2023 분석 BASE Session

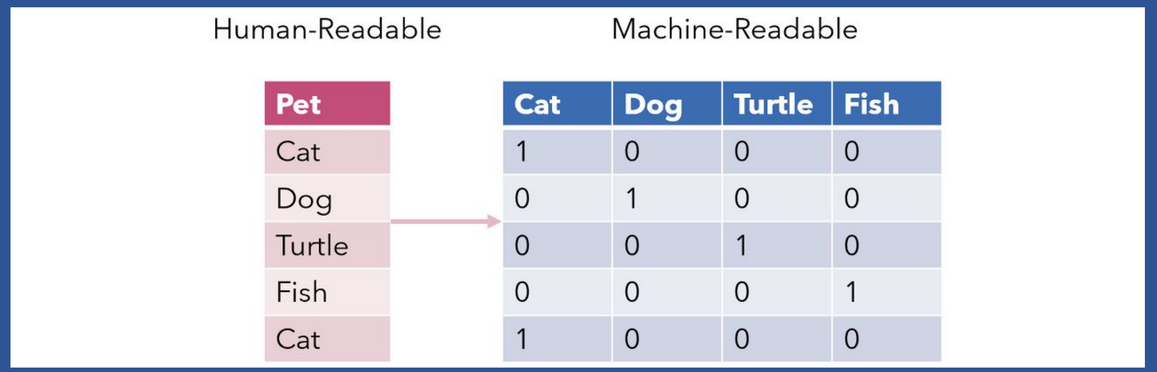
Word2Vec

20기 분석 정원준
20기 분석 최영우

CONTENTS

One-Hot encoding

- One-Hot encoding



Pet	Cat	Dog	Turtle	Fish
Cat	1	0	0	0
Dog	0	1	0	0
Turtle	0	0	1	0
Fish	0	0	0	1
Cat	1	0	0	0

- One-Hot encoding의 한계

1. 대용량 메모리 필요
   => 사전의 크기만큼 메모리가 필요함

2. 유사성 비교 불가능.
   => 코사인유사도를 구해야하는데 전부 0이나옴
   => 고양이 = 개 = 거북이 ?

Distributed Representation (분산표현)

- 단어를 문맥에 기반하여 표현하는 방법
- 비슷한 문맥에서 등장하는 단어는 비슷한 의미를 가질 것이라는 가정에서 출발

신종 **코로나** 바이러스 집단 감염증이 일상생활을 통해 지속 확산되고 있다.

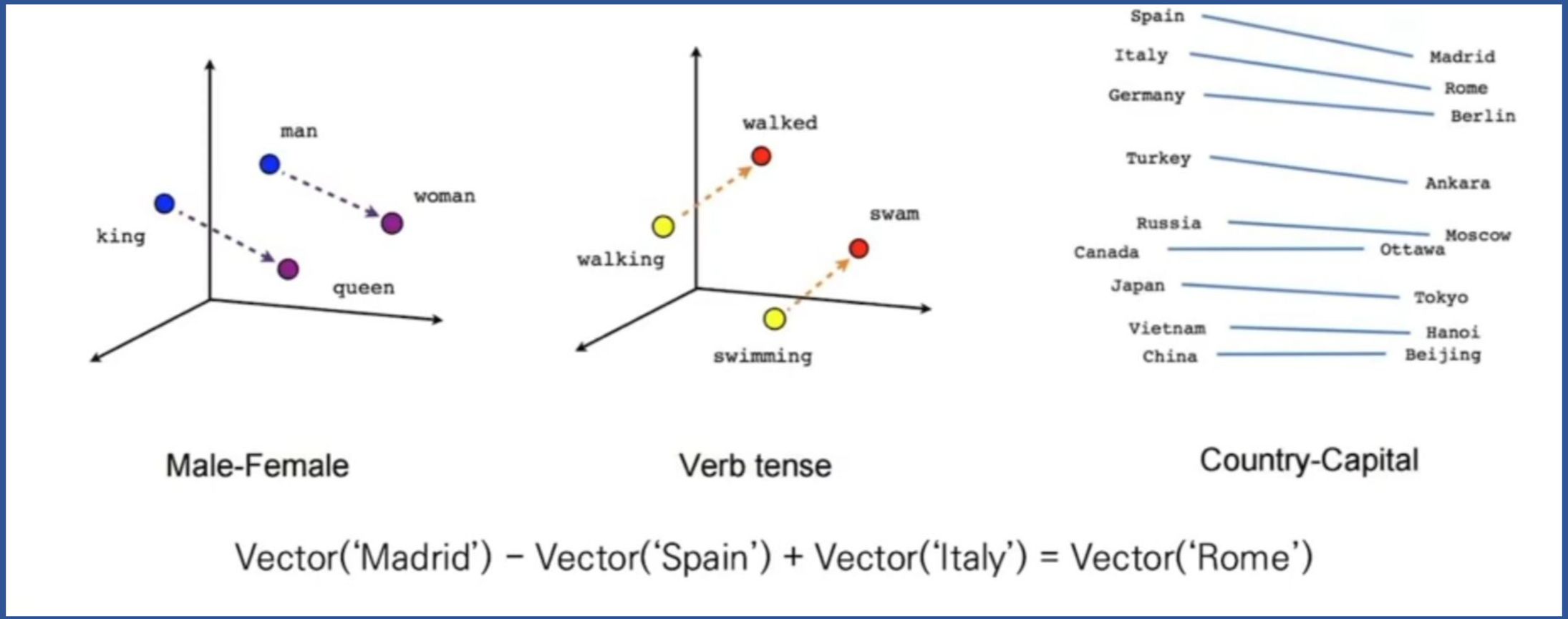
부산 수영구 댄스 동호회발 신종 **코로나** 바이러스 집단 감염이 목욕탕으로 퍼지는 등 확산되고 있다.

