

Word Embedding :

GloVe, Swivel, 단어 유사도 평가, 가중임베딩

4.5절 GloVe

- 1) LSA, Word2Vec 방법의 한계
- 2) 윈도우 기반 동시등장행렬
- 3) 동시등장확률
- 4) 손실함수
- 5) 모델 구조

4.6절 Swivel

- 1) PMI 행렬
- 2) 손실함수

4.7절 단어 임베딩 평가

- 1) 단어 유사도 평가(word similarity test)
- 2) 단어 유추 평가(word analogy test)
- 3) 단어 임베딩 시각화

4.8절 가중 임베딩

- 1) 주제벡터



4.5절

GloVe

(Global Vectors for Word Representation, 스탠포드, 2014)

GloVe

- 행렬분해 기반 단어 임베딩
- 어떤 행렬을 분해하나?) 동시등장행렬

1) LSA(Latent Semantic Analysis, 잠재의미분석) & Word2Vec 기법의 단점 극복

LSA	:	카운트 기반 (더 자세히) 각 단어의 빈도수를 카운트한 행렬이라는 전체적인 통계 정보를 입력 받아 차원축소하여 잠재된 의미를 끌어내는 방법론
	:	말뭉치 전체 통계정보를 모두 활용할 수 있음
	:	단어 간 유사도 = 단어 의미의 유추작업 측정에는 어려움
word2vec	:	예측 기반 (더 자세히) 실제값과 예측값에 대한 오차를 손실함수를 통해 줄여나가며 학습하는 방법론
	:	단어 간 유사도 측정에는 성능이 좋음
	:	임베딩 벡터가 윈도우 내에서만 로컬문맥을 학습하기 때문에 말뭉치 전체 통계정보를 반영하지 못함

목표

임베딩 된 단어 벡터 간 1) 말뭉치 전체의 통계 정보를 반영 2) 유사도 측정 수월하게

LSA의 메커니즘인 카운트 기반 방법과 word2vec의 메커니즘인 예측 기반 방법론 모두 사용

LSA의 메커니즘
카운트 기반 방법

For

말뭉치 전체 통계정보 반영

word2vec의 메커니즘
예측 기반 방법론

For

단어 간 유사도 측정 수월

윈도우기반
동시/등장행렬

word2vec의 메커니즘
예측 기반 방법론
For
단어 간 유사도 측정 수월

GloVe

2) 윈도우 기반 동시등장행렬(window based Co-occurrence Matrix)

- : 행과 열을 전체 단어 집합의 단어들로 구성하고, i단어의 윈도우 크기 내에서 k단어가 등장한 횟수를 i행 k열에 기재한 행렬
- : 윈도우 크기가 N일 때, 양쪽 좌 / 우에 존재하는 N개의 단어만 참고한다.

예시)

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

<표 1. 윈도우 크기=1 일 때, 동시등장행렬 >

	I	Like	Enjoy	Deep	Learning	NLP	flying
I	0	2	1	0	0	0	0
like	2	0	0	1	0	1	0
Enjoy	1	0	0	0	0	0	1
Deep	0	1	0	0	1	0	0
Learning	0	0	0	1	0	0	0
NLP	0	1	0	0	0	0	0
flying	0	0	1	0	0	0	0

i 단어 'I'일 때)

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

i 단어 'like'일 때)

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

GloVe

3) 동시등장확률(Co-occurrence Probability)

: 동시등장확률 $P(k | i)$ 는 특정 단어 i 의 전체 등장 횟수를 카운트하고, 특정단어 i 가 등장했을 때 어떤 단어 k 가 등장한 횟수를 카운트하여 계산한 조건부 확률

<표 1. 윈도우 크기=1 일 때, 동시등장행렬 >

	I	Like	Enjoy	Deep	Learning	NLP	flying
I	0	2	1	0	0	0	0

- X_{ij} : 중심 단어 i 가 등장했을 때 윈도우 크기 내 주변 단어 j 가 등장한 횟수 $\mathcal{X}_{I \text{ like}} = 2$
- $X_i = \sum_k X_{ik}$: 동시등장행렬에서 i 행의 값을 모두 더한 값 $\mathcal{X}_{I \text{ like}} + \mathcal{X}_{I \text{ enjoy}} + \mathcal{X}_{I \text{ deep}} + \mathcal{X}_{I \text{ learning}} + \mathcal{X}_{I \text{ NLP}} + \mathcal{X}_{I \text{ flying}} = 3$
- $P_{ik} = P(k|i) = X_{ik} / X_i$: 중심단어 i 가 등장했을 때 윈도우 크기 내에 주변단어 k 가 등장할 확률 $= 2/3$

