Natural Language Processing

자연어 처리: 임베딩

강사 양석환

임베딩

임베딩이란?

- 컴퓨터의 자연어 이해
  - 딥러닝 기술의 발달로 다양한 언어의 처리, 대응이 가능해 짐
  - 기계번역, 문서요약, 문장 생성 등 많은 성과 도출
  - 그러나 컴퓨터는 자연어를 이해하지 못함

    → 수치 데이터를 계산하여 사람에게 그럴듯하게 보이도록 표현할 뿐
  - 컴퓨터의 자연어 이해와 생성은 연산과 처리의 한 부분이다

임베딩이란?

- 제시된 문제
  - 언어의 표현력은 무한하다!! (대전제)
  - 컴퓨터가 무한한 언어의 표현력을 연산할 수 있도록 수치화는 가능한가?
  - 수치화가 가능하다면 말과 글을 숫자로 변환할 때 어떤 정보를 함축 시킬 것인가?
  - 정보 압축 과정에서 손실이 발생하지 않을까?
  - 손실은 어떻게 줄일 수 있을 것인가?
- 이러한 문제를 해결해 나가는 과정이 “자연어 이해”의 연구 과정

임베딩이란?

- 자연어 처리에서의 임베딩이란?
  - 사람이 쓰는 자연어를
  - 기계가 이해할 수 있는 숫자의 나열인 벡터로 바꾼 결과, 또는 그 과정 전체
  - 단어나 문장 각각을 벡터로 변환하여 “벡터 공간으로 끼워 넣는다”는 의미에서 임베딩이라는 이름이 붙게 됨

임베딩이란?

- 임베딩 기술의 사용
  - 임베딩의 개념은 꽤 오래 전부터 자연어 처리 분야에서 사용되었음
  - 본격적인 부상은..
    - 딥러닝의 대부 “요슈아 벤지오“ 연구팀이 2003년, “A Neural Probabilistic Language Model” 이라는 논문을 발표한 후부터이며
    - 임베딩 기술과 딥러닝 기술이 융합, 통합되어 활용되기 시작함

임베딩이란?

- 생각할 수 있는 가장 간단한 형태의 임베딩은?
  - 단어의 빈도를 그대로 벡터로 사용하는 것
    - 단어-문서 행렬의 예

구분	메밀꽃 필 무렵	운수 좋은 날	사랑 손님과 어머니	삼포 가는 길
기차	0	2	10	7
막걸리	0	1	0	0
선술집	0	1	0	0

    - 기차: '사랑 손님과 어머니', '삼포 가는 길' 에서는 '기차'라는 단어만 다수 사용됨

      → 두 작품이 비슷한 주제 또는 소재를 지녔음을 추측할 수 있음
    - 막걸리, 선술집: '운수 좋은 날'에서만 등장 → '막걸리-선술집' 간의 의미 차이가 '막걸리-기차' 보다 작을 것이라고 추측할 수 있음

임베딩이란?

- 임베딩의 역할
  - 단어/문장 간 관련도 계산
  - 의미적/문법적 정보 함축
  - 전이 학습

임베딩의 역할

- 단어/문장 간 관련도 계산
  - 단어-문서 행렬
    - 가장 단순한 형태의 임베딩 (실무에서는 더 복잡한 형태의 임베딩을 사용함)
  - Word2Vec
    - 2013년 구글에서 발표
    - 단어들을 벡터로 바꾸는 방법, 모델

임베딩의 역할

- 한국어 위키백과, KorQuAD, 네이버 영화 리뷰 등의 말뭉치를 형태소 분석 후 100차원으로 학습한 Word2Vec 임베딩 한 결과 중
  - “희망” 이라는 단어를 기준으로 보면
    - 벡터 표현: [-0.00209 -0.03918 0.02419 … 0.01715 -0.04975 0.009300]
    - 100차원으로 임베딩 → 벡터의 요소는 100개
    - 단어 벡터를 봐서는 의미를 이해할 수 없음

      → 그러나, 단어를 벡터로 표현 → 벡터 사이의 유사도 계산이 가능해 짐
    - 각 쿼리 단어별로 코사인 유사도를 적용한 결과 상위 5개의 단어를 나열해 보면…

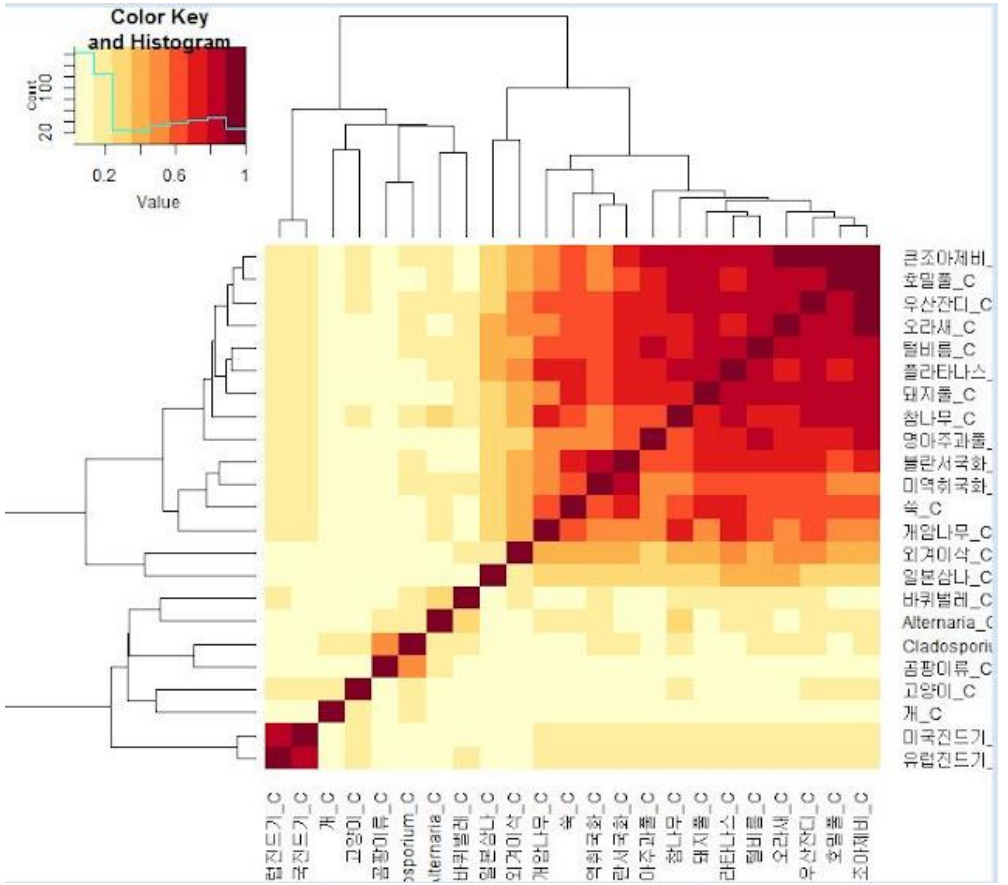
임베딩의 역할

쿼리 단어들의 코사인 유사도 기준 상위 5개 단어 목록

희망	절망	학교	학생	가족	자동차
소망	체념	초등	대학생	아이	승용차
행복	고뇌	중학교	대학원생	부모	상용차
희망찬	절망감	고등학교	고학생	편부모	트럭
꿈	상실감	야학교	교직원	고달픈	대형트럭
열망	번민	중학	학부모	사랑	모터사이클

코사인 유사도가 가장 높은 단어
→ 가장 관련성이 높은 단어

자연어일때 불가능했던 코사인 유사도 계산이
단어 임베딩으로 인해 가능해짐



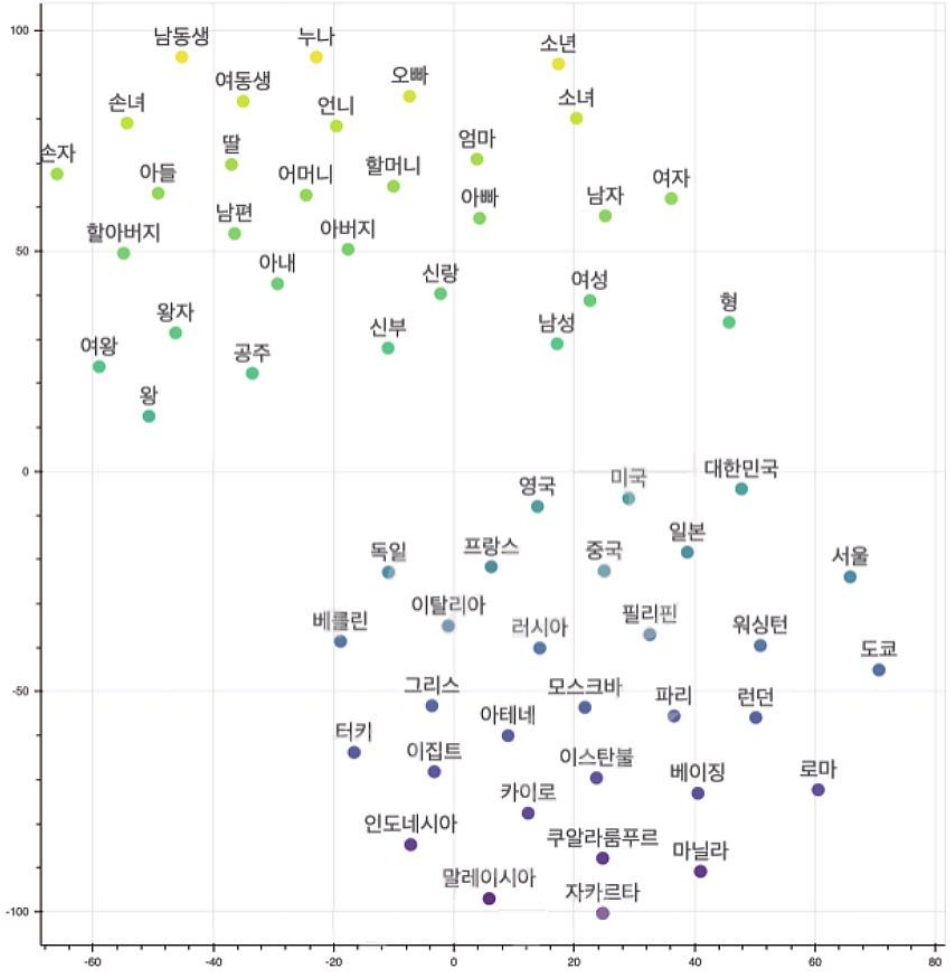
Word2Vec을 통한 단어(벡터) 쌍 간 코사인 유사도 시각화
(색이 짙을수록 유사도가 높음)

