

딥러닝 스터디

Distributed Representations of Words and Phrases and their Compositionality

김제우

목차

1. 제목 보고 흐름 예상하기
2. 모델 구조 Main Feature 보고 예상하기
3. 논문 읽기
4. 정리

Distributed Representations of Words and Phrases and their Compositionality

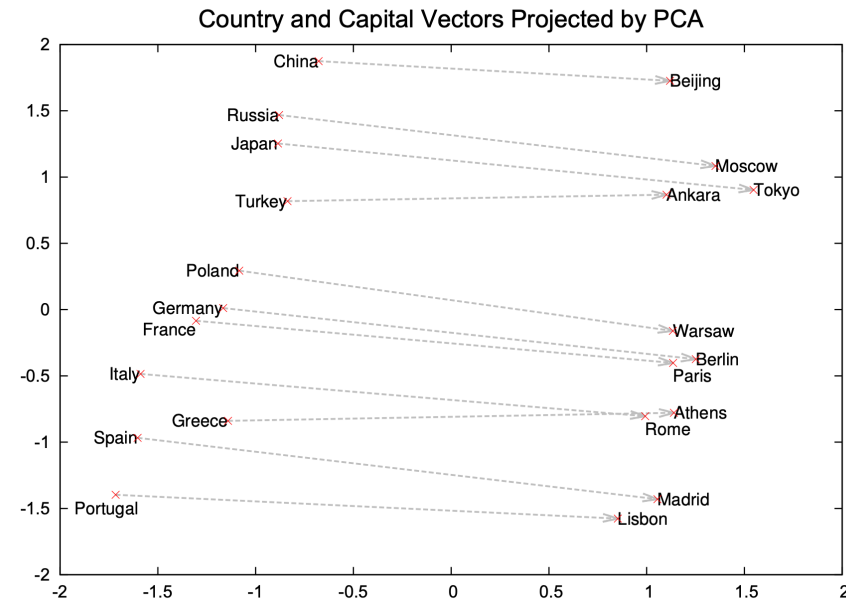
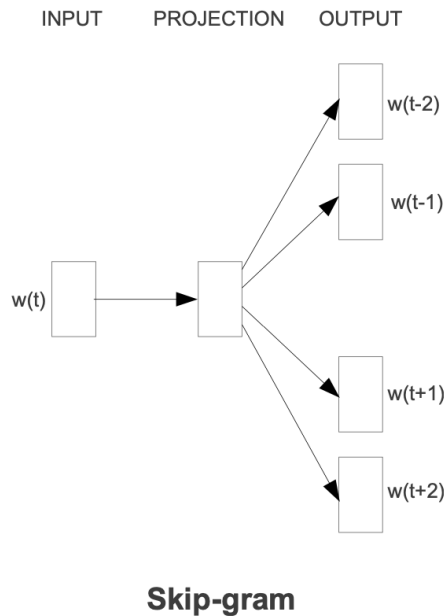
제목 보고 감 잡기

- Distributed representations of words and phrases and their compositionality
- 단어와 구문의 분산 표현과 그들의 구성성??

예상할 수 있는 것

- 단어만 벡터로 만드는게 아닌 구문도 분산 표현
- compositionality -> 이 의미에 대해서 나올 것 같다. 단어들의 구성?

모델 구조 main feature 보고 예상 해보기



예상할 수 있는 것

- Skip-gram의 구조를 활용한 무언가를 할 예정
- w2v에서도 단어의 이런 벡터 표현이 나왔는데 이것로 평가를 했었다. 이후 연구에서 이 데이터셋의 크기를 늘릴거라고 했었음.
- 수도와 국가간의 벡터 위치가 거의 유사한 방향을 가지고 있음

저자 & 출시 시기

- Tomas Mikolov – Google
- Ilya Sutskever – Google
- Kai Chen – Google
- Greg Corrado – Google
- Jeffrey Dean – Google
- 2013년 10월 16일

예상할 수 있는 것

- 아.. 구글
- Tomas Mikolov, Kai Chen, Greg Corrado, Jeffrey Dean 4명 모두 Efficient Estimation of Word Representations in Vector Space(2013년 9월)의 저자들이기 때문에 w2v에서 심화된 내용이 진행될것이다.