3) 동시등장확률(Co-occurrence Probability)

<표 2. 동시등장확률과 크기관계 비 - 1보다 훨씬 크거나 훨씬 작은 경우>

	k= solid	k=gas	k=water	k=fashion
$P(k \mid \text{ice})$	0.00019	0.000066	0.003	0.000017
$P(k \mid \text{steam})$	0.000022	0.00078	0.0022	0.000018
$P(k \mid \text{ice}) / P(k \mid \text{steam})$	8.9	0.085	1.36	0.96

k = solid일 때, ice가 등장할 확률은 steam이 등장할 확률보다 약 8.9배가 크다.

k = gas일 때, gas는 ice보다 steam과 더 자주 등장하므로 1보다 훨씬 작은 값인 0.085가 나온다.

중심단어 i가 주변단어 중 특정단어 k와 함께 자주 등장하면, 확률값이 1보다 훨씬 커진다.

<표 2. 동시등장확률과 크기관계 비 - 1과 가까운 경우>

	k= solid	k=gas	k=water	k=fashion
$P(k \mid \text{ice})$	0.00019	0.000066	0.003	0.000017
$P(k \mid \text{steam})$	0.000022	0.00078	0.0022	0.000018
$P(k \mid \text{ice}) / P(k \mid \text{steam})$	8.9	0.085	1.36	0.96

k = water일 때, ice와 steam 두 단어와 동시등장하는 경우가 많으므로 1에 가까운 값인 1.36이 나온다.

k = fashion일 때, ice와 steam 두 단어와 동시등장하는 경우가 적으므로 1에 가까운 값인 0.96이 나온다.

중심단어 i와 특정단어 k가 서로 (관련이 있어) 자주 등장하거나 (관련이 없어) 거의 등장하지 않으면, 확률값이 1과 가까운 수가 나온다.

LSA의 메커니즘
카운트 기반 방법

For

말뭉치 전체 통계정보 반영

word2vec의 메커니즘
예측 기반 방법론

For

단어 간 유사도 측정 수월

LSA의 메커니즘
카운트 기반 방법

For

말뭉치 전체 통계정보 반영

손실함수가 최소값이 되는
두 단어의 벡터값을 찾음

GloVe

4) 손실함수(Loss function)

: **설립가설**) 문맥 내에서 두 단어의 동시등장비율은 두 단어의 의미와 관련이 깊다.

: **목적함수**) 임베딩 된 중심단어와 주변단어 벡터의 내적이 전체 말뭉치에서의 동시 등장 확률이 되도록 만드는 것

$$U_i * V_j = X_{ij}$$

: **손실함수**) 중심단어 벡터 u_i , 주변단어 벡터 v_j 의 내적과 두 단어 i, j 가 동시에 등장할 빈도인 A_{ij} 의 차이가 최소가 되도록 하는 u_i, v_j 을 찾을 것.

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

(V = 단어 집합의 크기)

두 단어 임베딩 벡터의 내적

두 단어 동시등장 빈도 로그값

GloVe

4) 손실함수(Loss function)

: **설립가설**) 문맥 내에서 두 단어의 동시등장비율은 두 단어의 의미와 관련이 깊다.

: **목적함수**) 임베딩 된 중심단어와 주변단어 벡터의 내적이 전체 말뭉치에서의 동시 등장 확률이 되도록 만드는 것

$$U_i * V_j = X_{ij}$$

: **손실함수**) 중심단어 벡터 U_i , 주변단어 벡터 V_j 의 내적과 두 단어 i, j 가 동시에 등장할 빈도인 X_{ij} 의 차이가 최소가 되도록 하는 U_i, V_j 을 찾을 것.

