



Word2Vec

17.07.20

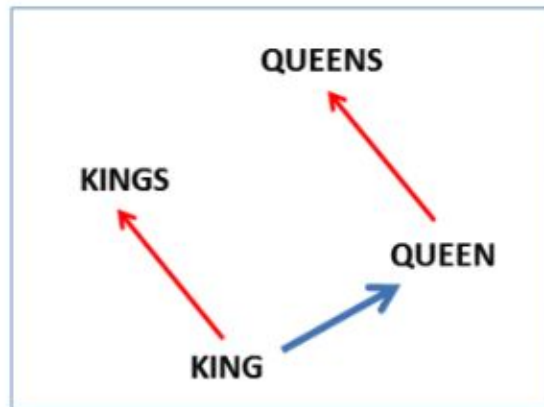
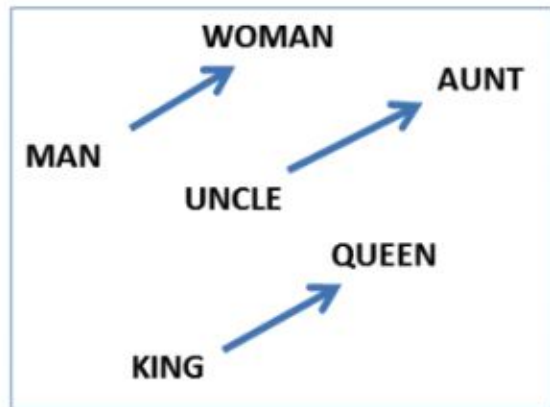


Index

- Word Embeddings
- Word2Vec Model
- Distributed Representations of Words and Phrases and their Compositionality
 - softmax function
 - hierarchical softmax
 - negative sampling
 - subsampling of frequent words
 - Learning Phrases

Word Embeddings

- 단어를 컴퓨터가 인지할 수 있도록 표현할 수 있어야 한다.
- 유니코드 ... 개념적인 차이를 나타내기 힘들다.
- **Vector**로 표현 -> 유사도 측정, 여러 단어 사이의 평균, 벡터 연산을 통한 단어 추론 가능!



(Mikolov et al., NAACL HLT, 2013)


Word Embeddings

- One-hot encoding : '사전(Dictionary)' 상에서 단어의 위치를 벡터로 나타낸 것.

motel [0 0 0 0 0 0 0 0 0 0 1 0 0 0 0]
hotel [0 0 0 0 0 0 0 1 0 0 0 0 0 0 0]

- dimensionality

Word Embeddings

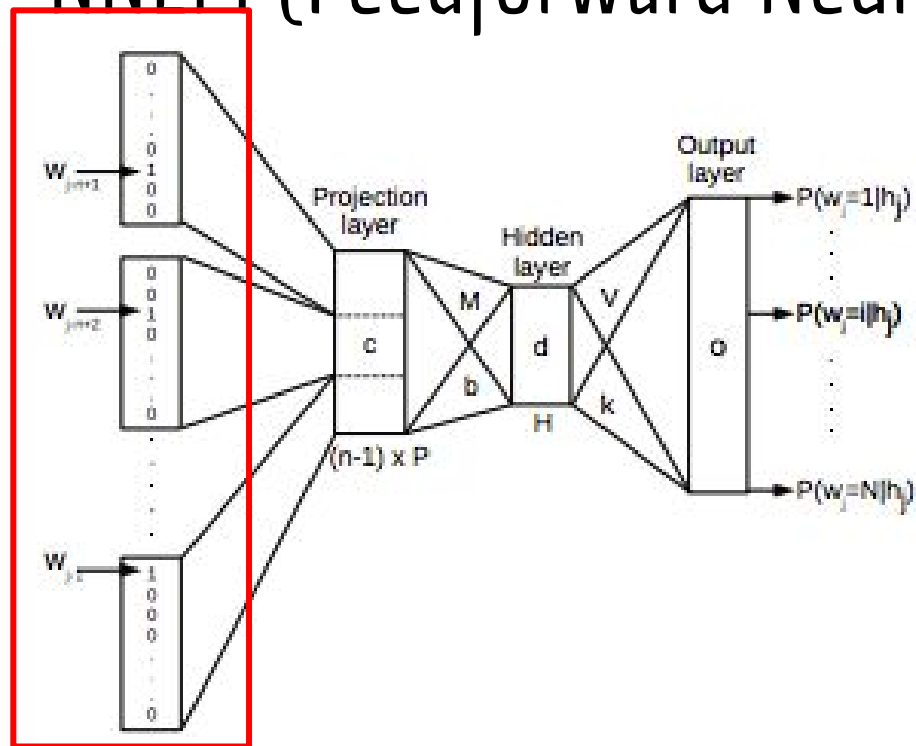
- One-hot encoding : '사전(Dictionary)' 상에서 단어의 위치를 벡터로 나타낸 것.
 - in web search, if user searched for [Seattle motel], we would match documents containing "Seattle hotel" but


