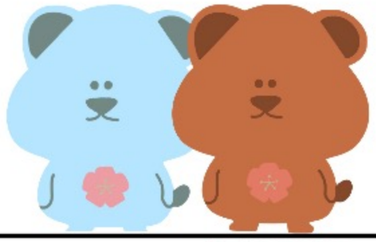
Word Vectors and Word Senses

Week2_발표자: 김나현, 김소민

EWHA EURON

목차

1. Skip-gram model with negative sampling
2. Word prediction method
3. GLOVE
4. Evaluating Word Vectors
5. Word Senses & Word Senses Ambiguity



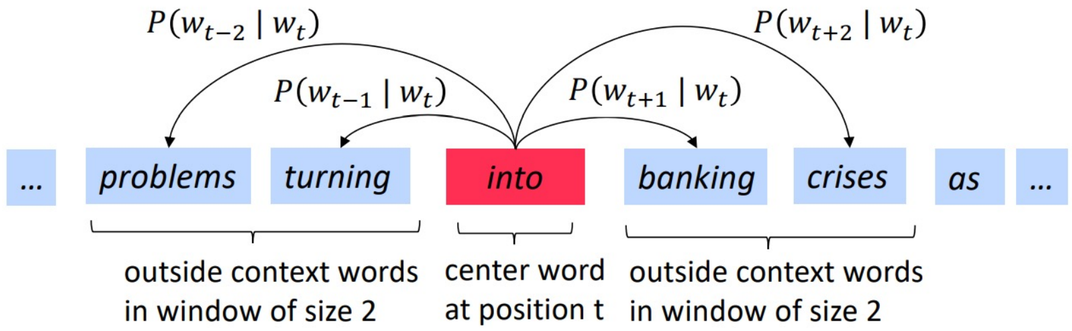
EWHA EURON

Word vectors, Word2vec 마무리

EWHA EURON

Review: word2vec

#1 Word2vec 주요개념

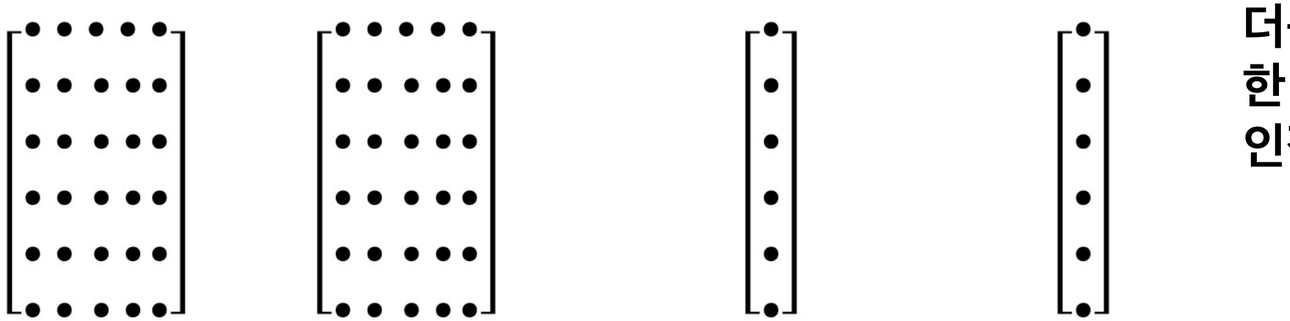


벡터 공간 내에 유사한 단어들을 가까이 위치시킴으로써 Objective Function을 maximize 함.

벡터 공간
사물이 서로 가까이 붙어 비슷한 의미를 갖는 것을 보여주는 공간
특정 의미를 지닌 방향도 보여주는 공간

더욱 고차원적인 벡터 공간에서는
한 단어가 다양한 다른 단어와 서로 다른 방향으로 인접할 수 있다.

