Text Mining & Word Embedding

22.08.30 / DSL 7기 최명헌

0. 목차

1. Text Mining

- What is Text Mining?
- Crawling
- Cleansing & Tokenization

2. Word Embedding

- word2vec
- CBoW
- Skip-Gram
- Hierarchical Softmax
- Negative Sampling
- Others...

3. Visualization

- PCA
- t-SNE

1. Text Mining

What is Text Mining?

Text + Mining = 글자를 채굴하는 것

글자를 가져오는 것? = Crawling?

BeautifulSoup

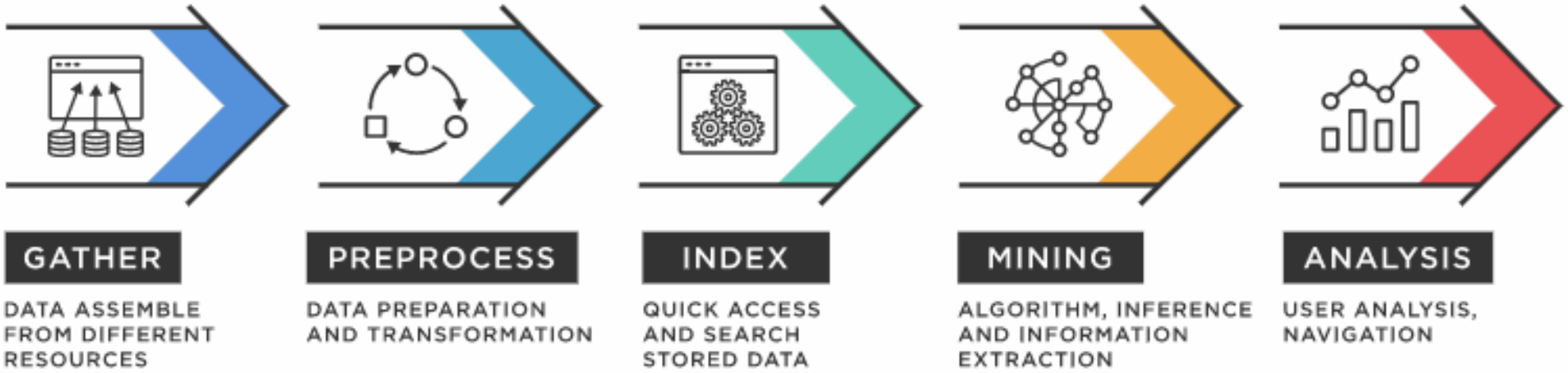
Selenium

1. Text Mining

What is Text Mining?

Text mining **is the process of** ***deriving high-quality information*** **from text.**

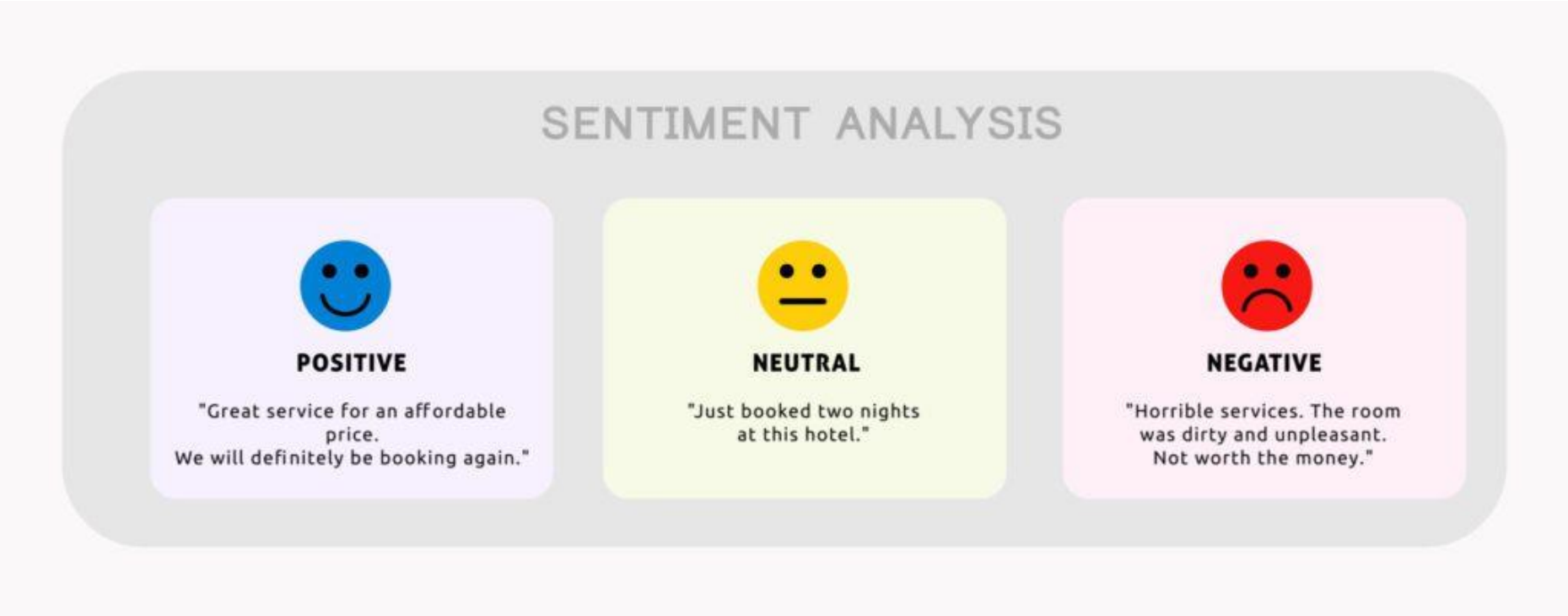
- Wikipedia



1. Text Mining

What is Text Mining?

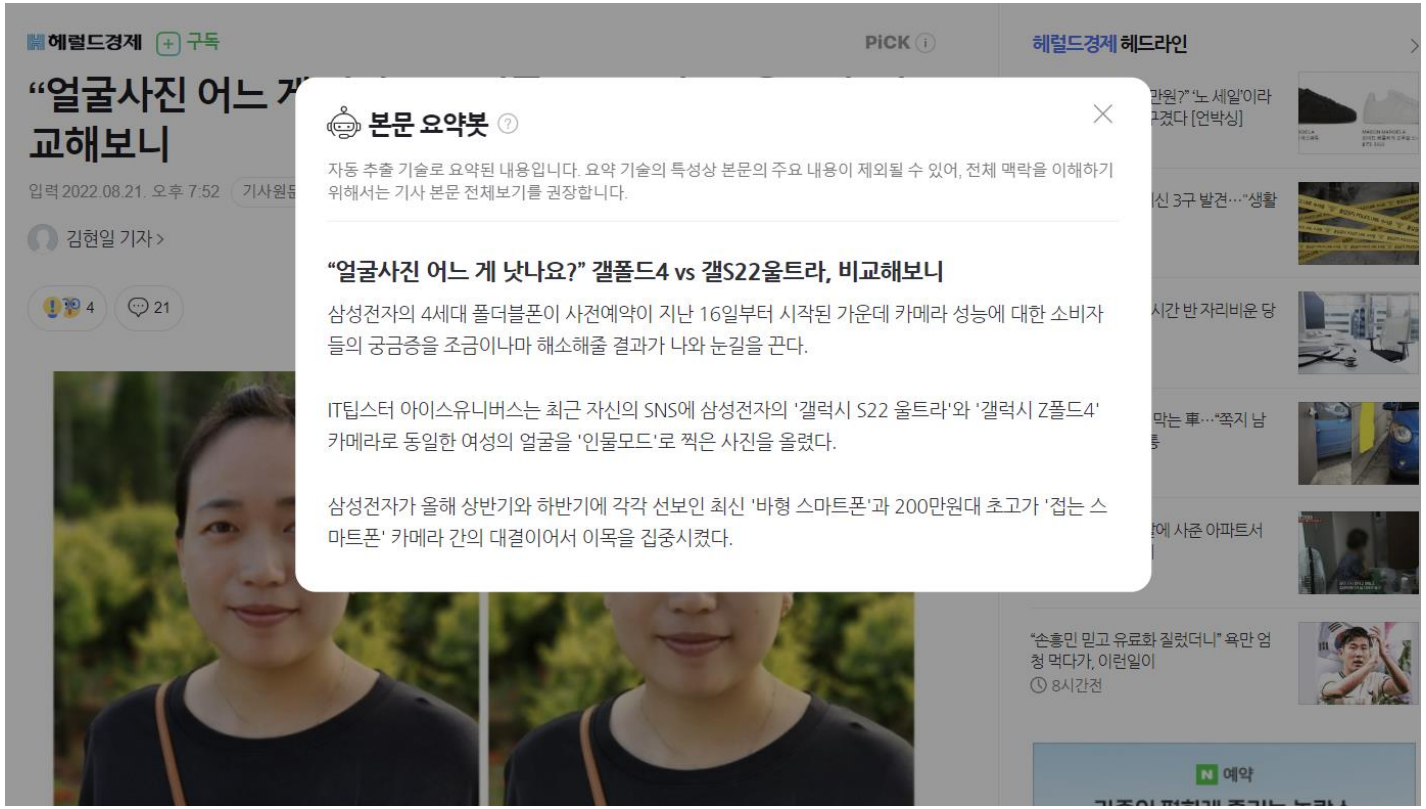
High Quality Information 1 : Sentiment Analysis



1. Text Mining

What is Text Mining?

