CS224d

Data Science & Business Analytics Lab
KyoungHyun Mo

Intro

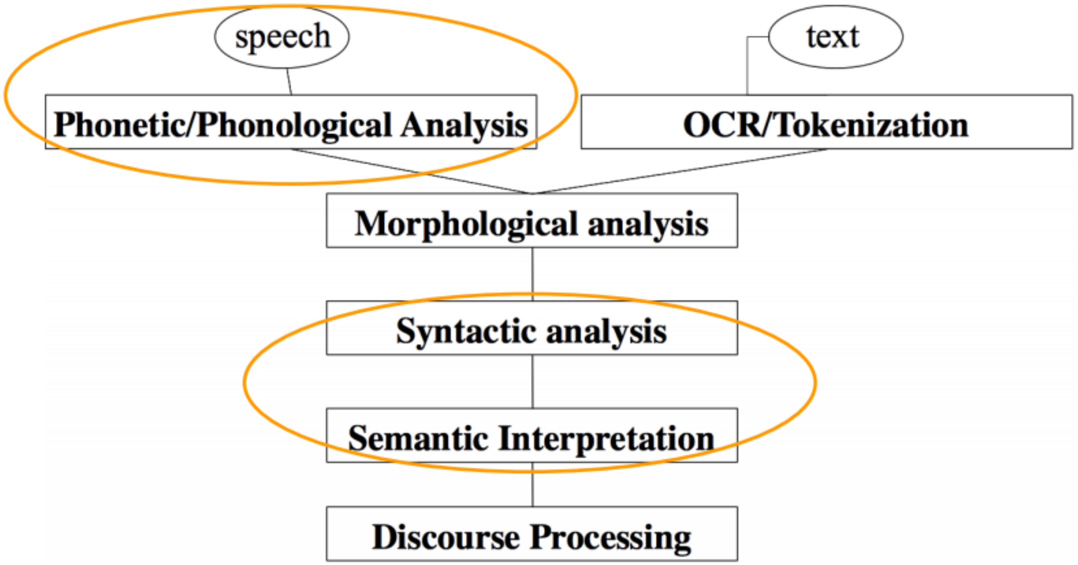
Word Vector

Intro

❖ NLP(Natural Language Processing)란 무엇인가?

- Computer science, artificial intelligence, linguistics의 intersaction에 있는 분야
- 목적 : 컴퓨터가 자연언어를 이해하고 처리하는 것
  e.g) Question Answering
- 언어를 완벽하게 이해하는 것이 궁극적인 목표이며 이를 AI-complete이라 함

❖ NLP Level

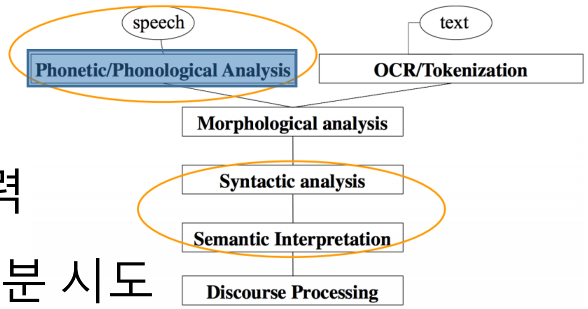


NLP Level

❖ NLP Level

➢ Phonetic/Phonological Analysis(음운 분석)

- 음성을 더 잘 이해하기 위한 목적
- 전통적인 방식은 음운 하나하나를 표현하려 노력
- 미세한 말소리, 소리의 장단, 고저, 강약, 억양 구분 시도
- 제작에 많은 시간이 필요



CONSONANTS (PULMONIC) © 2005 IPA

	Bilabial	Labiodental	Dental	Alveolar	Postalveolar	Retroflex	Palatal	Velar	Uvular	Pharyngeal	Glottal
Plosive	p b			t d		ʈ ɖ	c ɟ	k ɡ	q ɢ		ʔ
Nasal	m	ɱ		n		ɳ	ɲ	ŋ	ɴ		
Trill	ʙ			r					ʀ		
Tap or Flap		ⱱ		ɾ		ɽ					
Fricative	ɸ β	f v	θ ð	s z	ʃ ʒ	ʂ ʐ	ç ʝ	x ɣ	χ ʁ	ħ ʕ	h ɦ
Lateral fricative				ɬ ɮ							
Approximant		ʋ		ɹ		ɻ	j	ɰ			
Lateral approximant				l		ɭ	ʎ	ʟ			

Where symbols appear in pairs, the one to the right represents a voiced consonant. Shaded areas denote articulations judged impossible.