코로나? 신종, 바이러스, 집단, 감염, 확산, 일상, 생활, ..., 목욕탕

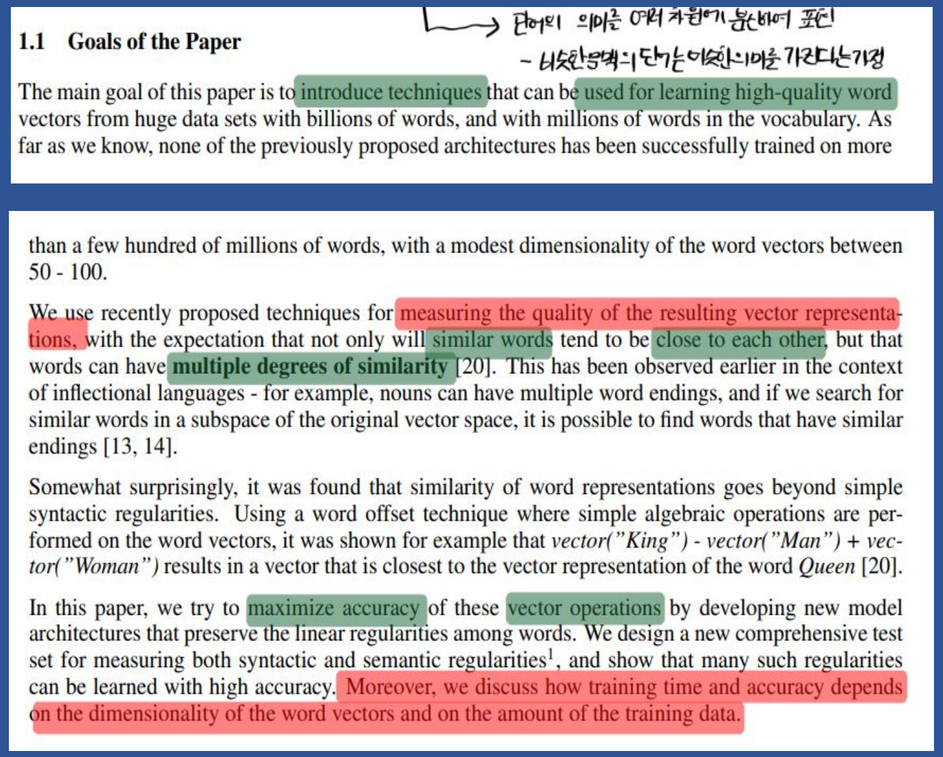
프랑스는 지금까지 북부도시 릴에서 2명의 **메르스** 바이러스 감염환자가 발생했다.

메르스? 바이러스, 감염, 프랑스, 지금, ..., 환자, 발생

Distributed Representation (분산표현)



Goals of Paper (목적!)



1.1 Goals of the Paper

The main goal of this paper is to introduce techniques that can be used for learning high-quality word vectors from huge data sets with billions of words, and with millions of words in the vocabulary. As far as we know, none of the previously proposed architectures has been successfully trained on more than a few hundred of millions of words, with a modest dimensionality of the word vectors between 50 - 100.

We use recently proposed techniques for measuring the quality of the resulting vector representations, with the expectation that not only will similar words tend to be close to each other, but that words can have **multiple degrees of similarity** [20]. This has been observed earlier in the context of inflectional languages - for example, nouns can have multiple word endings, and if we search for similar words in a subspace of the original vector space, it is possible to find words that have similar endings [13, 14].

Somewhat surprisingly, it was found that similarity of word representations goes beyond simple syntactic regularities. Using a word offset technique where simple algebraic operations are performed on the word vectors, it was shown for example that *vector("King") - vector("Man") + vector("Woman")* results in a vector that is closest to the vector representation of the word *Queen* [20].

