exam 개인학습

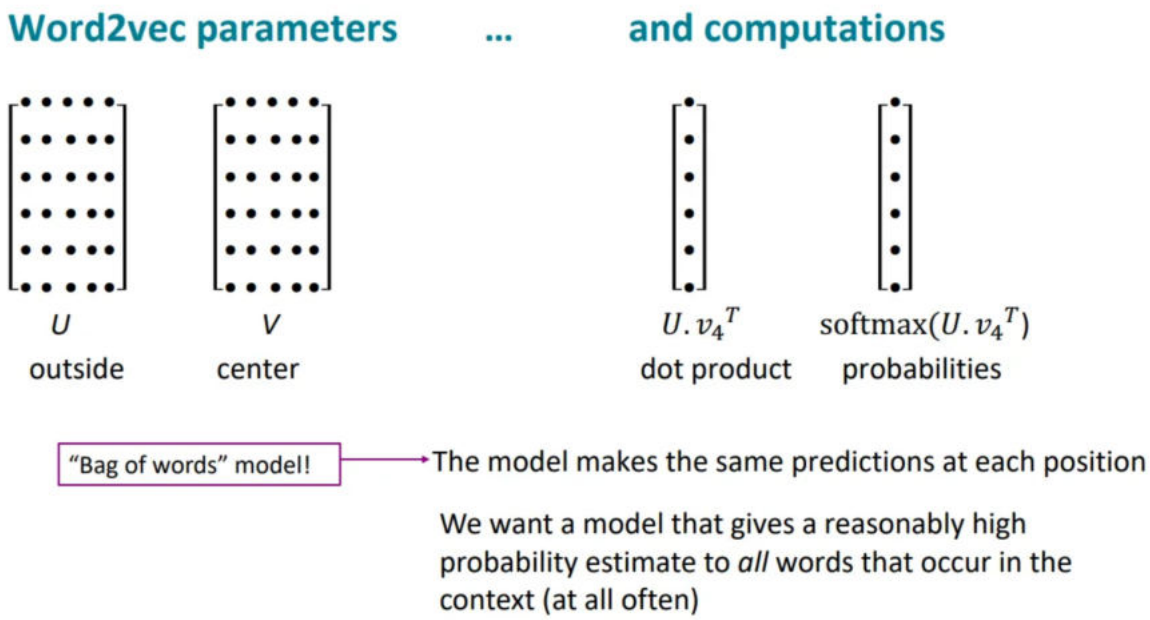
Lec 2

AI summary Try on this page

Type something...

01) Neural Classifiers

01_Review



- Outside와 center 중 input에 해당하는 vector를 뽑아서 행렬곱을 시켜주면 각각의 값은 out과 center의 내적값이다.
- 각각의 값을 softmax에 넣어서 cross-entropy loss를 구할 수 있다.

02_Negative Sampling

The skip-gram model with negative sampling (HW2)

- The normalization term is computationally expensive

- $P(o|c) = \frac{\exp(u_o^T v_c)}{\sum_{w \in V} \exp(u_w^T v_c)}$

- Hence, in standard word2vec and HW2 you implement the skip-gram model with **negative sampling**

- Main idea: train binary logistic regressions for a true pair (center word and a word in its context window) versus several noise pairs (the center word paired with a random word)

13

- 하지만 분모를 계산하는 비용이 크기 때문에 우리는 negative sampling을 이용할 것이다.
- 핵심 아이디어는 center word 근처 vs center word 근처 x 로 이진 로지스틱 회귀를 사용하는 것이다.

The skip-gram model with negative sampling [Mikolov et al. 2013]

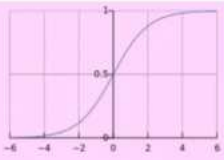
- We take *K* negative samples (using word probabilities*)
- Maximize probability of real outside word; minimize probability of random words
- Using notation consistent with this class, we minimize:

$$J_{neg-sample}(\boldsymbol{u}_o, \boldsymbol{v}_c, U) = -\log \sigma(\boldsymbol{u}_o^T \boldsymbol{v}_c) - \sum_{k \in \{K \text{ sampled indices}\}} \log \sigma(-\boldsymbol{u}_k^T \boldsymbol{v}_c)$$

sigmoid rather than softmax

- The logistic/sigmoid function:
  - (we'll become good friends soon)

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



- *Sample with P(w) = U(w)$^{3/4}$/Z, the unigram distribution U(w) raised to the 3/4 power
  - The power makes less frequent words be sampled a bit more often

- Sampling할 때 3/4승을 하여 희귀한 단어와 자주 나오는 단어의 차이를 줄여준다.
- K개의 negative sampling을 하여 각각의 loss들을 더해준다.
- negative sampling한 벡터와 center 벡터의 내적값에 -를 씌워줘서 계산한다.

