Neural Word Embedding as Implicit Matrix Factorization

9 Levy & Goldberg, 2014 NIPS

NIPS 2014 의 **“Neural Word Embedding as Implicit Matrix Factorization”** 에서는 **Word2Vec 과 같은 word embedding 이 학습하는 공간과 비슷한 word representation 을 얻는 방법을 소개**합니다. Word - context matrix 에 Point Mutual Information (PMI) 을 적용한 뒤, PMI matrix 에 Singular Vector Decomposition (SVD) 를 적용하면 Word2Vec 과 비슷한 word embedding 공간이 학습됩니다. 이번 포스트에서는 이 논문을 리뷰하고, **PMI + SVD 의 특징**에 대해서 알아봅니다.

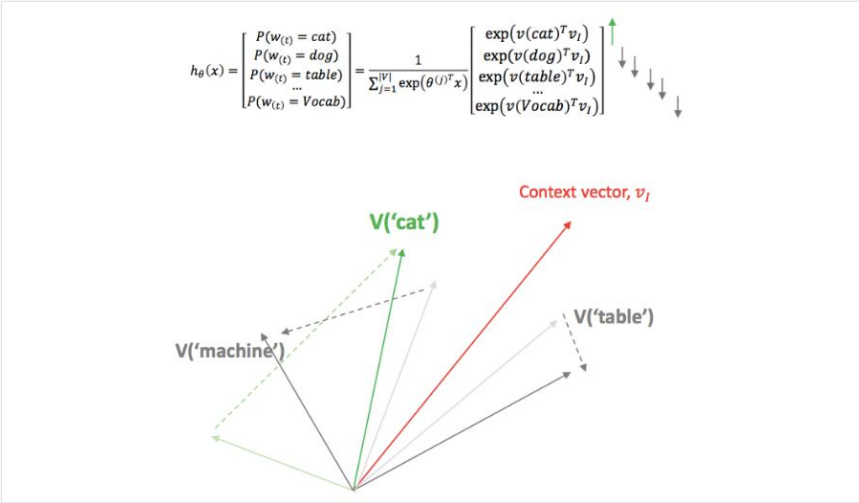
Brief review of Word2Vec

Word2Vec은 Softmax regression 을 이용하여 단어의 의미적 유사성을 보존하는 embedding space 를 학습합니다. 문맥이 유사한 (의미가 비슷한) 단어는 비슷한 벡터로 표현됩니다. ‘cat’ 과 ‘dog’ 의 벡터는 매우 높은 cosine similarity 를 지니지만, 이들은 ‘topic modeling’ 의 벡터와는 매우 작은 cosine similarity 를 지닙니다.

Word2Vec 은 softmax regression 을 이용하여 문장의 한 스냅샷에서 기준 단어의 앞/뒤에 등장하는 다른 단어들 (context words) 이 기준 단어를 예측하도록 classifier 를 학습합니다. 그 과정에서 단어의 embedding vectors 가 학습됩니다.

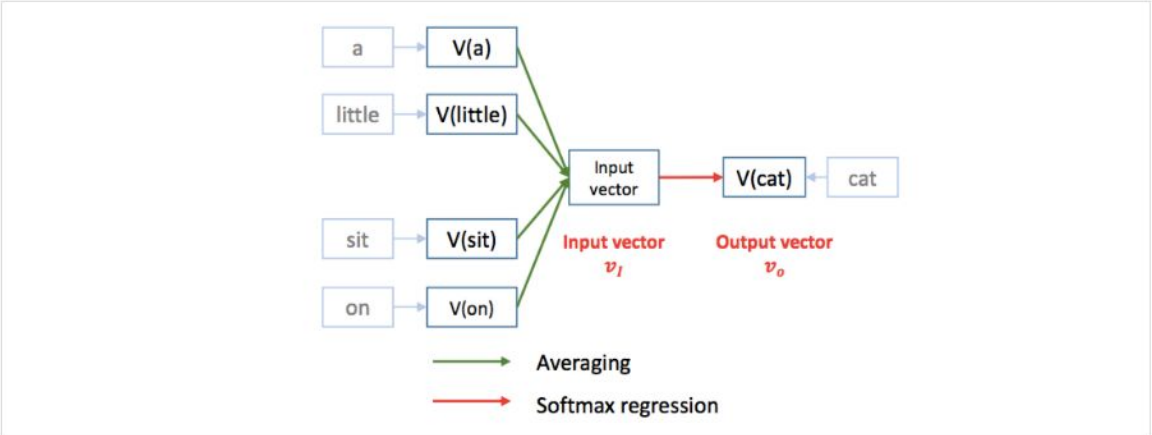
Brief review of Word2Vec

Context vector 는 앞/뒤 단어들의 평균 임베딩 벡터 입니다. [a, little, cat, sit, on, the, table] 문장에서 context words [a, little, sit, on] 를 이용하여 cat 을 예측합니다. 이는 cat 의 임베딩 벡터를 context words 의 평균 임베딩 벡터에 가깝도록 이동시키는 역할을 합니다. 비슷한 문맥을 지니는 dog 도 비슷한 context words 의 평균 임베딩 벡터에 가까워지기 때문에 cat 과 dog 의 벡터가 비슷해집니다.