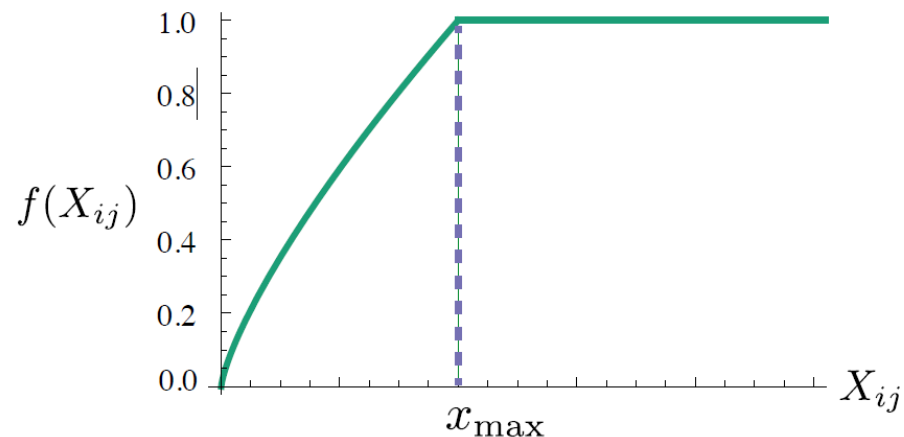
$$J = \sum_{i,j=1}^V (w_i^T \tilde{w}_j + b_i + \tilde{b}_j - \log(X_{ij}))^2$$

(V = 단어 집합의 크기)

!!! 문제발생

0이 되면(즉 동시등장 빈도가 0) 손실함수가 최소가 되는 두 벡터값을 찾지 못하는데???

- 동시등장행렬 X 는 '희소행렬' 일 가능성이 다분하다.
- 즉 X 의 많은 값들이 '0'이거나, 그 빈도가 적어서 많은 값들이 아주 작은 수치가 될 경우가 많다.
- 동시등장빈도인 X_{ij} 가 굉장히 낮으면 정보에 거의 도움이 안된다.
- 따라서, 정보에 도움이 되도록 X_{ij} 에 '**가중치 함수 $f(X_{ij})$** '을 도입한다.



- X_{ij} 작으면 → 상대적으로 작은 값 갖도록
- X_{ij} 크면 → 상대적으로 큰 값 갖도록 (단, 지나치게 높다고 해서 지나치게 큰 가중치를 주는 걸 방지하기 위해 최댓값이 1이 정해져있음)

GloVe

4) 손실함수(Loss function)

: **설립가설**) 문맥 내에서 두 단어의 동시등장비율은 두 단어의 의미와 관련이 깊다.

: **목적함수**) 임베딩 된 중심단어와 주변단어 벡터의 내적이 전체 말뭉치에서의 동시 등장 확률이 되도록 만드는 것

$$U_i * V_j = X_{ij}$$

: **손실함수**) 중심단어 벡터 u_i , 주변단어 벡터 v_j 의 내적과 두 단어 i, j 가 동시에 등장할 빈도인 A_{ij} 의 차이가 최소가 되도록 하는 u_i, v_j 을 찾을 것.

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

가중치 함수 두 단어 임베딩 벡터의 내적 두 단어 동시등장 빈도 로그값

5) 모델 구조

- ① 학습 말뭉치를 대상으로 타깃단어-문맥단어 행렬 A 를 만든다. ➔ 카운트 기반
- ② 목적함수를 최소화하는 임베딩 벡터를 찾기 위해 동시등장행렬 분해를 수행한다.
- ③ 목적함수를 최소화하기 위해 그래디언트 디센트 기법을 적용해 U, V 를 조금씩 업데이트 한다. ➔ 예측 기반
- ④ 학습이 종료되면 U 를 단어임베딩으로 쓸 수 있다.



4.6절 Swivel

(Submatrix-Wise Vector Embedding Learner, 구글, 2016)

Swivel

- 행렬분해 기반 단어 임베딩
- 어떤 행렬을 분해하나?) PMI(Point-wise Mutual Information) Matrix

1) PMI행렬

$$PMI(x, y) = \log \frac{p(x, y)}{p(x) \times p(y)}$$

** PMI행렬이란??

- 두 확률변수 사이의 상관성을 계량화한 지표
- 두 단어의 등장 빈도가 독립이라고 가정할 때 얼마나 자주 같이 등장하나
- PMI가 클수록 두 토큰은 함께 등장하는 경향이 강하고, 작을수록 함께 등장하지 않으려는 경향이 강하다.
- PMI값의 범위는 말뭉치 크기에 따라 달라지기 때문에 서로 다른 쌍들 간 값을 비교하는 데에는 적합하지 않다 .