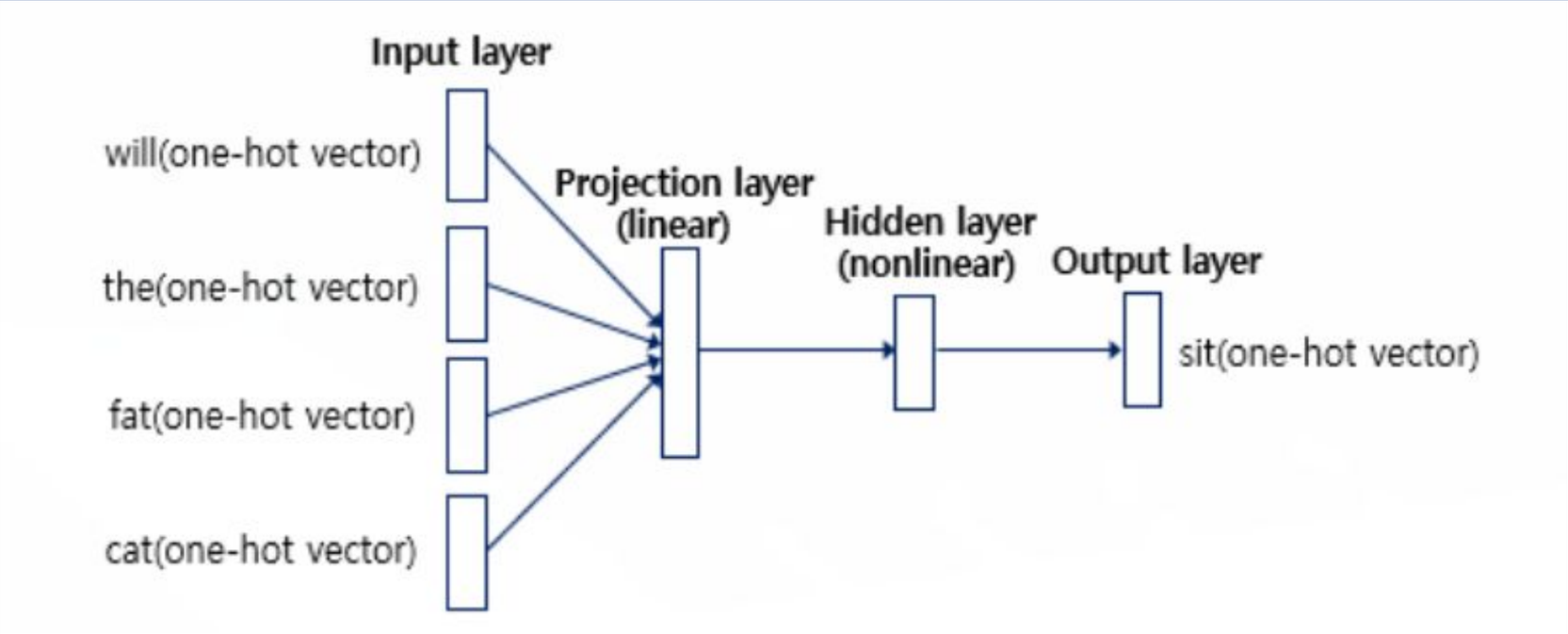
In this paper, we try to maximize accuracy of these vector operations by developing new model architectures that preserve the linear regularities among words. We design a new comprehensive test set for measuring both syntactic and semantic regularities[1], and show that many such regularities can be learned with high accuracy. Moreover, we discuss how training time and accuracy depends on the dimensionality of the word vectors and on the amount of the training data.

- 높은 수준의 단어 벡터를 학습 하는 기술을 소개

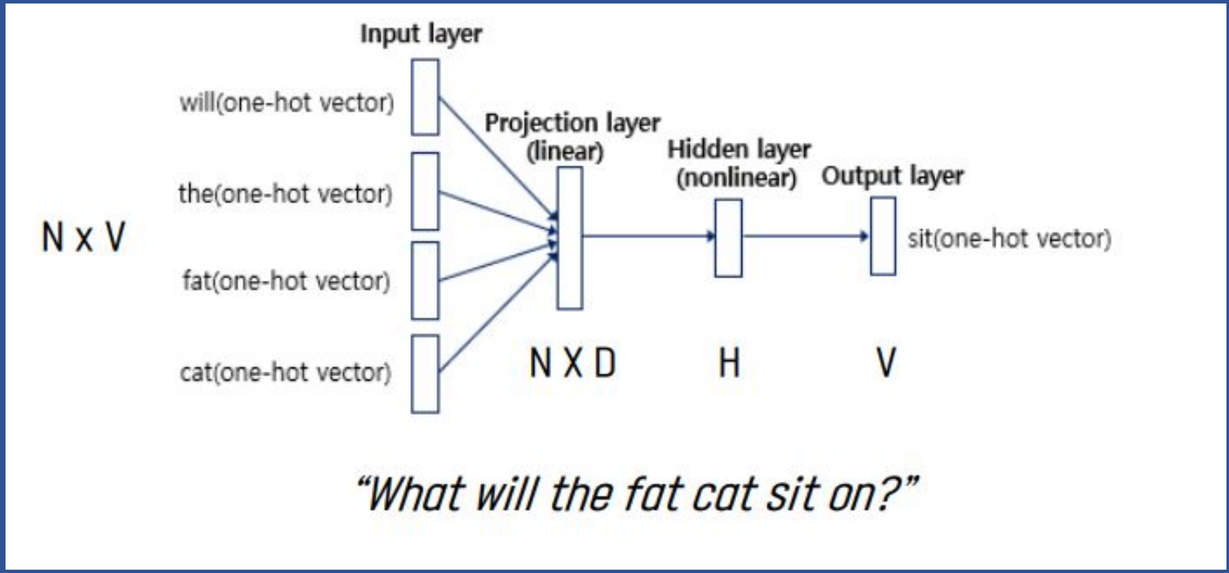
- 정확도 측정하기 위한 기술 제안

- 학습 시간과 정확도가 워드 벡터의 차원과 train data 양에 어떻게 의존하는지

NNLM (Neural Network Language Model)



NNLM (Neural Network Language Model)



N: input으로 들어가는 이전 단어의 개수
D: projection 후의 dimension
H: hidden layer size
V: Vocabulary size

N개의 단어로 N+1번째 단어를 예측

입력으로 원-핫 인코딩된 각각의 단어가 들어감

Input layer, Projection layer, Hidden layer, Output layer

Q = N x D + N x D x H + H x V(-> log2(V))

NNLM (Neural Network Language Model)

