NLP paper study week-02 NLP - Efficient Estimation of Word Representations in Vector Space

Word2Vec (CBOW & Skip-gram) 핵심

Cheonghae Kim

Why Word Representations?

+ 기존 단어 표현: One-hot -> 공간도 많이 차지하고 유사한 단어를 알 수 없음

단어	나는	너는	밥을	먹었다
나는	1	0	0	0
너는	0	1	0	0
밥을	0	0	1	0
먹었다	0	0	0	1

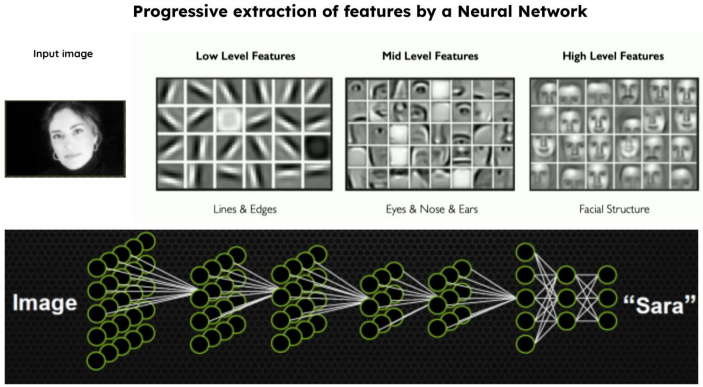
So **유사한 단어는 유사한 벡터**로 표현되는 방법

Embedding Layer

단어를 고차원 희소벡터(one-hot vector)에서 **저차원 밀집벡터(dense vector)로 변환해주는 층**

-> 단어는 **실수(real number) 벡터로 바꾸어 의미 정보**를 담을 수 있다.

> 1. associate with each word in the vocabulary a distributed *word feature vector* (a real-valued vector in $\mathbb{R}^m$),



출저: Nvidia website

one-hot에서
특징을 추출해
실수로 표현!

```
Token String    Token ID        Embedded Token Vector
    '<s>' ->   0 -> [ 0.1150, -0.1438,  0.0555, ... ]
  '<pad>' ->   1 -> [ 0.1149, -0.1438,  0.0547, ... ]
   '</s>' ->   2 -> [ 0.0010, -0.0922,  0.1025, ... ]
  '<unk>' ->   3 -> [ 0.1149, -0.1439,  0.0548, ... ]
      '.' ->   4 -> [-0.0651, -0.0622, -0.0002, ... ]
    ' the' ->   5 -> [-0.0340,  0.0068, -0.0844, ... ]
      ',' ->   6 -> [ 0.0483, -0.0214, -0.0927, ... ]
    ' to' ->   7 -> [-0.0439,  0.0201,  0.0189, ... ]
   ' and' ->   8 -> [ 0.0523, -0.0208, -0.0254, ... ]
    ' of' ->   9 -> [-0.0732,  0.0070, -0.0286, ... ]
     ' a' ->  10 -> [-0.0194,  0.0302, -0.0838, ... ]
                    ...
```

출저:
https://www.lesswrong.com/posts/pHPmMGEMYefk9jLeh/llm-basics-embedding-spaces-transformer-token-vectors-are