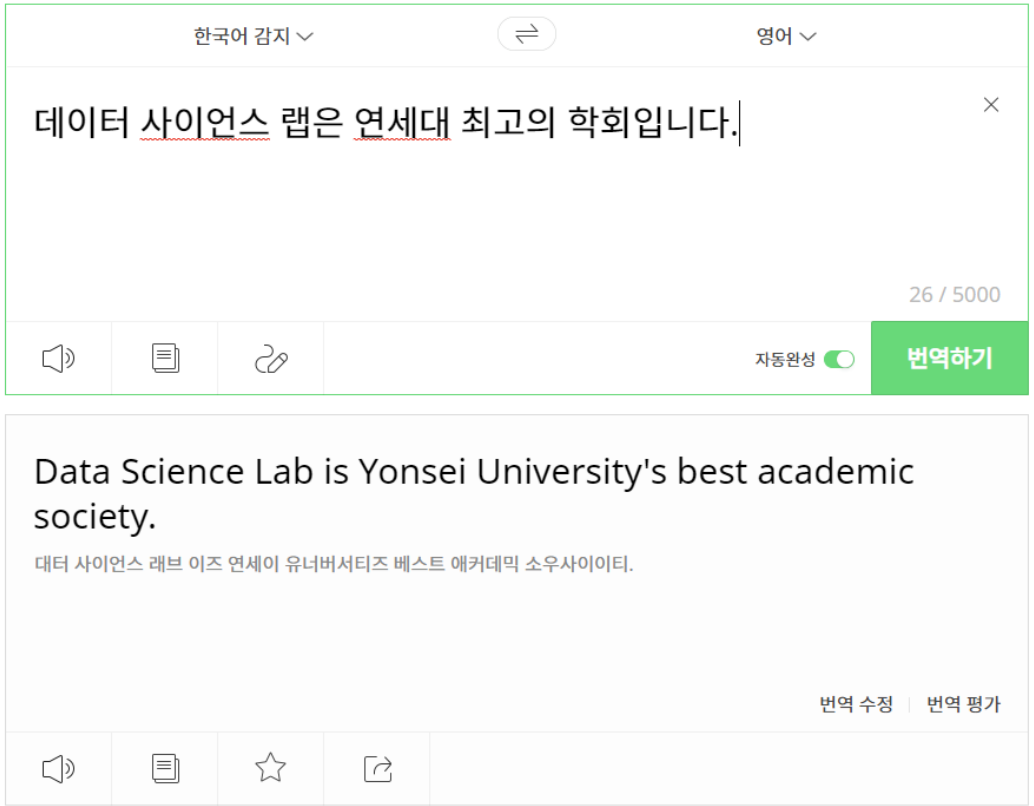
High Quality Information 2 : Topic Analysis



1. Text Mining

What is Text Mining?

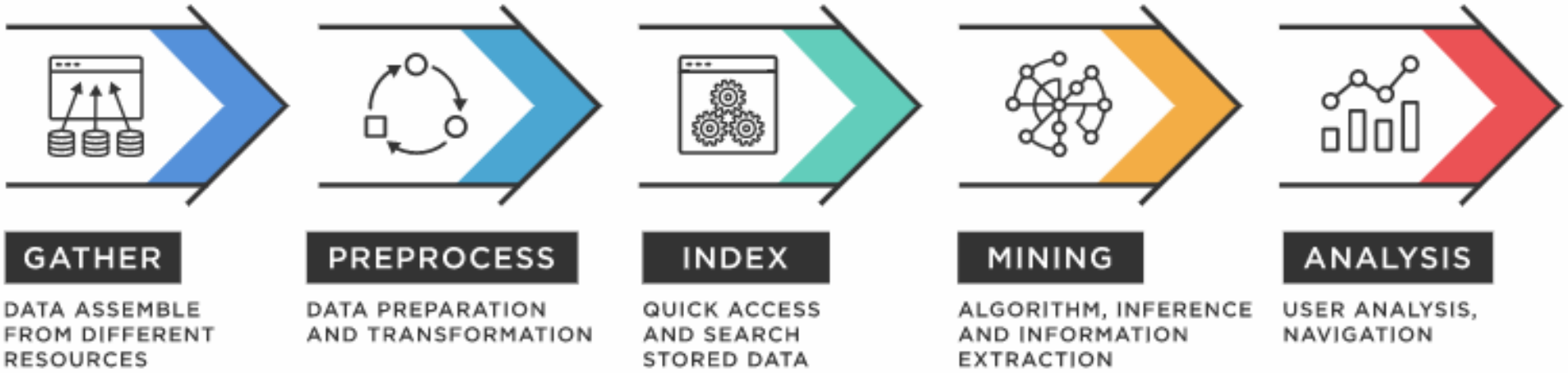
High Quality Informations...



1. Text Mining

What is Text Mining?

텍스트 마이닝이란 어떤 글 속에서 **중요한 정보**를 뽑아내고 그 정보를 이용하여 **다양한 목적**으로 사용하는 것



1. Text Mining

Crawling

1. Text Mining

Cleansing & Tokenization

Tokenization : 주어진 코퍼스(corpus)에서 토큰(token)이라 불리는 단위로 나눈 작업

Cleansing : 갖고 있는 코퍼스(corpus)로부터 노이즈 데이터(오타, 특수문자, stop words 등)를 제거하는 것

Normalization : 표현 방법이 다른 단어들을 통합시켜서 같은 단어로 만들어 주는 것

1. Text Mining

Cleansing & Tokenization

ex) 데이터-사이언스-랩은 “텍스트 마이닝”에 대해서도 공부할 수 있는 좋은 학회입니다.

Tokenization : 주어진 코퍼스(corpus)에서 토큰(token)이라 불리는 단위로 나눈 작업

=> 형태소 tokenizer : 데이터 / 사이언스 / 랩 / 은 / 텍스트 / 마 / 이닝 / 에 / 대해 / 서도 / 공부 / 할 / 수 / 있는 / 좋은 / 학회 / 입니다 / .

=> 음절 단위 tokenizer : 데 / 이 / 터 / 사 / … / 학 / 회 / 입 / 니 / 다 / .

Cleansing : 갖고 있는 코퍼스(corpus)로부터 노이즈 데이터(오타, 특수문자, stop words 등)를 제거하는 것

=> 데이터 사이언스 랩은 텍스트 마이닝에 대해서도 공부할 수 있는 좋은 학회입니다

Normalization : 표현 방법이 다른 단어들을 통합시켜서 같은 단어로 만들어 주는 것

=> 할, 하고, 하는, 함 - 하다 / 좋은, 좋다, 좋 - 좋다 / 입니다, 니다 - 이다 / ㅋㅋㅋㅋㅋㅋ, ㅋㅋㅋㅋㅋㅋㅋㅋㅋㅋ - ㅋㅋㅋ

Korean Tokenizer -konlpy

1. Kkma
   ['데이터', '사이언스', '랩', '은', '텍스트', '마이닝', '에', '대하', '어서', '도', '공부', '하', 'ㄹ', '수', '있', '는', '좋', '은', '학회', '이', 'ㅂ니다', '.']

2. Komoran
   ['데이터', '사이언스', '랩', '은', '텍스트', '마', '이닝', '에', '대하', '아서', '도', '공부', '하', 'ㄹ', '수', '있', '는', '좋', '은', '학회', '이', 'ㅂ니다', '.']

3. Hannanum
   ['데이터', '사이언스', '랩', '은', '텍스트', '마이닝', '에', '대', '어', '하', '어서', '도', '공부', '하', 'ㄹ', '수', '있', '는', '좋', '은', '학회', '이', 'ㅂ니다', '.']

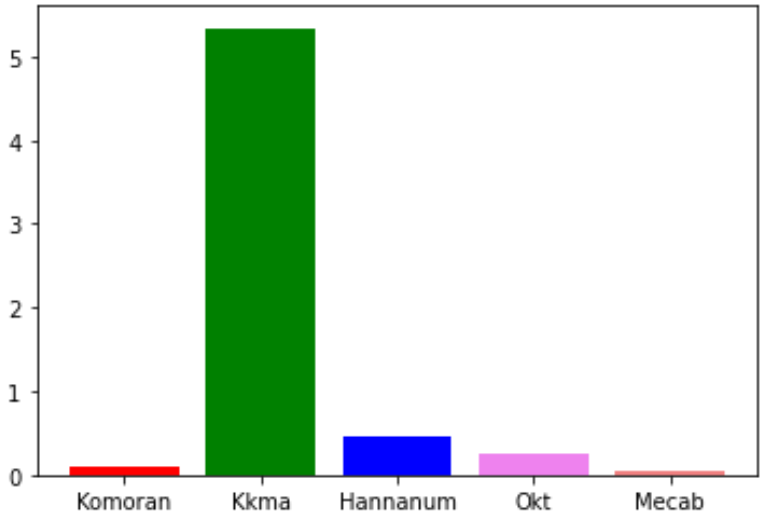
4. Okt
   ['데이터', '사이언스', '랩', '은', '텍스트', '마', '이닝', '에', '대해', '서도', '공부', '할', '수', '있는', '좋은', '학회', '입니다', '.']

5. Mecab
   ['데이터', '사이언스', '랩', '은', '텍스트', '마이닝', '에', '대해서', '도', '공부', '할', '수', '있', '는', '좋', '은', '학회', '입니다', '.']

1. Text Mining

Korean Tokenizer

형태소 분리 시 소요 시간



Okt 만의 특별한 기능

ex) 데싸랩 진짜 좋은 것 같앜ㅋㅋㅋㅋㅋㅋㅋㅋ!

1. Stem : 단어의 어간을 추출해주는 기능
2. Norm : 문장을 정규화 해주는 기능

- Stem = False, Norm = False

  ('데', 'Noun'), ('싸', 'Verb'), ('랩', 'Noun'), ('진짜', 'Noun'), ('좋은', 'Adjective'), ('것', 'Noun'), ('같앜', 'Noun'), ('ㅋㅋㅋㅋㅋㅋㅋㅋ', 'KoreanParticle'), ('!', 'Punctuation')

- Stem = True, Norm = False

  ('데', 'Noun'), ('싸다', 'Verb'), ('랩', 'Noun'), ('진짜', 'Noun'), ('좋다', 'Adjective'), ('것', 'Noun'), ('같앜', 'Noun'), ('ㅋㅋㅋㅋㅋㅋㅋㅋ', 'KoreanParticle'), ('!', 'Punctuation')

- Stem = False, Norm = True

  ('데', 'Noun'), ('싸', 'Verb'), ('랩', 'Noun'), ('진짜', 'Noun'), ('좋은', 'Adjective'), ('것', 'Noun'), ('같아', 'Adjective'), ('ㅋㅋㅋ', 'KoreanParticle'), ('!', 'Punctuation')

- Stem = True, Norm = True

  ('데', 'Noun'), ('싸다', 'Verb'), ('랩', 'Noun'), ('진짜', 'Noun'), ('좋다', 'Adjective'), ('것', 'Noun'), ('같다', 'Adjective'), ('ㅋㅋㅋ', 'KoreanParticle'), ('!', 'Punctuation')

1. Text Mining

Cleansing : 갖고 있는 코퍼스(corpus)로부터 노이즈 데이터(오타, 특수문자, stop words 등)를 제거하는 것

오타 제거

특수문자 제거

의미 없는 단어 (stop words) 제거 ex. 은, 는, 이, 가, 다, 이다, ㅋㅋㅋㅋ …

! or ? : 특수문자네? 무조건 제거해야지!
ㅋㅋㅋㅋㅋㅋ : 제거해야 할까?

