Python을 활용한 데이터 분석 강의

Text Mining

HYUNDAI MOBIS

Text Data

- Documents (문서)
  - articles, books, novels
  - 메일, 웹페이지, 블로그, 트위터
  - 태그, 코멘트

- Collection of documents
  - 메세지
  - 소셜네트워크 데이터
  - publications

Text Data

- WHY text?
  - Understanding
    - 문서의 핵심을 파악하기 위해
  - Grouping
    - 전체를 조망하기 위해 cluster를 만들거나 분류를 하기 위해
  - Comparing
    - document collection을 비교하거나 collection이 어떻게 시간이 따라 변화해 왔는지 파악하기 위해
  - Correlation
    - 텍스트에 나타나는 패턴을 다른 데이터와 비교하기 위해

Text Preprocessing

Tokenization (토큰화)

Stemming (어간 추출)

lemmatization (표제어 추출)

Bag of Words(BoW)

- 단어들의 순서를 고려하지 않고 빈도에만 집중하는텍스트 데이터의 수치화 표현 방법
- 문서 내에서 특정 단어가 N번 등장했다면, 이 가방에는 그 특정 단어가 N 개 있는 것

- BoW를 만드는 과정
  - 각 단어에 고유한 인덱스(index)를 부여
  - 각 인덱스의 위치에 토큰의 등장 횟수를 기록한 벡터(vector)를 만듦

Term-Document Matrix(TDM)

- 서로 다른 문서들의 BoW를 결합한 표현 방법
- 다수의 문서에서 등장하는 각 단어들의 빈도를 행렬로 표현한 것

문서1 : 먹고 싶은 사과

문서2 : 먹고 싶은 바나나

문서3 : 길고 노란 바나나 바나나

문서4 : 저는 과일이 좋아요

	과일이	길고	노란	먹고	바나나	사과	싶은	저는	좋아요
문서1	0	0	0	1	0	1	1	0	0
문서2	0	0	0	1	1	0	1	0	0
문서3	0	1	1	0	2	0	0	0	0
문서4	1	0	0	0	0	0	0	1	1

Term-Document Matrix(TDM)

- TDM을 만들었을 때, stopwords인 “the”는 어떤 문서든 자주 등장
- 문서1, 문서2, 문서3에서 동일하게 “the”가 빈도수가 높다고 해서 이 문서들이 유사한 문서라고 판단해서는 안된다!
- TDM에 stopwords 중요한 단어에 대해서 가중치를 줄 수 있는 방법?

HYUNDAI MOBIS

TF-IDF
(Term Frequency-Inverse Document Frequency)

- 단어의 빈도와 역 문서 빈도(문서의 빈도에 특정 식을 취함)를 사용하여 TDM 내의 각 단어들마다 중요한 정도를 가중치로 주는 방법

- TDM을 만든 후에, TF-IDF 가중치를 준다
  - 문서의 유사도를 구하기
  - 검색 시스템에서 검색 결과의 중요도 정하기
  - 문서 내에서 특정 단어의 중요도를 구하기

TF-IDF
(Term Frequency-Inverse Document Frequency)

- TF-IDF는 TF와 IDF를 곱한 값을 의미
- 문서를 d, 단어를 t, 문서의 총 개수를 n로 두고
  - tf(d,t) : 특정 문서 d에서의 특정 단어 t의 등장 횟수
  - df(t) : 특정 단어 t가 등장한 문서의 수
  - idf(t) : df(t)에 반비례하는 수 $\mathrm{idf}(d,t) = \log \frac{n}{1+\mathrm{df}(t)}$

cf) idf를 df의 역수로 사용할 경우, n이 커질 수록 idf의 값은 기하급수적으로 커지게 된다. 이 때문에 log를 사용. log 분모에 1을 더해주는 이유는 특정 단어가 전체 문서에서 한 번도 등장하지 않을 경우에 분모가 0이 되는 상황을 방지하기 위함이다.

