Word2Vec 논문 리뷰

(Efficient Estimation of Word Representations in Vector Space, 2013)

집현전 초급반 이영빈

(dudqls4019@gmail.com)

CONTENTS

01 논문의 개요

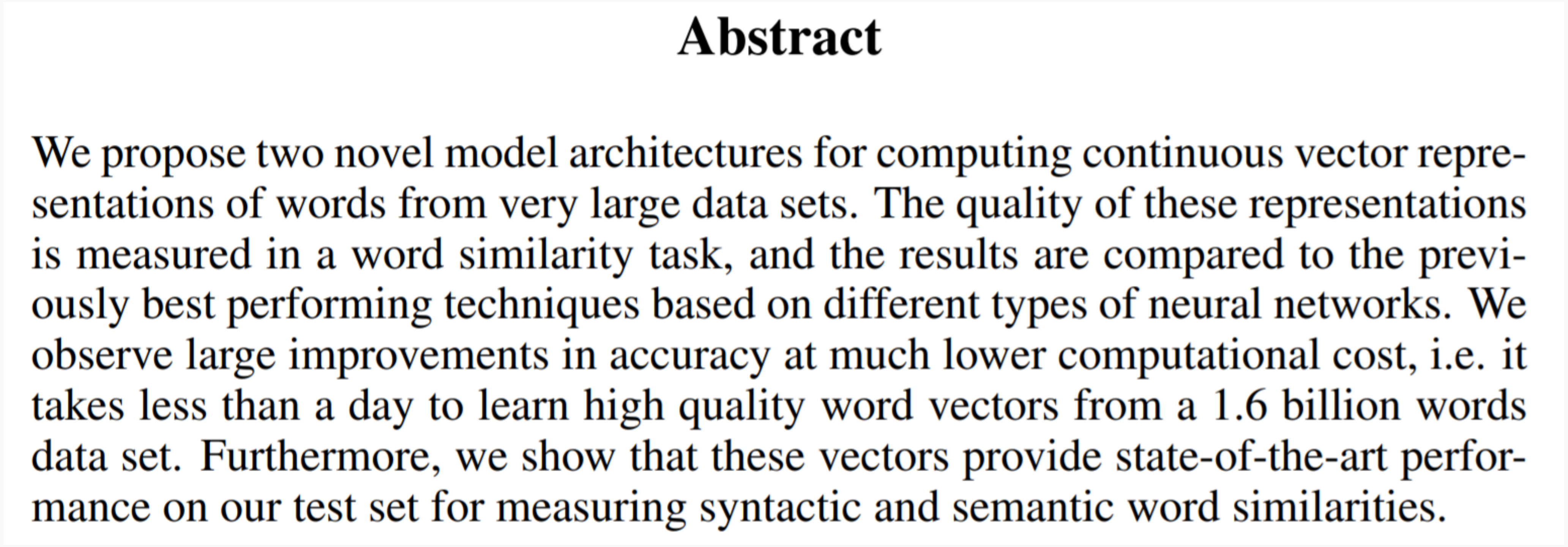
논문 저자 – Tomas Mikolov

Tomas Mikolov

前 구글 연구원(2013)
前 페이스북 research scientist(2014 – 2020)
現 체코 정보 과학 기술원 (CIIRC CTU Prague) 연구원
주요 논문
Distributed representations of words and phrases and their compositionality(2013)
Efficient estimation of word representations in vector space(2013)

주요 수상
Neuron Research Discovery Award in the Field of Computer Science (2018)

01 논문의 개요
Abstract

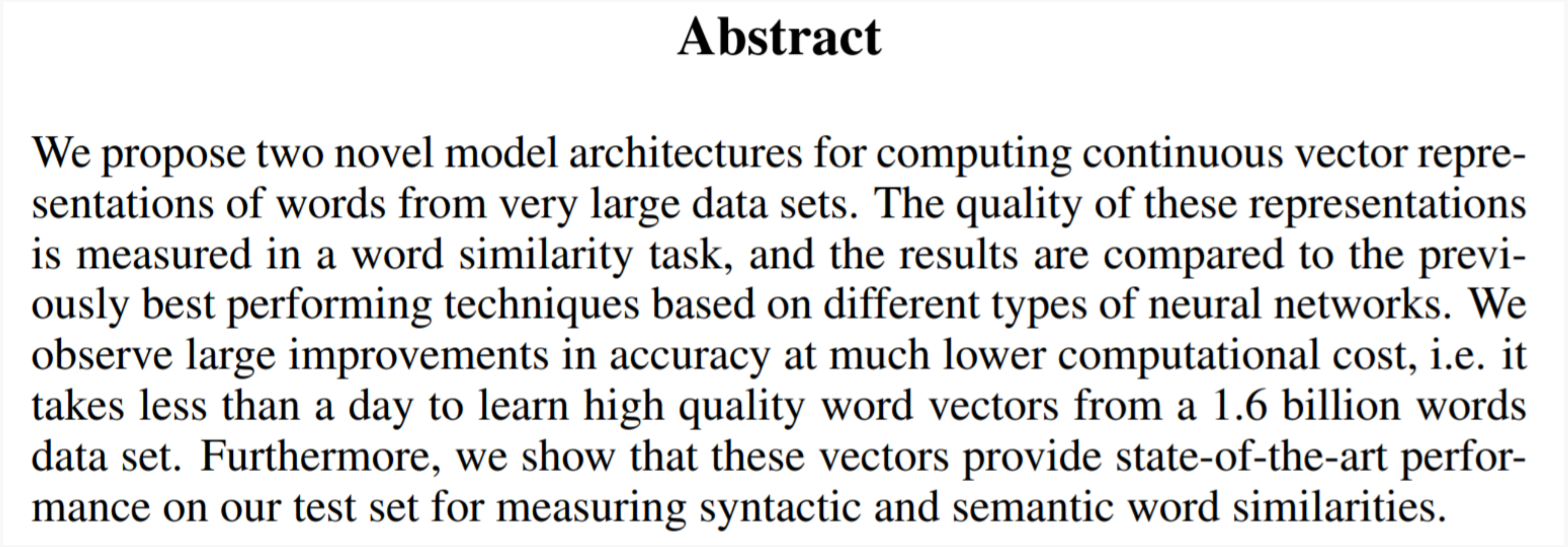


Abstract

We propose two novel model architectures for computing continuous vector representations of words from very large data sets. The quality of these representations is measured in a word similarity task, and the results are compared to the previously best performing techniques based on different types of neural networks. We observe large improvements in accuracy at much lower computational cost, i.e. it takes less than a day to learn high quality word vectors from a 1.6 billion words data set. Furthermore, we show that these vectors provide state-of-the-art performance on our test set for measuring syntactic and semantic word similarities.

데이터셋으로부터 단어들의 연속적인 벡터 표현을 계산하는 2가지 새로운 모델 설계했다.

01 논문의 개요
Abstract

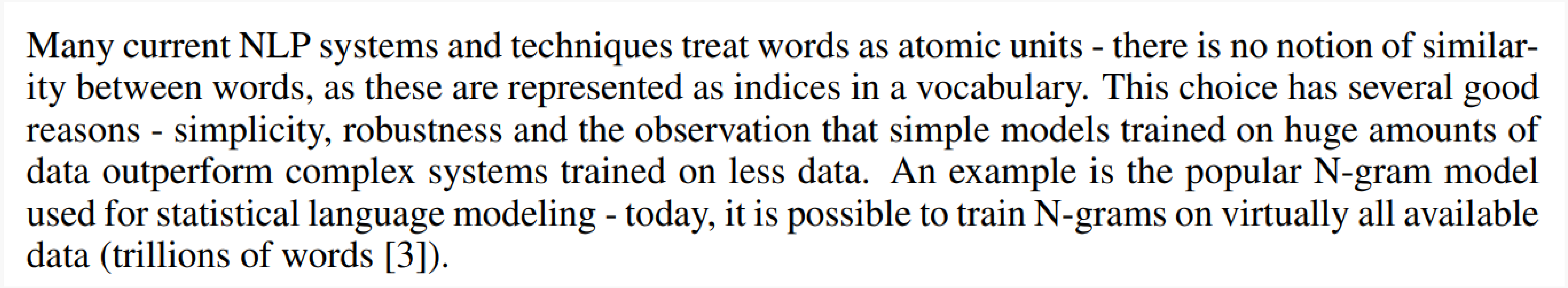


Abstract

We propose two novel model architectures for computing continuous vector representations of words from very large data sets. The quality of these representations is measured in a word similarity task, and the results are compared to the previously best performing techniques based on different types of neural networks. We observe large improvements in accuracy at much lower computational cost, i.e. it takes less than a day to learn high quality word vectors from a 1.6 billion words data set. Furthermore, we show that these vectors provide state-of-the-art performance on our test set for measuring syntactic and semantic word similarities.

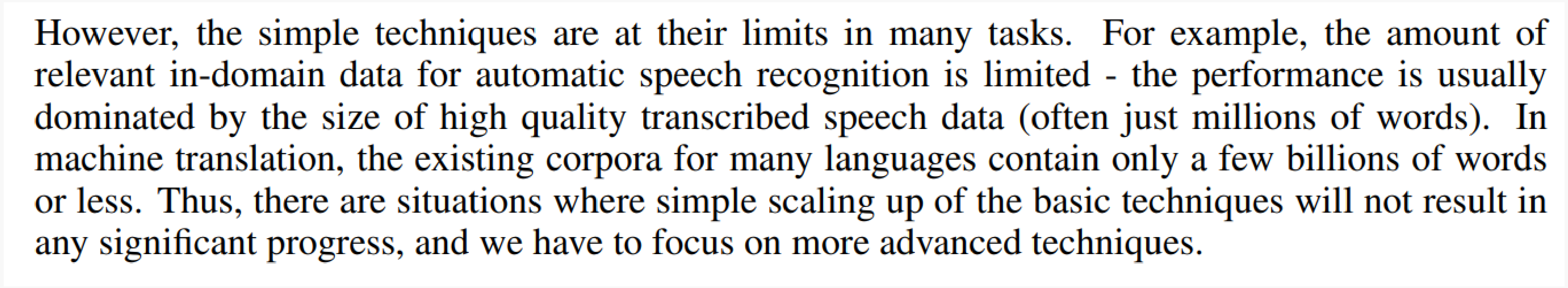
이 모델들은 훨씬 적은 비용으로 정확도를 더 높일 수 있다.

02 논문 Introduction
Introduction


Many current NLP systems and techniques treat words as atomic units - there is no notion of similarity between words, as these are represented as indices in a vocabulary. This choice has several good reasons - simplicity, robustness and the observation that simple models trained on huge amounts of data outperform complex systems trained on less data. An example is the popular N-gram model used for statistical language modeling - today, it is possible to train N-grams on virtually all available data (trillions of words [3]).

많은 NLP시스템에서 단어를 하나의 원자적 unit(단어들 사이에 유사성에 대한 개념이 없다)을 보았다.
대표적인 모델로 통계적 언어 모델링에 사용되는 N-gram 모델이 있다.

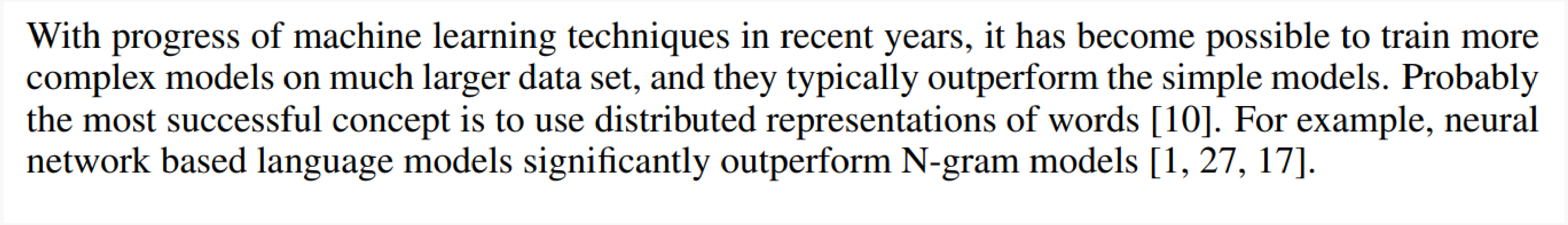
02 논문 Introduction
Introduction


However, the simple techniques are at their limits in many tasks. For example, the amount of relevant in-domain data for automatic speech recognition is limited - the performance is usually dominated by the size of high quality transcribed speech data (often just millions of words). In machine translation, the existing corpora for many languages contain only a few billions of words or less. Thus, there are situations where simple scaling up of the basic techniques will not result in any significant progress, and we have to focus on more advanced techniques.

