Efficient Estimation of Word Representations in Vector Space

Abstract

우리는 매우 큰 데이터 집합으로부터 단어의 연속 벡터 표현을 계산하기 위한 두 개의 새로운 모델 구조를 제안합니다. 이러한 표현의 품질은 단어 유사성 작업에서 측정되며 결과는 다양한 유형의 신경 네트워크를 기반으로 이전에 수행 된 최상의 기술과 비교됩니다. 훨씬 낮은 계산 비용으로 정확도가 크게 향상되었습니다. 즉, 16억 단어 데이터 세트에서 고품질 단어 벡터를 학습하는 데 하루가 걸리지 않습니다. 또한 이들 벡터가 구문론적, 통사론적 단어 유사성을 측정하기위한 테스트 세트에서 최첨단 성능을 제공함을 보여줍니다.

1 Introduction

이전의 NLP시스템은 단어를 하나의 atomic unit으로 취급함. 즉, 단어간의 유사성이라는 개념이 없음. 이 방법은 간단하고, 강고하며 엄청 많은 데이터를 가지고 학습시킨다면, 복잡한 모델로 적은 데이터를 학습시키는 것 보다 더 나은 결과를 보여줌. 한 예시는 N-gram model임.

하지만 이는 한계가 있음. 결국 작은 데이터에도 좋은 output을 얻어야 하기 때문. 이런 이유때문에 word2vec이 필요함.

https://shuuki4.wordpress.com/2016/01/27/word2vec-%EA%B4%80%EB%A0%A8-%EC%9D%B4%EB%A1%A0-%EC%A0%95%EB%A6%AC/ (https://shuuki4.wordpress.com/2016/01/27/word2vec-%EA%B4%80%EB%A0%A8-%EC%9D%B4%EB%A1%A0-%EC%A0%95%EB%A6%AC/) 참고; 정리를 잘 해놓음

1.1 Goals of the Paper

이 논문의 주요 목표는 수십억 단어의 어휘와 수백만 단어의 거대한 데이터 세트에서 고품질 단어 벡터를 학습하는 데 사용할 수있는 기술을 소개하는 것입니다. 우리가 아는 한, 이전에 제안 된 아키텍처 중 어느 것도 50-100 사이의 단어 벡터의 적당한 차원을 가지고 수억 개 이상의 단어에 대해 성공적으로 훈련 된 사례가 없습니다.

우리는 **유사 단어가 서로 가까울 뿐만 아니라, 그 단어들이 여러 층위의 유사도를 가질 수 있다**는 기대와 함께 결과 벡터 표현의 품질을 측정하기 위해 최근에 제안된 기술을 사용합니다. 예를 들어, 명사는 여러 개의 단어 어미를 가질 수 있으며, 원래의 벡터 공간의 부분 공간에서 비슷한 단어를 검색하면 유사한 어미가있는 단어를 찾는 것이 가능합니다 [13 , 14].

다소 놀랍게도, 단어 표현의 유사성은 단순한 구문 규칙성을 뛰어 넘는 것으로 나타났습니다. 간단한 대수 연산이 단어 벡터에서 수행되는 단어 오프셋 기법을 사용하면 벡터 ("King") - 벡터 ("Man") + 벡터 ("Woman")가 벡터에 가장 가까운 단어 "여왕" 을 표현합니다.

이 논문에서는 단어 사이의 선형 규칙성을 유지하는 새로운 모델 구조를 개발함으로써 이러한 벡터 연산의 정확성을 극대화하려고 노력합니다. 구문 및 의미 규칙 성을 측정하기 위한 새로운 포괄적 인 테스트 세트를 설계하고 이러한 규칙성을 높은 정확도로 학습 할 수 있음을 보여줍니다. 또한 훈련 시간과 정확도가 단어 벡터의 차원과 훈련 자료의 양에 어떻게 의존 하는지를 논의합니다.