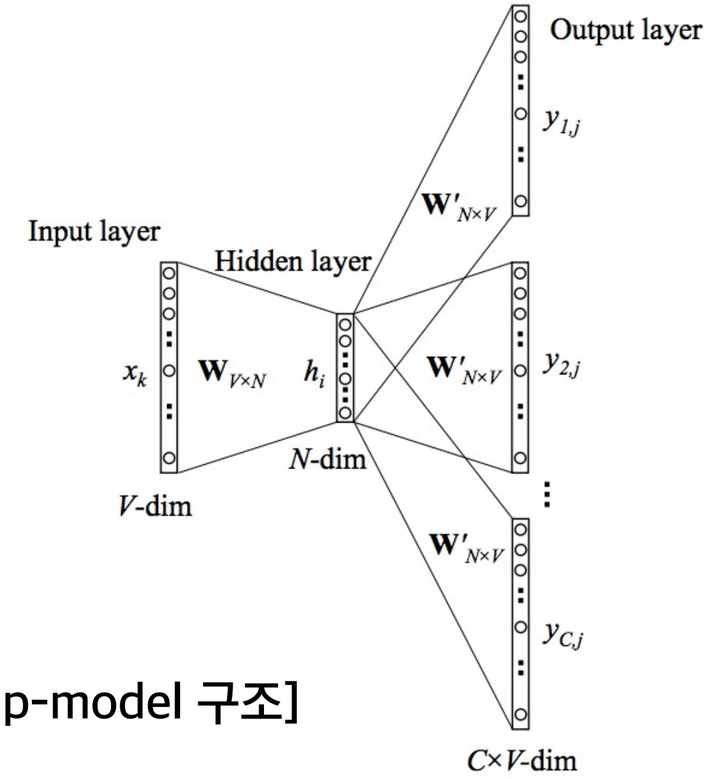
#2 Word2vec의 parameter 와 computation 방식



Skip-gram model with negative sampling

#1 Skip-grams 방식과 CBOW 방식

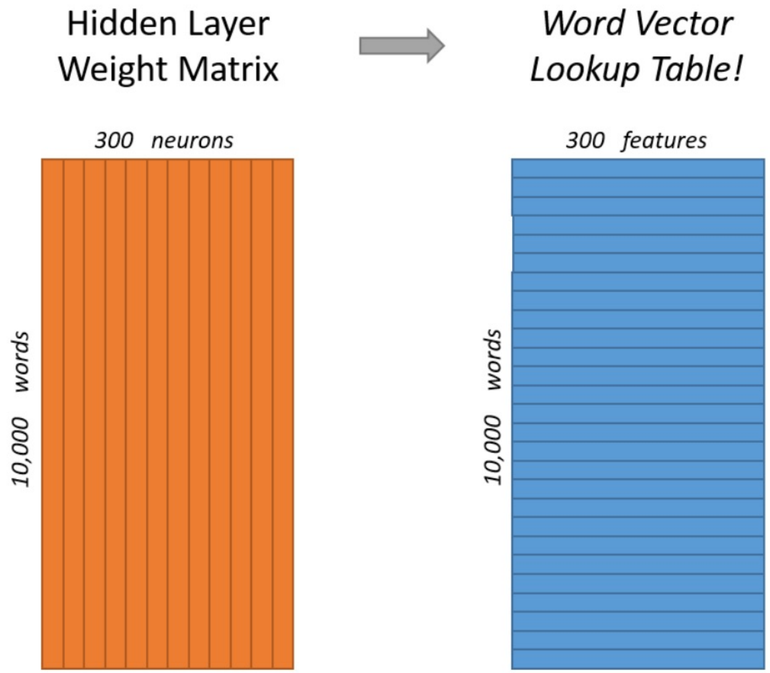
CBOW: 복수의 단어의 중심에 있는 단어를 맞추는 방식
Skip-Gram: 특정한 단어로부터 주변에 있는 단어를 맞추는 방식



[Skip-model 구조]

Weight matrix W,W′을 수정해가면서 학습

W: one-hot-encoding된 입력벡터와 hidden layer을 연결, word2vec의 최종 결과물인 임베딩 단어벡터 모음.



Skip-gram model with negative sampling

max

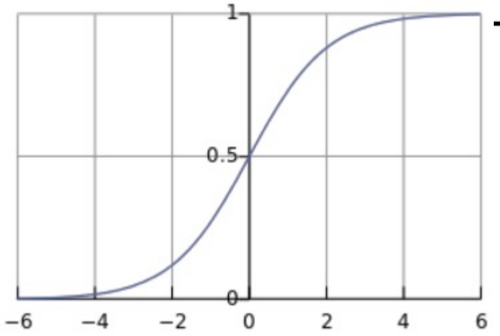
$$\log \sigma({v'_{w_O}}^T v_{w_I}) + \sum_{i=1}^{k} \mathbb{E}_{w_i \sim P_n(w)} \left[\log \sigma(-{v'_{w_i}}^T v_{w_I})\right]$$

min

유사한 context의 단어들은 유사한 벡터로 표현되고,
다른 context에서 발견된 단어들은 서로 유사도가
떨어지는 word vector를 갖도록 함.

[Sigmoid function]

$$\sigma(x) = \frac{1}{1+e^{-x}}$$



$$P_n(w) = \frac{U(w)^{\frac{3}{4}}}{Z}$$

(Z는 normalization constant)

Word2Vec
: 출력층이 내놓는 값에 softmax 함수를 적용해 확률값으로 변환한 후, 이를 정답과 비교해 역전파(backpropagation)하는 구조.

But! Softmax 를 적용하는 데 어마어마한 계산량이 요구됨.
(모든 단어의 dot product 후, exp까지 취해줘야 함)

Softmax 확률을 구할 때 전체 단어를 대상으로 구하는 대신에,
일부 단어만 뽑아서 계산 → Negative Sampling

Negative Sampling의 절차
- 사용자가 지정한 윈도우 사이즈 내에 등장하지 않는 단어(negative sample)를 5~20개 정도 뽑는다.
- Negative Sample을 정답 단어와 합쳐서 전체 단어처럼 softmax 확률을 구한다.

https://ratsgo.github.io
https://www.baeldung.com/cs/nlps-word2vec-negative-sampling

Skip-gram model with negative sampling

Skip-Gram with Negative Sampling, SGNS

Negative Sampling

하나의 중심 단어에 대해서 전체 단어 집합보다 훨씬 작은 단어 집합을
만들어놓고 마지막 단계를 이진 분류 문제로 변환한다.
주변 단어들을 긍정(positive), 랜덤으로 샘플링 된 단어들을
부정(negative)으로 레이블링한다면 이진 분류 문제를 위한 데이터셋이 된다.

이는 기존의 단어 집합의 크기만큼의 선택지를 두고 다중 클래스 분류 문제를
풀던 Word2Vec보다 훨씬 연산량에서 효율적이다.

