[Lec2] Word Vectors and Word senses

Owners	창대 오
Property	
Property 1	

Contents

▼ word2vec 리뷰

▼ 최적화 기본

▼ count-based접근법

▼ GloVe모델

▼ 단어벡터의 평가

▼ Word senses & Ambiguity

SUMMARY

💡 워드 임베딩 방법론
- 카운트 기반 VS 직접예측
- 둘을 통합한 GloVe 모델 굳
- 내부적/외부적 평가를 통해 단어 임베딩이 잘 됐는가 확인할 것
- 다의어 임베딩 주의

---

word2vec review

- 중심단어로 주변단어를(skip-gram) 혹은 주변단어로 중심단어(CBOW) 예측을 목적으로 하는
  "단어 임베딩" 모델 (하나의 은닉층을 갖는 얕은 신경망모델임)

→ input : 말뭉치, output : 각 단어들의 dense한 벡터표현 (vector representation)

→ 원핫벡터로 단어를 표현하는데 존재하는 많은 한계점을 극복하게 한다.

word2vec의 목적함수

$$J(\theta) = -\frac{1}{T}\sum_{t=1}^{T}\sum_{\substack{-m\le j\le m\\ j\neq 0}} \log P\left(w_{t+j} \mid w_t; \theta\right)$$

목적함수의 주요 최적화 대상 part는 아래와 같다.

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

<분모> : 주어진 중심단어에 대한 모든 다른 단어들의 코사인 유사도 합 (Normalizing term)
<분자> : 주어진 중심단어에 대한 출현 주변단어의 코사인 유사도
단어사전 내 모든 단어들이 한 번씩 중심단어로 설정되어 학습이 진행된다.
→ **중심단어와 그 주변단어들의 유사도는 높이면서**
그렇지 않은 단어들과의 유사도는 낮추는 방향으로 학습

이때, 최적화 대상이 되는 소프트맥스 함수에 정규화항이 포함되어있어 계산 cost가 매우크다.
(최적화를 위한 반복시마다 전체 단어들에 대한 계산을 수행하게 되므로)

이를 개선하기 위해 **<계층적 소프트맥스(hierarchical softmax)>**, **<네거티브 샘플링 (negative sampling)>**, **<subsampling frequent words>**등의 방법을 고려할 수 있다.

강의에서 다룬 네거티브 샘플링에 대해서만 간단히 살펴보자.

Negative sampling

1) 사용자가 지정한 윈도우 사이즈 내 등장하지 않는 단어(negative sample)을 5 ~ 20 개 정도 뽑음

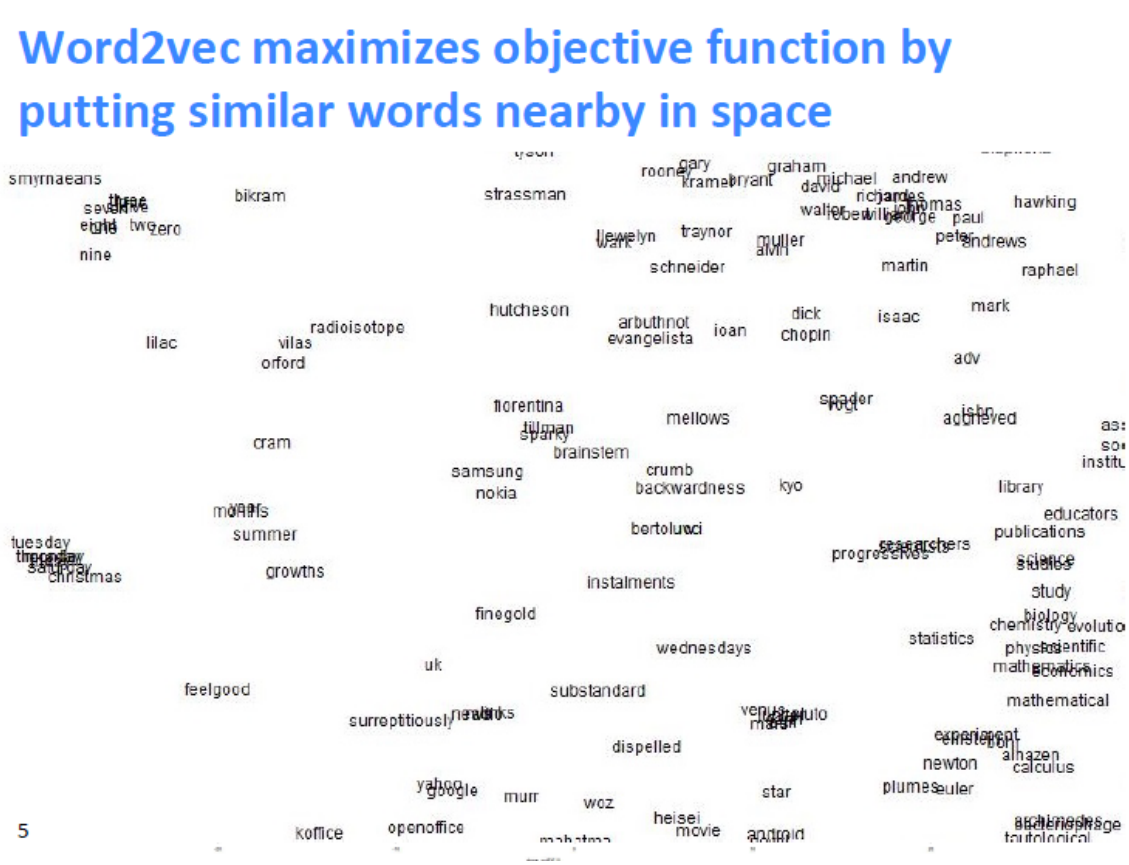
2) 실제로 윈도우 사이즈 내 등장한 단어(true context words)와 위 negative 샘플들을 합친다.