하고자 하는 task에 맞게끔..!

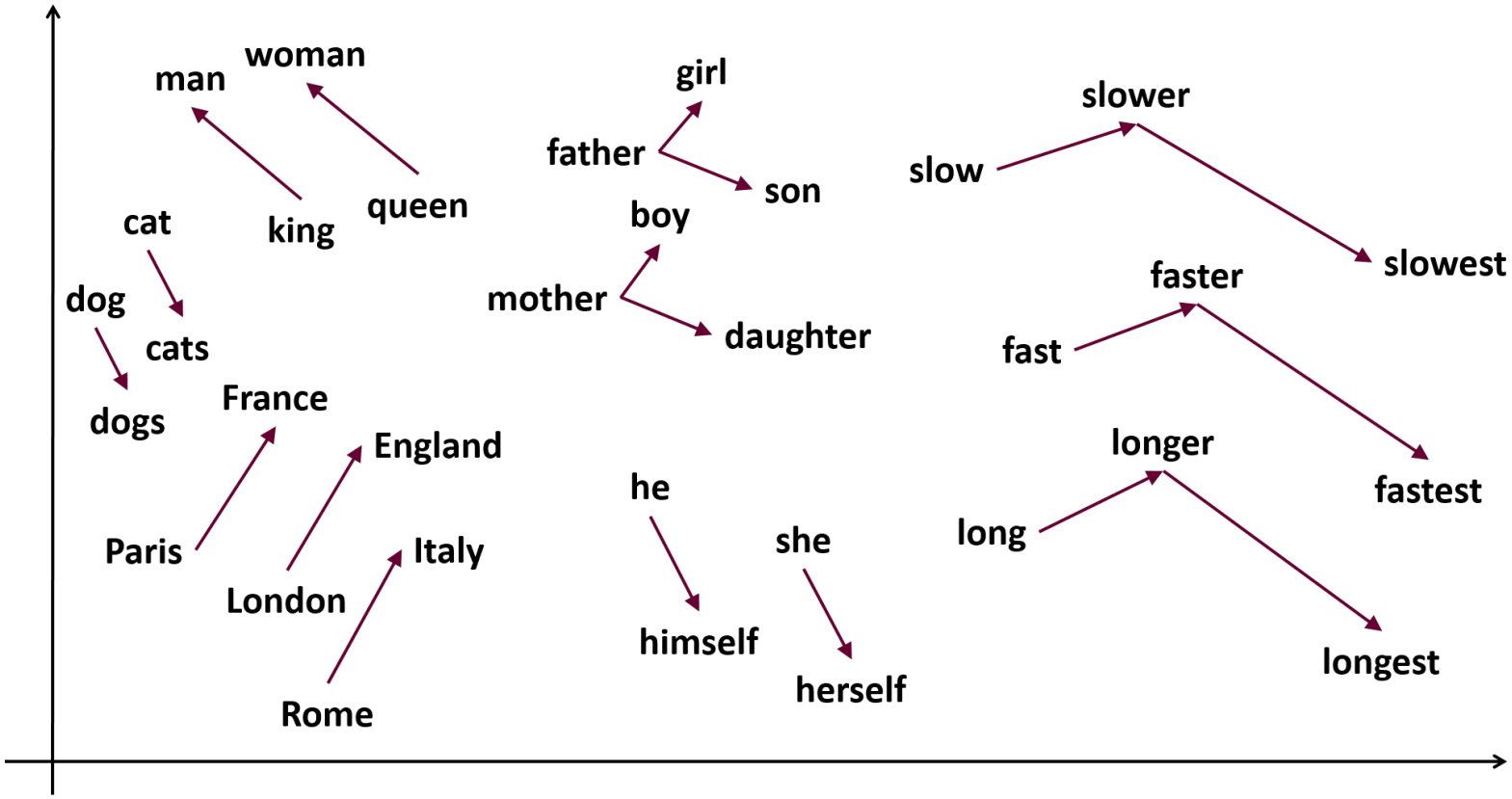
Tokenization & Cleansing

정답은 없습니다! **하고자 하는 task에 맞게끔..!**

다양한 시도와 실험 속에서 유의미한 정보를 찾는 것이 중요합니다!

2. Word Embedding

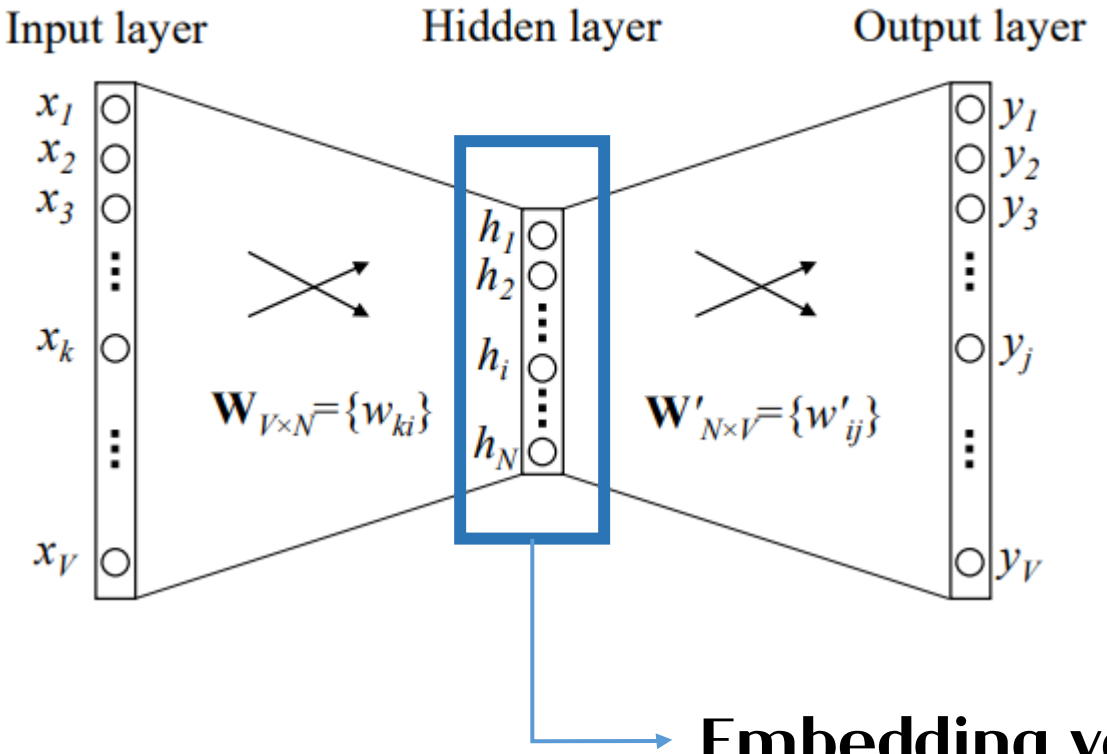
word2vec : 단어 벡터 간 **유의미한 유사도**를 반영하도록 단어의 의미를 **수치화**하는 것



https://word2vec.kr/search/

2. Word Embedding

word2vec : 단어 벡터 간 **유의미한 유사도**를 반영하도록 단어의 의미를 **수치화**하는 것



1. Input Data : One Hot Encoded Vector
2. Input -> Hidden : Embedding Matrix
3. Hidden -> Output : Score Matrix
4. Output -> Softmax function

2. Word Embedding

Corpus Tokenization & One Hot Encoding

Tokenization

우리집 강아지는 복슬 강아지 -> 우리 / 집 / 강아지 / 는 / 복슬 / 강아지

One Hot Encoding

우리 : [1, 0, 0, 0, 0] **집 : [0, 1, 0, 0, 0]**

강아지 : [0, 0, 1, 0, 0] **는 : [0, 0, 0, 1, 0]**

복슬 : [0, 0, 0, 0, 1]

2. Word Embedding

word2vec - CBoW vs SkipGram

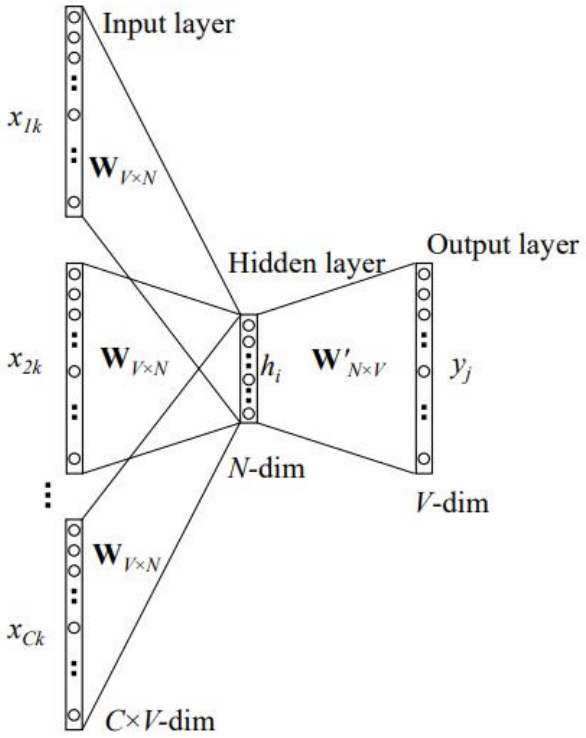


Figure 2: Continuous bag-of-word model

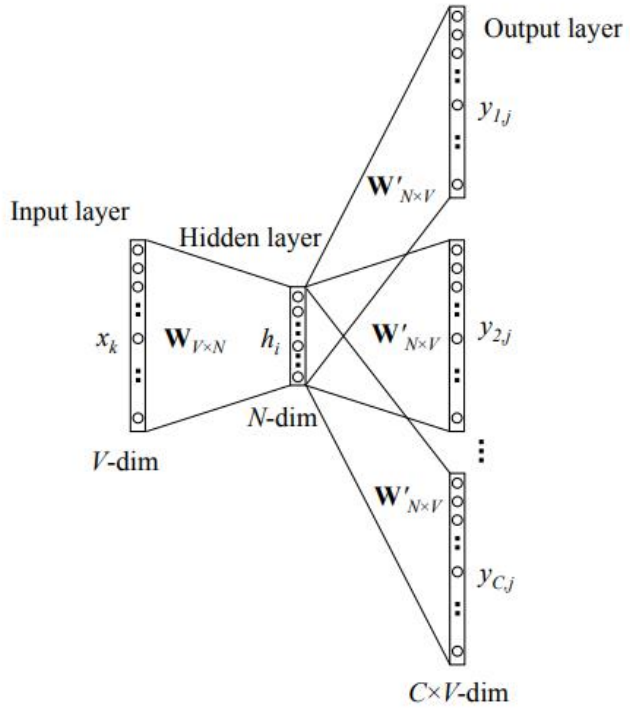


Figure 3: The skip-gram model.