'cat'이라는 중심단어를 통해 'sat'이라는
주변단어 예측

중심단어와 주변단어 모두 입력,
두 단어가 실제로 윈도우 크기 내에
존재하는 이웃 관계인지 그 확률을 예측

https://wikidocs.net/69141

주변단어 중심단어

The fat cat sat on the mat

입력과 레이블의 변화

입력1	입력2	레이블
cat	The	1
cat	fat	1
cat	sat	1
cat	on	1

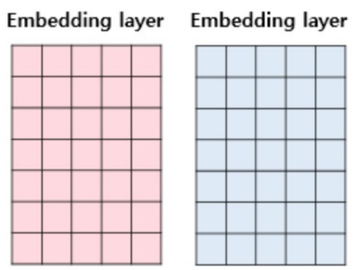
Negative Sampling

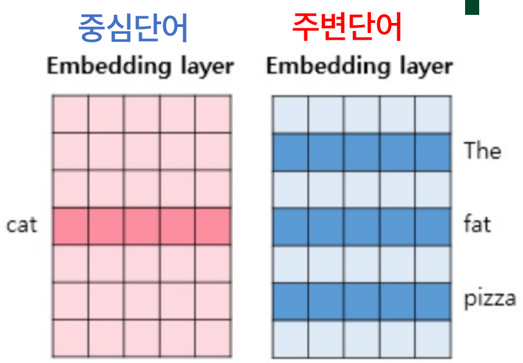
입력1	입력2	레이블
cat	The	1
cat	fat	1
cat	**pizza**	**0**
cat	**computer**	**0**
cat	sat	1
cat	on	1

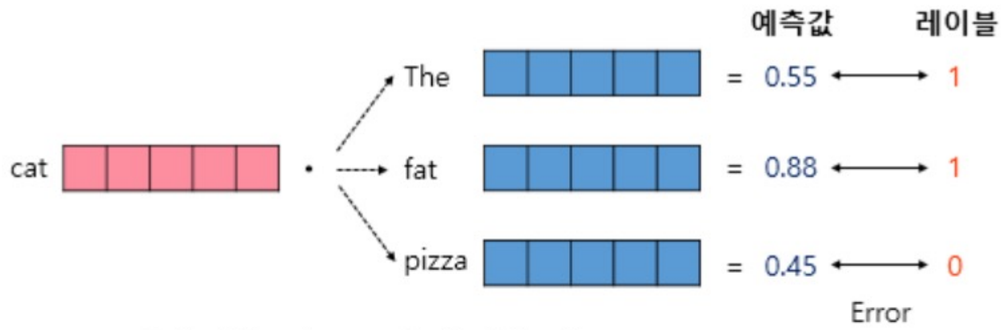
단어 집합에서 랜덤으로 선택된 단어들을 레이블 0의 샘플로 추가.

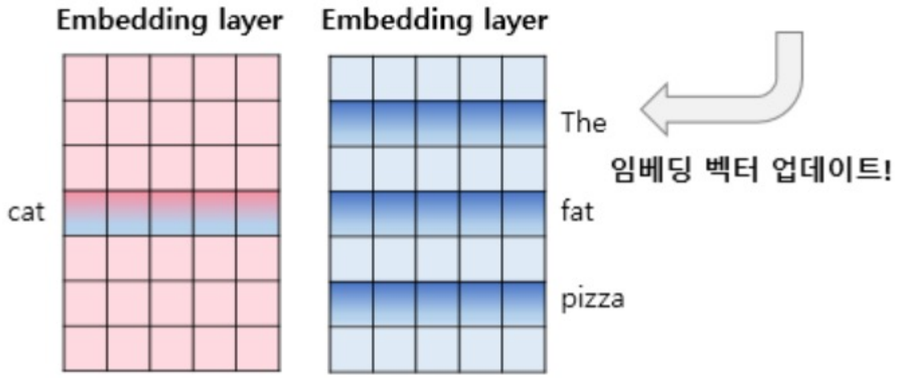
입력1과 입력2가 실제로 윈도우 크기 내에서 이웃 관계인 경우에는 레이블이 1, 아닌 경우에는 레이블이 0인 데이터셋이 된다.

입력1	입력2	레이블
cat	The	1
cat	fat	1
cat	pizza	0
cat	computer	0
cat	sat	1
cat	on	1
cat	cute	1
cat	mighty	0
...	...	...









https://wikidocs.net/69141

Co-occurrence matrix

< window based co-occurrence matrix >

- I like deep learning.
- I like NLP.
- I enjoy flying.

counts	I	like	enjoy	deep	learning	NLP	flying	.
I	0	2	1	0	0	0	0	0
like	2	0	0	1	0	1	0	0
enjoy	1	0	0	0	0	0	1	0
deep	0	1	0	0	1	0	0	0
learning	0	0	0	1	0	0	0	1
NLP	0	1	0	0	0	0	0	1
flying	0	0	1	0	0	0	0	1
.	0	0	0	0	1	1	1	0

Co-occurrence counts 를 바탕으로
벡터 간 유사도 측정
→ 그렇지만 비효율적이라 느껴지지 않나?

: 전체 corpus 에 대한 co-occurrence 정보를 사용하는 방법.

특정 크기의 window 내에 나타나는 단어들을 모두 count
- A라는 단어가 B에 인접하게 몇 번 사용되었는지 확인해서 matrix를 구성한다.

유사한 쓰임/의미를 보유하고 있는 단어들끼리는
비슷한 벡터 구성을 보유하게 된다.
- 비슷한 단어들은 비슷한 환경/문맥에서 사용되기 때문에, 비슷한 단어들과 인접하게 된다.

