



Word Vectors, Word Senses, and Neural Classifiers

김경민, 문예지

목차

#01 Word vectors and word2vec

#02 Optimization : Gradient Descent

#03 The GloVe model of word vectors

#04 Evaluating word vectors

#05 Word senses



Word vectors and word2vec

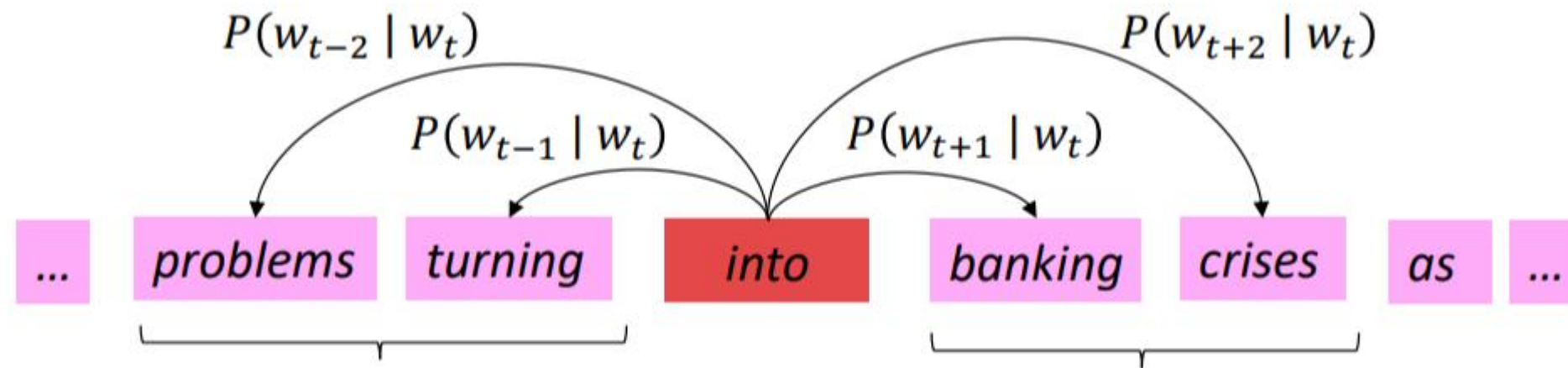


#01 Word vectors and word2vec

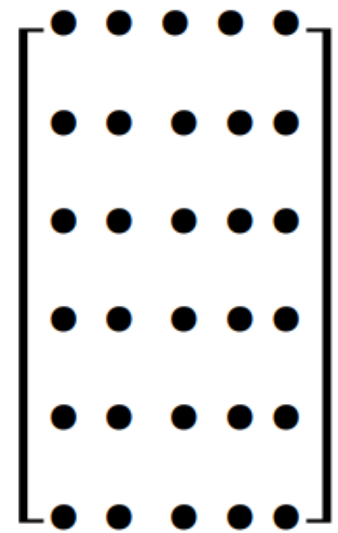
#1 주요 feature : 중심 단어(center word), 맥락 단어(context word), 윈도우(window)

#2 의미적으로 유사성을 가진 단어들은 서로 가까운 위치에 존재한다는 아이디어 이용
-> 중심단어를 기준으로 양쪽으로 윈도우 크기만큼의 단어를 맥락 단어로 설정
-> 이를 입력 벡터와 출력 벡터로 사용

#3 단어들에 대한 확률 분포는 단순히 단어의 곱으로 정의

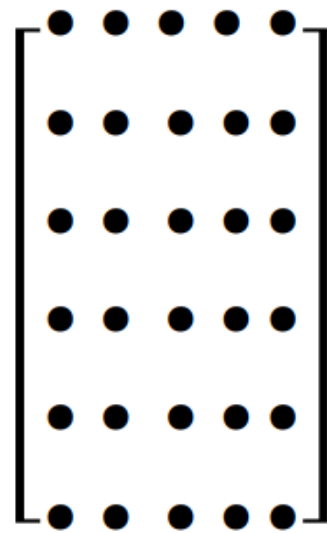


#01 Word vectors and word2vec



U

outside



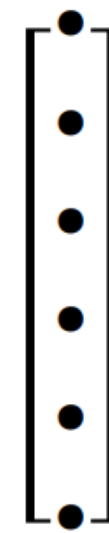
V

center



$U \cdot v_4^T$

dot product



$\text{softmax}(U \cdot v_4^T)$

probabilities

#01 단어 벡터는 행으로 표시

#02 단어 벡터는 행으로 표시 (그림에서는 6개의 단어와 5차원 벡터)

