

논문 발표 (1) - Efficient Estimation of Word Representations in Vector Space

Date	@September 7, 2021
Status	

Word Embeddings (Word2Vec, GloVe)



Efficient Estimation of Word Representations in Vector Space
벡터 공간에서의 단어 표현의 효율적인 추정

▼ paper

Efficient Estimation of Word Representations in Vector Space.pdf

https://drive.google.com/file/d/1nz_GpWxFFUvGUMMWgYAZzmYExVBlu4w/view?usp=sharing

arXiv:1301.3781v3 [cs.LG]

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 that large representations in training as much better representations, i.e. it takes less than a day to learn high quality word vectors from a 1.6 billion word data set. Furthermore, we show that these vectors provide state-of-the-art performance on our test set for measuring syntactic and semantic word similarity.

1 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, efficiency and the observation that simple models trained on large amounts of data, especially complex systems trained on less data. An example is the popular 'glove' model used for statistical language modeling - today, it is possible to train N-grams on virtually all available data (billions of words) [5].

However, the simple techniques are at their limits in many tasks. For example, the amount of relevant information data for automatic speech recognition is limited - the performance is mostly dominated by the size of high quality transcribed speech data (often just millions of words). In machine translation, the training corpora for many languages contain only a few millions 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.

With progress of machine learning techniques in recent years, it has become possible to train more complex models on much larger data sets, and they typically outperform the simple models. Probably

▼ References

[NLP 논문 리뷰] Efficient Estimation Of Word Representations In Vector Space (Word2Vec)

Archive Link Submit Date: Jan 16, 2013 one-hot encoding 방식은 word를 단순히 표현하는 방법이다. word 자체가 갖는 정보를 담고 있지 않고 단순히 index만을 담고 있는데, index 역시 word에 내재된 어떤 정보와도 관련이 없다. 본 논문에서는 word vector에 word 자체가 담고 있는 의미를 확실하게 담아내고자 했다.

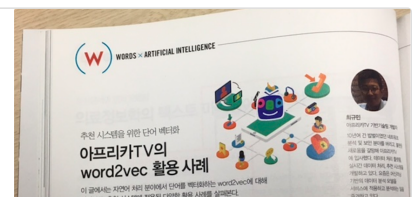
<https://cpm0722.github.io/paper-review/efficient-estimation-of-word-representations-in-vector-space>



Word2Vec 그리고 추천 시스템의 Item2Vec

마이크로소프트웨어 시특집 복간호 기고글 | 개발자들의 마소(마이크로 소프트웨어 잡지)가 다시 발간되었다. 우연히 기회가 되어 재 발간되는 마소에 나의 글을 기고할 수 있게 된 것은 가문의 영광이다. 이 번호 테마는 개발자의 인공지능(AI)이다. 글 쓸 때마다 느끼지만, 가장 좋은 학습 방

<https://brunch.co.kr/@goodvc78/16>



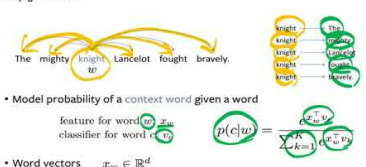
05-2: Text Representation II - Distributed Representation Part 2 (Word2Vec)

고려대학교 산업경영공학과 일반대학원 Unstructured Data

Analysis05_Text_Representation_II_Distributed Representation_Part_2_KorWord2Vec, CBOW, Skip-gram, Gradient ascent <https://...>

<https://youtu.be/s2KePv-OxZM>

• Skip-gram model



• Model probability of a context word given a word

$$p(c|w) = \frac{\exp(\mathbf{w}_c \cdot \mathbf{w}_w)}{\sum_{k=1}^K \exp(\mathbf{w}_k \cdot \mathbf{w}_w)}$$