The diagram shows two horizontal rows of circles representing one-hot encodings. The top row is labeled 'motel' and has 15 circles, with the 11th circle from the left being a solid blue '1' and all others being white circles with blue outlines. The bottom row is labeled 'hotel' and has 15 circles, with the 10th circle from the left being a solid blue '1' and all others being white circles with blue outlines. To the right of the 'hotel' row is an equals sign followed by a white circle with a blue outline.
 - Orthogonal - no natural notion of **similarity** in a set of 'one-hot' vectors.

Word Embeddings

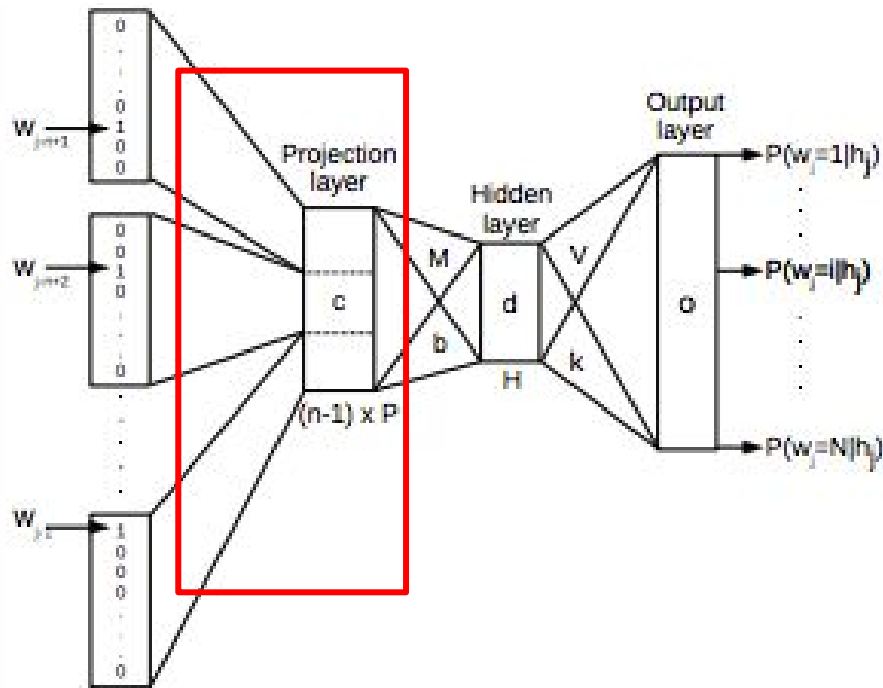
- Idea : “Distributional Hypothesis”, 비슷한 분포를 가진(비슷한 문맥에서 등장하는) 단어들은 비슷한 의미를 가진다.
 - neural network를 이용
 - NNLM
 - RNNLM
 - CBOW
 - Skip-Gram

NNLM (Feedforward Neural Net Language Model)