임베딩의 역할

- 차원 축소 후 시각화

  → 100차원의 단어 벡터를 2차원으로 축소하여 시각화한 경우

  → 관련성이 높은 단어들이 주변에 몰려 있음을 확인할 수 있음



임베딩의 역할

- 의미적/문법적 정보 함축
  - 임베딩은 벡터로 표현되므로 사칙연산이 가능함

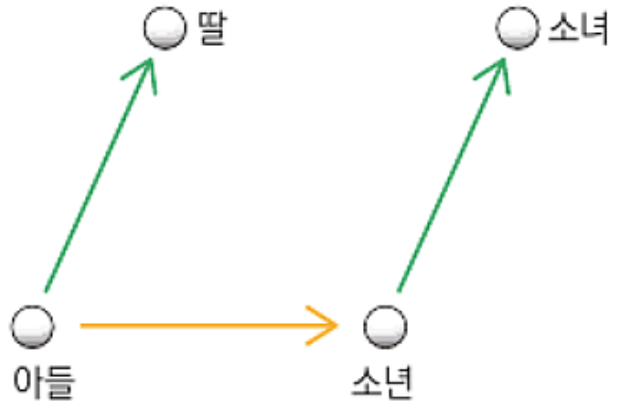
    → 단어 벡터 간 덧셈/뺄셈을 통해

    → 단어들 사이의 의미적, 문법적 관계를 도출할 수 있음

    - 첫 번째 단어 벡터 – 두 번째 단어 벡터 + 세 번째 단어 벡터…
    - 예시
      - 아들 – 딸 + 소녀 = 소년 → 아들 – 딸 = 소년 – 소녀 (이항 연산을 통해서도 가능)

        → 아들-딸 사이의 관계와 소년-소녀 사이의 관계, 의미 차이가 임베딩에 함축되어 있다면

        → 품질이 좋은 임베딩이라고 할 수 있음
    - 이렇게 단어 임베딩을 평가하는 방법: 단어 유추 평가(Word Analogy Test)



임베딩의 역할

- **Word2Vec 임베딩에 단어 유추 평가를 수행한 결과(예시)**

단어 1	단어 2	단어 3	단어 4
아들	딸	소년	소녀
아들	딸	아빠	엄마
아들	딸	남성	여성
남동생	여동생	소년	소녀
남동생	여동생	아빠	엄마
남동생	여동생	남성	여성
신랑	신부	왕	여왕
신랑	신부	손자	손녀
신랑	신부	아빠	엄마

임베딩의 역할

- 전이 학습
  - 특정 분야에서 학습된 신경망의 일부 능력을 유사하거나 전혀 새로운 분야에서 사용되는 신경망의 학습에 이용하는 것
  - 사람의 경우: 무엇인가를 배울 때...
    - 기존에 쌓아온 지식을 바탕으로 새로운 사실만을 추가로 이해 → 전이 학습과 동일한 개념
  - 임베딩에서도 동일하게 활용 가능
    - 대규모 말뭉치를 활용하여 학습된 임베딩 데이터 확보 → 의미적, 문법적 정보 내포됨
    - 임베딩 데이터를 입력 값으로 사용하는 전이 학습 모델 → 문서 분류 등의 태스크를 빠르게 적용 가능

임베딩의 역할

- 전이 학습의 예시
  - 문장의 극성 예측 모델(Bidirectional LSTM 모델 + Attention 적용) 개발
  - 모델의 입력값: FastText 임베딩(100차원)
    - FastText: Word2Vec의 개선된 버전. 59만건의 한국어 문서 사전학습 완료 모델
  - 학습데이터 예시
    - 이 영화 꿀잼

      → “이”, “영화”, “꿀잼“ 임베딩 3개가 순차적으로 입력
    - 이 영화 노잼

      → “이”, “영화”, “노잼“ 임베딩 3개가 순차적으로 입력

$$e_{이} = [-0.20195 \ldots -0.0674]$$

$$e_{영화} = [-0.07806 \ldots 0.11161]$$

$$e_{꿀잼} = [-0.08889 \ldots 0.02243]$$

$$e_{노잼} = [-0.00956 \ldots 0.01423]$$

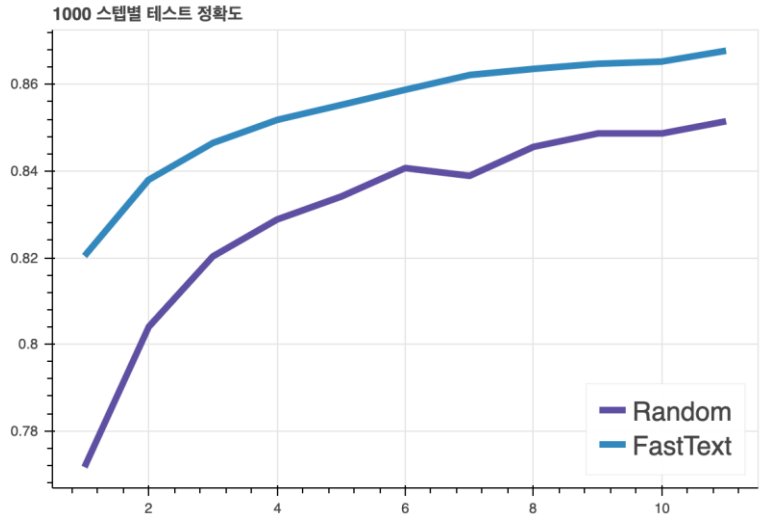
임베딩의 역할

- 전이 학습 모델에 적용

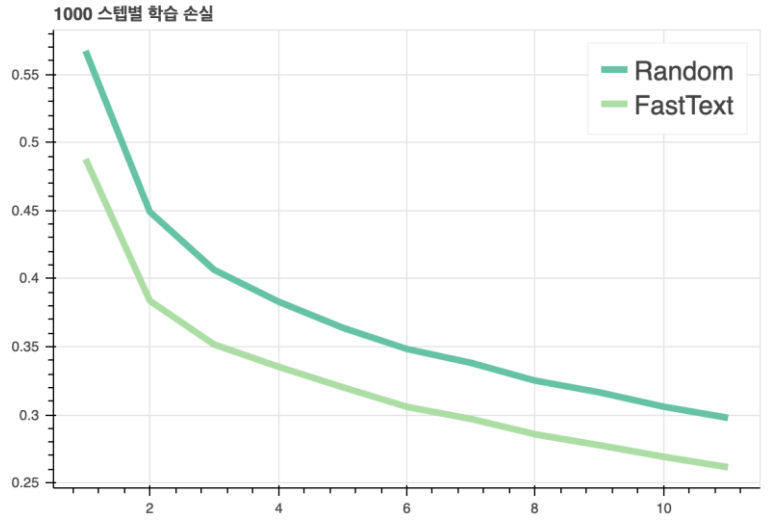
→ 모델은 문장을 입력 받으면 해당 문장이 긍정인지 부정인지 출력함

  - 이 영화 꿀잼 + 긍정(Positive) / 이 영화 노잼 + 부정(Negative)

→ 문장을 형태소 분석한 후 각각의 형태소에 해당하는 FastText 단어 임베딩이 모델의 입력값이 됨



임베딩 종류별 긍정/부정 문서 분류 정확도



임베딩 종류별 학습 손실

FastText 임베딩을 사용한 모델(전이학습을 사용한 모델)의 성능이 더 뛰어남

Random: 입력벡터를 랜덤 초기화 한 후 모델을 학습한 방식 → 제로부터 학습을 시작한 모델

임베딩 기법의 역사와 종류

- 통계 기반에서 뉴럴 네트워크 기반으로
  - 초기 임베딩: 말뭉치의 통계량을 직접적으로 활용하는 경향
    - 대표적인 기법: 잠재 의미 분석
      - 단어 사용 빈도 등 말뭉치의 통계량 정보가 들어있는 행렬에
      - 특이값 분해 등의 수학적 기법을 적용하여
      - 행렬에 속한 벡터들의 차원을 축소 하는 방법

임베딩 기법의 역사와 종류

단어-문서 행렬

	문서1	문서2	문서3	문서4
단어1	2	0	0	0
단어2	0	1	0	1
단어3	0	0	0	3
단어4	1	0	1	2

→ ①

	주제1	주제2
단어1	0.42	1.92
단어2	1.03	-0.29
단어3	2.88	-0.69
단어4	2.29	0.64

↓ ②

	문서1	문서2	문서3	문서4
주제1	0.81	0.27	0.59	3.70
주제2	2.08	-0.13	0.30	-0.49

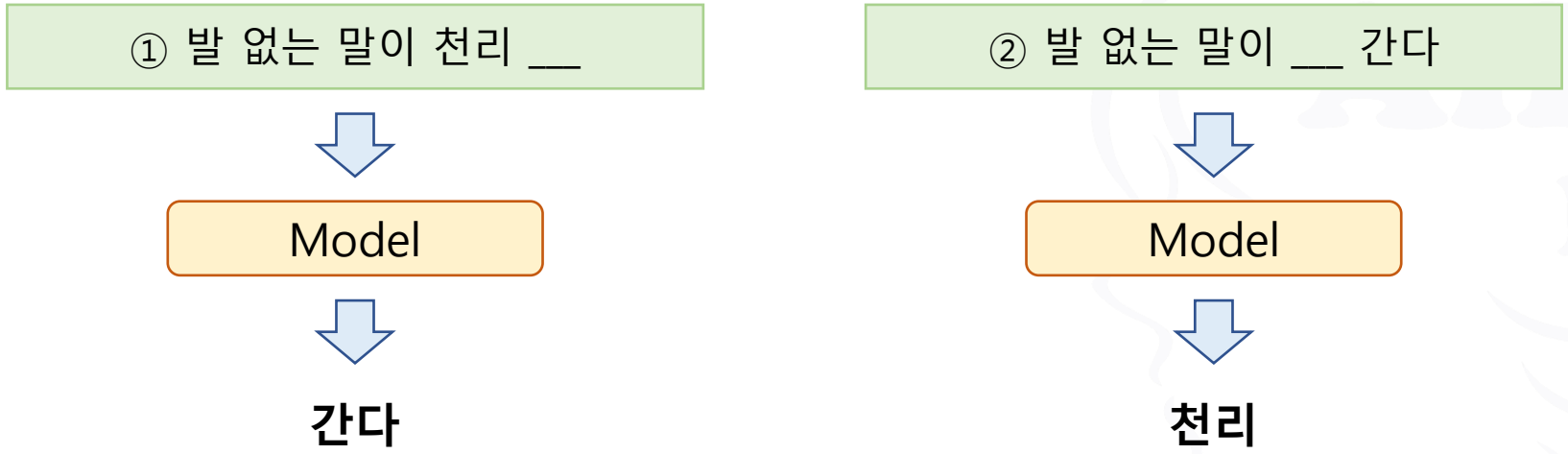
잠재 의미 분석

- 단어-문서 행렬에 잠재 의미 분석 적용
- 단어-문서 행렬
  - 수 십만개의 어휘 →매우 큰 행렬 사용
  - 희소행렬 사용
- 원래 행렬의 차원 축소
  - 단어를 기준으로 축소(①) → 단어 수준 임베딩
  - 문서를 기준으로 축소(②) → 문서 임베딩