Abstract

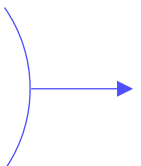
- 최근에 소개되었던 continuous skip-gram model은 많은 정확한 문법적, 의미적 단어 관계를 기록하는 분산 벡터를 높은 성능으로 학습할 수 있는 효과적인 방법이었다.
- 이번 논문에서는 벡터의 성능과 학습 속도를 올릴 수 있는 확장 방법들을 제공할 것이다.
- 빈번한 단어들의 sub샘플링을 통해 매우 빠른 속도 향상을 얻고 더 많은 규칙적인 단어표현들을 학습한다.
- negative sampling이라는 것을 통해 hierarchical softmax를 간단히 대체할 방법을 소개할 것이다.
- 단어 표현은 단어 순서에 대해 고려하지 못하고, 관용 구문의 의미를 담지 못한다는 고유의 제약을 가지고 있다.
- 예를 들면 Canada와 Air 이 두 단어를 Air Canada와 같이 연결 짓기는 어려운 일이다.
- 이런 예시에 영감을 받아, 이런 "구"들을 텍스트속에서 쉽게 찾는 방법과 수백만의 "구"들을 좋은 벡터 형태로 학습하는 것이 가능함을 보여줄 것이다.

- 이전에 제안된 모델 Skip-gram
 - word vector를 학습시키기에 neural network 모델 들보다 행렬곱 연산을 하지 않기 때문에 효율적이었다.
 - 단어의 의미를 명확하게 담고 있는 벡터를 만들 수 있었다.
 - $\text{Vector}(\text{Madrid}) - \text{Vector}(\text{Spain}) + \text{Vector}(\text{France}) = \text{Vector}(\text{Paris})$
- 본 논문에서는 Skip-gram을 몇가지 측면에서 개선할 것이다.
 1. 빈도수 높은 단어를 sub-sampling해서 속도와 성능을 향상 시킴
 2. NCE를 단순화 시켜서 hierarchical softmax를 대체할 것임 -> negative sampling
 3. recursive autoencoder를 사용해서 word vector를 phrase vector로 대체함
- 최종적으로 $\text{vector}(\text{Russia}) + \text{vector}(\text{river}) = \text{vector}(\text{Volga River})$ 와 같이 각 단어가 가지고 있지 않은 의미의 결과물을 표현할 수 있다.

1. Introduction

관용구 (idiomatic phrase)

Boston 보스턴
Glove 글러브



word representation

Boston Glove → phrase representation

Boston과 Glove의 의미 두개와 상관 없는 신문의 이름이다.

word representation은 Boston Glove 두 단어를 합친 의미를 담지 못한다.

“Boston Glove”를 하나의 Dictionary token으로 취급

2. The Skip-gram Model

- Skip-gram의 목표는 문장이나 문서에서 주변 단어들을 예측하는 것
- w_t (가운데 단어)가 주어질때 주변 단어들이 등장할 log 확률을 최대화 함.

주어진 sequence의 길이 ← $\frac{1}{T} \sum_{t=1}^T \sum_{-c \leq j \leq c, j \neq 0} \log p(w_{t+j} | w_t)$

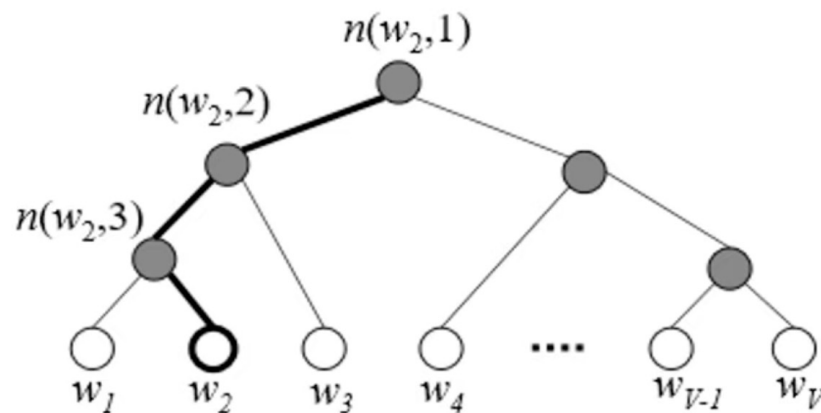
- c 는 문맥의 크기, c 가 커질수록 결과는 좋지만 시간이 늘어남
- 기본 skip-gram 모델을 softmax할때의 수식

$$p(w_O | w_I) = \frac{\exp(v'_{w_O}{}^\top v_{w_I})}{\sum_{w=1}^W \exp(v'_w{}^\top v_{w_I})}$$