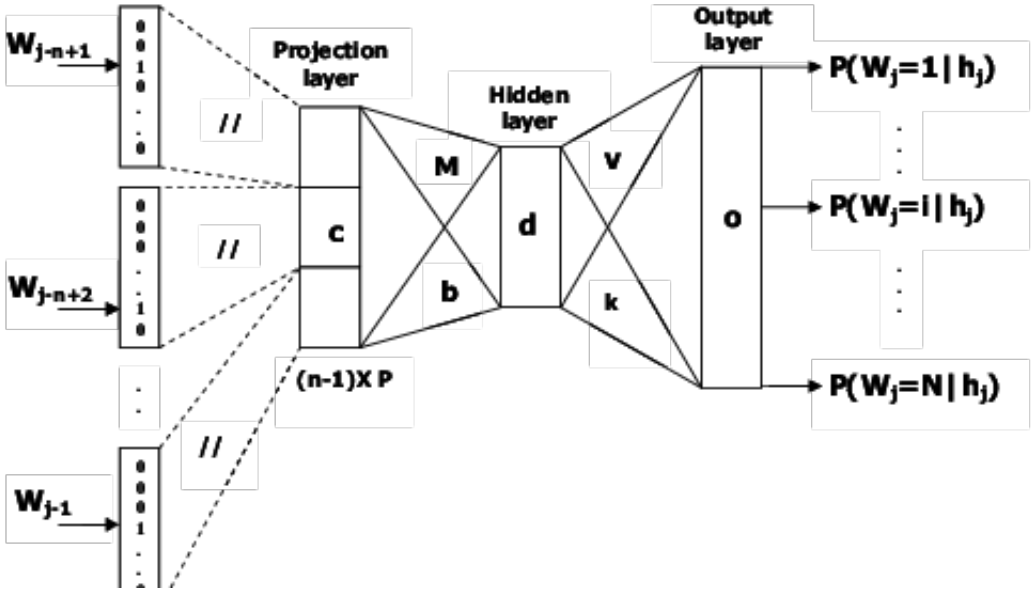
2. Model Architectures

1. Distributed representations of words learned by neural networks
2. Computational complexity = the number of parameters
3. Maximizing accuracy while minimizing the computational complexity

$$O = E \times T \times Q$$
$$O : training\ complexity$$
$$E : number\ of\ training\ epochs$$
$$T : number\ of\ words\ in\ the\ training\ set\ Q$$

결국, 단어를 벡터화하기위한 작업들.

2.1 Feedforward Neural Net Language Model (NNLM)




```
input layer:N previous words are encoded using 1-of-V coding, where V i
s size of the vocabulary.
```

```
projection: dimensionality = N * D = P
```

(The projection layer maps the discrete word indices of an n-gram context to a continuous vector space. Each neuron in the projection layer is represented by a number of weights equal to the size of the vocabulary) https://www.quora.com/What-is-the-difference-between-hidden-layer-and-projection-in-neural-networks (https://www.quora.com/What-is-the-difference-between-hidden-layer-and-projection-in-neural-networks) 참고

```
hidden layers: H
```

