Efficient Estimation of Word Representations in

Vector Space

Abstract

우리는 매우 큰 데이터셋에 포함되어 있는 단어들의 연속적인 벡터 표현을 컴퓨팅하기 위한 두 개의 새로운 모델 아키텍쳐를 제안한다. 이러한 표현들의 퀄리티는 word similarity task로 측정되고, 결과는 다른 종류의 신경망을 기반으로 한, 이전의 최고 수준 기술과 비교한다. 우리는 훨씬 더 낮은 컴퓨팅 비용으로 큰 정확도 향상을 이뤄냈다. 1.6 billion words data set으로부터 양질의 워드 벡터를 학습하는 데에 하루가 채 걸리지 않았다. 더욱이, 우리는 이러한 벡터가 문법적・의미적 단어 유사도를 측정하기 위한 테스트 셋에서, state-of-the-art performance를 달성했음을 보인다.

1 Introduction

많은 현재의 NLP 시스템과 기술은 단어를 원자 단위로서 처리한다. 단어들 사이의 유사도에 대한 개념 없이, 이것들은 단어 내에서 인덱스로 표현된다. 이 선택은 몇 가지 괜찮은 이유를 갖는다. - 단순하고, 견고하다. 그리고 거대한 양의 데이터로 학습된 단순한 모델이 더 적은 데이터로 학습된 복잡한 데이터보다 더 좋은 성능을 낸다는 것을 보인다. 한 예시는 통계적 언어 모델링에 사용되는, 유명한 N-gram model이다. 오늘날, N-gram은 가상으로 모든 쓸 수 있는 데이터에 학습 시킬 수 있다. (Trillion of words [3])

하지만, 간단한 기술들은 많은 태스크에서 그들의 한계에 다달았다. 예를 들어, 자동 음성 인식을 위한 관련 도메인의 데이터의 양은 제한된다. 퍼포먼스는 보통 양질의 번역된 음성 데이터의 크기에 지배된다. (종종 몇백만 단어이기도 하다.) 기계 번역에서, 많은 언어에 대해 존재하는 corpora(말뭉치)는 몇십억개의 단어(or less) 정도만 포함한다. 그러므로, 기본적인 기술을 단순히 규모만 키우는 것은 큰 진전을 보이진 않을 것이다. 그리고, 우리는 더 진보된 기술에 주목해야만 한다.

최근 머신 러닝 기술의 발전으로, 더 복잡한 모델을 훨씬 더 큰 데이터셋에 대해 학습시키는 것이 가능해졌다. 그리고, 그들은 보통 단순한 모델보다 우수한 성능을 보인다. 아마도 가장 성공적인 개념은 단어의 분산된 표현을 사용하는 것이다.([10]) 예를 들어서, 언어 모델을 기반으로 하는 신경망은 N-gram 모델보다 상당히 더 좋은 성능을 보였다.([1], [27], [17])

1.1 Golas of the Paper

이 연구의 주된 목표는 수십억개의 단어와 수백만개의 어휘를 갖는 거대 데이터셋에서, 높은 퀄리티의 단어 벡터를 학습시키기 위해 쓰일 수 있는 기술을 도입하는 것이다. 우리가 아는 한, 이전에 제안된 아키텍쳐 중, 몇억 개의 단어에서, 50~100 사이의 적절한 단어 벡터 차원을 갖는, 성공적으로 학습된 것은 존재하지 않는다.

우리는 최근에 제안된 벡터 표현 결과의 퀄리티를 평가하는 기술을 사용한다. 우리는 비슷한 단어는 서로에게 가까운 거리에 있는 경향을 가질 뿐만 아니라, 여러 수준의 유사도(multiple degrees of similarity, [20])를 갖는다고 예상을 하였다.[2] 이것은 굴절어의 맥락에서 일찍이 발견되었다. 예를 들어, 명사는 다수의 단어 ending을 가질 수 있고, 만약 우리가 원래 벡터 공간의 서브 스페이스에서 비슷한 단어를 찾는다면, 비슷한 ending을 갖는 단어를 찾을 수 있을 것이다. ([13], [14])

놀랍게도, 단어 표현의 유사도는 단순한 문법적 regularity 넘어선다. 간단한 선형대수적 연산들이 단어 벡터에서 수행되는 word offset technique을 사용함으로써, 예시를 확인해볼 수 있다. vector(“King”) - vector(“Man”) + vector(“Woman”)의 결과는 단어 Queen의 벡터 표현과 가장 가까운 벡터이다. [20]

이 연구에서, 우리는 단어들 사이의 선형 regularity를 보존하는 새로운 모델 아키텍쳐를 개발함으로써 이러한 벡터 연산의 정확도를 최대화 하려 한다. 우리는 문법적, 그리고 의미적 regularity를 둘 다 측정하기 위한 새로운 종합적 테스트셋을 디자인한다. 그리고, 이러한 많은 regularity가 높은 정확도로 학습 가능함을 보인다. 더욱이, 우리는 학습 시간과 정확도가 단어 벡터의 차원과 학습 데이터의 양에 얼마나 의존적인지를 논의한다.

