NLP I

Deep Session 7차시

빅데이터 분석 학회 D&A

CONTENTS.

01. NLP란?

- NLP 이해하기
- NLP Task

02. Preprocessing

- Text Data
- Cleaning
- Normalization

03. Conversion

- Tokenization
- Padding
- Word Embedding

NLP 이해하기

Sequence Data

- 연속적으로 이루어진 데이터로 순서나 위치에 따라 다른 의미를 가짐

  시계열 데이터: 시간에 따른 시퀀스 데이터

  자연어 문장: 위치에 따른 시퀀스 데이터

TRIANGLE INTEGRAL

NLP란?

NLP 이해하기

NLP (Natural Language Processing)

- 자연어 처리 (NLP): Text 데이터를 분석하고 모델링하는 분야
    - 자연어 이해 (NLU, Natural Language Understanding)
    → 주어진 텍스트의 의미 파악 (Text → Meaning)
    - 자연어 생성 (NLG, Natural Language Generation)
    → 주어진 의미에 대한 자연스러운 Text 생성 (Meaning → Text)

NLP란?

NLP 이해하기

NLP Process

1. Preprocessing: 수행하는 Task에 맞는 텍스트 데이터 수집 & 전처리
2. Tokenization: 데이터를 숫자로 표현하기 위해 적당한 기준으로 자르기
3. Word Embedding: 토큰화 된 데이터를 컴퓨터가 이해할 수 있도록 숫자로 변형
4. Modeling: Embedding 된 데이터를 가지고 Modeling 진행

NLP Task

NLP의 여러 가지 Task

- 감정 분석 (Sentiment Analysis): 텍스트에서 감정, 감성, 의견 등을 자동으로 인식하고 분류하는 task

  → 소비자 피드백, 온라인 리뷰, 소셜 미디어 게시물 등의 반응 분석

- 요약 (Summarization): 주어진 텍스트의 핵심 내용을 간결하게 전달하기 위한 task

  1) 추출적 요약 (Extractive Summarization): 원문에서 가장 중요한 문장 or 구문을 직접 선택하여 요약

  2) 생성적 요약 (Abstractive Summarization): 원문의 핵심 내용을 새로운 문장으로 표현하여 요약

- 기계 번역 (Machine Translation): 한 언어로 작성된 텍스트를 다른 언어로 번역하는 task

  → ex) 구글 번역기, 네이버 파파고, DeepL 등

- 질의 응답 (Question Answering): 자연어 질문에 대한 정확한 답변을 찾거나 생성하는 task

NLP Task

NLP의 여러 가지 Task

- 분류 (Classification): 텍스트를 사전에 정의된 카테고리 또는 label에 할당하는 task

  → 텍스트 데이터의 구조와 정보를 이해하고, 의미 있는 카테고리로 구분함으로써 분석과 처리를 용이하게 하는 것이 목표

  ex) 감정 분류, 스팸 메일 분류, 토픽 분류, 의도 분류 등

- 품사 태깅 (POS Tagging): 텍스트 내의 각 단어를 적절한 품사 정보에 할당하는 task

  → 텍스트의 구조와 문법적 특정을 이해하고, 이를 바탕으로 다른 NLP 작업의 성능을 향상시키는 것이 목표

- 챗봇 (ChatBot): 자연어로 된 사용자 질문에 응답하는 인공지능 대화 시스템을 만드는 task

  → 사람과 컴퓨터 간의 자연스러운 커뮤니케이션을 가능하게 하여 다양한 업무와 서비스를 효율적으로 지원하는 것이 목표

Preprocessing

Text Data

텍스트 데이터 (Text Data)

- 자연어로 구성된 문자, 문장, 문단 등의 정보를 포함하는 데이터 형태
- 인간이 사용하는 언어로 표현되는 정보를 다룸
- 다양한 형태와 출처를 가짐

ex) 웹페이지, SNS, 이메일, 뉴스기사, 리뷰, 채팅 기록 등

텍스트 데이터의 특징

- 비정형 데이터: 정형화된 DB 스키마 or 표 형식의 구조가 없음
- 노이즈와 불규칙성: 오타, 문법 오류, 구어체 등과 같은 노이즈 포함
- 다양한 표현: 다양한 언어, 방언, 유행어, 전문 용어 등 다양한 형태의 표현 사용
- 애매한 의미: 동음이의어, 다의어, 비유, 은유와 같은 애매한 의미를 가진 요소 포함
- 순서 정보: 문맥 이해와 문법 구조를 파악하는데 순서 정보가 필요함