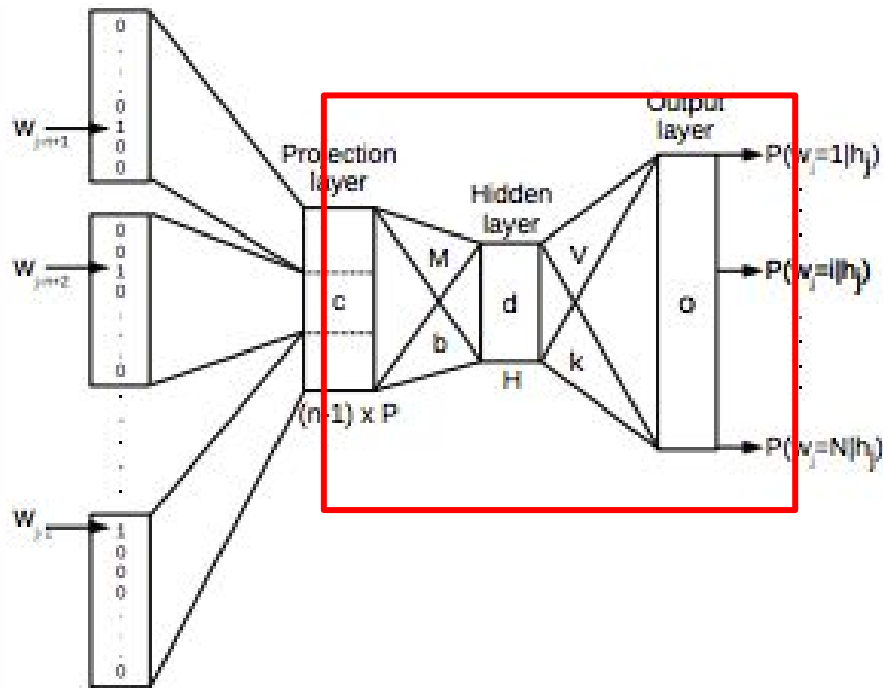
1. 현재 보고 있는 단어 앞에 있는 N 개의 단어를 **one-hot vector**로 표현
 - 사전의 크기가 V 이면 $V \times 1$ vector

NNLM (Feedforward Neural Net Language Model)



2. Projection layer 의 크기가 D (dimensionality)일 때, Projection matrix ($N \times D$) 에 의해 각 단어들이 $D \times 1$ vector가 됨.

NNLM (Feedforward Neural Net Language Model)



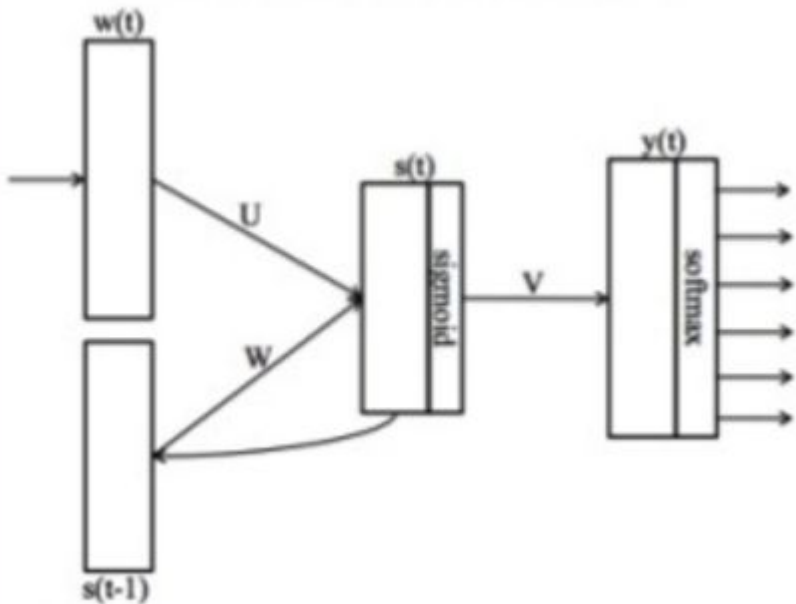
3. 크기 H 의 hidden layer을 거친 output이 각 단어가 나올 확률.

이를 실제 단어의 one-hot vector와 비교, 에러 계산, back propagation 으로 weight matrix를 최적화한다.