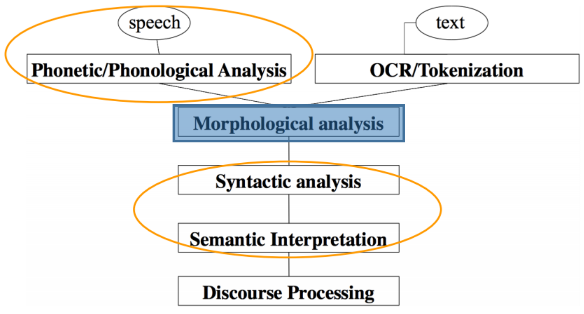
International Phonetic Association(국제 음성학회)에서 1888년에 지정한 Phonetic symbol(음성 기호)

❖ NLP Level

➢ Morphological Analysis(형태학적 분석)

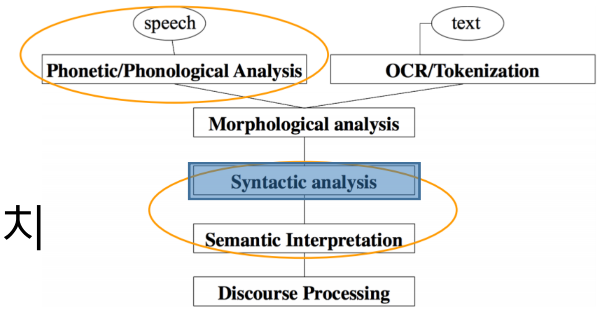
- 형태소를 분류하는 체계
- 접두사, 접미사, 어간 등 세분화

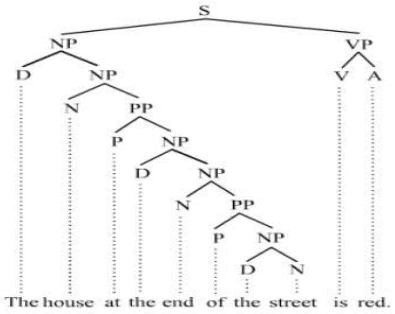
prefix	stem	suffix
un	interest	ed



❖ Syntactic Analysis(구문 분석)

- 문법과 유사한 개념
- 전체 문장의 의미를 형성하기 위해 단어들을 배치
- NP(Noun phrases), VP(Verb phrases)와 같은 개별적인 카테고리 구(phrases)를 이용





NLP Level

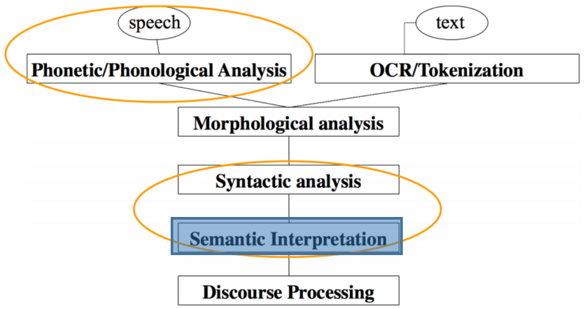
❖ NLP Level

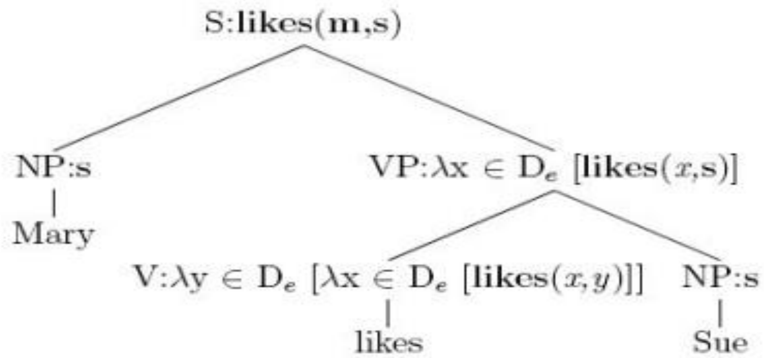
➢ Semantic Interpretation(의미 해석)

- Sentence의 의미

- Lambda Calculus

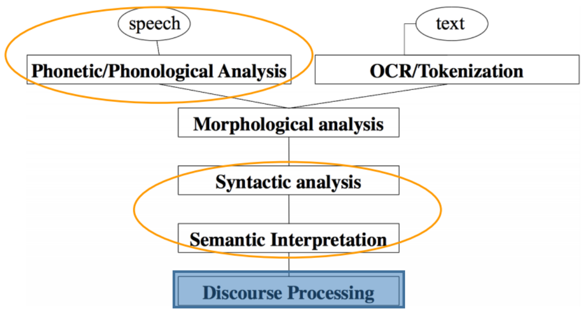
- Discrete representation





❖ Discourse Processing(담화 처리)

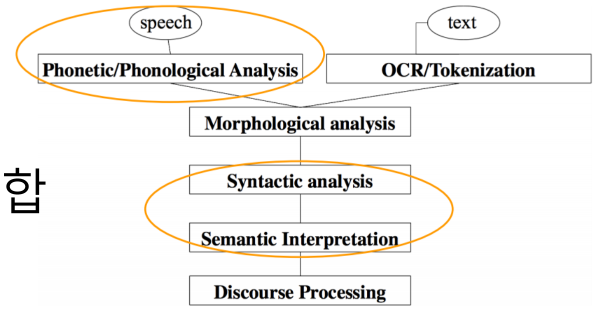
- Full 담화를 이해



❖ NLP Level

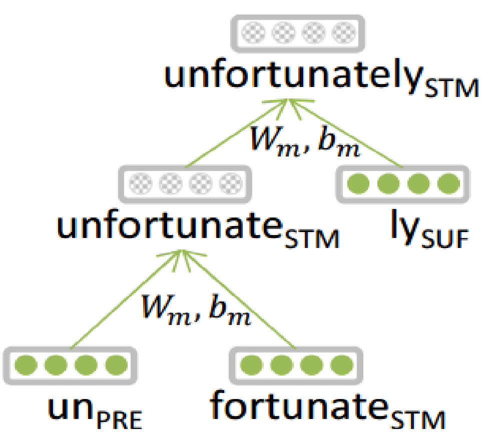
➢ DL(Deep Learning)