문제점

sparse matrix를 형성하게 된다.
단어 수가 많아지는 경우 matrix의 크기가 커진다.

"Low dimensional vectors"
높은 co-occurrence counts 행렬 남겨두기 + dimensionality 축소

차원 축소: 특이값 분해 (SVD)

실제로 행렬 A는 특이값을 대각 원소로 하는 대각 행렬(Σ)과 2개의 직교 행렬 U와 V로 분해할 수 있으며, 이 방법을 [특이값 분해(SVD)]라고 한다.

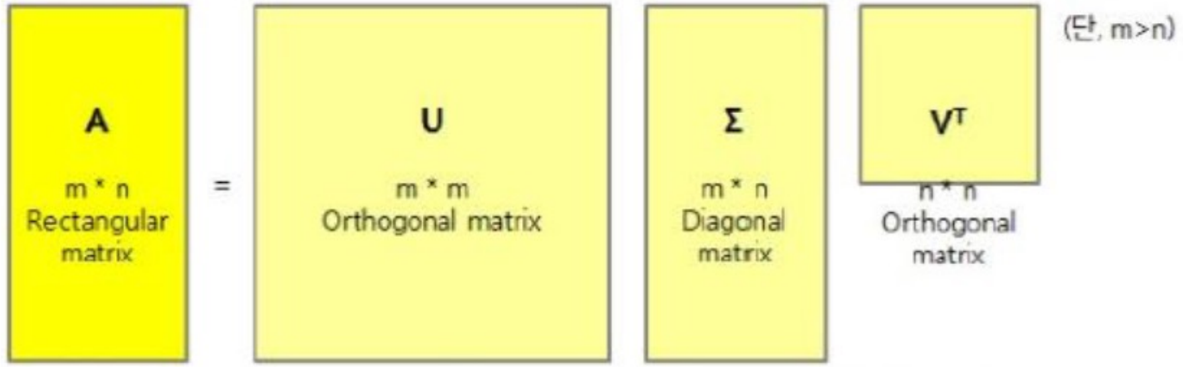
$$A = U\Sigma V^T$$

A : m * n 직사각행렬(m by n rectangular matrix)

U : A의 left singular vector로 이루어진 m * m 직교행렬(orthogonal matrix)

Σ : 주대각성분이 $\sqrt{\lambda_i}$로 이루어진 $m * n$ 직사각대각행렬(diagonal matrix)

V : A의 right singualr vector로 이루어진 n * n 직교행렬(orthogonal matrix)

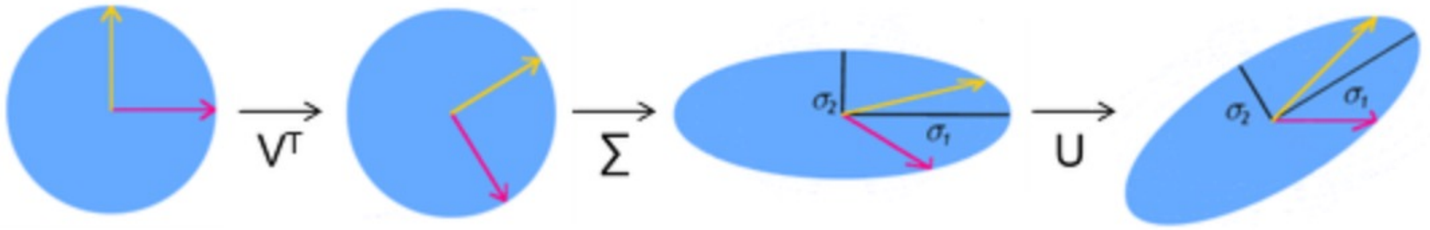


U와 V는 직교행렬이므로 아래 조건 만족

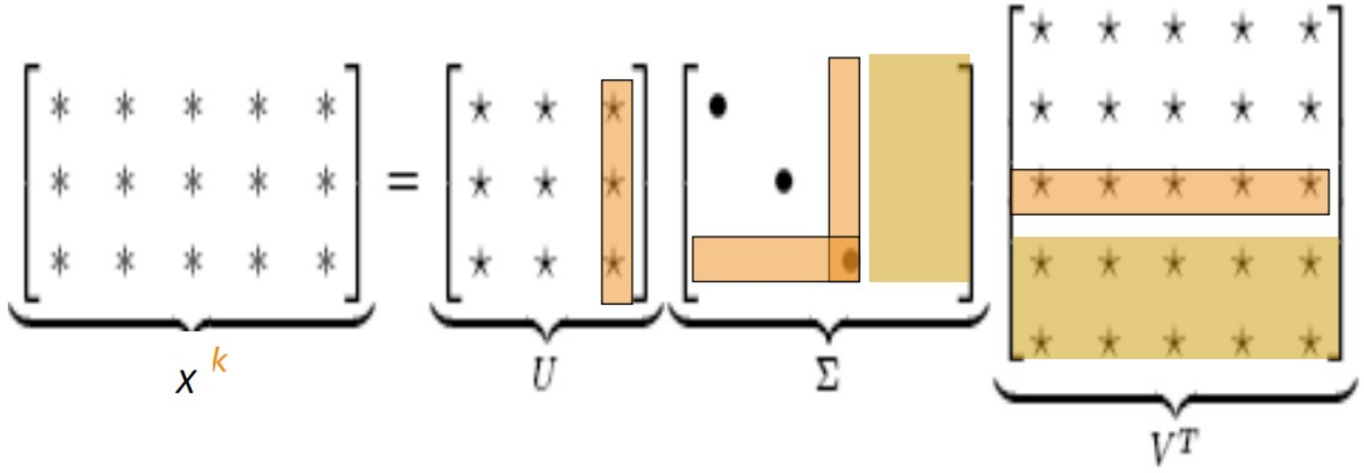
$$UU^T = VV^T = E, \quad U^{-1} = U^T, \quad V^{-1} = V^T$$

https://movefast.tistory.com/308
https://blog.naver.com/PostView.nhn?blogId=tjdqja0508&logNo=222298714