- 잠재 의미 분석 수행 대상 행렬
  - 단어-문서 행렬
  - TF-IDF 행렬
  - 단어-문맥 행렬
  - 점별 상호 정보량(PMI) 행렬 등

임베딩 기법의 역사와 종류

- **뉴럴 네트워크(신경망 기반 임베딩)**
  - 요슈아 벤지오, “Neural Probabilistic Language Model” 발표 후부터 주목
  - 이전 단어들이 주어졌을 때 다음 단어가 무엇이 될지 예측하거나, 문장 내 일부분을 비워놓고 해당 단어가 무엇일지 맞추는 과정에서 학습 수행



  - 신경망은 구조가 유연하고 표현력이 풍부 → 자연어의 문맥을 상당 부분 학습 가능

임베딩 기법의 역사와 종류

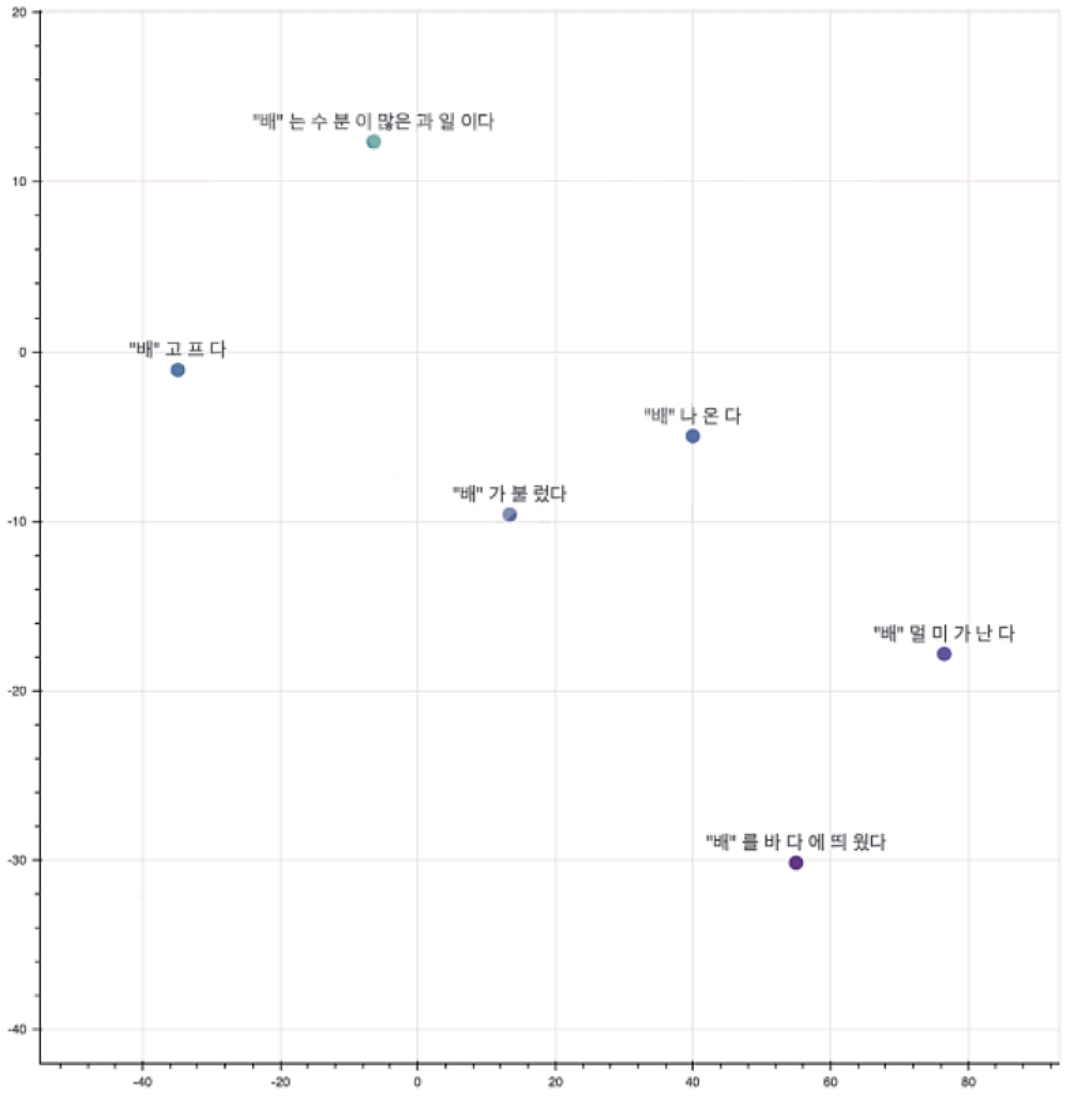
- 단어 수준에서 문장 수준으로
  - 2017년 이전의 임베딩 기법은 대체로 단어 수준 모델
    - NPLM, Word2Vec, GloVe, FastText, Swivel 등
    - 단어 수준 임베딩
      - 각각의 벡터에 해당 단어의 문맥적 의미 함축
      - 단점: 동음이의어 분간이 어렵다

        (단어의 형태가 같으면 동일한 단어로 보고 모든 문맥 정보를 해당 단어 벡터에 투영)

임베딩 기법의 역사와 종류

- **2018년 초, ELMo(Embeddings from Language Models) 발표 후, 문장 수준 임베딩 기법이 주목 받기 시작**
  - BERT(Bidirectional Encoder Representations from Transformer), GPT(Generative Pre-Training) 모델 등
  - 개별 단어가 아닌 단어 시퀀스 전체의 문맥적 의미를 함축
    → 단어 임베딩 기법보다 전이 학습 효과가 좋음

임베딩 기법의 역사와 종류



"배"의 BERT 임베딩 시각화

임베딩 기법의 역사와 종류

- Rule → End to End → Pre-Train / Fine-Tuning
  - ~1990년대
    - 대부분의 자연어 처리 모델은 사람이 Feature(모델의 입력값)를 직접 추출
    - Feature 추출 시 언어학적 지식 활용 → Rule 정의
  - 2000년대 중반 ~
    - 자연어 처리 분야에 딥러닝 모델 적용
    - 딥러닝 모델: 입력, 출력 사이의 관계를 잘 근사(Approximation)

      → 사람이 Rule을 알려주지 않아도 됨
    - End to End 모델 : 데이터를 통째로 모델에 넣고 입출력 사이의 관계를 사람의 개입 없이 모델 스스로 처음부터 끝까지 처리하도록 유도(대표모델: Sequence to Sequence)

임베딩 기법의 역사와 종류

- **2018년 ELMo 모델 제안 이후**
  - End to End 모델에서 벗어나 Pre-Train, Fine-Tuning 방식으로 발전
  - 처리방식
    - 대규모 말뭉치로 임베딩 만들기(Pre-Train) → 임베딩에 말뭉치의 의미적, 문법적 맥락 포함
    - 임베딩을 입력으로 하는 새로운 딥러닝 모델 작성
    - 풀고자 하는 구체적 문제에 맞는 소규모 데이터에 맞게 임베딩을 포함한 모델 전체 업데이트
      → Fine-Tuning, 전이학습(Transfer Learning)

임베딩 기법의 역사와 종류

- **Down-Stream Task**
  - **해결하고자 하는 자연어 처리의 구체적인 문제**
    - **개체명 인식(Named Entity Recognition, NER)**
    - **품사 판별(=품사 태깅)** 나는 네가 지난 여름에 한 [일]을 알고 있다. → 일: 명사
    - **문장 성분 분석** [나는 네가 지난 여름에 한 일]을 알고 있다. → 네가 지난 여름에 한 일: 명사구
    - **의존 관계 분석** [자연어 처리는] 늘 그렇듯이 [재미있다]. → 자연어 처리는, 재미있다: 주격 명사구
    - **의미역 분석** [나는 네가 지난 여름에 한 일]을 알고 있다. → 네가 지난 여름에 한 일: 피행위주역(Patient)
    - **상호 참조 해결** 나는 어제 [성빈이]를 만났다. [그]는 스웨터를 입고 있었다 → 그 : 성빈이