#03 U와 V_4 의 Product를 이용하여 내적벡터를 제공

#04 각 숫자에 대해 Softmax를 실행하여 단어에 대한 확률 분포를 제공

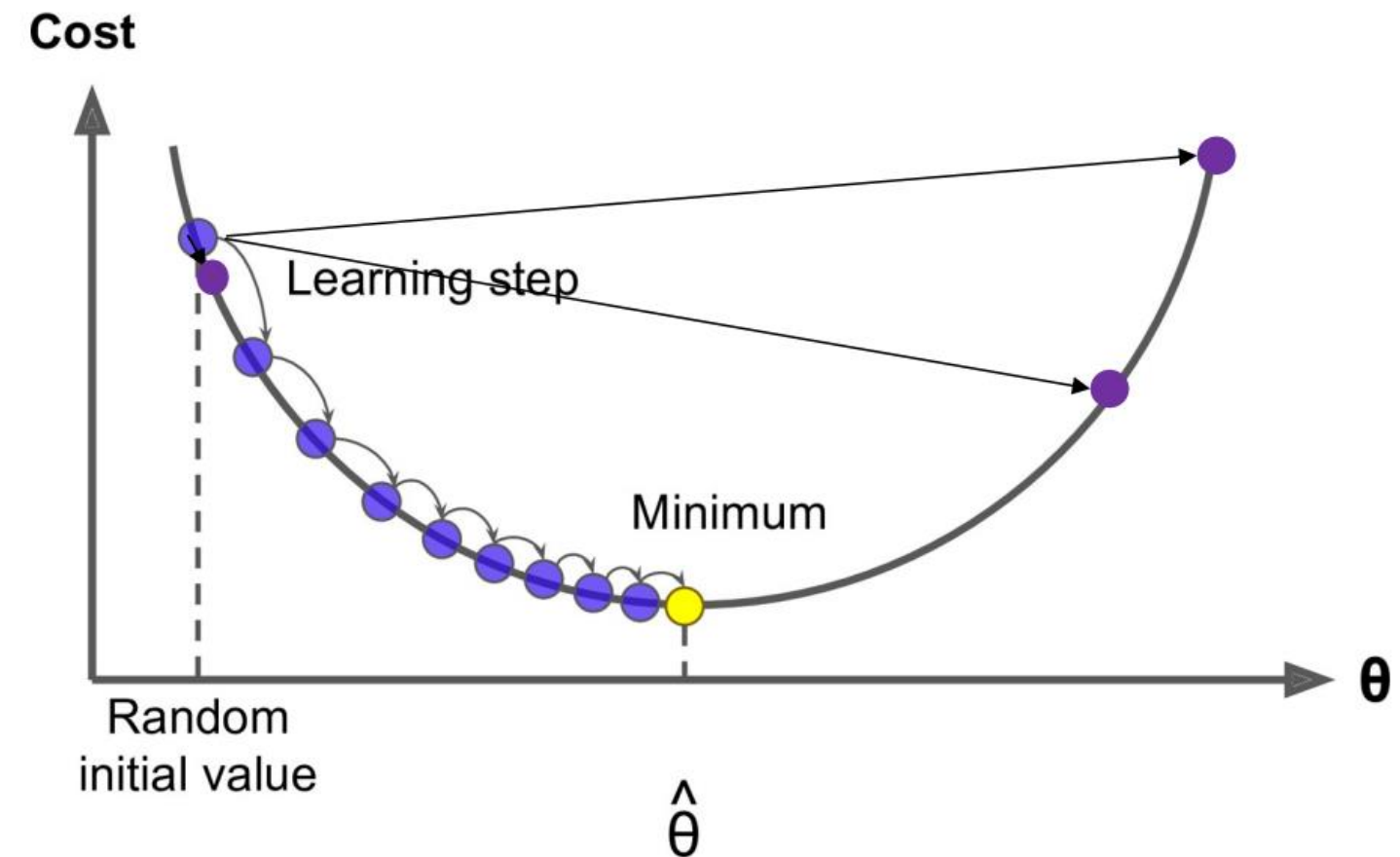
Optimization: Gradient Descent



#02 Optimization: Gradient Descent

#01 Gradient Descent

- 비용 함수를 최소화하기 위한 알고리즘
- 랜덤한 초기값 θ 에서 시작
- 단어 벡터에 대한 비용 함수의 기울기를 계산
- Gradient의 음수 방향으로 이동
- 최솟값을 찾을 때까지 반복



#02 약간만 움직여야 하기 때문에 아주 작은 α 값을 곱한다

$$\theta^{new} = \theta^{old} - \alpha \nabla_{\theta} J(\theta)$$

α = step size or learning rate

gradient

#02 Optimization: Gradient Descent

- Stochastic Gradient Descent

1. Gradient Descent Algorithm의 문제점

- 비용 함수는 모든 데이터에 대해 gradient를 계산한 후 θ 를 업데이트
- 모든 데이터를 고려하기 때문에 비교적 정확한 방향으로 이동 but 학습속도가 매우 느림

2. Stochastic Gradient Descent

- 학습 데이터 중에서 Random 하게 Sample 을 한 개씩 뽑아 gradient 를 계산 후 θ 를 업데이트
- 하나의 데이터만을 고려하기 때문에 길을 헤맬 수 있음
- Fluctuation 클 수 있지만 학습속도는 빠름

3. Mini-Batch Gradient Descent

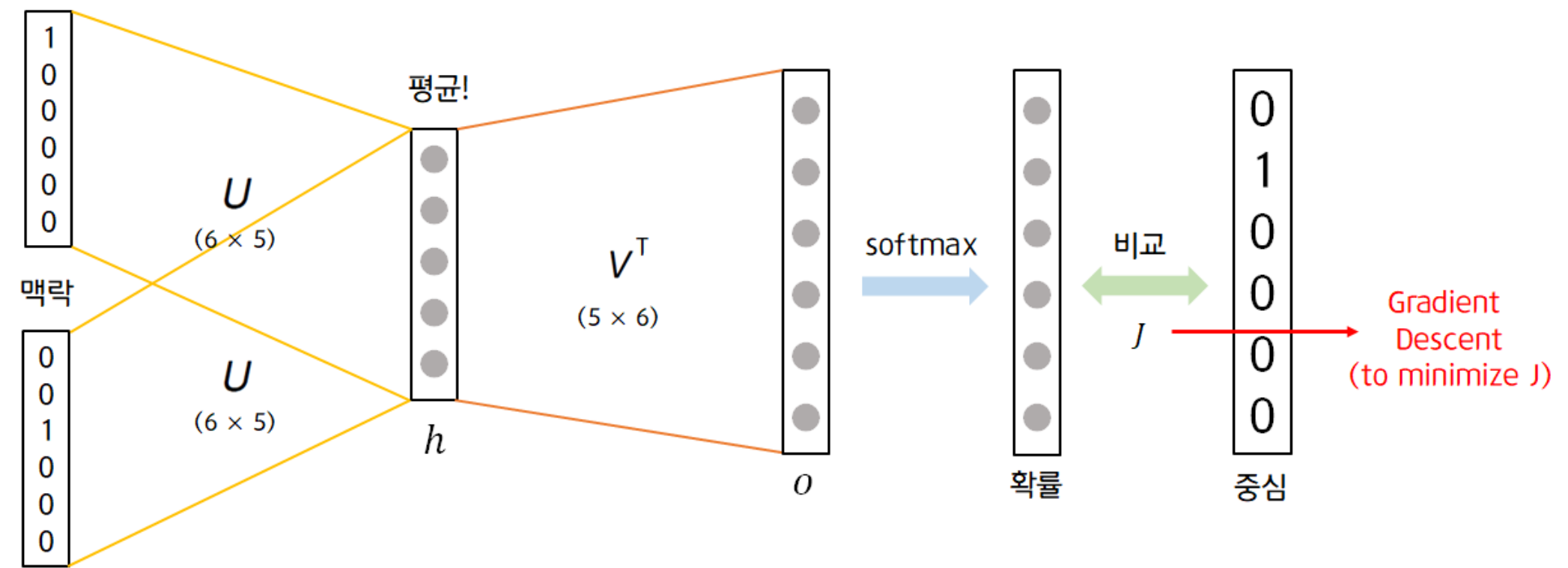
- Mini-Batch 만큼의 데이터에 대해서 gradient를 계산한 후 θ 를 업데이트 -> 현실적인 방안