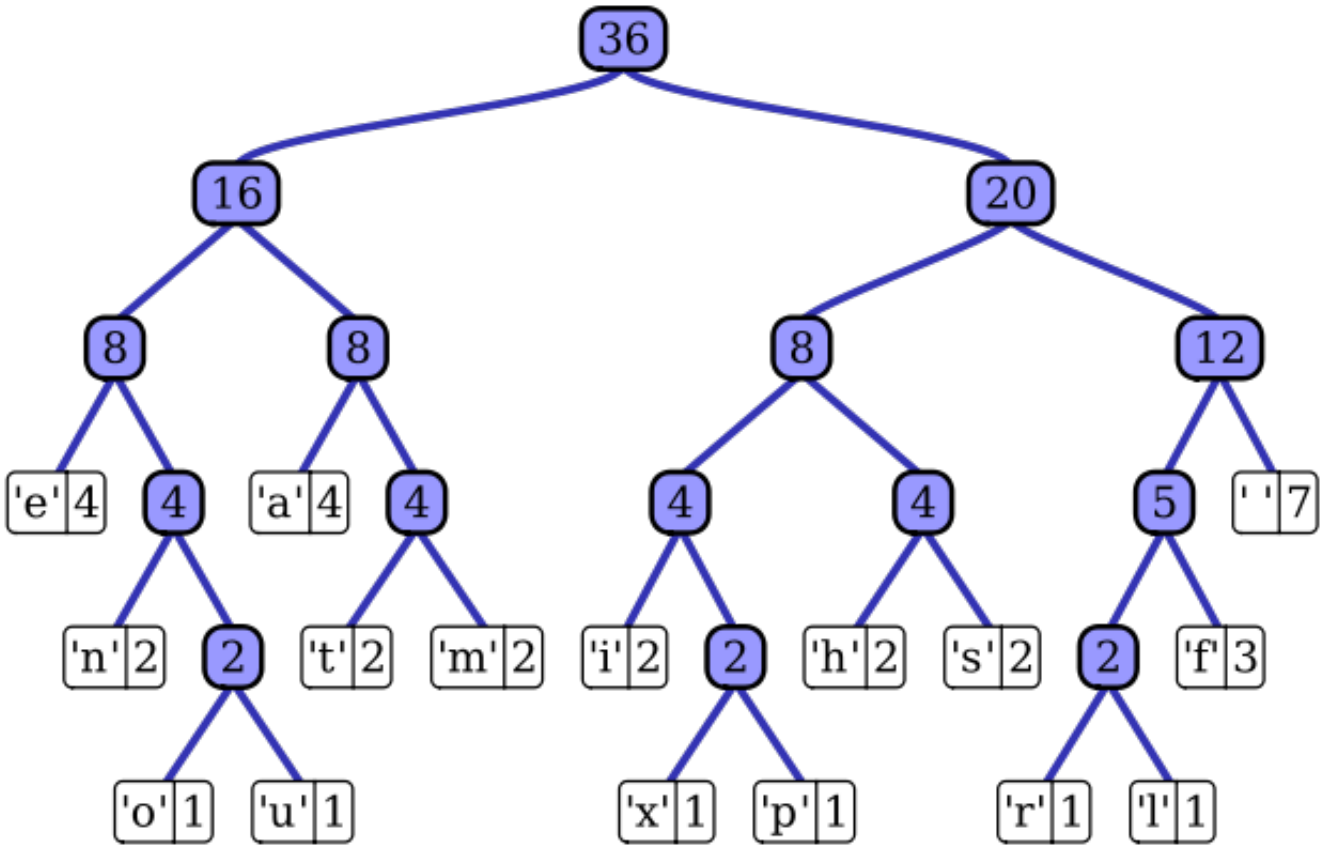
```
output layers: V
```

$$Q = N \times D + N \times D \times H + H \times V$$
$$O : training\ complexity$$
$$E : number\ of\ training\ epochs$$
$$T : number\ of\ words\ in\ the\ training\ set\ Q$$

dominating term: H * V

```
In this Model: hierarchical softmax where the vocabulary is represented
as a Huffman binary tree.
```

Huffman binary tree:



Hierarchical softmax: NNLM의 경우 모든 단어에 대한 조건부 확률을 계산해야 했음(과거 단어들 보고): $p(w|history), w\ in\ Vocab$ -> 시간 너무 오래걸림.

따라서 단어들의 구조를 Huffman tree로 바꾸고, 그 tree에서 각 타겟 단어를 찾는 과정을 변환시킴; 예를 들어, 왼쪽 서브트리는 0.7, 오른쪽은 0.3확률로 가는 식. 이는 곧 normalization step이라고..

“In probabilistic terms, one N-way normalization is replaced by a sequence of O(logN) local (binary) normalizations.“

```
the Huffman tree based hierarchical softmax requires only about log_2 ( Unigram perplexity(V ))
```

결국 이로 인해서 문제가 되었던 `H * V`부분을 `H * log(V)`로 빨리 계산할수 있게 만듬.

2.2 Recurrent Neural Net Language Model (RNNLM)

NNLM의 한계들을 극복하기 위해 제시됨. NNLM에서는 N(context length)을 특정해 주어야 했음. RNN모델은 projection layer가 없음. 여기서 **특수한 점은 히든레이어를 자기 자신과 연결하는 recurrent matrix를 사용한다는 점** (time-delayed connections 쓰면서). 이는 곧 short term memory를 만드는 것과 비슷함.