장점

단어 유사도 표현

희소문제를 해결함

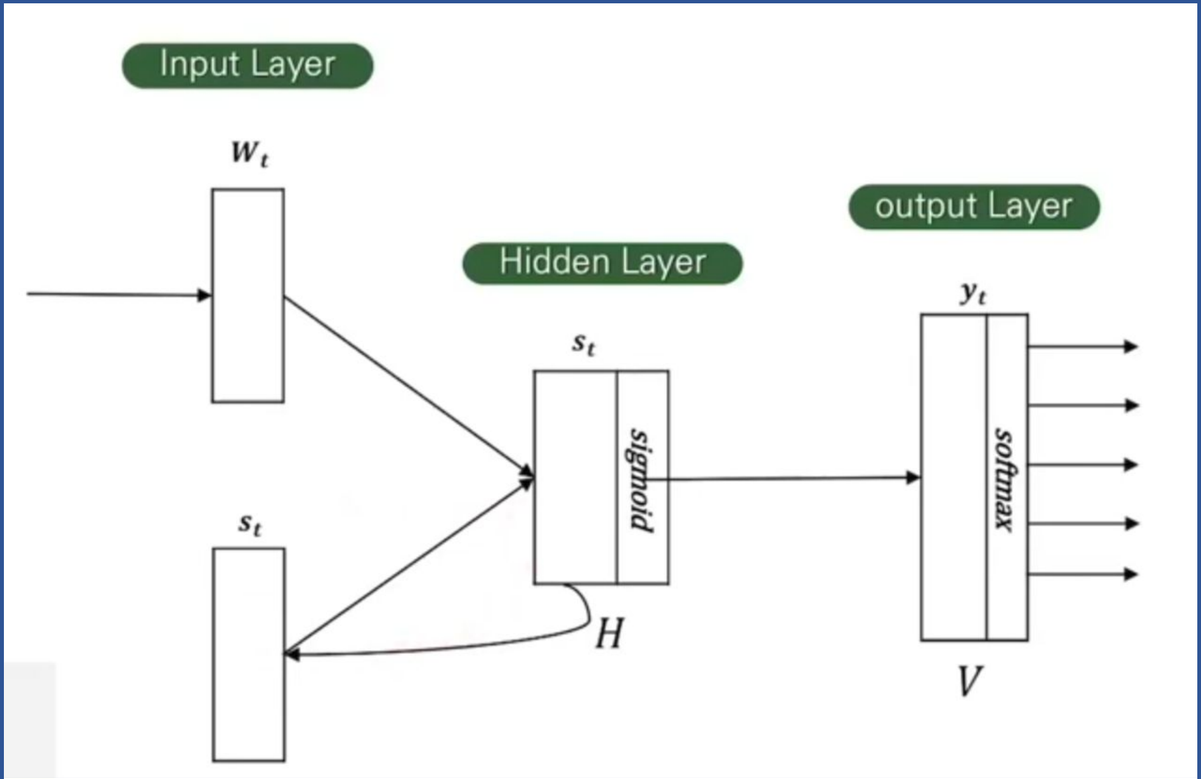
단점

입력 벡터수 n이 고정

이전 단어만 고려하고 이후에 단어는 고려하지 못함

높은 계산 복잡도

RNNLM (Recurrent Neural Network Language Model)