TF-IDF
(Term Frequency-Inverse Document Frequency)

	과일이	길고	노란	먹고	바나나	사과	싶은	저는	좋아요
문서1	0	0	0	1	0	1	1	0	0
문서2	0	0	0	1	1	0	1	0	0
문서3	0	1	1	0	2	0	0	0	0
문서4	1	0	0	0	0	0	0	1	1

단어	IDF(역 문서 빈도)
과일이	ln(4/(1+1)) = 0.693147
길고	ln(4/(1+1)) = 0.693147
노란	ln(4/(1+1)) = 0.693147
먹고	ln(4/(2+1)) = 0.287682
바나나	ln(4/(2+1)) = 0.287682
사과	ln(4/(1+1)) = 0.693147
싶은	ln(4/(2+1)) = 0.287682
저는	ln(4/(1+1)) = 0.693147
좋아요	ln(4/(1+1)) = 0.693147

TF-IDF
(Term Frequency-Inverse Document Frequency)

	과일이	길고	노란	먹고	바나나	사과	싶은	저는	좋아요
문서1	0	0	0	0.287682	0	0.693147	0.287682	0	0
문서2	0	0	0	0.693147	0.287682	0	0.287682	0	0
문서3	0	0.693147	0.693147	0	0.575364	0	0	0	0
문서4	0.693147	0	0	0	0	0	0	0.693147	0.693147

Topic Modeling

- 토픽 모델(Topic model)이라는 문서들의 추상적인 주제를 발견하기 위한 통계적 모델 중 하나

- 텍스트 본문의 숨겨진 의미구조를 발견하기 위해 사용되는 텍스트 마이닝 기법

  - ex. 뉴스 기사를 토픽 별로 clustering

LDA(Latent Dirichlet Allocation)

- 어떤 토픽 또는 주제들이 존재하는지 분류하기 위해 사용되는 토픽 모델

V1	V2	V3	V4	V5	V6	V7	V8	V9	V10	V11	V12
will	movie	movie	movie	hope	acting	like	freeman	movie	movie	movie	movie
movie	top	will	don	can	great	iti	morgan	end	movies	good	can
film	imdb	can	will	life	story	fantastic	robbins	film	like	best	say
one	film	life	know	will	film	shawshank	tim	story	prison	ever	film
shawshank	time	film	just	movie	movie	redemption	best	time	one	thing	just
movies	one	time	film	never	amazing	people	performance	every	film	hope	much
redemption	list	feel	see	andy	cinematography	soul	great	ending	don	one	said
films	shawshank	watch	people	man	direction	movies	role	watch	films	watch	words
ever	movies	one	like	one	actors	many	performances	even	great	can	anything
time	best	every	can	prison	score	ones	actor	just	good	film	really
best	redemption	hope	really	give	music	movie	film	scene	even	great	good
many	number	make	watch	free	perfect	touch	actors	well	just	really	everything
can	deserves	see	review	world	beautiful	simple	one	will	much	see	great
people	rated	like	say	always	script	film	movie	plot	many	things	nothing
years	godfather	just	much	live	directing	music	cast	beginning	better	watched	think
come	films	end	want	thing	good	looks	acting	start	think	awesome	describe
see	rating	makes	reviews	freedom	excellent	lot	good	get	feel	maybe	else
classic	see	heart	write	matter	cast	parts	character	way	escape	just	like
favorite	think	get	good	get	everything	hope	ever	never	see	never	already
seen	people	movies	one	even	well	funny	gives	long	drama	say	something