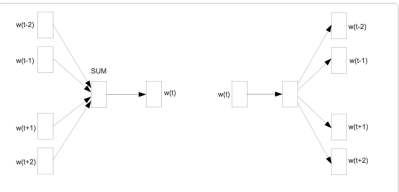
• Word vectors

$$\mathbf{w}_w \in \mathbb{R}^d$$

[논문 번역] Efficient Estimation of Word Representations in Vector Space 번역

Word2Vec 관련 논문을 직역한 것입니다. (의역이 아닙니다!) 짧은 영어라 오역이 있을 수 있으니 발견시 피드백 부탁드립니다. 초록 우리는 대량의 데이터로부터 계산된 단어의 연속적인 벡터 표현을 위한 새로운 모델 구조를 제안한다. 이 표현들의 성능은 단어 유사도로 측정되며 이 결과

<https://medium.com/@mldevhong/efficient-estimation-of-word-representations-in-vec>



개념 요약

- 해당 논문 발표 전까지 사용되던 NNLM(Neural Net Language Model)을 기반으로 대량의 문서 코퍼스를 엄청난 학습 속도와 성능을 개선한 SOTA 수준의 혁신적인 자연어 처리 기술이다.
- distributed representation
 - 분포 가설(distributional hypothesis)를 이용한다.
- 간단한 FFN(Feed Forward Network)을 구현하여 은닉층의 값을 그 단어를 설명하는 벡터 값으로 사용한다.
- 학습 결과 벡터로 표현된 단어들 중 비슷한 단어들은 벡터 공간 내에서 가까운 곳에 위치하게 된다.
- 단어를 벡터로 표현했기 때문에 단어(벡터) 끼리의 연산이 가능하다.

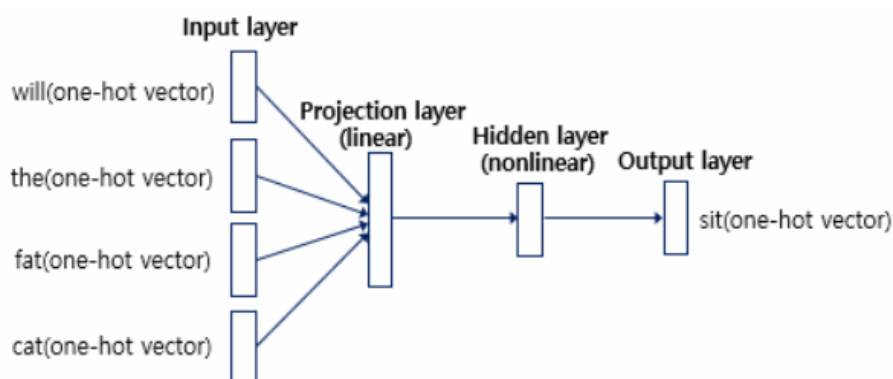
Model Architectures

- 여러 모델들을 비교하기 위해 computational complexity 를 정의한다.

$$\text{training complexity} = E \times T \times Q$$

- E : number of epochs
 - T : number of training dataset
 - Q : 모델에서 specific 하게 정의된 value
- 기호 정리
 - N : input word 갯수
 - V : vocabulary size
 - D : word representation dimension
 - H : hidden layer size

Feedforward Neural Net Language Model (NNLM)



- 일반적인 feed forward neural network 를 사용한 language model 이다.
- 이전 단어들 중 N 개의 word에 대한 one-hot encoding 을 input 으로 받는다.
- computational complexity

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

- 위 수식에서 가장 부하가 큰 연산은 hidden layer 에서 output을 만들어내는 연산인 $H \times V$ 이다.
- 하지만 이는 hierarchical softmax 를 사용하게 되면 $H \times \log_2 V$ 로 연산량을 줄일 수 있다.
- 이 경우 가장 부하가 큰 연산은 projection layer 에서 hidden layer를 만들어내는 연산인 $N \times D \times H$ 이다.