#02 Word2Vec의 두가지 학습 방식

#01 Continuous Bag Of Words

- 주변 단어를 통해 중심 단어를 예측

- 맥락 벡터와 hidden layer 사이에 있는 가중치 U 와 맥락 벡터를 곱하고 hidden layer에선 이 결과들을 hidden layer의 값으로 사용

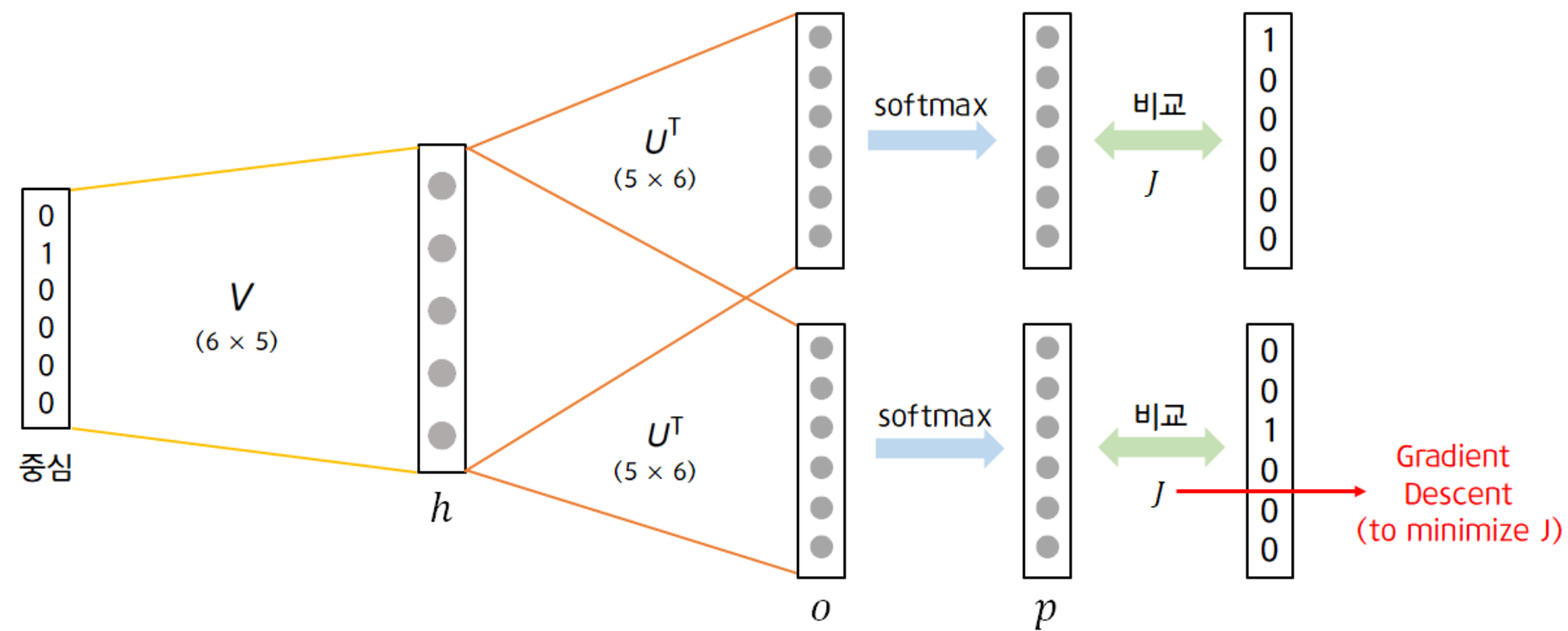


- 중심 벡터 사이의 가중치 V 와 hidden layer의 벡터를 곱하여 만든 outer layer의 벡터에 Softmax 함수를 적용하여 이 값을 확률 값으로 만든 후 중심 벡터와 비교하여 모델의 Loss를 계산