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

where word representations D have the same dimensionality as the hidden layer H. H * V 또한 H * log(V)로 변환 가능.

2.3 Parallel Training of Neural Networks

DistBelief라는 프레임워크 사용함.

For this parallel training, we use mini-batch asynchronous gradient descent with an adaptive learning rate procedure called Adagrad.

3 New Log-linear Models

대부분의 complexity는 모델의 non-linear hidden layer에서 옴. 그래서 더 심플한 모델을 쓰기로 함. 이전 페이퍼들에서 NN 언어 모델은 두가지 과정으로 잘 학습될 수 있었음. 첫번째, continuous word vectors를 심플한 모델로 학습시키고, 이런 distributed representations of words 위에 N-gram NNLM모델을 쓰는 것임. 하지만 여기에서는 < T. Mikolov. Language Modeling for Speech Recognition in Czech, Masters thesis, Brno University of Technology, 2007. >에서 제시된 방법을 쓰겠음.

3.1 CBOW(Continuous Bag-of-Words Model)

여기서 non-linear 히든레이어가 제거되고 projection layer가 모든 단어에 대해 공유된다; 따라서 모든 단어가 같은 position으로 projected된다. 우리는 이를 단어의 순서가 projection에 영향을 끼치지 않는 아키텍처라고 부른다. 더해서, 우리는 미래(뒷문장을 말하는 듯)의 단어 또한 이용한다; 우리는 4개의 전 단어, 4개의 뒷 단어를 인풋으로 가지고 중간 단어를 예측하는 log-linear 분류기로 좋은 결과를 얻었다.

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

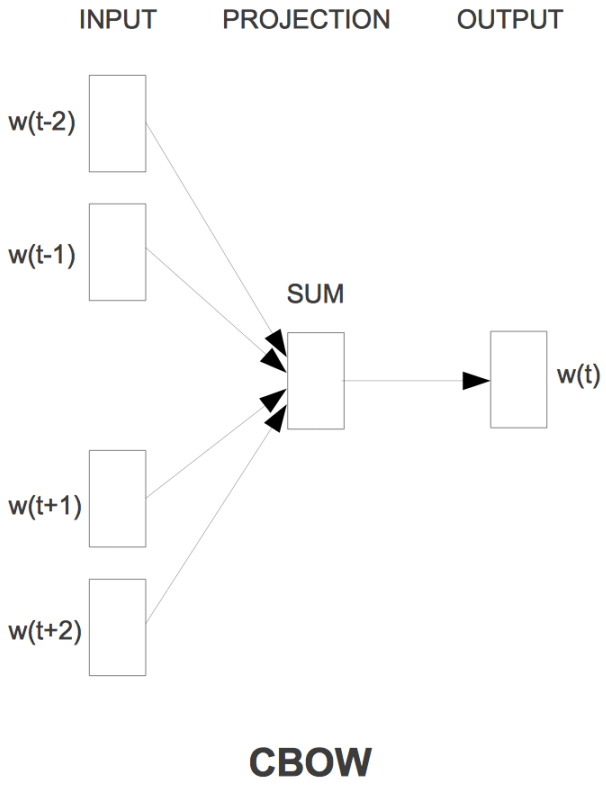
3.3 Continuous Skip-gram Model

이 모델은 같은 문장안에 있는 다른 단어를 가지고 한 단어를 맞추는 것을 목표로(maximizing) 한다. 더 정확하게 말하면, 각 단어를 continuous projection layer를 가지는 log-linear 분류기의 인풋으로 두고, 그 단어의 앞뒤 단어들을 예측하게 한다. 이 range를 늘리면 word vector의 퀄리티는 좋아졌지만, complexity는 높아진다. 더 멀리 떨어져 있는 단어에게 적은 weight를 주었다.

