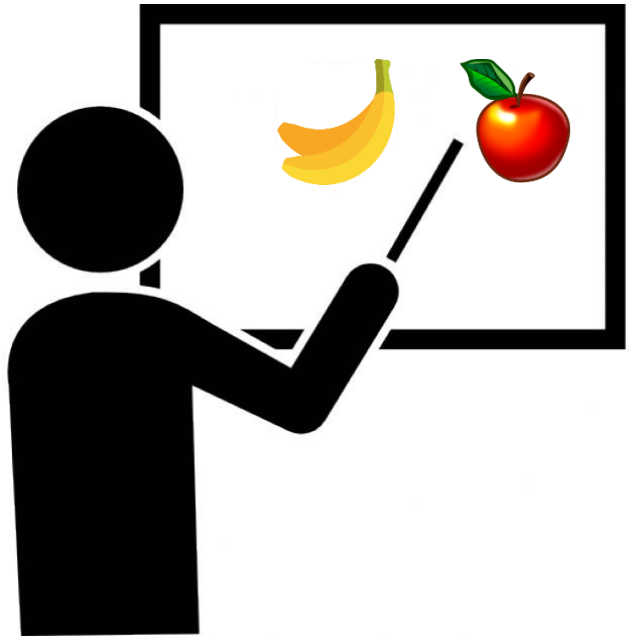
Word Vector

딥사이어인 파트3

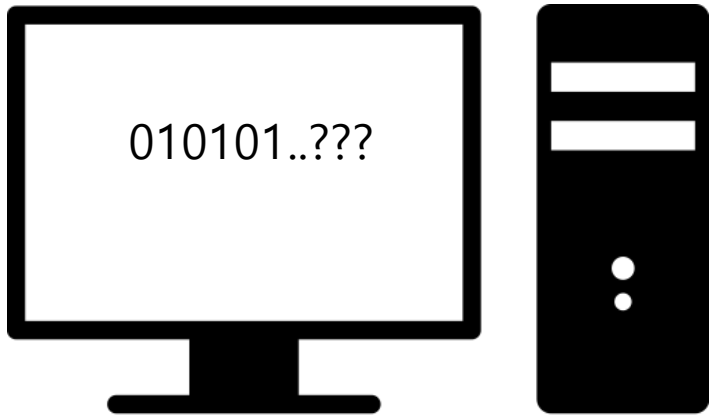
발표자 김성동

How to represent meaning in a computer?

이건 사과고 이건 바나나야
이 둘은 과일이라는 공통점이 있지만 색깔도 다르고 맛도 달라. 블라블라…….



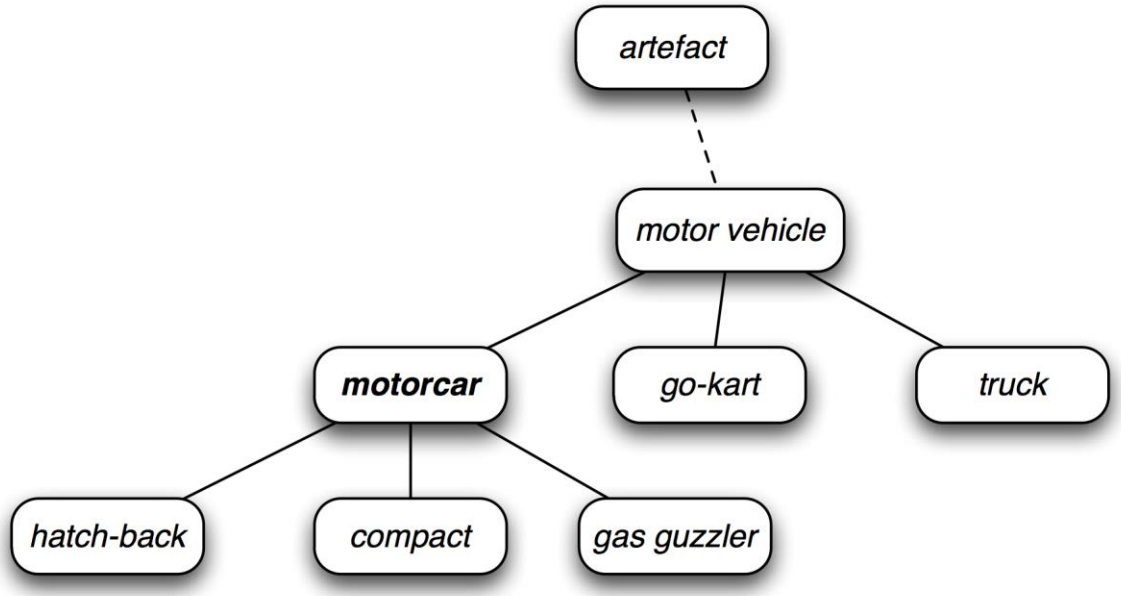
컴퓨터는 단지 유니코드의 조합으로서 문자를 인식할 뿐, 두 단어의 의미적 차이를 알지 못한다!!