Swivel

- 행렬분해 기반 단어 임베딩
- 어떤 행렬을 분해하나?) PMI(Point Mutual Information) Matrix

1) PMI행렬

$$PMI(x, y) = \log \frac{p(x, y)}{p(x) \times p(y)} < 0$$

- 음수
- 두 단어의 동시등장확률이 각각의 단어가 등장할 때보다 작을 때
- 두 단어가 서로 음의 상관성이다
- 의미론적 관점에서 그 값을 신뢰하기 어려움
(말뭉치 크기가 충분히 클 때만 신뢰 가능)

Swivel

5) 손실함수

: PMI행렬의 단점을 극복할 수 있도록 설계함.

>> 두가지를 기억하자!
첫째, PMI값의 범위는 말뭉치 크기에 따라 달라진다.
둘째, 0이하의 값은 말뭉치 크기가 크지 않는 이상 신뢰하기 어렵다.

$$1) \quad L1 = \frac{1}{2} f(x_{ij})(U_i * V_j - PMI(i, j))^2$$

$$2) \quad L0 = \log[1 + \exp(U_i * V_j - PMI^*(i, j))]$$

- $f(x_{ij})$ 함수는 단어 i, j 의 동시 등장 빈도
- $x_{ij} = 0$ 인 경우에는 PMI값이 $-\infty$ 가 되므로 $x_{ij} = 0$ 때의 Loss를 다르게 처리함
 - ① $x_{ij} > 0$ 일때는 $L1$ Loss를 사용
 - ② $x_{ij} = 0$ 일 때는 $L0$ loss를 사용

① 타깃단어와 문맥단어가 윈도우 내에서 단 한번이라도 동시에 등장할 때

- $f(x_{ij})$ 가 클수록 U_i, V_j 벡터 내적 값이 실제 PMI값과 비슷해져야 학습 손실이 줄어든다.
- 다시 말해 단어 i, j 가 자주 등장할 수록 두 단어 벡터의 내적이 PMI값과 일치하게끔 U, V 를 학습한다.

$$L1 = \frac{1}{2} f(x_{ij})(U_i * V_j - PMI(i, j))^2$$

자주 등장하는 단어

최대한 작게

Swivel

5) 손실함수

: PMI행렬의 단점을 극복할 수 있도록 설계함.

>> 두가지를 기억하자!

첫째, PMI값의 범위는 말뭉치 크기에 따라 달라진다.

둘째, 0이하의 값은 말뭉치 크기가 크지 않는 이상 신뢰하기 어렵다.

$$1) \quad L1 = \frac{1}{2} f(x_{ij})(U_i * V_j - PMI(i, j))^2$$

$$2) \quad L0 = \log[1 + \exp(U_i * V_j - PMI^*(i, j))]$$

PMI^* : 단 한번 같이 등장했을 때

- $f(x_{ij})$ 함수는 단어 i, j 의 동시 등장 빈도
- $x_{ij} = 0$ 인 경우에는 PMI 값이 $-\infty$ 가 되므로 $x_{ij} = 0$ 때의 Loss를 다르게 처리함
 - ① $x_{ij} > 0$ 일때는 $L1$ Loss를 사용
 - ② $x_{ij} = 0$ 일 때는 $L0$ loss를 사용

② 타깃단어와 문맥단어가 윈도우 내에서 동시에 등장한 적 없을 때

- $f(x_{ij})$ 를 '0'이 아닌 '1'로 가정한다(즉 1번 동시등장했다고 가정).
- PMI^* : 단 한번 같이 등장했다고 가정한 결과값
- 만약 i, j 가 고빈도 단어
 - 두 단어는 정말 같이 등장하지 않는, 의미상 관계가 없는 단어
 - $\log A_{i*}$ 와 $\log A_{*j}$ 의 값이 커지게 된다.
 - 이 경우 Loss값을 작게하려면 $U_i * V_j$ 값을 PMI^* 보다 약간 작게하거나 음수로 만든다.

$$= \log[1 + \exp(\underbrace{U_i * V_j}_{\text{작게 해주기}} - \log|D| + \log A_{i*} + \log A_{*j})]$$

↑ ↑ ↑
값이 커지게 된다 값이 커지게 된다 값이 커지게 된다

- 만약 i, j 가 저빈도 단어
 - 두 단어는 의미상 관계가 일부 있다.
 - 왜냐하면 말뭉치 크기가 작아서 해당 쌍의 동시등장빈도가 전혀 없다고 할 수도 있기 때문에(unobserved).
 - $\log A_{i*}$ 와 $\log A_{*j}$ 의 값이 작아지므로 $U_i * V_j$ 에 대한 제약이 없어진다.
 - $U_i * V_j$ 을 PMI^* 보다 크게하여 손실을 줄인다.