텍스트 데이터는
전처리가 중요

Preprocessing

Cleaning

데이터 정제 (Data Cleaning)

- 텍스트 데이터에서 불필요한 요소를 제거 or 수정하는 과정
- 데이터의 품질을 향상시키고, 분석 및 모델링에 영향을 미칠 수 있는 오류와 노이즈 제거

1. 불필요한 문자 제거: 텍스트 데이터에서 분석에 불필요한 문자 제거 ex) 숫자, 특수 문자, 이모티콘 등
2. 결측치 처리: 누락된 데이터를 처리
3. 이상치 제거: 데이터에서 정상 범위를 벗어난 값이나 패턴을 제거 ex) 오타 및 문법 오류, 특이 단어, 무작위 문자열 등

Cleaning 예시

- 예시: “This is 1 amazing example!! A person’s watching a movie. They’re not? Watching the new show”

불필요한 문자 제거 → “This is amazing example A persons watching a movie Theyre not Watching the new show”

Preprocessing

Normalization

데이터 정규화 (Data Normalization)

- 텍스트 데이터를 일관된 형식으로 변환하는 과정
- 단어의 다양한 변형을 통일하고, 모델의 성능을 향상시킴

1. 소문자 변환: 모든 문자를 소문자로 변환하여 단어의 변형을 최소화함 ex) Hello → hello
2. 불용어 제거: 불용어를 제거해 텍스트의 크기를 줄이고 중요한 정보에 집중함

   → 불용어 (출현 빈도가 높지만, 문장의 의미 분석에 영향이 적은 단어) ex) “a”, “an”, “the”, “and” 등
3. Stemming & Lemmatization: 단어의 다양한 형태를 그 단어의 기본 형태로 변환하여 텍스트의 일관성 향상
   - Stemming: 언어적 규칙 기반 ex) went → went, leaves → leav
   - Lemmatization: 사전에 기반한 단어의 원형 ex) went → go, leaves → leaf

Preprocessing

Normalization

Normalization 예시

- 예시: “This is amazing example A persons watching a movie Theyre not Watching the new show”

1. 소문자 변환 → “this is amazing example a persons watching a movie theyre not watching the new show”

2. 불용어 제거 → “amazing example persons watching movie not watching new show”

3-1. Stemming 적용 → “amaz example person watch movi not watch new show”

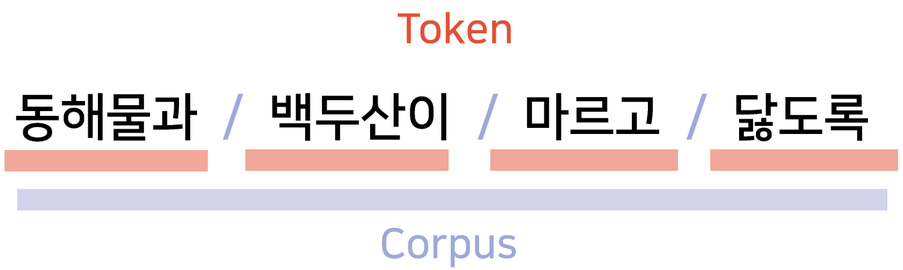
3-2. Lemmatization 적용 → “amaze example person watch movie not watch new show”

Conversion

Tokenization

NLP에 등장하는 용어 정의

- Corpus (말뭉치): 자연어 처리를 위해 수집된 텍스트 데이터들의 집합
- Token (토큰): 텍스트를 나누는 최소 단위로, NLP에서 처리할 수 있는 의미 있는 단위
- Vocabulary (사전): 특정 Corpus에서 사용되는 고유한 토큰들의 집합으로 토큰들의 key와 value를 저장



Vocabulary

key	Value
1	동해물과
2	백두산이
3	마르고
4	닳도록

Conversion

Tokenization

사전에 없는 단어가 등장하면? (OOV: Out-of-Vocabulary)

- 기존에 만들었던 사전에 특정 토큰이 존재하지 않는다면…
- OOV 문제를 해결하기 위해 사전에 정의된 특수한 토큰 (<UNK> or <[UNK]>)으로 대체하여 문제를 해결