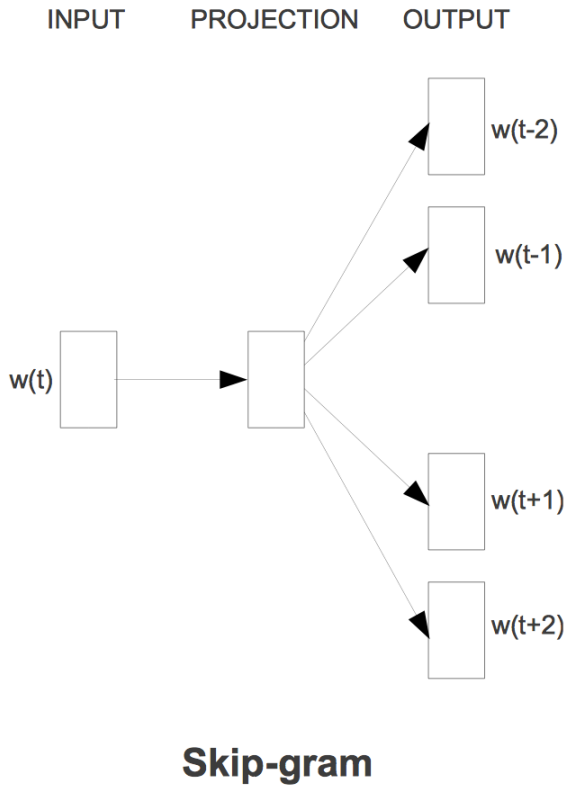
$$Q = C \times (D + D \times log_2(V))$$

`C = maximum distance of words,` 여기선 `C = 5`

그 후 1:C (1:C)range에서 랜덤하게 숫자 R을 뽑고, 단어 앞뒤로 R개의 단어를 correct label로 사용했다. 즉 이는 R * 2 의 단어 분류기가 필요하다(input: current word). 그리고 output은 R+R이다.



CBOW



Skip-gram

4. Results

다른 버전의 단어 벡터의 품질을 비교하기 위해 이전 논문은 일반적으로 예제 단어와 가장 유사한 단어를 보여주는 표를 사용하고 이를 직관적으로 이해합니다. 프랑스라는 단어가 이탈리아와 다른 나라와 비슷하다는 것을 보여주기는 쉽지만, 다음과 같이 좀 더 복잡한 유사성 과제에 해당 벡터를 적용하는 것이 훨씬 더 어렵습니다. 우리는 단어 사이에 많은 다른 유형의 유사점이 있을 수 있다는 이전의 관찰을 따릅니. 예를 들어, big은 bigger과 small-smaller의 관계와 같은 의미로 비슷합니다. 다른 유형의 관계의 예는 큰 단어 쌍 (big - most and small - smallest [20]) 일 수 있습니다. 우리는 질문과 같은 관계를 가진 두 쌍의 단어를 더 표시합니다. "big과 biggest의 관계를 가지는 small과 비슷한 단어는?"

다소 놀랍게도 이러한 질문은 단어의 벡터 표현으로 간단한 대수 연산을 수행하여 대답 할 수 있습니다. 가장 큰 것과 같은 의미로 small과 비슷한 단어를 찾기 위해 vector X = vector ("biggest") - vector ("big") + vector ("small")를 간단히 계산할 수 있습니다. 그런 다음 코사인 거리로 측정 한 X에 가장 가까운 단어를 벡터 공간에서 검색하고 이를 질문에 대한 답변으로 사용합니다 (이 검색 중에 입력된 질문 단어는 무시합니다). 단어 벡터가 잘 훈련되면 이 방법을 사용하여 정답 (smallest)을 찾는 것이 가능합니다.

마지막으로, 우리는 많은 양의 데이터에 고차원의 단어 벡터를 훈련시킬 때 결과 벡터를 사용하여 도시와 그것이 속한 국가와 같은 단어 사이의 매우 미묘한 의미론적 관계에 대답 할 수 있음을 발견했습니다. ex) 프랑스 - 파리, 독일 - 베를린. 이러한 의미론적 관계가 있는 단어 벡터는 기계 번역, 정보 검색 및 질문 응답 시스템과 같은 기존의 많은 NLP 응용 프로그램을 개선하는 데 사용될 수 있으며 아직 다른 미래의 응용 프로그램을 아직 개발할 수 있습니다.