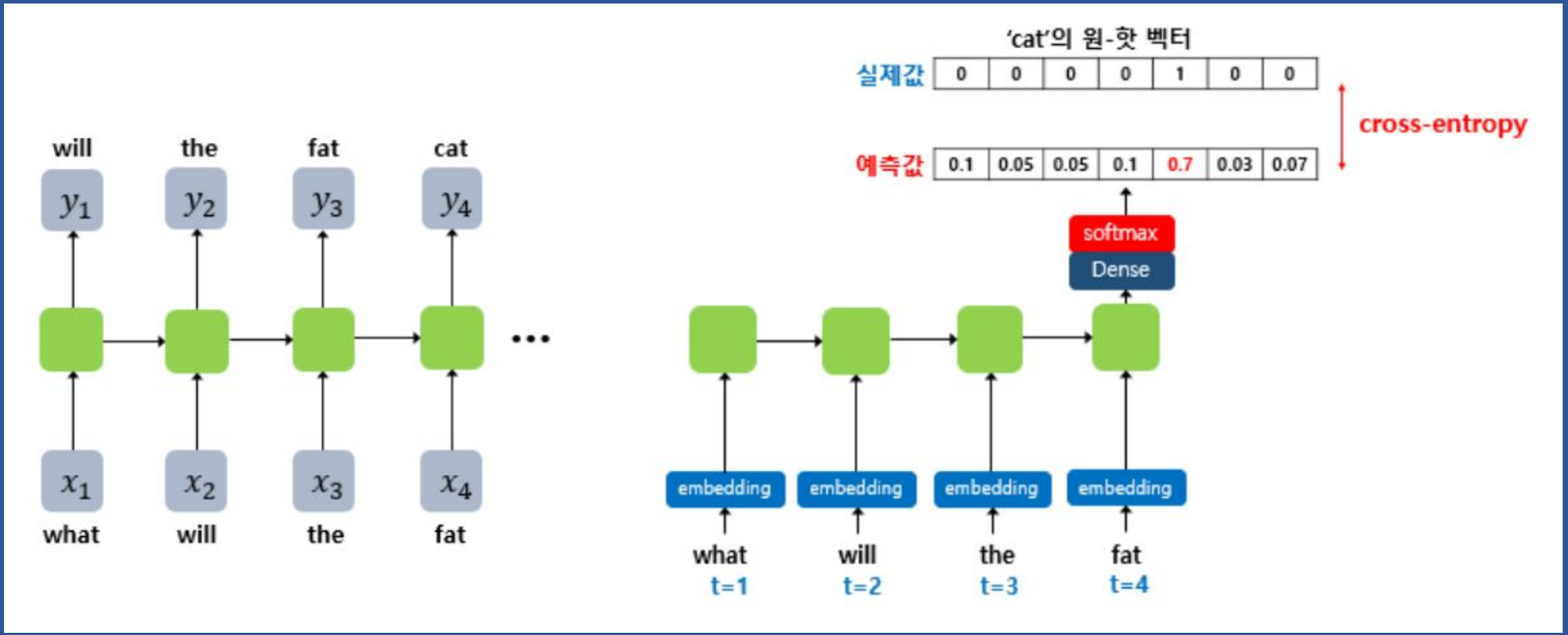


H: *hidden layer size*

V: *Vocabulary size*

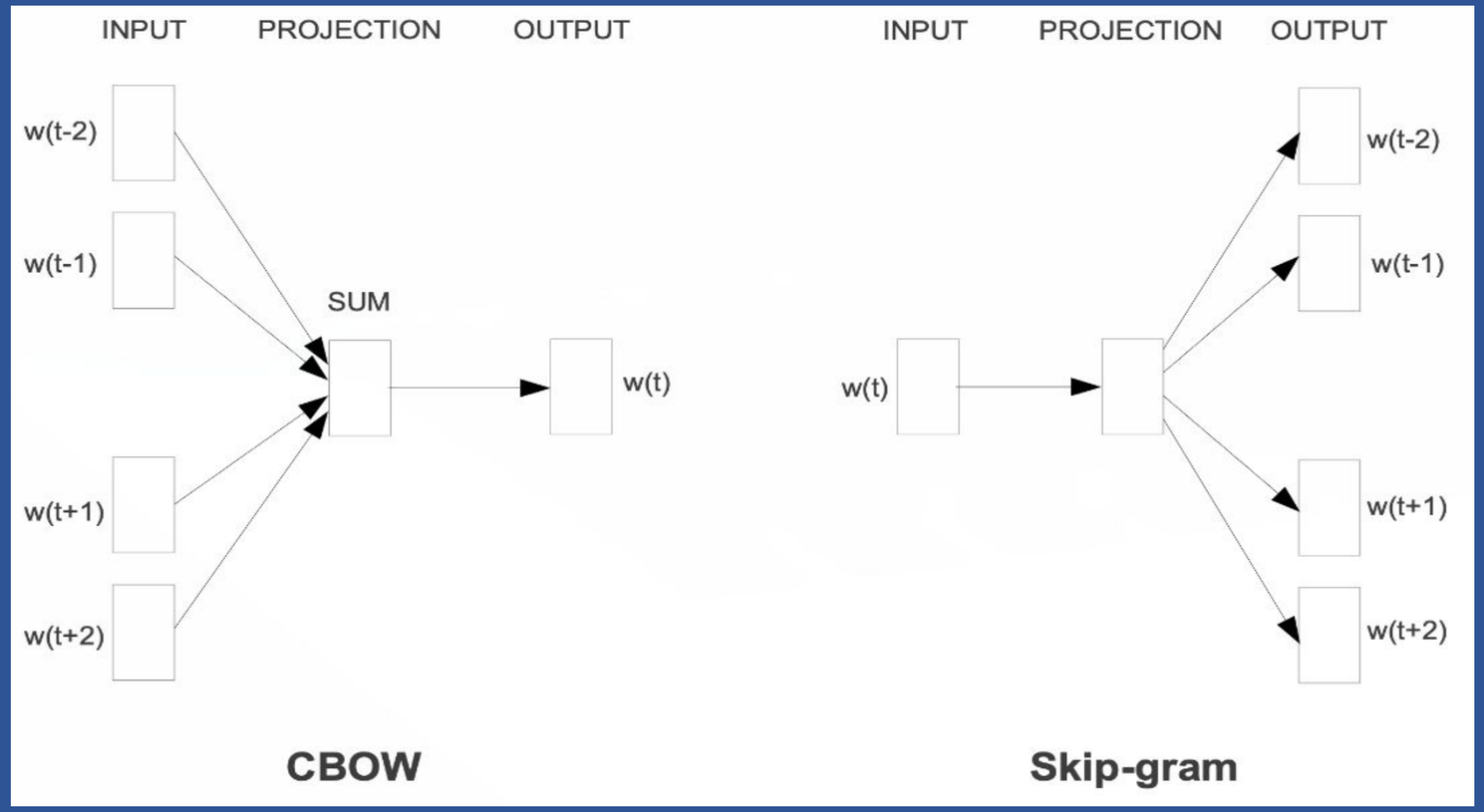
$$Q = H \times H + H \times V$$

RNNLM (Recurrent Neural Network Language Model)



단점
1. 이전 단어만 고려, 그 이후의 단어는 고려하지 못함
2. 높은 계산 복잡도

CBOW 와 Skip-gram



CBOW 와 Skip-gram

학습할 문장: “… 어머님 나 는 별 하나 에 아름다운 말 한마디 씩 불러 봅니다 …”