- 모든 단어, 구, logical expression을 벡터로 표현
- Neural network에서 두 벡터가 하나의 벡터로 결합

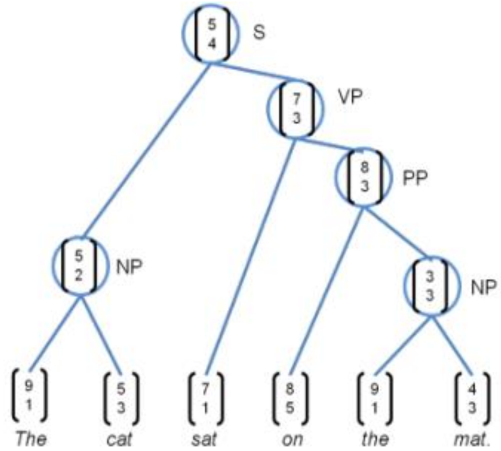


❖ Deep NLP(Deep Learning + NLP)

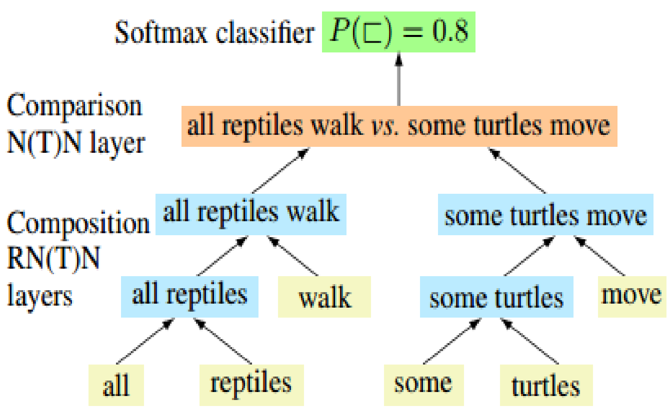
Morphological



Syntactic



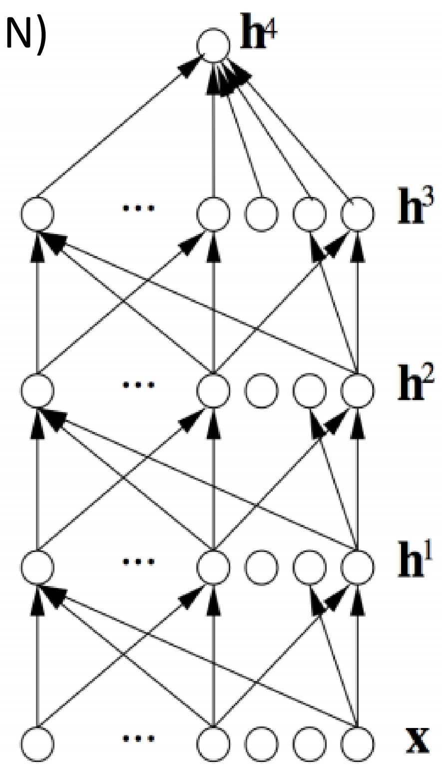
Semantic



NLP Level

❖ Deep Learning

- Deep Learning은 Machine Learning의 Subfield
- Good Feature, representation을 학습하는 Representation Learning
- “Raw”데이터를 input으로 받는다(e.g. pixel, character, image, word)
- Different kinds of neural networks(e.g. ANN, DNN, CNN, RNN)
- End-To-End model
- Learned Features are easy to adapt, fast to learn
- Deep learning은 유연하고 광범위한 Framework를 제공
- Supervised & unsupervised 학습이 가능
- A lot of data
- Faster machine and multicore CPU/GPU
- New models, algorithms, ideas



➔ Improved performance!!

NLP

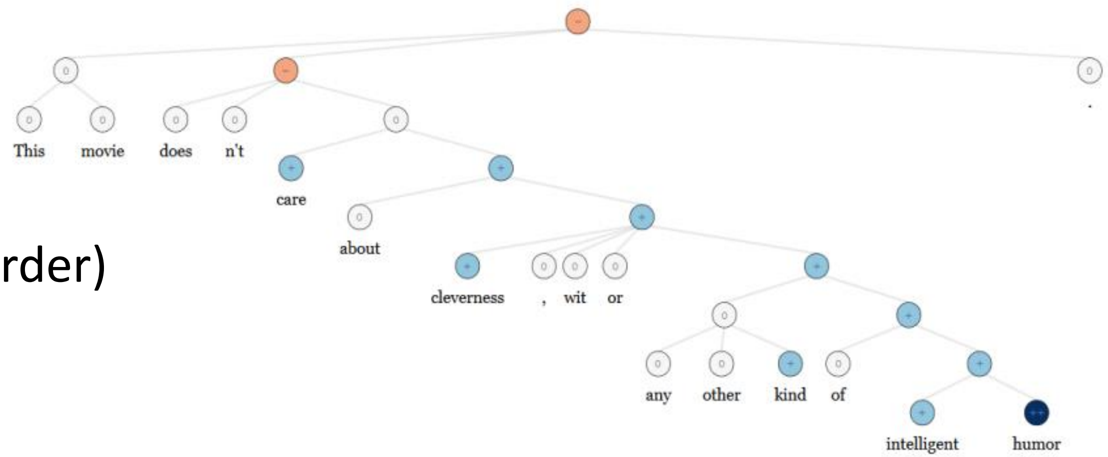
❖ Application

➢ Easy

- Spell Checking
- Keyword Search
- Finding Synonyms

➢ Sentiment Analysis

- Sentiment dictionaries를 만들고 bag-of-words representation 또는 hand-designed negation features와 결합