우리 집 ___는 복슬 강아지

__ _ 강아지_ __ ___

2. Word Embedding

word2vec – CBoW vs SkipGram

CBOW

Input	Output
boy, is, going	the
the, is, going, to	boy
the, boy, going, to school	is
the, boy, is, to, school	going
boy, is, going, school	to
is, going, to	school

the	1
boy	1
is	1
going	1
to	1
school	1

Skip-Gram

Input	Output
the	boy, is, going
boy	the, is, going, to
is	the, boy, going, to school
going	the, boy, is, to, school
to	boy, is, going, school
school	is, going, to

the	3
boy	4
is	5
going	5
to	5
school	3

SkipGram이 CBoW보다 같은 epoch으로 학습을 한다해도

각 단어들은 여러 번 여러 context에 걸쳐 빈번하게 학습됨.

성능 : SkipGram > CBoW

2. Word Embedding

word2vec – SkipGram 학습과정 살펴보기

Hyperparameter !

Hidden layer의 차원 (N) : 몇 차원의 벡터로 단어를 표현할 것인가

Window Size (m) : 주변 몇 개의 단어를 사용하여 context를 파악할 것인가

Goal !

Maximize $P(w_{c-m}, \dots, w_{c+m}|w_c)$

=> Loss function = $-\log P(w_{c-m}, \dots, w_{c+m}|w_c)$

Example

우리 집 강아지는 복슬 강아지 (__ _ 강아지_ __ ___)

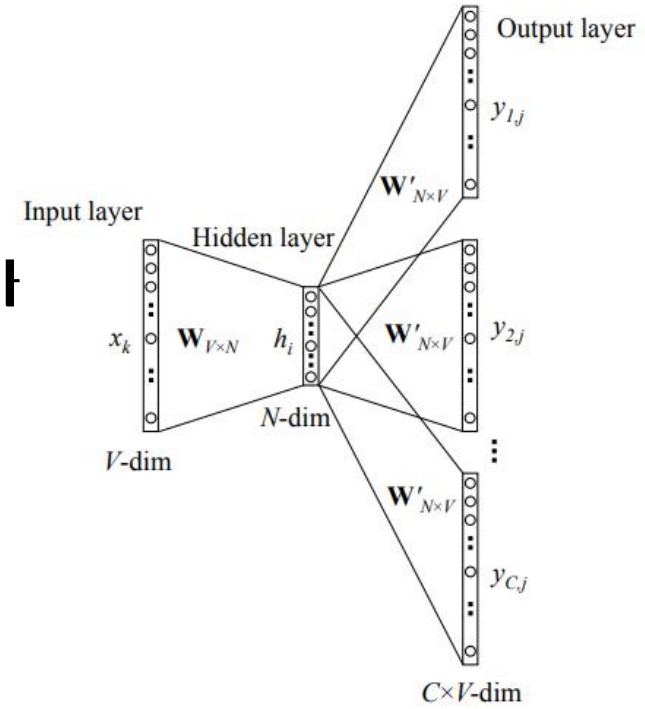


Figure 3: The skip-gram model.

2. Word Embedding

word2vec - SkipGram 학습과정 살펴보기 (Input Layer)

Tokenization

우리집 강아지는 복슬 강아지

=> 우리 / 집 / 강아지 / 는 / 복슬 / 강아지

One Hot Encoding

우리 : [1, 0, 0, 0, 0]

집 : [0, 1, 0, 0, 0]

강아지 : [0, 0, 1, 0, 0]

는 : [0, 0, 0, 1, 0]

복슬 : [0, 0, 0, 0, 1]

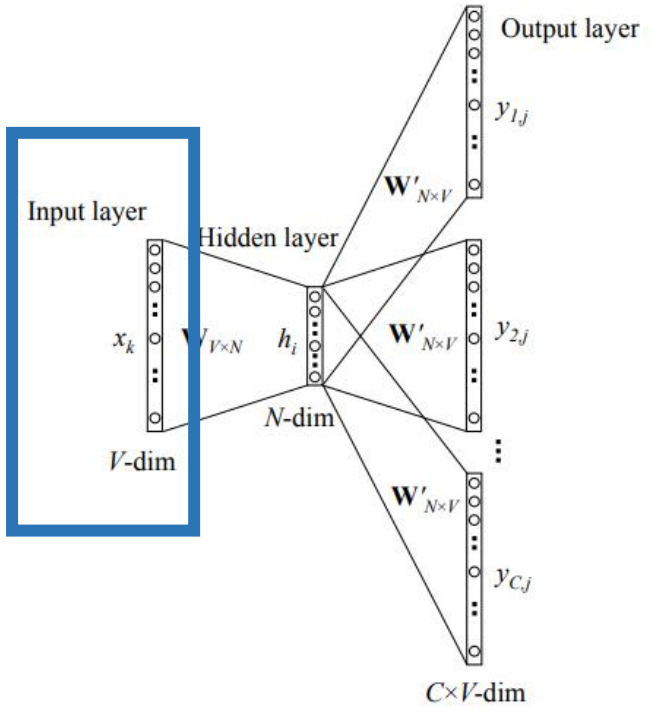


Figure 3: The skip-gram model.

Example

우리 집 강아지는 복슬 강아지 (__ _ 강아지_ __ ___)

2. Word Embedding

word2vec - SkipGram 학습과정 살펴보기 (Input Layer -> Hidden Layer)

One Hot Encoding 한 vector와 Embedding matrix를 곱해,

N 차원의 embedding vector를 얻는다.

강아지 : [0, 0, 1, 0, 0] @ Embedding matrix

=> 강아지의 embedding vector : [0.2, 0.8, 1.9, -0.5]

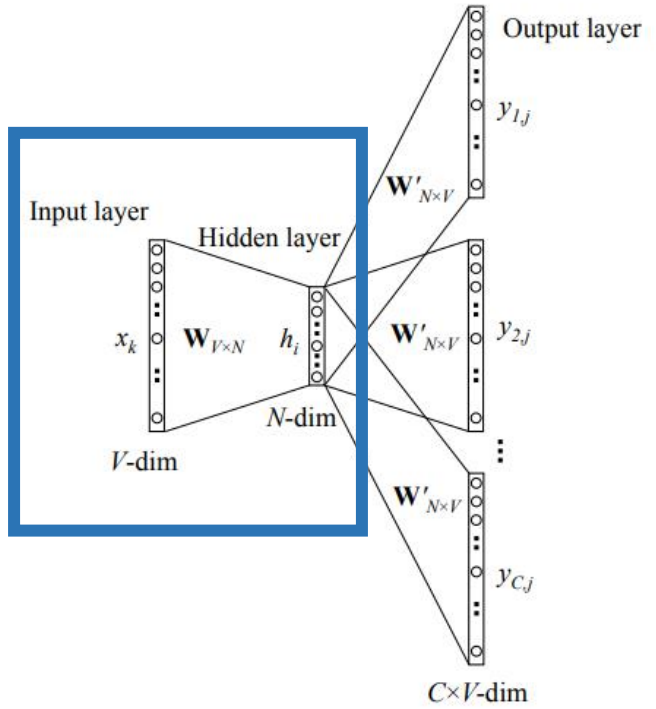


Figure 3: The skip-gram model.

Example

우리 집 강아지는 복슬 강아지 (__ _ 강아지_ __ ___)

2. Word Embedding

word2vec - SkipGram 학습과정 살펴보기 (Hidden Layer -> Output Layer)

N 차원의 embedding vector와 Score matrix를 곱해

C(=2m) 개 만큼 output vector (score vector)를 얻는다.

강아지 : [0.2, 0.8, 1.9, -0.5] @ Score matrix

=> 강아지의 score vector : [0.5, 0.1, -0.8, -1.5, 0.9] X C

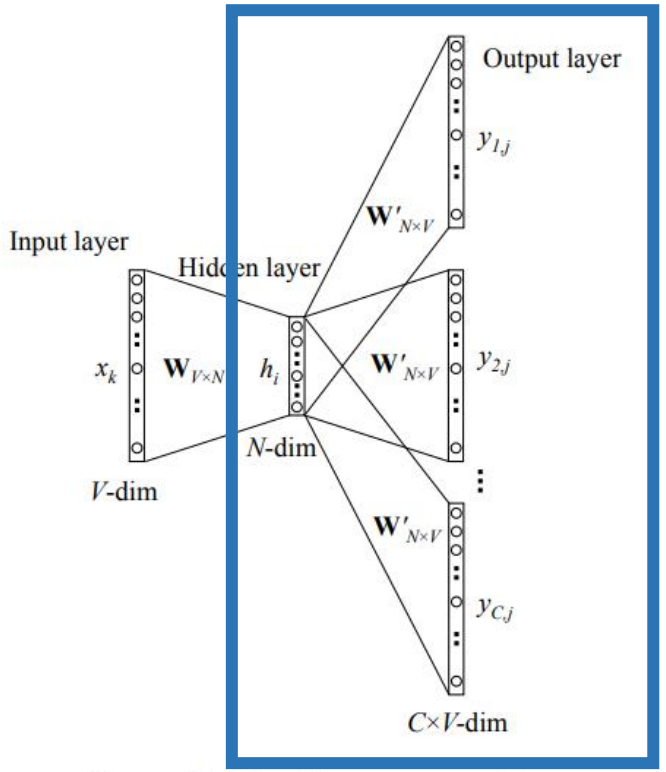


Figure 3: The skip-gram model.