How to represent meaning in a computer?

Common answer：Use a taxonomy like **Wordnet** that has hypernyms relationships and synonym sets.

Words를 tree 구조를 이용하여 관계 표현



Discrete representation에
의한 문제점들이 있고,
뉘앙스를 놓치기 쉬우며
새로운 단어들에 취약하다.
또한 주관적이며
구축하는데 엄청난 노가다를 요구한다.

How to represent meaning in a computer?

another answer : Words as atomic symbols

흔히 알고 있는 **One-hot vector 방식**

Sparse representation에 의해
단어의 수에 따라 차원수가 엄청나게 증가

또한 여전히 단어가 본질적으로
다른 단어와 어떻게 다른가를 알지 못함.

1	**0**	**0**	**0**

0	**1**	**0**	**0**

How to represent meaning in a computer?

New Idea : You shall know a word by the company it keeps

그 단어가 수반하는 Context(문맥) 이용

그렇다면 그 단어의 이웃들(neighbors)을 어떻게 파악할 것인가?

Co-occurence matrix X를 이용하자! 그 단어와 함께 등장하는 단어들을 카운트하는 매트릭스

하지만 이것 역시 Vocab size가 커질수록 저장 공간이 많이 필요하게 되고 차원수도 엄청나게 커진다. (Sparcity)

=〉 Models are less robust

Window based co-occurrence matrix

Example corpus:

1) I like deep learning.
2) I like NLP.
3) 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

Pycon India 2015

How to represent meaning in a computer?

Solution : Low dimensional vectors

차원수가 너무 커지는게 문제면 그 차원을 줄이면 되지!!

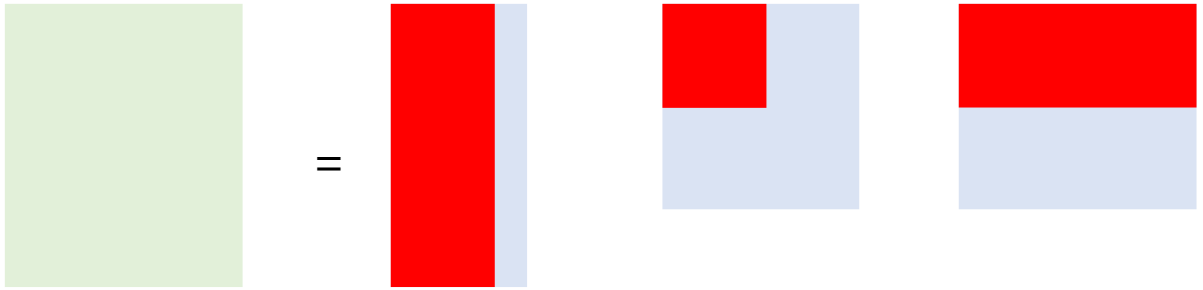
가장 핵심적인 정보들을 작고 고정된 차원의 Vector 안에 저장
–〉 A dense vector usually around 25–1000 dimensions

* How to reduce the dimensionality?

Method1 : Dimensionality Reduction on X

Ex. SVD(특이값 분해)

=〉 Semantic pattern을 발견할 수 있음.
하지만! 계산 비용이 아주 높다!



How to represent meaning in a computer?

Idea : Directly learn low-dimensional word vectors

처음부터 곧바로 낮은 차원의 word vectors를 학습하자!

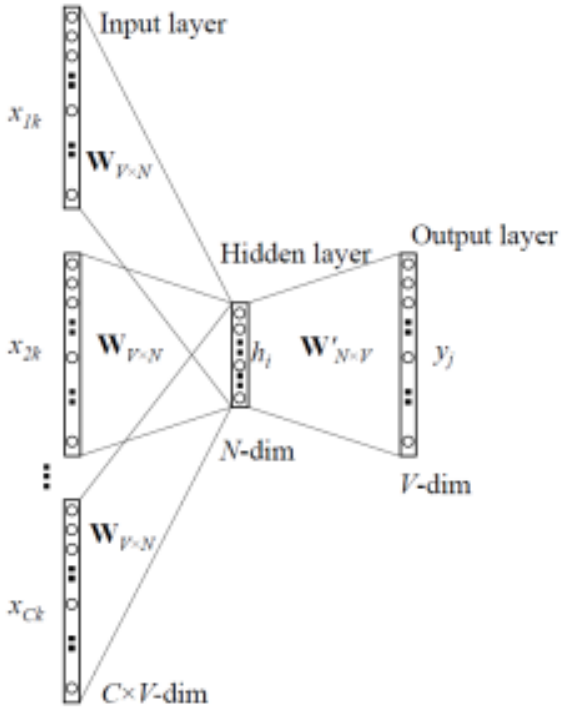
- ...
- A neural probabilistic language model (Bengio et al, 2003)
- NLP (almost) from scratch(Collobert & Weston, 2008)
- Efficient Estimation of Word Representation in Vector space(Mikolov et al, 2013)