  ! But, 특수한 토큰으로 대체해서 처리하면 성능 저하 문제가 발생함

OOV 문제 해결

- 사전에 없는 새로운 단어가 등장하지 않도록 사전을 풍부하게 만들기

  → 말뭉치의 크기를 키워 사전을 풍부하게 할 수 있지만, 모델의 사이즈도 함께 증가해 메모리 문제 발생

Token을 효율적으로 만들기 위한 **Tokenization 방법 선택**이 중요

Conversion

Tokenization

Work Tokenization (단어 기반 토큰화)

- 텍스트를 공백, 구두점, 특수문자 등을 기준으로 분리해 개별 단어로 나누는 방식 (가장 기본적이고 널리 사용됨)

• 장점

1) 직관적이고 이해하기 쉬움

2) 많은 자연어 처리 작업에서 높은 성능을 보임

• 단점

1) 다양한 형태의 단어 변형이나 오타 등으로 인한 OOV 문제 발생 가능

2) 희소성 문제로 인해 데이터 효율성이 떨어질 수 있음

- 예시: “동해물과 백두산이 마르고 닳도록 하느님이 보우하사 우리나라 만세”

→ [“동해물과”, “백두산이”, “마르고”, “닳도록”, “하느님이”, “보우하사”, “우리나라”, “만세”]

Conversion

Tokenization

Subword Tokenization (서브워드 기반 토큰화)

- 텍스트를 의미 있는 단위의 조각으로 분리하는 방식으로 BPE, WordPiece, SentencePiece 등의 알고리즘으로 구현

• 장점

1) OOV 문제를 크게 줄여 모델의 일반화 성능 향상

2) 희소성 문제 완화해 데이터 효율성 증가

3) 다양한 언어 및 도메인에 적용 가능

• 단점

1) 단어 기반 토큰화에 비해 복잡하고 이해하기 어려움

2) 토큰화 과정에서 추가적인 계산이 필요함

- 예시: “동해물과 백두산이 마르고 닳도록 하느님이 보우하사 우리나라 만세”

→ [“동해물”, “과”, “백두산”, “이”, “마르”, “고”, “닳”, “도록”, “하느님”, “이”, “보우”, “하”, “사”, “우리”, “나라”, “만세”]

Conversion

Padding

Padding이란?

- 서로 다른 길이의 문장이나 시퀀스를 모델에 입력하기 전에 동일한 길이로 맞춰주는 과정

- 문장의 시작 or 끝에 패딩을 추가할 수 있음 (모델 구조나 문제 설정에 따라 적절한 위치 선택)

- 최대 시퀀스 길이 결정 후 이보다 긴 문장의 경우에는 일정한 기준에 따라 자르고, 짧은 문장의 경우 패딩 추가함

Padding 방법

- Zero-Padding: 가장 일반적인 패딩 방법 / 길이가 부족한 부분을 0으로 채움

  → 계산과 처리가 편리하고, 실제 데이터와의 구분이 용이해 사용하기 적합함

- Constant Padding: 0 대신 다른 고정된 상수 값으로 부족한 부분을 채움

  → 실제 데이터에 포함되지 않는 값이라는 점이 보장되어야 함

- Custom Padding: 특정 도메인이나 데이터셋에 따라 사용자가 직접 패딩 방식을 정의

  → 특정 텍스트의 패턴 & 도메인 지식 활용하여 의미있는 패딩 값 설정

Conversion

Padding

Padding의 필요성

- 딥러닝 기반의 NLP 모델들은 일반적으로 고정된 길이의 입력을 처리

  → 서로 다른 길이의 입력 데이터를 처리하기 위해 padding을 사용해 모든 문장 or 시퀀스의 길이를 동일하게 맞춤

Padding 유의사항

- Padding 값은 실제 텍스트 데이터에 포함되지 않는 고유한 값으로 설정
- 과도한 패딩은 학습에 부정적이기 때문에 적절한 최대 시퀀스 길이 설정이 중요함

Conversion

Padding

Padding 예시

- 예시 데이터

[[“동해물과”, “백두산이”, “마르고”, “닳도록”, “하느님이”, “보우하사”, “우리나라”, “만세”],
[“무궁화”, “삼천리”, “화려”, “강산”, “대한”, “사람”, “대한으로”, “길이”, “보전하세”]]

→ 최대 길이 시퀀스를 10으로 설정해 Padding 진행

- Padding 후

[[“동해물과”, “백두산이”, “마르고”, “닳도록”, “하느님이”, “보우하사”, “우리나라”, “만세”, “<pad>”, “<pad>”],
[“무궁화”, “삼천리”, “화려”, “강산”, “대한”, “사람”, “대한으로”, “길이”, “보전하세”, “<pad>”]]

Conversion

Word Embedding

Word Embedding이란?