그러나 간단한 technique은 데이터 양이 제한될 때 문제가 생긴다.
또한 단순한 모델을 크게 만든다고 결과에 영향을 미치지 않는다. => 더 진보된 기술이 필요

02 논문 Introduction

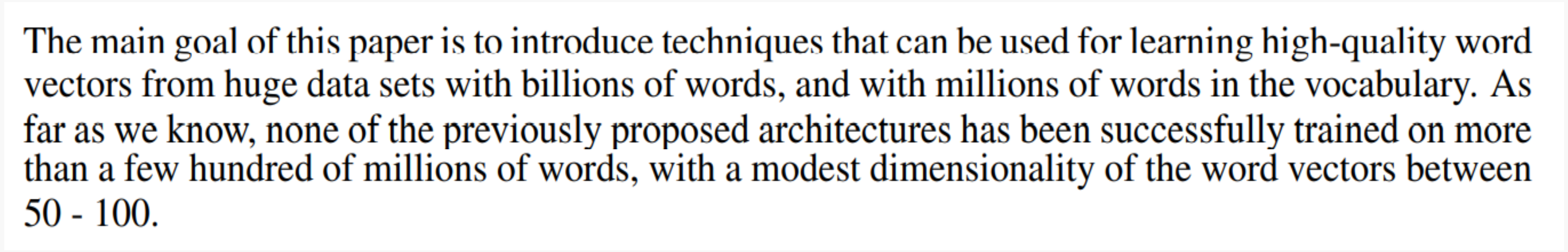
Introduction


With progress of machine learning techniques in recent years, it has become possible to train more complex models on much larger data set, and they typically outperform the simple models. Probably the most successful concept is to use distributed representations of words [10]. For example, neural network based language models significantly outperform N-gram models [1, 27, 17].

최근 기술 발전으로 인해
복잡한 모델(neural network 모델)이 단순한 모델(N-gram 모델)보다 성능이 더 좋아졌다.

02 논문 Introduction

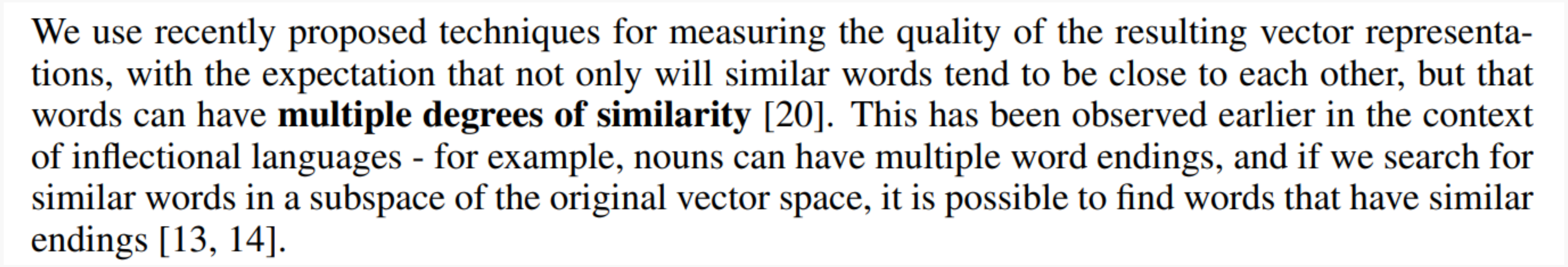
Goals of paper


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.

논문의 목표는 수십억 개의 단어들과 수백만개의 단어들로 구성된 데이터셋으로부터 높은 퀄리티의 단어벡터들을 학습시키는데 사용되는 기술을 소개하는 것이다.

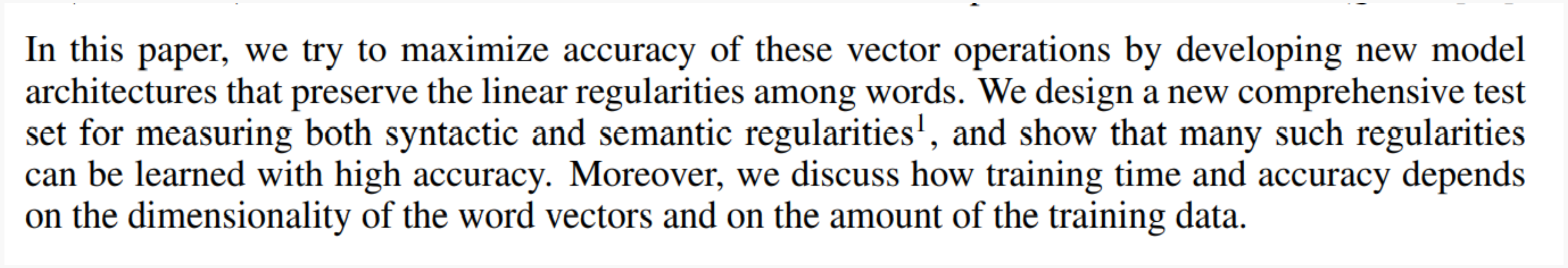
02 논문 Introduction

Goals of paper

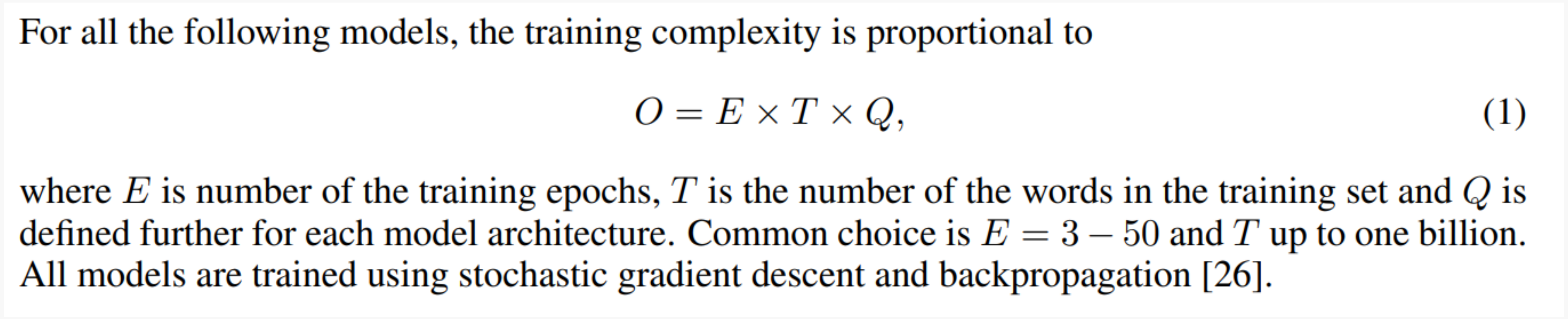

We use recently proposed techniques for measuring the quality of the resulting vector representations, with the expectation that not only will similar words tend to be close to each other, but that words can have **multiple degrees of similarity** [20]. This has been observed earlier in the context of 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].

우리가 제안할 기술은 비슷한 단어들이 서로 가까이 있다는 것처럼 보일 뿐더러 단어들이 유사도를 갖고 있다는 예측으로 벡터 representation의 품질을 측정할 수 있다.

02 논문 Introduction
Goals of paper


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.

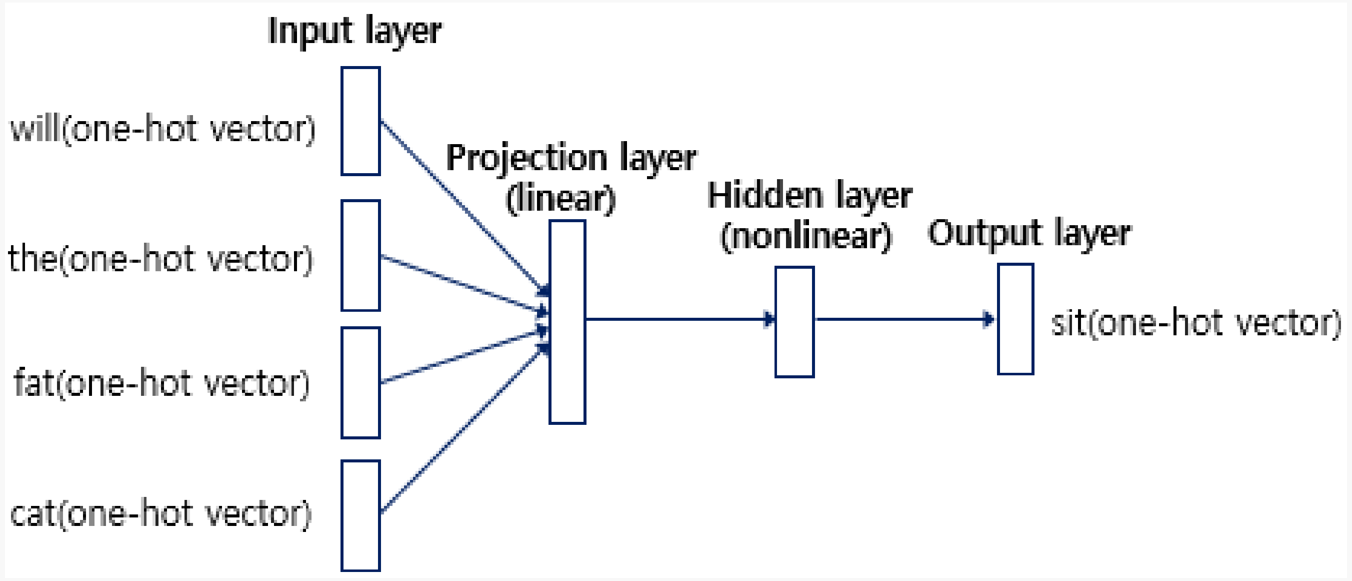
이번 논문에서 저자들은 단어들 사이에 선형적 규칙을 보존하는 새로운 모델을 개발해 이러한 벡터들의 정확도를 극대화하려고 할 것이다. 이걸 측정하기 위해 의미론, 문법적 유사성을 동시에 측정하는 test set을 디자인했다.


For all the following models, the training complexity is proportional to

$$O = E \times T \times Q, \tag{1}$$

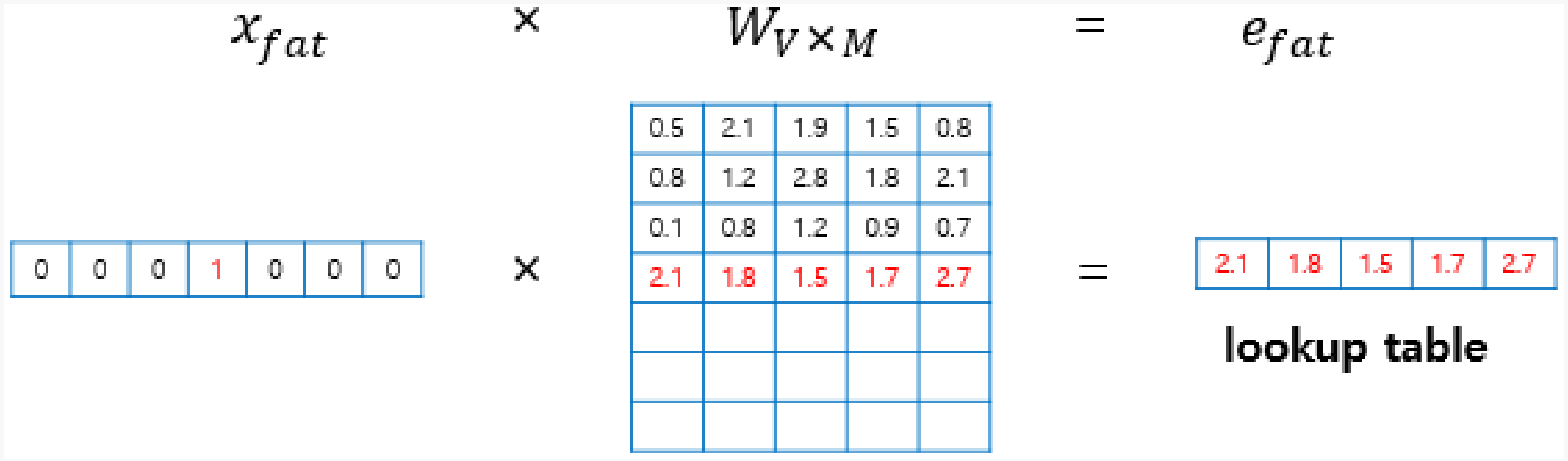
where E is number of the training epochs, T is the number of the words in the training set and Q is defined further for each model architecture. Common choice is $E = 3 - 50$ and T up to one billion. All models are trained using stochastic gradient descent and backpropagation [26].