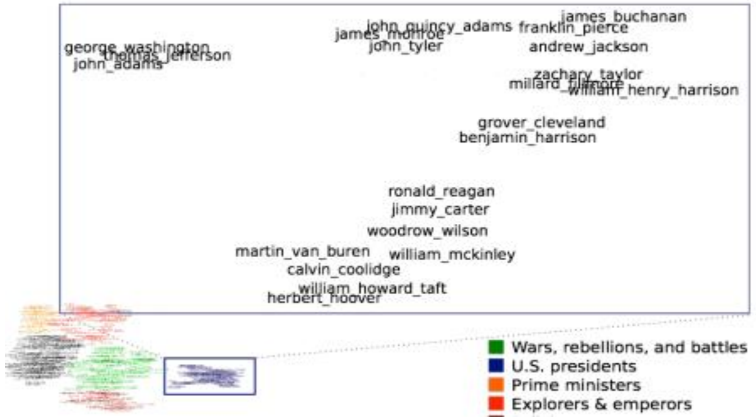


➢ Intermediate(Little harder)

- Extracting Information
- Classifying
- Positive/negative sentiment of longer documents

➢ Hard

- Machine Translation
- Spoken dialog Systems
- Complex Question Answering



NLP

❖ Application

➢ Machine Translation

- Many levels of translation

  1) One sentence to the other

  2) Understand the grammatical structure

  3) Understand meaning first and translate into another language

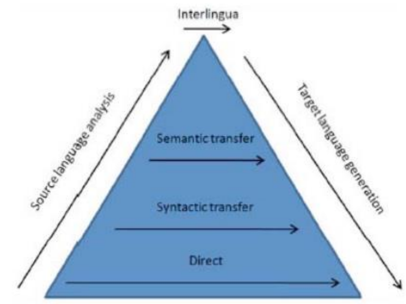
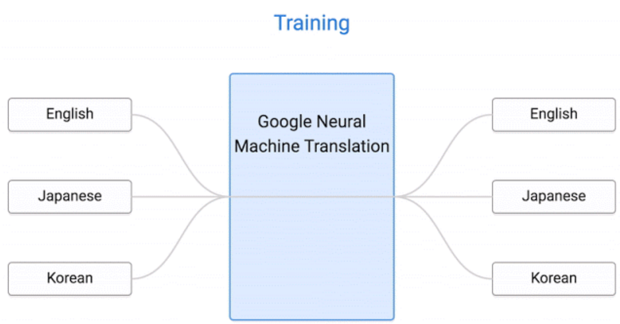


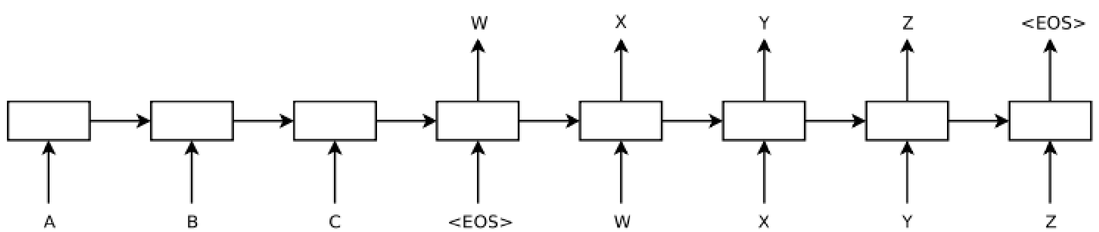
Figure 1: The Vauquois triangle

- Need a lot of RAM

- Use interlingua



- Sequence to Sequence Learning with Neural Networks(Sutskever et al. 2014)



NLP

❖ Why Hard?

➢ Complexity, Ambiguity

- 상황에 따라 의미가 달라짐

- 다양한 해석 여지가 존재

- 예외 사항이 많음

❖ Why Hard?

- Complexity, Ambiguity
  - 상황에 따라 의미가 달라짐
  - 다양한 해석 여지가 존재
  - 예외 사항이 많음

ex1) Jane hit June and then she [fell/ran].   Q. Who Is she??

❖ Why Hard?

➢ Complexity, Ambiguity

- 상황에 따라 의미가 달라짐
- 다양한 해석 여지가 존재
- 예외 사항이 많음

ex1) Jane hit June and then she [fell/ran]. Q. Who Is she??

=> Jane hit June and then she [fell/ran].
=> Jane hit June and then she [fell/ran].

❖ Why Hard?

- Complexity, Ambiguity
  - 상황에 따라 의미가 달라짐
  - 다양한 해석 여지가 존재
  - 예외 사항이 많음

ex1) Jane hit June and then she [fell/ran]. Q. Who Is she??

=> Jane hit June and then she [fell/ran].
=> Jane hit June and then she [fell/ran].

ex2) I made her duck.