Co-occurrence를 사용하지 말고, Predict surrounding words of every word

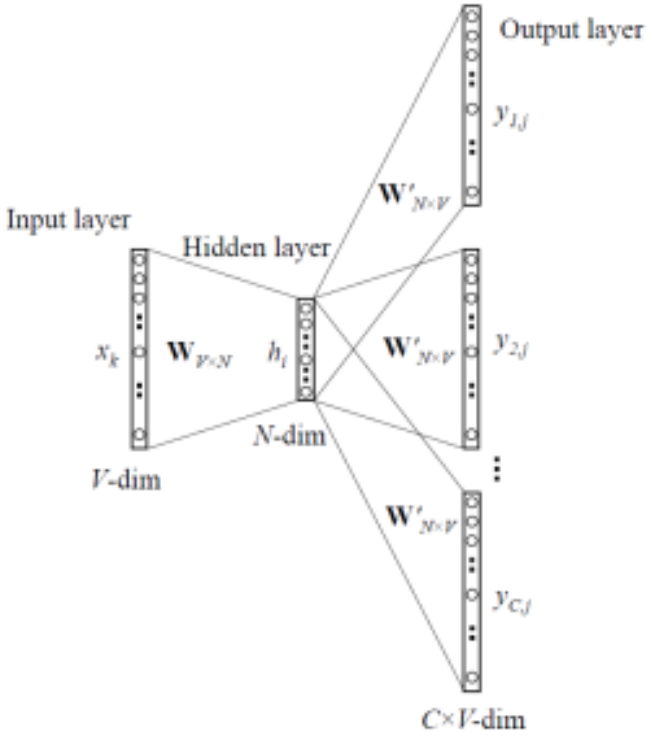
Word2Vec(Simpler and faster model)

Word2Vec

2가지 모델



CBOW
주변 단어들로부터 현재 단어 예측



Skip-gram
현재 단어로부터 주변 단어들 예측

Word2Vec

Skip-gram 모델의 Object function

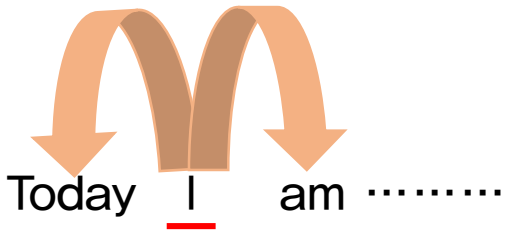
$$J(\theta) = \frac{1}{T}\sum_{t=1}^{T}\sum_{-m\leq j\leq m, j\neq 0} \log p(w_{t+j}|w_t)$$

$$p(o|c) = \frac{\exp\left(u_o^T v_c\right)}{\sum_{w=1}^{W}\exp\left(u_w^T v_c\right)}$$

u – outside vector
v – center vector

Word2Vec

1 window example



$$p(o|c) = \frac{\exp\left(u_o^T v_c\right)}{\sum_{w=1}^{W} \exp\left(u_w^T v_c\right)}$$

1st window, m=1

1) $\exp(u_{today}^T v_I) \,/\, \Sigma$
2) $\exp(u_{am}^T v_I) \,/\, \Sigma$

Log를 취해 다 더한다. (for Maximum Likelihood)

그리고 이러한 과정을 모든 윈도우(트레이닝 셋)만큼 거치고 나면 Cost Function이 된다.

$$J(\theta) = \frac{1}{T} \sum_{t=1}^{T} \sum_{-m \le j \le m, j \neq 0} \log p(w_{t+j}|w_t)$$