3) 합친 샘플집합을 전체 단어집합인 것으로 간주하고 소프트맥스 확률값을 계산하고 파라미터 업데이트를 진행한다.

요약

→ 소프트맥스 확률값 계산 시 전체 단어를 대상으로 구하지 않고 일부 단어들만 뽑아 계산 수행

학습결과 단어간 유사도내포는 물론 단어들 간 깊은 관계까지 내포하는 벡터공간이 생성된다.



Optimization - gradient descent

목적함수 최소화를 위해 경사하강법을 고려한다.

Gradient Descent

1) 파라미터 초기값 설정

2) 파라미터 공간상 현재 설정된 초기값에서 목적함수에 대한 gradient vector를 계산

3) gradient부호 반대 방향으로 파라미터 갱신

4) 최적점에 수렴/종료조건 만족 시까지 위의 과정 반복

→ 적절한 초기값과 학습률을 결정하는 것이 관건이다.

배치 경사하강법
* 매 반복마다 **전체 데이터포인트**에 대해 그레디언트 계산
* 계산 비용이 매우 큼
* 수렴이 안정적
* local minima(지역 최소점)에 빠질 확률이 높음

확률적 경사 하강법 (SGD)
* 매 반복마다 하나의 **샘플데이터포인트**에 대해 그레디언트가 계산됨
* 개별 반복 당 계산비용 매우 낮음
* 가중치가 자주 업데이트되어 배치 GD보다 수렴이 빠름
* shooting이 심하여 궤적이 어지러움
* local minima에 빠질 확률이 적다

미니배치 경사하강법
* 매 반복마다 사용자가 지정한 크기의 샘플데이터포인트들에 대해 그레디언트 계산
* SGD보다 less noisy
* 배치(batch)에 대해 병렬적 계산을 수행하므로(GPU활용가능) 효율적

Count-based method

셈 기반 방법 혹은 통계기반 방법

주어진 데이터의 통계 정보를 최대한으로 이용하기위한 방법

→ "직접 동시발생 빈도를 세어 단어간 유사도가 반영 될 수 있는 벡터표현을 구축하자"

- co-occurrence matrix(동시발생행렬)

: 말뭉치(corpus)의 각 문장에서 동시에 발생하는 개별 단어들을 counting

예시

I like deep learning.

I like NLP.

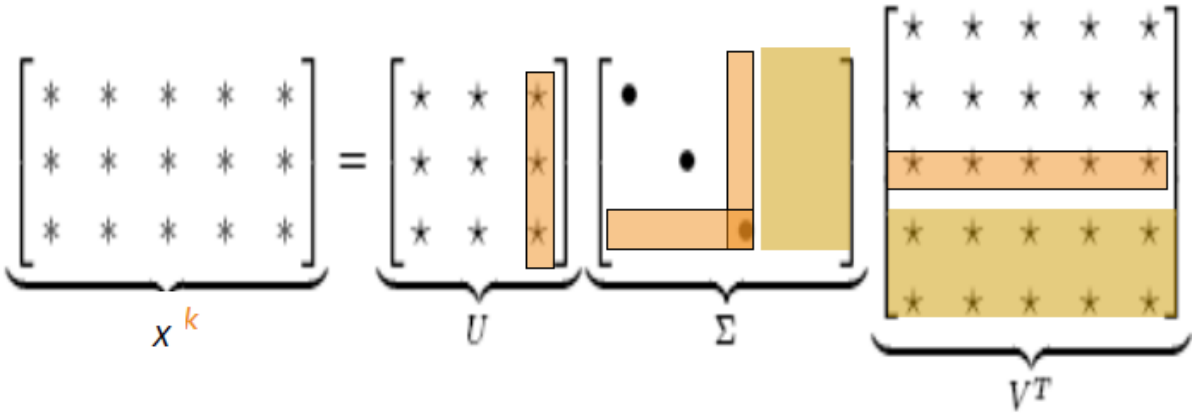
I enjoy flying.