Example

우리 집 강아지는 복슬 강아지 (__ _ 강아지_ __ ___)

2. Word Embedding

word2vec - SkipGram 학습과정 살펴보기 (Output Layer)

Score vector를 확률값으로 나타내기 위해 Softmax 함수를 취한다.

그 후 target vector와 차이를 이용하여 역전파 알고리즘 사용하여
Embedding Matrix, Score Matrix 를 update한다.

[0.5, 0.1, -0.8, -1.5, 0.9] => Predict : [0.28, 0.19, 0.08, 0.04, 0.41]

Target : 우리 : [1, 0, 0, 0, 0]
집 : [0, 1, 0, 0, 0]
는 : [0, 0, 0, 1, 0]
복슬 : [0, 0, 0, 0, 1]
강아지 : [0, 0, 1, 0, 0]

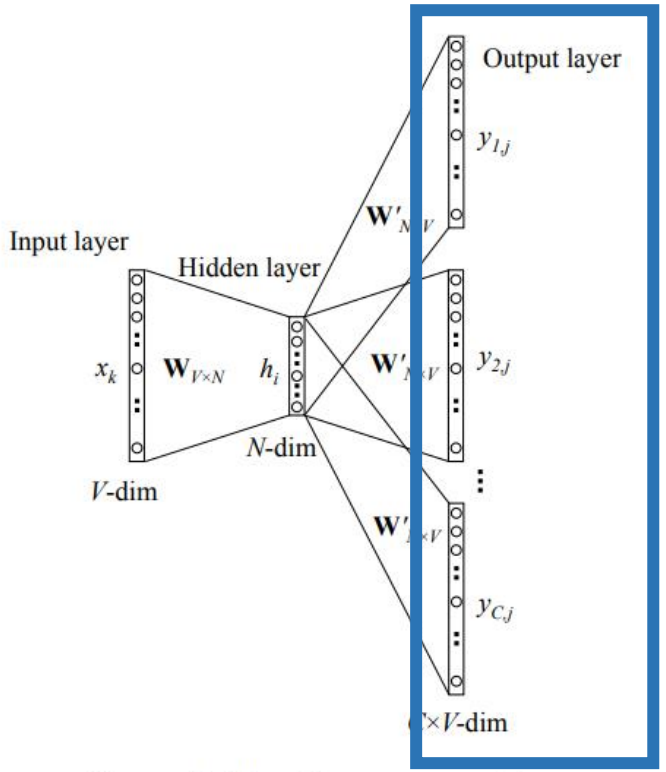


Figure 3: The skip-gram model.

Example

우리 집 강아지는 복슬 강아지 (__ _ 강아지_ __ ___)

2. Word Embedding

word2vec : 단어 벡터 간 유의미한 유사도를 반영하도록 단어의 의미를 수치화하는 것

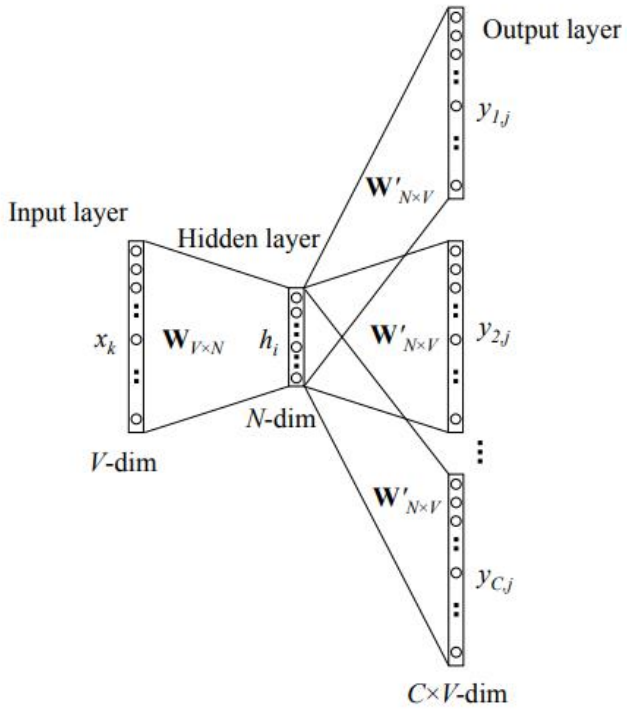
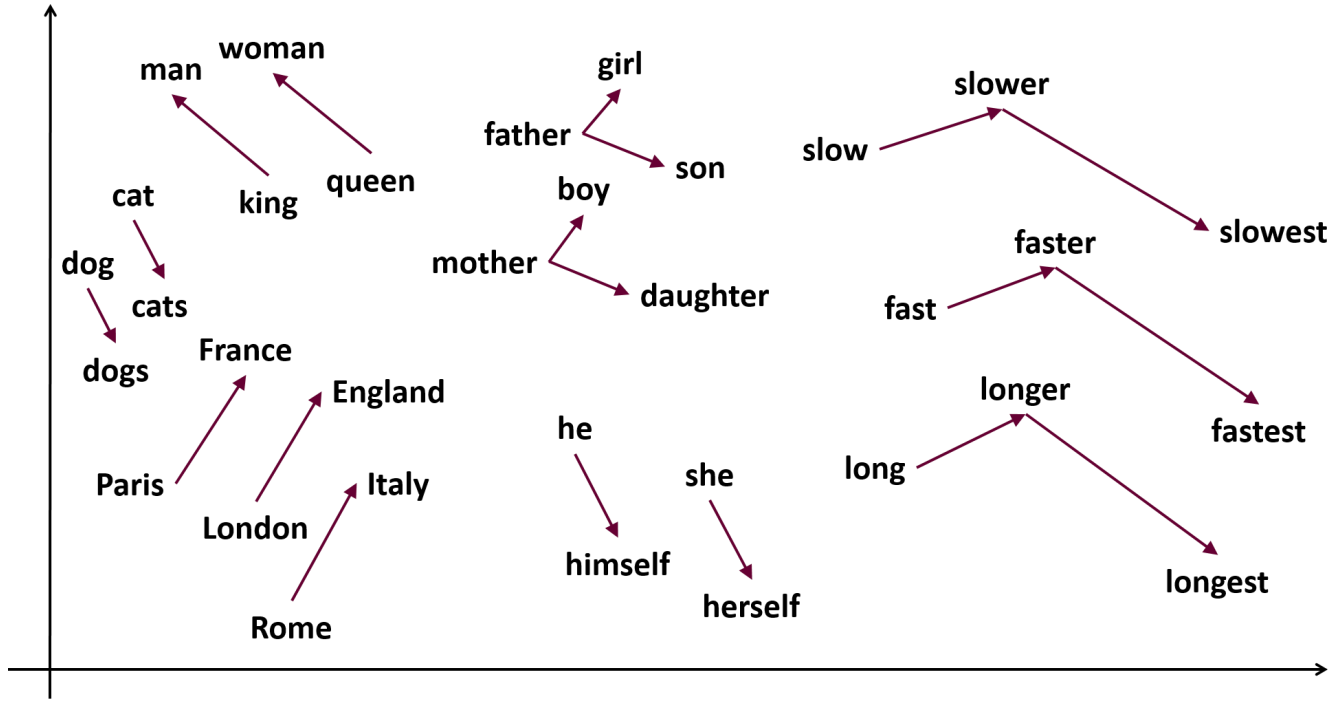
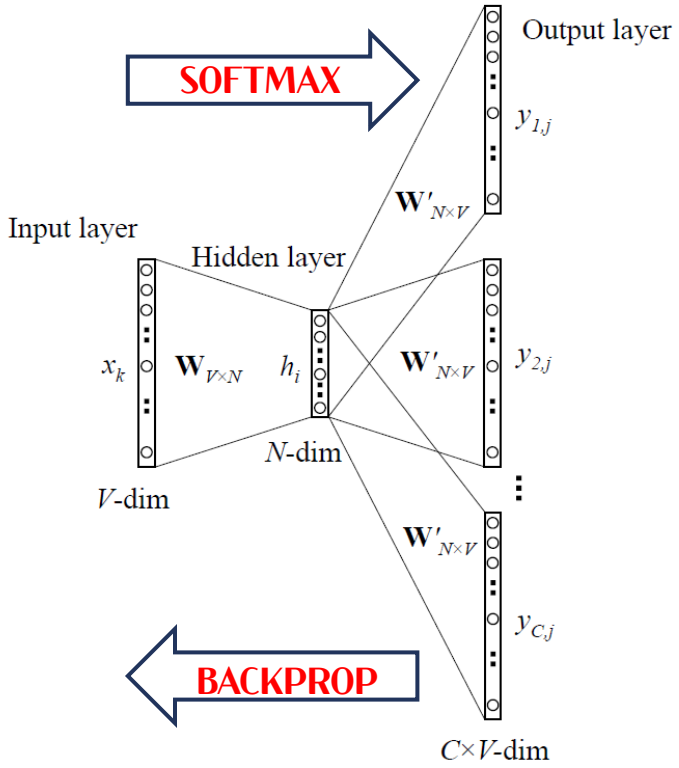


Figure 3: The skip-gram model.