NNLM (Feedforward Neural Net Language Model)

- 단점
 - N을 미리 정해주어야함. (몇 개의 단어를 볼건지)
 - 현재 보고 있는 단어의 앞 단어를 고려하지 못함
 - 느림
- Cost
 - $N * D + N * D * H + H * V$

RNNLM(Recurrent -)



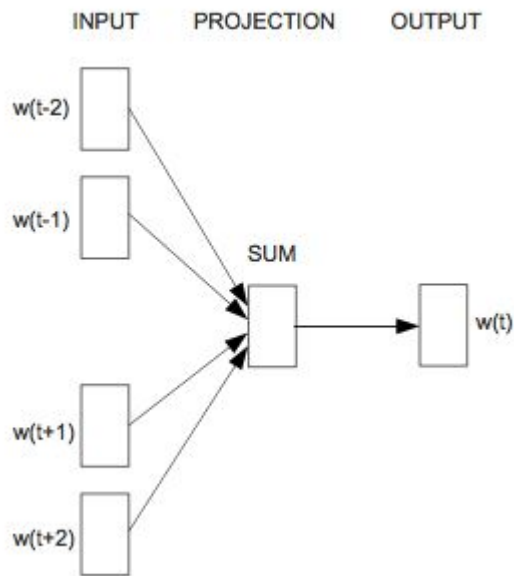
- U 로 word embedding 가져옴.
- $s(t-1)$: recurrent 한 input
- Hidden layer의 길이 H 일 때,
 - > Cost : $H + H*H + H*V$
- 미리 N 정할 필요 없이 순차적으로 학습
- $s(t-1)$ 이 short term memory 역할 - 앞에 오는 단어를 고려할 수 있다.

Word2Vec

- 빠르게 학습할 수 있는 방법이 필요
- 'model architecture for learning distributed representations of words that try to minimize computational complexity'
- Most of the complexity is caused by the non-linear hidden layer in the model.
- simpler model that might not be able to represent the data as precisely as neural networks, but can possibly be trained on much more data efficiently. => **Continuous word vectors, on the top of N-gram NNLM**

CBOW Model (Continuous Bag-of-Words)

- NNLM에서 hidden layer을 없애고 projection layer 를 모든 단어가 공유.



- Projection layer의 길이를 D
- Output 은 V vector (애에 softmax 계산한게 각 단어가 나올 확률)
- 한 단어를 처리하기 위해 앞뒤 C 개의 단어를 본다고 하면,
 $Cost = C * D + D * V$ ($\ln V$ 로 줄일 수 있음)

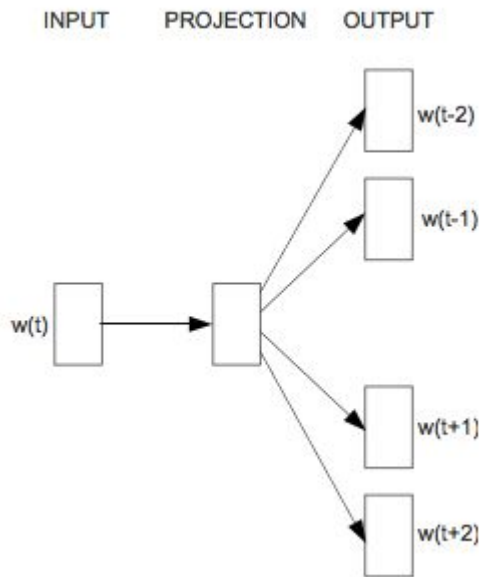
Continuous Skip-gram Model

- CBOW(문맥에서 한 단어를 추측) 와 반대 방향. 주어진 한 단어로부터 다른 단어들을 추

Source Text	Training Samples						
<table><tr><td>The</td><td>quick</td><td>brown</td></tr></table> fox jumps over the lazy dog. ➡	The	quick	brown	(the, quick) (the, brown)			
The	quick	brown					
<table><tr><td>The</td><td>quick</td><td>brown</td><td>fox</td></tr></table> jumps over the lazy dog. ➡	The	quick	brown	fox	(quick, the) (quick, brown) (quick, fox)		
The	quick	brown	fox				
<table><tr><td>The</td><td>quick</td><td>brown</td><td>fox</td><td>jumps</td></tr></table> over the lazy dog. ➡	The	quick	brown	fox	jumps	(brown, the) (brown, quick) (brown, fox) (brown, jumps)	
The	quick	brown	fox	jumps			
<table><tr><td>The</td><td>quick</td><td>brown</td><td>fox</td><td>jumps</td><td>over</td></tr></table> the lazy dog. ➡	The	quick	brown	fox	jumps	over	(fox, quick) (fox, brown) (fox, jumps) (fox, over)
The	quick	brown	fox	jumps	over		