Brief review of Word2Vec

Word2Vec 은 feed-forward neural language model (Bengio et al., 2003) 을 간소화한 모델입니다. 불필요한 hidden layers 를 제거하고, concatenation 을 averaging 으로 바꿈으로써 모델이 학습해야 하는 벡터 공간의 크기를 줄였습니다. 하지만 본질이 langauge model 이기 때문에 softmax 는 그대로 유지하였습니다.



Brief review of Point Mutual Information

확률에서 두 확률 분포가 **서로 독립인가** (*두 확률은 서로 상관이 없는가*)의 조건은 joint probability와 각각의 marginal probability 의 **곱의 비율이 1**인가 입니다.
식으로는 다음처럼 표현됩니다.

$$\frac{p_{i,j}}{p_i \times p_j}$$

$$I(A;B) = \sum_{a \in A} \sum_{b \in B} P(a,b) \cdot pmi(a;b)$$

$$pmi(x;y) = \log \frac{p(x,y)}{p(x)p(y)} = \log \frac{p(x|y)}{p(x)} = \log \frac{p(y|x)}{p(y)}$$

Brief review of Point Mutual Information

안경을 쓰는 것과 저녁을 먹는 것은 서로 상관없는 문제입니다. 전체 1200 명의 사람 중 저녁을 먹은 사람은 400 명, 1/3 입니다. 1200 명 중 안경을 쓴 사람은 300 명, 1/4 입니다. 그리고 안경을 썼으며, 저녁을 먹은 사람은 1/12 입니다. 각각의 확률의 곱과 같습니다.

다른 관점으로 살펴봅니다. 안경을 썼을 경우로 제한하면, 저녁을 먹은 사람의 비율은 100 명 / 300 명 = 1/3 입니다. 안경을 쓰지 않은 사람이 저녁을 먹은 비율은 300 / 900 = 1/3 입니다. 안경을 썼는지와 저녁을 먹었는지가 서로 상관없는 일이라면, 안경을 썼는지와 상관없이 저녁을 먹은 사람의 비율이 일정해야 합니다. 위 식은 이 말을 수식으로 표현한 것입니다.

$\frac{p_{i,j}}{p_i \times p_j}$ 은 $\frac{p_{j|i}}{p_j}$ 이기 때문

Brief review of Point Mutual Information

PMI 는 위 수식에 log 를 취합니다. log(1)=0 이므로, 서로 상관이 없는 i 와 j 의 pmi 는 0입니다. 서로 양의 상관관계가 있으면 0 보다 큰 값을, 음의 상관관계가 있으면 0 보다 작은 값을 지닙니다.

안경 \ 저녁	저녁을 먹었다	저녁을 안먹었다	marginal prob
안경을 썼다	100	200	$\frac{1}{4} = \frac{300}{1200}$
안경을 안썼다	300	600	$\frac{3}{4} = \frac{900}{1200}$
marginal prob	$\frac{1}{3} = \frac{400}{1200}$	$\frac{3}{4} = \frac{800}{1200}$	..

Brief review of Point Mutual Information

Positive PMI 는 음의 값을 지니는 pmi 를 0 으로 변환합니다. 이는 **pmi 에 threshold = 0 을 적용하여 필터링하는 것**과 같습니다.

$$PPMI(x,y) = max(0, PMI(x,y))$$

Shifted PPMI 는 threshold δ 보다 작은 pmi 는 0 으로 만들고, δ 보다 큰 pmi 에서는 δ 를 뺍니다.

$$SPPMI(x,y) = max(0, PMI(x,y) - \delta)$$

Review of Neural Word Embedding as Implicit Matrix Factorization

Levy & Goldberg (2014) 의 논문의 아이디어입니다. Word2Vec 의 context words 의 embedding matrix 를 C, base word embedding matrix 를 W 라 할 때, $W \cdot C^T = M$ 은 base word 와 context word 의 유사도입니다. 반대로 M 이 있다면, 이를 context words 와 base words 의 embedding vector 로 decomposition 을 할 수 도 있습니다. 이런 맥락에서 **Word2Vec 학습은 matrix factorization 의 역할을 하는 것으로 해석**할 수 있습니다.

