GloVe

이주형
juhyungee1025@gmail.com
NLP team
2024/03/19

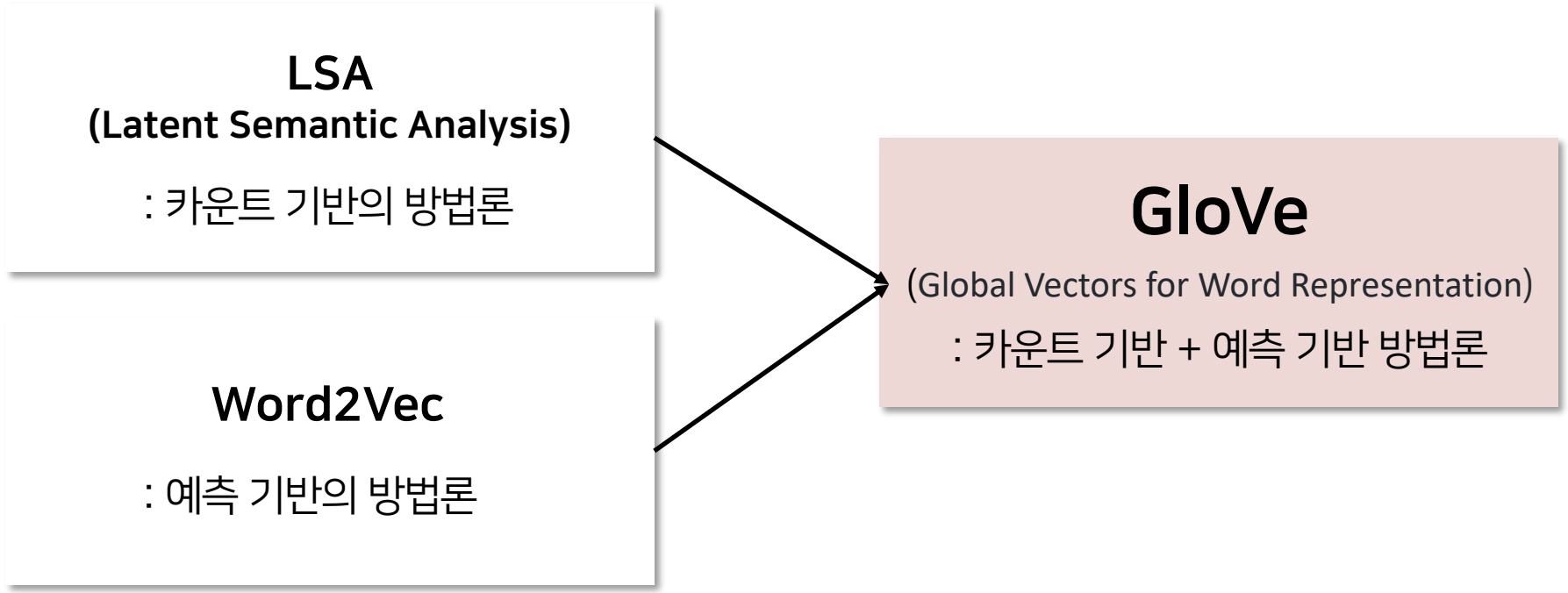
TRAIN AND TEST

Contents

TRAIN AND TEST

Background : Previous Models

Previous Models



TRAIN AND TEST

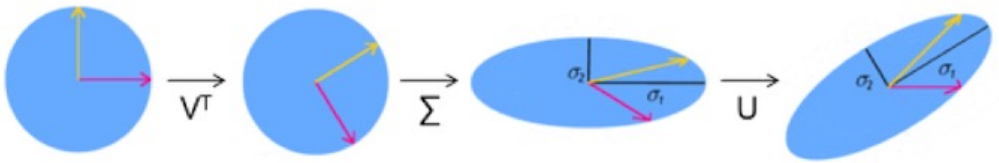
Background : Previous Models

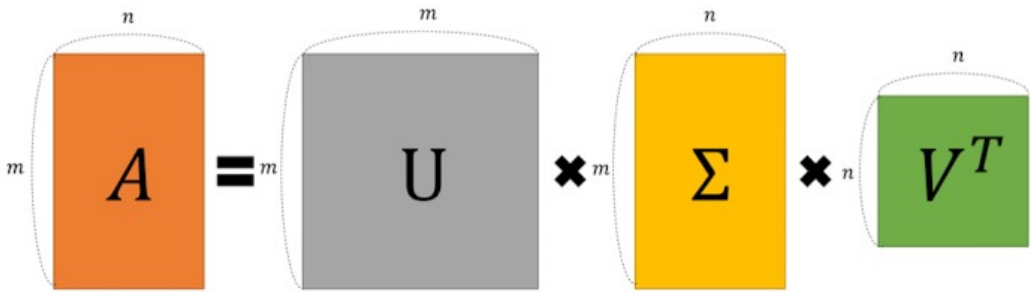
Matrix Factorization Methods

: 코퍼스에 대한 통계적 정보를 담은 큰 행렬들을 decompose

: matrix가 크고 sparse → low rank approximation 이용해서 분해

: 대표적 방법) LSA (Latent Semantic Analysis)





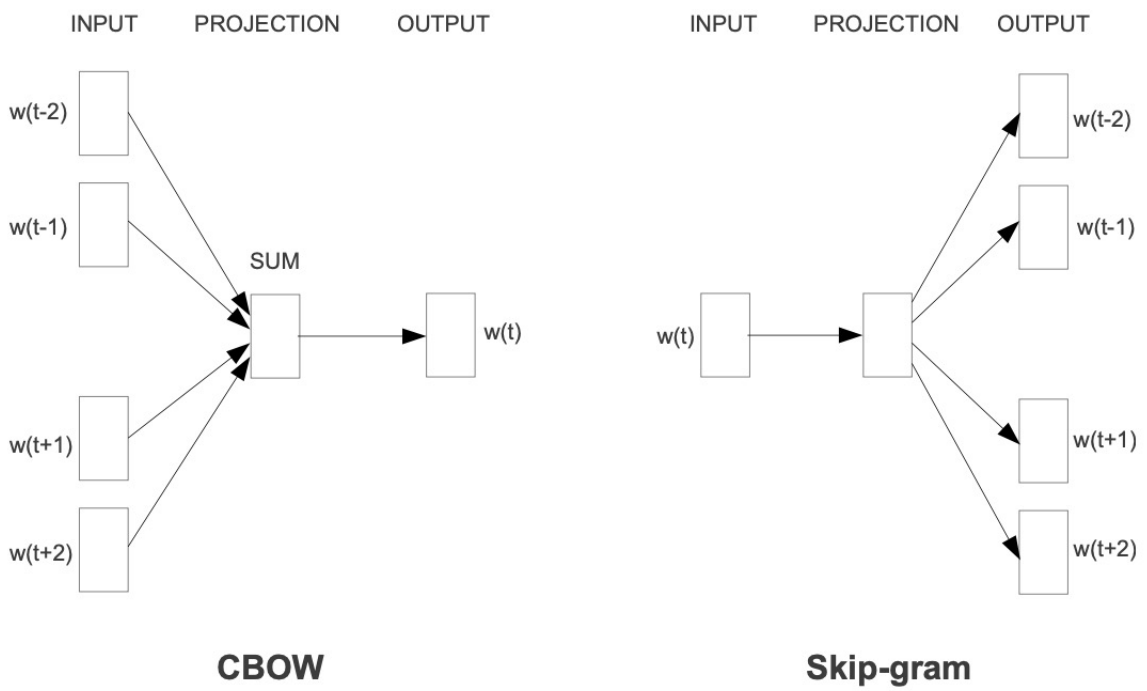
TRAIN AND TEST

Background：Previous Models

Shallow Window-Based Method

: local context window 내에서 단어 표현을 학습하는 것

: 대표적 방법) Word2Vec



TRAIN AND TEST

Background : Previous Models

Previous Models

LSA

장점 : 코퍼스의 전체 통계정보와 co-occurrence 고려

단점 : 단어의 의미 유추에 한계

+

Word2Vec

장점 : 예측기반 → 단어 유추 능력 우수

단점 : 코퍼스의 전체 통계정보와 co-occurrence를 고려하지 못함

코퍼스에서 반복해서 등장하는 data를 제대로 활용하지 못함

⇒ 두 가지 방법론을 모두 사용하는 GloVe

TRAIN AND TEST

GloVe

GloVe : Aim

Aim

GloVe

(Global Vectors for Word Representation)