!texte (4_table1.png) <img src = "4_table1.png">

4.1 Task Description

단어 벡터의 품질을 측정하기 위해 우리는 다섯 가지 유형의 의미 론적 질문과 9 가지 유형의 구문 질문을 포함하는 포괄적인 테스트 세트를 정의합니다. 각 카테고리의 두 가지 예가 표 1에 나와 있습니다. 전체적으로 8869 개의 의미론적 질문과 10675 개의 문법적 질문이 있습니다. 각 카테고리의 질문은 두 단계로 작성되었습니다. 먼저 유사한 단어 쌍 목록을 수동으로 작성했습니다. 그런 다음 두 단어 쌍을 연결하여 많은 질문 목록을 만듭니다. 예를 들어 우리는 68 개의 미국 대도시와 그들이 속한 주에 대한 목록을 만들었고 무작위로 2 개의 단어 쌍을 선택하여 약 2.5K 개의 질문을 만들었습니다. 테스트 세트에는 단일 토큰 단어 만 포함되어 있으므로 다중 단어 엔터티가 없습니다 (예 : New York).

우리는 모든 질문 유형에 대해 그리고 각 질문 유형별로 (의미론적, 통사론적으로) 전반적인 정확도를 평가합니다. 질문은 위의 방법을 사용하여 계산 된 벡터에 가장 가까운 단어가 문제의 정확한 단어와 정확히 일치하는 경우에만 올바르게 대답한다고 가정합니다. 따라서 동의어는 실수로 간주됩니다. 또한 현재 모델에는 단어 형태에 대한 입력 정보가 없으므로 100 % 정확도에 도달하는 것은 불가능할 수 있음을 의미합니다. 그러나 특정 응용 프로그램에 대한 단어 벡터의 유용성은 이 정확성 메트릭과 양의 상관 관계가 있어야한다고 생각합니다. 단어의 구조, 특히 구문론적 질문에 대한 정보를 통합하여 더 많은 발전을 이룩할 수 있습니다.

4.2 Maximization of Accuracy

Google은 단어 벡터 교육을 위해 Google 뉴스 자료를 사용했습니다. 이 코퍼스에는 약 6B 개의 토큰이 있습니다. 우리는 어휘 크기를 1 백만 개의 가장 빈번한 단어로 제한했습니다. 분명히 우리는 시간 제약적인 최적화 문제에 직면하고 있습니다. 더 많은 데이터와 더 높은 차원의 단어 벡터를 사용하면 정확도가 향상 될 것으로 예상되기 때문입니다. 가능한 최상의 결과를 얻기위한 최선의 모델 아키텍처를 평가하기 위해 가장 빈번한 30k 단어로 제한된 어휘를 사용했습니다. 단어 벡터 차원을 다르게하고 훈련 데이터의 양을 늘린 CBOW 아키텍처를 사용한 결과는 표 2와 같습니다.

어느 시점 이후에 더 많은 차원을 추가하거나 더 많은 교육 데이터를 추가하면 점차 개선되는 것을 볼 수 있습니다. 따라서 우리는 벡터 차원과 훈련 데이터의 양을 함께 늘려야합니다. 상대적으로 많은 양의 데이터에 대해 단어 벡터를 훈련시키는 것은 현재 인기가 있지만, 크기가 불충분 한 경우 (예 : 50 - 100)에 주목해야합니다. 수학 식 4가 주어지면, 학습 데이터의 양이 두 배가되면 벡터 크기를 두 배로 증가시킬 때와 거의 같은 계산 복잡성이 증가하게 됩니다.

표 2와 4에서 보고 된 실험에서 SGD와 역전파, 그리고 epoch=3으로 두었습니다. learning rate= 0.025를 선택하고 선형적으로 줄였습니다. 따라서 이는 마지막 훈련이 끝날 때 0에 접근합니다.