Review of Neural Word Embedding as Implicit Matrix Factorization

Negative sampling 을 이용하여 학습하는 skip-gram (SGNS) 의 식을 정리하면 M 은 Shifted PPMI matrix 입니다. **Skip-gram with negative sampling 의 loss function** 은 아래와 같습니다. #(w,c) 는 base word w 와 context word c 의 co-occurrence 이며, k 는 negative sampling loss 의 배수값입니다.

$$l = \sum_{w \in V_W} \sum_{w \in V_C} \#(w,c) \cdot \left(log\delta(\vec{w} \cdot \vec{c}) + k \cdot E_{C_N P_D}[log\delta(-\vec{w} \cdot \vec{c_N})] \right)$$

수학적인 정리 과정을 거치면 다음의 식을 얻을 수 있습니다.

$$\vec{w} \cdot \vec{c} = log\left(\frac{\#(w,c) \cdot |D|}{\#(w) \cdot \#(c)} \cdot \frac{1}{k} \right) = log\left(\frac{\#(w,c) \cdot |D|}{\#(w) \cdot \#(c)} \right) - logk$$

$log\left(\frac{\#(w,c) \cdot |D|}{\#(w) \cdot \#(c)} \right)$ 은 $PMI(w,c)$ 입니다.

그러나 $\#(w,c) = 0$ 인 경우, $log0 = -\inf$ 이기 때문에 정의가 어렵습니다.

Review of Neural Word Embedding as Implicit Matrix Factorization

Word - context matrix 는 sparse matrix 이기 때문에 **co-occurrence 가 0 이거나 작은 경우는 무시**합니다. 이를 위해 PMI 대신 **PPMI 를 이용**합니다.

Experiments: Building word - context matrix

(참고) 이 블로그에서는 2016-10-20 일 뉴스기사와 '박근혜', '아이오아이'의 예시가 자주 등장합니다. 제가 주로 수행한 미등록단어 처리나 토픽모델링, 연관어분석 등의 작업에 이용할만한 한국어 정답데이터가 없었습니다. 그러다보니 이 날의 뉴스를 자주 보아 데이터의 경향을 아예 외웠습니다. 정답데이터가 없으면 분석할 데이터라도 잘 알고 있어야 정성적으로 평가가 가능하니까요. 그래서 이 블로그에서는 해당 날의 뉴스 기사가 예시로 자주 등장합니다.

논문의 내용처럼 **word - context 의 PMI 를 계산하여 word embedding vector 를 학습합니다**. 이 실험을 위하여 2016-10-20 뉴스 기사, 30,091 건을 이용합니다.

Experiments: Building word - context matrix

토크나이징을 위해 미등록단어 문제가 발생하는 trained tokenizers 대신 soynlp 의 Cohesion score 와 L-Tokenizer 를 이용합니다. 이용 당시 soynlp 의 버전은 (0.0.42) 입니다.

```
from soynlp import DoublespaceLineCorpus
from soynlp.word import WordExtractor
from soynlp.tokenizer import LTokenizer

corpus_path = 'YOURS'
corpus = DoublespaceLineCorpus(corpus_path, iter_sent=True)
print('num sents = {}'.format(len(corpus)))

word_extractor = WordExtractor()
word_extractor.train(corpus)
cohesions = word_extractor.all_cohesion_scores()

l_cohesions = {word:score[0] for word, score in cohesions.items()}
tokenizer = LTokenizer(l_cohesions)

print(tokenizer('하루의 뉴스를 학습했습니다'))

 # ['하루', '의', '뉴스', '를', '학습', '했습니다']
```

Experiments: Building word - context matrix

Word - context matrix 를 만들기 위하여 soynlp.vectorizer.sent_to_word_context_matrix 를 이용합니다. 문맥의 길이는 windows=3 으로 설정하며, **최소 10 번 이상 등장한 단어만 이용**합니다.

```
from soynlp.vectorizer import sent_to_word_contexts_matrix

x_news, idx2vocab_news = sent_to_word_contexts_matrix(
    corpus,
    windows=3,
    min_tf=10,
    tokenizer=tokenizer,

    verbose=True)
```

총 48,583 개의 단어로 이뤄진 word - context matrix 를 얻습니다.

```
print(x.shape) (48583, 48583)
```

Experiments: Word similarity using context vector

PMI 를 적용하지 않고 context vector 만을 이용하여도 유사어 검색이 가능합니다. 비슷한 문맥에서 등장한 단어는 비슷한 문맥 벡터를 지니기 때문입니다.