Recurrent Neural Net Language Model (RNNLM)

- RNN 을 사용한 Language Model 이다.
- RNN의 경우에는 projection layer 를 제거한다.
- 또한 고정된 개수의 word만을 input 으로 받는 NNLM 과 달리 이전의 모든 word들에 대한 정보를 recurrent 하게 hidden layer 에 담는다.
- computational complexity

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

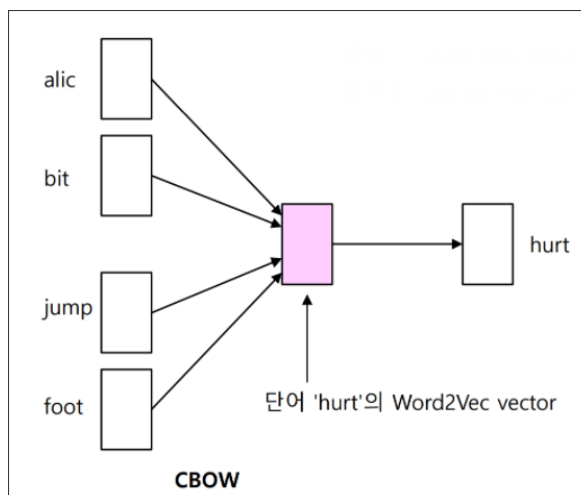
- D 와 H 를 동일하게 만들었기 때문에 위와 같은 수식이 된다.
- NNLM 과 마찬가지로 hierarchical softmax 를 사용하면 $H \times V$ 를 $H \times \log_2 V$ 로 줄일 수 있다.
- 이 경우 가장 부하가 큰 연산은 $H \times H$ 이다.

New Log-linear Models

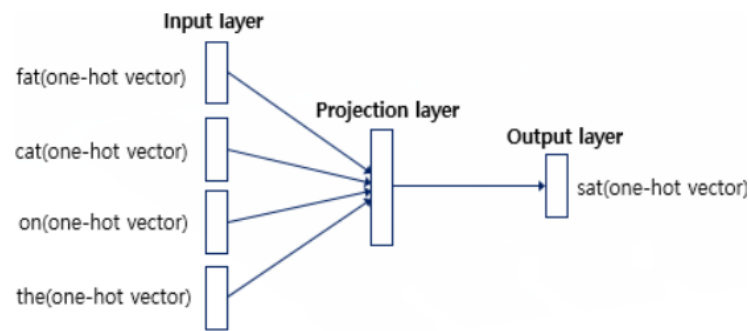
- computational complexity 를 줄이기 위해 2가지 단계를 제안한다.
 - CBOW (Continuous Bag-of Words)
 - Skip-gram
- 해당 모델들은 활성화 함수가 없는 네트워크 구조 (linear Neuron) 이다.
- 기존에 사용되던 모델에 비해 계산 복잡도를 최소화하는 모델이다.

CBOW

- 주변 단어를 입력으로 사용하고 중심 단어를 예측하는 모델을 생성한다.



- NNLM 에서 hidden layer 를 제거한 구조이다.



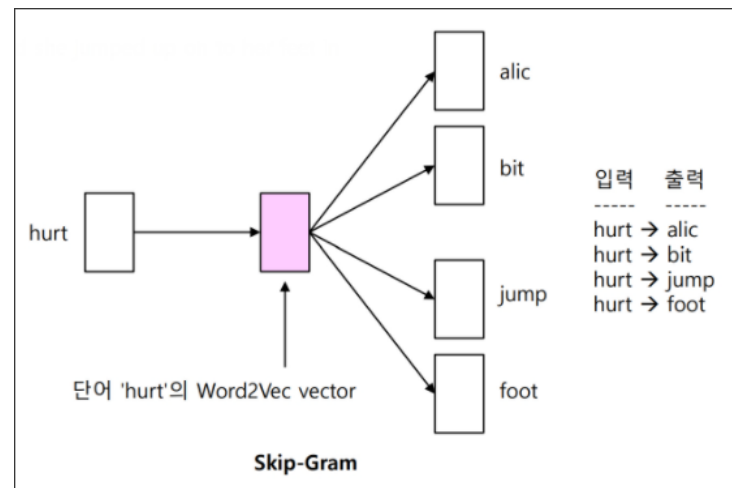
- NNLM 이 이전 단어들을 사용해 다음 단어를 예측한 것과 달리 양방향의 단어를 사용하여 예측을 진행한다.
- computational complexity