$$= \log[1 + \exp(\underbrace{U_i * V_j}_{\text{제약없음. 크게 해주기}} - \log|D| + \log A_{i*} + \log A_{*j})]$$

↓ ↓ ↓
값이 작아진다 값이 작아진다 값이 작아진다

4.7절

단어 임베딩 평가

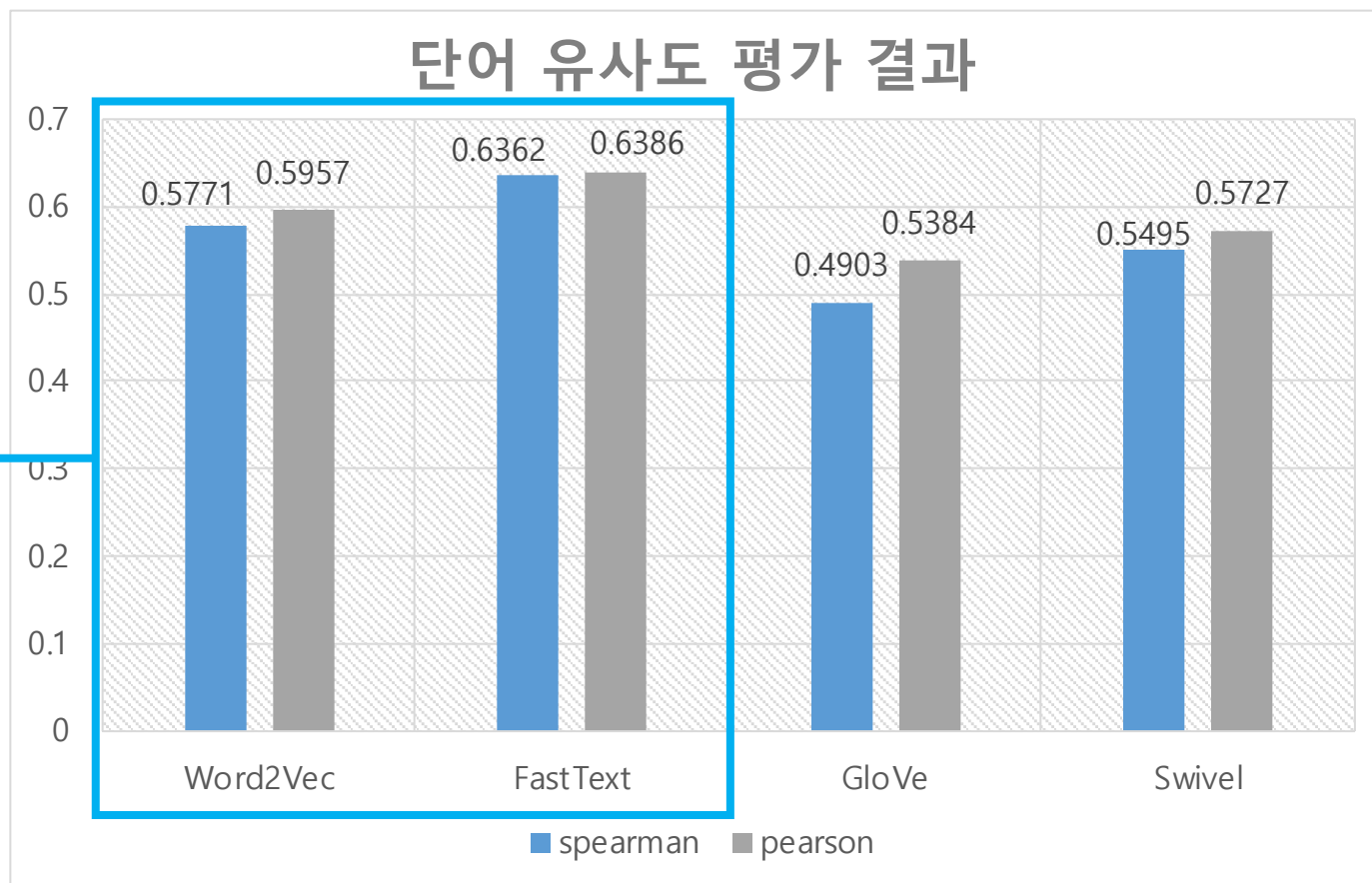
자연어 단어 간 통사론(문법적), 의미론 관계가 임베딩에 얼마나 잘 녹아있나