앞으로 유사어를 찾기 위한 계산을 반복할테니 미리 함수를 만들어둡니다. 질의어가 입력되면 cosine distance 기준 가장 가까운 10 개 단어를 찾는 함수를 만듭니다. 질의어와 같은 단어는 출력되지 않도록 합니다. 이 함수에서 vocab2idx, idx2vocab, x 는 context vectors 를 이용할 때와 PMI 를 이용할 때 모두 다르게 적용해야 합니다. 이후 포스트의 뒤에서는 유사어 검색 함수에 대한 이야기는 하지 않습니다.

Experiments: Word similarity using context vector

```
from sklearn.metrics import pairwise_distances

def most_similar_words(query, topk=10):

    if not (query in vocab2idx):

        return []

    query_idx = vocab2idx[query]

    dist = pairwise_distances(x[query_idx,:], x, metric='cosine')[0]

    similars = []

    # sorting
```

Experiments: Word similarity using context vector

```
    for similar_idx in dist.argsort():
        # filtering query term
        if similar_idx == query_idx:
            continue
        if len(similars) >= topk:
            break
        # decoding index to word
        similar_word = idx2vocab[similar_idx]
        similars.append((similar_word, 1-dist[similar_idx]))
    return similars
```

Experiments: Word similarity using context vector

위 함수를 이용하여 4개 질의어의 유사어를 cosine similarity 와 함께 출력합니다. 2016-10-20 은 최순실 - 박근혜 게이트의 언론보도가 시작되던 때입니다. 관련 단어들이 어느 정도 검색이 되고 있음을 볼 수 있습니다.

그런데 잘 이해되지 않은 결과도 있습니다. '아이오아이'의 유사어에 '아프리카'가 있습니다. 이상하여 '아프리카'의 유사어를 확인해보니 일반적인 단어들이 등장합니다. Context vectors 를 이용한 유사어 검색은 언제나 잘 되는 것은 아니라 생각됩니다.

그 이유는 자명합니다. context vectors 의 각 차원 중에는 유사한 단어가 있음에도 불구하고 vector space models 은 이들을 모두 다른 단어로 취급하기 때문입니다.

Experiments: Word similarity using context vector

query = 박근혜	query = 이화여대	query = 아이오아이	query = 아프리카
('박', 0.847)	('이대', 0.729)	('트와이스', 0.581)	('국가', 0.778)
('두테르테', 0.809)	('사퇴했지만', 0.548)	('블랙핑크', 0.546)	('회사', 0.775)
('노', 0.795)	('사퇴한', 0.520)	('에이핑크', 0.542)	('과거', 0.773)
('아키노', 0.776)	('교수', 0.510)	('아프리카', 0.541)	('일부', 0.753)
('오바마', 0.744)	('사임했습니다', 0.509)	('주제', 0.540)	('변화', 0.749)
('노무현', 0.740)	('총장', 0.506)	('현재', 0.539)	('문제', 0.733)
('박정희', 0.731)	('학교', 0.504)	('이야기', 0.534)	('관계', 0.733)
('백악관에', 0.715)	('최', 0.499)	('경우', 0.533)	('경우', 0.726)
('방북', 0.712)	('정씨', 0.492)	('과거', 0.529)	('이야기', 0.723)
('올랑드', 0.704)	('정유라씨', 0.487)	('태도', 0.527)	('도시', 0.721)

Experiments: Word similarity using PMI of word & contexts

논문처럼 PMI 를 계산한 뒤 SVD 를 적용하는 과정을 재현합니다. PMI 계산을 위하여 soynlp.word.pmi 를 이용합니다. min_pmi=0 으로 설정하면 PPMI 를 얻을 수 있습니다.

```
from soynlp.word import pmi as pmi_func

# Shift PPMI at k=0, (equal PPMI)
  pmi= pmi_func(x, min_pmi=0, alpha=0, verbose=True)
```

Singular Vector Decomposition (SVD) 를 학습하기 위하여 scikit-learn 을 이용합니다. 48,583 차원을 300 차원으로 축소하였습니다.

```
from sklearn.decomposition import TruncatedSVD

svd = TruncatedSVD(n_components=300)
  y = svd.fit_transform(pmi)
```

Experiments: Word similarity using PMI of word & contexts

차원이 축소된 y 를 이용하여 유사어를 검색합니다. 이 날 뉴스에서는 '박근혜' > '이화여대' > '아이오아이' 순으로 단어들이 자주 등장하였으며, 각 단어의 문맥들은 대부분 명확합니다. 아래의 결과는 납득이 될 수준의 유사어 검색 결과입니다. '박근혜'의 유사어는 크게 달라지지 않았습니다만, '아이오아이'의 유사어는 많이 개선되었습니다. '불독'은 이 시기에 등장한 프로듀스101 출신 그룹입니다. '몬스'는 '몬스터엑스'가 잘못 토크나이징 된 단어입니다 (몬스터엑스 미안..). 대체로 아이돌그룹의 이름이나 노래 제목이 유사어로 검색됩니다.