: 카운트 기반 + 예측 기반 방법론

- **단어 간 유사도를 보존**하면서 **코퍼스 전체의 통계 정보를 반영**
- **동시 등장 확률을 이용해 새로운 목적함수 정의**

→ 두 단어벡터 사이 의미관계 고려, 희소단어 간 정보 보존, 효율성 증대

TRAIN AND TEST

GloVe : 동시 등장 행렬

Window Based Co-occurrence Matrix

: i 단어의 정해진 윈도우 크기 내에서 k 단어가 등장한 횟수를 나타낸 행렬

eg) I like deep learning. / I like NLP. / I enjoy flying.

window size = 1 → 앞뒤 1개의 단어 참고

count	I	like	enjoy	deep	learning	NLP	flying
I	0	2	1	0	0	0	0
like	2	0	0	1	0	1	0
enjoy	1	0	0	0	0	0	1
deep	0	1	0	0	1	0	0
learning	0	0	0	1	0	0	0
NLP	0	1	0	0	0	0	0
flying	0	0	1	0	0	0	0

GloVe : 동시 등장 행렬

Window Based Co-occurrence Matrix

: i 단어의 정해진 윈도우 크기 내에서 k 단어가 등장한 횟수를 나타낸 행렬

eg) I like deep learning. / I like NLP. / I enjoy flying.

window size = 1 → 앞뒤 1개의 단어 참고

count	I	like	enjoy	deep	learning	NLP	flying
I	0	2	1	0	0	0	0
like	2	0	0	1	0	1	0
enjoy	1	0	0	0	0	0	1
deep	0	1	0	0	1	0	0
learning	0	0	0	1	0	0	0
NLP	0	1	0	0	0	0	0
flying	0	0	1	0	0	0	0

Symmetric !

i단어의 윈도우 내에서 k단어 등장 빈도

= k단어의 윈도우 내에서 i단어 등장 빈도

TRAIN AND TEST

GloVe : 동시 등장 확률

Co-occurrence Probability

: 동시 등장 행렬에서 계산한 조건부 확률

Probability and Ratio	$k = solid$	$k = gas$	$k = water$	$k = fashion$
$P(k\|ice)$	1.9×10^{-4}	6.6×10^{-5}	3.0×10^{-3}	1.7×10^{-5}
$P(k\|steam)$	2.2×10^{-5}	7.8×10^{-4}	2.2×10^{-3}	1.8×10^{-5}
$P(k\|ice)/P(k\|steam)$	8.9	8.5×10^{-2}	1.36	0.96

특정 단어 i의 전체 등장 횟수 & 특정 단어 i가 등장했을 때 어떤 단어 k가 등장한 횟수 count

→ 이 둘을 가지고 조건부확률 계산 : $P(k|i)$

TRAIN AND TEST

GloVe : 동시 등장 확률

Co-occurrence Probability

: 동시 등장 행렬에서 계산한 조건부 확률

동시 발생 행렬서 이해한다면?

count	I	like	enjoy	deep	learning	NLP	flying
I	0	2	1	0	0	0	0
like	2	0	0	1	0	1	0
enjoy	1	0	0	0	0	0	1
deep	0	1	0	0	1	0	0
learning	0	0	0	1	0	0	0
NLP	0	1	0	0	0	0	0
flying	0	0	1	0	0	0	0