저자들은 training 복잡도를 다음과 같이 설정했다.
O = E(epoch) x T(training set에 있는 단어들의 수) x Q(각 모델 설계에서 정의)



출처 : 9) 피드 포워드 신경망 언어 모델(Neural Network Language Model, NNLM) - 딥 러닝을 이용한 자연어 처리 입문 (wikidocs.net)

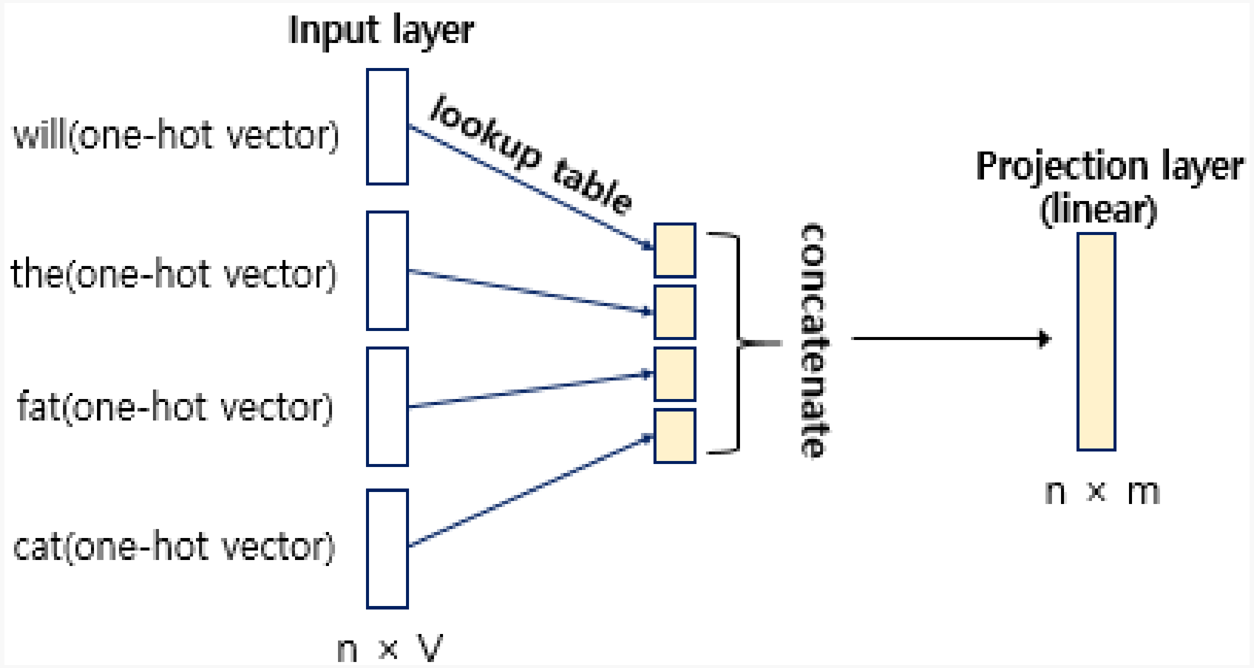
Feed Forward Neural Net Language Model



출처 : 9) 피드 포워드 신경망 언어 모델(Neural Network Language Model, NNLM) - 딥 러닝을 이용한 자연어 처리 입문 (wikidocs.net)

one-hot vector는 W라는 가중치 행렬과 곱해지고 나온 결과값은 차원의 수가 줄어든다.
W는 V(단어 집합의 크기) x M(투사층의 크기)로 이루어진 가중치 행렬이다.

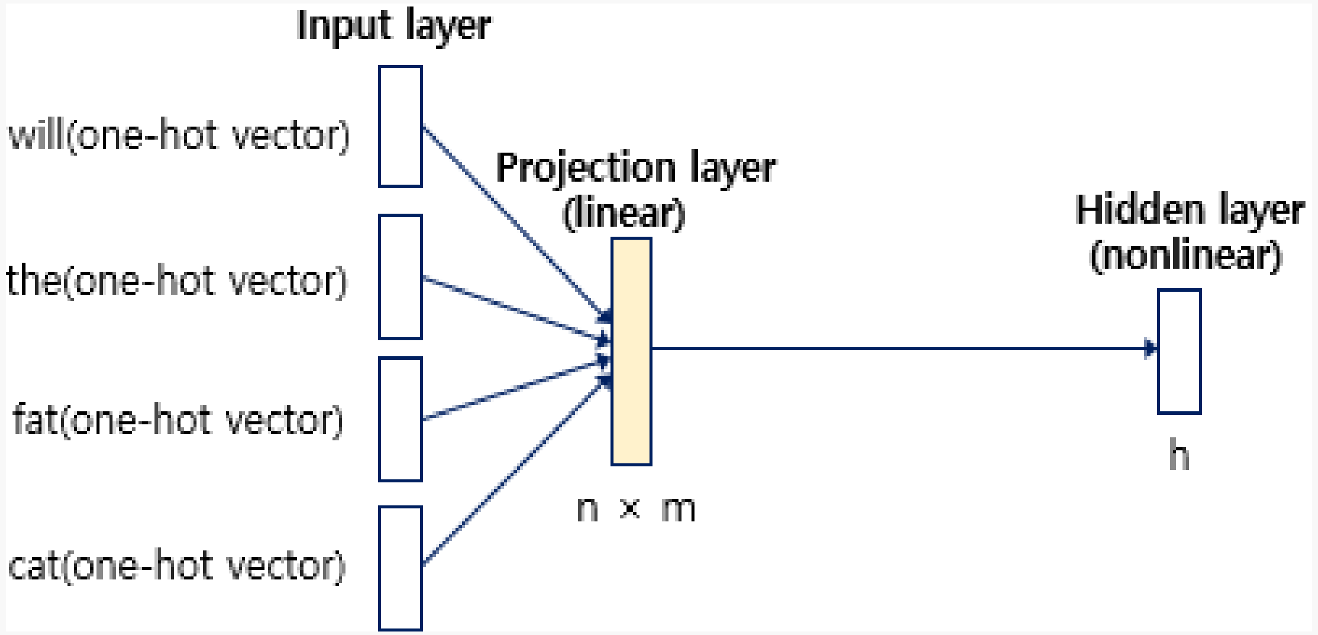
02 논문 Introduction
Model Architecture



출처 : 9) 피드 포워드 신경망 언어 모델(Neural Network Language Model, NNLM) - 딥 러닝을 이용한 자연어 처리 입문 (wikidocs.net)

Lookup table을 거치게 되면 one-hot vector는 차원이 더 작은 임베딩 벡터로 변경되고 모든 임베딩값을 concatenate한다. 이를 projection layer라 부른다.

02 논문 Introduction
Model Architecture

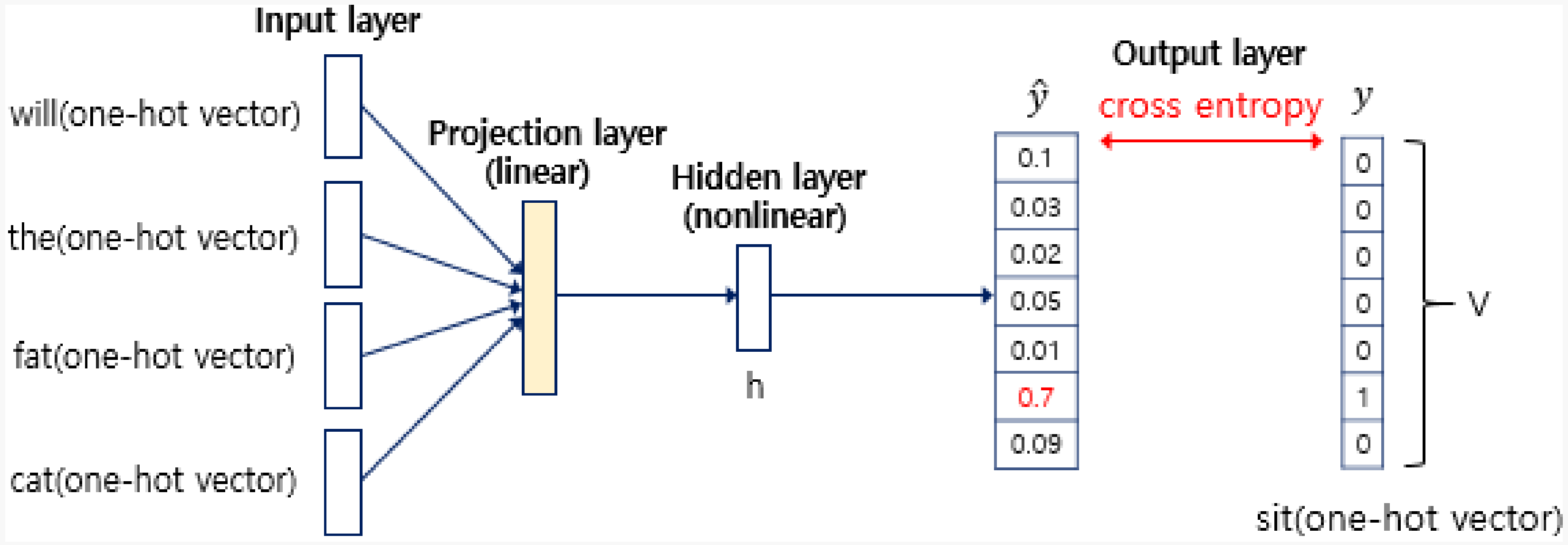


출처 : 9) 피드 포워드 신경망 언어 모델(Neural Network Language Model, NNLM) - 딥 러닝을 이용한 자연어 처리 입문 (wikidocs.net)

이후 h의 크기를 가지고 있는 은닉층을 지나는데 이때 활성화 함수가 입력된다.

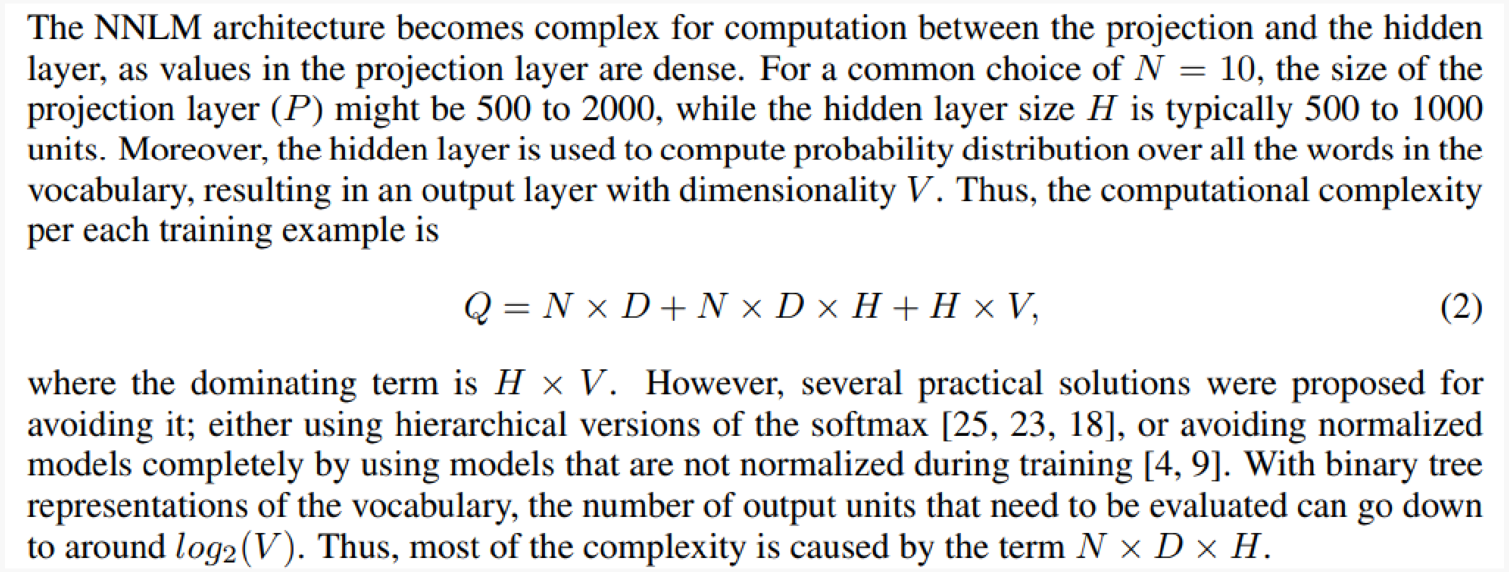
논문 Introduction

Model Architecture



출처 : 9) 피드 포워드 신경망 언어 모델(Neural Network Language Model, NNLM) - 딥 러닝을 이용한 자연어 처리 입문 (wikidocs.net)

출력층에서 활성화 함수로 Softmax를 사용한다. 이후 one-hot vector로 바꿔주면 다음과 같은 결론이 나온다.