=> 요리했다
=> 마법을 부렸다
=> 오리를 선물했다

Word Vector

❖ 컴퓨터가 의미를 이해하는 방식

➢ NLP의 목적

- 컴퓨터가 자연언어를 이해하는 것

- 언어를 완벽하게 이해하는 것을 AI-complete이라 함

➢ WordNet과 같은 Texonomy를 사용

- (is-a) relationship

- synonym sets(good)

➢ 문제점

- 신조어를 반영되기 어려움

- 인간의 노동력을 많이 필요

- 정확한 Similarity를 계산하기 어려움

S: (adj) full, good
S: (adj) estimable, good, honorable, respectable
S: (adj) beneficial, good
S: (adj) good, just, upright
S: (adj) adept, expert, good, practiced, proficient, skillful
S: (adj) dear, good, near
S: (adj) good, right, ripe
...
S: (adv) well, good
S: (adv) thoroughly, soundly, good
S: (n) good, goodness
S: (n) commodity, trade good, good

Word Vector

❖ 컴퓨터가 의미를 이해하는 방식

➢ “One-Hot” representation

- 단어 공간에서 많은 0과 single 1을 사용하는 방식

[0 0 0 0 0 0 0 0 0 0 1 0 0 0 0]

➢ 문제점

- 단어가 늘어날수록 메모리가 많이 필요

Dimensionality: 20K(speech) – 50K(PTB) – 500K(big vocab) – 13M(google 1T)

- 정확한 Similarity를 계산하기 어려움

motel [0 0 0 0 0 0 0 0 0 0 1 0 0 0 0] AND
hotel [0 0 0 0 0 0 0 1 0 0 0 0 0 0 0] = 0

Word Vector

❖ Similarity

➢ Distributional Similarity

- 단어의 주변 정보를 이용하여 표현하는 방식

- 현대 통계적 NLP에서 가장 성공적인 아이디어

government debt problems turning into	banking	crises as has happened in
saying that Europe needs unified	banking	regulation to replace the hodgepodge

↖Banking을 나타내는 단어들↗

➢ Window based Cooccurrence Matrix

- Window length 1(보통 5~10)

- Symmetric Matrix

Word Vector

❖ Similarity

- Window based Cooccurrence Matrix

- 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

Word Vector

❖ Similarity

➢ Window based Cooccurrence Matrix

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

➢ 문제점

- 단어가 늘어날수록 메모리가 많이 필요

- Matrix가 Sparse함

Word Vector

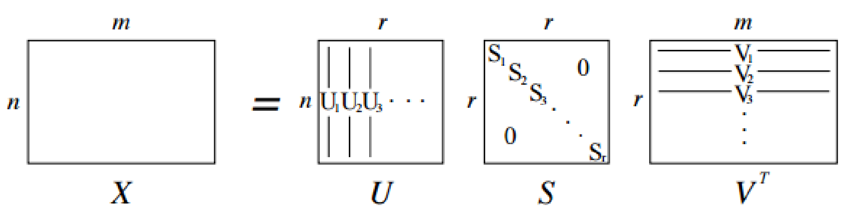
❖ Dimensional Reduction

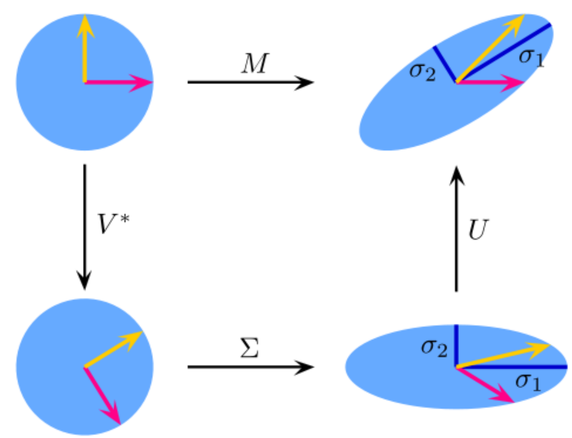
- 중요한 정보를 유지한 채 작은 차원으로 축소시키기
- 보통 25~1000 사이의 dimension을 사용

➢ SVD(Singular Value Decomposition)

- The singular value decomposition is a factorization of a real or complex matrix.
- 역행렬을 구하거나 선형식의 해를 구하는데 사용

$$M = U \cdot \Sigma \cdot V^*$$





Word Vector

- SVD(Singular Value Decomposition) $M = U \cdot \Sigma \cdot V^*$

$$M = \begin{bmatrix} 4 & 0 \\ 3 & -5 \end{bmatrix}, \quad M^T = \begin{bmatrix} 4 & 3 \\ 0 & -5 \end{bmatrix}, \quad M^T M = \begin{bmatrix} 16 & 12 \\ 12 & 34 \end{bmatrix}$$

$$M^T M - cI = \begin{bmatrix} 16-c & 12 \\ 12 & 34-c \end{bmatrix}, \quad |M^T M - cI| = 0$$

$$(16-c)(34-c) - (12)(12) = 0$$

$$(c^2 - 50c + 400) = 0$$

$$Eigenvalues = \ c_1, c_2 = 40, 10 \qquad Singularvalues = \ s_1, s_2 = \sqrt{40}, \sqrt{10} \qquad \Sigma = \begin{bmatrix} \sqrt{40} & 0 \\ 0 & \sqrt{10} \end{bmatrix}$$