- Loss 최소화 하는 모델을 구하기 위해 Gradient Descent를 이용하여 가중치를 계속적으로 업데이트

#02 Word2Vec의 두가지 학습 방식

#02 Skip-gram

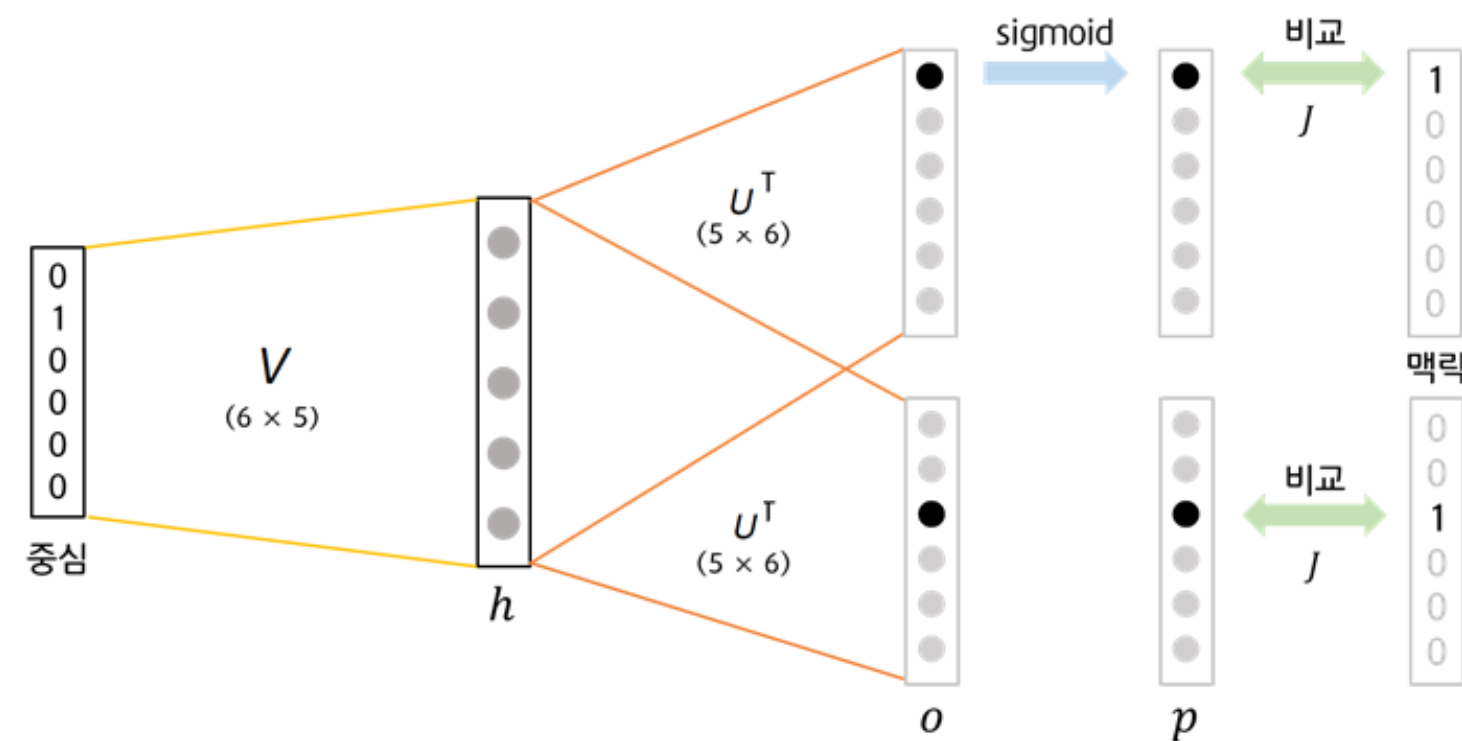


- 중심 단어에서 주변 단어를 예측

#02 Word2Vec의 두가지 학습 방식

#03 Negative Sampling with skip-gram


- sparsity로 인한 문제 해결하기 위해 등장
(0에 해당하는 위치에서 계산이 이루어지더라도 계속해서 0이기 때문에 실제로 Gradient 업데이트 되지 않지만 불필요하게 계산이 이루어짐)
- 0이 아닌 행에 대해서만 gradient 계산, sparse한 행렬을 add, subtract 함으로서 gradient 업데이트



The GloVe model of word vectors



#03 GloVe

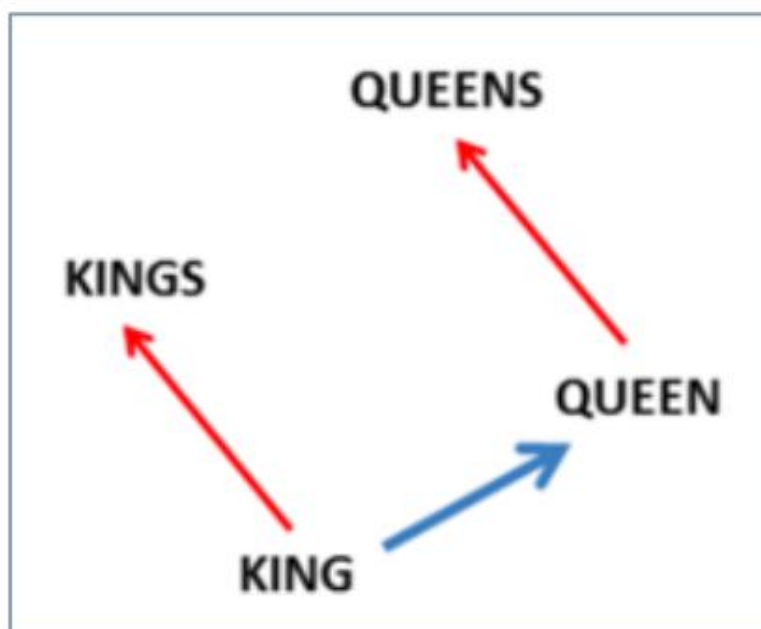


counts	i	like	enjoy	deep	learning	NLP	flying	.
i	0	2	1	0	0	0	0	0
like	2	0	0	1	0	1	0	0
enjoy	1	0	0	0	0	0	1	0
deep	0	1	0	0	1	0	0	0
learning	0	0	0	1	0	0	0	1
NLP	0	1	0	0	0	0	0	1
flying	0	0	1	0	0	0	0	1
.	0	0	0	0	1	1	1	0

Co-occurrence matrix 정보

+  GloVe

word2vec 장점



Count based 의 빠른 학습 속도와
통계 정보 사용의 효율성

+
Word2vec 과 같은 Direct prediction의 좋은 성능
단어 간의 복잡한 패턴 포착 가능

#03 GloVe

#01 Co-occurrence matrix

1) Window basend co- occurrence matrix

-word2vec과 비슷한 방식

-한 문장을 기준으로 윈도우에 각 단어가 몇 번 등장하는지를 세어 구성한다.

- I like an apple.
- I like you.