The NNLM architecture becomes complex for computation between the projection and the hidden layer, as values in the projection layer are dense. For a common choice of $N = 10$, the size of the projection layer (P) might be 500 to 2000, while the hidden layer size H is typically 500 to 1000 units. Moreover, the hidden layer is used to compute probability distribution over all the words in the vocabulary, resulting in an output layer with dimensionality V. Thus, the computational complexity per each training example is

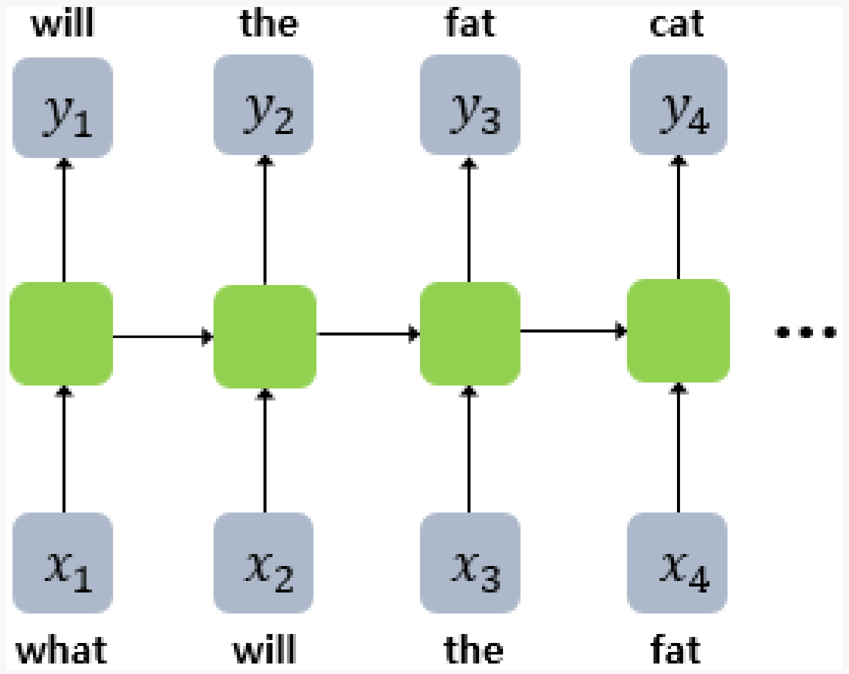
$$Q = N \times D + N \times D \times H + H \times V, \tag{2}$$

where the dominating term is $H \times V$. However, several practical solutions were proposed for avoiding it; either using hierarchical versions of the softmax [25, 23, 18], or avoiding normalized models completely by using models that are not normalized during training [4, 9]. With binary tree representations of the vocabulary, the number of output units that need to be evaluated can go down to around $log_2(V)$. Thus, most of the complexity is caused by the term $N \times D \times H$.

FeedForward Neural Network language model은 다음과 같은 Q를 얻는다.
Q = N(단어)xD(projection 크기) + NxDxH(Hidden layer size) + HxV(output layer)

02 논문 Introduction

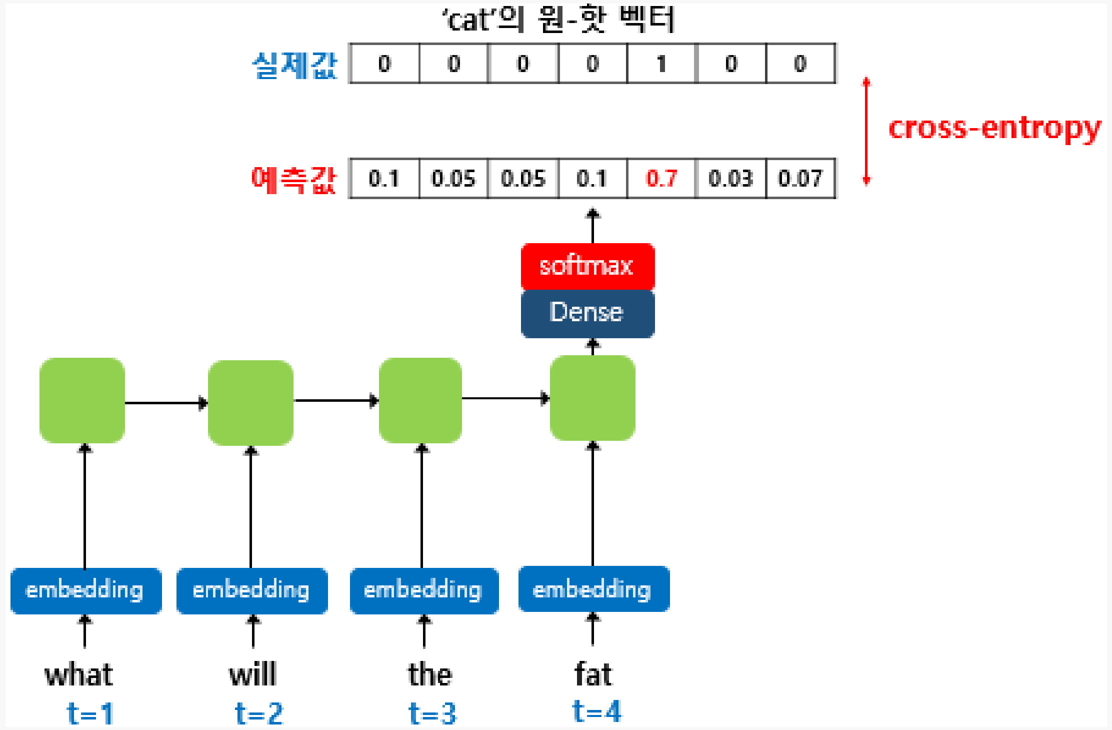
Model Architecture



출처 : 4) RNN 언어 모델(Recurrent Neural Network Language Model, RNNLM) - 딥 러닝을 이용한 자연어 처리 입문 (wikidocs.net)

Recurrent Neural Net Language Model

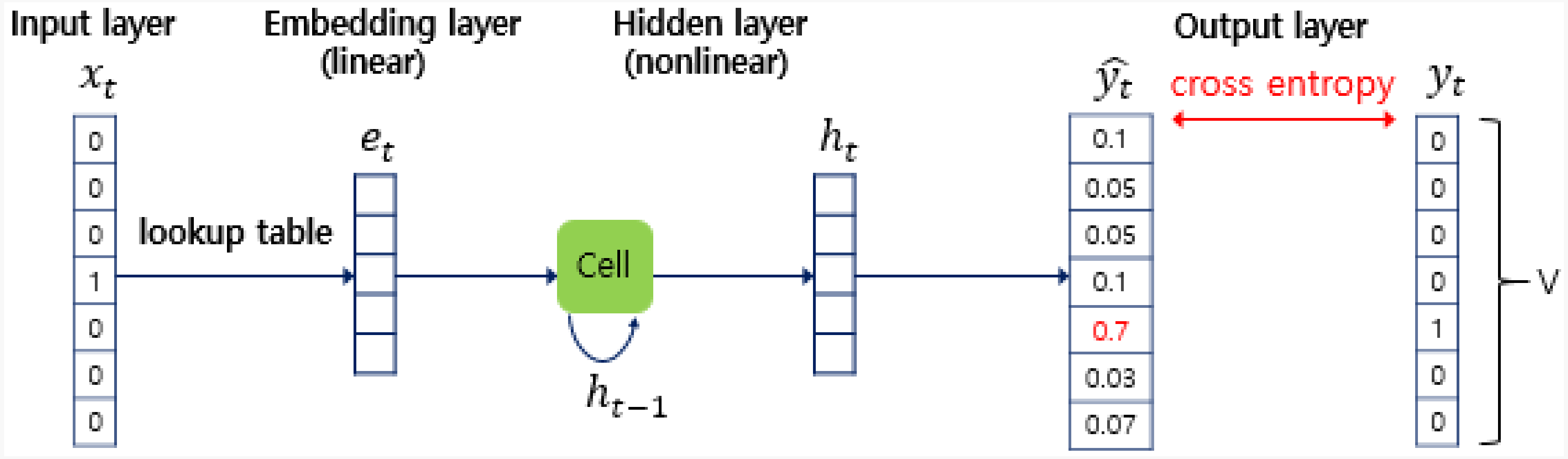
02 논문 Introduction
Model Architecture



출처 : 4) RNN 언어 모델(Recurrent Neural Network Language Model, RNNLM) - 딥 러닝을 이용한 자연어 처리 입문 (wikidocs.net)

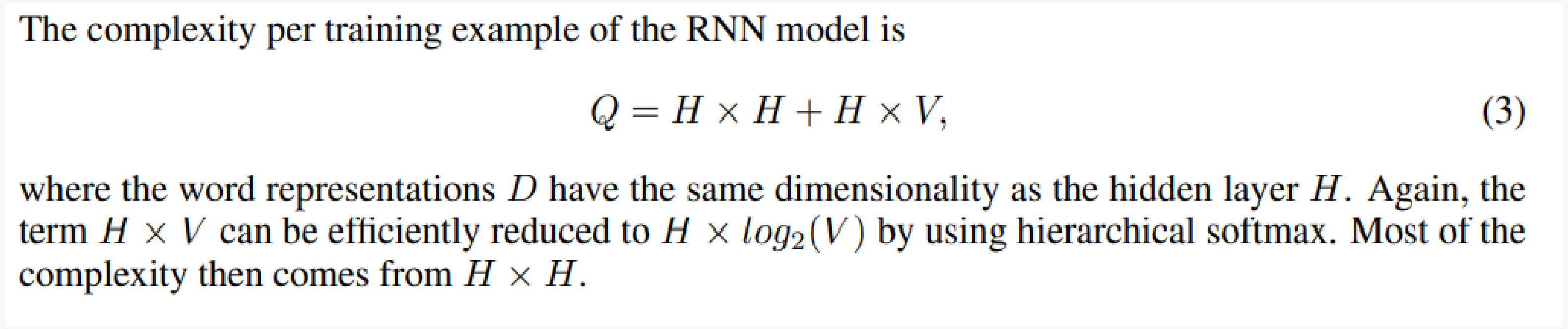
RNNLM에는 교사강요 기법이 들어간다.
교사강요를 사용하면 모델이 t시점에서 예측한 값을 t시점의 레이블이 아닌 실제 알고 있는 정답을 t+1 시점의 입력으로 사용한다.