차원 축소: 특이값 분해 (SVD)



벡터 x를 행렬 A로 변환했을 때, Ax는 x를 먼저 V^T에 의해 회전시킨 후
Σ로 스케일을 변화시키고 다시 U로 회전시킨다.
(*도형의 모양을 바꾸는 것은 오로지 특이값에 의해서만 결정이 된다.)



가장 작은 특이값을 0으로 만든다는 것은 원본 행렬 A에 가장 근사하면서도
불필요한 특징(noisy feature)을 제거한 행렬 B를 얻을 수 있다는 의미.
동시에 두 개의 직교 행렬의 벡터들을 사용하지 않으니 차원을 축소했다고도 볼 수 있다.

Word prediction method

Count based Prediction

- LSA, HAL, COALS, Hellinger-PCA 등
- 빠른 훈련 속도
- 단어의 유사도 계산하는 데에 주로 쓰임
  (단어 간 관계는 고려 X)
- 복잡한 패턴은 인식할 수 없음.
- 큰 빈도수에 과도하게 중요도를 부여함.

Direct Prediction

* Nerual networks method

- NNLM, HLBL, RNN, Skip-gram, CBOW 등
- 전체적인 통계 정보를 활용하는 대신에, window 내의 데이터만 사용함.
- 단어의 유사도에 대해서 복잡한 패턴도 인식할 수 있음.
- 대부분의 영역에서 좋은 성능을 내는 모델 생성

GLoVe model of Word Vectors

EWHA EURON

GLoVe - 등장배경

Count Method
⇒ LSA
⇒ 코퍼스의 전체적인 통계 정보 고려
⇒ 유추(Analogy) 성능 떨어짐

Direct Prediction Method
⇒ Word2Vec
⇒ 코퍼스의 전체적인 통계 정보 고려 x
⇒ 유추 성능은 LSA보다 뛰어남

↓

GLoVE
(Global VEctors for Word Representation)

GLoVe - Insight

Ratios of Co-occurrence probabilities (동시 등장 확률)
can encode meaning components

$P(k \mid i)$ (동시 등장 확률)이란?
➔ 동시 등장 행렬로부터 특정 단어 i의 전체 등장 횟수를 카운트하고,
특정 단어 i가 등장했을 때 어떤 단어 k가 등장한 횟수를 카운트하여 계산한 조건부 확률

GLoVe - Insight

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

동시발생 확률의 비율의 특성을 이용하여
단어 벡터 생성을 해보자

↓

목표: 임베딩 된 중심 단어와 주변 단어 벡터의 내적이
전체 코퍼스에서의 동시 등장 확률(로그값) 이 되도록 임베딩 벡터를 만드는 것

GLoVe - 목적함수 도출하기

목표: 임베딩 된 중심 단어와 주변 단어 벡터의 내적이 전체 코퍼스에서의 동시 등장 확률의 로그값이 되도록 목적함수 (F)를 정의하는 것

$$dot\ product(w_i, \tilde{w}_k) \approx log\ P(k \mid i) = log\ P_{ik}$$

W_i : 중심 단어 i의 임베딩 벡터
W_k : 주변 단어 k의 임베딩 벡터
P_{ik} : 중심 단어 i가 등장했을 때 윈도우 내 주변 단어 k가 등장할 확률

GLoVe - 목적함수 도출하기

목표: 임베딩 된 중심 단어와 주변 단어 벡터의 내적이 전체 코퍼스에서의 동시 등장 확률의 로그값이 되도록 목적함수 (F)를 정의하는 것

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

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

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

⇒ 선형 공간(Linear space)에서 단어의 의미 관계를 표현하기 위해 뺄셈과 내적을 선택함

⇒ GLoVe 모델의 3가지 조건

1. Center word는 Context word로 등장할 수 있기 때문에, 벡터간 교환 법칙이 성립해야 함

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

2. Co-occurrence matrix는 symmetric 해야 함

$$X \leftrightarrow X^T$$

3. 목적함수 F 는 Homomorphism을 만족해야 함

$$F(X + Y) = F(X) \times F(Y)$$

GLoVe - 목적함수 도출하기

Homomorphism 때문에 F = e^x 사용함

$$F\left((w_i - w_j)^T \tilde{w}_k\right) = F\left(w_i^T \tilde{w}_k - w_j^T \tilde{w}_k\right) = \frac{F(w_i^T \tilde{w}_k)}{F(w_j^T \tilde{w}_k)} \quad \because e^{(x-y)} = {e^x}/{e^y}$$

교환법칙 성립을 위해 (위치가 바뀌어도 성립이 되어야 함) bias 추가됨

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

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

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

$$w_k^T \tilde{w}_i = log(P_{ki}) = log(X_{ki}) - log(X_k)$$

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

$$Loss\ function = \sum_{m,n=1}^{V} (w_m^T \tilde{w}_n + b_m + \tilde{b_n} - logX_{mn})^2$$