03_SVD

Example: Window based co-occurrence matrix

- Window length 1 (more common: 5–10)
- Symmetric (irrelevant whether left or right context)
- Example corpus:
  - 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

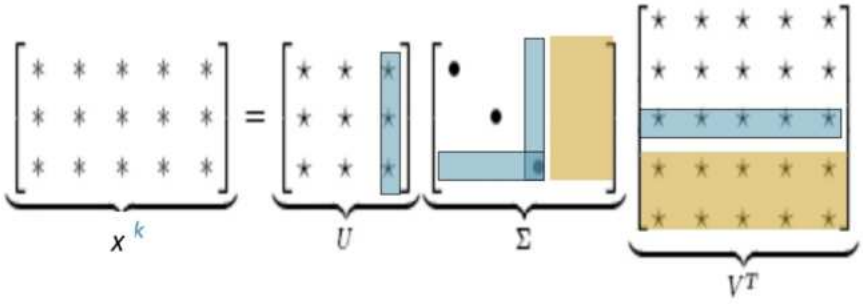
17

- 간단한 co-occurrence matrix를 살펴보자.
- 문장의 전체 정보를 잘 포착하는 행렬이다.
- 이 행렬의 row 혹은 column 벡터를 그대로 임베딩 벡터로 사용할 수 있다.
- 하지만 이 행렬은 매우 sparse하기 때문에 비효율적인 벡터이고 제대로 의미가 반영되지 않을 것이다.
- 그리고 단어의 숫자에 따라 행렬이 매우 커져 공간이 비효율적으로 사용될 것이다.
- 그렇다면 어떻게 차원을 줄일 수 있을까?

Classic Method: Dimensionality Reduction on X (HW1)

Singular Value Decomposition of co-occurrence matrix X

Factorizes X into $U\Sigma V^T$, where U and V are orthonormal



Retain only k singular values, in order to generalize.
$\hat{X}$ is the best rank k approximation to X, in terms of least squares.
Classic linear algebra result. Expensive to compute for large matrices.

19

- 고전적인 방법으로 SVD를 생각해볼 수 있다.
- 하지만 계산이 매우 비싸기 때문에 사용되지 않는다.

04_GloVe

- 우리는 지금까지 2가지 word embedding 방법을 배웠다.
- 첫 번째는 단어 개수와 행렬 분해에 의존하는 방법이다. 이는 빠르고 통계 정보를 잘 활용하지만 단어 유추에는 별로 좋지 않다.
- 두 번째는 window 창 기반 방법이다. 지역적인 문맥을 파악하고 단어 유사성을 넘어 복잡한 언어 패턴을 파악하지만 전체적인 co-occurrence 통계를 활용하지 못한다.

Encoding meaning components in vector differences
[Pennington, Socher, and Manning, EMNLP 2014]

Crucial insight: Ratios of co-occurrence probabilities can encode meaning components

	x = solid	x = gas	x = water	x = random
$P(x\|\text{ice})$	large	small	large	small
$P(x\|\text{steam})$	small	large	large	small
$\frac{P(x\|\text{ice})}{P(x\|\text{steam})}$	large	small	~1	~1

- co-occurence를 count로 사용하기 보다 확률로 사용해보자.
- 모두 관련이 있거나 모두 관련이 없는 단어는 1에 가깝고 한 단어에만 가까운경우 1에서 멀어지는 것을 확인할 수 있다.
- 우리는 단어들간의 관계를 선형적으로 보고싶다.

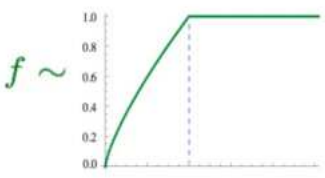
GloVe [Pennington, Socher, and Manning, EMNLP 2014]: Encoding meaning components in vector differences

Q: How can we capture ratios of co-occurrence probabilities as linear meaning components in a word vector space?