임베딩 기법의 역사와 종류

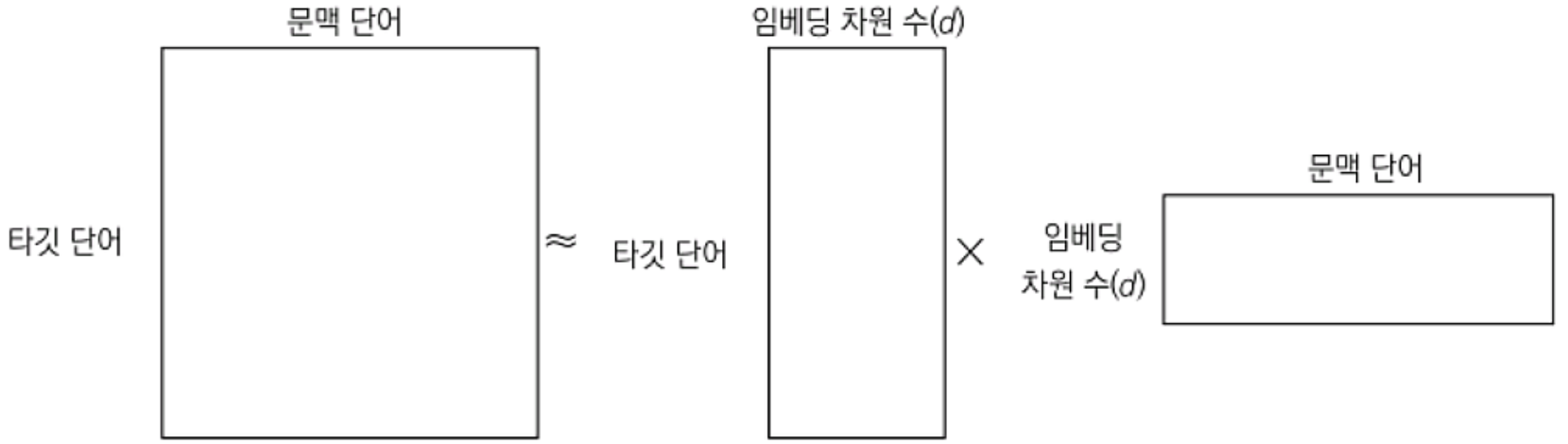
- **Up-Stream Task**
  - **Down-Stream Task에 앞서 해결해야 할 과제**
    - 단어/문장 임베딩의 Pre-Train 작업

임베딩 기법의 역사와 종류

- 임베딩의 종류
  - 행렬 분해 기반 방법
  - 예측 기반 방법
  - 토픽 기반 방법
  - 임베딩 성능 평가

임베딩 기법의 역사와 종류

- **임베딩의 종류**
  - **행렬 분해(factorization) 기반 방법**
    - 말뭉치 정보가 들어있는 원리 행렬을 두 개 이상의 작은 행렬로 쪼개는 방식
    - 분해한 후, 둘 중 하나의 행렬만 쓰거나, 둘을 더하거나(sum) 이어 붙여(concatenate) 임베딩으로 사용
    - GloVe, Swivel 등



임베딩 기법의 역사와 종류

- **예측 기반 방법**
  - 어떤 단어 주변에 특정 단어가 나타날지 예측하거나, 이전 단어들이 주어졌을 때 다음 단어가 무엇일지 예측하거나, 문장 내 일부 단어를 지우고 해당 단어가 무엇일지 맞추는 과정에서 학습하는 방법
  - 뉴럴 네트워크 방법들이 예측 기반 방법에 속함
  - Word2Vec, FastText, BERT, ELMo, GPT 등

임베딩 기법의 역사와 종류

- **토픽 기반 방법**
  - 주어진 문서에 잠재된 주제를 추론하는 방식으로 임베딩을 수행하는 방법
  - 대표적인 기법: 잠재 디리클레 할당(Latent Dirichlet Allocation, LDA)
    - 학습이 완료되면 각 문서가 어떤 주제 분포를 갖는지 확률 벡터 형태로 반환

      → 임베딩 기법의 일종으로 받아들여짐

임베딩 기법의 역사와 종류

- **임베딩 성능 평가**
  - Down-Stream Task 에 대한 임베딩 종류별 성능 분석
  - 성능 측정 대상 Down-Stream Task
    - 형태소 분석, 문장 성분 분석, 의존 관계 분석, 의미역 분석, 상호 참조 해결 등
  - 파인튜닝 모델의 구조를 고정한 뒤 각각의 임베딩을 전이학습 시키는 형태로 정확도 측정
  - 문장 임베딩 기법(ELMo, GPT, BERT 등)이 단어 임베딩 기법(GloVe 등)을 크게 앞섬
  - 한국어는 공개된 데이터가 적어 측정하기 어려움

벡터는 어떻게 의미를 가지게 되는가?

벡터는 어떻게 의미를 가지게 되는가?

- 컴퓨터가 수행하는 자연어 이해의 본질은 계산이다.
- 임베딩에 자연어의 의미를 어떻게 함축할 수 있는가?
  - 자연어의 통계적 패턴 정보를 통째로 임베딩에 넣는 것
  - 자연어의 의미는 해당 언어의 화자들이 실제 사용하는 일상 언어에서 드러남

벡터는 어떻게 의미를 가지게 되는가?

- **임베딩에 사용되는 통계정보**
  - **문장에 어떤 단어가 (많이) 쓰였는가?**
  - **단어가 어떤 순서로 등장하는가?**
  - **문장에 어떤 단어가 같이 나타났는가?**

구분	백오브워즈(BOW) 가정	언어 모델	분포 가정
내용	어떤 단어가 (많이) 쓰였는가?	단어가 어떤 순서로 등장하는가?	어떤 단어가 같이 나타났는가?
대표 통계량	TF-IDF	-	PMI
대표 모델	Deep Averating Network	ELMo, GPT	Word2Vec

벡터는 어떻게 의미를 가지게 되는가?

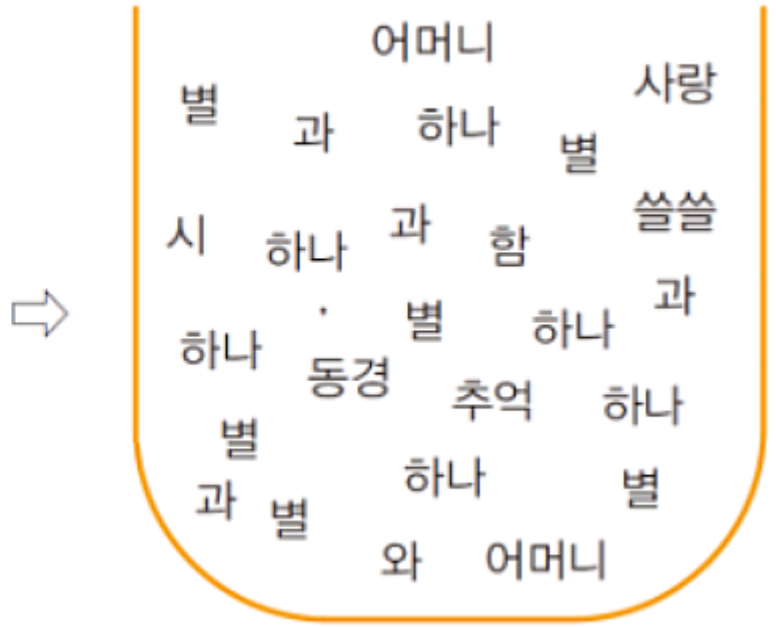
- 각 방법의 연관성
  - 언어모델은 단어의 등장 순서, 분포 가정은 이웃 단어(문맥)을 우선시
  - 단어가 문장에서 주로 나타나는 순서는 해당 단어의 주변 문맥과 밀접한 관계
  - PMI는 어떤 단어 쌍이 얼마나 자주 같이 나타나는지에 대한 정보를 수치화하며, 그를 위하여 개별 단어, 단어 쌍의 빈도 정보를 활용
  - BOW, 언어모델, 분포가정은 말뭉치의 통계적 패턴을 서로 다른 각도에서 분석하며, 상호 보완적으로 동작함

벡터는 어떻게 의미를 가지게 되는가?

- 어떤 단어가 많이 쓰였는가?
  - BOW(Bag of Words)
    - Bag(가방)
      - 수학에서 중복 원소를 허용한 집합을 의미함
      - 원소의 순서는 고려하지 않음
    - 자연어 처리 분야에서 BOW란 단어의 등장 순서에 관계없이 문서 내 단어의 등장 빈도를 임베딩으로 쓰는 기법을 말함

벡터는 어떻게 의미를 가지게 되는가?

별 하나 에 추억 과
별 하나 에 사랑 과
별 하나 에 쓸쓸함 과
별 하나 에 동경 과
별 하나 에 시 와
별 하나 에 어머니 , 어머니



별	하나	에	추억	과	사랑	쓸쓸	함	동경	시	와	어머니	.
6	6	6	1	4	1	1	1	1	1	1	2	1

BOW(Bag of Words) 임베딩

벡터는 어떻게 의미를 가지게 되는가?

- **BOW의 가정**
  - 저자가 생각한 주제가 문서에서의 단어 사용에 녹아 있다
  - 주제가 비슷한 문서라면 단어의 빈도 또는 단어의 등장여부도 비슷할 것이므로 BOW 임베딩 역시 유사할 것이다
  - 빈도를 그대로 BOW로 쓴다면 많이 쓰인 단어가 주제와 더 강한 관련을 맺고 있을 것이다

벡터는 어떻게 의미를 가지게 되는가?

- **TF-IDF**
  - **BOW의 단점**
    - 단어의 빈도, 등장 여부를 그대로 임베딩으로 쓰는 경우

      → 해당 단어가 어떤 문서에서든 많이 쓰이는 경우, 문서의 주제를 가늠하기 어렵다
  - **BOW의 단점 개선을 위하여 → TF-IDF(Term Frequency-Inverse Document Frequency) 기법 제안**
    - 단어-문서 행렬에 $TF - IDF(w) = TF(w) \times \log\left(\frac{N}{DF(w)}\right)$ 과 같이 가중치를 계산하여 행렬의 원소를 바꿈
    - TF-IDF 역시 단어의 등장 순서를 고려하지 않음 → BOW의 일종

벡터는 어떻게 의미를 가지게 되는가?

- $TF - IDF(w) = TF(w) \times \log\left(\frac{N}{DF(w)}\right)$ **에서**
  - **TF: 어떤 단어가 특정 문서에 얼마나 많이 쓰였는지에 대한 빈도 수**
    - 많이 쓰인 단어가 중요하다는 가정을 전제로 한 수치
  - **DF: 특정 단어가 나타난 문서의 수**
    - DF가 클 수록 다수 문서에 사용되는 범용적인 단어
  - **TF는 같은 단어라도 문서마다 다른 값, DF는 문서가 달라져도 단어가 같다면 동일한 값**
  - **IDF: 전체 문서 수(N)를 해당 단어의 DF로 나눈 뒤 로그를 취한 값 → 값이 클 수록 특이한 단어 → 단어의 주제 예측 능력(해당 단어만 보고 문서의 주제를 가늠해 볼 수 있는 정도)과 직결**

벡터는 어떻게 의미를 가지게 되는가?