- 단어를 고정된 크기의 벡터로 표현하는 것
- 기계가 텍스트를 이해하고 처리할 수 있도록 도움
- 유사한 단어들은 벡터 공간에서 가깝게 위치하고, 고차원의 데이터를 저차원으로 변환해 연산량이 감소함

단어의 분산 표현과 분포 가설

- 분산 표현 (Distributed Representation): 단어를 고차원이 아닌 저차원의 연속 벡터로 표현하는 방식

  → 단어의 의미를 벡터 공간 상의 근접한 위치에 배치함으로써 단어 간의 유사성 계산 가능

- 분포 가설 (Distribution Hypothesis): '단어의 의미는 주변 단어들에 의해 결정된다'

  → 같은 문맥에서 자주 나타나는 단어들은 비슷한 의미를 가짐

Conversion

Word Embedding

One-hot Encoding

- 각 단어에 대해 고유한 인덱스를 부여하고, 단어 수만큼의 차원을 가진 벡터를 생성

→ 해당 단어의 인덱스 위치의 값은 1, 나머지 위치의 값은 0

["동해물과", "백두산이", "마르고", "닳도록"] → [[1,0,0,0], [0,1,0,0], [0,0,1,0], [0,0,0,1]]

동해물과	1	0	0	0
백두산이	0	1	0	0
마르고	0	0	1	0
닳도록	0	0	0	1

One-hot Encoding의 한계

- 차원의 저주: 사전의 크기가 클수록 벡터의 차원이 커지며, 이로 인해 계산 복잡성이 증가하고 공간 낭비 발생
- 단어 간의 관계 표현 부족: One-hot 벡터는 모든 단어가 직교하므로 단어 간의 의미적 유사성이나 관계를 전혀 표현하지 못함
- 데이터 희소성 (Sparsity): 대부분의 값이 0인 희소 벡터는 계산에 비효율적

Conversion

Word Embedding

Word2Vec

- 분포 가설에 기반한 분산 표현
- 텍스트 데이터의 단어들을 고정된 길이의 벡터로 표현함
- 학습 방법
  - CBOW: 주변 단어들을 이용해 타겟 단어를 예측하는 방식으로 학습
  - Skip-gram: 타겟 단어를 이용해 주변 단어들을 예측하는 방식으로 학습

target

The cat sat on the mat

context context

- 특징
  - 단어 간의 유사성을 기하학적으로 표현할 수 있음
  - 단어 간의 관계를 잘 반영한 벡터를 생성해, 이를 기반으로 한 연산이 가능함 ex) 왕 – 남자 + 여자 = 여왕
  - 단어의 다양한 의미를 구분하지 못하고 하나의 벡터로 표현함 ex) '배'를 신체의 일부, 과일 등 여러 의미로 구분 불가

Conversion

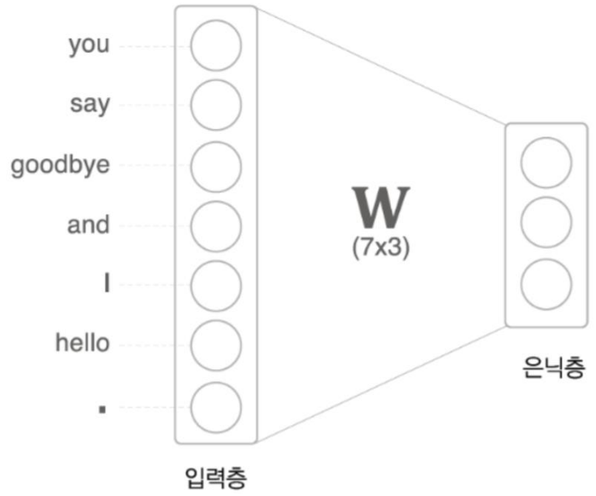
Word Embedding

Word2Vec - CBOW

- 학습 예문

you ? goodbye and I say hello.