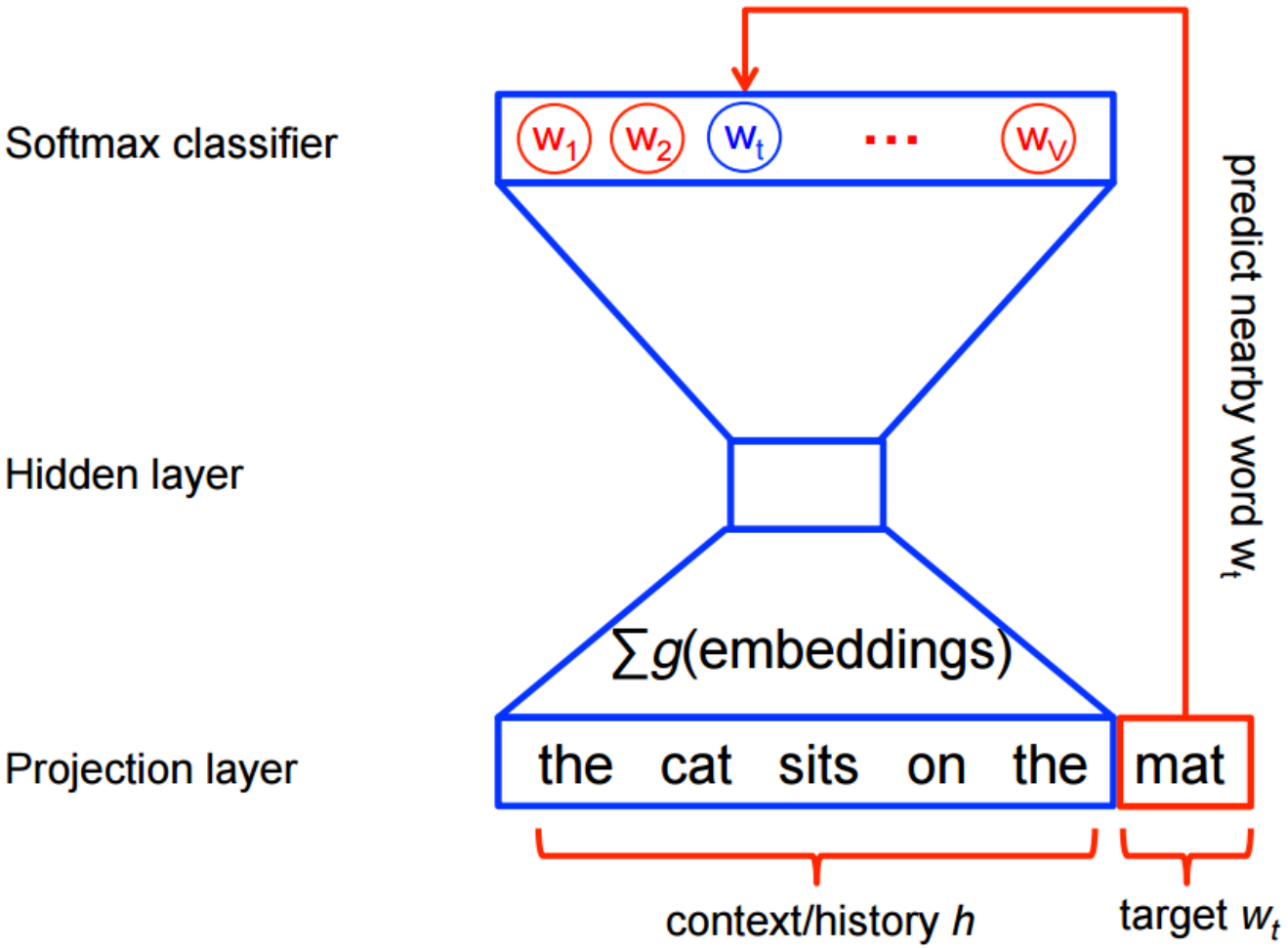
이를 Gradient Descent로 최적화한다. (수식 유도 생략.. 동영상 참고 바람)

Word2Vec

Approximation

하나의 윈도우의 Likelihood를 구할 때 마다 모든 단어들과 합쳐 Softmax 연산을 하기 때문에 계산복잡도가 매우 높다!

=〉 Approximation



Word2Vec

Negative Sampling (Noise Contrastive Estimation (NCE))

Main Idea : Train binary logistic regressions for a true pair(center word and word in its context window) and a couple of random pairs(the center word with a random word)

다른 words들과 모두 비교하는게 아니라 k개를 샘플링한 후, sigmoid를 사용함.

$$J_t(\theta) = \log \sigma\left(u_o^T v_c\right) + \sum_{i=1}^{k} \mathbb{E}_{j \sim P(w)} \left[\log \sigma\left(-u_j^T v_c\right)\right]$$

So we maximize the probability of two words co-occurring in first log!

Maximize probability that real outside word appears, minimize probability that random words appear around center word.