중심 단어 i의 행의 모든 값을 sum한게 분모, i행 k열 값이 분자

TRAIN AND TEST

GloVe : 동시 등장 확률

Co-occurrence Probability

: 동시 등장 행렬에서 계산한 조건부 확률

Probability and Ratio	$k = solid$	$k = gas$	$k = water$	$k = fashion$
$P(k\|ice)$	1.9×10^{-4}	6.6×10^{-5}	3.0×10^{-3}	1.7×10^{-5}
$P(k\|steam)$	2.2×10^{-5}	7.8×10^{-4}	2.2×10^{-3}	1.8×10^{-5}
$P(k\|ice)/P(k\|steam)$	8.9	8.5×10^{-2}	1.36	0.96

- P(solid | ice) (ice가 등장했을 때 solid가 등장할 확률) = 0.00019
- P(solid | steam) (steam이 등장했을 때 solid가 등장할 확률) = 0.000022

] P(solid I ice) / P(solid I steam) = 8.9

- P(gas | ice) (ice가 등장했을 때 gas가 등장할 확률) = 0.000066
- P(gas | steam) (steam이 등장했을 때 gas가 등장할 확률) = 0.00078

] P(gas I ice) / P(gas I steam) = 0.085

TRAIN AND TEST

GloVe : 동시 등장 확률

Co-occurrence Probability

: 동시 등장 행렬에서 계산한 조건부 확률

Probability and Ratio	$k = solid$	$k = gas$	$k = water$	$k = fashion$
$P(k\|ice)$	1.9×10^{-4}	6.6×10^{-5}	3.0×10^{-3}	1.7×10^{-5}
$P(k\|steam)$	2.2×10^{-5}	7.8×10^{-4}	2.2×10^{-3}	1.8×10^{-5}
$P(k\|ice)/P(k\|steam)$	8.9	8.5×10^{-2}	1.36	0.96

조건부 확률 각각보다 비율로 보는 것이 단어 사이의 관계를 파악하는 데에 도움을 줌

우리의 목적 : 단어 사이의 관계를 잘 설정

→ 조건부 확률의 비율에 관한 손실 함수를 building

GloVe : 손실함수

손실함수

X : 단어-단어의 동시 발생 횟수의 matrix

X_{ij} : 단어 j가 단어 i의 맥락에서 발생한 횟수
(중심단어 i가 등장했을 때 윈도우 내 주변 단어 j가 등장하는 횟수)

$X_i = \Sigma_k X_{ik}$: 동시 발생 행렬서 단어 i의 문맥에서 어떤 단어든 등장한 횟수
(i행의 값을 모두 더한 값)

$P_{ij} = P(j|i) = X_{ij}/X_i$: 단어 j가 단어 i의 맥락에 나타날 확률
(중심단어 i가 등장했을 때 윈도우 내 주변 단어 k가 등장할 확률)

eg) P(solid | ice) = 단어 ice가 등장시 단어 solid가 등장할 확률

$\frac{P_{ik}}{P_{jk}}$: P_{ik}를 P_{jk}로 나눠준 값

eg) P(solid | ice) / P(solid | steam) = 8.9

w_i : 중심단어 i의 임베딩 벡터

$\tilde{w}_k$: 주변단어 k의 임베딩 벡터

TRAIN AND TEST

GloVe : 손실함수

손실함수

$$F(w_i, w_j, \tilde{w}_k) = \frac{P_{ik}}{P_{jk}}$$

$w_i, w_j, \tilde{w_k}$ 를 함수 F에 태우면
Pij/Pjk 가 나온다는 식에서 출발

→ 가능한 F는 무수히 많음
우리는 최적의 F를 찾을 것

$$F(w_i - w_j, \tilde{w}_k) = \frac{P_{ik}}{P_{jk}}.$$

F의 목적 : 단어 벡터 공간에
Pij/Pjk의 비율에 대한 정보 인코딩

→ 벡터간의 차이에만
의존하는 함수로 조작

$$F\left((w_i - w_j)^T \tilde{w}_k\right) = \frac{P_{ik}}{P_{jk}}$$

F의 argument : vector
우변 : scalar

→ 함수의 입력
: 두 값의 내적으로 바꿈

TRAIN AND TEST

GloVe : 손실함수

손실함수

동시 발생 행렬을 위해서는

단어와 문맥 단어 사이의 구분이 정해져 있지 않기 때문에 (random choice)