2. Word Embedding

word2vec 단점 - Too Expensive Computation



for each $v_j \in W$ **do** (**each word in context words**)

  for each $u_k \in Context\ Words$ **do**

    $EH \leftarrow 0$

    for each $i = 1\ to\ V$ **do**

      $EI_i \leftarrow softmax(W'^T v_{W_j}) - t_i$

      $EH \leftarrow EH + W' \cdot EI_i$

      $v_{W'_i}^{new} \leftarrow v_{W'_i}^{old} - \alpha \cdot EI_i \cdot h$

    end for

    $v_{W_j}^{new} \leftarrow v_{W_j}^{old} - \alpha \cdot EH^T$

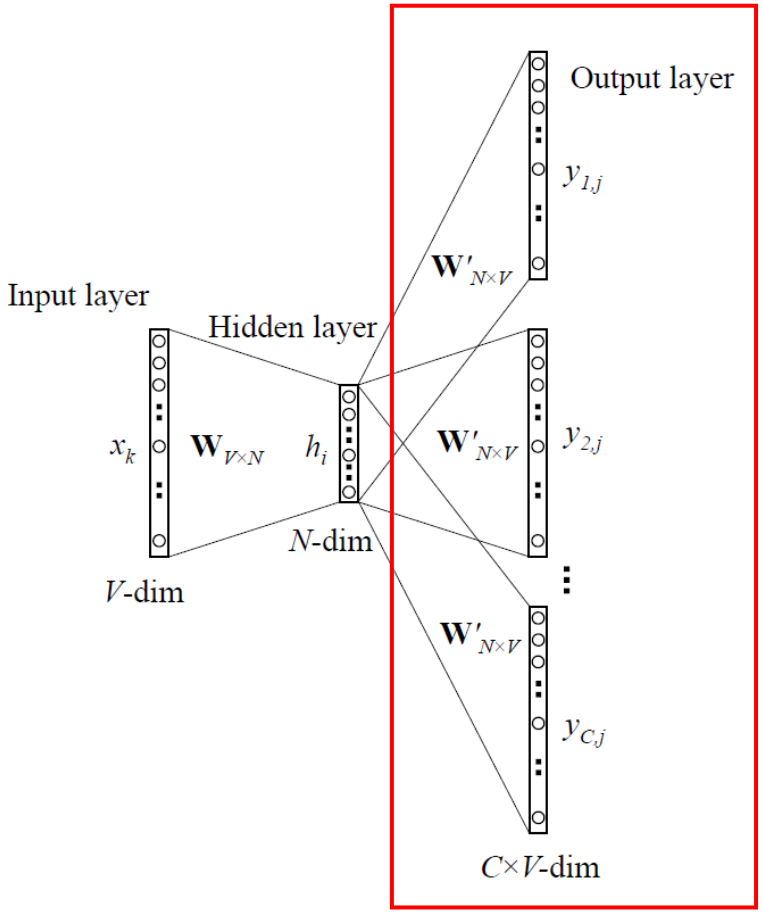
  end for

end for

➔ **Too expensive computation**

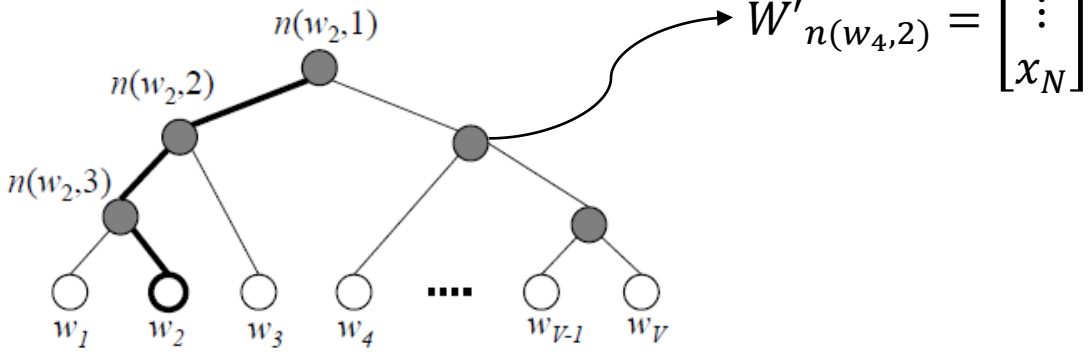
2. Word Embedding

word2vec – Hierarchical Softmax



$V \rightarrow \log_2 V$

Build Binary Tree



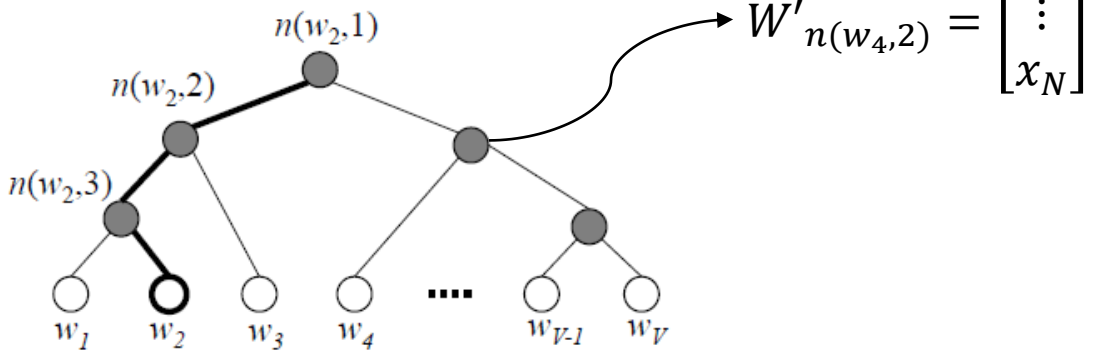
Build Binary Tree : V leaves, $V-1$ inner units

$$p(w = w_O) = \prod_{j=1}^{L(w_O)-1} \sigma([\![n(w_O, j+1) = ch(n(w_O, j))]\!] \cdot W'^{T}_{n(w,j)} \cdot h)$$

2. Word Embedding

word2vec – Hierarchical Softmax

Build Binary Tree



Build Binary Tree : V leaves, $V-1$ inner units

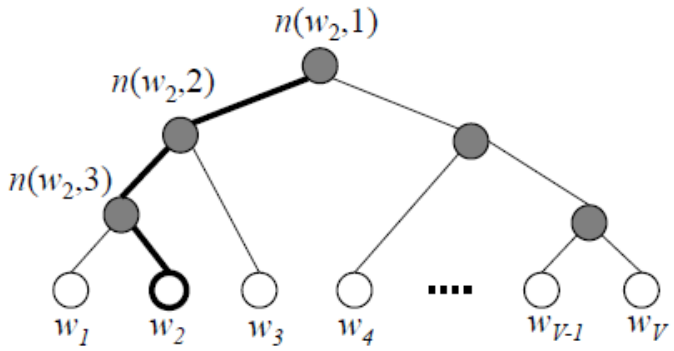
$$p(w = w_O) = \prod_{j=1}^{L(w_O)-1} \sigma([\![n(w_O, j+1) = ch(n(w_O, j))]\!] \cdot W'^{T}_{n(w,j)} \cdot h)$$

2. Word Embedding

word2vec – Hierarchical Softmax

$$p(w = w_O) = \prod_{j=1}^{L(w_O)-1} \sigma([\![n(w_O, j+1) = ch(n(w_O, j))]\!] \cdot W'^{T}_{n(w,j)} \cdot h)$$

Build Binary Tree



$[\![x]\!] = 1$ **if x is True**
$[\![x]\!] = -1$ **if x is False**

$ch(n)$ **: left child of unit n**

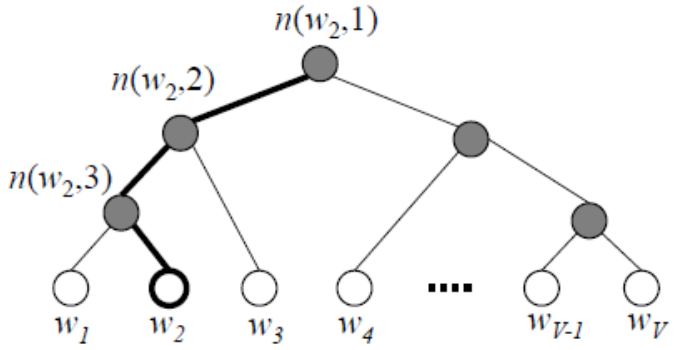
$n(w_O, j+1) = ch(n(w_O, j))$ **:** w_O**까지 가는 경로에서 왼쪽으로 뻗어 나가면 True, 오른쪽으로 뻗어 나가면 False**

Ex) w_2**까지 가는 경로에서 1, 1, -1 을 반환하게 됨.**

2. Word Embedding

word2vec – Hierarchical Softmax

Build Binary Tree



$$p(w = w_O) = \prod_{j=1}^{L(w_O)-1} \sigma([\![n(w_O, j+1) = ch(n(w_O, j))]\!] \cdot W'^{T}_{n(w,j)} \cdot h)$$

$$p(w = w_2) = p(n(w_2,1), left) \cdot p(n(w_2,2), left) \cdot p(n(w_2,3), right)$$

$$= \sigma(W'^{T}_{n(w_2,1)} \cdot h) \cdot \sigma(W'^{T}_{n(w_2,2)} \cdot h) \cdot \sigma(-W'^{T}_{n(w_2,3)} \cdot h)$$

=> **연산량 : 3**

$$p(w = w_2) = \frac{\exp(w_2)}{\sum_{j=1}^{V} \exp(w_j)}$$

=> **연산량 : V**

word2vec – Hierarchical Softmax

Error Function

Softmax

$$E = -\log \prod_{c=1}^{C} \frac{\exp\left(u_{c,j_c^*}\right)}{\sum_{j'=1}^{V} \exp\left(u_{j'}\right)} \quad (u_j = W'^{T}_{j} \cdot h)$$

Hierarchical Softmax

$$E = -\log \prod_{j=1}^{L(w_O)-1} \sigma([\![n(w_O, j+1) = ch(n(w_O, j))]\!] \cdot W'^{T}_{n(w,j)} \cdot h)$$

Update W Matrix

Softmax

$$W_{w_I}^{new} = W_{w_I}^{old} - \alpha \cdot EH^T, \quad (EH_i = \sum_{j=1}^{V} EI_j \, W'_{ij})$$

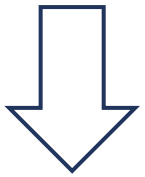
Hierarchical Softmax

$$W_{w_I}^{new} = W_{w_I}^{old} - \alpha \cdot EH^T, \quad (EH = \sum_{j=1}^{L(w)-1} (\sigma\left(W'^{T}_{n(w,j)} \cdot h\right) - t_j) \cdot W'_{n(w,j)})$$

word2vec – Negative Sampling

Only the weights corresponding to the target word might get a significant update.