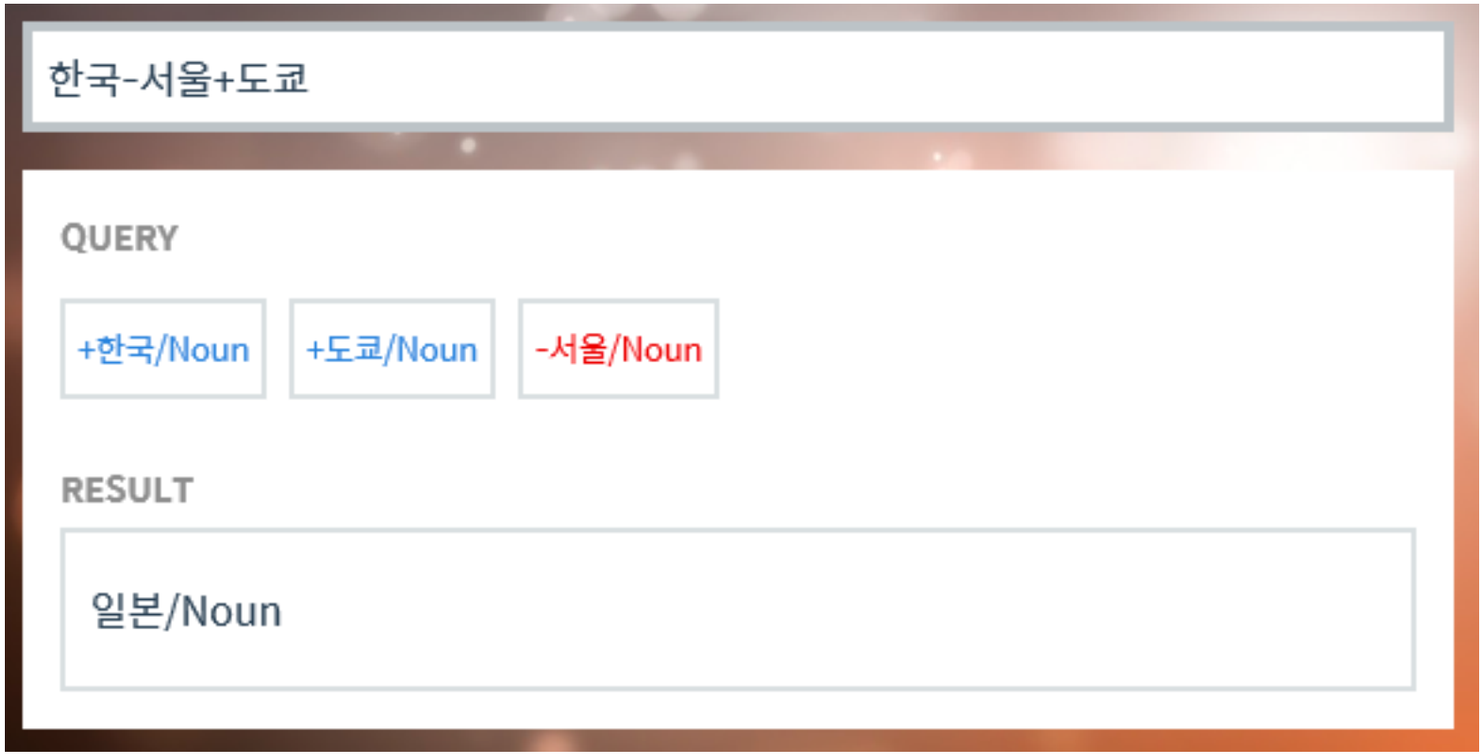
LDA(Latent Dirichlet Allocation)

- 어떤 토픽 또는 주제들이 존재하는지 분류하기 위해 사용되는 토픽 모델

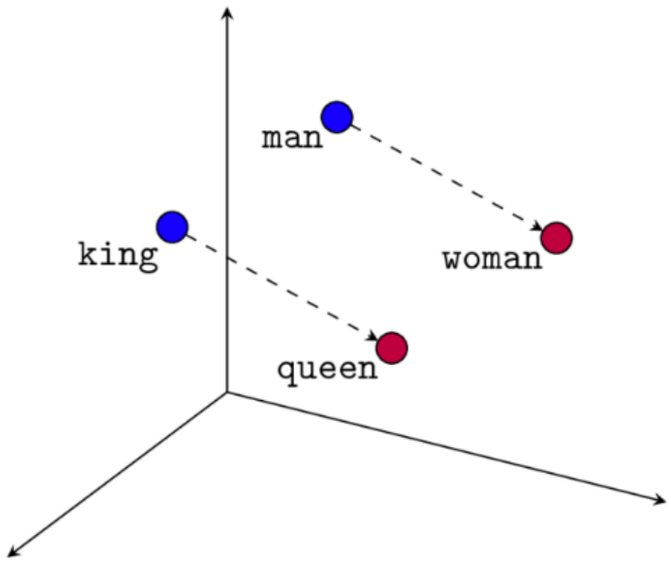
```
Topic: 0
Words: 0.031*"world" + 0.024*"cup" + 0.020*"take" + 0.017*"australia" + 0.014*"inquiry" + 0.014*"fear" + 0.013*"natio
nal" + 0.012*"industry" + 0.010*"injury" + 0.009*"woe"
Topic: 1
Words: 0.016*"indigenous" + 0.013*"doctor" + 0.012*"china" + 0.012*"case" + 0.011*"bomb" + 0.010*"safety" + 0.010*"nu
clear" + 0.010*"warning" + 0.010*"tourism" + 0.010*"u"
Topic: 2
Words: 0.018*"high" + 0.014*"hit" + 0.014*"three" + 0.013*"move" + 0.012*"australian" + 0.011*"japan" + 0.011*"ban" +
0.010*"warns" + 0.010*"big" + 0.010*"israeli"
Topic: 3
Words: 0.023*"new" + 0.019*"strike" + 0.014*"deal" + 0.013*"port" + 0.013*"win" + 0.012*"sydney" + 0.012*"tour" + 0.0
11*"title" + 0.010*"black" + 0.010*"solomon"
Topic: 4
Words: 0.034*"police" + 0.026*"u" + 0.020*"iraq" + 0.019*"crash" + 0.015*"probe" + 0.015*"bali" + 0.013*"man" + 0.012
*"car" + 0.010*"missing" + 0.010*"coast"
Topic: 5
Words: 0.020*"killed" + 0.017*"u" + 0.014*"aussie" + 0.014*"kill" + 0.013*"one" + 0.012*"attack" + 0.012*"two" + 0.01
2*"soldier" + 0.011*"return" + 0.010*"team"
Topic: 6
Words: 0.031*"govt" + 0.015*"vic" + 0.014*"help" + 0.014*"report" + 0.013*"urged" + 0.013*"bush" + 0.013*"sheep" + 0.
013*"worker" + 0.012*"appeal" + 0.012*"offer"
Topic: 7
Words: 0.036*"court" + 0.033*"man" + 0.030*"face" + 0.019*"charge" + 0.016*"new" + 0.015*"hospital" + 0.014*"put" +
0.014*"charged" + 0.013*"murder" + 0.012*"open"
Topic: 8
Words: 0.030*"council" + 0.021*"plan" + 0.018*"nsw" + 0.017*"seek" + 0.015*"govt" + 0.014*"mp" + 0.009*"wa" + 0.009
*"rate" + 0.009*"funding" + 0.009*"concern"
Topic: 9
Words: 0.022*"boost" + 0.018*"reject" + 0.017*"test" + 0.016*"water" + 0.016*"england" + 0.016*"claim" + 0.015*"powe
r" + 0.015*"back" + 0.014*"cut" + 0.012*"final"
```