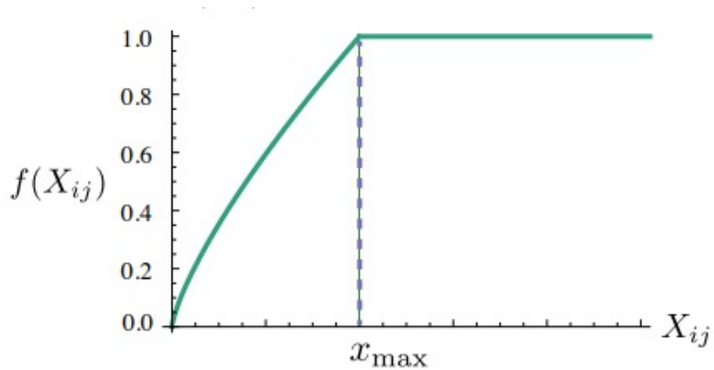
GLoVe - 손실함수

But, 다 끝난게 아니다

1. $\log X_{mn} = 0$ 이 되는 경우 빈번 (X: sparse matrix일 가능성 높음)
2. 너무 빈번한 단어 (is, the ..)를 대비한 가중치 조절을 위해

⇒ **가중치 함수 추가한 최종 손실 함수**

$$Loss\ function = \sum_{m,n=1}^{V} f(X_{mn})(w_m^T \tilde{w}_n + b_m + \tilde{b_n} - logX_{mn})^2$$



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

GLoVe - 장점과 단점

+ Global statistical Information의 효율적 사용
+ 빠른 학습 속도
+ Size가 큰 Corpora에 대해서도 확장성 갖음
+ Small corpus, Small vector size 에서도 좋은 성능 보임

- co-occurrence matrix & global information 을 사용하기 때문에 Word2Vec에 비해 메모리 cost가 더 높음
- Polysemous word (다의어) 문제를 해결하지 못함

GLoVe - Hyperparameters

❖ Hyperparameter: 사용자가 직접 세팅해주는 파라미터를 의미

```
from glove import Corpus, Glove
corpus = Corpus()

# 훈련 데이터로부터 GloVe에서 사용할 동시 등장 행렬 생성
corpus.fit(result, window=5)
glove = Glove(no_components=30, learning_rate=0.05, alpha=0.75, max_count=100, max_loss=10.0, random_state=None)
```

- no_components: word vector의 dimension 크기 설정
- Learning_rate: 학습 속도 설정 (SGD estimation시 사용)
- Alpha, max_count: weight 부여할 때 사용
- Random_state: optimization시 초기화 때 사용되는 상태

GLoVe - 결과

```
pip install glove_python_binary
```

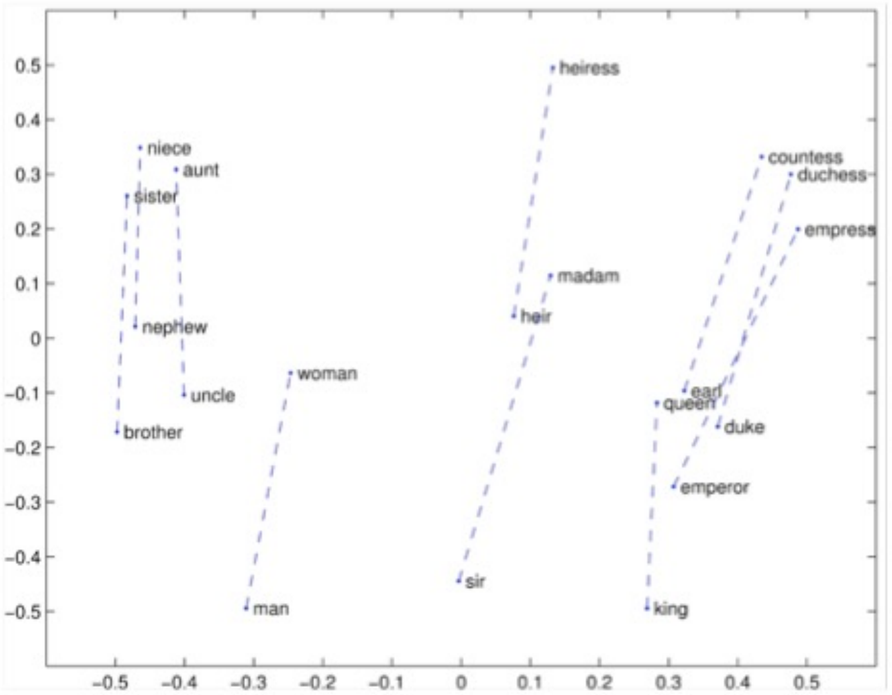
```
print(glove.most_similar("man"))
```

```
[('woman', 0.9621753707315267), ('guy', 0.8860281455579162), ('girl', 0.8609057388487154), ('kid', 0.8383640509911114)]
```