- TF-IDF의 지향점
  - 어떤 단어의 주제 예측 능력이 강할수록 가중치가 커지고, 그 반대의 경우는 작아짐
  - 어떤 단어의 TF가 높으면 TF-IDF값도 커짐

    → 단어 사용 빈도는 저자가 상정한 주제와 관련을 맺고 있을 것이라는 가정에 기초

단어-문서 행렬

구분	메밀꽃 필 무렵	운수 좋은 날	사랑 손님과 어머니	삼포 가는 길
기차	0	2	10	7
막걸리	0	1	0	0
선술집	0	1	0	0

TF-IDF 행렬

구분	메밀꽃 필 무렵	운수 좋은 날	사랑 손님과 어머니	삼포 가는 길
어머니	0.066	0.0	**0.595**	0.0
것	0.2622	0.098	0.145	0.0848

정보성이 없는 단어는 가중치가 줄어들게 되어 불필요한 정보가 사라짐

두 번째 표가 더 품질이 좋은 임베딩

벡터는 어떻게 의미를 가지게 되는가?

- **BOW의 경우**
  - 문장 내에 어떤 단어가 쓰였는지, 쓰였다면 얼마나 많이 쓰였는지에 대한 빈도만 계산
  - 해당 문서가 어떤 범주인지 분류(classification)하는 문제에 대해서는 간단한 아키텍처임에도 불구하고 성능이 좋음 → 해당 문제에서는 현업에서도 자주 활용됨

벡터는 어떻게 의미를 가지게 되는가?

- 단어가 어떤 순서로 쓰였는가?
  - 통계 기반 언어 모델
    - 언어 모델
      - 단어 시퀀스에 확률을 부여하는 모델
      - 단어의 등장 순서를 무시하는 BOW와 달리 시퀀스 정보를 명시적으로 학습
      - BOW의 반대편에 있는 모델이라고 볼 수 있음
    - n 개의 단어가 주어졌다면 언어 모델은 n개 단어가 동시에 나타날 확률, $P(w_1, w_2, \dots, w_n)$을 반환
    - 통계 기반의 언어 모델은 말뭉치에서 해당 단어 시퀀스가 얼마나 자주 등장하는지 빈도를 세어 학습

벡터는 어떻게 의미를 가지게 되는가?

- 잘 학습된 언어 모델이 있다면... 다음을 알 수 있음
  - 어떤 문장이 그럴듯한지(확률 값이 높은지)
  - 주어진 단어 시퀀스 다음 단어는 무엇이 오는 것이 자연스러운지

누명을 쓰다 → 0.41
누명을 당하다 → 0.02

두시 삼십이분 → 0.51
이시 서른두분 → 0.08

난폭 운전 → 0.39
무모 운전 → 0.01

선생님께는 낡은 집이 한 채 있으시다 → 0.12
진이에게는 존경하는 선생님이 한 분 있으시다 → 0.01

진이는 이 책을 세 번을 읽었다. → 0.47
이 책이 진이한테 세 번을 읽혔다. → 0.23
세 번이 진이한테 이 책을 읽혔다. → 0.07

한국어 언어 모델의 예시

: 자연스러운 한국어 문장에 높은 확률 값을 부여

벡터는 어떻게 의미를 가지게 되는가?

- **n-gram**
  - n-gram 이란 n개의 단어를 뜻함
    - 난폭, 운전 : 2-gram (=bigram)
    - 눈, 뜨다 : 2-gram (=bigram)
    - 누명, 을, 쓰다 : 3-gram (=trigram)
    - 바람, 잘, 날, 없다 : 4-gram
  - 경우에 따라서 n-gram은 n-gram에 기반한 언어모델을 의미
    - 말뭉치 내 단어들을 n개씩 묶어서 그 빈도를 학습했다는 의미

벡터는 어떻게 의미를 가지게 되는가?

- **네이버 영화 리뷰 말뭉치에서**
  - 각 표현이 등장한 횟수
  - 띄어쓰기 단위인 어절을 하나의 단어로 보고 빈도 계산
  - “내 마음 속에 영원히 기억될 최고의 명작이다.” 의 경우, 말뭉치에서 한 번도 등장하지 않음

    → 언어모델은 해당 표현이 나타날 확률을 0으로 부여

    → 문법적, 의미적으로 결함이 없는 훌륭한 한국어 문장이지만 언어모델은 해당 표현을 말이 되지 않는 문장으로 취급함

표현	빈도
내	1309
마음	172
속에	155
영원히	104
기억될	29
최고의	3503
명작이다	298
내 마음	93
마음 속에	9
속에 영원히	7
영원히 기억될	7
기억될 최고의	1
최고의 명작이다	23
영원히 기억될 최고의 명작이다	1
내 마음 속에 영원히 기억될 최고의 명작이다	0

네이버 영화 말뭉치의 각 표현 별 등장 횟수

벡터는 어떻게 의미를 가지게 되는가?

- “내 마음 속에 영원히 기억될 최고의” 다음에 “명작이다”가 나타날 확률의 계산
  - 조건부 확률(conditional probability)의 정의를 활용하여 최대우도추정법(Maximum Likelihood Estimation)으로 유도할 수 있음

$$P(\text{명작이다} \mid \text{내, 마음, 속에, 영원히, 기억될, 최고의})$$

$$= \frac{\text{Freq}(\text{내, 마음, 속에, 영원히, 기억될, 최고의, 명작이다})}{\text{Freq}(\text{내, 마음, 속에, 영원히, 기억될, 최고의})}$$

  - 우변의 분자가 0이 되므로 전체 값은 0
- 이러한 문제를 n-gram 모델을 이용하여 해결 가능함
  - 직전 n-1개 단어의 등장 확률로 전체 단어 시퀀스의 등장 확률을 근사
    - “한 상태의 확률은 그 직전 상태에만 의존한다”라는 Markov Assumption(마르코프 가정)에 기반

벡터는 어떻게 의미를 가지게 되는가?

$$P(\text{명작이다}|\text{내, 마음, 속에, 영원히, 기억될, 최고의}) \approx P(\text{명작이다}|\text{최고의})$$

$$= \frac{\text{Freq}(\text{최고의, 명작이다})}{\text{Freq}(\text{최고의})} = \frac{23}{3503}$$

- **바이그램 모델에서 “내 마음 속에 영원히 기억될 최고의 명작이다”라는 단어 시퀀스가 나타날 확률은?**

$$P(\text{내, 마음, 속에, 영원히, 기억될, 최고의, 명작이다})$$

$$\approx P(\text{내}) \times P(\text{마음}|\text{내}) \times P(\text{속에}|\text{마음}) \times P(\text{영원히}|\text{속에}) \times$$

$$P(\text{기억될}|\text{영원히}) \times P(\text{최고의}|\text{기억될}) \times P(\text{명작이다}|\text{최고의})$$

$$= \frac{1309}{|V|} \frac{93}{1309} \frac{9}{172} \frac{7}{155} \frac{7}{104} \frac{1}{29} \frac{23}{3503}$$

- **단어를 하나씩 이어가면서 끝까지 계산한 결과가 원하는 확률이 됨**

벡터는 어떻게 의미를 가지게 되는가?

- **바이그램 모델의 일반화 수식**

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

$$P(w_1^n) = P(w_1, w_2, \dots, w_n) = \prod_{k=1}^{n} P(w_k|w_{k-1})$$

- **그런데 데이터에 한 번도 등장하지 않는 n-gram이 존재한다면 예측 단계에서 문제 발생**
  - **예시**
    - 또바기: “언제나 한결같이 꼭 그렇게”라는 뜻을 가진 한국어 부사
    - 학습 데이터에서 “아이는” 다음에 “또바기”라는 단어가 한 번도 등장하지 않았다면

      → 언어 모델은 예측 단계에서 “그 아이는 또바기 인사를 잘한다”라는 문장이 등장할 확률을 0으로 부여

벡터는 어떻게 의미를 가지게 되는가?

- n-gram 모델의 한계 예시

$$
\begin{aligned}
&P(\text{그 아이는 또바기 인사를 잘한다}) \\
&= P(\text{그}) \times P(\text{아이는}|\text{그}) \times P(\text{또바기}|\text{아이는}) = 0 \\
&\quad \times P(\text{인사를}|\text{또바기}) \times P(\text{잘한다}|\text{인사를}) \\
&= 0
\end{aligned}
$$

- 한계의 해결을 위해 Back-Off, Smoothing 등의 방식 제안

벡터는 어떻게 의미를 가지게 되는가?

- Back-Off
  - n-gram 등장 빈도를 n보다 작은 범위의 단어 시퀀스 빈도로 근사하는 방식
  - n을 크게 하면 할수록 등장하지 않는 케이스가 많아질 가능성이 높다
  - 예시
    - 7-gram 모델 적용 시 “내 마음 속에 영원히 기억될 최고의 명작이다”의 등장 빈도는 0
    - Back-Off 방식으로 n을 4로 줄여서 7-gram 빈도를 근사하면

$$\begin{aligned}&\text{Freq}(\text{내 마음 속에 영원히 기억될 최고의 명작이다})\\&\approx \alpha\text{Freq}(\text{영원히 기억될 최고의 명작이다})+\beta\end{aligned}$$

벡터는 어떻게 의미를 가지게 되는가?

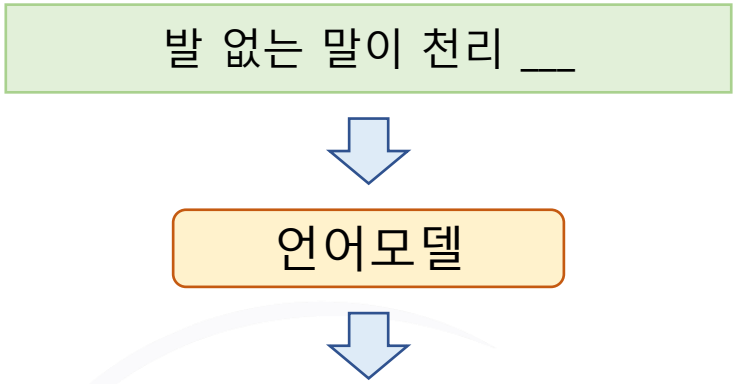
- Smoothing
  - 등장 빈도 표에 모두 k 만큼을 더하는 기법
  - “내 마음 속에 영원히 기억될 최고의 명작이다”의 등장 빈도는 $k(=0+k)$가 됨 → Add-k Smoothing 이라고도 부름
  - K=1인 경우, Laplace Smoothing(라플라스 스무딩)
  - Smoothing을 시행하면 높은 빈도를 가진 문자열 등장 확률을 일부 깎고, 학습 데이터에 전혀 등장하지 않는 케이스들에는 작은 값이지만 일부 확률을 부여함

표현	빈도
내	1309
마음	172
속에	155
영원히	104
기억될	29
최고의	3503
명작이다	298
내 마음	93
마음 속에	9
속에 영원히	7
영원히 기억될	7
기억될 최고의	1
최고의 명작이다	23
영원히 기억될 최고의 명작이다	1
내 마음 속에 영원히 기억될 최고의 명작이다	0

벡터는 어떻게 의미를 가지게 되는가?