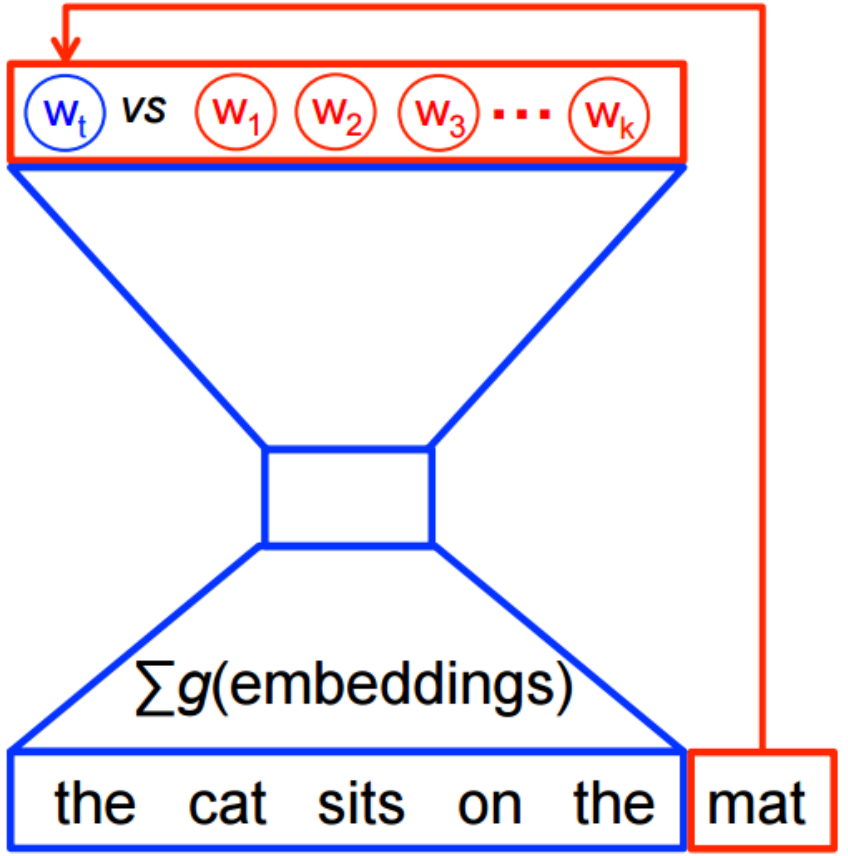
Word2Vec

Negative Sampling (Noise Contrastive Estimation (NCE))

이 때 P(w) = $\frac{U(w)^{\frac{3}{4}}}{Z}$ 에서 샘플링 하는데,
유니그램 분포에 $\frac{3}{4}$ 승을 해주게 되면
빈도가 낮은 단어들이 더 많이 샘플링 되
도록하는 효과가 있다고 한다…

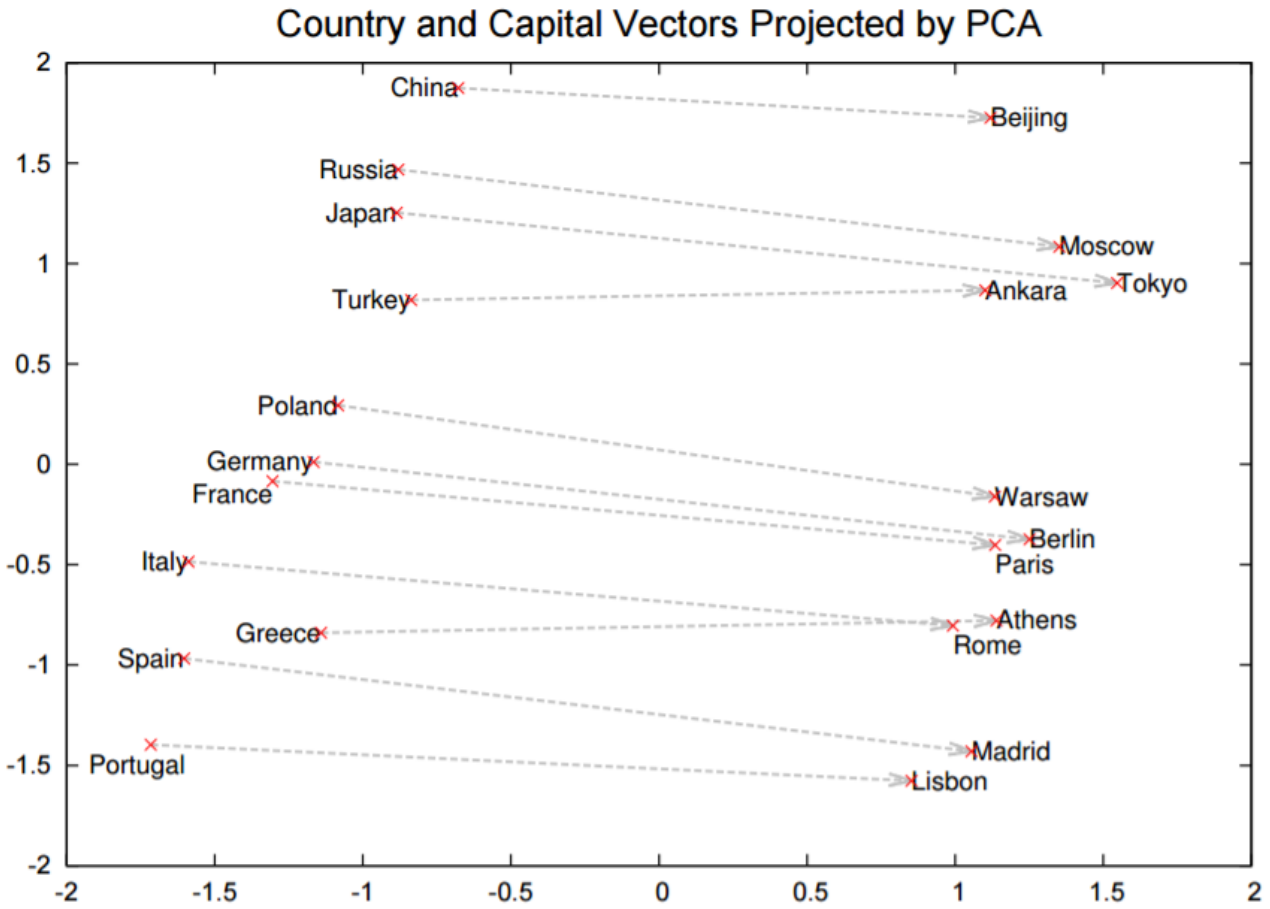
샘플링 개수는 보통 5~20개

이를 통해 계산 효율을 올릴 수 있다!