Word2Vector

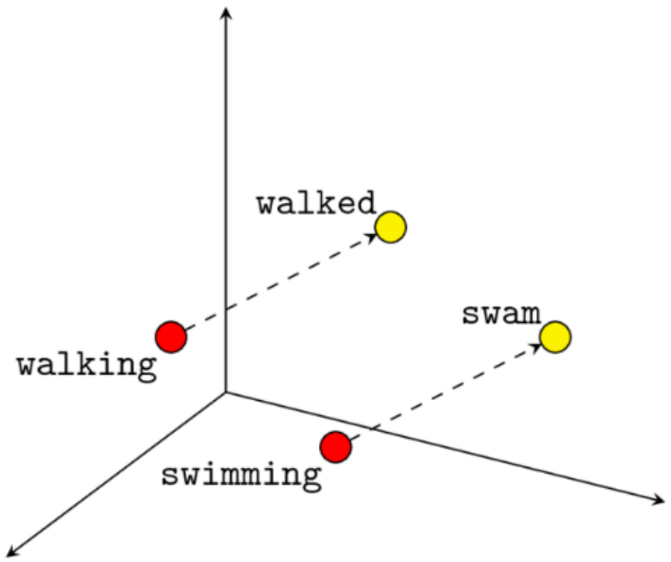
- 고양이 + 애교 = 강아지
- 한국 - 서울 + 도쿄 = 일본
- 박찬호 - 야구 + 축구 = 호나우두



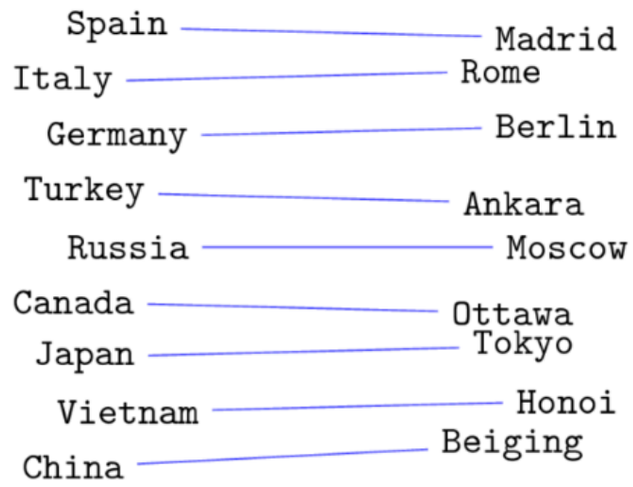
Word2Vector



성별



시제



국가-수도

HYUNDAI MOBIS

CBOW(Continuous Bag of Words)