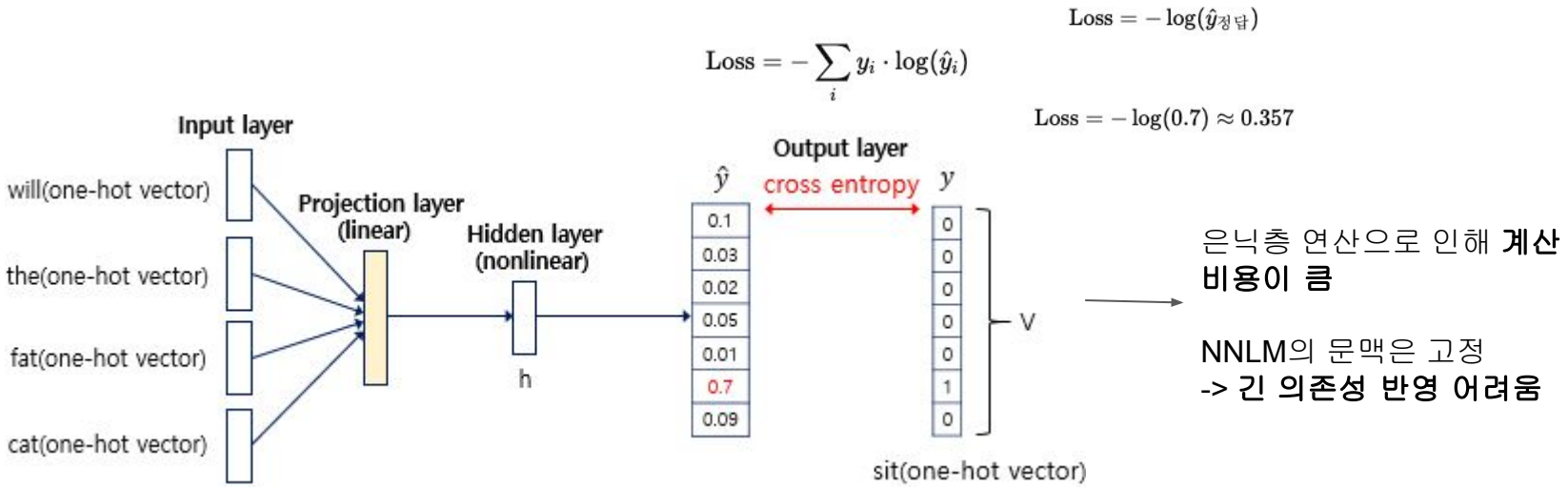
학습과정을 통해
대략적인 단어 의미파악 가능
but 동음이의어, 문맥 변화 등은 해석불가

NNLM (Feedforword Neural Net Language Model)

문맥(앞의 N개 단어)으로부터 **다음 단어를 예측**하는 모델

+ Constructure

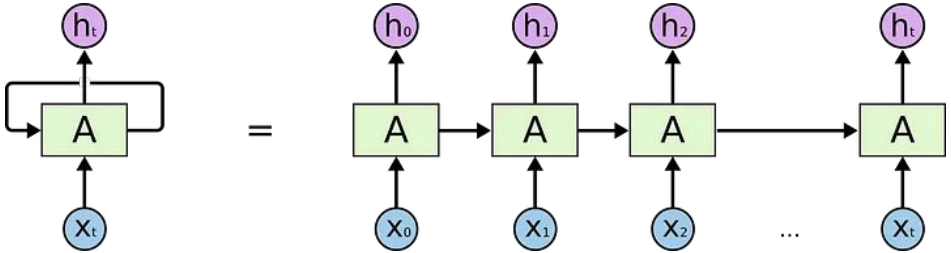
[input: words] -> [Embedding Layer] → [Hidden Layer] → [Softmax Output Layer]

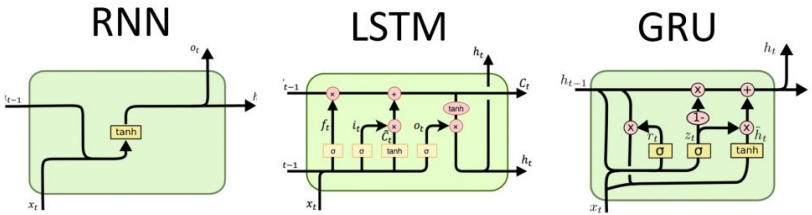


RNNLM(Recurrent Neural Net Language Mode)

순환신경망을 이용해 긴 문맥을 반영하는 모델

why is RNN(순환신경망)? for Sequence





RNN한계

한계	설명
장기 의존성 문제	입력 간 거리가 멀수록 앞의 정보가 소실됨 (vanishing gradient)
학습 어려움	긴 문장에서 정보가 왜곡되거나 사라지기 쉬움
병렬 처리 불가	입력을 순차적으로 처리해야 해서 속도가 느림

Vanishing Gradient (기울기 소실)?