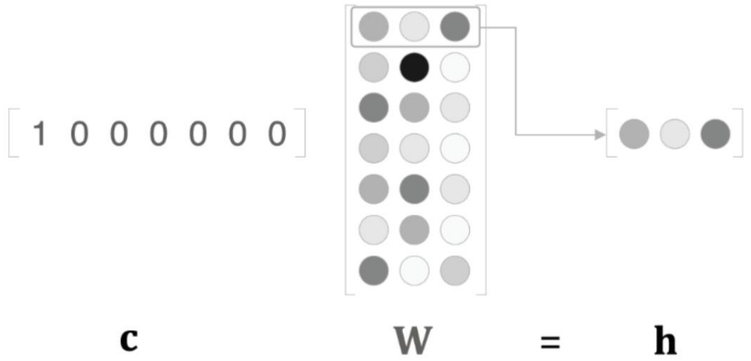
- FC-layer 에 의한 표현 단순화 모형



- Context, Text ID, One-hot vector

단어(텍스트)	단어 ID	원핫 표현
you goodbye	0 2	(1, 0, 0, 0, 0, 0, 0) (0, 0, 1, 0, 0, 0, 0)

- 맥락 벡터 c, 가중치 행렬 W, 은닉 벡터 h

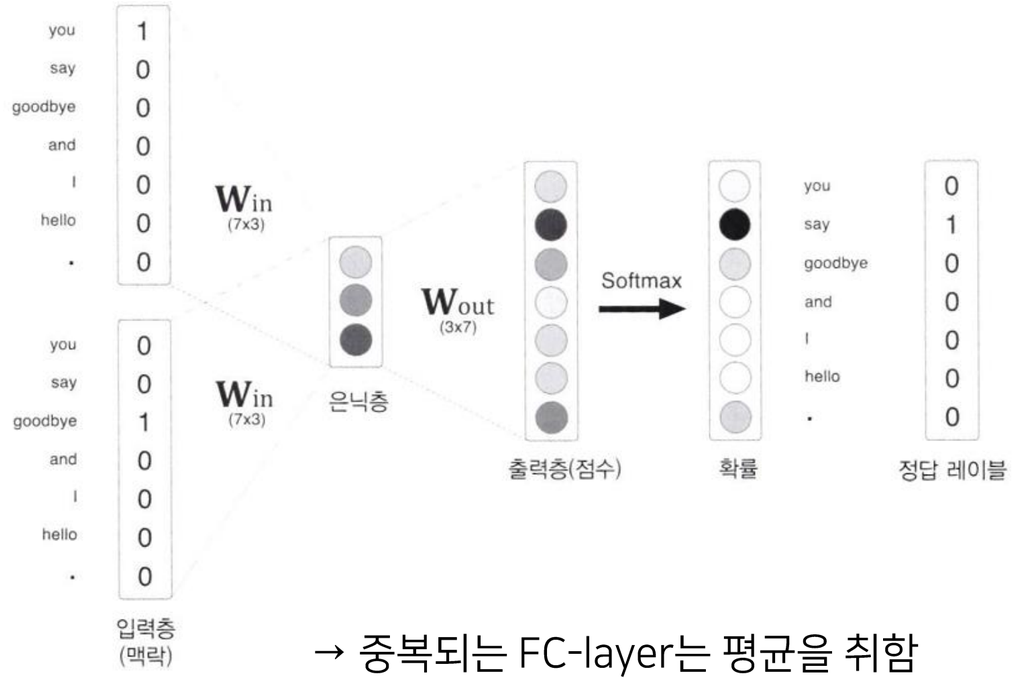


Conversion

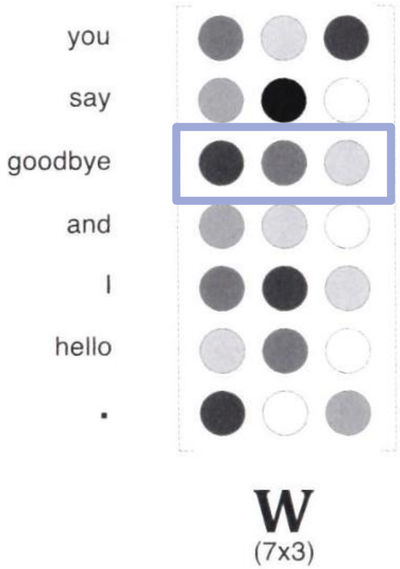
Word Embedding

Word2Vec - CBOW

- CBOW 모델의 신경망 구조



→ 중복되는 FC-layer는 평균을 취함