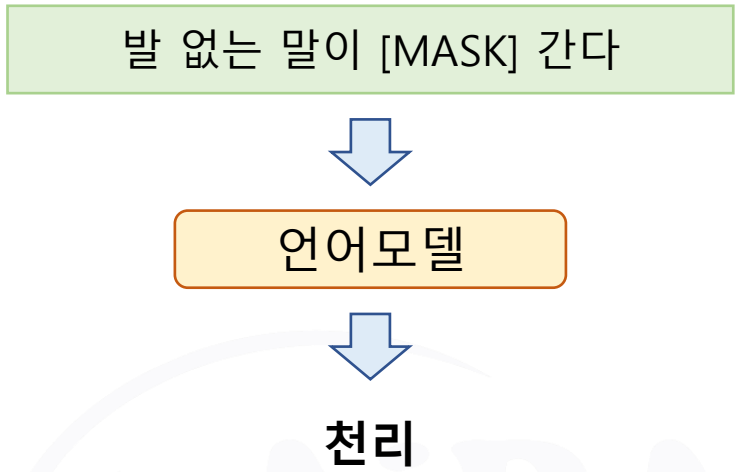
- **뉴럴 네트워크 기반 언어 모델**
  - 뉴럴 네트워크
    - 입력과 출력 사이의 관계를 유연하게 포착할 수 있다
    - 그 자체로 확률모델로 기능할 수 있다

      → 통계 기반 언어 모델을 대신할 수 있다
  - 주어진 단어 시퀀스를 가지고 다음 단어를 맞추는 과정에서 학습을 수행
  - 학습이 완료되면 모델의 중간 혹은 말단 계산 결과물을 단어나 문장의 임베딩으로 활용
  - ELMo, GPT 등의 모델이 여기에 해당함
  - 단어를 순차적으로 입력 받아 다음 단어를 맞춰야 함 → 일방향



벡터는 어떻게 의미를 가지게 되는가?

- 마스크 언어 모델
  - 기본 뉴럴 네트워크 기반 언어 모델과 큰 틀에서 유사하지만 디테일에서 차이를 보임
  - 문장의 중간에 마스크를 씌우고 해당 마스크 위치에 어떤 단어가 올지 예측하는 과정에서 학습 수행
  - 문장 전체를 다 보고 중간에 있는 단어를 예측하므로 양방향 학습이 가능함



- 이런 이유로 마스크 언어 모델 기반의 방법이 기존 언어 모델 기법과 비교하여 임베딩 품질이 뛰어남
- BERT 모델에 여기에 해당함

어떤 단어가 같이 쓰였는가?

어떤 단어가 같이 쓰였는가?

- **분포 가정**
  - **(자연어 처리에서의) 분포: 특정 범위 →**

    윈도우 내에 동시에 등장하는 이웃 단어 또는 문맥의 집합을 가리킴
    - 개별 단어의 분포는 그 단어가 문장 내에서 주로 어느 위치에 나타나는지, 이웃한 위치에 어떤 단어가 자주 나타나는지에 따라 달라짐
    - 어떤 단어 쌍이 비슷한 문맥 환경에서 자주 등장한다면 그 의미 또한 유사할 것인다.. 라는 것이 분포 가정의 전제
    - 모국어의 화자들이 해당 단어를 실제로 어떻게 사용하고 있는지 문맥을 살핌으로써 그 단어의 의미를 밝힐 수 있다는 주장

어떤 단어가 같이 쓰였는가?

- **빨래, 세탁이라는 단어를 모른다고 할 때**
  - **빨래, 세탁: 타겟 단어**
  - **청소, 물 등: 주위에 등장한 문맥 단어**
  - **단어를 살펴보면**
    - **빨래: 청소, 요리, 물, 속옷 등과 함께 등장**
    - **세탁: 청소, 요리, 물, 옷과 같이 등장**

한국어 위키 백과에 언급된 빨래, 세탁

... 특기 는 자칭 청소 와 빨래 지만 요리 는 절망 적 ...
... 재 를 우려낸 물 로 빨래 할 때 나 ...
... 개울가 에서 속옷 빨래 를 하 는 남녀 ...

... 찬 물 로 옷 을 세탁 한다 ...
... 세탁 , 청소 , 요리 와 가사 는 ...

→ 빨래와 세탁은 비슷한 의미를 가질 가능성이 높다

→ 빨래가 청소, 요리, 물, 속옷 등과 같이 등장하므로 이들 단어끼리도 직간접적으로 관계를 가질 가능성 역시 낮지 않다

→ 빨래: 물을 이용해 수행하는 어떤 행위 또는 속옷에 가하는 어떤 행위, 청소/요리와 비슷한 부류의 행위 등으로 추측 가능

어떤 단어가 같이 쓰였는가?

- **그런데...**
  - 각 단어의 의미는 추측이 가능해 졌지만 개별 단어의 분포, 의미 사이의 논리적 연관성이 드러나지 않음

    → “분포 정보가 곧 의미”라는 분포 가설에 의문 제기 가능

어떤 단어가 같이 쓰였는가?

- 형태소(Morpheme)
  - 의미를 가지는 최소 단위 → 더 쪼개면 의미가 사라짐. '의미'에는 어휘적, 문법적 의미 포함
  - 예시
    - 철수가 밥을 먹었다 → 철수, 가, 밥, 을, 먹, 었다
    - 만약 철수 → 철/수로 쪼개진다면 → 철수라는 사람을 지칭하는 의미가 사라짐
    - 밥 → "ㅂ/압"으로 쪼개진다면 → 먹는 밥(Rice)이라는 의미가 사라짐
    - 따라서 "철수", "밥"은 형태소

어떤 단어가 같이 쓰였는가?

- **언어학자들은 형태소 분석을 어떻게? → 계열 관계를 바탕으로 분석함**
  - 계열 관계: 해당 형태소의 자리에 다른 형태소가 “대치 " 되어 쓰일 수 있는가를 따지는 것
  - “철수”의 자리에는 “영희”가 올 수 있다
  - “밥”의 자리에는 “빵”이 올 수 있다

} 대치가 가능하므로 형태소 자격 부여

- **언어학자는 계열 관계를 바탕으로 형태소를 분석한다**

  → 언어학자들은 특정 타겟 단어 주변의 문맥 정보를 바탕으로 형태소를 확인한다.

  → 말뭉치의 분포 정보와 형태소가 밀접한 관계를 이루고 있다.

어떤 단어가 같이 쓰였는가?

- 품사
  - 품사: 문법적 성질의 공통성에 따라 언어학자들이 각 단어를 몇 갈래로 묶어 놓은 것
  - 품사의 분류 기준: 기능, 의미 , 형식
    - 기능: 한 단어가 문장 가운데서 다른 단어와 맺는 관계
    - 의미: 단어의 형식적인 의미
    - 형식: 단어의 형태적 특징

어떤 단어가 같이 쓰였는가?

- 예시
  - 이 샘의 깊이가 얼마냐?
  - 저 산의 높이가 얼마냐?
  - 이 샘이 깊다.
  - 저 산이 높다.

기능

깊이, 높이 → 문장의 주어
깊다, 높다 → 문장의 서술어

기능이 같은 단어 부류를 같은 품사로 묶을 수 있다.
품사의 분류에선 가장 중요

의미

깊이, 깊다 → 묶음
높이, 높다 → 묶음

같은 의미를 가진 군집으로 묶을 수 있다

형식

깊이, 높이 → 불변
깊다, 높다 → 깊었다, 높았다, 깊겠다... → 다양한 변화 가능

어떤 단어가 같이 쓰였는가?

- 품사의 분류에서 가장 중요한 기준은 "**기능**"
  - 의미, 형식은 어느정도 고려 사항은 될 수 있으나 결정적인 기준이 되지 못함
- 한국어를 비롯한 많은 언어에서는
  - 어떤 단어의 기능이 그 단어의 분포와 매우 밀접한 관계를 맺고 있음
- 기능과 분포
  - 기능: 특정 단어가 문장 내에서 어떤 역할을 하는지
  - 분포: 그 단어가 어느 자리에 있는지
  - **기능과 분포는 개념적으로 엄밀히 다르지만 밀접한 관련이 있다**
- 즉, 형태소의 경계를 정하거나 품사를 나누는 등의 언어학적 문제는 말뭉치의 분포 정보와 깊은 관계가 있으며
- 이로 인해 **임베딩에 분포 정보를 함축하면 해당 벡터에 해당 단어의 의미를 내재 시킬 수 있다**

어떤 단어가 같이 쓰였는가?

- 한국어 품사 분류의 일반적 기준
  - 체언(명사): 관형사가 그 앞에 올 수 있고 조사가 그 뒤에 올 수 있음
  - 용언(동사/형용사): 부사가 그 앞에 올 수 있고 선어말어미가 그 뒤에 올 수 있으며 어말어미가 그 뒤에 와야 함
  - 관형사: 명사가 그 뒤에 와야 함
  - 부사: 용언, 부사, 절이 그 뒤에 와야 함
  - 조사: 체언 뒤에 와야 함
  - 어미: 용언 뒤에 와야 함
  - 감탄사(간투사): 특별한 결합 제약 없이.. 즉, 문장 내의 다른 단어와 문법적인 관계를 맺지 않고 따로 존재함

어떤 단어가 같이 쓰였는가?

- **점별 상호 정보량(PMI, Pointwise Mutual Information)**
  - 두 확률변수 사이의 상관성을 계량화하는 단위
  - 두 확률변수가 완전히 독립인 경우, 그 값이 0이 됨
    - 독립: 단어 A가 나타나는 것이 단어 B의 등장 확률에 전혀 영향을 주지 않음(반대도 성립)
    - 단어 A 또는 B의 등장이 단어 B 또는 A와 자주 같이 나타난다면 → PMI 값은 커짐
  - PMI는 두 단어의 등장이 독립일 때와 대비해 얼마나 자주 같이 등장하는지를 수치화 한 것

어떤 단어가 같이 쓰였는가?