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

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

4.3 Comparison of Model Architectures

먼저 동일한 학습 데이터를 사용하여 단어 벡터를 유도하고 단어 벡터 중 640 개의 동일한 차원을 사용하는 여러 모델 아키텍처를 비교합니다. 추가 실험에서, 우리는 새로운 Semantic-Syntactic Word Relationship 테스트 세트, 즉 30k 어휘에 제한되지 않은 완전한 세트의 질문을 사용합니다. 우리는 [3] 단어들 사이의 구문 유사성에 초점을 맞춘 [20]에서 소개 된 시험 세트에 대한 결과도 포함합니다.

훈련 데이터는 여러 LDC 자료로 구성되며 [18] (320M 단어, 82K 어휘)에 자세히 설명되어 있습니다. 우리는 이 데이터를 사용하여 단일 CPU에서 학습하는 데 약 8 주가 걸린 이전에 훈련된 신경망 언어 모델을 비교했습니다. 8 개의 이전 단어의 히스토리를 사용하여 DistBelief 병렬 학습 [6]을 사용하여 동일한 수의 640 개의 숨겨진 유닛으로 피드 포워드 NNLM을 교육했습니다 (따라서 NNLM은 RNNLM보다 많은 매개 변수를 갖습니다. 투영 레이어의 크기는 640 × 8).

표 3에서, RNN으로부터의 단어 벡터 ([20]에서 사용 된)는 대부분 문법적 질문에서 잘 수행된다는 것을 알 수있다. NNLM 벡터는 RNN보다 훨씬 뛰어납니다. RNNLM의 단어 벡터가 비선형 히든 레이어에 직접 연결되어 있기 때문에 놀라운 것은 아닙니다. CBOW 아키텍처는 구문 작업에서 NNLM보다 잘 작동하며 의미론적 작업에서는 NNLM과 거의 동일합니다. 마지막으로, Skip-gram 아키텍처는 CBOW 모델보다 문법 작업에서 약간 더 잘 작동하지만 (다른 NNLM보다 훨씬 좋음) 의미론 부분에서 훨씬 뛰어납니다.

다음으로 우리는 하나의 CPU만을 사용하여 훈련 된 모델을 평가하고 공개적으로 사용 가능한 단어 벡터와 결과를 비교했습니다. 비교는 표 4에 나와 있습니다. CBOW 모델은 약 하루 만에 Google 뉴스 데이터의 하위 집합에 대해 훈련 받았지만 Skip-gram 모델의 훈련 시간은 약 3 일이었습니다.

이후 실험에서 우리는 one traning epoch를 사용했습니다 (다시 말하면, 훈련의 끝에서 0에 가까워 지도록 학습 속도를 선형적으로 감소시킵니다). 하나의 epoch를 사용하여 두 배의 데이터를 모델링하는 것은 표 5에 나와있는 것처럼 세 번의 에포크에 대해 동일한 데이터를 반복하는 것보다 비슷하거나 더 나은 결과를 제공하며 추가로 빠른 속도 향상을 제공합니다.

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**

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

4.4 Large Scale Parallel Training of Models

앞서 언급했듯이 DistBelief라는 분산 프레임 워크에서 다양한 모델을 구현했습니다. 아래는 Google News 6B 데이터 세트에서 훈련 된 여러 모델의 결과를 mini-batch asynchronous gradient descent와 Adagrad를 썼습니다. 우리는 훈련 도중 50-100 개의 모델 복제본을 사용했습니다. CPU 코어의 수는 데이터 센터 컴퓨터가 다른 생산 작업과 공유되기 때문에 추정치이며, 추정치가 상당히 바뀔 수 있습니다. 분산된 프레임 워크의 오버헤드로 인해 CBOW 모델과 Skip-gram 모델의 CPU 사용량은 단일 시스템 구현보다 서로 훨씬 비슷합니다.