The calculation of the final probabilities using the softmax is quite an expensive operation.



The idea of negative sampling is more straightforward than hierarchical softmax: in order to deal with the difficulty of having too many output vectors that need to be updated per iteration, we only update a sample of them.

Apparently the output word (i.e., the ground truth, or positive sample) should be kept in our sample and gets updated, and we need to sample a few words as negative samples (hence "negative sampling"). A probabilistic distribution is needed for the sampling process, and it can be arbitrarily chosen. We call this distribution the noise distribution, and denote it as $P_n(w)$. One can determine a good distribution empirically.[6]

word2vec – Negative Sampling

The idea of negative sampling is more straightforward than hierarchical softmax: in order to deal with the difficulty of having too many output vectors that need to be updated per iteration, we only update a sample of them.

Apparently the output word (i.e., the ground truth, or positive sample) should be kept in our sample and gets updated, and we need to sample a few words as negative samples (hence "negative sampling"). A probabilistic distribution is needed for the sampling process, and it can be arbitrarily chosen. We call this distribution the noise distribution, and denote it as $P_n(w)$. One can determine a good distribution empirically.[6]

$$E = -\log \sigma({\mathbf{v}'_{w_O}}^T \mathbf{h}) - \sum_{w_j \in \mathcal{W}_{\text{neg}}} \log \sigma(-{\mathbf{v}'_{w_j}}^T \mathbf{h})$$

$$W_{neg} = \{w_j | j = 1, 2, 3, \dots, k\} \sim P_n(w): noise\ dist'n$$

$\mathrm{P}_n(w_i) = \frac{f(w_i)}{\sum_{j=0}^{n} f(w_j)}$, $f(w_i): frequency\ of\ w_i$, for better result, $\mathrm{P}_n(w_i) = \frac{f(w_i)^{\frac{3}{4}}}{\sum_{j=0}^{n} f(w_j)^{\frac{3}{4}}}$

$0.9^{\frac{3}{4}} = 0.9240, \qquad 0.2^{\frac{3}{4}} = 0.2990$

2. Word Embedding

word2vec – Negative Sampling

$$
\begin{aligned}
\theta &= \underset{\theta}{\text{argmax}} \prod_{(w,c)\in D} P(D=1|w,c,\theta) \prod_{(w,c)\in \tilde{D}} P(D=0|w,c,\theta) \\
&= \underset{\theta}{\text{argmax}} \prod_{(w,c)\in D} P(D=1|w,c,\theta) \prod_{(w,c)\in \tilde{D}} (1-P(D=1|w,c,\theta)) \\
&= \underset{\theta}{\text{argmax}} \sum_{(w,c)\in D} \log P(D=1|w,c,\theta) + \sum_{(w,c)\in \tilde{D}} \log(1-P(D=1|w,c,\theta)) \\
&= \underset{\theta}{\text{argmax}} \sum_{(w,c)\in D} \log \frac{1}{1+\exp(-u_w^T v_c)} + \sum_{(w,c)\in \tilde{D}} \log\left(1-\frac{1}{1+\exp(-u_w^T v_c)}\right) \\
&= \underset{\theta}{\text{argmax}} \sum_{(w,c)\in D} \log \frac{1}{1+\exp(-u_w^T v_c)} + \sum_{(w,c)\in \tilde{D}} \log\left(\frac{1}{1+\exp(u_w^T v_c)}\right)
\end{aligned}
$$

Maximize the probability of co-occurrence for actual words that lie in the context

Minimize the probability of co-occurrence for some random words that don't lie in the context

$$
O = \log \sigma({\mathbf{v}'_{w_O}}^T \mathbf{h}) + \sum_{w_j \in \mathcal{W}_{\text{neg}}} \log \sigma(-{\mathbf{v}'_{w_j}}^T \mathbf{h}) \qquad E = -\log \sigma({\mathbf{v}'_{w_O}}^T \mathbf{h}) - \sum_{w_j \in \mathcal{W}_{\text{neg}}} \log \sigma(-{\mathbf{v}'_{w_j}}^T \mathbf{h})
$$

2. Word Embedding

word2vec – Negative Sampling

Ex. 우리 집 강아지는 복슬 강아지

Input	Target
강아지	집
강아지	는
는	강아지
는	복슬
복슬	는
복슬	강아지

Input1	Input2	Output
강아지	집	1
강아지	는	1
는	강아지	1
는	복슬	1
복슬	는	1
복슬	강아지	1

2. Word Embedding

word2vec – Negative Sampling

Ex. 우리 집 강아지는 복슬 강아지

Input1	Input2	Output
강아지	집	1
강아지	는	1
는	강아지	1
는	복슬	1
복슬	는	1
복슬	강아지	1

Randomly choose from $P_n(w)$

Randomly choose from $P_n(w)$

Randomly choose from $P_n(w)$

Input1	Input2	Output
강아지	집	1
강아지	는	1
강아지	우리	0
는	강아지	1
는	복슬	1
는	우리	0
복슬	는	1
복슬	강아지	1
복슬	집	0

2. Word Embedding

word2vec – Negative Sampling

Ex. 우리 집 강아지는 복슬 강아지

minimize $E = -\log \sigma({\mathbf{v}'_{w_O}}^T \mathbf{h}) - \sum_{w_j \in \mathcal{W}_{\text{neg}}} \log \sigma(-{\mathbf{v}'_{w_j}}^T \mathbf{h})$

$1 - \sigma({v'_{w_j}}^T h)$

Input1	Input2	Output
강아지	집	1
강아지	는	1
강아지	우리	0
는	강아지	1
는	복슬	1
는	우리	0
복슬	는	1
복슬	강아지	1
복슬	집	0

word2vec – Negative Sampling

Error Function

Softmax

$$E = -\log \prod_{c=1}^{C} \frac{\exp\left(u_{c,j_c^*}\right)}{\sum_{j'=1}^{V} \exp\left(u_{j'}\right)} \quad (u_j = {v'}_{w_j}^{T} \cdot h)$$

Negative Sampling

$$E = -\log \sigma\left({v'}_{w_O}^{T} h\right) - \sum_{w_j \in W_{neg}} \log \sigma(-{v'}_{w_j}^{T} h)$$

Backpropagation

Softmax

$$v_{W_I}^{new} = v_{W_I}^{old} - \alpha \cdot EH^T, \ (EH_i = \sum_{j=1}^{V} EI_j \, {v'}_{ij})$$

Negative Sampling

$$v_{W_I}^{new} = v_{W_I}^{old} - \alpha \cdot EH^T, \ (EH = \sum_{w_j \in \{w_O\} \cup W_{neg}} (\sigma\left({v'}_{w_j}^{T} \cdot h\right) - t_j) \cdot {v'}_{w_j})$$

2. Word Embedding

Other Methods

FastText : word2vec의 확장판.

1. 내부단어(subword)의 학습

자연어처리 -> <자연, 자연어, 연어처, 어처리, 처리>
-> <ㅈㅏ, ㅈㅏ_, ㅏ_ㅇ, _ㅇㅓ, ㅇㅓㄴ, …

2. 모르는 단어(Out Of Vocabulary, OOV)에 대한 대응

3. 단어 집합 내 빈도 수가 적었던 단어(Rare Word)에 대한 반응

GloVe : word2vec의 예측 기반 방법론과 카운트 기반 방법론을 모두 사용함.

3. Visualization

PCA

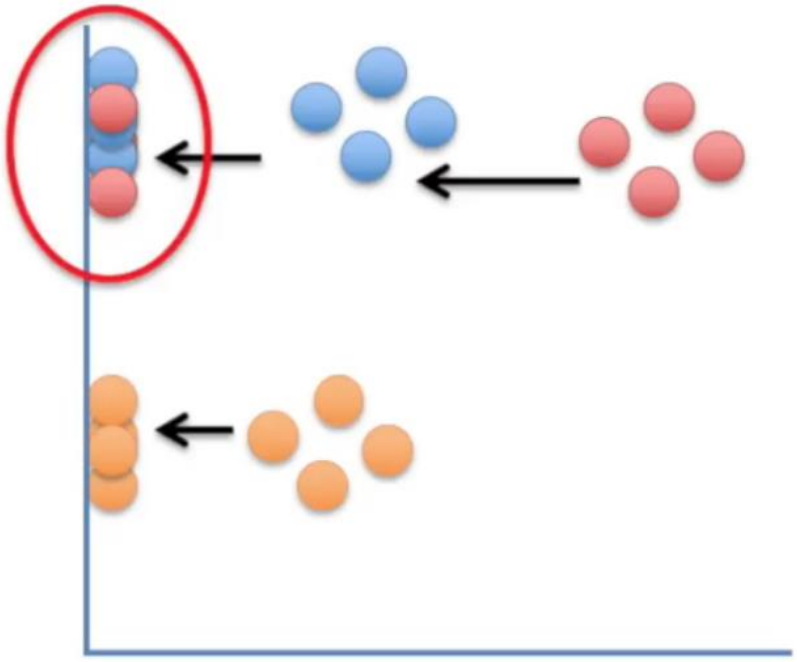
Dimension Reduction : N차원으로 나타낸 embedding vector를 눈으로 보고 싶다!

설명력이 높은 2~3개의 주성분을 이용하여 차원 축소 후 plot!

옛날에 배웠으니까 넘어갈게요~!

3. Visualization

PCA



PCA의 단점

선형 분석 방식으로 값을 mapping 하기 때문에
차원이 감소하면서 군집화 되어 있는 데이터들이 뭉개져서
제대로 구별할 수 없는 문제가 발생할 수 있음.