역전파할수록 기울기가 점점 작아져, 앞쪽 레이어나 오래된 시점의 가중치가 거의 업데이트되지 않는 현상

ex) 나는 어제 친구와 카페에서 커피를 마셨는데, **기분이 좋았다**. -> 기분이 좋았다"를 예측하려면 앞쪽 "카페" 같은 단서가 필요 (RNN소실로 기억못함)

New non-linear Model

3 New Log-linear Models

In this section, we propose two new model architectures for learning distributed representations of words that try to minimize computational complexity. The main observation from the previous section was that most of the complexity is caused by the non-linear hidden layer in the model. While this is what makes neural networks so attractive, we decided to explore simpler models that might not be able to represent the data as precisely as neural networks, but can possibly be trained on much more data efficiently.

The new architectures directly follow those proposed in our earlier work [13, 14], where it was found that neural network language model can be successfully trained in two steps: first, continuous word vectors are learned using simple model, and then the N-gram NNLM is trained on top of these distributed representations of words. While there has been later substantial amount of work that focuses on learning word vectors, we consider the approach proposed in [13] to be the simplest one. Note that related models have been proposed also much earlier [26, 8].

NNLM Complexity

$$Q = N \times D + N \times D \times H + H \times V,$$

RNNLM Complexity

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

CBOW Complexity

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

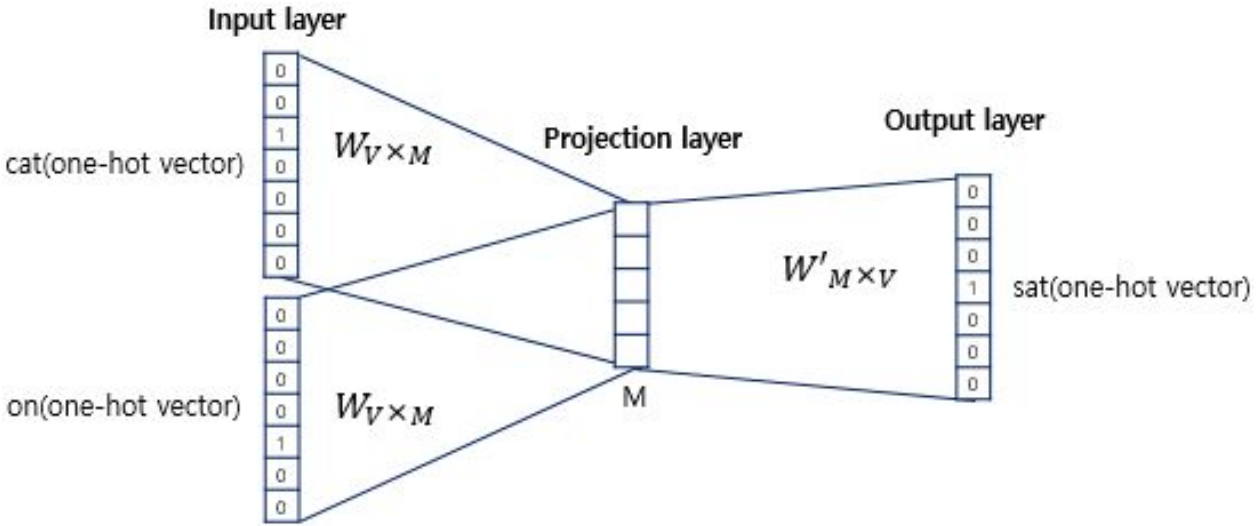
Skip-gram Complexity

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

CBOW (Continuous Bag-of-Words Model)

주변 단어(Context Words)를 보고 중심 단어(Target Word)를 예측하는 방식

분포 가설(distributional hypothesis): 비슷한 문맥에서 같이 등장하는 경향이 있는 단어들은 비슷한 의미를 가진다.



CBOW 계산과정

if. window size = 2

단어	인덱스
i	0
like	1
natural	2
language	3
processing	4

→ 단어장 크기 $V = 5$
→ 임베딩 차원 $N = 3$ (3차원 벡터로 예시)

✅ 1단계: 원-핫 벡터

단어	원-핫 벡터
i	[1, 0, 0, 0, 0]
like	[0, 1, 0, 0, 0]
language	[0, 0, 0, 1, 0]
processing	[0, 0, 0, 0, 1]