'아프리카' 역시 유사어 검색 성능 향상이 눈에 띕니다. 나라 이름이나 대륙 이름이 유사어로 검색됩니다.

차원을 축소하는 과정에서 어떤 semantics 이 하나의 축 혹은 하나의 방향으로 뭉쳐졌으리라 짐작이 됩니다. 그렇다면 축소한 차원의 개수를 더 작게 줄이면 semantics 이 제대로 남지 않을 수 있습니다.

Experiments: Word similarity using PMI of word & contexts

query = 박근혜	query = 이화여대	query = 아이오아이	query = 아프리카
('대통령', 0.849)	('이대', 0.905)	('신용재', 0.809)	('남미', 0.525)
('박', 0.795)	('최경희', 0.845)	('오블리스', 0.807)	('유럽', 0.510)
('정권', 0.722)	('이화여자대학교', 0.837)	('불독의', 0.773)	('중남미', 0.503)
('국정', 0.713)	('특혜', 0.818)	('백퍼센트', 0.765)	('중동', 0.498)
('비선', 0.676)	('사퇴했지만', 0.811)	('몬스', 0.762)	('호주', 0.496)
('정권의', 0.666)	('총장은', 0.809)	('너무너무', 0.759)	('아시아', 0.482)
('비선실세', 0.661)	('정씨', 0.798)	('갓세븐', 0.755)	('대만', 0.475)
('실세', 0.645)	('총장이', 0.795)	('타이틀곡', 0.752)	('오세아니아', 0.470)
('수석비서관회의에서', 0.640)	('입학', 0.790)	('엠카운트다운', 0.748)	('케냐', 0.465)
('최순실', 0.633)	('특혜입학', 0.783)	('불독은', 0.744)	('해외', 0.463)

Experiments: Word similarity using PMI of word & contexts with various SVD dimension

위 질문을 확인하기 위하여 SVD 의 차원을 다르게 적용하며 유사어 검색을 수행합니다. SVD 를 이용하여 word - context PMI matrix 를 [300, 30, 10] 차원으로 축소하였습니다.

네 개의 질의어 모두 30 차원까지는 semantics 이 어느 정도 보존되지만, 10 차원에서는 조금씩 정보를 잃어감을 발견할 수 있습니다. 정확히는 topic 관련 semantic 은 남아있지만, **(아프리카, 남미)** 와 같은 **단어 level 의 semantics 은 조금씩 사라짐**을 볼 수 있습니다.

Experiments: Word similarity using PMI of word & contexts with various SVD dimension

query=박근혜, SVD d=300	query=박근혜, SVD d=30	query=박근혜, SVD d=10
('대통령', 0.849)	('대통령', 0.934)	('박', 0.980)
('박', 0.795)	('국정', 0.911)	('민주당', 0.978)
('정권', 0.722)	('박', 0.911)	('새누리당', 0.978)
('국정', 0.713)	('정권', 0.909)	('대통령', 0.976)
('비선', 0.676)	('청와대', 0.897)	('문재인', 0.974)
('정권의', 0.666)	('게이트', 0.884)	('대표가', 0.972)
('비선실세', 0.661)	('김대중', 0.871)	('우', 0.967)
('실세', 0.645)	('정권의', 0.867)	('원장은', 0.966)
('수석비서관회의에서', 0.640)	('파문', 0.845)	('국민의당', 0.965)
('최순실', 0.633)	('수석', 0.844)	('더민주', 0.964)

Experiments: Word similarity using PMI of word & contexts with various SVD dimension

query=이화여대, SVD d=300	query=이화여대, SVD d=30	query=이화여대, SVD d=10
('이대', 0.905)	('이대', 0.972)	('원장', 0.987)
('최경희', 0.845)	('최경희', 0.950)	('이대', 0.980)
('이화여자대학교', 0.837)	('총장은', 0.946)	('총장', 0.978)
('특혜', 0.818)	('입학', 0.932)	('정', 0.977)
('사퇴했지만', 0.811)	('학사', 0.930)	('최', 0.971)
('총장은', 0.809)	('총장이', 0.929)	('변호사', 0.970)
('정씨', 0.798)	('이화여자대학교', 0.928)	('총장은', 0.959)
('총장이', 0.795)	('정씨', 0.921)	('최경희', 0.957)
('입학', 0.790)	('사퇴했지만', 0.917)	('위원장', 0.956)
('특혜입학', 0.783)	('특혜', 0.916)	('현직', 0.951)