→ 입력 가중치 행렬의 각 행이 각 단어의 분산 표현

Conversion

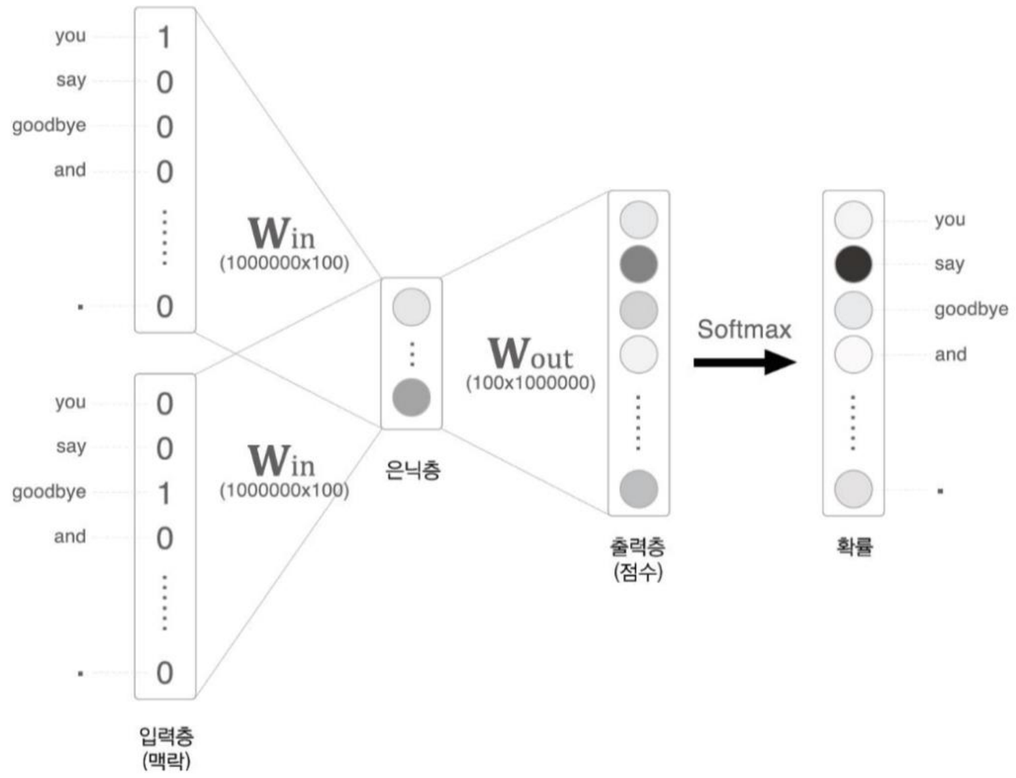
Word Embedding

Word2Vec – CBOW 문제점

- 입력 및 출력층의 크기가 커질수록 계산량이 많아져 병목현상 발생함

  1) 입력층의 One-hot 표현과 가중치 행렬 W_{in} 의 곱 계산

  2) 은닉층과 가중치 행렬 W_{out} 의 곱

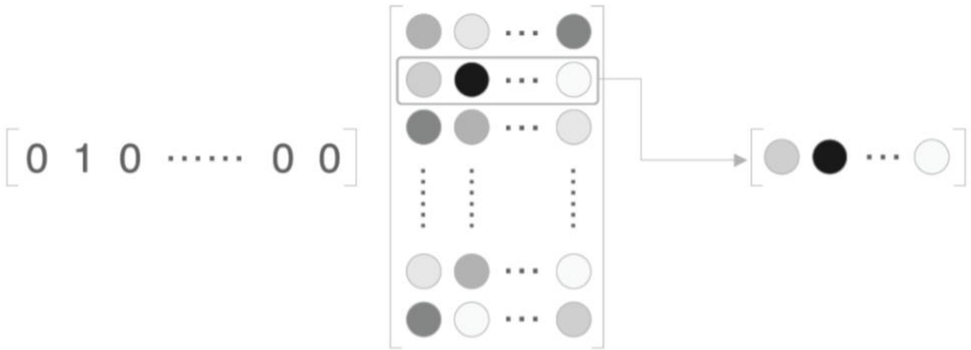


Conversion

Word Embedding

Word2Vec – CBOW 개선하기 1

- 입력층의 One-hot 표현과 가중치 행렬 W_{in} 의 곱 계산



맥락 벡터와 가중치 행렬의 곱 연산을 통해 은닉 벡터를 얻는 과정