02 논문 Introduction
Model Architecture



출처 : 4) RNN 언어 모델(Recurrent Neural Network Language Model, RNNLM) - 딥 러닝을 이용한 자연어 처리 입문 (wikidocs.net)

Recurrent Neural Net Language Model의 훈련 설계



The complexity per training example of the RNN model is

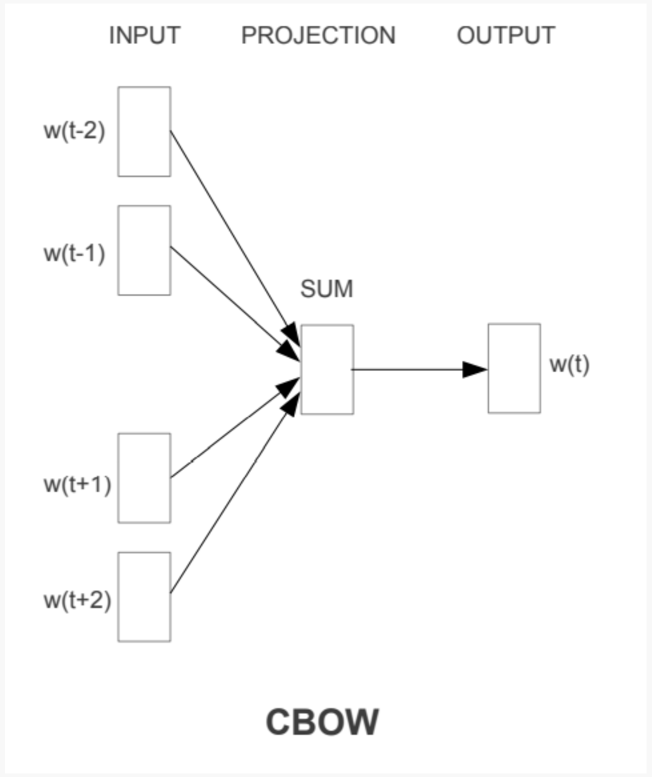
$$Q = H \times H + H \times V, \tag{3}$$

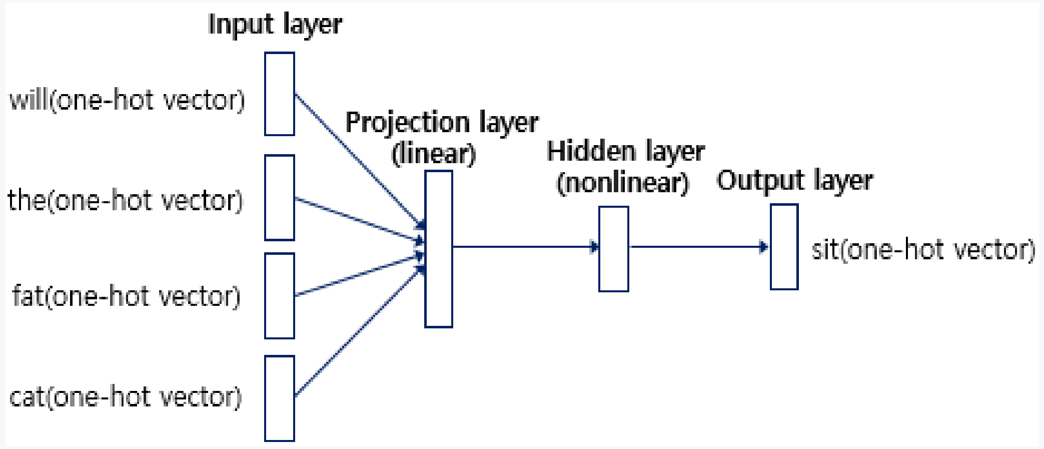
where the word representations D have the same dimensionality as the hidden layer H. Again, the term $H \times V$ can be efficiently reduced to $H \times log_2(V)$ by using hierarchical softmax. Most of the complexity then comes from $H \times H$.

Recurrent Neural Network language model은 다음과 같은 Q를 얻는다.
Q = HxH(Hidden layer size) + HxV(output layer)

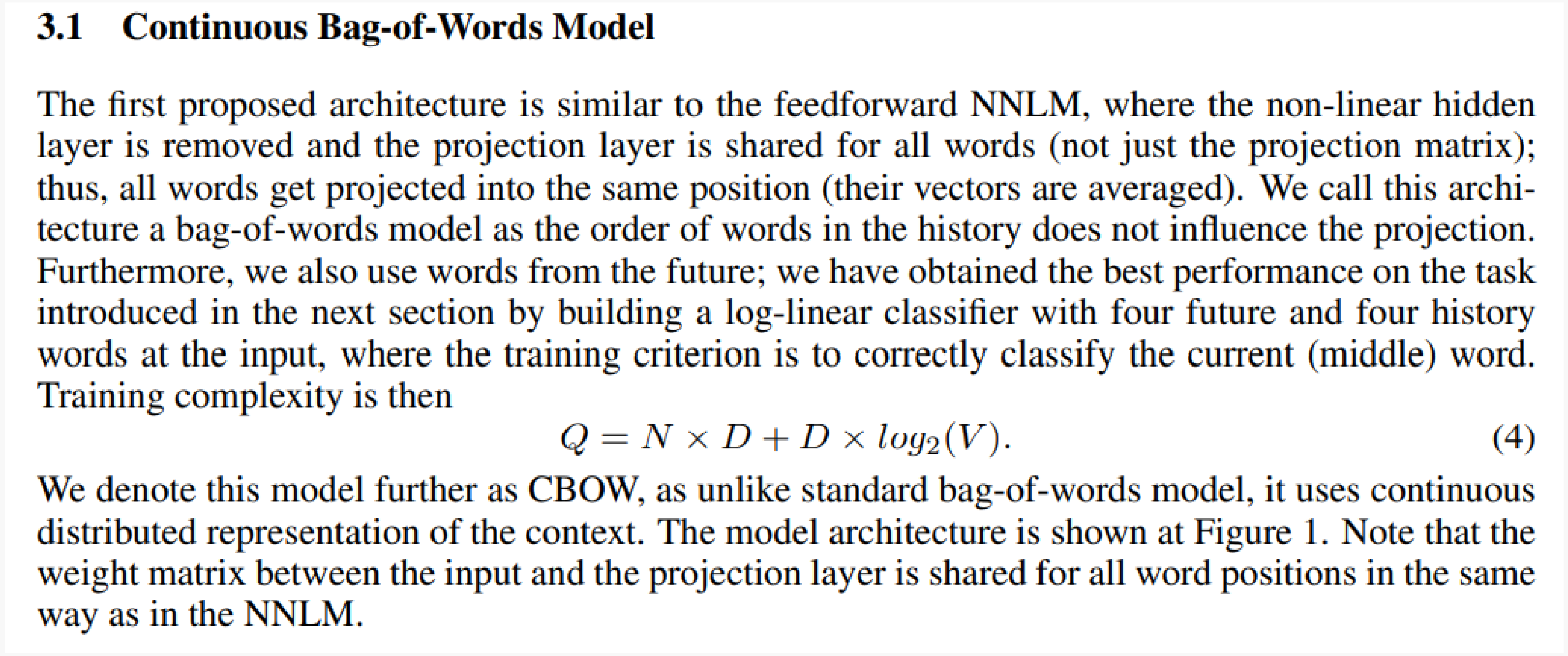
Word2Vec 소개

Model Architecture





Word2Vec 2개의 모델중 하나인 Continuous Bag Of Words는 FeedForward Neural Network Language model과 유사하다.



3.1 Continuous Bag-of-Words Model

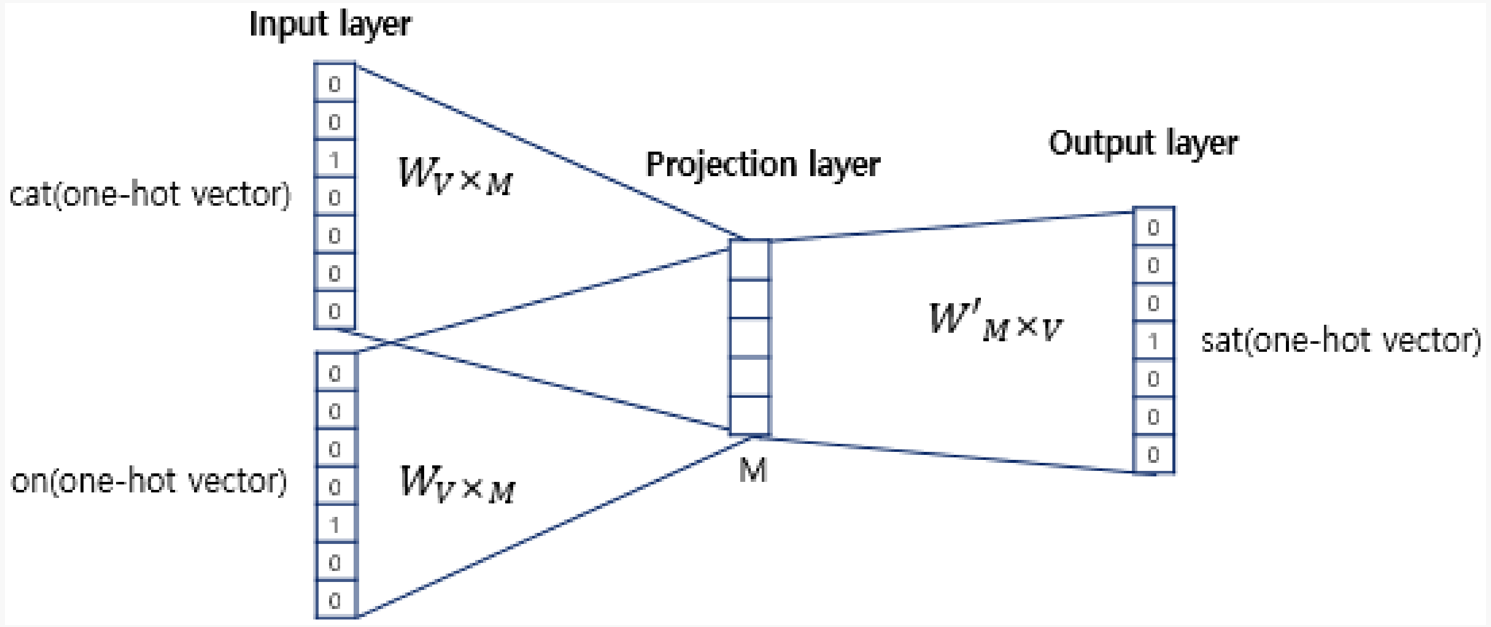
The first proposed architecture is similar to the feedforward NNLM, where the non-linear hidden layer is removed and the projection layer is shared for all words (not just the projection matrix); thus, all words get projected into the same position (their vectors are averaged). We call this architecture a bag-of-words model as the order of words in the history does not influence the projection. Furthermore, we also use words from the future; we have obtained the best performance on the task introduced in the next section by building a log-linear classifier with four future and four history words at the input, where the training criterion is to correctly classify the current (middle) word. Training complexity is then

$$Q = N \times D + D \times log_2(V). \tag{4}$$

We denote this model further as CBOW, as unlike standard bag-of-words model, it uses continuous distributed representation of the context. The model architecture is shown at Figure 1. Note that the weight matrix between the input and the projection layer is shared for all word positions in the same way as in the NNLM.

CBOW은 다음과 같은 Q를 얻는다.
Q = N(단어)xD(projection의 크기) + Dxlog2(V) (output layer의 차원)

03 Word2Vec 소개
Model Architecture

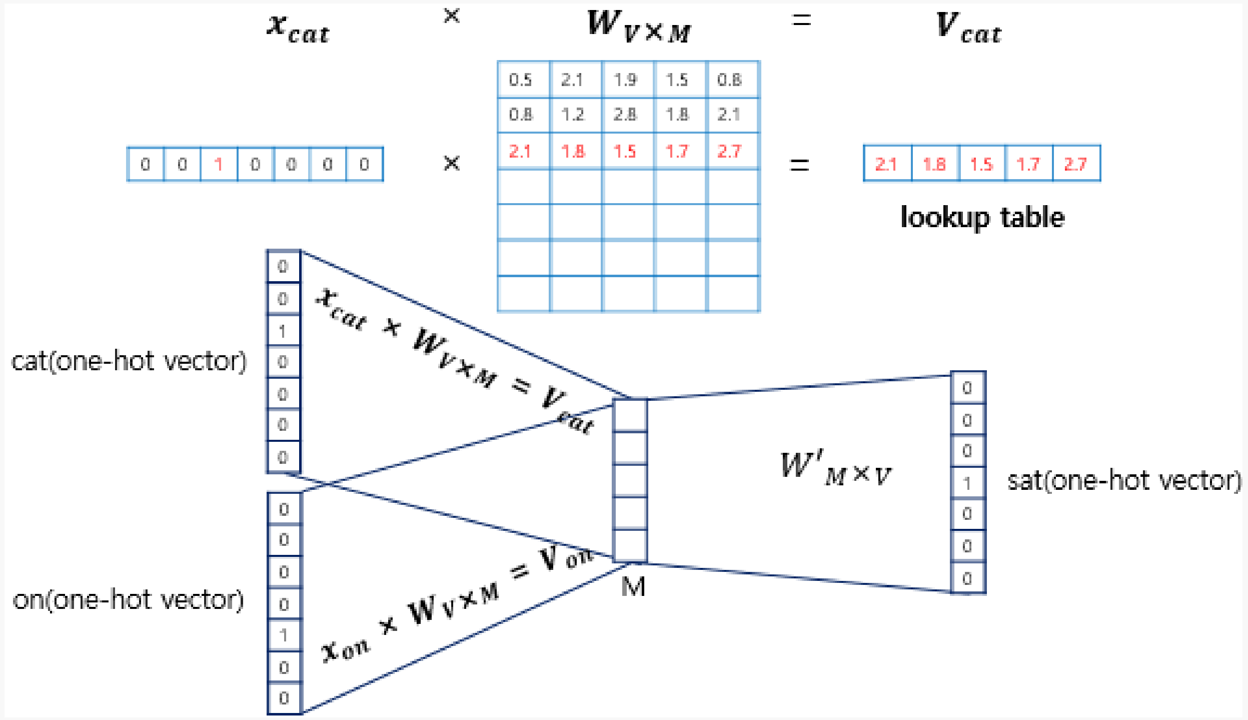


출처 : 02) 워드투벡터(Word2Vec) - 딥 러닝을 이용한 자연어 처리 입문 (wikidocs.net)