Experiments: Word similarity using PMI of word & contexts with various SVD dimension

query=아이오아이, SVD d=300	query=아이오아이, SVD d=30	query=아이오아이, SVD d=10
('신용재', 0.809)	('몬스', 0.958)	('라디오스타', 0.989)
('오블리스', 0.807)	('샤이니', 0.957)	('불독', 0.989)
('불독의', 0.773)	('불독은', 0.949)	('불독은', 0.987)
('백퍼센트', 0.765)	('멤버', 0.941)	('예능프로그램', 0.987)
('몬스', 0.762)	('불독의', 0.939)	('몬스', 0.987)
('너무너무', 0.759)	('불독', 0.936)	('사랑하기', 0.986)
('갓세븐', 0.755)	('타엑스', 0.929)	('한끼줍쇼', 0.986)
('타이틀곡', 0.752)	('엑소', 0.929)	('구르미', 0.985)
('엠카운트다운', 0.748)	('타이틀곡', 0.928)	('주연을', 0.984)
('불독은', 0.744)	('너무너무', 0.928)	('달빛', 0.984)

Experiments: Word similarity using PMI of word & contexts with various SVD dimension

query=아프리카, SVD d=300	query=아프리카, SVD d=30	query=아프리카, SVD d=10
('남미', 0.525)	('남미', 0.897)	('코', 0.973)
('유럽', 0.510)	('호주', 0.882)	('에선', 0.971)
('중남미', 0.503)	('유일한', 0.881)	('엘리', 0.971)
('중동', 0.498)	('자국', 0.880)	('전쟁', 0.969)
('호주', 0.496)	('에선', 0.880)	('비롯', 0.966)
('아시아', 0.482)	('이스라엘', 0.878)	('에서도', 0.963)
('대만', 0.475)	('대만', 0.877)	('인도', 0.963)
('오세아니아', 0.470)	('노리는', 0.874)	('박사는', 0.961)
('케냐', 0.465)	('칠레', 0.874)	('계에', 0.961)
('해외', 0.463)	('인도', 0.874)	('성공을', 0.960)

Experiments: Comparison “Word2Vec” vs “PMI + SVD” vs Context vector + SVD” vs. “Context vector + PMI”

Word2Vec 모델도 학습하여 PMI + SVD 와 비교합니다. PMI + SVD 는 300 차원, Word2Vec 은 100 차원입니다. Context vector 만으로는 semantic 이 아주 잘 학습되지는 않음을 확인하였습니다. PMI + SVD 가 semantics 을 학습 할 수 있던 원동력이 PMI 에 있는지, 아니면 SVD 에 있는지를 확인하기 위하여 context vector + SVD 및 Context vector + PMI 도 함께 비교합니다. Context vector 는 L2 norm 을 적용한 뒤 SVD 를 학습한 버전과 co-occurrence 를 그대로 이용한 두 버전을 모두 비교합니다.

Experiments: Comparison “Word2Vec” vs “PMI + SVD” vs Context vector + SVD” vs. “Context vector + PMI”

```
class Word2VecCorpus:
    def __init__(self, sents, tokenizer):
        self.sents = sents
        self.tokenizer = tokenizer
    def __iter__(self):
        for sent in self.sents:
            words = self.tokenizer(sent)
            if not words:
                continue
            yield words

corpus.iter_sent = True
word2vec_corpus = Word2VecCorpus(corpus, tokenizer)

from gensim.models import Word2Vec

 word2vec_model = Word2Vec(word2vec_corpus, min_count=10, window=3)
```

Experiments: Comparison “Word2Vec” vs “PMI + SVD” vs Context vector + SVD” vs. “Context vector + PMI”

Word2Vec 과 PMI + SVD 의 경향이 조금 다르기는 합니만, semantics 이 보존되어 유사어 검색이 이뤄짐을 볼 수 있습니다.

이와 다르게 context vectors 에 SVD 를 적용하여도 semantic 이 더 잘 학습되었다는 느낌을 받기는 어렵습니다. 하지만 context vector 에 PMI 를 적용하면 Word2Vec 과 비슷한 결과가 나옵니다.

Word2Vec 과 PMI + SVD 관계의 핵심은 PMI 입니다. Frequent words 는 contexts 에서 빈번하게 등장합니다. Co-occurrence frequency 만을 고려하면 빈번한 단어와의 co-occurrence 만을 고려할 가능성이 높습니다. PMI 는 word 와 contexts 가 상대적으로 얼마나 함께 등장하였는지를 고려합니다. Word - context matrix 에서의 PMI 는 frequent context words 의 frequency bias 를 제거하는 term weighting 역할을 합니다.