= 가중치 행렬에서 맥락 벡터의 인덱스에 해당하는 행 추출

→ 곱 연산이 아닌 행렬에서의 인덱싱을 통해 병목현상 제거

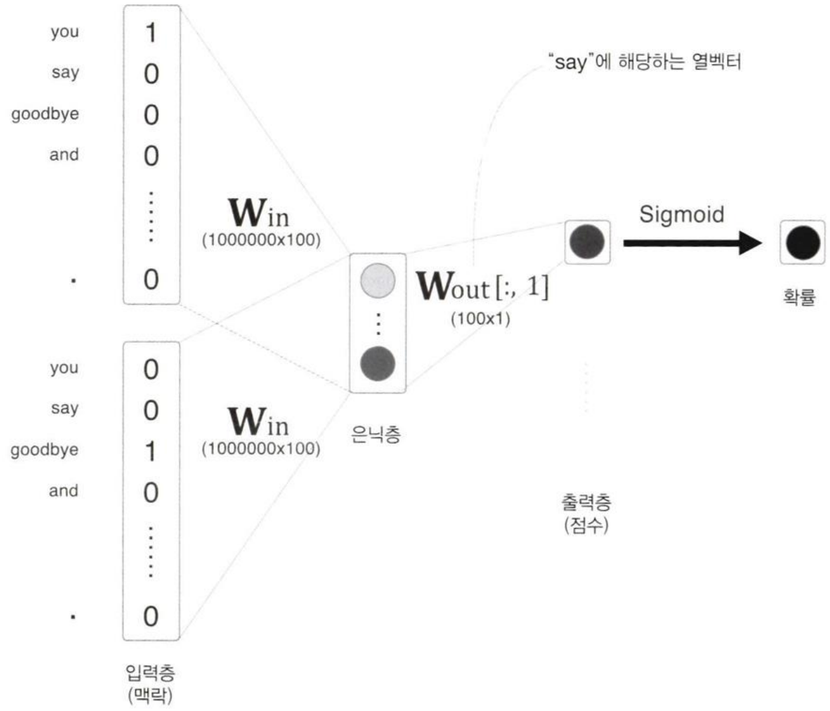
Conversion

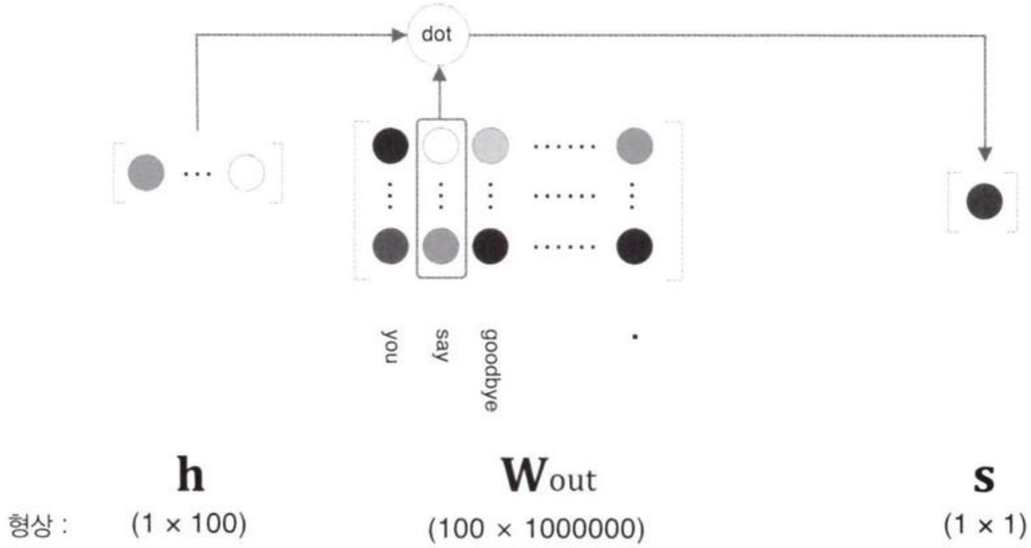
Word Embedding

Word2Vec – CBOW 개선하기 2

- 은닉층과 가중치 행렬 W_{out} 의 곱

・다중 분류의 문제를 이진 분류로 만들기





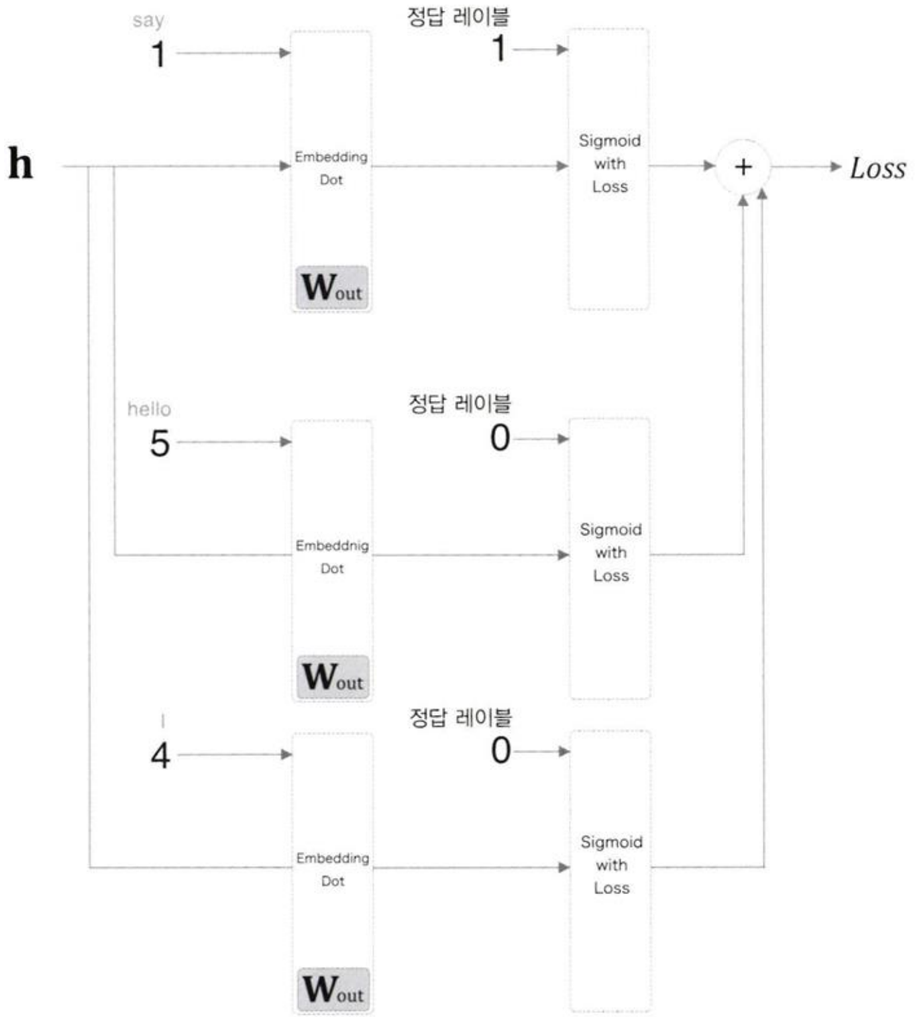
Conversion

Word Embedding

Word2Vec – 네거티브 샘플링

- 정답을 타깃으로 한 경우의 손실을 구하고, 오답을 몇 개 샘플링하여 동시에 손실을 구한 뒤, 두 손실 값의 합을 최종 손실로 사용하는 기법

→ 타깃이 정답일 확률은 1에 가깝게, 오답일 확률은 0에 가깝게 학습



과제

과제 안내

- 수업에서 진행한 Work2Vec의 Skip-gram 방법에 대해서 조사하기 (1~2페이지 워드파일 제출)
- 지금까지 수업에서 나간 논문 중 아직 본인이 리뷰하지 않은 논문 하나 리뷰하기

→ 5.15(수) 23:59분까지 제출해주세요.

REFERENCE

자연어 처리
: 딥 러닝을 이용한 자연어 처리 입문 (유원준, 안상준)
Sequence Data
: (1) Basic of RNN - Sequence Data (tistory.com)
CBOW 모델
: 밑바닥부터 시작하는 딥러닝 2 (사이토 고키)

THANK YOU

Deep Session 7차시