CBOW 동적 메커니즘을 도식화

Word2Vec 소개

Model Architecture

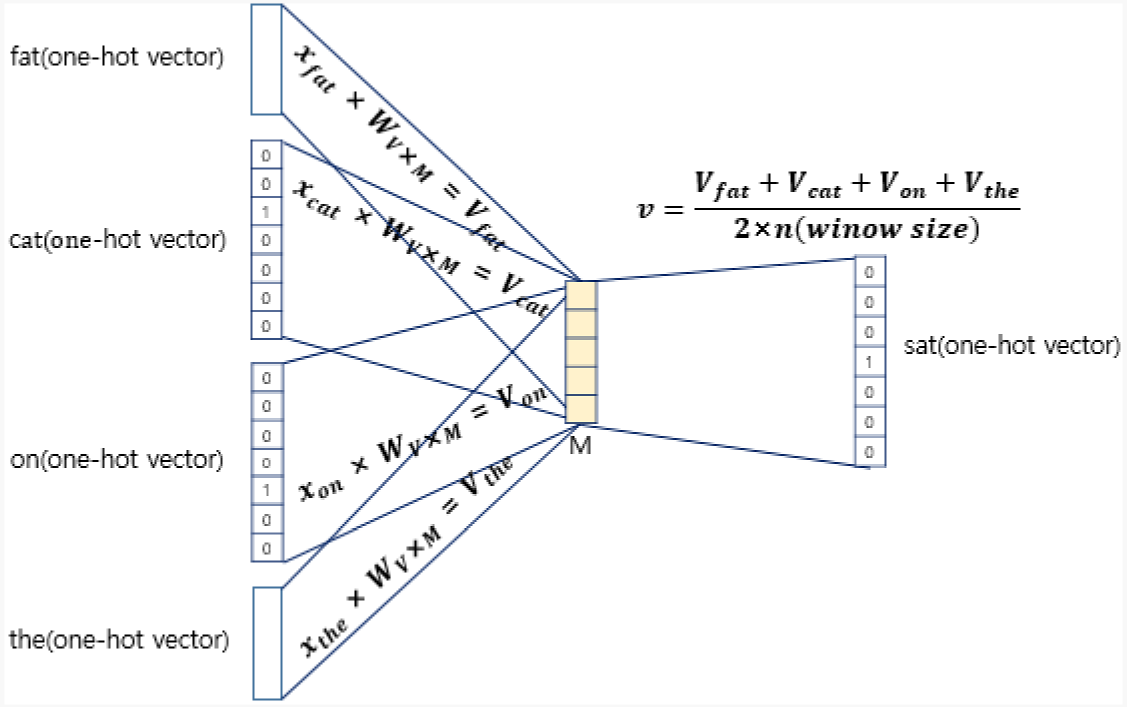


출처 : 02) 워드투벡터(Word2Vec) - 딥 러닝을 이용한 자연어 처리 입문 (wikidocs.net)

CBOW는 cat과 on이라는 벡터를 통해 sat이라는 vector를 잘 맞추기 위해서 W와 W`를 학습하는 구조이다.

03 Word2Vec 소개

Model Architecture



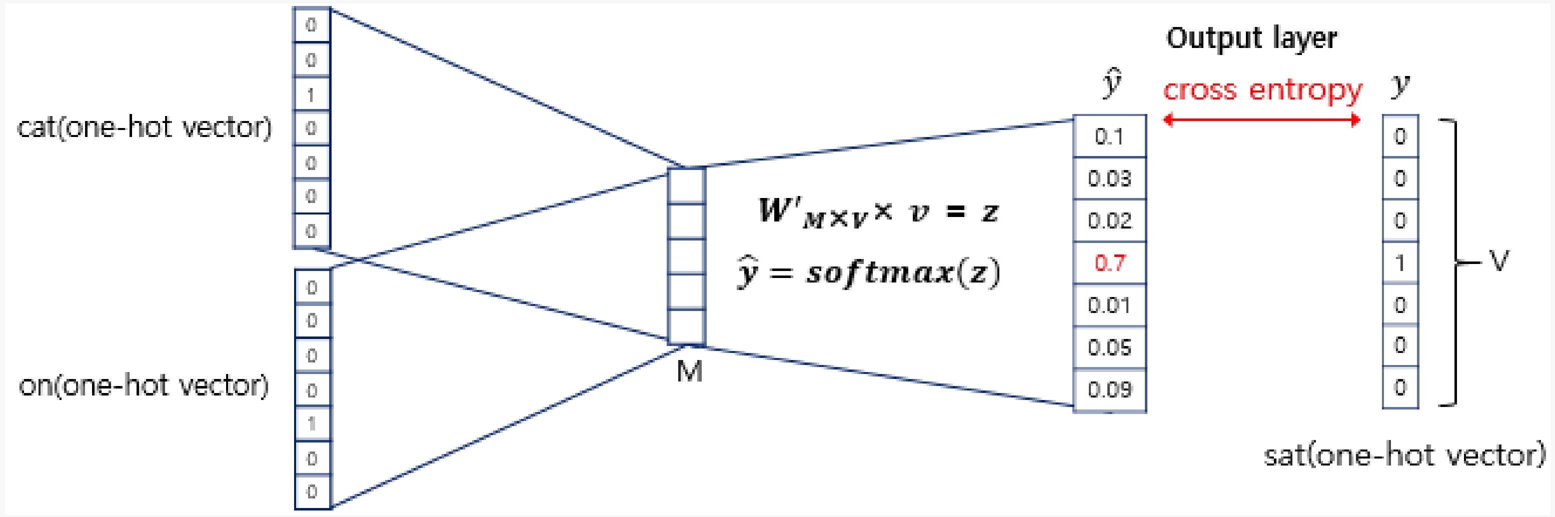
fat(one-hot vector)
cat(one-hot vector)
on(one-hot vector)
the(one-hot vector)
sat(one-hot vector)
$x_{fat} \times W_{V\times M} = V_{fat}$
$x_{cat} \times W_{V\times M} = V_{cat}$
$x_{on} \times W_{V\times M} = V_{on}$
$x_{the} \times W_{V\times M} = V_{the}$
$v = \frac{V_{fat} + V_{cat} + V_{on} + V_{the}}{2\times n(winow\ size)}$
M

출처 : 02) 워드투벡터(Word2Vec) - 딥 러닝을 이용한 자연어 처리 입문 (wikidocs.net)

CBOW는 모든 lookup table을 만든 다음 이 벡터들의 평균인 벡터들을 구한다.

Word2Vec 소개

Model Architecture

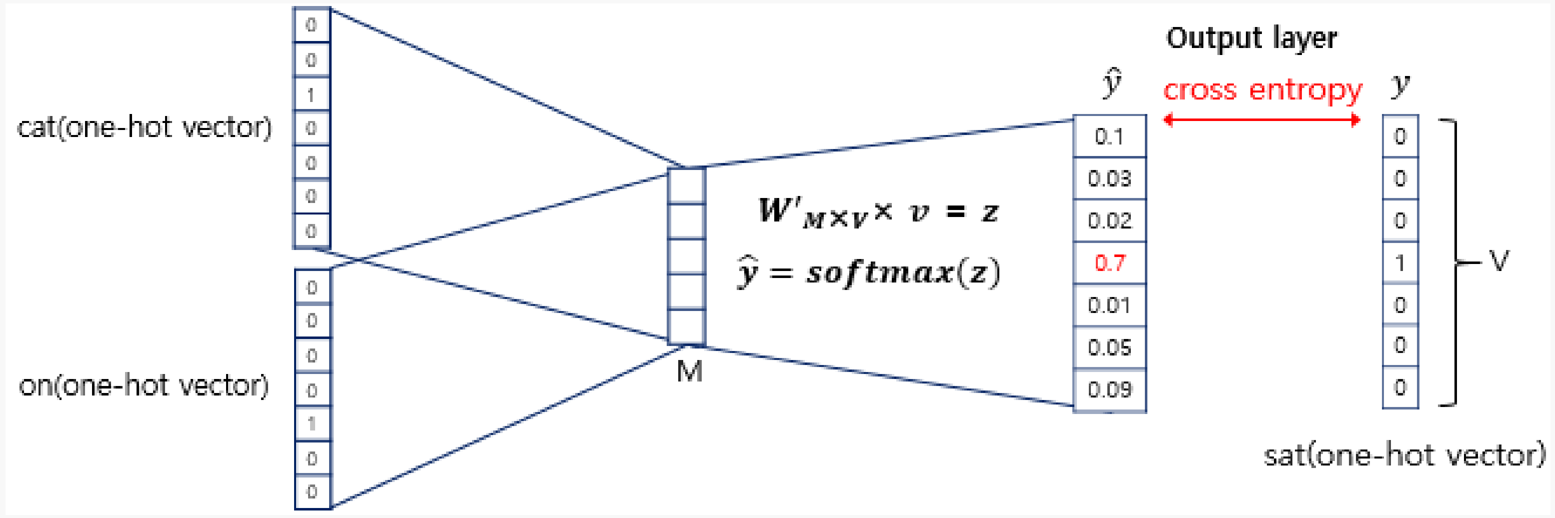


출처 : 02) 워드투벡터(Word2Vec) - 딥 러닝을 이용한 자연어 처리 입문 (wikidocs.net)

평균으로 구한 벡터는 두번째 가중치 행렬과 곱한다.
구한 결과를 softmax를 이용해 확률값으로 만들어주고 원-핫벡터로 바꿔준다.

03 Word2Vec 소개

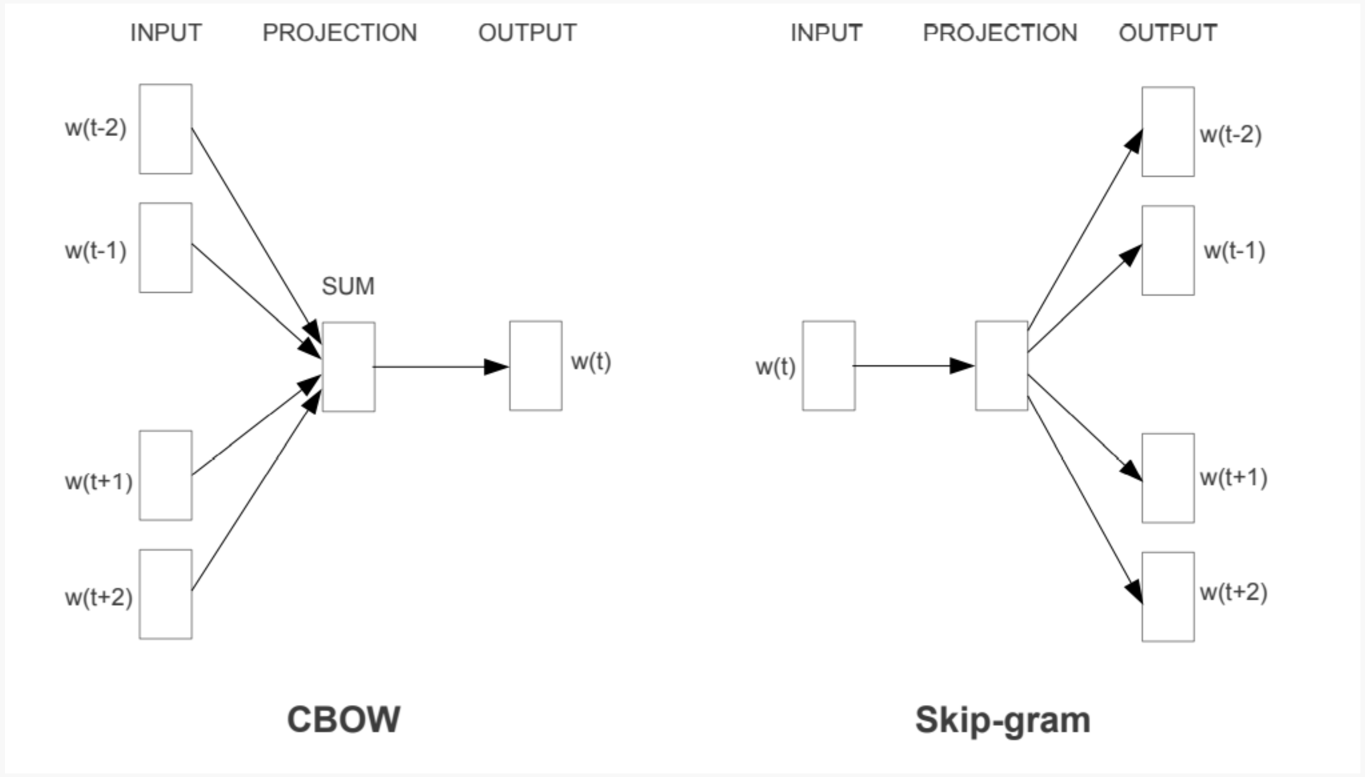
Model Architecture



출처 : 02) 워드투벡터(Word2Vec) - 딥 러닝을 이용한 자연어 처리 입문 (wikidocs.net)

평균으로 구한 벡터는 두번째 가중치 행렬과 곱한다.
구한 결과를 softmax를 이용해 확률값으로 만들어주고 원-핫벡터로 바꿔준다.

