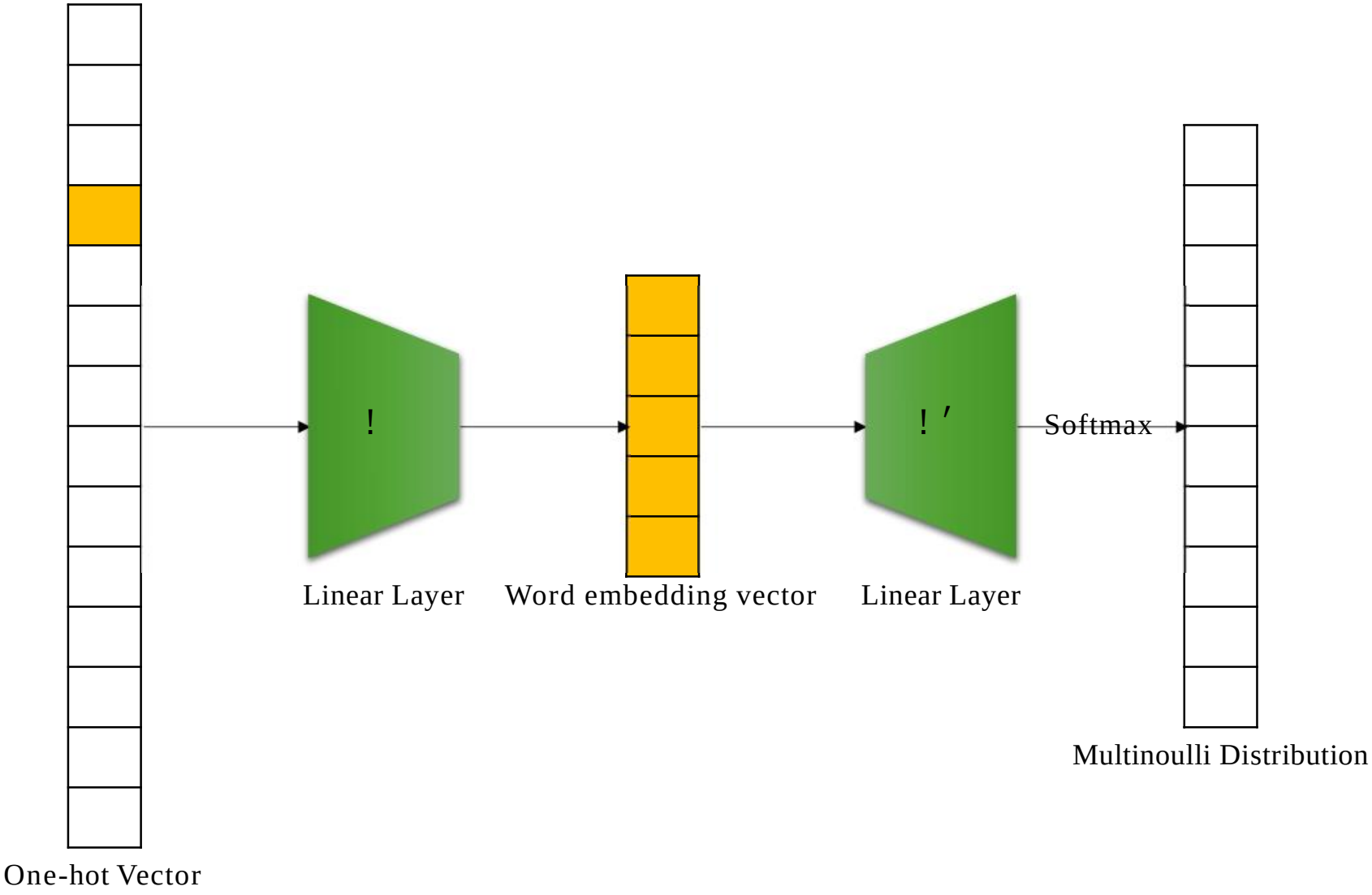
Skip-gram Example (|W|=5)

예식장	은	용궁	예식장	주례	는	문어	아저씨	.	
피아노	는	오징어	예물	은	조개껍데기	.			