	I	like	an	apple	you	.
I	0	2	0	0	0	0
like	2	0	1	0	1	0
an	0	1	0	1	0	0
apple	0	0	1	0	0	1
you	0	1	0	0	0	1
.	0	0	0	1	1	0

#03 GloVe

#01 Co-occurrence matrix

2) Word-Document matrix

- 한 문서를 기준으로 각 단어가 몇 번 등장하는 지를 세어 구성
- 문서에 있는 많은 단어들 중 빈번하게 등장하는 특정 단어가 존재한다는 것을 전제
- 문서 간 유사도 측정

-	doc1	doc2	doc3
나	1	0	0
는	1	1	2
학교	1	1	0
에	1	1	0
가	1	1	0
ㄴ	1	0	0
다	1	0	1
영희	0	1	1
중	0	0	1

But, count-based matrix는 단어의 개수가 증가할수록
차원이 폭발적으로 증가
->SVD 또는 LSA 등을 이용하여 차원을 축소

#03 GloVe

#02 SVD (Singular Value Decomposition): 특이값 분해

- 고유값 분해와 다르게 분해할 행렬이 정방 행렬이 아니어도 분해 가능

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

$m \times n$ $m \times m$ $m \times n$ $n \times n$

정방 행렬이 아니어도 가능!
(→ 고유값 분해)

$UU^T = I$
orthogonal

Diagonal

$VV^T = I$
orthogonal

$\Sigma_k = \sqrt{\lambda_k}$
 $\lambda_k : AA^T, A^T A$ 의 kth 고유값 (eigen value)

- LSA : co-occurrence matrix를 단어 또는 문서를 기준으로 차원을 축소한 후 각 기준별로 잠재적인 의미를 분석하는 방법

#03 GloVe

#03 GloVe의 목적 함수 도출

- 임베딩된 두 단어벡터의 내적이 말뭉치 전체에서의 동시 등장 확률 로그 값이 되도록 목적함수 정의
- 아래와 같이 계산한 동시 발생 확률들을 가지고 목적 함수 도출

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

$$\frac{P_{ice,solid}}{P_{steam,solid}} = \frac{P(solid|ice)}{P(solid|steam)} = \frac{1.9 \times 10^{-4}}{2.2 \times 10^{-5}} = 8.9$$

#03 GloVe

#03 GloVe의 목적 함수 도출

- GloVe

Crucial Insight : 임베딩된 **두 단어벡터의 내적**이 말뭉치 전체에서의 **동시 등장확률 로그값**이 되도록 목적함수를 정의

F의 input F의 output

$$\underline{F(w_{ice}, w_{steam}, w_{solid})} = \frac{P_{ice,solid}}{P_{steam,solid}} = \frac{P(solid|ice)}{P(solid|steam)} = \frac{1.9 \times 10^{-4}}{2.2 \times 10^{-5}} = 8.9$$

8.9라는 결과를 만들어주는 함수 F를 찾아야 한다!

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

i번째 단어 (맥락 단어)가 있을 때 k번째 단어가 등장할 조건부 확률
j번째 단어 (맥락 단어)가 있을 때 k번째 단어가 등장할 조건부 확률

두 단어벡터의 내적이 input으로 들어가게 만들어야 한다!

#03 GloVe

#03 GloVe의 목적 함수 도출

- input은 단어 벡터 3개로 구성되어 있지만, 두 단어벡터의 내적이 input으로 들어가야 한다
- 동시 발생 확률의 비를 두 맥락 단어벡터의 차와 중심벡터의 내적 값으로 변환하여 단어 벡터 스페이스 내에 선형으로 표현 가능

- GloVe

Crucial Insight : 임베딩된 **두 단어벡터의 내적**이 말뭉치 전체에서의 **동시 등장확률 로그값**이 되도록 목적함수를 정의

F의 input F의 output

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

세 단어 벡터의 함수 → 두 단어 벡터의 함수

두 단어 벡터의 함수 → 두 단어 벡터의 내적

동시 발생 확률의 비를, 단어 벡터 스페이스 내에 선형으로 표현!

#03 GloVe

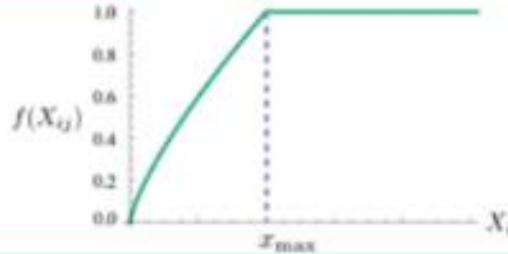
#03 GloVe의 목적 함수 도출

- GloVe 모델의 세가지 조건
 - 1) 단어 벡터간 교환 법칙 성립
 - 2) Co-occurrence Matrix 는 Symmetric 해야함
 - 3) Homomorphism 을 만족하는 지수함수 사용

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

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

Weighting Function



Evaluating Word Vectors



#04 Evaluating Word Vectors

Intrinsic(내적) vs. Extrinsic(외적) Evaluation

	Intrinsic Evaluation	Extrinsic Evaluation
설명	특정/중간 하위 작업을 통해 성능 평가	실제 작업을 통해 성능 평가
장점	빠른 성능 평가 속도 시스템 이해에 유리	효율성이 확실하다는 증거
단점	효율성이 확실하다는 증거 x	느린 성능 평가 속도 (시스템 크기 ↑) 어느 부분이 문제 요소인지 판단하기 어려움