Continuous Skip-gram Model

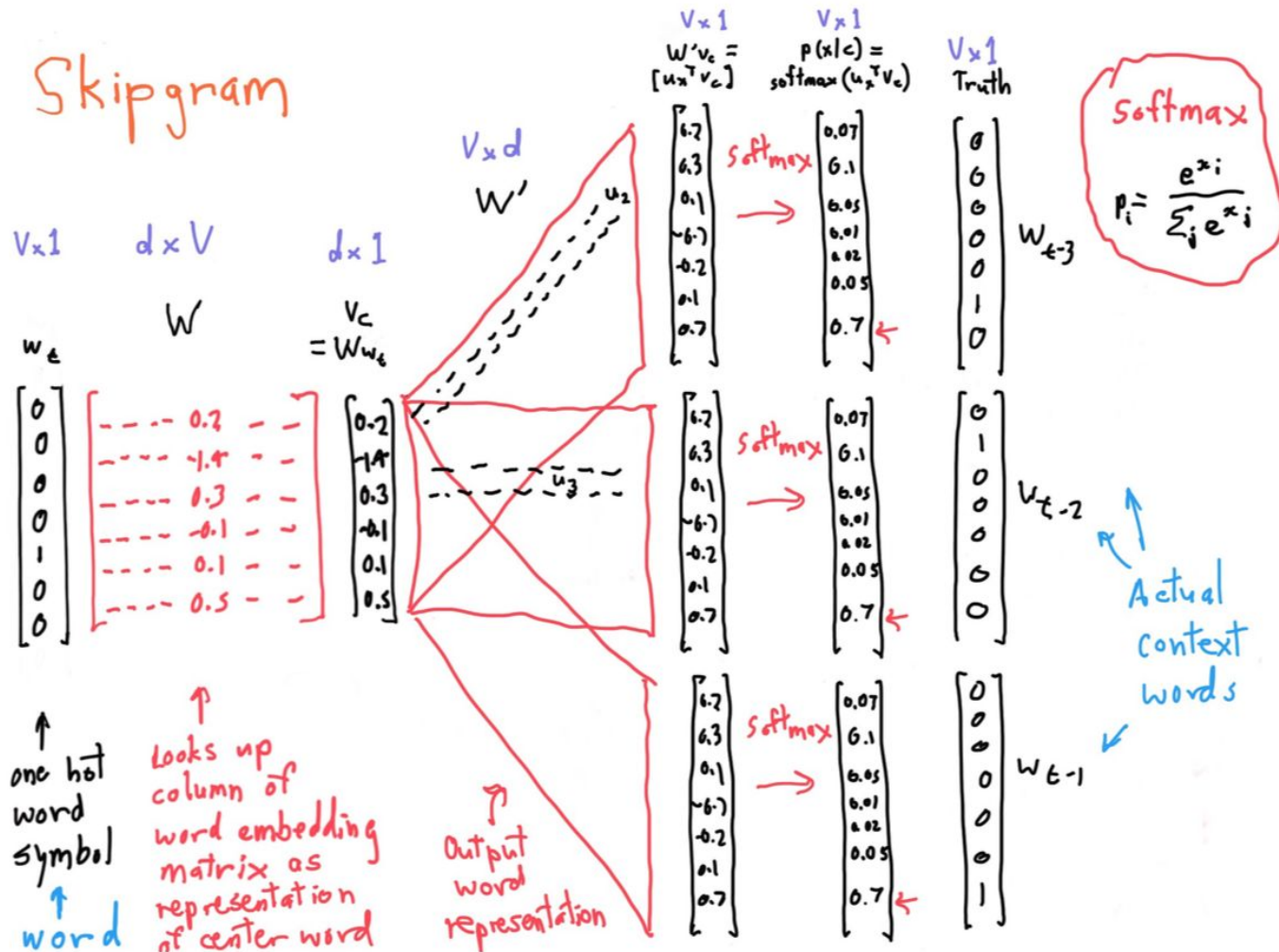
- CBOW(문맥에서 한 단어를 추측) 와 반대 방향. 주어진 한 단어로부터 다른