- W 는 vocab의 단어의 개수

2.1 Hierarchical Softmax

- Full softmax를 빠르게 연산하기 위해서 Hierarchical softmax를 이용했었다.
- 이진트리를 이용해서 W 의 output node를 표현한다.
- 자주 등장하는 단어의 Path를 앞에 배치해서 빠르게 탐색



Hierarchical Softmax를 위한 Huffman binary tree

2.1 Hierarchical Softmax

- 트리를 통해 정답 단어의 확률만을 다른 단어의 확률을 굳이 구하지 않고 구한다.
- $D \times V$ 만큼의 연산을 $D \times \ln(V)$ 로 줄어든다.

$$p(w|w_I) = \prod_{j=1}^{L(w)-1} \sigma \left(\mathbb{I}[n(w, j+1) = \text{ch}(n(w, j))] \cdot v'_{n(w, j)}{}^T v_{w_I} \right)$$

Notation	
$L(w)$	leaf w 에 도달하기까지의 $path$ 의 길이
$n(w, i)$	$root$ 에서부터 leaf w 에 도달하는 $path$ 에서 만나는 i 번째 $node$ 예) $n(w, 1) = root, n(w, L(w)) = w$
$ch(n)$	$node$ 의 왼쪽 자식
$v'_{n(w, j)}$	W' matrix 대신 사용하는 길이 D 인 $weight vector$
h	$Input word$ 에 대한 $projection matrix row$. 길이 D
$[x] = \begin{cases} 1 & \text{if } x \text{ is true} \\ -1 & \text{otherwise} \end{cases}$	

$$P(n, left) = \sigma(v'_n{}^T \cdot h)$$

: node n 이 left로 갈 확률

$$P(n, right) = \sigma(-v'_n{}^T \cdot h) = 1 - \sigma(v'_n{}^T \cdot h)$$

: sigmoid이므로 위의 식이 성립

$$\begin{aligned} P(w_2 = w_0) &= P(n(w_2, 1), left) \cdot P(n(w_2, 2), left) \cdot P(n(w_2, 3), right) \\ &= \sigma(v'_{n(w_2, 1)}{}^T \cdot h) \cdot \sigma(v'_{n(w_2, 2)}{}^T \cdot h) \cdot \sigma(-v'_{n(w_2, 3)}{}^T \cdot h) \end{aligned}$$



$$P(w = w_0) = \prod_{j=1}^{L(w)-1} \sigma(\mathbb{I}[n(w, j+1) = \text{ch}(n(w, j))] \cdot v'_{n(w, j)}{}^T h)$$

2.2 Negative Sampling

- Hierarchical softmax의 대안으로 Noise Contrastive Estimation(NCE)가 제안되었고 이것의 간소화된 형태를 사용할것이다.
- NCE의 핵심은 데이터를 노이즈로부터 로지스틱 회귀식을 통해 이진 분류함
- Negative Sampling은 실제 문맥 속에 존재하지 않는 단어들을 noise로 간주하고 noise 단어들의 unigram 확률 분포를 통해 sampling
- 중심 단어와 실제 문맥 속 단어의 pair로 구성된 집합 -> 문맥에 있음을 최대화
- 중심 단어와 noise의 pair로 구성된 집합 -> 문맥에 없음을 최소화

$$\theta = \operatorname{argmax}_{\theta} \prod_{(w,c) \in D} P(D = 1 \mid w, c, \theta) \prod_{(w,c) \in \tilde{D}} P(D = 0 \mid w, c, \theta)$$

2.3 Subsampling of Frequent Words

- Subsampling
- 너무 빈번하게 출현하는 단어는 학습 효과를 감소시킴
 - 다빈도 단어를 거를 수 있는 discard probability를 설정 후 sampling 진행
 - 저빈도 단어 벡터의 효과적인 학습 유도
 - 다빈도 단어는 선택될 확률이 낮아지고 저빈도 단어는 선택될 확률을 높인다.