두 가지의 역할을 자유자재로 바꿀 수 있어야 함.

$$X \leftrightarrow X^T$$

$$w \leftrightarrow \tilde{w}$$

⬇

준동형 (Homomorphism) 도입

$$F(a+b) = F(a)F(b)$$

TRAIN AND TEST

GloVe : 손실함수

손실함수 : 준동형

Probability and Ratio	$k = solid$	$k = gas$	$k = water$	$k = fashion$
$P(k\|ice)$	1.9×10^{-4}	6.6×10^{-5}	3.0×10^{-3}	1.7×10^{-5}
$P(k\|steam)$	2.2×10^{-5}	7.8×10^{-4}	2.2×10^{-3}	1.8×10^{-5}
$P(k\|ice)/P(k\|steam)$	8.9	8.5×10^{-2}	1.36	0.96

우리가 원하는 것 : P(k|ice) / P(k|steam)의 관계를 $F\left((w_i - w_j)^T \tilde{w}_k\right) = \frac{P_{ik}}{P_{jk}}$ 를 이용하여 보존

$$\frac{P(solid|ice)}{P(solid|steam)} = F((ice - steam)^T solid)$$

$$\frac{P(solid|steam)}{P(solid|ice)} = F((steam - ice)^T solid)$$

TRAIN AND TEST

GloVe : 손실함수

손실함수 : 준동형

$$F((ice - steam)^T solid) = \frac{P(solid|ice)}{P(solid|steam)} = \frac{1}{F(steam - ice)^T solid)}$$

✓ 덧셈에 대한 항등원

$$(ice - steam)^T solid = -(steam - ice)^T solid$$

✓ 곱셈에 대한 항등원

$$F((ice - steam)^T solid) = \frac{1}{F((steam - ice)^T solid)}$$

⇒ 준동형 : 덧셈에 대한 항등원이 곱셈에 대한 항등원으로 표현되도록 함

$$(\mathbb{R}, +) \rightarrow (\mathbb{R}, \times)$$

TRAIN AND TEST

GloVe : 손실함수

손실함수 : 준동형

$$w_i^T \tilde{w}_k = (w_i - w_j)^T \tilde{w}_k + w_j^T \tilde{w}_k$$

$$F(w_i^T \tilde{w}_k) = F((w_i - w_j)^T \tilde{w}_k + w_j^T \tilde{w}_k)$$

$$= F((w_i - w_j)^T \tilde{w}_k) \times F(w_j^T \tilde{w}_k)$$

$$F((w_i - w_j)^T \tilde{w}_k) = \frac{F(w_i^T \tilde{w}_k)}{F(w_j^T \tilde{w}_k)} = \frac{P_{ik}}{P_{jk}}$$

TRAIN AND TEST

GloVe : 손실함수

손실함수

$$F((w_i - w_j)^T \tilde{w}_k) = \frac{F(w_i^T \tilde{w}_k)}{F(w_j^T \tilde{w}_k)} = \frac{P_{ik}}{P_{jk}}$$

이 식을 완벽히 만족하는 함수 F : **Exponential 함수**

F를 Exponential 함수로 두고 정리하면,

전 페이지에 정리한 식 참고

$$exp(w_i^T \tilde{w}_k - w_j^T \tilde{w}_k) = \frac{exp(w_i^T \tilde{w}_k)}{exp(w_j^T \tilde{w}_k)}$$

$$exp(w_i^T \tilde{w}_k) = P_{ik} = \frac{X_{ik}}{X_i}$$

$$w_i^T \tilde{w}_k = log(P_{ik}) = log(X_{ik}) - log(X_i)$$

TRAIN AND TEST

GloVe : 손실함수

손실함수

$$w_i^T \tilde{w}_k = log(P_{ik}) = log(X_{ik}) - log(X_i)$$

우리가 주목해야할 부분은 Symmetric한 특성

이 식의 문제점 = 단어 간 교환이 불가능하다 ⇒ $log(X_i)$만 없으면 가능.