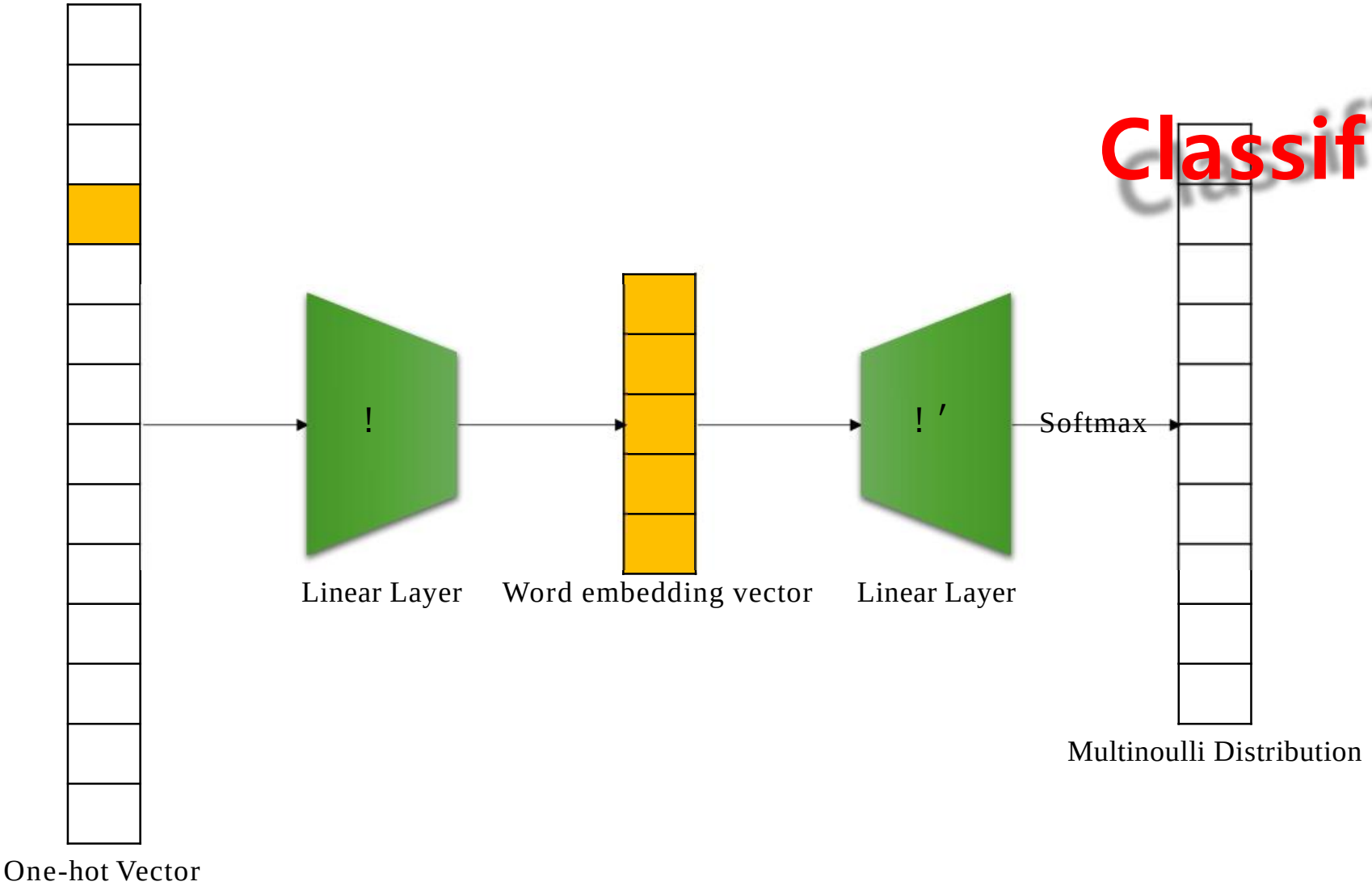
Skip-gram Example (|W|=5)

예식장	은	용궁	예식장	주례	는	문어	아저씨	.	
피아노	는	오징어	예물	은	조개껍데기	.			

Skip-gram


!
!′
Softmax
One-hot Vector
Linear Layer
Word embedding vector
Linear Layer
Multinoulli Distribution

Skip-gram


Classification
!
!′
Softmax
Linear Layer
Word embedding vector
Linear Layer
One-hot Vector
Multinoulli Distribution

Skip-gram

- 장점(at that time):
  - 쉽다.
  - 빠르다.
  - 비교적 정확한 벡터를 구할 수 있다.

- 단점(currently):
  - 헌데 느리다.
  - 출현 빈도가 적은 단어일 경우 벡터가 정확하지 않다.

코랩으로 실습을 해봅시다!