✅ 2단계: 임베딩 행렬 $E \in \mathbb{R}^{5\times3}$

단어	임베딩 벡터 (3차원)
i	[0.1, 0.0, 0.3]
like	[0.0, 0.2, 0.1]
natural	[0.3, 0.4, 0.2]
language	[-0.2, 0.1, 0.0]
processing	[0.2, -0.1, 0.2]

✅ 3단계: 투사층 연산 = 주변 단어 4개의 임베딩 벡터 평균

context words: `"i"`, `"like"`, `"language"`, `"processing"`

해당 벡터:

```
"i"          → [0.1,  0.0, 0.3]
"like"       → [0.0,  0.2, 0.1]
"language"   → [-0.2, 0.1, 0.0]
"processing" → [0.2, -0.1, 0.2]
```

합산:

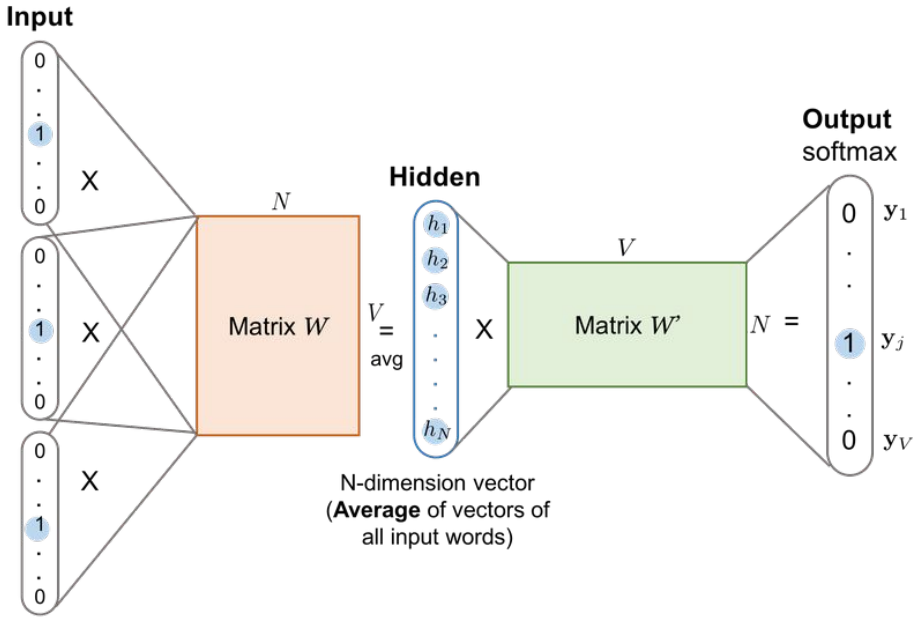
```
= [0.1 + 0.0 + (-0.2) + 0.2,  0.0 + 0.2 + 0.1 + (-0.1),  0.3 + 0.1 + 0.0 +
= [0.1, 0.2, 0.6]
```

평균:

```
= [0.1 / 4, 0.2 / 4, 0.6 / 4]
= [0.025, 0.05, 0.15]
```

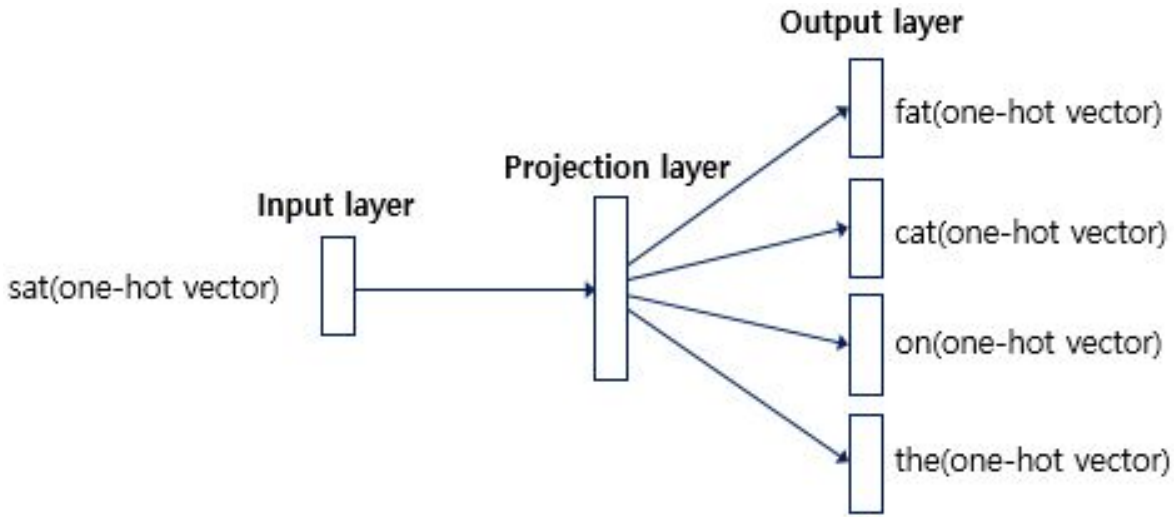
```
= [0.1 / 4, 0.2 / 4, 0.6 / 4]
= [0.025, 0.05, 0.15]
```