**1.2 Previous Work**

연속적인 벡터로서 단어의 표현은 긴 역사를 갖는다. ([10], [26], [8]) Neural Network Language Model (NNLM)을 평가하기 위한 매우 인기 있는 모델 아키텍쳐는 [1]에서 제안되었다. 선형 투사층과 비선형 은닉층을 갖는 feedforward neural network는 단어 벡터 표현과 통계적 언어 모델을 동시에 학습하기 위해 사용되었다.

NNLM의 또 다른 흥미로운 아키텍쳐는 [13], [14] 에서 소개되었다. 워드 벡터가 하나의 은닉층을 가진 신경망을 사용해서, 첫번째로 학습되었다. 단어 벡터들은 NNLM을 학습시키기 위해 사용되었다. 그러므로, 단어 벡터는 전체 NNLM을 구성하지 않고도 학습된다. 이 연구에서, 우리는 직접적으로 이 구조를 확장하고, 워드 벡터가 단순한 모델을 사용해서 학습되는 첫번째 단계에만 초점을 맞춘다.

단어 벡터는 많은 NLP 응용을 향상시키고, 간단하게 만들기 위해 사용될 수 있음이 드러났다. ([4], [5], [29]) 단어 벡터를 추정하는 것 자체는 다른 모델 아키텍쳐를 사용해서 수행되었고, 다양한 말뭉치로 학습되었다. ([4], [29], [23], [19], [9]). 그리고 결과 단어 벡터들의 일부는 추후 연구와 비교를 위해 사용될 수 있도록 만들어졌다. 하지만, 우리가 아는 한, 이러한 구조들은 [13] 에서 제안된 것보다 학습 비용이 상당히 비싸다. (대각선의 가중치 매트릭스가 사용된 log-bilinear model의 특정 버전을 제외하고)

**2 Model Architectures**

잘 알려진 Latent Semantic Analysis (LSA)와 Latent Dirichlet Allocation (LDA)를 포함하여, 많은 다른 종류의 모델들은 단어의 연속적인 표현을 추정하기 위해 제안되었다. 이 논문에서, 우리는 신경망에 의해 학습된 단어들의 분산된 표현(distributed representation)을 주목한다. 그들이 단어들 간의 선형 regularity를 보존하는 것에서 LSA보다 상당히 더 잘 수행한다는 것이 이전에 보여졌기 때문이다. ([20], [31]) LDA는 더욱이 큰 데이터셋에 대해 비용이 매우 비싸다.

[18]과 비슷하게, 다른 모델 아키텍쳐를 비교하기 위해, 우리는 첫번째로 모델의 컴퓨팅 복잡도를 모델을 완전히 학습시키기 위해 접근되어야 하는 파라미터의 개수로서 정의한다. 다음으로, 우리는 컴퓨팅 복잡도를 최소화함과 동시에, 정확도를 최대화할 것이다.

모든 다음의 모델들에 대해, 학습 복잡도는 아래 식에 비례하다.

(1)

E는 학습 epoch 수이고, T는 훈련 집합의 단어 수이다. Q는 각 모델 아키텍쳐에 대해 추가적으로 정의된다. 보통 E=(3, 50) 이고, T는 10억까지이다. 모든 모델은 SGD와 backpropagation을 사용하여 학습된다.

**2.1 Feedforward Neural Net Language Model (NNLM)**

확률적 피드포워드 신경망 언어 모델은 [1]에서 제안되었다. 그것은 input, projection, hidden, output 레이어로 이루어져 있다. 입력층에서, N개의 이전 단어들은 1-of-V 코딩을 사용하여 인코딩 된다. V는 vocabulary의 크기이다. 입력층은 그 다음에 N\*D 차원을 갖는 투사층 P로 투사된다. 이때, 공유된 projection matrix를 사용한다. 주어진 시간에 오직 N개의 입력만 활성화되기 때문에, 투사층의 구성은 상대적으로 싼 연산이다.