Word2Vec

Negative Sampling (Noise Contrastive Estimation (NCE))



Content based vs Direct prediction

LSA, HAL (Lund & Burgess), COALS (Rohde et al), Hellinger-PCA (Lebret & Collobert)

- Fast training
- Efficient usage of statistics
- Primarily used to capture word similarity
- Disproportionate importance given to large counts

- NNLM, HLBL, RNN, Skip-gram/CBOW, (Bengio et al; Collobert & Weston; Huang et al; Mnih & Hinton; Mikolov et al; Mnih & Kavukcuoglu)

- Scales with corpus size
- Inefficient usage of statistics
- Generate improved performance on other tasks
- Can capture complex patterns beyond word similarity

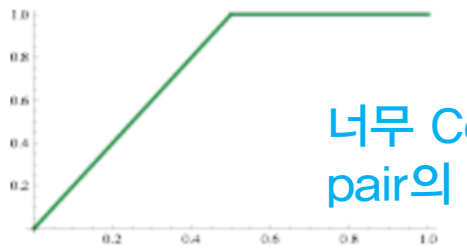
Combining the best of both worlds : GloVe

2 모델의 장점 합체!

$$J(\theta) = \frac{1}{2}\sum_{i,j=1}^{W} f(P_{ij})(u_i^T v_j - \log P_{ij})^2$$

Prediction

How often i&j co-occur

$f \sim$ 

너무 Cooccurence가 높은 pair의 영향력을 제어

- Fast training
- Scalable to huge corpora
- Good performance even with small corpus, and small vectors

How to evaluate word vectors?

Intrinsic vs Extrinsic

Intrinsic

- Evaluation on a specific/intermediate subtask
- Fast to compute
- Helps to understand that system
- Not clear if really helpful unless correlation to real task is established

Extrinsic

- Evaluation on a real task
- Can take a long time to compute accuracy
- Unclear if the subsystem is the problem or its interaction or other subsystems
- If replacing one subsystem with another improves accuracy

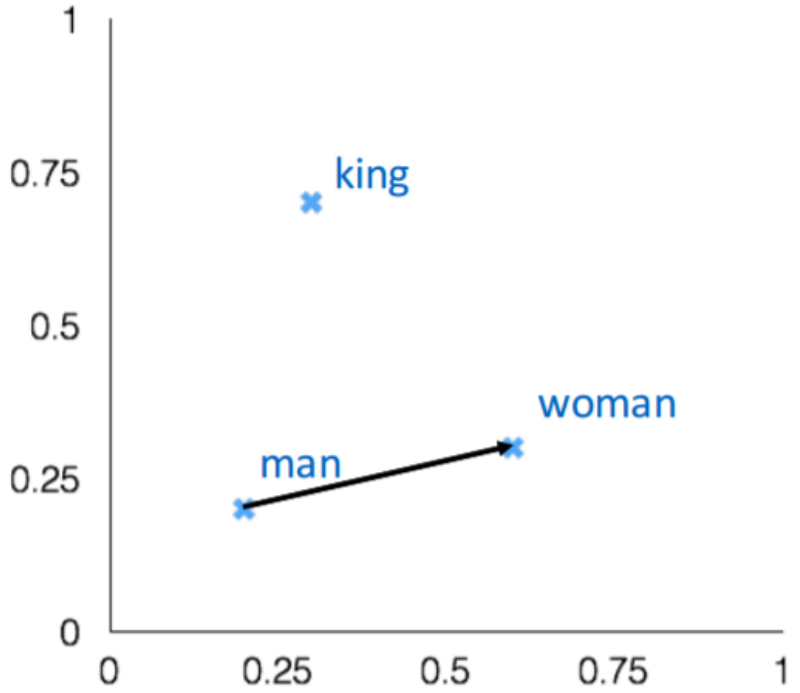
How to evaluate word vectors?