Experiments: Comparison “Word2Vec” vs “PMI + SVD” vs Context vector + SVD” vs. “Context vector + PMI”

질의어 ‘박근혜’의 유사 단어

Word2Vec	PMI + SVD_300	norm(Context) + SVD_300	Context + SVD_300	Context + PMI
(‘노무현’, 0.780)	(‘대통령’, 0.849)	(‘위도도’, 0.885)	(‘위도도’, 0.896)	(‘대통령’, 0.334)
(‘테메르’, 0.774)	(‘박’, 0.795)	(‘노’, 0.883)	(‘두테르테’, 0.885)	(‘박’, 0.185)
(‘연설문’, 0.765)	(‘정권’, 0.722)	(‘두테르테’, 0.872)	(‘노’, 0.884)	(‘정권’, 0.156)
(‘오바마’, 0.762)	(‘국정’, 0.713)	(‘오바마’, 0.869)	(‘아키노’, 0.879)	(‘최순실’, 0.153)
(‘박’, 0.753)	(‘비선’, 0.676)	(‘올랑드’, 0.863)	(‘오바마’, 0.875)	(‘청와대’, 0.146)
(‘두테르테’, 0.709)	(‘정권의’, 0.666)	(‘박’, 0.857)	(‘올랑드’, 0.873)	(‘정부의’, 0.128)
(‘연설문을’, 0.708)	(‘비선실세’, 0.661)	(‘블라디미르’, 0.848)	(‘국가안보실’, 0.865)	(‘비선’, 0.127)
(‘위도도’, 0.704)	(‘실세’, 0.645)	(‘국가안보실’, 0.845)	(‘박’, 0.864)	(‘정권의’, 0.126)
(‘노’, 0.696)	(‘수석비서관회의에서’, 0.640)	(‘방북’, 0.841)	(‘테메르’, 0.857)	(‘정부’, 0.118)
(‘국가안보실과’, 0.682)	(‘최순실’, 0.633)	(‘아키노’, 0.840)	(‘블라디미르’, 0.855)	(‘의혹’, 0.113)

Experiments: Comparison “Word2Vec” vs “PMI + SVD” vs Context vector + SVD” vs. “Context vector + PMI”

질의어 ‘이화여대’의 유사 단어

Word2Vec	PMI + SVD_300	norm(Context) + SVD	Context + SVD	Context + PMI
(‘이대’, 0.905)	(‘이대’, 0.905)	(‘이대’, 0.854)	(‘이대’, 0.846)	(‘이대’, 0.298)
(‘최경희’, 0.836)	(‘최경희’, 0.845)	(‘총장이’, 0.744)	(‘정유라씨’, 0.760)	(‘최경희’, 0.291)
(‘총장이’, 0.794)	(‘이화여자대학교’, 0.837)	(‘교수협의회’, 0.716)	(‘총장이’, 0.742)	(‘총장이’, 0.264)
(‘교수협의회’, 0.786)	(‘특혜’, 0.818)	(‘고려대’, 0.708)	(‘더블루케이’, 0.737)	(‘특혜’, 0.250)
(‘총장’, 0.779)	(‘사퇴했지만’, 0.811)	(‘최’, 0.702)	(‘한겨레’, 0.735)	(‘이화여자대학교’, 0.228)
(‘총학생회’, 0.772)	(‘총장은’, 0.809)	(‘총장은’, 0.691)	(‘교수’, 0.735)	(‘입학’, 0.221)
(‘변호사’, 0.760)	(‘정씨’, 0.798)	(‘사퇴’, 0.684)	(‘최’, 0.726)	(‘입학과’, 0.211)
(‘정씨의’, 0.752)	(‘총장이’, 0.795)	(‘체육과학부’, 0.684)	(‘입학’, 0.717)	(‘정유라’, 0.209)
(‘체육과학부’, 0.744)	(‘입학’, 0.790)	(‘교수’, 0.682)	(‘사퇴’, 0.713)	(‘정유라씨의’, 0.208)
(‘부장판사’, 0.742)	(‘특혜입학’, 0.783)	(‘한겨레’, 0.680)	(‘경기’, 0.713)	(‘총장’, 0.190)

Experiments: Comparison “Word2Vec” vs “PMI + SVD” vs Context vector + SVD” vs. “Context vector + PMI”