차원 ↓ / 단어 →	i	like	natural	language	processing
dim1	0.1	0.2	0.3	-0.1	0.0
dim2	0.0	-0.1	0.4	0.3	0.2
dim3	0.2	0.0	0.1	0.2	-0.2



Skip-gram (reverse CBOW)

- (i, like) (like, I), (like, natural), (natural, like), (natural, language), (language, natural), (language, processing), (processing, language) -> 각각 (중심 단어, 주변 단어로 구성됨)



사용하는 이유는 ? -> 희귀단어에 대한 예측에서 사용된다.

CBOW vs Skip-gram

✅ 6. CBOW vs Skip-gram 비교 요약

항목	CBOW	Skip-gram
입력	주변 단어들	중심 단어 1개
출력	중심 단어 1개	주변 단어 여러 개
연산량	적음 (벡터 평균 1번)	많음 (softmax 여러 번)
희귀 단어	학습 어려움	학습 잘 됨
데이터 크기	작아도 괜찮음	클수록 효과적
예측 방향	context → center	center → context

쫌 더 미세한 의미론적인 뜻을 해석할 수 있다고 주장한다.

4 Results

To compare the quality of different versions of word vectors, previous papers typically use a table showing example words and their most similar words, and understand them intuitively. Although it is easy to show that word *France* is similar to *Italy* and perhaps some other countries, it is much more challenging when subjecting those vectors in a more complex similarity task, as follows. We follow previous observation that there can be many different types of similarities between words, for example, word *big* is similar to *bigger* in the same sense that *small* is similar to *smaller*. Example of another type of relationship can be word pairs *big - biggest* and *small - smallest* [20]. We further denote two pairs of words with the same relationship as a question, as we can ask: "What is the word that is similar to *small* in the same sense as *biggest* is similar to *big*?"

Somewhat surprisingly, these questions can be answered by performing simple algebraic operations with the vector representation of words. To find a word that is similar to *small* in the same sense as *biggest* is similar to *big*, we can simply compute vector $X = vector("biggest") - vector("big") + vector("small")$. Then, we search in the vector space for the word closest to X measured by cosine distance, and use it as the answer to the question (we discard the input question words during this search). When the word vectors are well trained, it is possible to find the correct answer (word $smallest$) using this method.

Finally, we found that when we train high dimensional word vectors on a large amount of data, the resulting vectors can be used to answer very subtle semantic relationships between words, such as a city and the country it belongs to, e.g. France is to Paris as Germany is to Berlin. Word vectors with such semantic relationships could be used to improve many existing NLP applications, such as machine translation, information retrieval and question answering systems, and may enable other future applications yet to be invented.

Matrics

Semantic Accuracy (의미 정확도) -> 프랑스 - 파리 와 같은 단어간의 의미론적 관계

Syntatic Accuracy (구문 정확도) -> 크다 - 더 큰 과 같은 구문적 관계

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

훨씬 낮은 단어수에도

불구하고 몇몇 지표에서

Skip-gram이 높다.

Table 4: *Comparison of publicly available word vectors on the Semantic-Syntactic Word Relationship 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**

성능은 더 높고 트레이닝타임은 훨씬 낮다.