단어 임베딩 평가 방법

1) 단어 유사도 평가(word similarity test)

- 일련의 단어 쌍을 미리 구성한 후 사람이 평가한 점수와 단어 벡터 간 코사인 유사도 사이의 상관관계를 계산해 단어 임베딩 품질을 평가하는 방법이다.
- 상관관계 척도는 스피어만(spearman) 또는 피어슨(pearson)을 사용한다.
- 1에 가까울수록 사람이 평가한 점수와 코사인 유사도 점수가 비슷하게 나타났다는 뜻으로 임베딩 품질이 더 좋다고 해석할 수 있다.
- Wordsim 데이터는 단어 간 유사도를 0~10점 척도로 사람이 직접 평가한 것이다.

단어 유사도 평가 기준에 있어서는
예측 기반 임베딩 기법(Word2vec, fastText)이
행렬 분해 기법(GloVe, Swivel)보다 **의미적
(semantic) 관계**가 잘 녹아 있다고 할 수 있다.

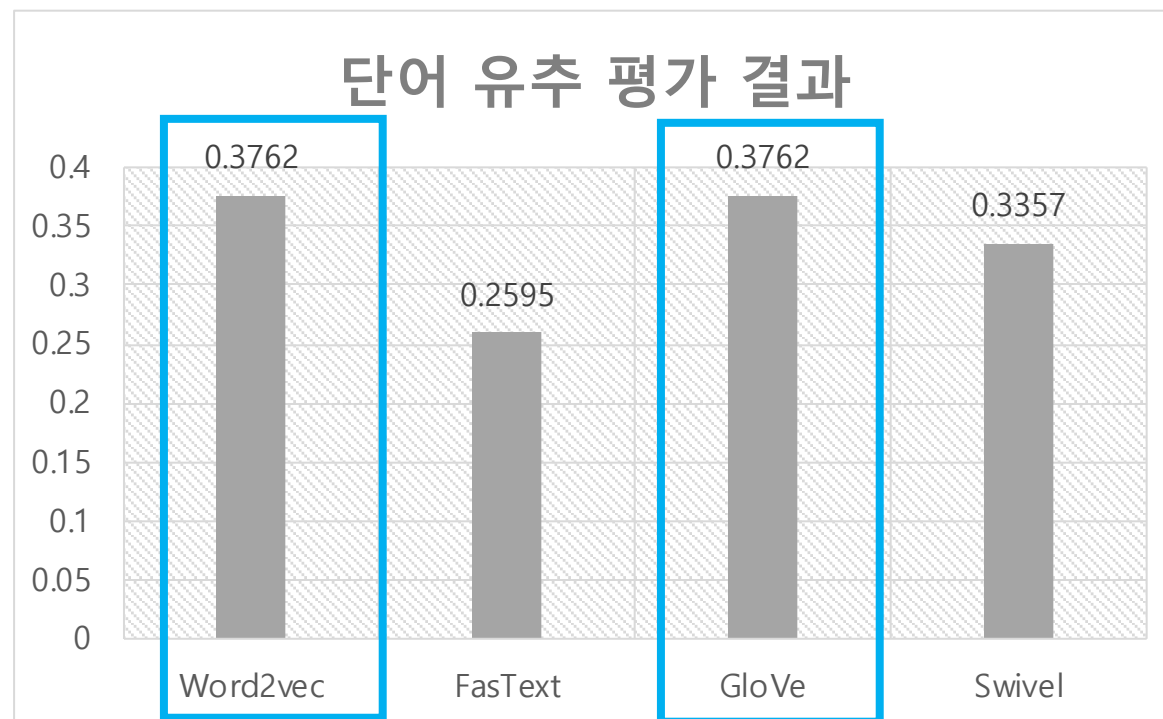


단어 임베딩 평가 방법

2) 단어 유추 평가

- "갑과 을의 관계는 병과 정 의 관계와 같다" 의미론적 유추에서 단어 벡터 간 계산을 통해 "갑-을+정=?? " 이라는 질의에 "병 " 을 도출 할 수 있는지를 평가한다.
- 갑-을+정 " 에 해당하는 벡터에 대해 코사인 유사도가 가장 높은 벡터에 해당하는 단어가 실제로 "병 " 인지 확인한다.
- 단어 유추 평가 기준에 있어서는 Word2vec, GloVe가 상대적으로 우수하다.