- Projection layer의 길이를 D
- Output 은 V vector (애에 softmax 계산한게 각 단어가 나올 확률)
- 한 단어를 처리하기 위해 앞뒤 C 개의 단어를 본다고 하면,
$$\text{Cost} = C * (D + D * V(\ln V \text{로 줄일 수 있음}))$$

Conti

Skipgram



Model Architecture	Semantic-Syntactic Word Relationship test set		MSR Word Relatedness Test Set [20]
	Semantic Accuracy [%]	Syntactic Accuracy [%]	
RNNLM	9	36	35
NNLM	23	53	47
CBOW	24	64	61
Skip-gram	55	59	56

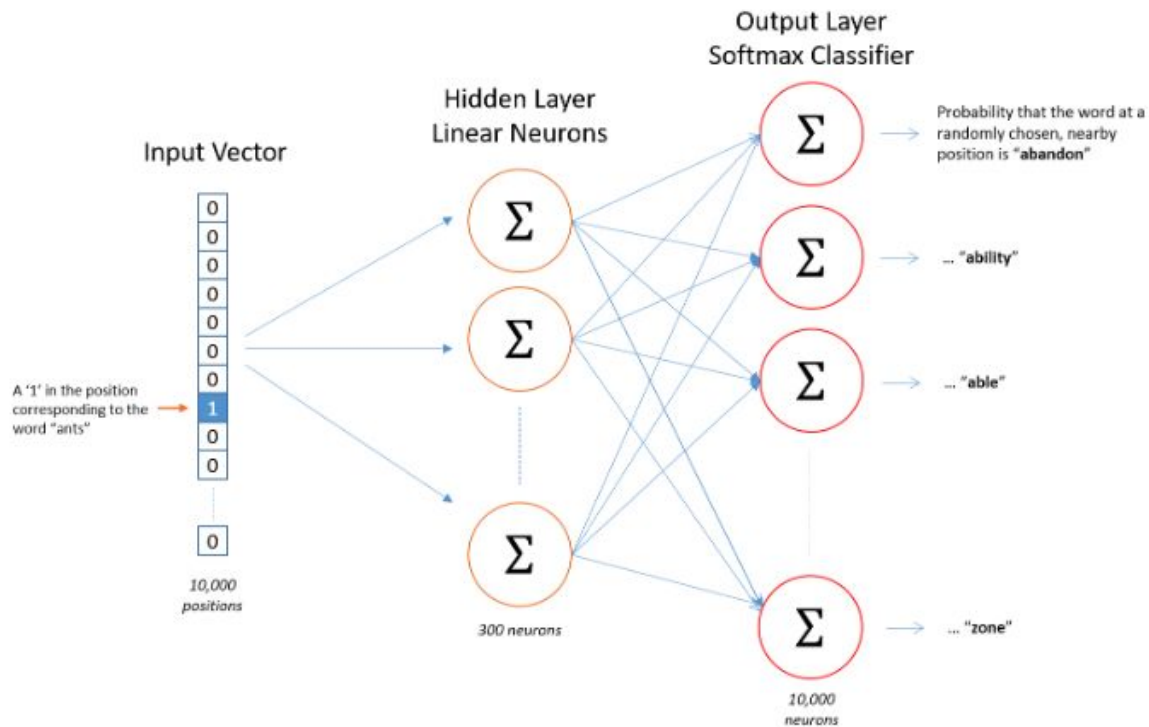
Table 6: *Comparison of models trained using the DistBelief distributed framework. Note that training of NNLM with 1000-dimensional vectors would take too long to complete.*