⬇

✓ $log(X_i)$ 를 w_i에 대한 편향 b_i로 대체

⬇

w_k에 대해서도 똑같이

$$w_i^T \tilde{w}_k + b_i + \tilde{b}_k = \log(X_{ik})$$

TRAIN AND TEST

GloVe : 손실함수

손실함수

$$w_i^T \tilde{w}_k + b_i + \tilde{b}_k = \log(X_{ik})$$

이 식의 문제점 : $X_{ik} = 0$ 일 수 있음 → 발산 !

✓ $log(X_{ik}) \rightarrow log(1 + X_{ik})$ 로 대체

$$w_i^T \tilde{w}_k + b_i + b_k = log(1 + X_{ik})$$

TRAIN AND TEST

GloVe : 손실함수

손실함수

$$w_i^T \tilde{w}_k + b_i + b_k = log(1 + X_{ik})$$

이 식의 문제점 : 동시 등장 행렬은 **희소행렬**일 가능성이 높음

but 현재의 식은 **모든 동시 발생을 동등하게 가중처리**

↓

✓ X_{ik}의 값에 영향을 주는 가중치 함수 도입 : $f(X_{ik})$

1.f(0) = 0 2. f(x)는 증가하지 않음 3. f(x)는 큰 x값에 대해서 상대적으로 작아야함

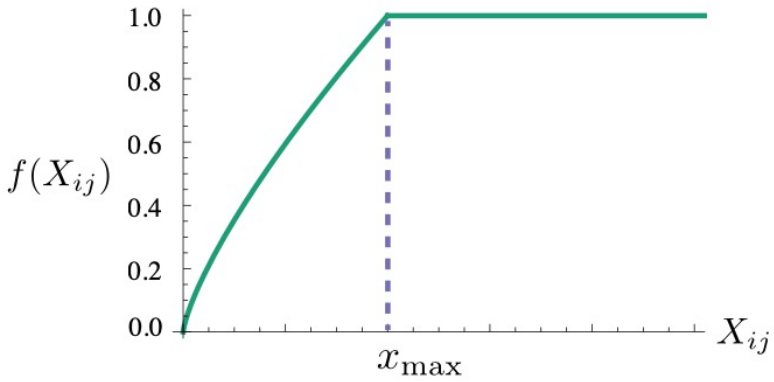
$$J = \sum_{i,j=1}^{V} f\left(X_{ij}\right)\left(w_i^T \tilde{w}_j + b_i + \tilde{b}_j - \log X_{ij}\right)^2$$

TRAIN AND TEST

GloVe : 손실함수

손실함수

$$f(x) = \begin{cases} (x/x_{\max})^{\alpha} & \text{if } x < x_{\max} \\ 1 & \text{otherwise} \end{cases}$$



- 1보다 큰 값을 반환하지 않도록 0~1사이의 값을 가짐
- 가장 높은 동시 출현 횟수보다 동시 출현 횟수가 큰 단어쌍에 대해 가중치 제한

→ 문서에 자주 등장하는 단어가 학습에 과도한 영향을 미치는 것 방지

TRAIN AND TEST

GloVe : 손실함수

손실함수

✓ GloVe의 손실함수

$$f(x) = min(1,\ (x/x_{max})^{3/4})$$

$$J = \sum_{i,j=1}^{V} f\left(X_{ij}\right)\left(w_i^T \tilde{w}_j + b_i + \tilde{b}_j - \log X_{ij}\right)^2$$

TRAIN AND TEST

참조

https://wikidocs.net/22885

https://blog.naver.com/jujbob/221155564430

https://sumim.tistory.com/entry/NLP-%EA%B7%BC%EB%B3%B8-%EB%85%BC%EB%AC%B8-1-GloVe-Global-Vectors-for-Word-Representation

https://youtu.be/JZI74rrMb_M?si=6iejYd8fObANkrj8

https://angeloyeo.github.io/2019/08/01/SVD.html#google_vignette

TRAIN AND TEST