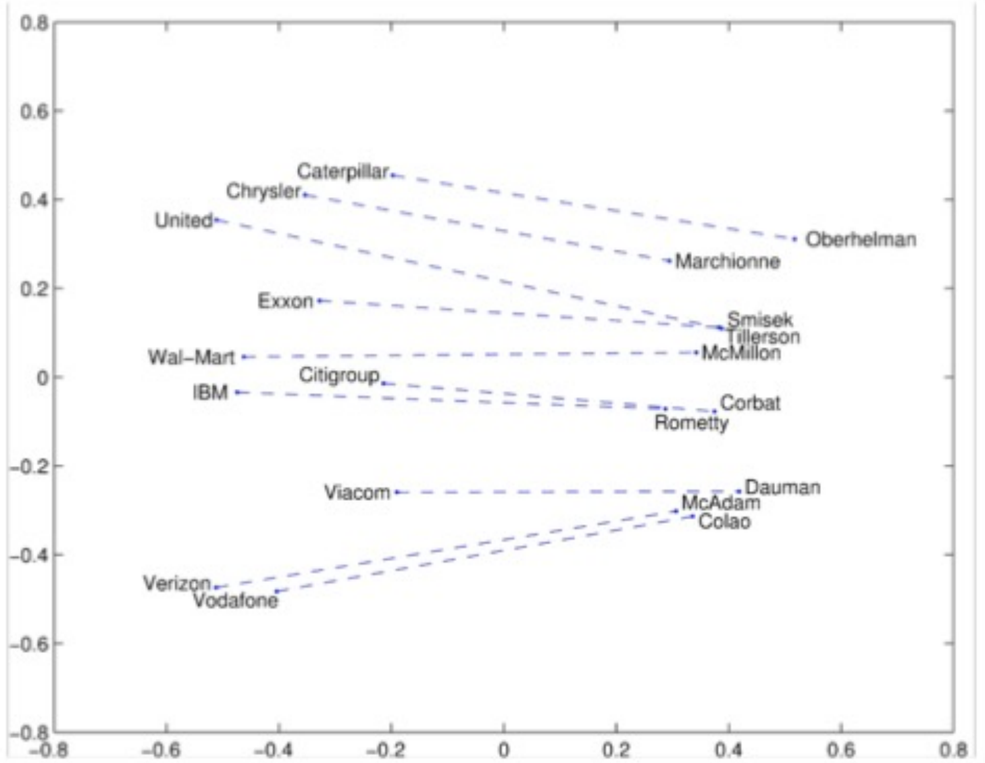
```
print(glove.most_similar("physics"))
```

```
[('chemistry', 0.8379143027061764), ('biology', 0.827856517644139), ('economics', 0.775563255616767), ('finance', 0.7736692309034663)]
```

GLoVe - Visualizations (시각화)



Gender Display



Company - CEO

How To Evaluate Word Vectors

EWHA EURON

Evaluating Word Embedding in NLP

Intrinsic Evaluation (내적 평가)

중간 단계의 특정한 subtask을
기준으로 측정
Ex) 단어 간의 유사성 측정
+ 계산 속도가 빠름
+ 해당 시스템을 이해하기에 편리함
- 현실에서 해당 시스템이 유용한지
판단하기 어려움

Extrinsic Evaluation (외적 평가)

실제로 상용되는 애플리케이션에 embedding을
직접 사용하여 성능을 측정하는 방식
Ex) 검색 (Web Search) ..
+ 현실에서 해당 시스템의 유용성 판단 가능
- 계산 속도가 느림
- 해당 시스템의 문제 or 다른 시스템의 문제
or 상호작용의 문제 인지 알 수 없음

GLoVe - 내적 평가

Intrinsic Evaluation (내적 평가)
- word analogies (단어 유추)

Model	Dim.	Size	Sem.	Syn.	Tot.
ivLBL	100	1.5B	55.9	50.1	53.2
HPCA	100	1.6B	4.2	16.4	10.8
GloVe	100	1.6B	67.5	54.3	60.3
SG	300	1B	61	61	61
CBOW	300	1.6B	16.1	52.6	36.1
vLBL	300	1.5B	54.2	64.8	60.0
ivLBL	300	1.5B	65.2	63.0	64.0
GloVe	300	1.6B	80.8	61.5	70.3
SVD	300	6B	6.3	8.1	7.3
SVD-S	300	6B	36.7	46.6	42.1
SVD-L	300	6B	56.6	63.0	60.1
CBOW†	300	6B	63.6	67.4	65.7
SG†	300	6B	73.0	66.0	69.1
GloVe	300	6B	77.4	67.0	71.7
CBOW	1000	6B	57.3	68.9	63.7
SG	1000	6B	66.1	65.1	65.6
SVD-L	300	42B	38.4	58.2	49.2
GloVe	300	42B	**81.9**	**69.3**	**75.0**

Google Analogy Test Set

Word Analogies
- A -> B , C -> D (D를 유추하는것이 목표)

Ex) Woman -> Queen, Man -> ?

1. Syntatic (Grammar)
2. Semantic (Meaning)

=> GLoVe가 제일 성능이 좋다

GLoVe - 외적 평가

Real task에 적용하여 평가
⇒ 사람 or 조직 or 지역 찾기

Model	Dev	Test	ACE	MUC7
Discrete	91.0	85.4	77.4	73.4
SVD	90.8	85.7	77.3	73.7
SVD-S	91.0	85.5	77.6	74.3
SVD-L	90.5	84.8	73.6	71.5
HPCA	92.6	**88.7**	81.7	80.7
HSMN	90.5	85.7	78.7	74.7
CW	92.2	87.4	81.7	80.2
CBOW	93.1	88.2	82.2	81.1
GloVe	**93.2**	88.3	**82.9**	**82.2**

=> GLoVe 성능이 좋다

Word Senses and Word Sense Ambiguity

EWHA EURON

Word Senses & Word Sense Ambiguity

- 단어들은 다양한 의미를 가지고 있음

Ex) Pike - 1. Fish 2. Pull out of doing something 3. Diving Move 4. Road … etc

단어들의 다양한 의미를 표현할 수 있는 방법?