$$Q = N \times D + D \times \log_2 V$$

- hierarichcial softmax 를 사용한다는 가정 하에 가장 연산량이 많이 소요되는 hidden layer 를 제거해 전체 연산량을 줄였다.
- NNLM 에서는 여러 word embedding vector를 하나의 vector로 압축하기 위해 hidden layer 를 사용했다.
- CBOW 에서는 이를 non-linear layer 를 거치지 않고 단순하게 평균을 내게 된다.

Skip-gram

- 중심 단어를 입력으로 사용하고 주변 단어를 예측하는 모델을 생성한다.



- Skip-gram 은 input word vector를 평균내지 않고 온전히 사용하기 때문에 등장 빈도가 낮은 단어들에 대해 CBOW 대비 train 효과가 크다는 장점이 있다.
- computational complexity

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

- C : 예측할 단어의 개수와 관련된 값
- CBOW 보다 Skip-gram이 gradient flow 관점에서 더 좋은 성능을 나타낸다.
 - CBOW 는 하나의 단어를 가지고 주변 단어들에게 gradient 를 전달한다.

- Skip-gram 은 주변 단어들의 모든 gradient 를 하나의 단어가 전달받게 되므로 해당 방식이 performance 가 더 좋다.

수행 과정

- 다음과 같은 문장이 있다고 가정하자.
 - "I love you very much."
- 중심 단어(center word)의 주변 단어(context word)를 이용하여 해당 단어를 표현할 수 있다. (skip-gram 방식)
- 먼저 window size 를 설정하여 단어 집합을 구성한다.
 - window size 를 1로 했을 때, 중심 단어와 좌우의 주변단어 하나씩을 포함하여 총 3개의 단어가 하나의 집합으로 구성된다.
 - I, love, you
 - love you, very
 - you, very, much
- 가운데 단어를 기준으로 학습 데이터를 구성한다.
 - I, love, you
 - love → I
 - love → you
 - love you, very
 - you → love
 - you → very
 - you, very, much
 - very → you
 - very → much
- 이와 같은 학습 데이터를 사용하여 FFN 을 학습 시킨다.
- 학습된 FFN의 hidden layer 의 값을 해당 단어의 벡터(vector) 값으로 사용한다.

단어 사이의 연산

- 단어를 벡터로 표현했기 때문에 단어 간의 연산이 가능하다.

```
vector("King") - vector("Man") + vector("Woman") = "Queen"
```

논문 구현 방법

- 문장 데이터를 불러온다.
- 문장 데이터에 대해 전처리를 수행한다.
 - 불용어 처리, stemming 등

- 단어 사전(vocabulary) 를 생성한다.
- 각 문장들을 window size 를 지정하여 단어 집합을 생성한다.
 - trigram 일 경우 3개의 단어 씩 하나의 집합으로 생성됨
- 단어 집합들을 이용하여 학습 데이터를 구축한다.
 - skip-gram 방식을 사용할 경우, 중심 단어를 입력으로 사용하고 2개의 주변 단어를 출력으로 사용하여 학습 데이터를 구축한다.
- 학습 데이터에 있는 단어들을 단어사전을 이용하여 각각 인덱스로 표현해준다.
- 인덱스로 표현한 단어를 one-hot encoding vector 로 표현해준다.
 - 해당 과정은 네트워크에서 Embedding layer 를 사용할 경우 생략이 된다.
- FFN(Feed Forward Network) 를 구현한다.
 - 입력층과 출력층의 크기는 단어사전의 크기와 동일하다.
 - 은닉층의 크기는 단어를 표현하는 벡터의 크기로 결정되며 이는 하이퍼파라미터이다.
- 2가지 모델을 생성한다.
 1. 전체 네트워크를 학습하는 모델
 2. 은닉층의 출력을 얻을 수 있는 모델 (단어가 벡터로 변환된 값을 출력)