$$if\ c_1 = 40, \qquad (M^T M - c_1 I)X_1 = \begin{bmatrix} 16-c_1 & 12 \\ 12 & 34-c_1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \qquad X_1 = \begin{bmatrix} 1 \\ 2 \end{bmatrix}$$

$$if\ c_2 = 10, \qquad (M^T M - c_1 I)X_2 = \begin{bmatrix} 16-c_2 & 12 \\ 12 & 34-c_2 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \qquad X_2 = \begin{bmatrix} -2 \\ 1 \end{bmatrix}$$

$$V = [X_1, X_2] = \begin{bmatrix} 1 & -2 \\ 2 & 1 \end{bmatrix} \qquad V^* = \begin{bmatrix} 1 & 2 \\ -2 & 1 \end{bmatrix} \qquad V^* V = \begin{bmatrix} 1 & 2 \\ -2 & 1 \end{bmatrix} \begin{bmatrix} 1 & -2 \\ 2 & 1 \end{bmatrix} = \begin{bmatrix} 5 & 0 \\ 0 & 5 \end{bmatrix}$$

$$M = U \cdot \Sigma \cdot V^* \quad \Rightarrow \quad M \cdot V = U \cdot \Sigma \cdot V^* \cdot V = 5 \cdot U \cdot \Sigma \quad \Rightarrow \quad \frac{1}{5} M V \Sigma^{-1} = U$$

$$U = \frac{1}{5} M V \Sigma^{-1} = \frac{1}{5} \begin{bmatrix} 4 & 0 \\ 3 & -5 \end{bmatrix} \begin{bmatrix} 1 & -2 \\ 2 & 1 \end{bmatrix} \begin{bmatrix} 0.1581 & 0 \\ 0 & 0.3162 \end{bmatrix} = \begin{bmatrix} 0.1264 & -0.5059 \\ -0.2213 & -0.6957 \end{bmatrix}$$

Word Vector

- SVD(Singular Value Decomposition)

$$M = \begin{bmatrix} 4 & 0 \\ 3 & -5 \end{bmatrix}, \quad M^T M = \begin{bmatrix} 16 & 12 \\ 12 & 34 \end{bmatrix}$$

$$V = [X_1, X_2] = \begin{bmatrix} 1 & -2 \\ 2 & 1 \end{bmatrix}$$

$$\Sigma = \begin{bmatrix} \sqrt{40} & 0 \\ 0 & \sqrt{10} \end{bmatrix} \qquad \Sigma^{-1} = \begin{bmatrix} 0.1581 & 0 \\ 0 & 0.3162 \end{bmatrix}$$

$$U = \begin{bmatrix} 0.1264 & -0.5059 \\ -0.2213 & -0.6957 \end{bmatrix}$$

```r
library(MASS)

m <- matrix(c(4,0,3,-5),nrow=2,byrow=T)
m%*%t(m)
v <- matrix(c(1,-2,2,1),nrow=2,byrow=T)
s <- matrix(c(sqrt(40),0,0,sqrt(10)),nrow=2,byrow=T)
ginv(s)
u <- (1/5)*m%*%v%*%ginv(s)

round(u%*%s%*%t(v))==m
```

```
> m
     [,1] [,2]
[1,]    4    0
[2,]    3   -5
> m%*%t(m)
     [,1] [,2]
[1,]   16   12
[2,]   12   34
> v
     [,1] [,2]
[1,]    1   -2
[2,]    2    1
> s
         [,1]     [,2]
[1,] 6.324555 0.000000
[2,] 0.000000 3.162278
> ginv(s)
          [,1]      [,2]
[1,] 0.1581139 0.0000000
[2,] 0.0000000 0.3162278
> u
           [,1]       [,2]
[1,]  0.1264911 -0.5059644
[2,] -0.2213594 -0.6957011
> round(u%*%s%*%t(v))
     [,1] [,2]
[1,]    4    0
[2,]    3   -5
> round(u%*%s%*%t(v))==m
     [,1] [,2]
[1,] TRUE TRUE
[2,] TRUE TRUE
```

Word Vector

❖ Dimensional Reduction

➢ SVD(Singular Value Decomposition)

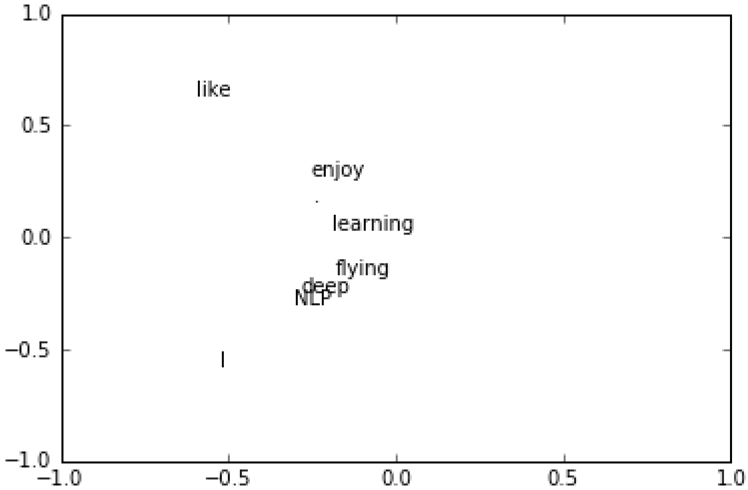
- I like deep learning.
- I like NLP.
- I enjoy flying.