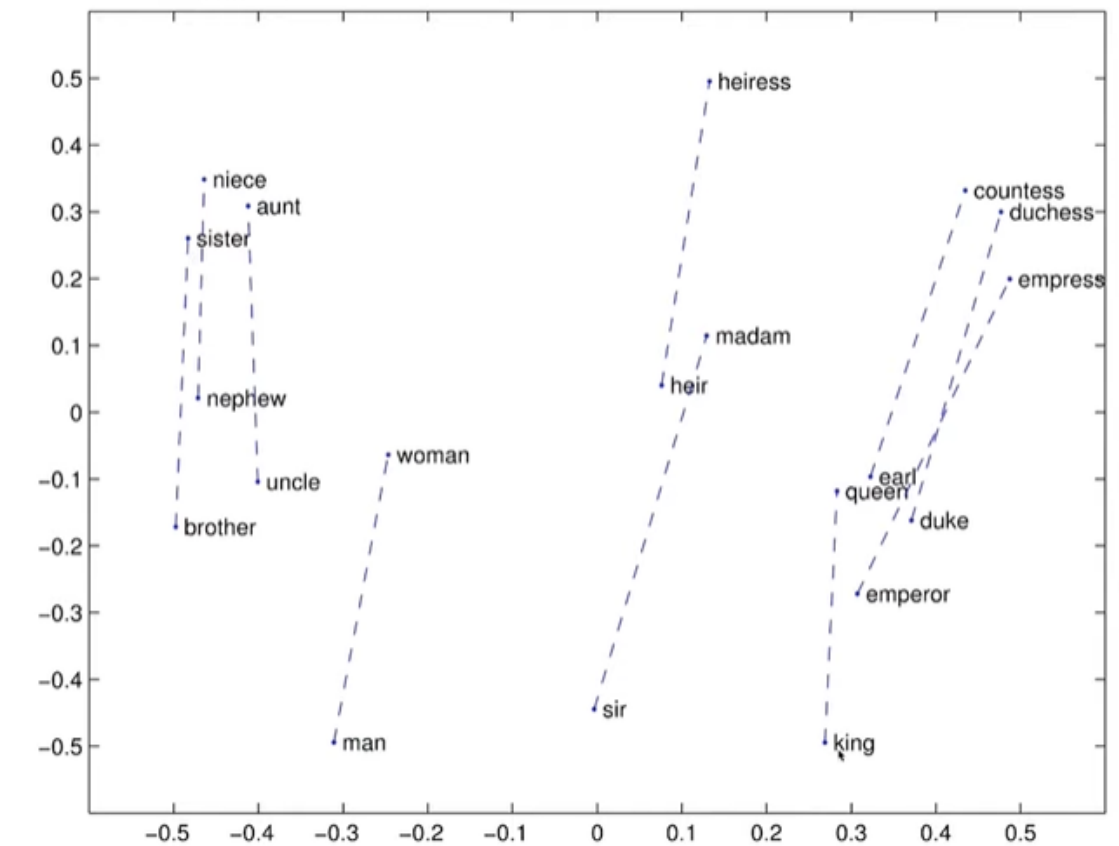
#04 Evaluating Word Vectors

Intrinsic Evaluation on GloVe Model

Word Analogies (유추)

- $a : b :: c : ?$
- Word vector 간의 코사인 거리를 비교

$$d = \arg \max_i \frac{(x_b - x_a + x_c)^T x_i}{||x_b - x_a + x_c||}$$



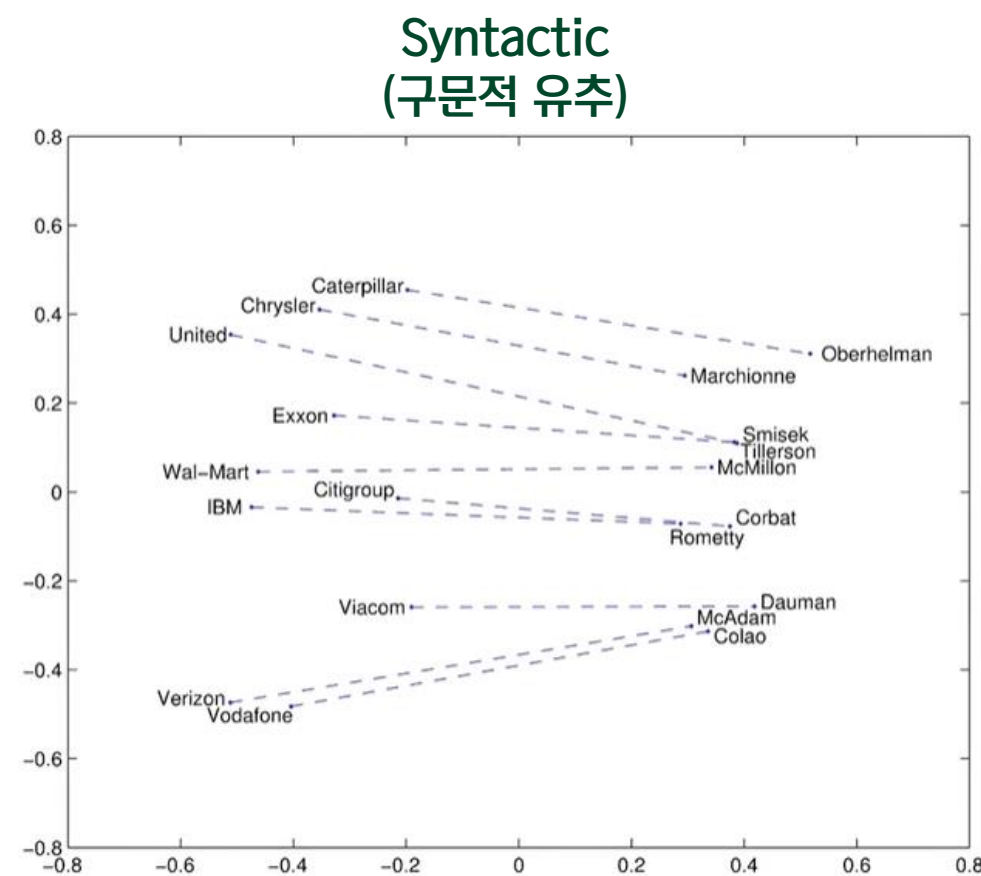
⚠ 문제 : Word vector가 존재하지만 linear한 위치에 있지 않다면...?