cf) embedding : 텍스트를 구성하는 하나의 단어를 수치화하는 방법의 일종

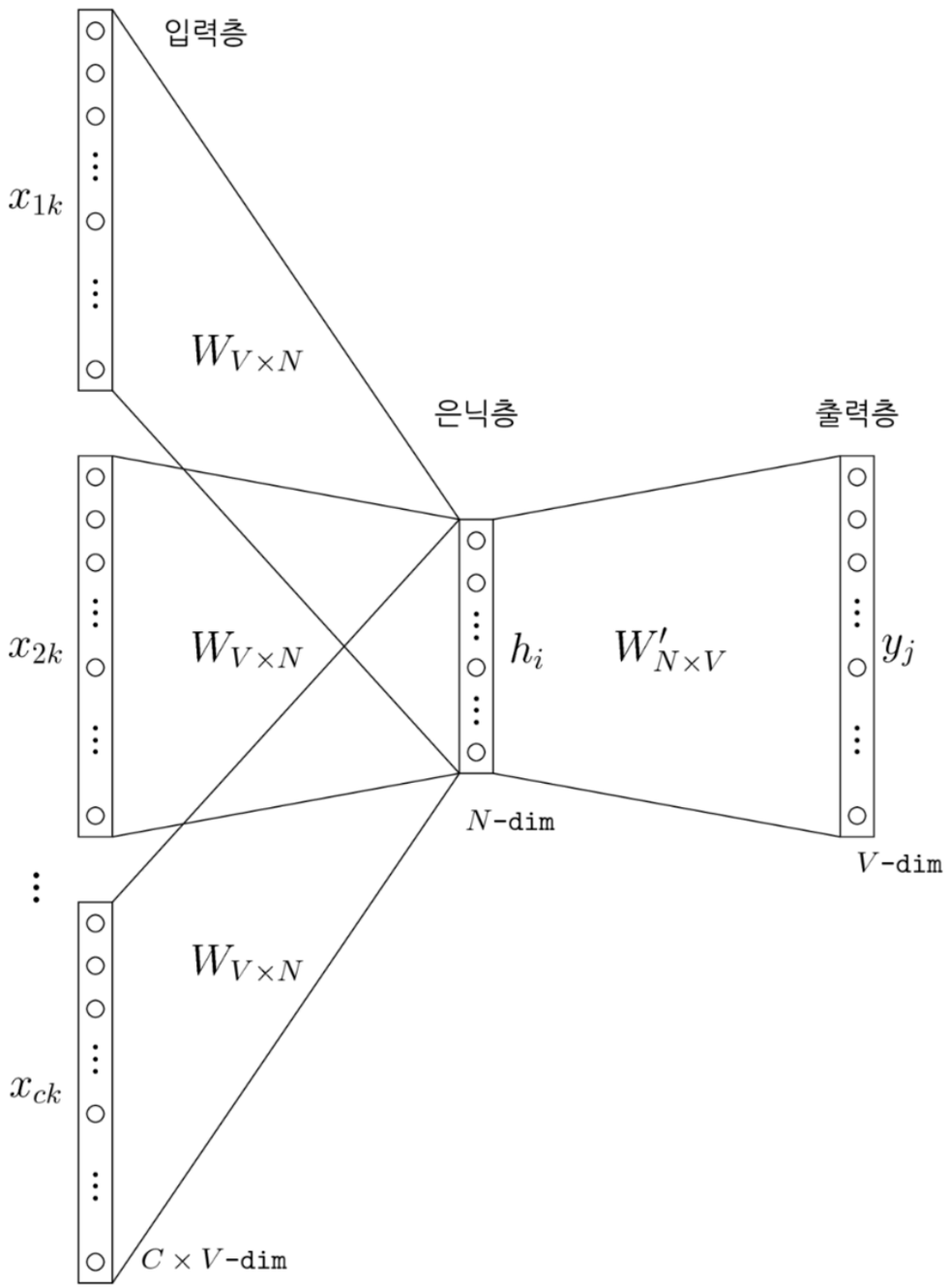
- 여러 개의 단어를 나열한 뒤 이와 관련된 단어를 추정

  => 문자에서 나오는 n개의 단어 열로부터 다음 단어를 예측

ex) the quick brown fox jumped over the lazy dog

> the, quick, brown을 주면 fox를 예측해야

CBOW(Continuous Bag of Words)