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

```python
import numpy as np
la = np.linalg
words = ["I","like","enjoy","deep","learning","NLP","flying","."]
x = np.array([[0,2,1,0,0,0,0,0],
              [2,0,0,1,0,1,0,0],
              [1,0,0,0,0,0,1,0],
              [0,1,0,0,1,0,0,0],
              [0,0,0,1,0,0,0,1],
              [0,1,0,0,0,0,0,1],
              [0,0,1,0,0,0,0,1],
              [0,0,0,0,1,1,1,0]])

u,s,v = la.svd(x, full_matrices=False)

for i in range(len(words)):
    plt.axis([-1, 1, -1, 1])
    plt.text(u[i,0],u[i,1],words[i])
```



NLP :

```
array([-0.18248984, -0.16102777, -0.39784243, -0.38322849, -0.51292322,
       -0.42757442,  0.4191856 , -0.11831383])
```

Word Vector

❖ Dimensional Reduction

➢ Cooccurrence Matrix

- 단어의 출현 빈도에 따라 지나치거나 부족한 영향력을 가짐
- 단어의 최소 빈도수를 기준으로 삼고 나머지는 무시
- Window를 늘리는 방법
- 단어의 출현 빈도 대신 Correlation을 사용

➢ SVD 문제점

- 계산 복잡도가 높다
- N x M Matrix의 경우 O(mn^2)
- 새로운 문서나 단어가 추가되면 새로운 SVD를 만들어야 함

Word Vector

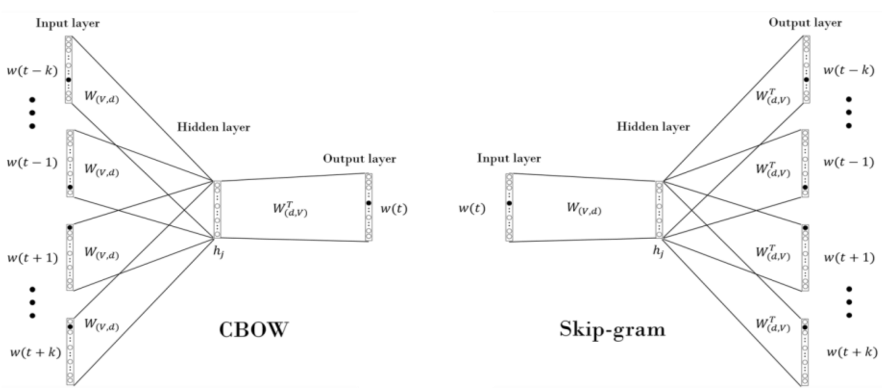
❖ Directly learn low-dimensional word vectors

➢ 관련연구

- Learning representations by back-propagating error(Rumelhart et al., 1986)
- A neural probabilistic language model(Bengio et al., 2003)
- NLP (almost) from Scratch(Collobert & Weston, 2008)
- A recent, even simpler and faster model:word2vec(Mikolov et al. 2013)

➢ Word2Vec

- Cooccurrence counts를 사용하는 대신 모든 단어의 주변 단어를 예측
- 새로운 단어나 문장을 적용하기 쉬움



Word Vector

❖ Word2Vec

- 모든 단어로부터 m만큼의 주위에 있는 단어들로(을) 예측

- Objective Function :
  현재 주어진 단어와 주변 단어의 Maximum log probability

$$J(\theta) = \frac{1}{T}\sum_{t=1}^{T}\sum_{-m\leq j\leq m\, j\neq 0} logp(w_{t+j}|w_t)$$

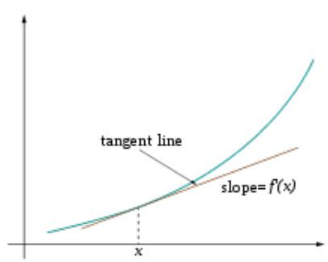
$$p(o|c) = \frac{\exp(u_o^T v_c)}{\sum_{w=1}^{W}\exp(u_w^T v_c)}$$

- o : 주변 단어
- c : 중심 단어
- u, v : word vector

- Objective function을 최적화하기 위해서 Gradient Descent와 Chain Rule을 사용

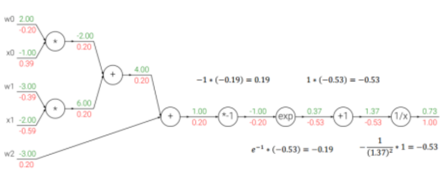
- Gradient Descent

$$\frac{\partial x^T a}{\partial x} = \frac{\partial a^T x}{\partial x} = a$$



- Chain Rule

$$\frac{dy}{dx} = \frac{dy}{du}\frac{du}{dx}$$

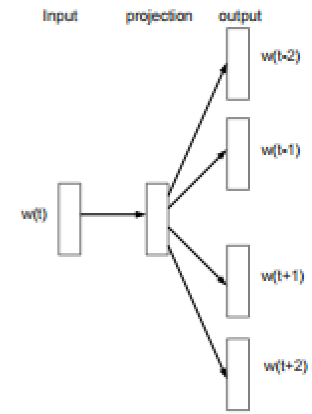


Word Vector

❖ Word2Vec

$$J(\theta) = \frac{1}{T}\sum_{t=1}^{T}\sum_{-m\le j\le m\, j\neq 0} logp(w_{t+j}|w_t)$$