- PMI 공식

$$PMI(A,B) = \log \frac{P(A,B)}{P(A) \times P(B)}$$

- PMI는 두 단어가 얼마나 자주 같이 등장하는지에 관한 정보를 수치화 한, 분포 가정에 따른 단어 가중치 할당 기법

  → PMI 행렬의 행 벡터 자체를 해당 단어의 임베딩으로 사용 가능

어떤 단어가 같이 쓰였는가?

window=2 인 단어-문맥 행렬

타겟 단어
▽
문맥 단어
▽
에서, 속옷, 를, 하는
▽
문맥단어가 빈도 계산의 대상이 되고 이 값들을 1씩 올려줌

개울가, 에서, 속옷, 빨래, 를, 하는, 남녀

단어 \ 문맥	개울가	에서	속옷	빨래	를	하는	남녀	total
개울가								
⋮								
빨래		+1	+1		+1	+1		20
⋮								
total			15					1000

단어-문맥 행렬의 구축 과정

빨래와 속옷이 동시에 등장한 빈도가 10회라고 가정하면

$$PMI(\text{빨래},\text{속옷})$$

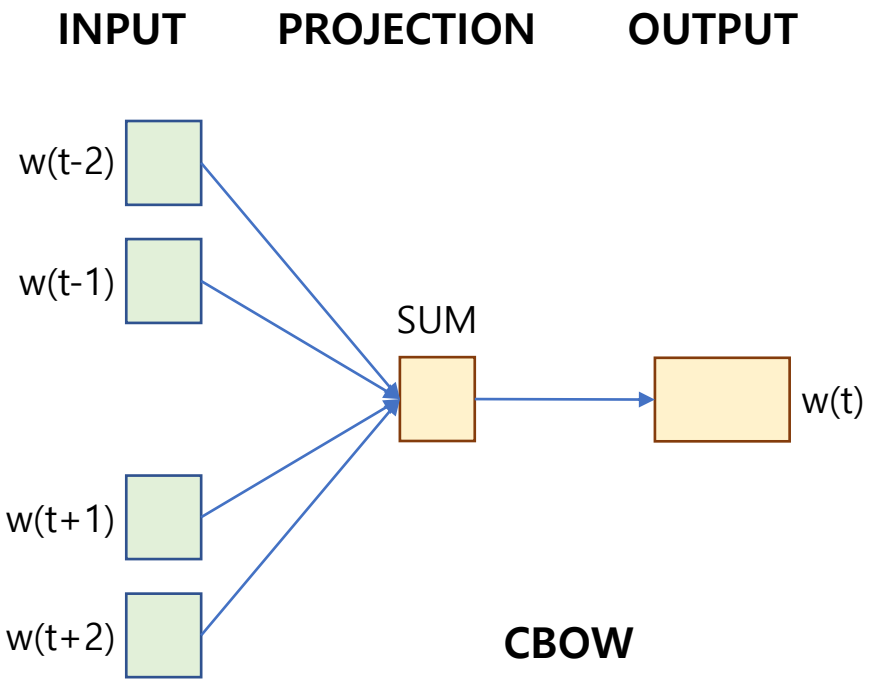
$$= \log \frac{P(\text{빨래},\text{속옷})}{P(\text{빨래}) \times P(\text{속옷})}$$

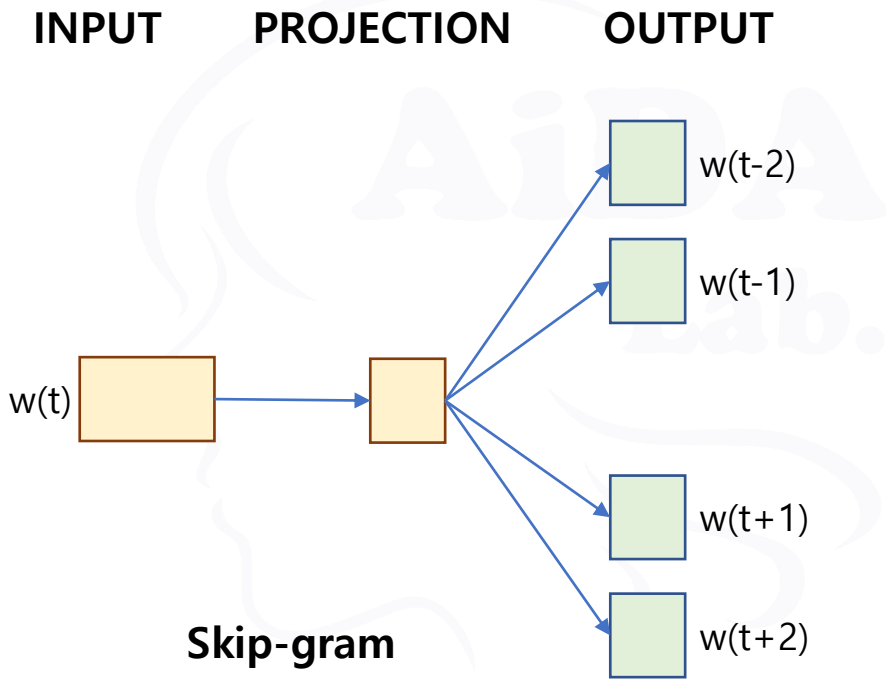
$$= \log \frac{\frac{10}{1000}}{\frac{20}{1000} \times \frac{15}{1000}}$$

$$= 1.5228787 \ldots$$

어떤 단어가 같이 쓰였는가?

- **Word2Vec**
  - 2013년 구글에서 발표한 임베딩 기법으로, 분포 가정의 대표적인 모델
  - 기본 구조



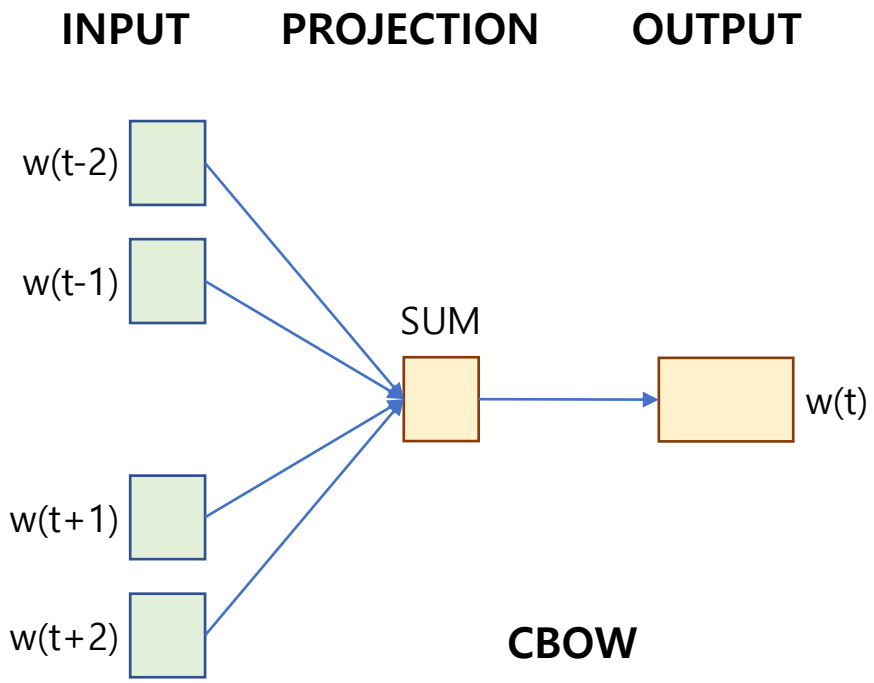


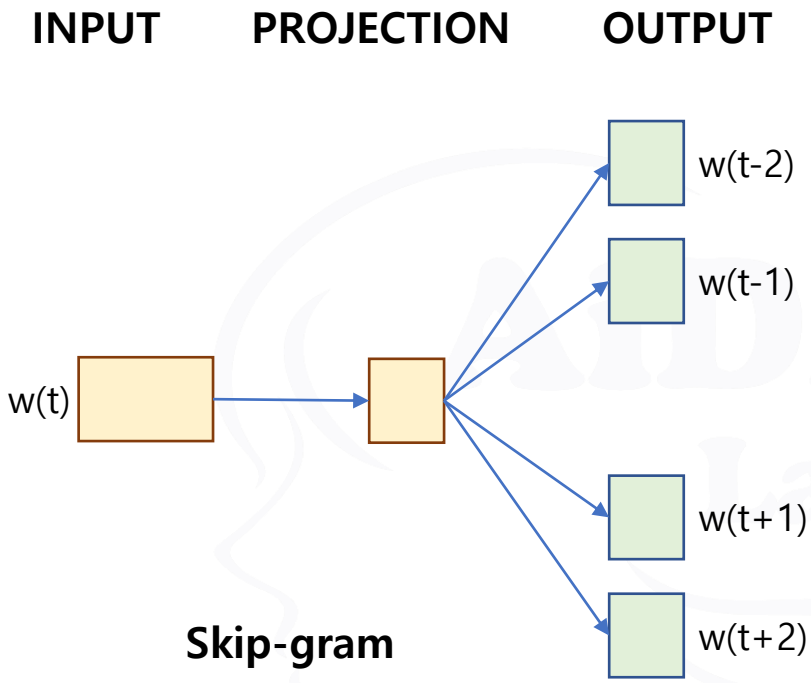
Word2Vec

Word2Vec

- **Word2Vec**
  - **2013, 구글 연구팀이 발표한 기법**
  - **가장 널리 쓰이고 있는 단어 임베딩 모델**
  - **두 개의 논문으로 나누어 발표**
    - Efficient Estimation of Word Representation is Vector Space
      - Skip-gram, CBOW 모델 제안
    - Distributed Representations of Words and Phrase and their Compositionality
      - Skip-gram, CBOW 모델을 근간으로 하되 네거티브 샘플링 등의 학습 최적화 기법 제안

- 모델 기본 구조


INPUT
PROJECTION
OUTPUT
w(t-2)
w(t-1)
SUM
w(t+1)
w(t+2)
w(t)
CBOW


INPUT
PROJECTION
OUTPUT
w(t)
w(t-2)
w(t-1)
w(t+1)
w(t+2)
Skip-gram

- **Skip-gram 모델**
  - **학습 데이터 구축 과정**
    - **한국어 위키백과에서...**

... 개울가에서 속옷 빨래를 하는 남녀 ...