center word context words

I like playing football with my friends

I like playing football with my friends

I like playing football with my friends

I like playing football with my friends

I like playing football with my friends

I like playing football with my friends

Skip-Gram

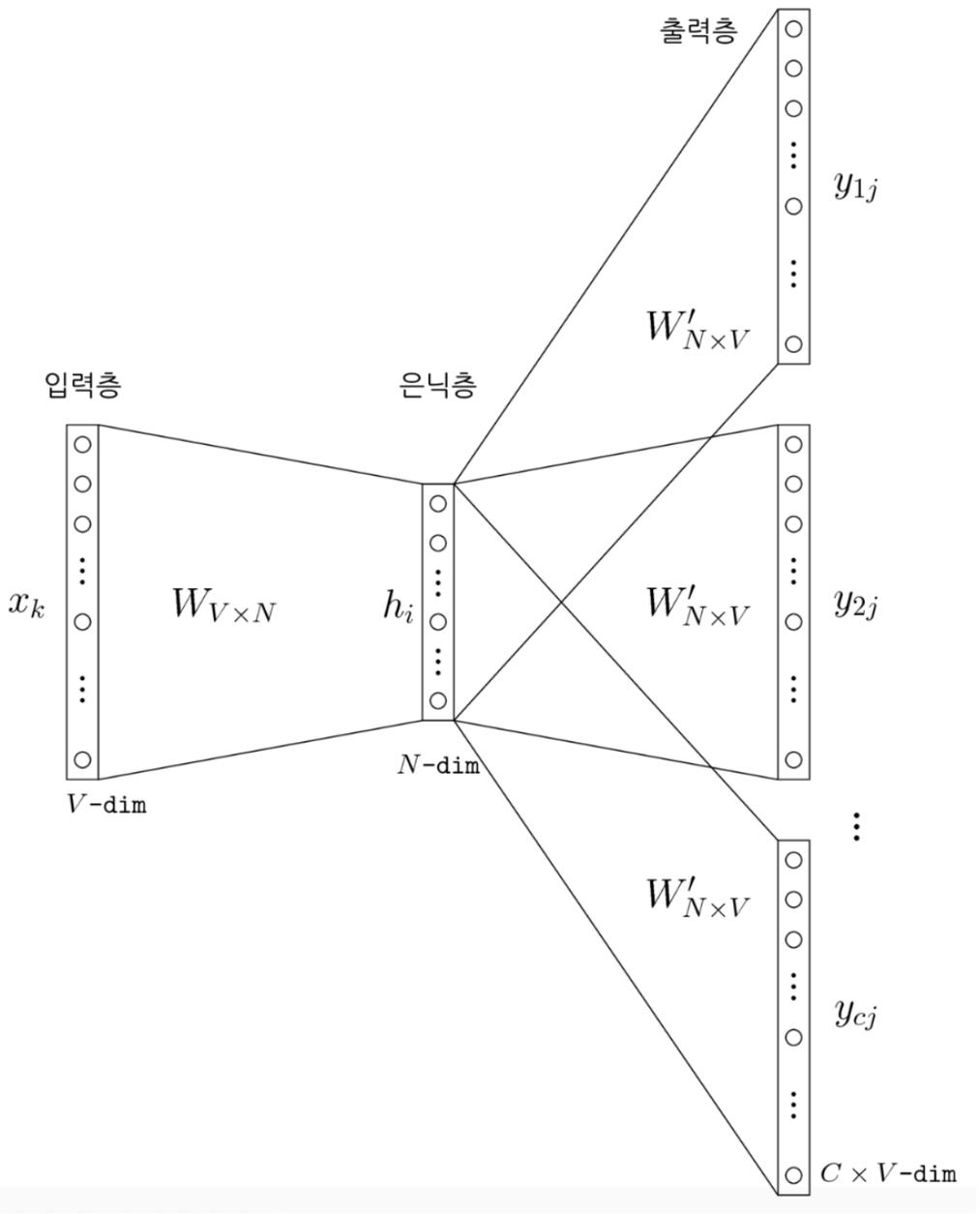
- 특정한 단어로부터 문맥이 될 수 있는 단어를 예측
- 입력 단어 주변의 k개 단어를 문맥으로 보고 예측 모형을 만드는데 이 k값을 window size라고 함

ex) the quick brown fox jumped over the lazy dog (k=1)

quick > the, brown

brown > quick, fox

Skip-Gram


출력층
y_{1j}
$W'_{N\times V}$
입력층
은닉층
x_k
$W_{V\times N}$
h_i
$W'_{N\times V}$
y_{2j}
N-dim
V-dim
$W'_{N\times V}$
y_{cj}
$C\times V$-dim