$$p(o|c) = \frac{\exp(u_o^T v_c)}{\sum_{w=1}^{W}\exp(u_w^T v_c)}$$



t : current vector,   m : window size,   w : 전체 word수 v_c: w에서 current vector, u_w^T : w의 transpose행렬에서 vector

$$[0\ 0\ 0\ 1\ \cdots 0\ 0\ 0] \times \begin{bmatrix} & \cdots & \\ & v_c & \\ \vdots & \ddots & \vdots \\ & \cdots & \end{bmatrix} = [a_1 a_2 a_3 \cdots a_h] \times \begin{bmatrix} & \cdots & \\ v_w & \ddots & \vdots \\ & \cdots & \end{bmatrix} = [w_1 w_2 w_3 \cdots w_W]$$

(1 X W) (W X h) (1 X h) (h X W) (1 X W)

Word Vector

- Word2Vec objective function gradient

$$J(\theta) = \frac{1}{T}\sum_{t=1}^{T}\sum_{-m\le j\le m j\neq 0} logp(w_{t+j}|w_t) \qquad p(o|c) = \frac{\exp(u_o^T v_c)}{\sum_{w=1}^{W}\exp(u_w^T v_c)}$$

$$\frac{\partial}{\partial v_c}\log p(o|c) = \frac{\partial}{\partial v_c}\log\frac{\exp(u_o^T v_c)}{\sum_{w=1}^{W}\exp(u_w^T v_c)}$$

$$\Rightarrow \frac{\partial}{\partial v_c}\left(\log\exp(u_o^T v_c) - \log\sum_{w=1}^{W}\exp(u_w^T v_c)\right) \Rightarrow \overset{1}{\boxed{\frac{\partial}{\partial v_c}\log\exp(u_o^T v_c)}} - \overset{2}{\boxed{\frac{\partial}{\partial v_c}\log\sum_{w=1}^{W}\exp(u_w^T v_c)}}$$

1 $$\frac{\partial}{\partial v_c}\log\exp(u_o^T v_c) = \frac{\partial}{\partial v_c}u_o^T v_c = u_0$$

2 $$-\frac{\partial}{\partial v_c}\underbrace{\boxed{\log}}_{F}\underbrace{\boxed{\sum_{w=1}^{W}\exp(u_w^T v_c)}}_{Z = g(V_c)} = \frac{\partial F}{\partial z}\frac{\partial z}{\partial v_c} = \frac{\partial F}{\partial z}\frac{\partial g(v_c)}{\partial v_c}$$

F $$= \frac{\partial\log(z)}{\partial z} = \frac{1}{z} = \frac{1}{\sum_{w=1}^{W}\exp(u_w^T v_c)}$$

$Z = g(V_c)$ $$= \frac{\partial g(v_c)}{\partial v_c} = \frac{\partial g(v_c)}{\partial v_c}\sum_{w=1}^{W}\exp(u_w^T v_c)$$

$$= \sum_{w=1}^{W}\frac{\partial g(v_c)}{\partial v_c}\exp(u_w^T v_c) = \sum_{w=1}^{W}\exp(u_w^T v_c)\cdot u_w$$

Distributed representations of words and Phrases and their Compositionality(Mikolov et al. 2013)

Word Vector

- Word2Vec objective function gradient

$$\frac{\partial}{\partial v_c}\log p(o|c) = u_o - \frac{1}{\sum_{w=1}^{W}\exp(u_w^T v_c)} \cdot \sum_{w=1}^{W}\exp(u_w^T v_c) \cdot u_w$$

$$= u_o - \frac{1}{\sum_{w=1}^{W}\exp(u_w^T v_c)} \cdot \sum_{w=1}^{W}\frac{\exp(u_w^T v_c)}{\sum_{w=1}^{W}\exp(u_w^T v_c)} \cdot u_w$$

$$= u_o - \sum_{w=1}^{W} P(w|c) \cdot u_w$$

- Word2Vec

- Similarity의 dimension을 encoding하였을 시(저차원 공간으로 임베딩 하였을 시) 매우 좋은 성능을 보임
  - $X_{apple} - X_{apples} \approx X_{car} - X_{cars} \approx X_{family} - X_{families}$
  - $X_{king} - X_{man} \approx X_{queen} - X_{woman}$
- Objective function이 scalable 하지 않음
- 계산 cost가 높음
- 문맥상 잘 사용되지 않는 단어들을 제외하여 negative prediction을 수행
- positive correlation 관계에 있는 단어들에 집중

Word Vector

❖ Count based vs Direct prediction

LSA, HAL (Lund & Burgess) COALS (Rohde et al) Hellinger-PCA (Lebret & Collobert)	NNLM, HLBL, RNN, Skip-gram/CBOW, (Bengio et al; Collobert & Weston; Huang et al; Mnih & Hinton; Mikolov et al; Mnih & Kavukcuoglu)
- 빠른 학습 - 통계적으로 효율적 - Word간 유사도 추정에 비효율적 - Count가 불균형한 데이터에 민감	- Corpus size에 민감 - 통계적으로 비효율적 - 일반적으로 성능이 좋음 - Word 유사도 사이의 복잡한 패턴을 잘 잡음

감사합니다