... 개울가 (에서 속옷 빨래 를 하는) 남녀 ...
c_1 c_2 t c_3 c_4

타겟 단어(t)와 그 주변에 실제로 등장한 문맥 단어(c)쌍

포지티브 샘플

t	c
빨래	에서
빨래	속옷
빨래	를
빨래	하는

타겟 단어(t)와 그 주변에 등장하지 않은 단어(말뭉치 전체에서 랜덤 추출) 쌍

네거티브 샘플

t	c
빨래	책상
빨래	안녕
빨래	자동차
빨래	숫자

t	c
빨래	커피
빨래	떡
빨래	사과
빨래	노트북

- 전체 말뭉치를 단어별로 슬라이딩 해 가면서 학습 데이터를 구축

  → 현재는 “빨래”가 타겟 단어이고 “를”은 문맥단어

  → 다음 학습 데이터를 만들 때는 “를 " 이 타겟 단어이고 “빨래”는 문맥 단어

- 이런 방식으로 동일한 말뭉치를 이용해서 매우 많은 학습 데이터 쌍을 만들 수 있음

- **처음 제안된 Skip-gram 모델은 Softmax를 사용하므로 계산량이 비교적 크다**
  - 타겟 단어를 입력 받아서 문맥 단어를 직접 출력하는 모델을 학습하려면
    → 정답 문맥 단어가 나타날 확률은 높이고, 나머지 단어들의 확률은 그에 맞게 낮춰야 함
  - 어휘 집합의 단어 수는 보통 수십만 개를 넘어가므로 → 모두 계산하려면 비효율적

- **네거티브 샘플링 제안**
  - 새로 제안된 Skip-gram 모델은 타겟 단어와 문맥 단어 쌍이 포지티브 샘플(+)인지, 네거티브 샘플(-)인지 이진 분류하는 과정에서 학습됨 → 이런 기법을 네거티브 샘플링이라고 함
  - 1개의 포지티브 샘플과 k개의 네거티브 샘플만 계산하면 되므로 계산량이 대폭 감소함

- 네거티브 샘플 확률

$$P_{negative}(w_i) = \frac{U(w_i)^{3/4}}{\sum_{j=0}^{n} U(w_j)^{3/4}}$$

  - 말뭉치에 용, 미르의 두 단어만 있고 그 구성 비율은 각각 0.99, 0.01 이라고 하면 용과 미르가 네거티브로 뽑힐 확률은

$$P(\text{용}) = \frac{0.99^{0.75}}{0.99^{0.75} + 0.01^{0.75}} = 0.97$$

$$P(\text{미르}) = \frac{0.01^{0.75}}{0.99^{0.75} + 0.01^{0.75}} = 0.03$$

  - 원래 값보다 용은 0.97로 낮아지고 미르는 0.03으로 살짝 높아짐

- 제안 논문에서는 말뭉치에 자주 등장하지 않는 희귀한 단어가 네거티브 샘플로 조금 더 잘 뽑힐 수 있도록 설계함

- **서브 샘플링 제안**
  - 자주 등장하는 단어는 학습에서 제외하는 기법
  - Skip-gram은 엄청나게 많은 학습 데이터 쌍을 만들어 낼 수 있기때문에 고빈도 단어의 경우 등장 횟수만큼 모두 학습시키는 것이 비효율적이라고 판단
  - 서브 샘플링 확률

$$P_{\text{subsampling}}(w_i) = 1 - \sqrt{\frac{t}{f(w_i)}}$$

    - $f(w_i)$: w_i의 빈도, t: 하이퍼 파라미터
    - $f(w_i)$가 0.01로 나타나는 빈도 높은 단어는 $P_{subsampling}(w_i)$가 0.9684
      → 해당 단어가 가질 수 있는 100번의 학습 기회 중 96번 정도는 학습에서 제외함
      → 등장비율이 적어 $P_{subsampling}(w_i)$가 0에 가깝다면 해당 단어가 나올 때마다 빼놓지 않고 학습시킴 → **학습량을 효과적으로 줄여 계산량을 감소시키는 전략 채택**

- **Skip-gram 모델의 학습 (1)**
  - Skip-gram 모델은 타겟 단어와 문맥 단어 쌍이 주어졌을 때
  - 해당 쌍이 포지티브 샘플(+)인지 아닌지를 예측하는 과정에서 학습됨

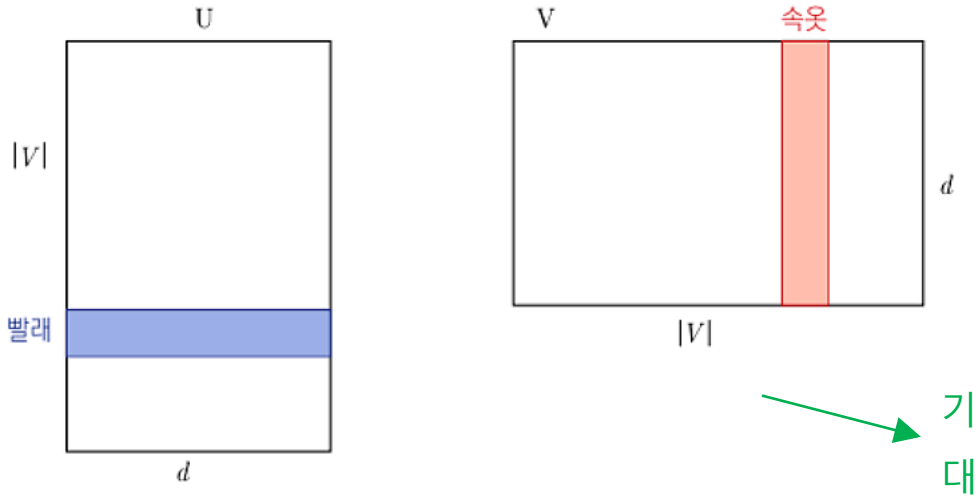
    → 타겟 단어와 문맥 단어 쌍이 실제 포지티브 샘플이라면 조건부 확률을 최대화할 필요가 있음

t, c가 포지티브 샘플(=t 주변에 c가 존재)일 확률

$$P(+|t,c) = \frac{1}{1+\exp(-\mathbf{u}_t\mathbf{v}_c)}$$

  - 즉 모델이 포지티브 샘플 단어 쌍을 입력 받았을 때, 이 쌍이 정말 포지티브 샘플이라고 잘 맞춰야 함

- **Skip-gram 모델의 학습 파라미터는 U와 V 행렬 두개 뿐**
  - **U, V 행렬의 크기는 어휘 집합 크기(|V|) x 임베딩 차원 수(d)로 동일함**
  - **U: 타겟 단어, V: 문맥 단어에 각각 대응**
    - u_i는 타겟 단어(t) 빨래에 해당하는 U의 행 벡터
    - v_c는 문맥 단어(c) 속옷에 해당하는 V의 열 벡터



기존의 NLPM 학습 파라미터와 비교하여 대폭 감소함

Skip-gram 모델 파라미터

- t, c가 포지티브일 확률을 최대화 하려면 분모를 줄여야 함

  → 포지티브 샘플에 대응하는 단어 벡터인 u_t와 v_c의 내적 값을 키워야 함

$$P(+|t,c) = \frac{1}{1+\exp(-\mathbf{u}_t\mathbf{v}_c)}$$

→ 두 벡터의 내적은 코사인 유사도와 비례하므로 → 내적 값의 상향은 포지티브 샘플 t와 c에 해당하는 단어 벡터간 유사도를 높이게 됨(벡터 공간상 가까워짐)

- **Skip-gram 모델의 학습 (2)**
  - Skip-gram 모델은 네커티브 샘플 단어 쌍에 관해 다음의 조건부 확률을 최대화해야 함

t, c가 네거티브 샘플(c를 t와 무관하게 말뭉치 전체에서 랜덤 샘플)일 확률

$$P(-|t,c) = 1 - P(+|t,c) = \frac{\exp(-\mathbf{u}_t\mathbf{v}_c)}{1+\exp(-\mathbf{u}_t\mathbf{v}_c)}$$

  - 즉 네거티브 샘플을 입력하면 모델은 이 데이터가 정말 네거티브 샘플이라고 잘 맞춰야 함
  - t, c가 네거티브일 확률을 최대화 하려면 식 우변의 분자를 최대화해야 함

    → 네거티브 샘플에 대응하는 단어 벡터인 u_t와 v_c의 내적 값을 줄여야 함

    → 두 벡터의 내적은 코사인 유사도와 비례하므로 → 내적 값의 하향은 네거티브 샘플 t와 c에 해당하는 단어 벡터 간 유사도를 낮추게 됨(벡터 공간상 멀어짐)

- **Skip-gram 모델의 학습 (3)**
  - Skip-gram 모델은 로그우도 함수(Log-likelihood function)을 최대화 하여야 함
    - 로그우도 함수는 다중분류에서 많이 사용되는 목적 함수의 하나임

Skip-gram 모델의 로그우도 함수

$$\mathcal{L}(\theta) = \log P(+|t_p, c_p) + \sum_{i=1}^{k} \log P(-|t_{n_i}, c_{n_i})$$

    - 수식에 따르면 모델의 파라미터인 θ를 한 번 업데이트할 때, 1개 쌍의 포지티브 샘플과 k개 쌍의 네거티브 샘플이 학습됨
  - 로그우도 함수를 최대화 하는 과정에서 Skip-gram 모델은 말뭉치의 분포 정보를 단어 임베딩에 함축시키게 됨

- Skip-gram 모델은 로그우도 함수를 계산할 때, 타겟 단어에 해당하는 단어 벡터 u_t는 행렬 U에서, 문맥 단어에 해당하는 단어 벡터 v_c는 행렬 V에서 참조
- 모델 학습이 완료되면
  - U 만 d차원의 단어 임베딩으로 쓸 수도 있고
  - $U + V^T$ 행렬을 임베딩으로 쓸 수도 있으며
  - U 와 V^T를 이어붙여 2d 차원의 단어 임베딩으로 사용할 수도 있음