Model	Vector Dimensionality	Training words	Accuracy [%]			Training time [days x CPU cores]
			Semantic	Syntactic	Total	
NNLM	100	6B	34.2	64.5	50.8	14 x 180
CBOW	1000	6B	57.3	68.9	63.7	2 x 140
Skip-gram	1000	6B	66.1	65.1	65.6	2.5 x 125

Complexity reduction

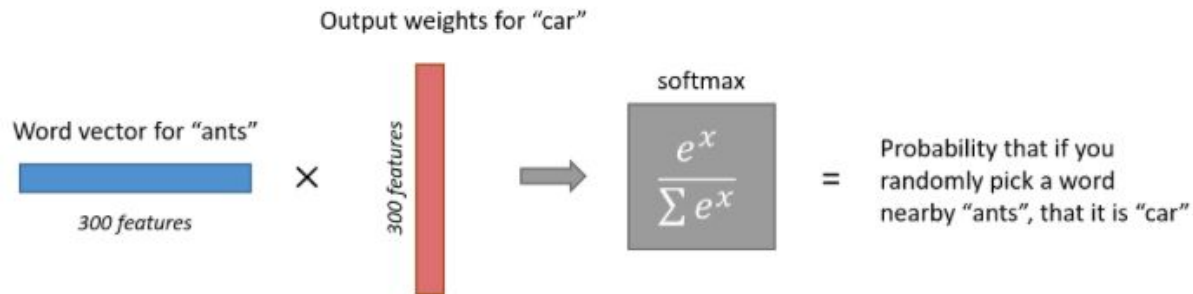
- Hierarchical SoftMax
- Negative Sampling
- Subsampling of Frequent Words

Softmax function



Softmax function

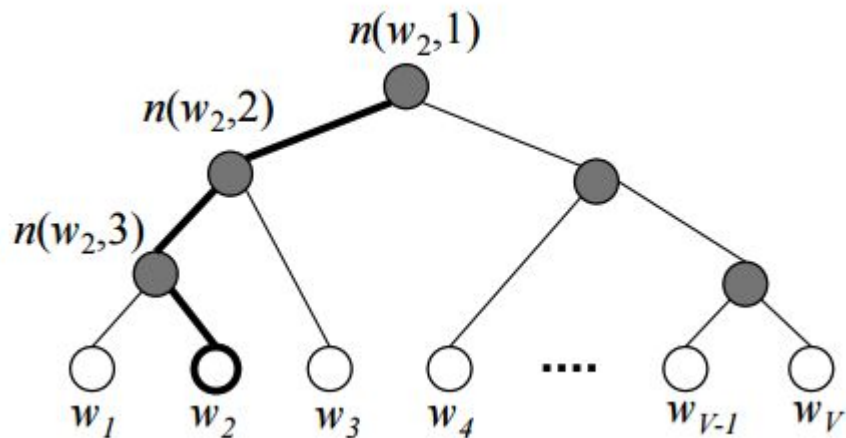
- Softmax Regression - Output 이 0~1, 모든 아웃풋 값의 합이 1이 되게 한다.



$$p(w_O|w_I) = \frac{\exp\left(v'_{w_O}{}^\top v_{w_I}\right)}{\sum_{w=1}^W \exp\left(v'_w{}^\top v_{w_I}\right)} \text{ proportional to } W \text{ (the number of words in Vocab)}$$

Hierarchical Softmax

- Leaf가 각 단어인 Binary tree. => W개의 word 계산 대신 $\ln W$ 개의 노드의 확률 곱



Hierarchical Softmax

$$p(w|w_I) = \prod_{j=1}^{L(w)-1} \sigma \left(\mathbb{I}[n(w, j+1) = \text{ch}(n(w, j))] \cdot v'_{n(w, j)}{}^\top v_{w_I} \right)$$

- Let $\mathbb{I}[x]$ be 1 if x is true, -1 otherwise.
- $\sigma(x) = 1/(1 + \exp(-x)) \Rightarrow p(\text{left}) = 1 - p(\text{right})$
- $\sum_{w=1}^W p(w|w_I) = 1.$
- Using Huffman Tree - 자주 등장하는 단어는 보다 짧은 path로 도달 가능.