질의어 ‘아이오아이’의 유사 단어

Word2Vec	PMI + SVD_300	norm(Context) + SVD	Context + SVD	Context + PMI
(‘에이핑크’, 0.833)	(‘신용재’, 0.809)	(‘트와이스’, 0.817)	(‘트와이스’, 0.863)	(‘신용재’, 0.236)
(‘샤이니’, 0.828)	(‘오블리스’, 0.807)	(‘에이핑크’, 0.790)	(‘아프리카발톱개구리’, 0.834)	(‘너무너무’, 0.232)
(‘타이틀곡’, 0.801)	(‘불독의’, 0.773)	(‘아프리카발톱개구리’, 0.785)	(‘칼라’, 0.830)	(‘엠카운트다운’, 0.226)
(‘빅스’, 0.793)	(‘백퍼센트’, 0.765)	(‘언니’, 0.784)	(‘언니’, 0.825)	(‘갓세븐’, 0.184)
(‘불독의’, 0.788)	(‘몬스’, 0.762)	(‘주니어’, 0.784)	(‘아프리카’, 0.825)	(‘걸그룹’, 0.183)
(‘엑소’, 0.779)	(‘너무너무’, 0.759)	(‘왕자’, 0.776)	(‘썬코어’, 0.821)	(‘컴백’, 0.181)
(‘트와이스’, 0.775)	(‘갓세븐’, 0.755)	(‘아프리카’, 0.771)	(‘사우어크라우트’, 0.821)	(‘완전체로’, 0.178)
(‘몬스’, 0.769)	(‘타이틀곡’, 0.752)	(‘비버’, 0.768)	(‘에이핑크’, 0.818)	(‘엠카’, 0.172)
(‘다이아’, 0.767)	(‘엠카운트다운’, 0.748)	(‘무대’, 0.762)	(‘스트레인지’, 0.818)	(‘오블리스’, 0.169)
(‘씨엔블루’, 0.756)	(‘불독은’, 0.744)	(‘창원시’, 0.761)	(‘톰’, 0.818)	(‘다비치’, 0.167)

Experiments: Comparison “Word2Vec” vs “PMI + SVD” vs Context vector + SVD” vs. “Context vector + PMI”

질의어 ‘아프리카’의 유사 단어

Word2Vec	PMI + SVD_300	norm(Context) + SVD	Context + SVD	Context + PMI
(‘인도’, 0.748)	(‘남미’, 0.525)	(‘과거’, 0.895)	(‘과거’, 0.919)	(‘밴쯔’, 0.130)
(‘에어비앤비’, 0.748)	(‘유럽’, 0.510)	(‘회사’, 0.891)	(‘회사’, 0.918)	(‘발톱’, 0.112)
(‘러시아’, 0.740)	(‘중남미’, 0.503)	(‘국가’, 0.882)	(‘국가’, 0.906)	(‘대도서관’, 0.101)
(‘외국’, 0.709)	(‘중동’, 0.498)	(‘일부’, 0.873)	(‘일부’, 0.895)	(‘윰댕’, 0.100)
(‘터키’, 0.702)	(‘호주’, 0.496)	(‘자유’, 0.859)	(‘변화’, 0.889)	(‘중남미’, 0.098)
(‘호주’, 0.699)	(‘아시아’, 0.482)	(‘변화’, 0.858)	(‘시대’, 0.886)	(‘밴쯔는’, 0.095)
(‘인도네시아’, 0.680)	(‘대만’, 0.475)	(‘시대’, 0.852)	(‘관계’, 0.885)	(‘응구기’, 0.093)
(‘최고경영자’, 0.672)	(‘오세아니아’, 0.470)	(‘관계’, 0.848)	(‘소비’, 0.882)	(‘유럽’, 0.093)
(‘유럽’, 0.667)	(‘케냐’, 0.465)	(‘소비’, 0.847)	(‘자동차’, 0.881)	(‘케냐’, 0.092)
(‘주류’, 0.665)	(‘해외’, 0.463)	(‘도시’, 0.846)	(‘핑크’, 0.880)	(‘중동’, 0.087)

Experiments: Comparison “Word2Vec” vs “PMI + SVD” vs Context vector + SVD” vs. “Context vector + PMI”

‘아프리카’ 유사어에는 ‘유럽’, ‘케냐’, ‘중동’과 같은 지리의 의미도 있지만, ‘윰댕’, ‘대도서관’ 같은 아프리카TV 의 BJ 이름들도 등장합니다. 두 의미가 구분이 되지 않았음을 볼 수 있습니다.

Conclusion

Word - context matrix 에 PMI 를 이용하여 term weighting 을 한 뒤, SVD 를 이용하여 차원을 축소하면 semantic 을 보존하는 distributed representation 을 얻을 수 있습니다.

Word - context PMI matrix 의 차원을 축소하는 과정에서 principal components 들이 서로 다른 의미를 지닐 수도 있다는 의심을 하였습니다. SVD 를 이용하여 차원을 매우 작게 축소하면, topical similarity 만 남는다는 느낌을 받았습니다. Word2Vec 의 공간을 이해하는 힌트가 될지도 모르겠습니다.

또한 context vector 의 dimension reduction 을 하다보면 '아프리카'의 Word Sense Disambiguation (WSD) 을 할 수 있을 것 같습니다.

References

Levy, O., & Goldberg, Y. (2014). Neural word embedding as implicit matrix factorization. In Advances in neural information processing systems (pp. 2177-2185).