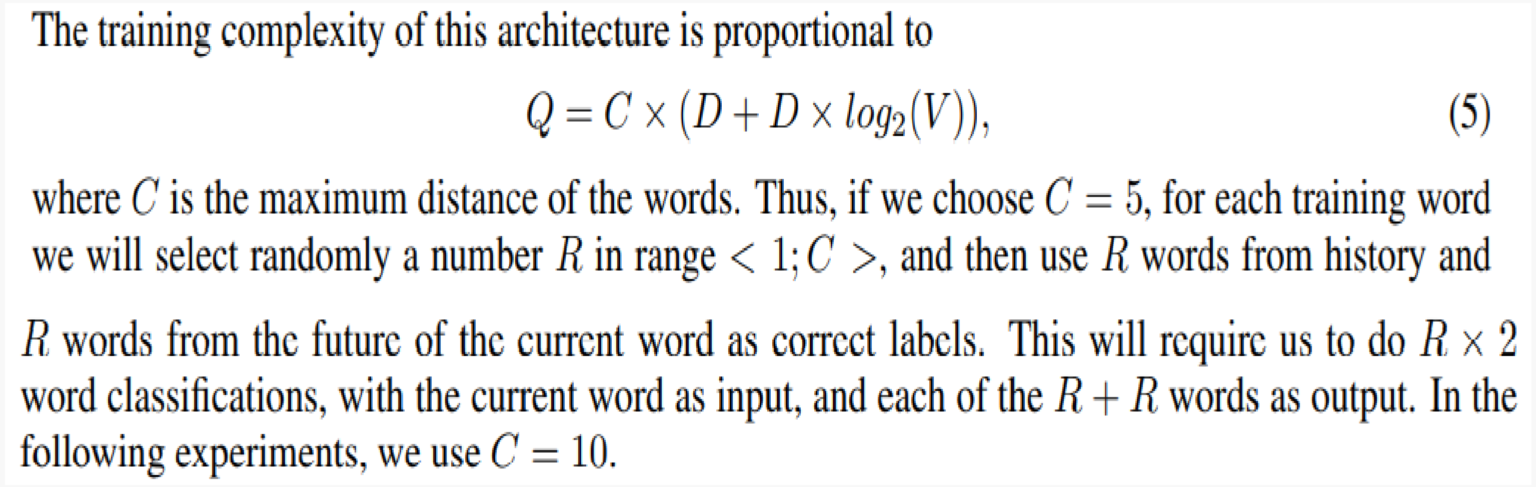
03 Word2Vec 소개
Model Architecture



Skip-gram은 CBOW와 유사하다.
CBOW같은 경우 주변부 단어를 통해 중심 단어를 예측하는 기법이고
Skip-gram은 중심단어를 통해 주변단어를 예측하는 기법이다.

Word2Vec 소개

Model Architecture



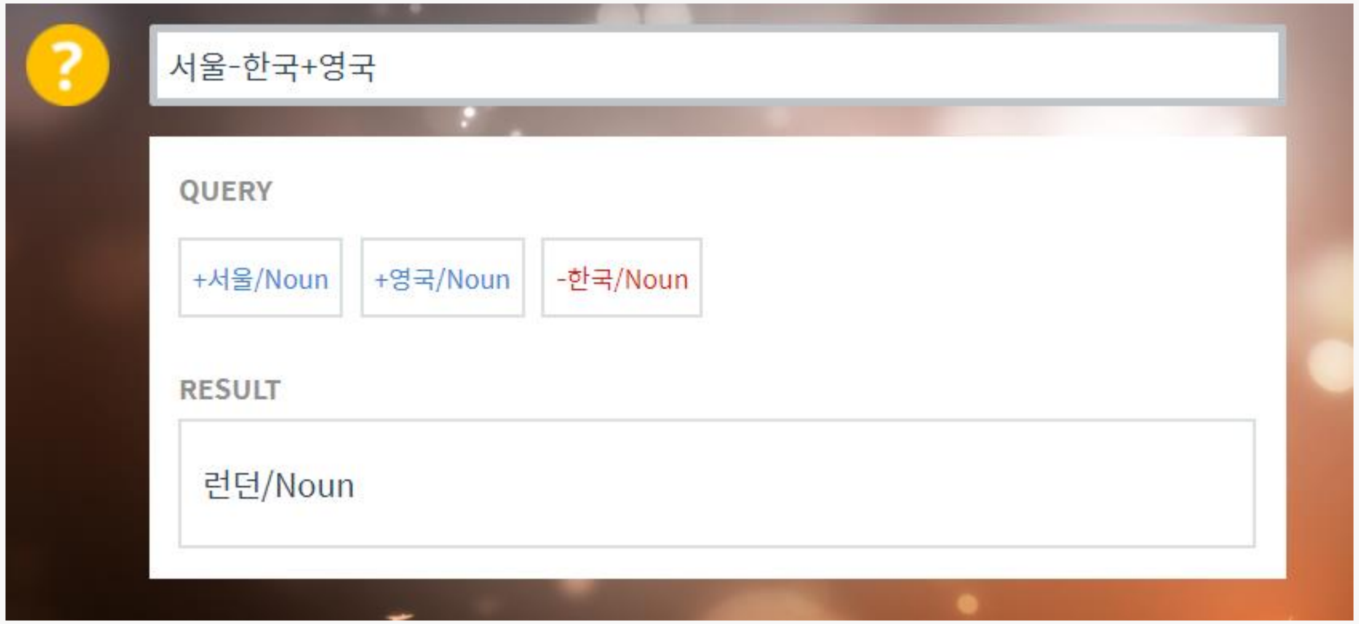
The training complexity of this architecture is proportional to

$$Q = C \times (D + D \times log_2(V)), \tag{5}$$

where C is the maximum distance of the words. Thus, if we choose $C = 5$, for each training word we will select randomly a number R in range $< 1; C >$, and then use R words from history and R words from the future of the current word as correct labels. This will require us to do $R \times 2$ word classifications, with the current word as input, and each of the $R + R$ words as output. In the following experiments, we use $C = 10$.

Skip-gram는 다음과 같은 Q를 얻는다.
Q = C(단어들 사이의 가장 먼 거리) x (D + D x log2(V))

04 결과


서울-한국+영국

QUERY

+서울/Noun +영국/Noun -한국/Noun

RESULT

런던/Noun

출처 : Korean Word2Vec

Word2Vec는 벡터 연산으로 유사도를 측정할 수 있어 실제 사진과 같은 계산을 가능하게 만든다.

04 결과

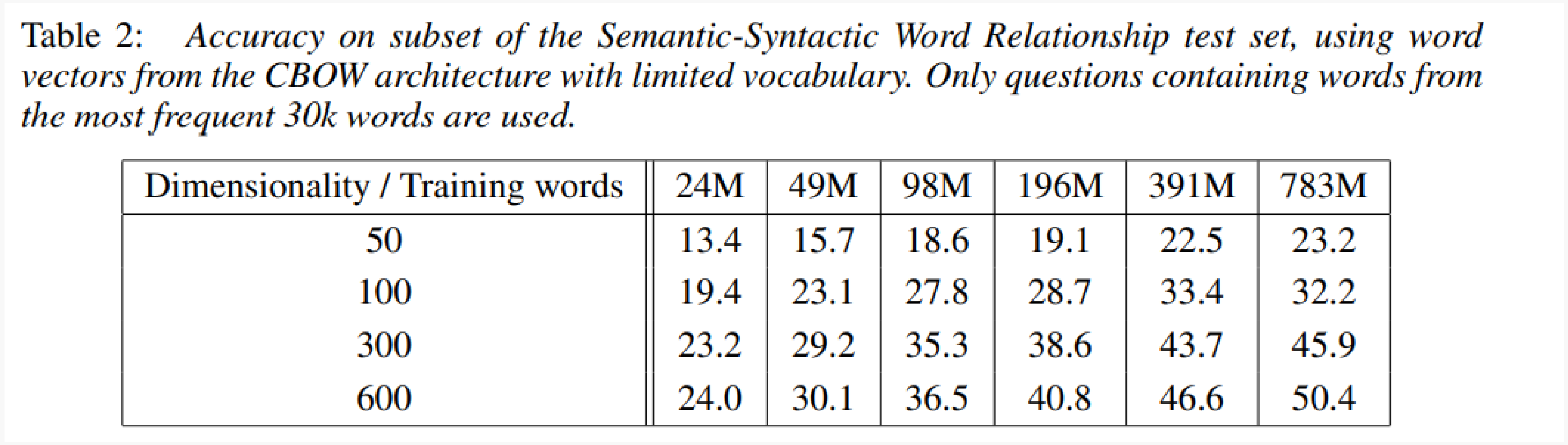


Table 2: *Accuracy on subset of the Semantic-Syntactic Word Relationship test set, using word vectors from the CBOW architecture with limited vocabulary. Only questions containing words from the most frequent 30k words are used.*

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

Dimensionality와 Training words를 동시에 들려주는 것도 성능효과가 있다.

04 결과

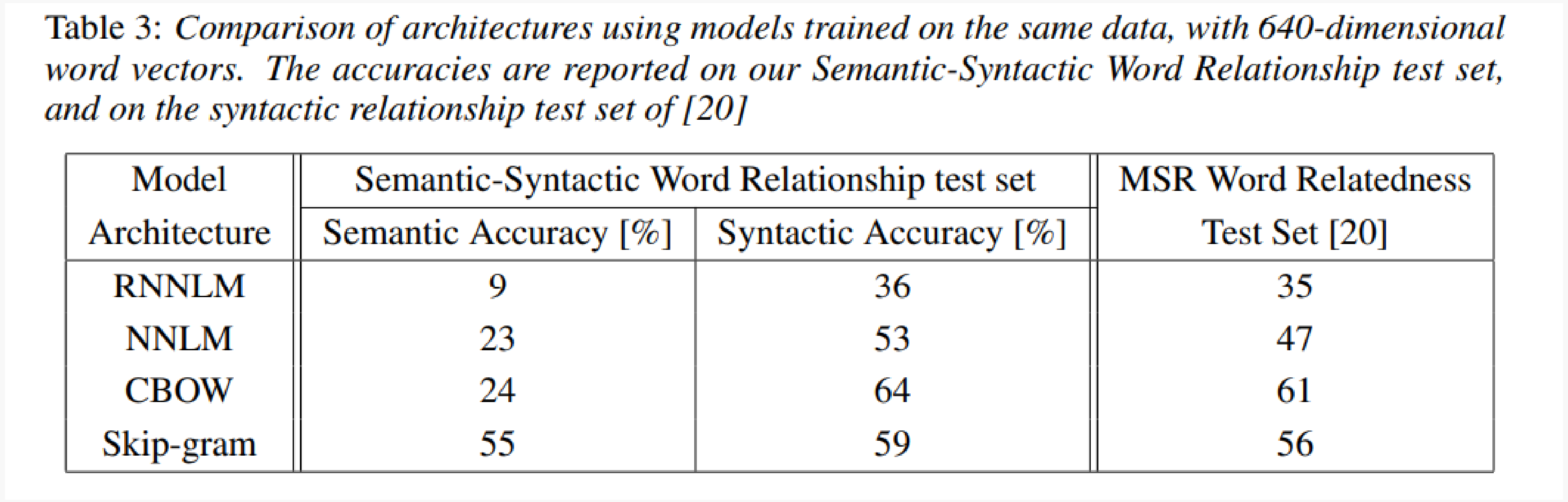


Table 3: *Comparison of architectures using models trained on the same data, with 640-dimensional word vectors. The accuracies are reported on our Semantic-Syntactic Word Relationship test set, and on the syntactic relationship test set of [20]*

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

성능을 비교했을 때 Word2Vec가 RNNLM과 NNLM에 비해 성능에서 절대우위를 갖는다.
CBOW같은 경우 문법적 정확성이 skip-gram보다 우수하지만 의미적 정확성은 Skip-gram이 성능이 좋다.