#04 Evaluating Word Vectors

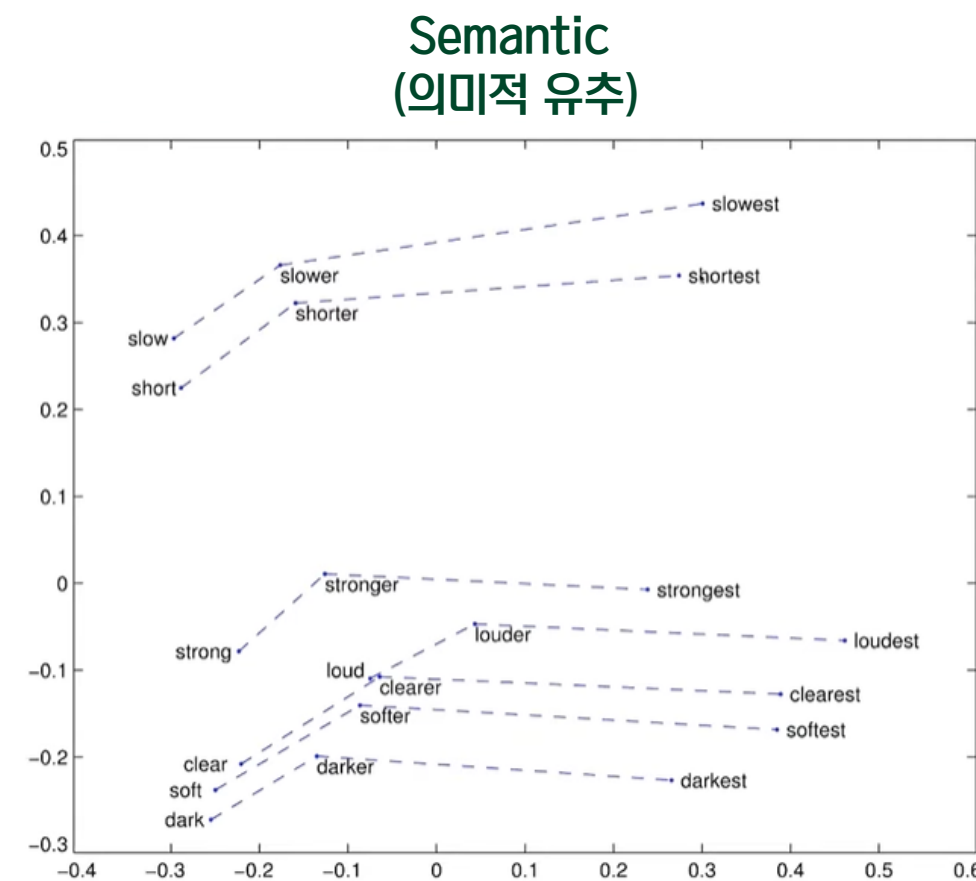
Intrinsic Evaluation on GloVe Model

Word Analogies (유추)

- $a : b :: c : ?$



Company CEO around 2010 to 2014



Positive, comparative, and superlative forms of adjectives

#04 Evaluating Word Vectors

Intrinsic Evaluation on GloVe Model

Model	Dim.	Size	Sem.	Syn.	Tot.
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	<u>67.4</u>	65.7
SG [†]	300	6B	73.0	66.0	69.1
GloVe	300	6B	<u>77.4</u>	67.0	<u>71.7</u>

=> GloVe Model의 성능 Good

#04 Evaluating Word Vectors

Intrinsic Evaluation on GloVe Model

Correlation Evaluation

- Word vector간의 거리 & 사람이 사전에 정의해둔 단어들 간의 상관관계 비교
- 사용된 데이터셋 : WordSim353

Word 1	Word 2	Human (mean)
tiger	cat	7.35
tiger	tiger	10
book	paper	7.46
computer	internet	7.58
plane	car	5.77
professor	doctor	6.62
stock	phone	1.62
stock	CD	1.31
stock	jaguar	0.92

Model	Size	WS353	MC	RG	SCWS	RW
SVD	6B	35.3	35.1	42.5	38.3	25.6
SVD-S	6B	56.5	71.5	71.0	53.6	34.7
SVD-L	6B	65.7	<u>72.7</u>	75.1	56.5	37.0
CBOW [†]	6B	57.2	65.6	68.2	57.0	32.5
SG [†]	6B	62.8	65.2	69.7	<u>58.1</u>	37.2
GloVe	6B	<u>65.8</u>	<u>72.7</u>	<u>77.8</u>	53.9	<u>38.1</u>
SVD-L	42B	74.0	76.4	74.1	58.3	39.9
GloVe	42B	75.9	83.6	82.9	59.6	47.8
CBOW*	100B	68.4	79.6	75.4	59.4	45.5

=> GloVe Model의 성능 **Good**

#04 Evaluating Word Vectors

Extrinsic Evaluation on GloVe Model

Named Entity Recognition (개체명 인식)