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

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**

4.5 Microsoft Research Sentence Completion Challenge

Microsoft Sentence Completion Challenge는 언어 모델링 및 기타 NLP 기술 향상을 위해 최근에 생겼습니다 [32]. 이 작업은 1040 문장으로 구성되어 있습니다. 각 문장마다 한 단어가 빠져 있고 목표는 나머지 문장과 가장 일관된 단어를 선택하는 것입니다. N-gram 모델, LSA 기반 모델 [32], 로그 쌍 선형 모델 (log-bilinear model) [24], 현재 최신 성능을 보유하고 있는 뉴럴 네트워크의 조합을 포함하여 여러 가지 기술의 성능이 이미 이 세트에서 보고되었습니다. 이 벤치 마크에서 55.4 %의 정확도가 거의 최고입니다 [19].

우리는이 작업에서 Skip-gram 아키텍처의 성능을 탐구했습니다. 먼저, [32]에 제공된 50M 단어에 대해 640 차원 모델을 학습합니다. 그런 다음 입력에서 알 수없는 단어를 사용하여 테스트 집합에서 각 문장의 점수를 계산하고 문장에서 주변 단어를 모두 예측합니다. 최종 문장 점수는 이러한 개별 예측의 합계입니다. 문장 점수를 사용하여 가장 가능성이 높은 문장을 선택합니다.

Skip-gram 모델 자체가 LSA 유사성보다 이 작업을 더 잘 수행하지는 않지만 이 모델의 점수는 RNNLM을 사용하여 얻은 점수와 보완적이며, 새로운 결과와 함께 이전 결과에 대한 간략한 요약이 표 7에 나와 있습니다. 가중 조합은 새로운 결과를 58.9 %의 정확도로 나타냅니다 (세트의 개발 부분에서는 59.2 %, 테스트 부분에서는 58.7 %).

5 Examples of the Learned Relationships

표 8은 다양한 관계를 따르는 단어를 보여줍니다. 우리는 위에서 설명한 접근법을 따릅니다. 두 단어 벡터를 뺀 관계를 정의하고 그 결과를 다른 단어에 더합니다. 예를 들어 파리 - 프랑스 + 이탈리아 = 로마. 분명히 볼 수 있듯이, 명확한 개선이 필요한 부분이 많았지만 정확성은 상당히 향상되었습니다 (정확히 일치한다고 가정하는 우리의 정확도 메트릭을 사용하면 표 8의 결과는 약 60 %에 불과합니다). 우리는 더 큰 차원의 더 큰 데이터 세트에 대해 훈련된 단어 벡터가 훨씬 더 잘 수행 될 것이며 새로운 혁신적인 응용 프로그램을 개발할 수 있다고 믿습니다. 정확성을 향상시키는 또 다른 방법은 관계에 대한 하나 이상의 예를 제공하는 것입니다. 관계 벡터를 형성하기 위해 하나의 예제 대신에 10 개의 예제를 사용했을때(개별 벡터를 평균) 의미론적, 구문적 테스트에서 절대적으로 약 10 %의 정확도 향상을 관찰했습니다.

벡터 작업을 적용하여 다른 작업을 해결할 수도 있습니다. 예를 들어, 목록 밖의 단어를 선택하거나, 단어 목록에 대한 평균 벡터를 계산하고, 가장 먼 단어 벡터를 찾는 등의 좋은 정확도를 보았습니다. 이것은 특정 인간 지능 검사에서 널리 사용되는 유형의 문제입니다. 분명히, 이러한 기술을 사용하여 만들어야 할 많은 발견이 아직도 있습니다.

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

Conclusion

우리는 인기있는 신경망 모델 (feedforwad, recurrent모두)과 비교하여 매우 간단한 모델 아키텍처를 사용하여 고품질 단어 벡터를 학습 할 수 있다는 것을 관찰했습니다. 계산 복잡도가 훨씬 더 낮기 때문에, 훨씬 더 큰 데이터 세트로부터 매우 정확한 고차원 워드 벡터를 계산하는 것이 가능하다. DistBelief 분산 프레임 워크를 사용하면 기본적으로 무제한 크기의 어휘에 대해 1 조 단어의 코퍼로 CBOW 및 Skip-gram 모델을 학습 할 수 있습니다. 비슷한 모델에 대해 이전에 발표 된 최상의 결과보다 몇 배 더 큰 수치입니다.