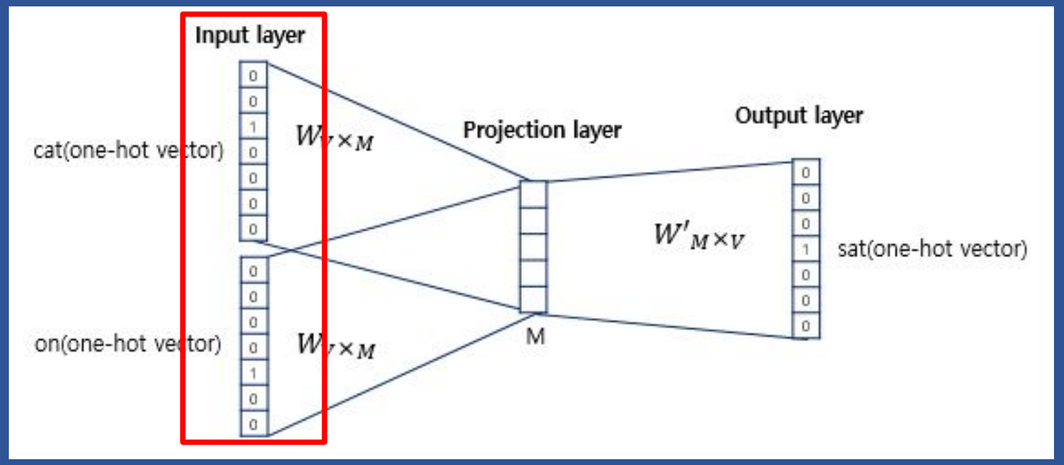
CBoW 를 사용하면 표시된 단어 정보를 바탕으로 아래의 [----] 에 들어갈 단어를 예측하는 과정으로 학습이 진행됩니다.

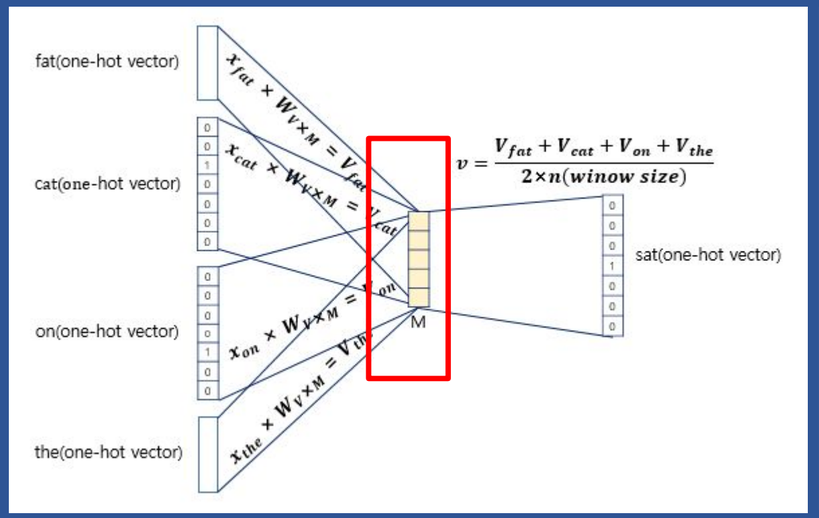
“… 나 는 [--] 하나 에 … “
“… 는 별 [----] 에 아름다운 …”
“… 별 하나 [--] 아름다운 말 …”
“… 하나 에 [--------] 말 한마디 …”

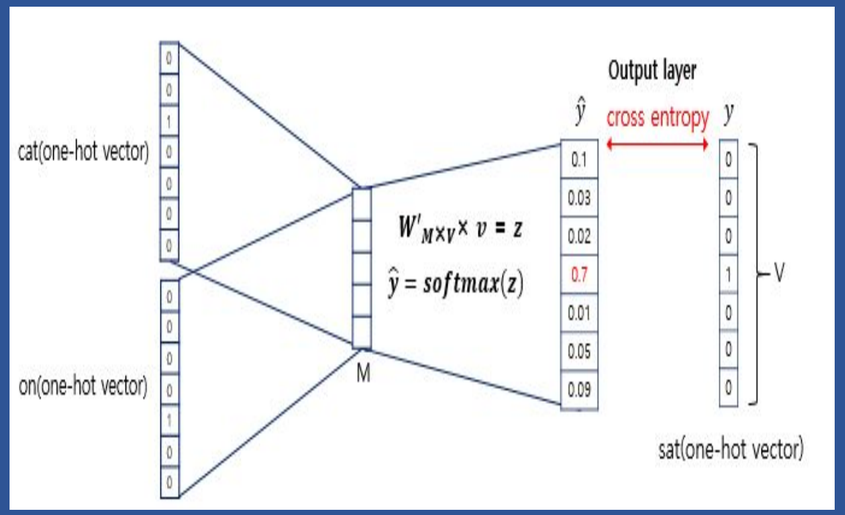
Skip-gram 을 사용하면 표시된 단어 정보를 바탕으로 다음의 [----] 에 들어갈 단어를 예측하는 과정으로 학습이 진행됩니다.

“… [--] [--] 별 [----] [--] …”
“… [--] [--] 하나 [--] [--------] …”
“… [--] [----] 에 [--------] [--] …”
“… [----] [--] 아름다운 [--] [------] …”