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

이 co-occurrence matrix를 그대로 word vector representation으로써 사용하는데에 다음의 문제점이 있음

- 메모리 많이 먹음
- 매우 고차원적
- sparsity issue

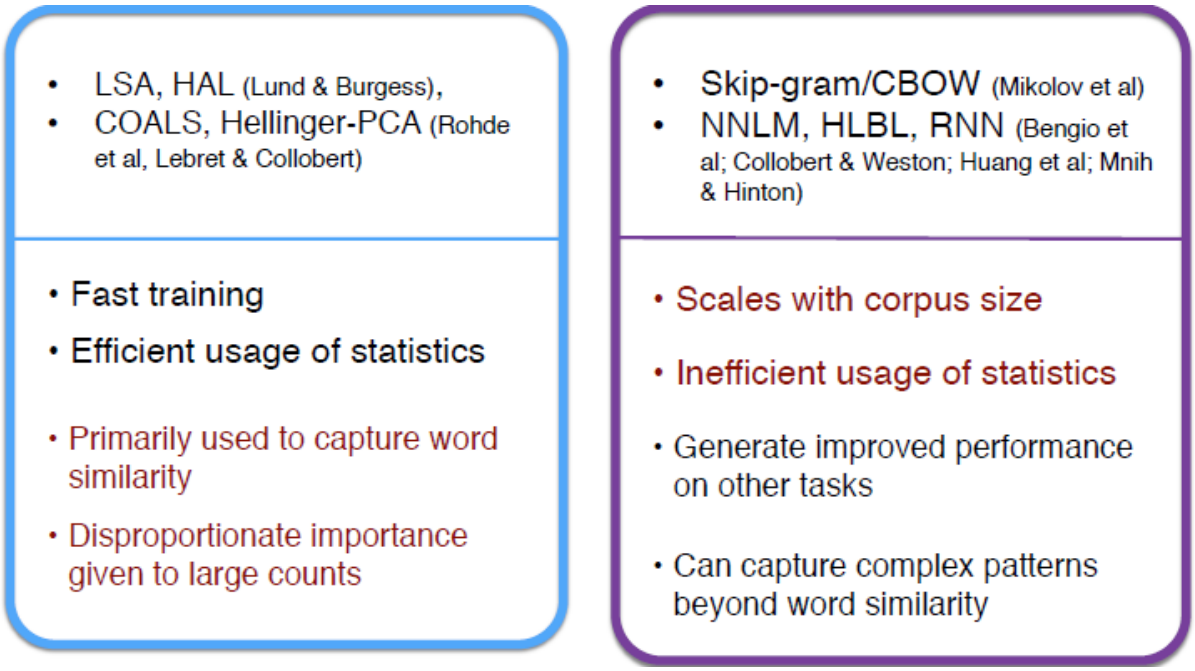
→ 차원 축소를 고려하자 (Singular Value Decomposition)



특이값분해를 통해 정보손실을 최소화 하면서 데이터셋의 차원을 축소할 수 있음

Count based vs Direct prediction

워드 임베딩에 있어 카운트기반, 직접예측 접근법 비교



카운트 기반방법

- 빠른 학습
- 통계정보 효율적 사용
- 단어간 유사성 이상의 고급 정보 포착 불가
- 큰 빈도수 단어에 과도한 중요도가 부여될 수 있음

직접예측 접근법

- 말뭉치 크기가 커야함
- 통계정보 효율적으로 사용 못함
- 대부분의 task에서 좋은 성능보임
- 단어간 유사성 이상의 복잡한 패턴 식별가능

정리하면,

> 💡 카운트기반 기법(svd - LSA)들은 통계정보를 효율적으로 활용하나 유추(analogy)성능이 딸리고,
> 직접예측(W2V)기법들은 유추성능은 좋으나 통계정보활용측면에서 비효율적

이 두 접근법의 장점만을 취합하여 절충하기 위한 모델

→ GloVe

: 유사도 측정, 의미 유추를 수월하게 하면서도 말뭉치 전체의 통계정보를 더 잘 반영하자

GloVe

<**아이디어**> : 단어들 간 동시 등장 확률의 비(ratio)를 정량화하여 단어벡터 representation에 녹여내는 것

목적함수

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

notation

Co-occurrence Matrix:

- X: word-word co-occurrence matrix
- X_{ij}: number of times word j occur in the context of word i
- $X_i = \sum_k X_{ik}$: the number of times any word k appears in the context of word i
- $P_{ij} = P(w_j|w_i) = \frac{X_{ij}}{X_i}$: the probability of j appearing in the context of word i

, f는 가중함수 - 빈도수에 대해 적절한 weighting을 수행하는 기능

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

학습순서

1. 전체 training data로부터 co-occurrence matrix를 만든다.
2. 위와같은 손실함수를 정의한다.
3. forward, backword 연산 후 매개변수 업데이트

특징

- global한 통계정보(말뭉치 전체 대상)의 효율적인 사용
- 학습이 빠르다
- 작은 말뭉치에서 큰 말뭉치까지 모두 좋은 성능을 보인다,

Evaluation - 단어벡터 평가

intrinsic eval (내부평가)

* 특정한 task 및 하위작업등에 대한 평가
* 계산이 빠름
* 시스템을 이해하는데 도움이 됨
* 실제 task에 유용한지는 unclear함

extrinsic eval (외부평가)

* 실제 현실문제에 대한 word vector의 활용 평가
* 측정하는데 오래걸림
* 문제점 발생시 system 문제인지 model 문제인지 규명하기 어려움

외부평가(ex - named entity recognition등 직접 nlp시스템에 단어임베딩을 사용해보고 평가)

내부평가(ex - word analogies(syntactic, semantic)평가, human judgement평가(미리 수작업한 단어간 correlation))

- 거의 모든 경우 GloVe의 성능이 다른 모델들(SG, CBOW, SVD 등)보다 우수했음
- 훈련을 오래 시킬수록 성능 좋아짐
- 사용된 말뭉치의 퀄리티가 좋을수록(뉴스기사보다는 위키백과) 성능 좋아짐

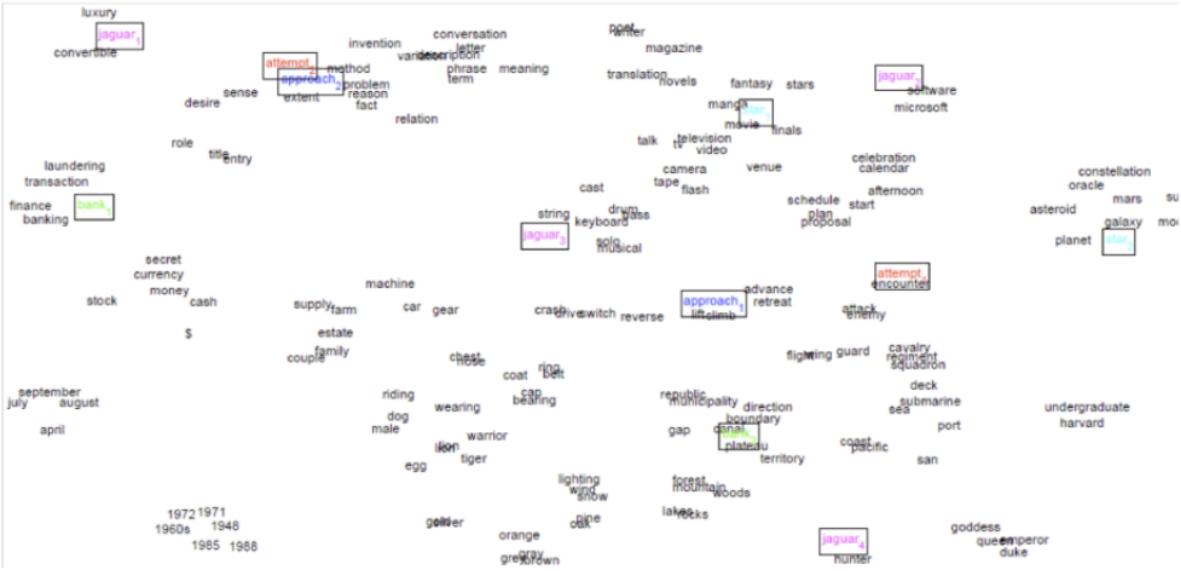
Word senses & ambiguity

단어는 문맥에 따라 여러가지 다른 뜻으로 해석될수도 있음

특히 오래된 단어, 빈번하게 사용되는 단어일수록. (ex : pike, tie)

→ 클러스터링을 통한 부분집합 라벨링을 통해 각기다른 문맥에서 representation을 학습 이후 아래와 같은 선형결합(가중평균)을 통해 최종 단어임베딩에 반영

- Idea: Cluster word windows around words, retrain with each word assigned to multiple different clusters bank_1, bank_2, etc



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

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

→ **sparse coding** 아이디어에로부터 위와같은 선형결합에 의해 각 분리된 의미 표현이 모두 잘 내제될 수 있음