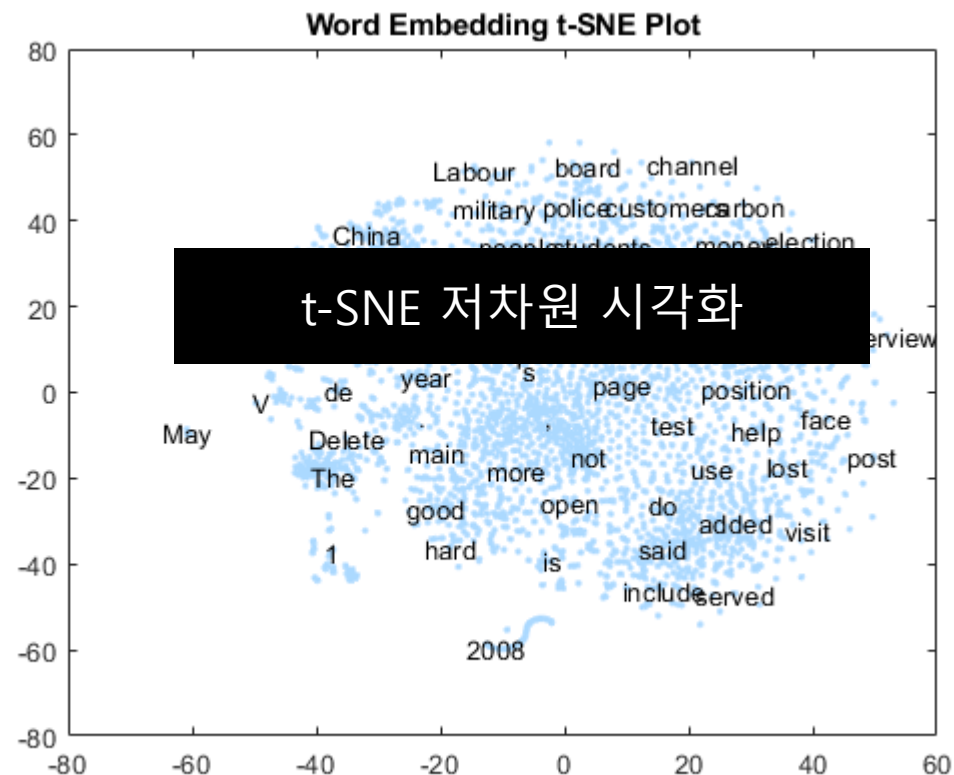
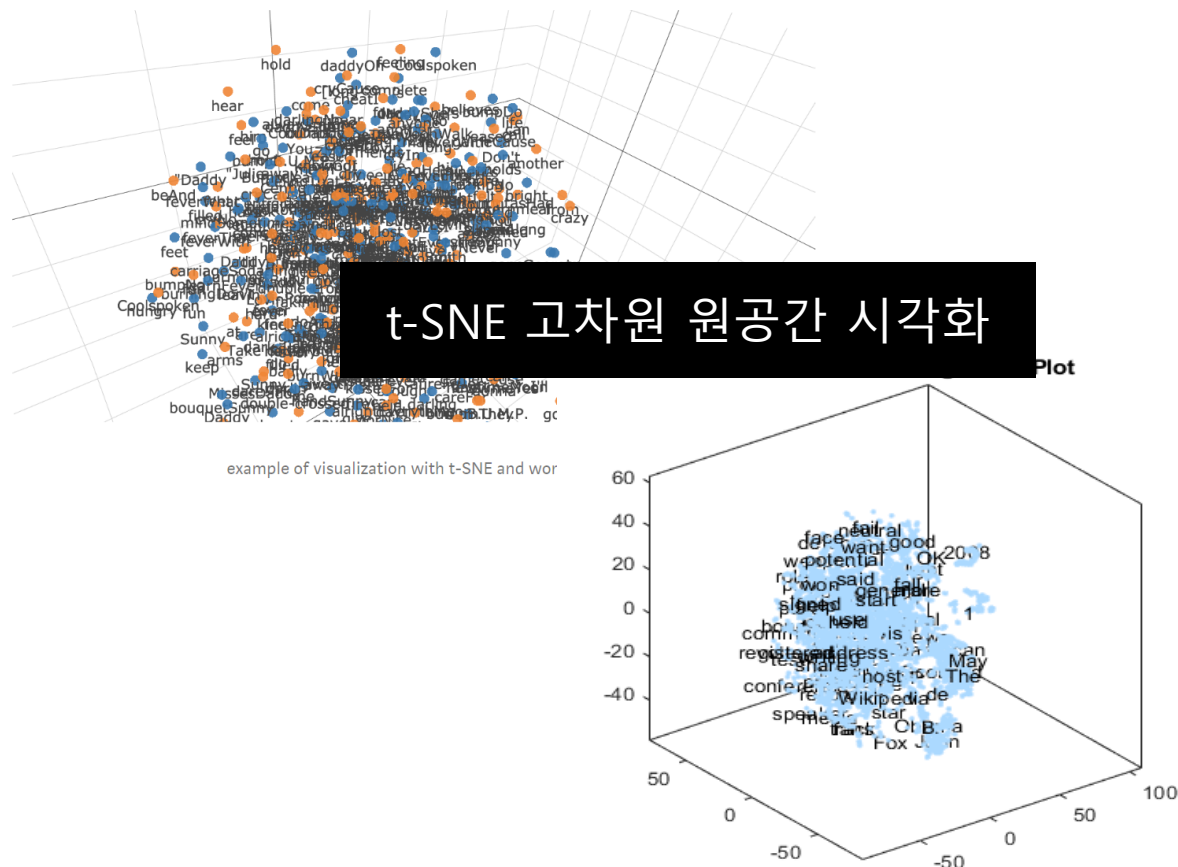
갑	을	병	정
대한민국	서울	일본	도쿄
		중국	베이징
		미국	워싱턴
		영국	런던
		프랑스	파리