GloVe

GloVe

- Global Vectors for Word Representation
  [Pennington et al.,2014]

- 단어 x와 윈도우 내에 함께 출현한 단어들의 출현 빈도를 맞추도록 훈련

GloVe

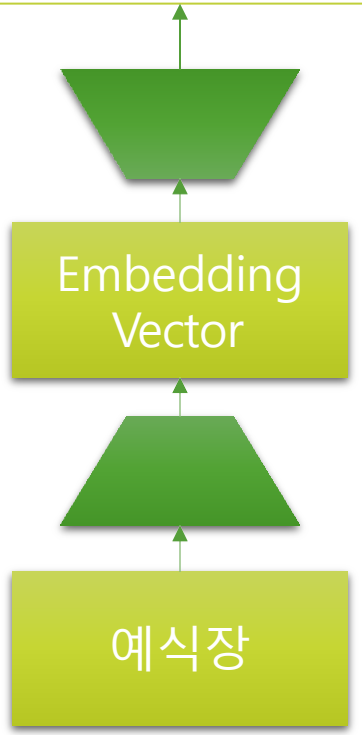
- Global Vectors for Word Representation
  [Pennington et al.,2014]

- 단어 x와 윈도우 내에 함께 출현한 단어들의 출현 빈도를 맞추도록 훈련

Regression

GloVe Example

오징어	예물	조개	피아노	은	는	결혼식	웨딩	주례	문어
5	17	2	23	31	27	41	29	34	1



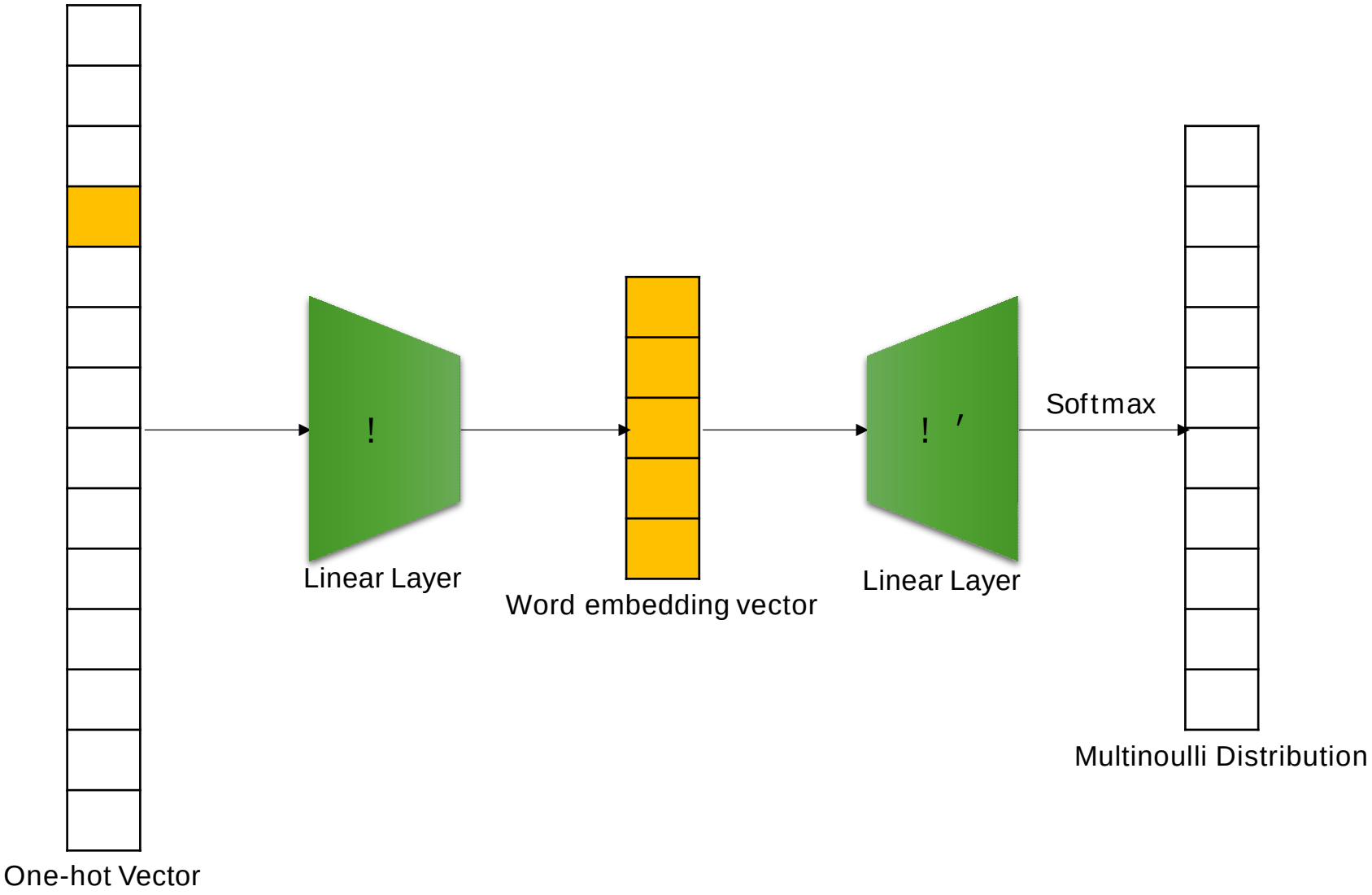