3. Visualization

t-SNE (t-distributed Stochastic Neighbor Embedding)

$$p_{j|i} = \frac{e^{-\frac{||\mathbf{x}_i - \mathbf{x}_j||^2}{2\sigma_i^2}}}{\sum_{k \neq i} e^{-\frac{||\mathbf{x}_i - \mathbf{x}_k||^2}{2\sigma_i^2}}}$$

원래 차원에서 i가 j를 이웃으로 선택할 확률

$$q_{j|i} = \frac{e^{-||\mathbf{y}_i - \mathbf{y}_j||^2}}{\sum_{k \neq i} e^{-||\mathbf{y}_i - \mathbf{y}_k||^2}}$$

축소한 차원에서 i가 j를 이웃으로 선택할 확률

y를 구할 수 있는가? 지금 당장은 못 구한다...

$$\Rightarrow \quad Cost = \sum_i KL(P_i||Q_i) = \sum_i \sum_j p_{j|i} log \frac{p_{j|i}}{q_{j|i}}$$

$$\frac{\partial C}{\partial \mathbf{y}_i} = 2 \sum_j (\mathbf{y}_j - \mathbf{y}_i)(p_{j|i} - q_{j|i} + p_{i|j} - q_{i|j})$$

!!!Update rule!!!

3. Visualization

t-SNE (t-distributed Stochastic Neighbor Embedding)

$$p_{j|i} = \frac{e^{-\frac{||\mathbf{x}_i - \mathbf{x}_j||^2}{2\sigma_i^2}}}{\sum_{k \neq i} e^{-\frac{||\mathbf{x}_i - \mathbf{x}_k||^2}{2\sigma_i^2}}} \qquad q_{j|i} = \frac{e^{-||\mathbf{y}_i - \mathbf{y}_j||^2}}{\sum_{k \neq i} e^{-||\mathbf{y}_i - \mathbf{y}_k||^2}}$$

원래 차원에서 i가 j를 이웃으로 선택할 확률 / 축소한 차원에서 i가 j를 이웃으로 선택할 확률

$$p_{ij} = \frac{e^{-\frac{||\mathbf{x}_i - \mathbf{x}_j||^2}{2\sigma_i^2}}}{\sum_{k \neq l} e^{-\frac{||\mathbf{x}_k - \mathbf{x}_l||^2}{2\sigma_i^2}}} \Rightarrow p_{ij} = \frac{p_{j|i} + p_{i|j}}{2n} \qquad \sum_j p_{ij} > \frac{1}{2n}$$

i와 j가 서로를 이웃으로 선택할 확률 (pairwise probability)

$$Cost = \sum_i KL(P_i||Q_i) = \sum_i \sum_j p_{j|i} log \frac{p_{j|i}}{q_{j|i}}$$

$$\frac{\partial C}{\partial \mathbf{y}_i} = 2 \sum_j (\mathbf{y}_j - \mathbf{y}_i)(p_{j|i} - q_{j|i} + p_{i|j} - q_{i|j})$$

$$Cost = \sum_i KL(P_i||Q_i) = \sum_i \sum_j p_{ij} log \frac{p_{ij}}{q_{ij}}$$

$$\frac{\partial C}{\partial \mathbf{y}_i} = 4 \sum_j (\mathbf{y}_j - \mathbf{y}_i)(p_{ij} - q_{ij})$$

Perplexity : SNE는 거리에 반비례하게 영향력을 정의하여 embedding 하는데 사용한다.

이때 Perplexity란 어느 범위까지 영향력을 강하게 할 것인가를 결정하는 hyperparameter

5~50 사이의 값에서 robust함.

3. Visualization

t-SNE (t-distributed Stochastic Neighbor Embedding)

왜 "t"-SNE일까??

Crowding problem : $$q_{j|i} = \frac{e^{-||\mathbf{y}_i - \mathbf{y}_j||^2}}{\sum_{k \neq i} e^{-||\mathbf{y}_i - \mathbf{y}_k||^2}}$$

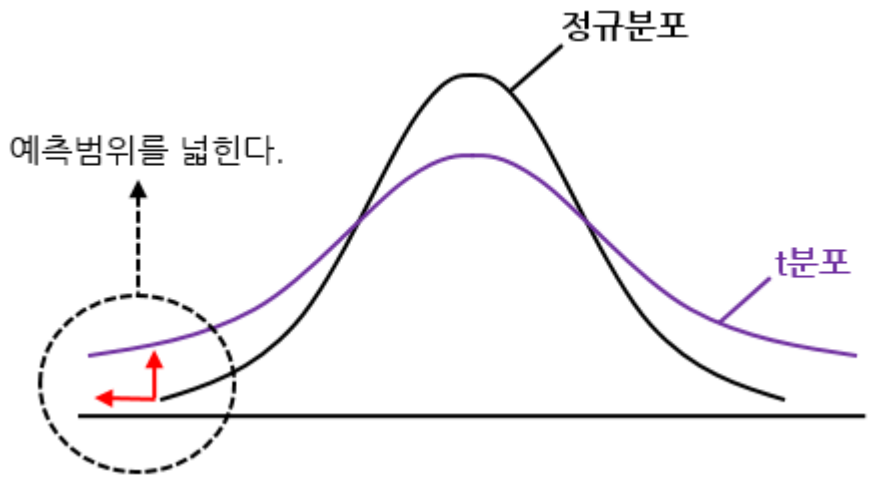
무슨 분포가 생각나시나요?

3. Visualization

t-SNE (t-distributed Stochastic Neighbor Embedding)

왜 "t"-SNE일까??

Crowding problem : 정규분포는 중심에서부터 멀어지면 급격히 값이 감소함.
어느 정도 멀리 있는 데이터들은 서로 간의 위치 정보를 담을 수 없음.

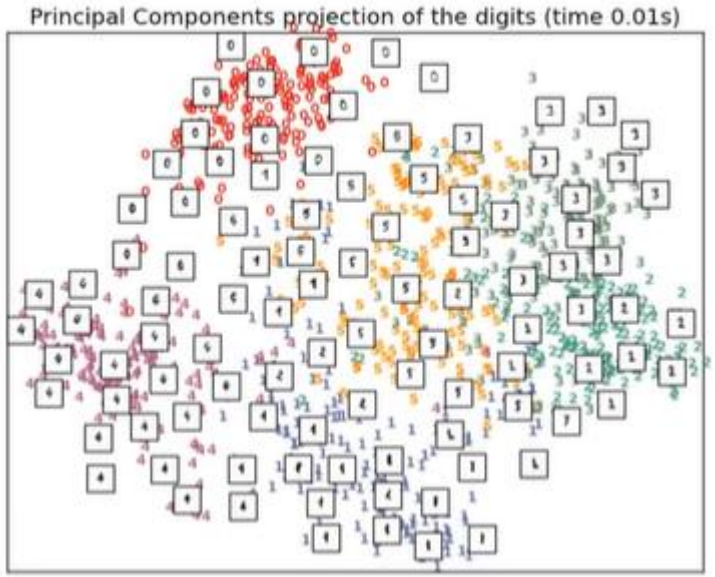


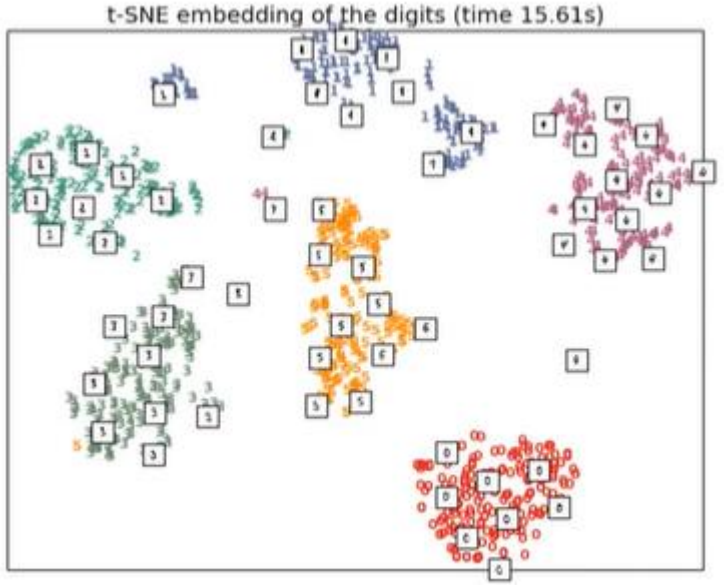
$$q_{j|i} = \frac{e^{-||\mathbf{y}_i - \mathbf{y}_j||^2}}{\sum_{k \neq i} e^{-||\mathbf{y}_i - \mathbf{y}_k||^2}} \Rightarrow q_{ji} = \frac{(1 + ||\mathbf{y}_i - \mathbf{y}_j||^2)^{-1}}{\sum_{k \neq l}(1 + ||\mathbf{y}_k - \mathbf{y}_l||^2)^{-1}}$$

$$\frac{\partial C}{\partial \mathbf{y}_i} = 4 \sum_j (\mathbf{y}_j - \mathbf{y}_i)(p_{ij} - q_{ij})(1 + ||\mathbf{y}_i - \mathbf{y}_j||^2)^{-1}$$

3. Visualization

t-SNE (t-distributed Stochastic Neighbor Embedding)





t-SNE를 사용 했을 때 PCA보다 더 군집이 잘 떨어져 있는 모습을 확인할 수 있음.

https://distill.pub/2016/misread-tsne/

Summary

1. Tokenization & Cleansing

1. 한국어의 경우 형태소 단위, 음절 단위 등으로 분석 가능함
2. KoNLPy + Huggingface 의 Tokenizer 사용 가능함
3. Tokenization이든 Cleansing에든 정해진 방법은 없음

2. Word Embedding - word2vec

1. word2vec은 좋은 word embedding 방법임
2. Heavy computation을 방지하기 위해 HS, NS가 고안됨
3. 그 외에도 FastText, GloVe 같은 embedding 방법이 있음

3. Visualization - PCA, t-SNE

1. PCA, t-SNE 모두 좋은 저차원 embedding 방법임
2. 여러 가지 시도해보고 좋은 것으로 visualization 하는 것이 좋음

Reference

Reference

https://en.wikipedia.org/wiki/Text_mining

딥 러닝을 이용한 자연어 처리 입문 https://wikidocs.net/book/2155

Rong, Xin. "word2vec parameter learning explained." (2014).

Mikolov, Tomas, et al. "Efficient estimation of word representations in vector space.“ (2013).

https://www.youtube.com/watch?v=INHwh8k4XhM

Image Reference

https://www.tibco.com/ko/reference-center/what-is-text-mining

https://www.expressanalytics.com/blog/social-media-sentiment-analysis/

https://www.samyzaf.com/ML/nlp/nlp.html

Thank you!!!

22.08.30 / DSL 7기 최명헌