Table 6: *Comparison of models trained using the DistBelief distributed framework. Note that training of NNLM with 1000-dimensional vectors would take too long to complete.*

Model	Vector Dimensionality	Training words	Accuracy [%]			Training time [days x CPU cores]
			Semantic	Syntactic	Total	
NNLM	100	6B	34.2	64.5	50.8	14 x 180
CBOW	1000	6B	57.3	68.9	63.7	2 x 140
Skip-gram	1000	6B	66.1	65.1	65.6	2.5 x 125

선형 규칙성 (linear regularities)

단어 벡터들 사이의 의미 관계가

선형 연산(덧셈,뺄셈 등)으로

표현된다는 성질

Table 8: *Examples of the word pair relationships, using the best word vectors from Table 4 (Skip-gram model trained on 783M words with 300 dimensionality).*

Relationship	Example 1	Example 2	Example 3
France - Paris	Italy: Rome	Japan: Tokyo	Florida: Tallahassee
big - bigger	small: larger	cold: colder	quick: quicker
Miami - Florida	Baltimore: Maryland	Dallas: Texas	Kona: Hawaii
Einstein - scientist	Messi: midfielder	Mozart: violinist	Picasso: painter
Sarkozy - France	Berlusconi: Italy	Merkel: Germany	Koizumi: Japan
copper - Cu	zinc: Zn	gold: Au	uranium: plutonium
Berlusconi - Silvio	Sarkozy: Nicolas	Putin: Medvedev	Obama: Barack
Microsoft - Windows	Google: Android	IBM: Linux	Apple: iPhone
Microsoft - Ballmer	Google: Yahoo	IBM: McNealy	Apple: Jobs
Japan - sushi	Germany: bratwurst	France: tapas	USA: pizza

assumes exact match, the results in Table 8 would score only about 60%). We believe that word vectors trained on even larger data sets with larger dimensionality will perform significantly better, and will enable the development of new innovative applications. Another way to improve accuracy is to provide more than one example of the relationship. By using ten examples instead of one to form the relationship vector (we average the individual vectors together), we have observed improvement of accuracy of our best models by about 10% absolutely on the semantic-syntactic test.

It is also possible to apply the vector operations to solve different tasks. For example, we have observed good accuracy for selecting out-of-the-list words, by computing average vector for a list of words, and finding the most distant word vector. This is a popular type of problems in certain human intelligence tests. Clearly, there is still a lot of discoveries to be made using these techniques.

6 Conclusion

In this paper we studied the quality of vector representations of words derived by various models on a collection of syntactic and semantic language tasks. We observed that it is possible to train high quality word vectors using very simple model architectures, compared to the popular neural network models (both feedforward and recurrent). Because of the much lower computational complexity, it is possible to compute very accurate high dimensional word vectors from a much larger data set. Using the DistBelief distributed framework, it should be possible to train the CBOW and Skip-gram models even on corpora with one trillion words, for basically unlimited size of the vocabulary. That is several orders of magnitude larger than the best previously published results for similar models.

An interesting task where the word vectors have recently been shown to significantly outperform the previous state of the art is the SemEval-2012 Task 2 [11]. The publicly available RNN vectors were used together with other techniques to achieve over 50% increase in Spearman's rank correlation over the previous best result [31]. The neural network based word vectors were previously applied to many other NLP tasks, for example sentiment analysis [12] and paraphrase detection [28]. It can be expected that these applications can benefit from the model architectures described in this paper.

Our ongoing work shows that the word vectors can be successfully applied to automatic extension of facts in Knowledge Bases, and also for verification of correctness of existing facts. Results from machine translation experiments also look very promising. In the future, it would be also interesting to compare our techniques to Latent Relational Analysis [30] and others. We believe that our comprehensive test set will help the research community to improve the existing techniques for estimating the word vectors. We also expect that high quality word vectors will become an important building block for future NLP applications.

낮은 계산복잡도가
더 많은 데이터를
학습할 수 있게
함으로써 시너지효과

word2vec 시각화

https://projector.tensorflow.org/

https://word2vec.kr/search/

reference

위키독스 딥 러닝을 이용한 자연어처리 입문: https://wikidocs.net/book/2155

Efficient Estimation of Word representation in vector space: https://arxiv.org/pdf/1301.3781

RNN: https://buly.kr/6BxBg5D