GloVe

- 출현 빈도가 적은 단어에 대해서는 loss의 기여도를 낮춤
  - 따라서 출현 빈도가 적은 단어에 대해 부정확해지는 단점을 보완

- 장점:
  - 더 빠르다
    - 전체 코퍼스에 대해 각 단어 별 co-occurrence를 구한 후, regression을 수행
  - 출현 빈도가 적은 단어도 벡터를 비교적 정확하게 잘 구할 수 있다.

FastText

Review: Skip-gram



FastText: Upgrade Version of Skip-gram

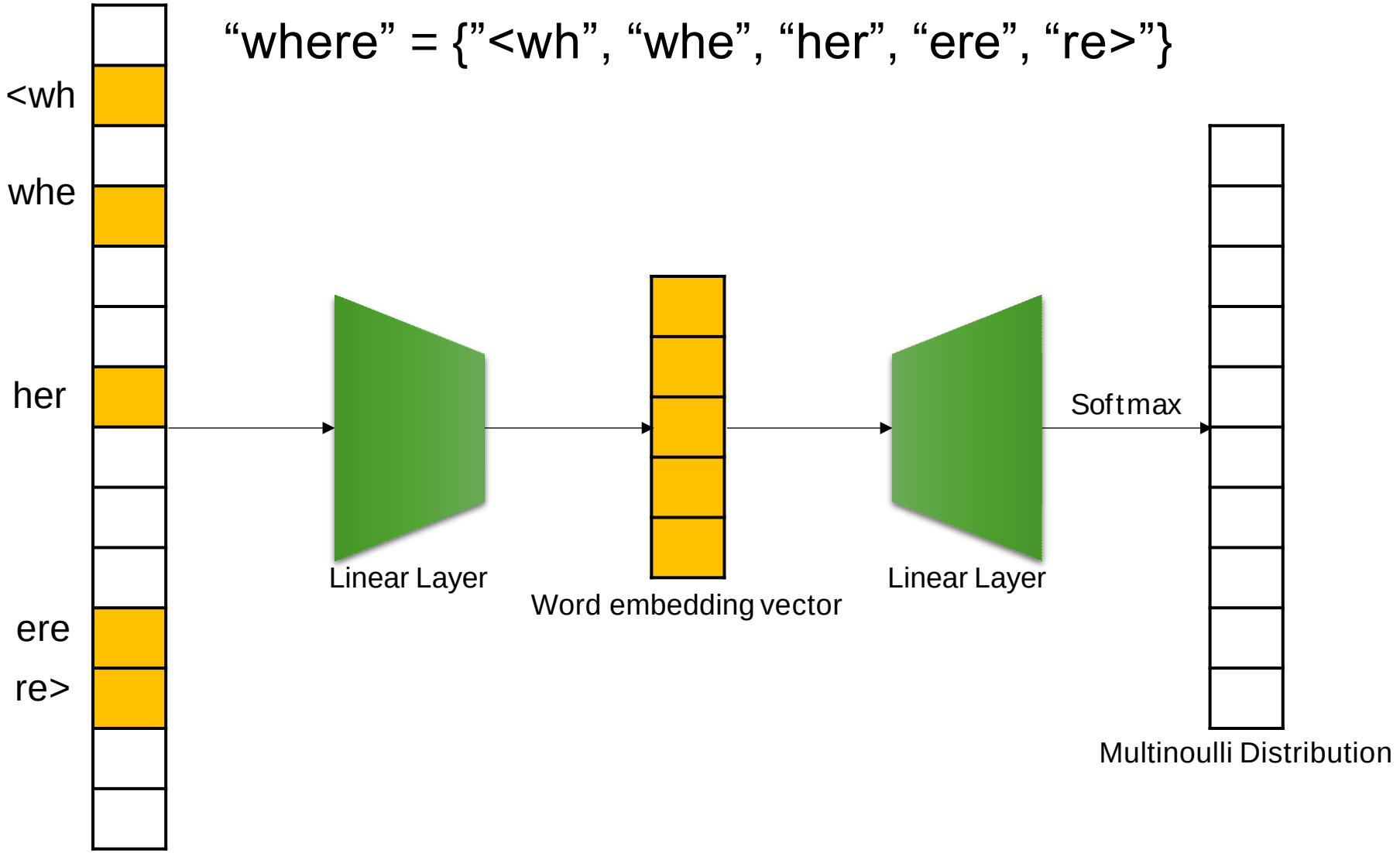
- Enriching Word Vectors with Subword Information [Bojanowski and Grave et al., 2016]