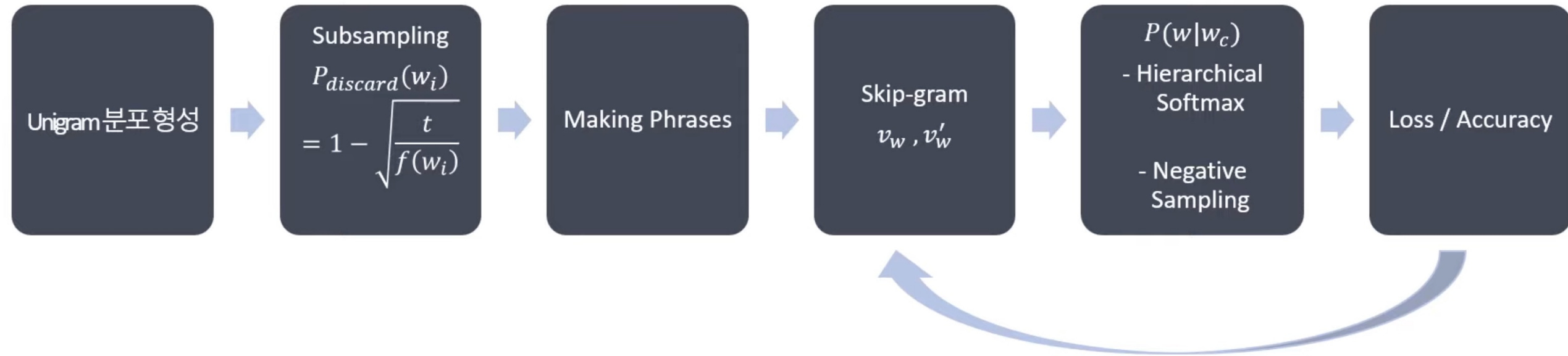
$$P_{discard}(w_i) = 1 - \sqrt{\frac{t}{f(w_i)}}$$

$f(w_i)$: the frequency of word w_i —————> 단어의 빈도수

t : Threshold($\approx 10^{-5}$)

2.3 Subsampling of Frequent Words

딥러닝스터디



- Analogical reasoning task를 이용하여 Hierarchical Softmax, NCE, Negative Sampling, subsampling을 평가한다.
- task는 'Germany': 'Berlin' :: 'France': ?와 같이 구성되어 있고 질문의 답은 cosine 거리가 가까운 벡터를 찾는것으로 실행된다.
- Skip-gram은 10억 개의 단어가 포함되어 있는 internal Google dataset으로 학습했음.
- 실험에서 5번 이하로 등장하는 어휘를 무시해서 672000개의 vocab을 얻었다.

3. Empirical Results

- 실험 결과 Negative Sampling은 Hierarchical Softmax보다 좋은 성능을 보이며, 심지어 NCE보다도 더 좋은 성능을 보였다.
- 빈번한 단어에 대한 subsampling은 training 속도를 향상시켰고, 단어의 정확성도 높였다.

Method	Time [min]	Syntactic [%]	Semantic [%]	Total accuracy [%]
NEG-5	38	63	54	59
NEG-15	97	63	58	61
HS-Huffman	41	53	40	47
NCE-5	38	60	45	53
The following results use 10^{-5} subsampling				
NEG-5	14	61	58	60
NEG-15	36	61	61	61
HS-Huffman	21	52	59	55

- 많은 Phrase들은 개별적인 단어들의 결합으로 얻을 수 없는 의미를 가지고 있다.
- Phrase에 대한 vector표현을 학습하기 위해서 다른 맥락에서는 자주 등장하지 않고 특정 맥락에서만 자주 등장하는 단어쌍을 표현했다.

This is 와 같이 계속 나오는 것은 그대로 둠

Toronto Maple Leafs 같이 드물게 나오는 애들은 unique한 토큰으로 대체

- uni-gram과 bi-gram을 이용하여 다음과 같은 점수를 이용함

$$\text{score}(w_i, w_j) = \frac{\text{count}(w_i w_j) - \delta}{\text{count}(w_i) \times \text{count}(w_j)}.$$

$$\text{score}(w_i, w_j) = \frac{\text{count}(w_i w_j) - \delta}{\text{count}(w_i) \times \text{count}(w_j)}.$$

- 계수로 사용되며 δ 는할인 매우 빈번하지 않은 단어로 구성된 너무 많은 구를 방지한다.
- threshold를 넘어서 score를 가진 bi-gram이 선택되고 그것은 phrase로 사용되게 된다.
- Phrase에 대한 vector표현을 학습하기 위해서 다른 context에서는 자주 등장하지 않고 특정 context에서만 자주 등장하는 단어쌍을 표현했다.