단어 임베딩 평가 방법

3) 단어 임베딩 시각화

- 의미가 유사한 단어를 쉽게 이해할 수 있는 형태의 그림으로 표현한 것이다.
- 단어 임베딩은 보통 고차원 벡터이기 때문에 사람이 인식하기 쉬운 2,3차원으로 축소해 시각화한다.
- t-SNE(t-stochastic neighbor embedding)은 고차원 원공간에 존재하는 벡터 x 의 이웃 간 거리를 최대한 보존하는 저차원 벡터 y 를 학습하는 방법론이다.
- Stochastic이라는 이름이 붙은 이유는 거리 정보를 확률적으로 나타내기 때문이다.



4.8절 가중 임베딩

단어임베딩을 문장 수준 임베딩으로 확장하는 방법, 프린스턴대학, 2016

가중 임베딩

1) 주제벡터(discourse vector)

- 문서 내 단어의 등장은 저자가 생각한 주제에 의존한다고 가정한다. 주제에 따라 단어의 사용 양상이 달라진다.
- 단어 w 가 등장할 확률을 최대화하는 주제벡터 c_s 를 찾게 되면 c_s 는 해당 단어의 사용을 제일 잘 설명하는 주제벡터가 된다.


$$p(w|c) = \alpha p(w) + (1 - \alpha) \frac{\exp(v_w \cdot c)}{Z_c}$$

c : 주제벡터
 v_w : 단어 w 의 벡터

단어 w 가 주제와 상관없이 등장할 확률(을/를)

단어 w 가 주제와 관련을 가질 확률
= 주제벡터 c 와 단어벡터 v_w 가 유사할수록(내적값이 클수록) 단어 w 는 주제 c 를 잘 설명한다고 할 수 있음

- 새로운 단어 벡터를 만들 때의 가중치는 해당 단어가 말뭉치에 얼마나 자주 등장하는지 $p(w)$ 를 고려해서 만든다.
- 희귀한 단어가 나오면 높은 가중치를 줘서 해당 단어 벡터의 크기를 키우고, 고빈도 단어라면 해당 벡터의 크기를 줄인다.
- 이는 정보성이 높은, 희귀한 단어에 가중치를 높게 주는 TF-IDF와 비슷하며, 문장 내 단어 등장 순서를 고려하지 않는 점에서 BoW와 비슷하다.



참고문헌

- GloVe
<https://wikidocs.net/22885>
- Swivel
<https://norman3.github.io/papers/docs/swivel.html>

감사합니다.