- 어떤 이름을 의미하는 단어를 보고 단어의 유형을 인식

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

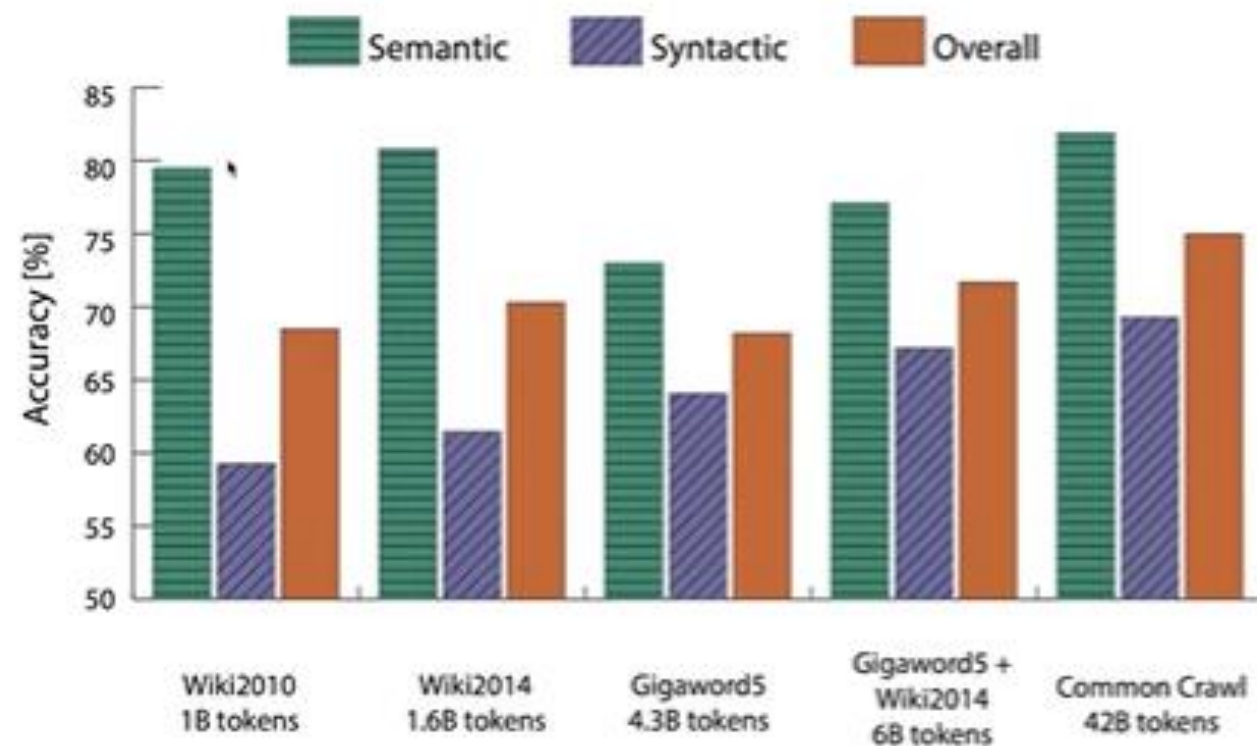
=> GloVe Model의 성능 Good

#04 Evaluating Word Vectors

+ 성능 평가에 영향을 미치는 요소들

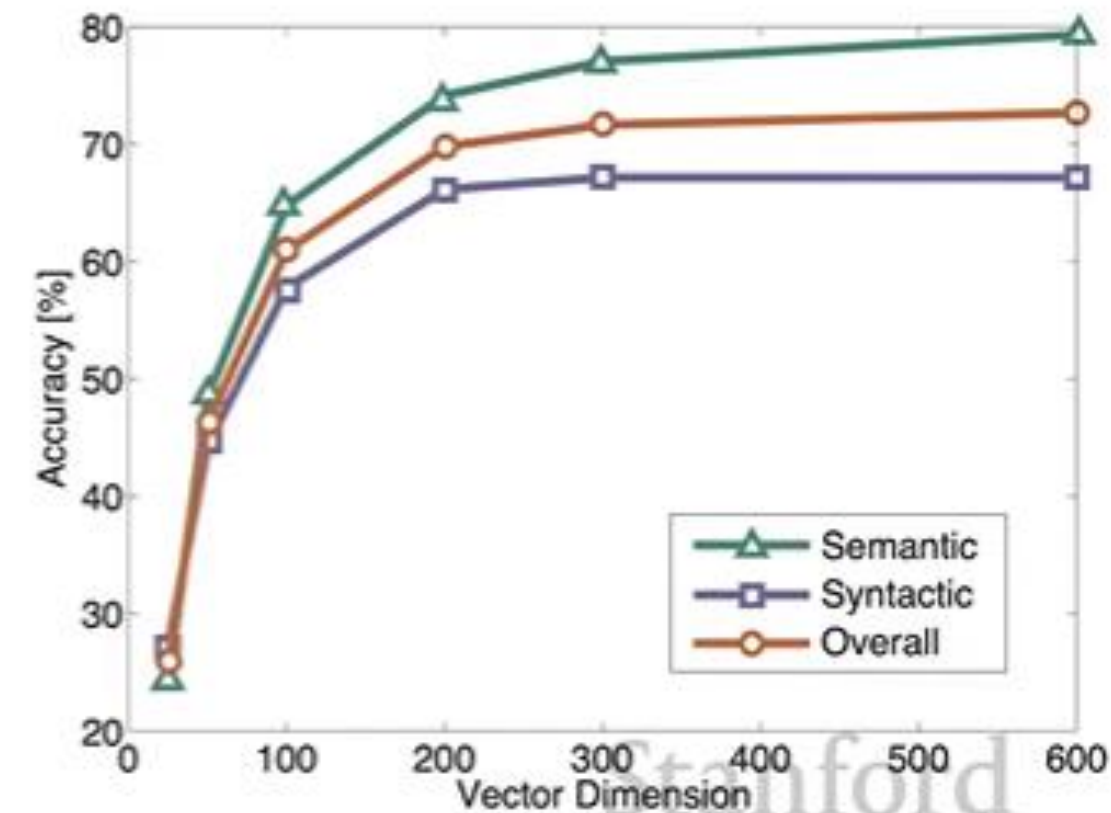
#1 Dataset

- 데이터가 많을수록 좋음
- 위키피디아의 데이터셋이 뉴스보다 좋은 결과



#2 Dimensionality

- 좋은 결과를 보이는 vector dimension은 ~300



Word Senses



#05 Word Senses

다의어와 어감 : 한 단어가 여러 뜻을 가지고 있다면 하나의 word vector로 정의할 수 있는가?

다의어 (Polysemy) : 두 가지 이상의 뜻을 가진 단어

- 자주 쓰이는 단어 중에 특히 많음
- 오래된 단어 중에 특히 많음

해결책

- 1) 다의어 한 개에 대한 하나의 vector w/ 여러 뜻 내포
- 2) 다의어 한 개에 대한 여러 개의 vector w/ vector마다 각기 다른 한 개의 뜻 내포

EWCHA
EUROPEAN

다의어 한 개에 대한 하나의 vector w/ 여러 뜻 내포

Improving Word Representations Via Global Context And Multiple Word Prototypes (Huang et al. 2012)

- Cluster word windows around words
- 군집을 기준으로 word vector 생성



단점

- 1) Word sense를 이해해야 word vector를 정의할 수 있음
- 2) 의미들 간의 차이가 불명확해서 군집으로 정리하기 어려움

#05 Word Senses

다의어 한 개에 대한 여러 개의 vector w/ vector마다 각기 다른 한 개의 뜻 내포

Linear Algebraic Structure of Word Senses, with Applications to Polysemy (Arora, ..., Ma, ..., TACL 2018)

- 의미에 따라 가중치를 부여 -> 선형결합 -> 하나의 word vector 생성
- 의문점 : 어떤 단어의 뜻도 명확히 정의되지 못하는 vector를 만들어내는 것이 아닌가?
 - 실제로 사용해봤을 때 상당히 비교 결과가 좋았음, 효율적
- Sparse coding(희소 코딩)을 적용하면 어감에 따라 나누는 것이 가능

$$\alpha_1 = \frac{f_1}{f_1+f_2+f_3}$$

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

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

THANK YOU