4.1 Phrase Skip-Gram Result

- 앞선 실험과 동일한 데이터로 실험
- phrase 기반의 training corpus를 구성하고 하이퍼 파라미터를 바꿔가며 skip-gram 모델을 훈련했다.
- 차원은 300, context size는 5

Method	Dimensionality	No subsampling [%]	10^{-5} subsampling [%]
NEG-5	300	24	27
NEG-15	300	27	42
HS-Huffman	300	19	47

- Negative sampling은 k(sampling 개수)가 높을 때 더 좋았다.
- subsampling을 HS와 함께 사용할 때 가장 성능이 좋았다.

5. Additive Compositionality

- $\text{Vec}(\text{지역명}) + \text{Vec}(\text{일반명사})$
 - 지역에 해당하는 특정 명사
 - $\text{Vec}(\text{'체코'}) + \text{Vec}(\text{'통화'}) = \text{Vec}(\text{'koruna(체코의 통화단위)'})$
- Skip-gram의 목적은 문맥의 단어를 알맞게 예측하는 것
- 두 단어의 벡터를 더하는 것 = 두 단어의 문맥의 분포 간의 연산을 수행하는 것
- 즉, 두 분포에서 모두 높은 확률을 갖는 단어의 벡터를 찾아온다.

5. Additive Compositionality

- Skip-gram을 이용한 단어나 구에 대한 표현이 단순한 벡터의 구조를 가지고 analogical reasoning에서 정확한 성능을 보였다.
- 벡터의 합으로 표현했을 때의 결과

Czech + currency	Vietnam + capital	German + airlines	Russian + river	French + actress
koruna	Hanoi	airline Lufthansa	Moscow	Juliette Binoche
Check crown	Ho Chi Minh City	carrier Lufthansa	Volga River	Vanessa Paradis
Polish zolty	Viet Nam	flag carrier Lufthansa	upriver	Charlotte Gainsbourg
CTK	Vietnamese	Lufthansa	Russia	Cecile De

Table 5: Vector compositionality using element-wise addition. Four closest tokens to the sum of two vectors are shown, using the best Skip-gram model.

6. Comparison to Published Word Representations 딥러닝스터디

- word analogy task 들과 비교한 결과

Model (training time)	Redmond	Havel	ninjutsu	graffiti	capitulate
Collobert (50d) (2 months)	conyers lubbock keene	plauen dzerzhinsky osterreich	reiki kohona karate	cheesecake gossip dioramas	abdicate accede rearm
Turian (200d) (few weeks)	McCarthy Alston Cousins	Jewell Arzu Ovitz	- - -	gunfire emotion impunity	- - -
Mnih (100d) (7 days)	Podhurst Harlang Agarwal	Pontiff Pinochet Rodionov	- - -	anaesthetics monkeys Jews	Mavericks planning hesitated
Skip-Phrase (1000d, 1 day)	Redmond Wash. Redmond Washington Microsoft	Vaclav Havel president Vaclav Havel Velvet Revolution	ninja martial arts swordsmanship	spray paint grafitti taggers	capitulation capitulated capitulating

- Skip-Phrase가 가장 빨리 학습을 완료했다.

- 단어와 구에 대한 분산적인 표현의 방법을 보여줬다.
- 이전 모델보다 더 큰 데이터에 대해 학습했음. word와 phrase 표현의 quality를 크게 향상한다. 빈번하게 등장하는 단어에 대한 subsampling이 학습 속도와 드물게 등장하는 단어에 대한 정확도를 높일 수 있었음
- 간단한 Negative sampling을 이용하여 빈번하게 등장하는 단어들의 정확도를 향상했음
- Task 에 따라서 hyper-parameter와 model이 달라져야하고, 가장 중요한 점은 모델 구조, vector의 크기, subsampling의 비율, window의 크기였다.
- 단어의 벡터 표현은 단순히 벡터를 더하여 할 수 있다는 것이 흥미로운 점.
- Phrase의 표현을 학습하는데 있어서 더 긴 문장도 관용구로 표현하면서 연산을 단순화 했다.