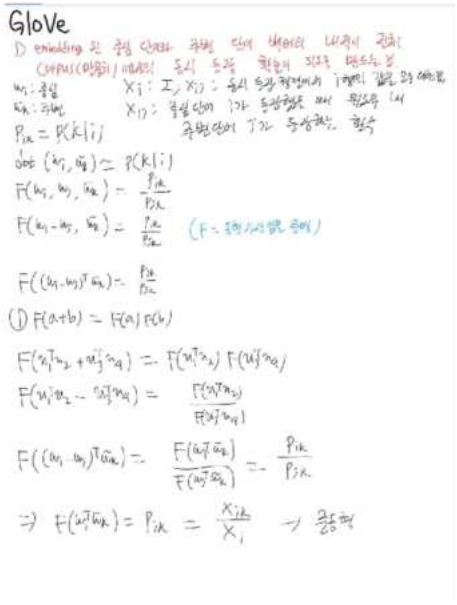
A: Log-bilinear model: $w_i \cdot w_j = \log P(i|j)$

with vector differences $w_x \cdot (w_a - w_b) = \log \frac{P(x|a)}{P(x|b)}$

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



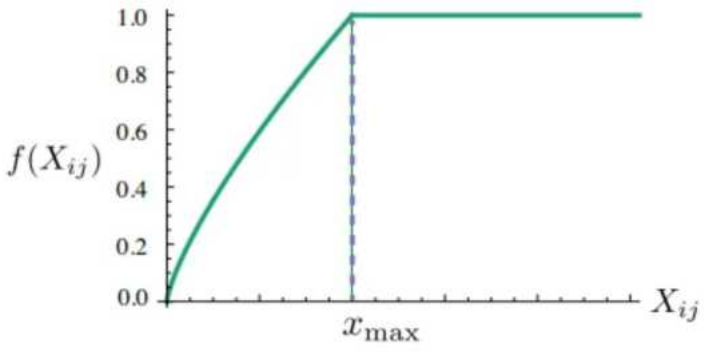
- Fast training
- Scalable to huge corpora



- 벡터의 내적값이 조건부확률과 log 관계와 비슷하게 있다는 사실을 수식을 전개하면서 알게되었다.

- 하지만 여기서 빈도수가 일정 이상일 경우 똑같이 가중치를 주도록 하여 단어 빈도에 너무 영향을 받지 않도록 했다.

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



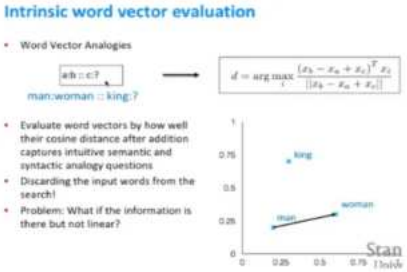
- X_ik가 작으면 loss가 작은 값을 X_ik가 크면 큰 값을 대신에 너무 커지지 않도록 조정한다.
- 따라서 loss function을 정리하면 다음과 같다.

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

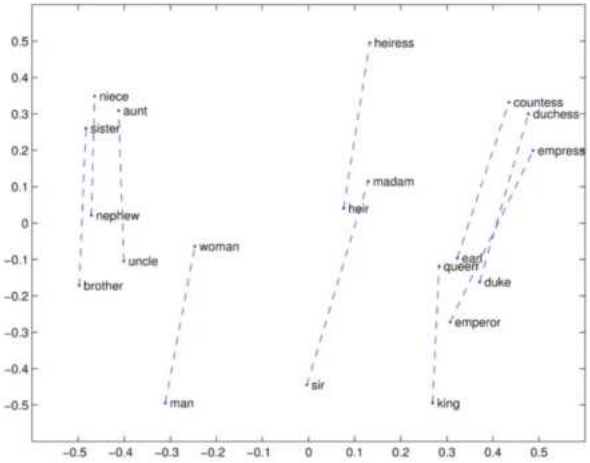
05 Evaluation

1. Intrinsic

- word embedding만을 사용하여 단어간 유사도 평가를 진행한다.



- argmax와 내적을 이용하여 d에 해당하는 문자를 찾을 수 있다.



- GloVe를 시각화하면 2차원에서 반의어가 비슷한 간격으로 공간 내에 위치하는 것을 확인할 수 있다.

2. Extrinsic

- 실제 시스템에서 사용해서 성능을 확인하는 방법
- Named Entity Recognition(NER)

06_Word senses and word sense ambiguity

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

- Different senses of a word reside in a linear superposition (weighted sum) in standard word embeddings like word2vec
- $v_{\text{pike}} = \alpha_1 v_{\text{pike}_1} + \alpha_2 v_{\text{pike}_2} + \alpha_3 v_{\text{pike}_3}$
- Where $\alpha_1 = \frac{f_1}{f_1+f_2+f_3}$, etc., for frequency f
- Surprising result:
  - Because of ideas from *sparse coding* you can actually separate out the senses (providing they are relatively common)!

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

- 한 단어가 여러 뜻을 가지기 때문에 구분하는 것은 쉬운일 이 아니다.
- 그래서 word embedding에서 처리할 때 빈도수로 가중치를 부여하여 처리할 수 있다.

- 자연어처리는 어렵다..