Word vector Analogies (one of Intrinsic)

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

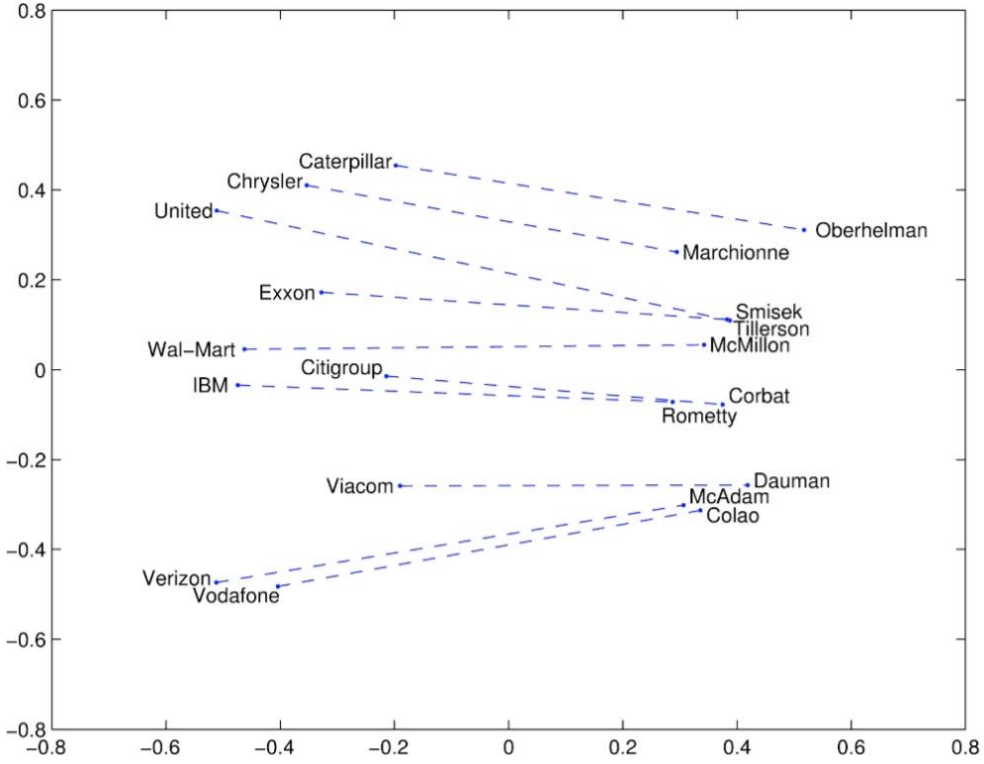
Man : woman :: king : ? (Queen)

직관적인 관계를 가진 단어 pair들을 이용하여
Cosine distance를 잘 나타내는지 확인 한다.

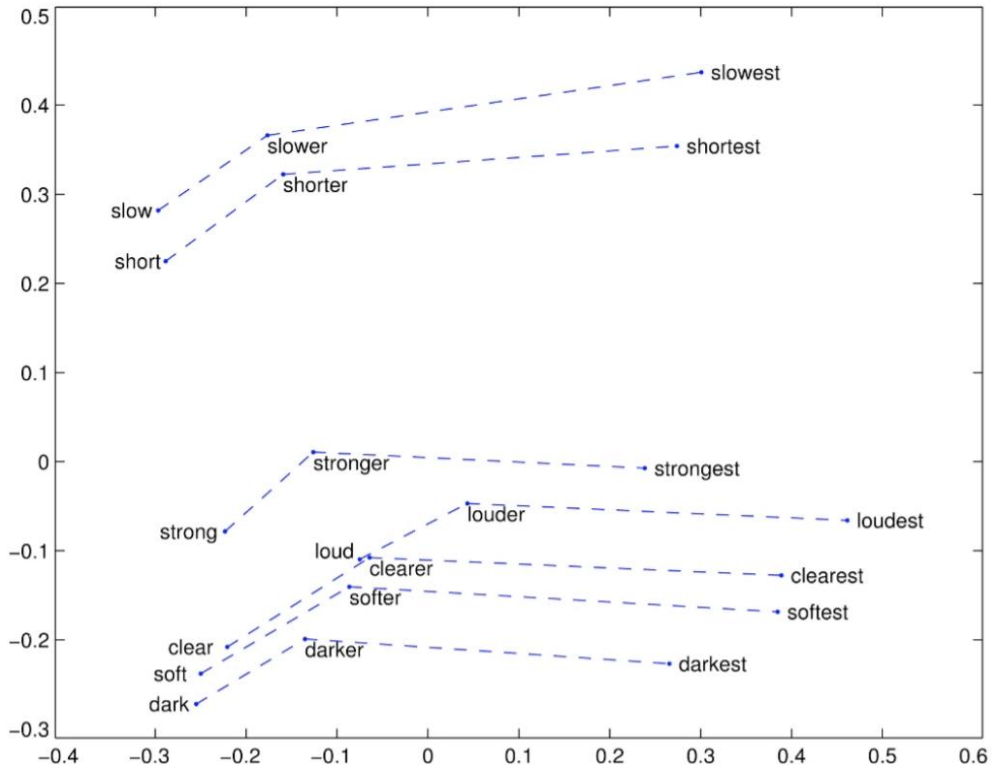


How to evaluate word vectors?

Word vector Analogies (one of Intrinsic)



Company – CEO



Superlatives

How to evaluate word vectors?