$\Rightarrow V$ to $\ln V$

Negative Sampling

- Softmax에서 너무 많은 단어들에 대한 계산이 필요하다 → K개만 뽑아서 계산하자.

$$\log \sigma(v'_{w_O}{}^T v_{w_I}) + \sum_{i=1}^k \mathbb{E}_{w_i \sim P_n(w)} \left[\log \sigma(-v'_{w_i}{}^T v_{w_I}) \right] \quad \text{를 maximize}$$

- $P_n(w)$, noise distribution로부터 logistic regression을 이용해 k개의 negative sample을 뽑는다고 함.
- 실험적으로 $P_n(w)$ 는 unigram distribution의 3/4승에서 좋은 결과를 얻을 수 있었다고 한다.

Subsampling of Frequent Words

- 'a', 'the' 처럼 자주 등장하는 단어들은 provide less information value than the rare words.
- 자주 나오는 단어들을 확률적으로 제외시킨다.

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

- $f(w)$ = 단어 w 의 등장 빈도.
- 실험적으로 $t = 10^{-5}$

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

Learning Phrases

- phrases : 단어 각각의 의미의 simple composition과는 다른 경우도 많음.
- To learn vector representation for phrases, first find words that appear frequently together, and not in other context.
- Simply represent the phrases with a '**single token**'

Learning Phrases-training data(3218 examples)

Newspapers			
New York San Jose	New York Times San Jose Mercury News	Baltimore Cincinnati	Baltimore Sun Cincinnati Enquirer
NHL Teams			
Boston Phoenix	Boston Bruins Phoenix Coyotes	Montreal Nashville	Montreal Canadiens Nashville Predators
NBA Teams			
Detroit Oakland	Detroit Pistons Golden State Warriors	Toronto Memphis	Toronto Raptors Memphis Grizzlies
Airlines			
Austria Belgium	Austrian Airlines Brussels Airlines	Spain Greece	Spainair Aegean Airlines
Company executives			
Steve Ballmer Samuel J. Palmisano	Microsoft IBM	Larry Page Werner Vogels	Google Amazon

Learning Phrases-Skip-gram Results

- phrase based training corpus 만들어서 trained several Skip-gram models.(dim 300, context size 5 with several hyperparameters)

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

- Subsampling also can improve accuracy.
- dataset을 33 billion words로, dim을 1000로 늘리고 hierarchical softmax사용하여 72% accuracy. -> large training data is crucial.

Learning Phrases-Skip-gram Results

- How different the representations learned by different models are.
- 자주 등장하지 않는 phrase 근처에 등장할 확률이 높은 단어들을 살펴보았음.

	NEG-15 with 10^{-5} subsampling	HS with 10^{-5} subsampling
Vasco de Gama	Lingsugur	Italian explorer
Lake Baikal	Great Rift Valley	Aral Sea
Alan Bean	Rebecca Naomi	moonwalker
Ionian Sea	Ruegen	Ionian Islands
chess master	chess grandmaster	Garry Kasparov

- HS with subsampling learns best representation of phrases.

Additive Compositionality

- Skip-gram 으로 learned 된 word, phrase representations는 analogical reasoning 수행할 수 있었다. (france - paris + italy = rome)
- Also possible to meaningfully combine words by element-wise addition of vectors. (주변 단어들을 predict하므로 vectors를 해당 단어가 등장하는 context들의 distribution 으로 볼 수 있다..)

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

- 4 closest tokens to sum of two vectors.

Closing

- Sentence2Vec, Paragraph2Vec
- Continuous Distributed Representation of Biological Sequences for Deep Proteomics and Genomics (<https://arxiv.org/pdf/1503.05140.pdf>)
- Gene2Vec (<https://nbviewer.ipython.org/github/davidcox143/Gene2vec/blob/master/report/Gene2vec.ipynb>)
 - Using nucleotide sequence : 27개 단위로 자름. non coding 부분(단백질을 생성하지 않는 이 많았다..) 결과가 크게 의미 없었다.
 - Using amino acid sequence
 - 244,140,625 (256) 6-gram amino acid sequences -> more detailed model.
 - disorder proteins..