- Motivation:
  - 기존의 word2vec은 저빈도 단어에 대한 학습과 OoV에 대한 대처가 어려웠음

- FastText는 학습시,
  1) 단어를 subword로 나누고,
  2) Skip-gram을 활용하여, 각 subword에 대한 embedding vector에 주변 단어의 context vector를 곱하여 더한다.
  3) 이 값이 최대가 되도록 학습을 수행한다.

- 최종적으로 각 subword에 대한 embedding vector의 합이 word embedding vector가 된다.

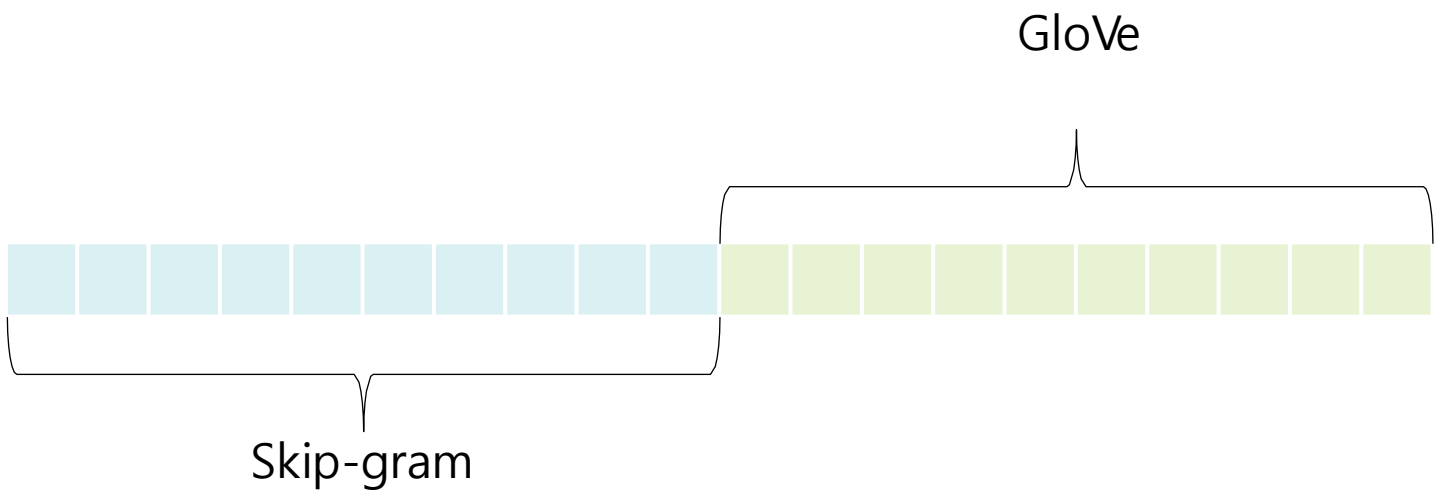
Example: where



Conclusion: Word Embedding

- 딱히 어떤 알고리즘이 더 뛰어나다고는 할 수 없다.
  - 구현이 쉽고 빠른 오픈소스를 사용하는 것이 낫다.

- 두 개의 다른 알고리즘 결과물을 concat하여 사용하기도



Open-source

- Gensim
  - https://radimrehurek.com/gensim/install.html
- GloVe
  - https://github.com/stanfordnlp/GloVe
- FastText
  - https://github.com/facebookresearch/fastText/
  - http://bit.ly/fasttext_win