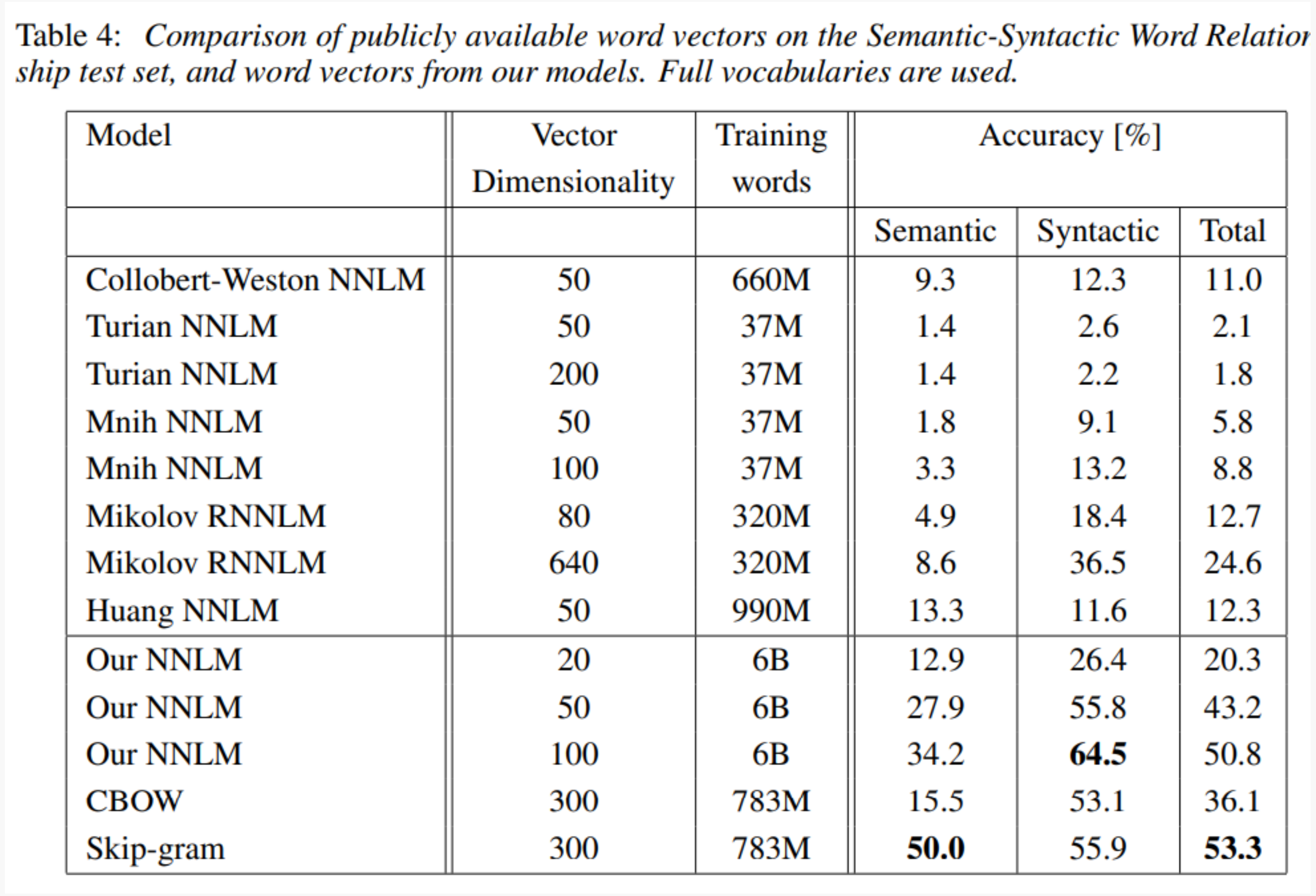


Table 4: *Comparison of publicly available word vectors on the Semantic-Syntactic Word Relation ship test set, and word vectors from our models. Full vocabularies are used.*

Model	Vector Dimensionality	Training words	Accuracy [%]		
			Semantic	Syntactic	Total
Collobert-Weston NNLM	50	660M	9.3	12.3	11.0
Turian NNLM	50	37M	1.4	2.6	2.1
Turian NNLM	200	37M	1.4	2.2	1.8
Mnih NNLM	50	37M	1.8	9.1	5.8
Mnih NNLM	100	37M	3.3	13.2	8.8
Mikolov RNNLM	80	320M	4.9	18.4	12.7
Mikolov RNNLM	640	320M	8.6	36.5	24.6
Huang NNLM	50	990M	13.3	11.6	12.3
Our NNLM	20	6B	12.9	26.4	20.3
Our NNLM	50	6B	27.9	55.8	43.2
Our NNLM	100	6B	34.2	**64.5**	50.8
CBOW	300	783M	15.5	53.1	36.1
Skip-gram	300	783M	**50.0**	55.9	**53.3**

Semantic-Syntactic Word Relationship test set에서 Skip-gram이 비교적 적은 training words를 이용해 의미 점수에서 가장 높은 점수를 기록했다.

04 결과

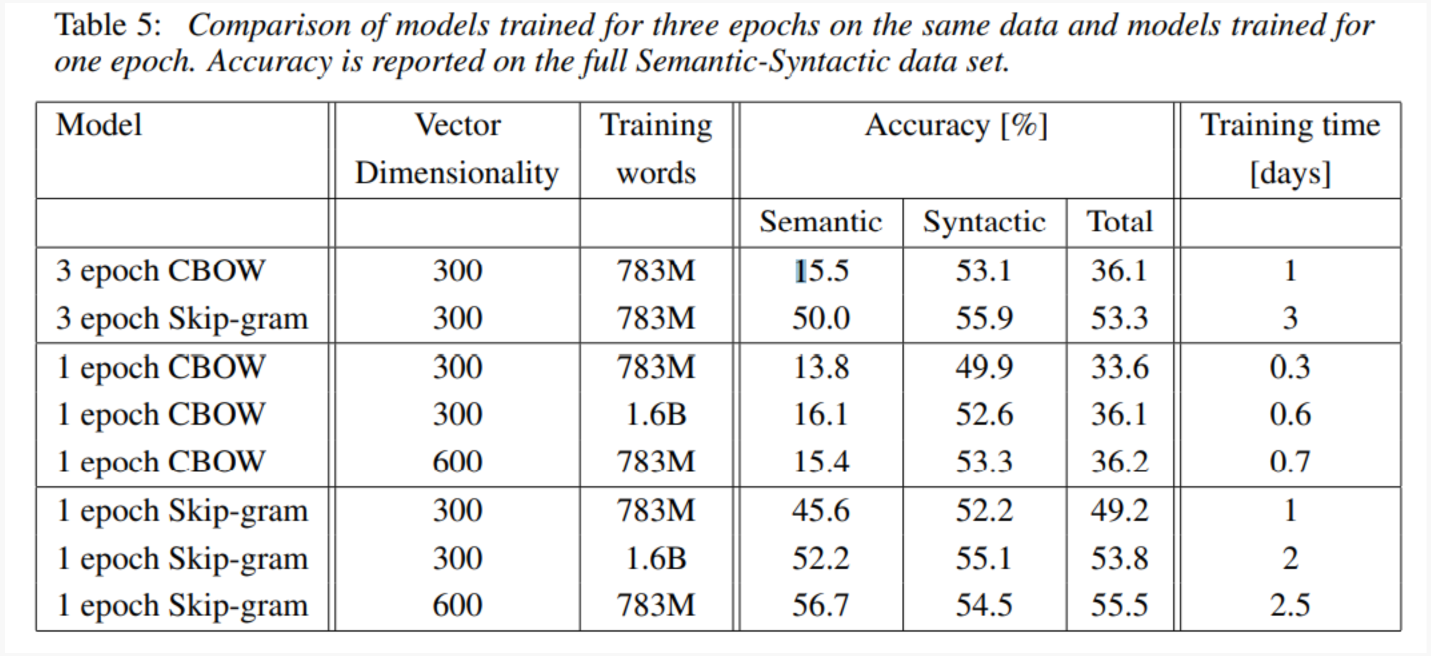


Table 5: *Comparison of models trained for three epochs on the same data and models trained for one epoch. Accuracy is reported on the full Semantic-Syntactic data set.*

Model	Vector Dimensionality	Training words	Accuracy [%]			Training time [days]
			Semantic	Syntactic	Total	
3 epoch CBOW	300	783M	15.5	53.1	36.1	1
3 epoch Skip-gram	300	783M	50.0	55.9	53.3	3
1 epoch CBOW	300	783M	13.8	49.9	33.6	0.3
1 epoch CBOW	300	1.6B	16.1	52.6	36.1	0.6
1 epoch CBOW	600	783M	15.4	53.3	36.2	0.7
1 epoch Skip-gram	300	783M	45.6	52.2	49.2	1
1 epoch Skip-gram	300	1.6B	52.2	55.1	53.8	2
1 epoch Skip-gram	600	783M	56.7	54.5	55.5	2.5

동일한 상황에서 CBOW와 Skip-gram간의 비교

04 결과

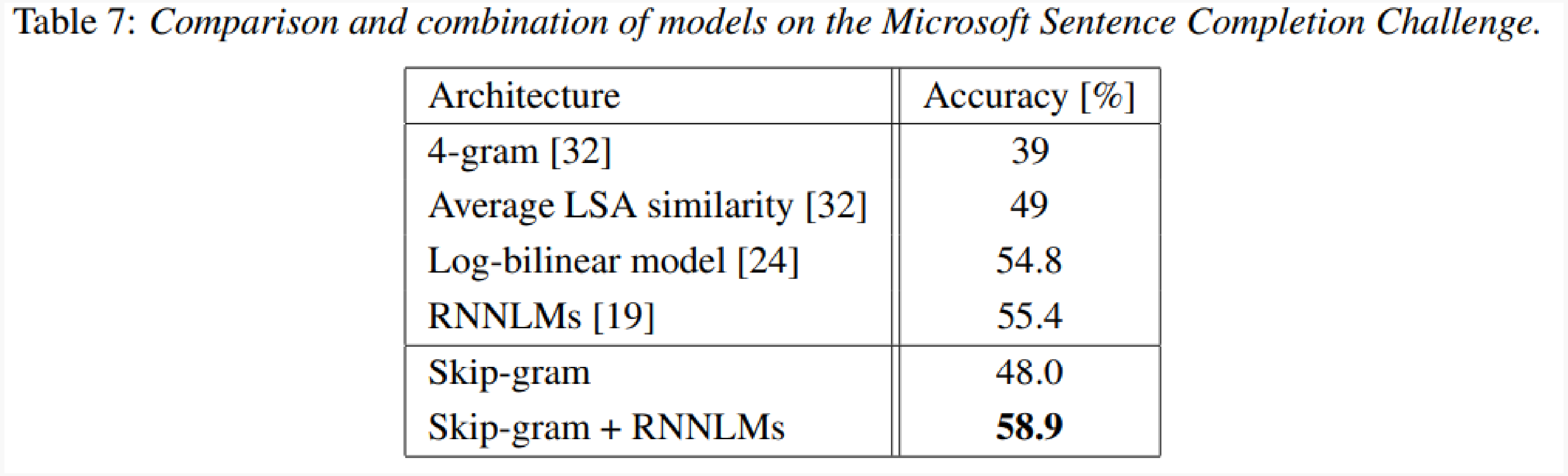


Table 7: *Comparison and combination of models on the Microsoft Sentence Completion Challenge.*

Architecture	Accuracy [%]
4-gram [32]	39
Average LSA similarity [32]	49
Log-bilinear model [24]	54.8
RNNLMs [19]	55.4
Skip-gram	48.0
Skip-gram + RNNLMs	**58.9**

Microsoft Sentence Completion Challenge에서 Skip-gram과 RNNLM을 동시에 쓴 모델의 정확도가 제일 높았음

04 결과 요약

적은 비용으로 더 좋은 성능을 내는 모델

Word2Vec에서 CBOW(Continuous Bag Of Words)는   FeedForward Neural Network Language Model과 유사하며 주변단어를 이용해 중심단어를 맞추는 CBOW와 달리 Skip-gram은 중심 단어를 이용해 주변단어를 맞춘다.

성능에 있어서 CBOW가 문법론적인 측면에서 약간 우세하긴 하나 의미론적인 측면과 전체적인 측면에서 Skip-gram이 좀 더 나은 성능을 갖고 있음,

Q & A

Q&A

THANK YOU -

경청해주셔서 감사합니다.