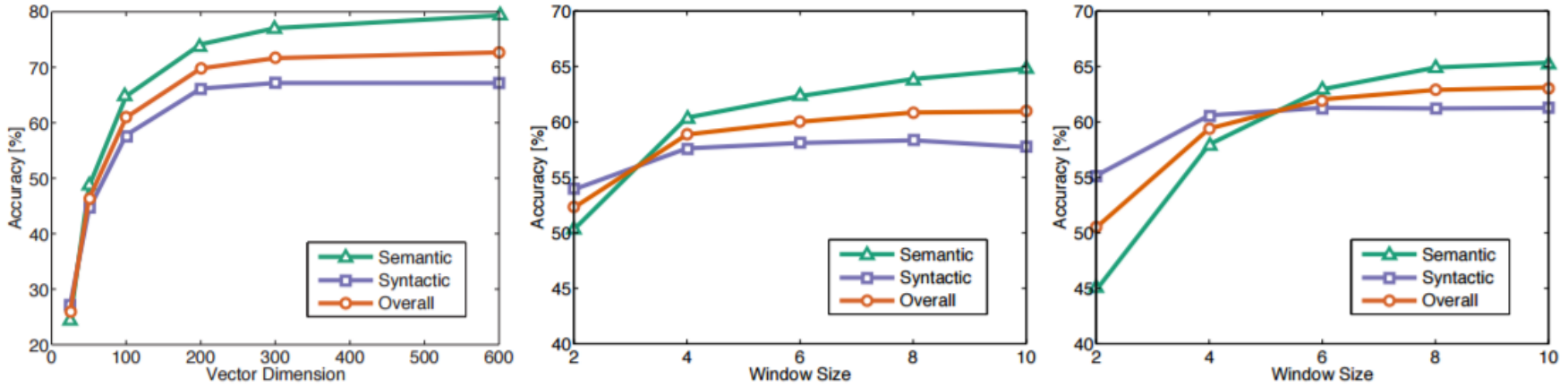
Analogy evaluation and hyperparameter

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**

How to evaluate word vectors?

Analogy evaluation and hyperparameter

Asymmetric context (only words to the left) are not as good

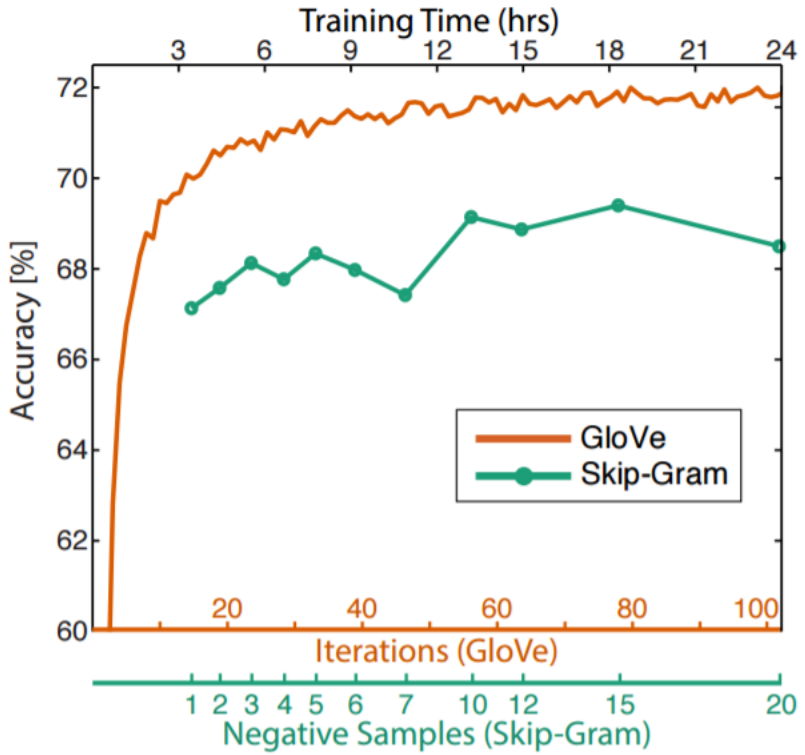


(a) Symmetric context (b) Symmetric context (c) Asymmetric context

Corpus 크기가 클 수록, Dim이 커질수록 정확도는 올라간다.
하지만 Vector dimension의 경우 300차원 넘어가면 Corpus 크기와 관계없이 정확도가 거의 개선되지 않는다.
또한 window size는 GloVe의 경우 8 정도가 적당.(4~10 사이)
비대칭(asymmetric)은 문법적으로 좋지만 의미적으로 좋지 않다.

How to evaluate word vectors?

Analogy evaluation and hyperparameter



트레이닝 많이 시킬수록 좋다

Reference

[1] http://cs224d.stanford.edu/

[2] http://solarisailab.com/archives/374

[3] https://shuuki4.wordpress.com/2016/01/27/word2vec-%EA%B4%80%EB%A0%A8-%EC%9D%B4%EB%A1%A0-%EC%A0%95%EB%A6%AC/

[4] https://arxiv.org/pdf/1301.3781v3.pdf

[5] https://arxiv.org/pdf/1310.4546.pdf