NNLM 구조는 투사층과 은닉층 사이의 컴퓨팅을 위해 더 복잡해진다. 투사층의 값이 dense하기 때문이다. 일반적으로 N=10을 고르면, P의 크기는 아마도 500~2000이 될 것이고, 은닉층 H는 보통 500~1000 개의 유닛이다. 더욱이, 은닉층은 vocabulary에서 전체 단어에 대한 확률 분포를 계산하기 위해 사용된다. 이는, 출력층의 차원이 V가 되도록 한다. 그러므로, 각 훈련 예시 당 컴퓨팅 복잡도는

(2)

이다. Dominating term은 H\*V이다. 하지만, 약간의 실용적 솔루션이 그것을 피하기 위해 제시되었다. softmax의 계층적 버전([25], [23], [18])을 사용하거나, 학습 동안 정규화되지 않은 모델을 사용함으로써, 정규화된 모델을 완전히 피하는 것이다.([4], [9]). vocabulary의 이진 트리 표현으로, 검사될 수 있는 출력 유닛의 개수는 약 log2(V)까지 내려간다. 그러므로, 복잡도의 대부분은 N\*D\*H에 의해 발생한다.

우리 모델에서, 우리는 vocabulary가 Huffman binary tree로서 표현되는 계층적 softmax를 사용한다. 이것은 단어의 빈도수가 NNLM에서 클래스를 얻는 것에 잘 작동한다는 이전의 관찰을 따른다.([16]) 허프만 트리는 자주 나오는 단어에 짧은 이진 코드를 할당한다. 그리고 이것은 평가되어야 하는 출력 유닛의 개수를 더 줄인다. 균형 이진 트리는 log2(V) 출력을 요구할 것이고, 계층적 소프트맥스를 기반으로 한 허프만 트리 오직 log2(Unigram\_perplexity(V))만을 요구할 것이다. 예를 들어서, vocabulary 사이즈가 100만일 때, evaluation에서 두 배 더 빠르다. 이것은 N\*D\*H 계산상의 병목 현상으로, NNLM에서 결정적인 속도 향상은 아니다. 우리는 나중에 은닉층을 갖지 않고, 따라서 softmax normalization의 효율에 상당히 의존적인 아키텍쳐를 제안할 것이다.

**2.2 Recurrent Neural Net Language Model (RNNLM)**

언어 모델을 기반으로 하는 RNN은, context 길이(모델 N의 order)를 명시해야 하는 것 같은 피드포워드 NNLM의 특정 한계점을 극복하기 위해 제안되었다. 이론적으로 RNN은 얕은 신경망보다 더 복잡한 구조를 효율적으로 표현할 수 있기 때문이다. ([15], [2]) RNN 모델은 투사층을 갖지 않는다. 오직 입력, 은닉, 그리고 출력층만 갖는다. 이러한 종류의 모델이 특별한 것은 time-delayed connection을 사용하여 은닉층을 그것 스스로와 연결하는 recurrent matrix이다. 이것은 recurrent model이 몇 가지 종류의 short term memory를 형성하도록 한다. 과거로부터의 정보는 현재 입력과 이전 시간 단계에서의 은닉층 상태에 의해 업데이트되는 은닉층 상태에 의해 표현될 수 있기 때문이다.

RNN의 모델의 훈련 예시 당 복잡도는

(3)

이다. 단어 표현 D는 은닉층 H와 같은 차원을 갖는다. 다시, H\*V는 계층적 softmax를 사용함으로써 H\*log2(V)로 효율적으로 감소시킬 수 있다. 복잡도의 대부분은 H\*H에서 온다.