CBOW







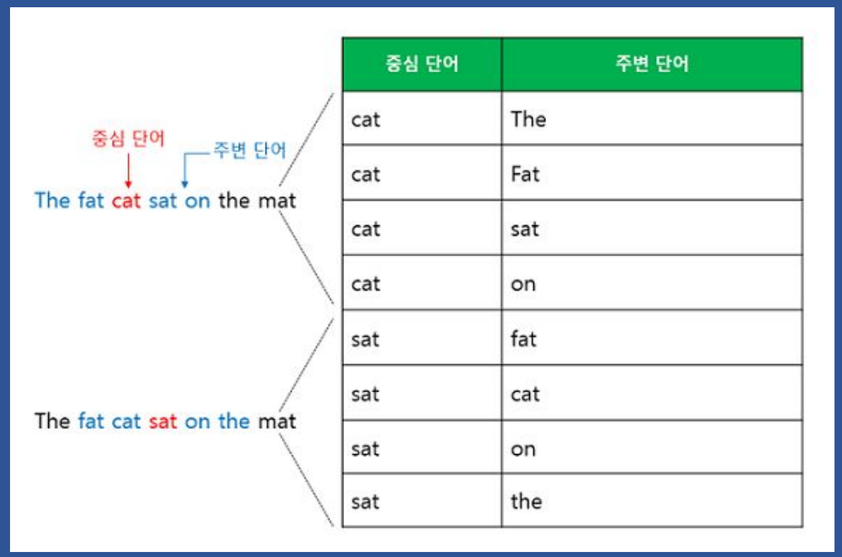
입력층

- 사용자가 정한 윈도우 크기 범위 안에 있는 주변 단어들의 원-핫 벡터가 들어감

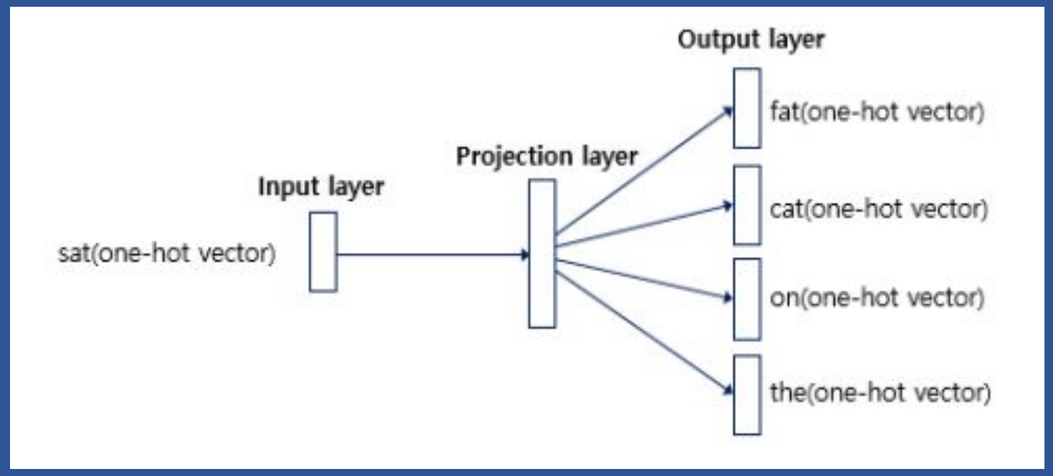
은닉층

- 은닉층이 1개인 얕은 신경망
- M은 임베딩하고 난 벡터의 차원을 의미함
- 첫번째 가중치 행렬과 곱해지고 평균을 구해서 벡터들의 평균을 만들어냄
- 이후에 두번째 가중치 행렬과 만나서 입력층의 차원(V)과 똑같은 크기의 벡터를 출력함
- 역전파 과정을 진행한 후에 첫번째 가중치 W나 두번째 가중치 W'의 행을 각 단어의 임베딩 벡터로 사용

Skip-gram



중심 단어	주변 단어
cat	The
cat	Fat
cat	sat
cat	on
sat	fat
sat	cat
sat	on
sat	the



- 중심단어만 입력됨 (one-hot)
- Projection layer에서 평균 구할 필요 없음

복잡도

CBow

$Q = N \times D + D \times \log_2(V)$.

- 현재 N개의 단어를 D차원으로 projection
- 이후 D차원에서 V에 대한 각각의 확률필요

Skip-gram

$Q = C \times (D + D \times \log_2(V))$

- 한개의 단어만 project이니 D
- 이후 D차원에서 V에 대한 각각의 확률 필요
- Output은 C개이니 C를 곱한다.

CBOW vs Skip-gram

CBOW

Input	Output
fat, cat	The
The, cat, sat	fat
The, fat, sat, on	cat
fat, cat, on, the	sat
cat, sat, the, table	on

Word	Count
The	1
fat	1
cat	1
sat	1
on	1

Skip-gram

Input	Output
The	fat, cat
fat	The, cat, sat
cat	The, fat, sat, on
sat	fat, cat, on, the
on	cat, sat, the, table

Word	Count
The	2
fat	3
cat	4
sat	4
on	4

결론: Skip-gram이 cbow에 비해서 더 많은 문맥을 고려해서 성능에서 더 좋다

정확도 측정

Type of relationship	Word Pair 1		Word Pair 2	
Common capital city	Athens	Greece	Oslo	Norway
All capital cities	Astana	Kazakhstan	Harare	Zimbabwe
Currency	Angola	kwanza	Iran	rial
City-in-state	Chicago	Illinois	Stockton	California
Man-Woman	brother	sister	grandson	granddaughter
Adjective to adverb	apparent	apparently	rapid	rapidly
Opposite	possibly	impossibly	ethical	unethical
Comparative	great	greater	tough	tougher
Superlative	easy	easiest	lucky	luckiest
Present Participle	think	thinking	read	reading
Nationality adjective	Switzerland	Swiss	Cambodia	Cambodian
Past tense	walking	walked	swimming	swam
Plural nouns	mouse	mice	dollar	dollars
Plural verbs	work	works	speak	speaks