-> Embedding Polysemy(다형성)

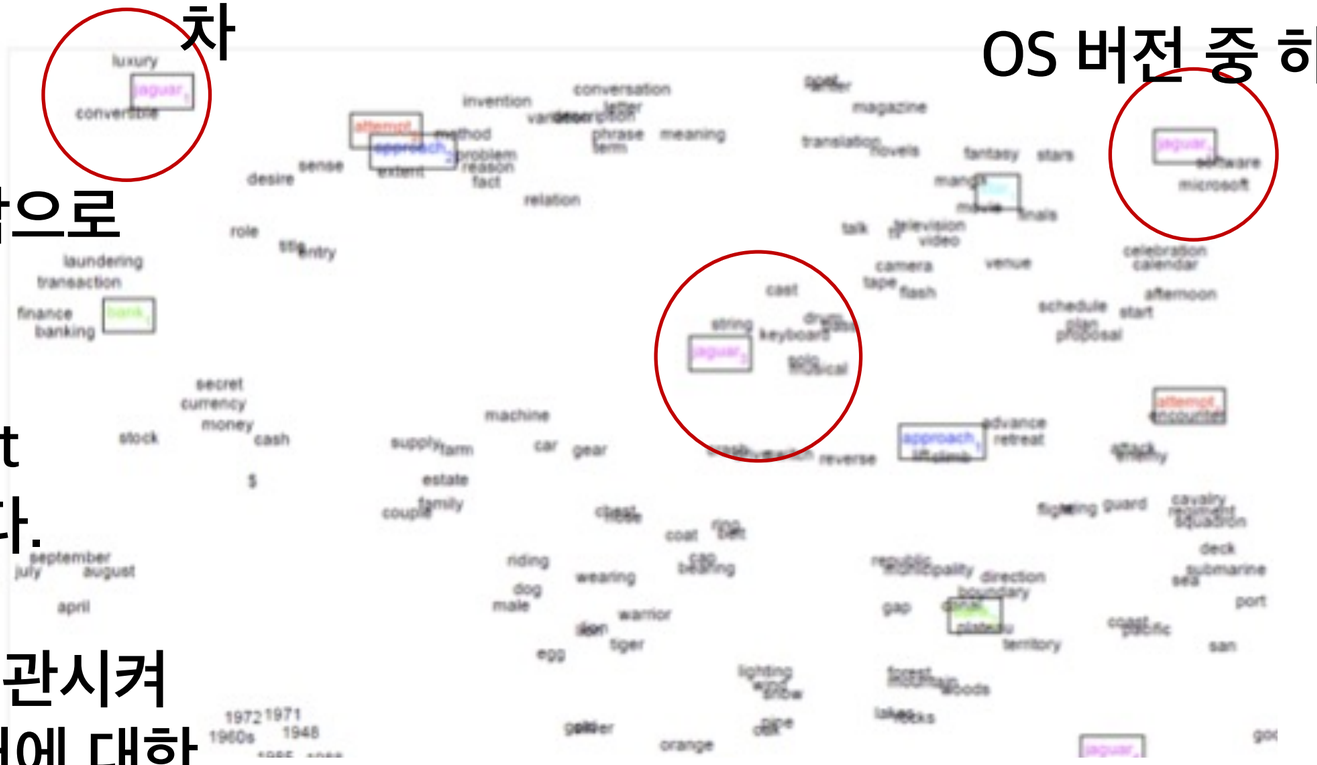
Embedding Polysemy

Key Idea

1. 단어마다 각각의 context의 가중합으로 represent 한다.

2. K-means clustering 으로 context representation을 clustering 한다.

3. 각각의 단어들에 대해 cluster와 연관시켜 re-labelling을 하고, 해당 클러스터에 대한 word representation을 학습한다.



Embedding Polysemy

Linear Algebraic Structure of Word Senses, with Applications to Polysemy
- Arora et al. 2018

Key-idea

Word2vec 같은 단어 embedding에는 단어의 다양한 의미가 linear-superposition(선형 중첩)되어 존재한다.

$$v_{\text{pike}} = \alpha_1 v_{\text{pike}_1} + \alpha_2 v_{\text{pike}_2} + \alpha_3 v_{\text{pike}_3}$$

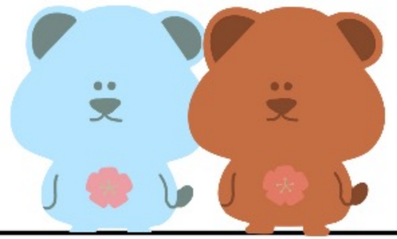
Where $\alpha_1 = \frac{f_1}{f_1+f_2+f_3}$, etc., for frequency f

Result

tie				
trousers	season	scoreline	wires	operatic
blouse	teams	goalless	cables	soprano
waistcoat	winning	equaliser	wiring	mezzo
skirt	league	clinching	electrical	contralto
sleeved	finished	scoreless	wire	baritone
pants	championship	replay	cable	coloratura

넥타이 케이블 타이

THANK YOU



EWHA EURON