Word Pair2에 오른쪽 열이 답변이 되는 열

해당 답변이 질문과 같을 때를 정확하게 된 것이라고 판단

ex) Athens - Greece + Oslo = Norway

해석) 아테네 - 그리스 + 오슬로 = 노르웨이

결과 (Results)

Dimensionality / Training words	24M	49M	98M	196M	391M	783M
50	13.4	15.7	18.6	19.1	22.5	23.2
100	19.4	23.1	27.8	28.7	33.4	32.2
300	23.2	29.2	35.3	38.6	43.7	45.9
600	24.0	30.1	36.5	40.8	46.6	50.4

훈련데이터와 **벡터 차원**의 크기를 증가시키면서 cbow를 테스트

- 차원과 훈련데이터를 늘리는 것이 높은 정확도를 보임
- but 시간복잡도 역시 증가함을 유의

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

모델 구조에 따른 테스트 결과

- NNLM은 projection layer를 거쳐서 hidden layer 거치고 RNNLM은 비선형 hidden layer에 들어가기 때문

- CBOW SKIP-GRAM 모두 기존의 모델보다 높은 성능을 보임

- 의미적에서는 SKIP-GRAM 문법적에서는 CBOW

결론

- 인기있는 모델인 신경망 모델과 비교했을 때 매우 간단한 모델 구조를 사용하여 얻은 단어 벡터가 좋은 성능을 가질 수 있음을 확인

- 거대한 데이터 셋이 있다면 매우 적은 계산 복잡도로 높은 차원의 단어 벡터를 정확하게 구할 수 있다.

- High-quality word vector가 NLP task의 매우 중요한 요소가 될 것이다

BOAZ

예시 사용 코드


```python
import numpy as np
from gensim.models import Word2Vec, KeyedVectors

model = Word2Vec.load('/gdrive/MyDrive/colab/text_rep/ko/ko.bin')

# 한국, 서울, 도쿄  각각의 벡터 출력
print([i for i in model.wv.most_similar("한국")[:3]])
print([i for i in model.wv.most_similar("서울")[:3]])
print([i for i in model.wv.most_similar("도쿄")[:3]])
print()

# 한국 - 서울 + 파리
new_word = model.wv['한국'] - model.wv['서울'] + model.wv['도쿄']
# 유사한 단어 가져오기
ms_word = model.wv.most_similar([new_word])
# 상위 3개 결과 출력
most3_similar = [i[0] for i in ms_word[:3]]
print("한국 - 서울 + 도쿄 =", most3_similar, "₩n")    # ['파리', '한국', '일본']
print()
```

```
[('대한민국', 0.6662065982818604), ('우리나라', 0.6312342882156372), ('동양', 0.5589770078659058)]
[('서울시', 0.6911441087722778), ('서울특별시', 0.6897671818733215), ('성남', 0.6770075559616089)]
[('오사카', 0.7090513110160828), ('도카이', 0.6812214851379395), ('홋카이', 0.6750696301460266)]

한국 - 서울 + 도쿄 = ['도쿄', '일본', '한국']
```

BOAZ

04 결론

예시 사용 코드


```
# 컴퓨터, 공학, 인문학  각각의 벡터 출력
print([i for i in model.wv.most_similar("컴퓨터")[:3]])
print([i for i in model.wv.most_similar("공학")[:3]])
print([i for i in model.wv.most_similar("인문학")[:3]])
print()

# 컴퓨터 + 공학 + 인문학
new_word = model.wv['컴퓨터'] + model.wv['공학'] + model.wv['인문학']
# 유사한 단어 가져오기
ms_word = model.wv.most_similar([new_word])
# 상위 3개 결과 출력
most3_similar = [i[0] for i in ms_word[:3]]
print("컴퓨터 + 공학 + 인문학 =", most3_similar, "₩n")  # ['공학', '컴퓨터', '인공지능']
print()


# 인간, 사랑, 인문학  각각의 벡터 출력
print([i for i in model.wv.most_similar("인간")[:3]])
print([i for i in model.wv.most_similar("사랑")[:3]])
print()

# 인간 - 사랑
new_word = model.wv['인간'] - model.wv['사랑']
# 유사한 단어 가져오기
ms_word = model.wv.most_similar([new_word])
# 상위 3개 결과 출력
most3_similar = [i[0] for i in ms_word[:3]]
print("인간 - 사랑 =", most3_similar, "₩n") # ['인간', '내재적', '유기체']
```

```
[('하드웨어', 0.7727354764938354), ('소프트웨어', 0.7308895587921143), ('마이크로프로세서', 0.7205492258071899)]
[('화학', 0.6704168319702148), ('공학부', 0.6422457695007324), ('물리학', 0.637071967124939)]
[('교육학', 0.8108000755310059), ('사회학', 0.7838379740715027), ('사회과학', 0.7743333578109741)]

컴퓨터 + 공학 + 인문학 = ['공학', '컴퓨터', '인공지능']


[('본능', 0.673734188079834), ('자아', 0.6703100204467773), ('본질', 0.6514253616333008)]
[('슬픔', 0.7216662764549255), ('행복', 0.6759077310562134), ('절망', 0.6468985080718994)]

인간 - 사랑 = ['인간', '내재적', '유기체']
```

출처

1. https://github.com/Kyubyong/wordvectors
2. https://heytech.tistory.com/353
3. https://wikidocs.net/45609
4. https://www.youtube.com/watch?v=sidPSG-EVDo&list=LL&inde=1
5. https://arxiv.org/abs/1301.3781

2023 논문발제

감사합니다

Word2Vec

20기 분석 정원준
20기 분석 최영우