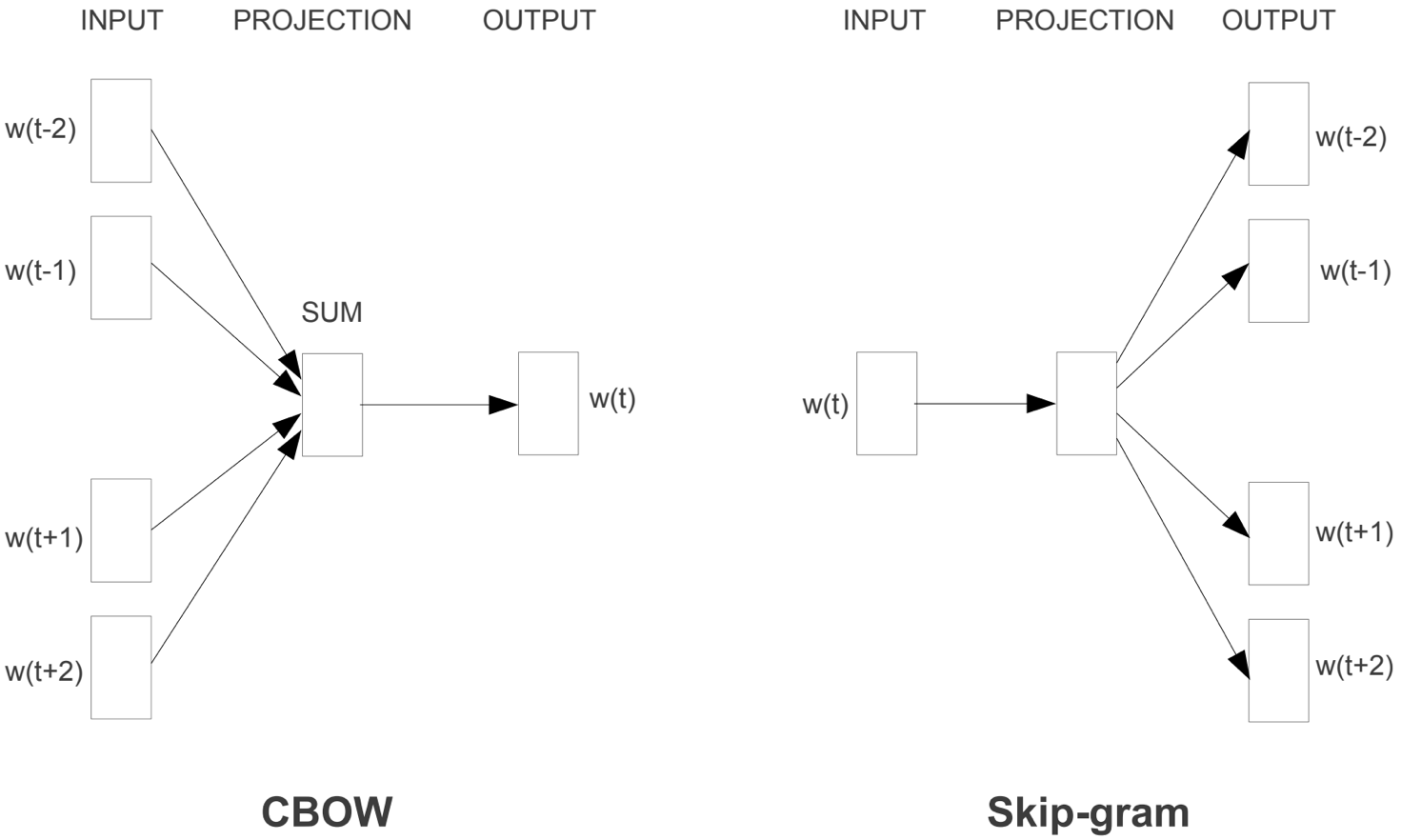
**2.3 Parallel Training of Neural Networks**

거대한 데이터셋에 모델을 학습시키기 위해, 우리는feedforward NNLM과 이 논문에서 제안된 새로운 모델을 포함하여, DistBelief([6])라고 불리는 대규모 분산 프레임워크의 top에서 몇 가지 모델을 구현했다. 프레임워크는 우리가 같은 모델의 복사본을 병렬적으로 실행시킬 수 있도록 해준다. 그리고 각각의 복사본은 모든 파라미터를 갖고 있는 중앙 서버를 통해, 그것의 gradient 업데이트를 동기화한다. 이 병렬 학습을 위해, 우리는 Adagrad라고 불리는 adaptive learning rate procedure를 사용해 mini-batch asynchronous gradient descent를 사용한다. 이 프레임워크에서, 백 개 이상의 모델 복사본을 사용하는 것은 흔하다. 각각은 데이터 센터의 다른 머신에서 많은 CPU 코어를 사용한다.

**3 New Log-linear Models**

이 섹션에서, 우리는 컴퓨팅 복잡도를 최소화하도록 단어 분산 표현 학습을 위한 두 개의 새로운 모델 아키텍쳐를 제안한다. 이전 섹션에서 주요한 관찰은 복잡도의 대부분이 모델의 non-linear hidden layer에 의해 발생한다는 것이었다. 이것은 신경망을 매우 매력적으로 만드는 것이지만, 우리는 아마도 신경망만큼 정확하게는 데이터를 표현하지 못할, 더 간단한 모델을 탐색하기로 했다. 하지만, 아마도 훨씬 더 많은 데이터에서 이 모델은 효율적으로 학습될 수 있을 것이다.

새로운 아키텍쳐는 우리의 초기 연구 ([13], [14])에서 제시된 그러한 모델을 직접적으로 따른다. 우리는 앞선 연구에서 NNLM이 두 가지 단계에서 성공적으로 학습될 수 있다는 것을 발견했다. 첫번째로, 연속적인 단어 벡터는 간단한 모델을 사용해 학습된다. 그리고, N-gram NNLM은 이러한 단어 분산 표현의 가장 윗 단계에서 학습된다. 최근에 단어 벡터를 학습하는 것에 초점을 맞춘 상당한 양의 연구가 나왔지만, 우리는 [13]에서 제시된 접근법이 가장 간단한 것으로 간주한다. 관련된 모델은 훨씬 더 일찍이 제시되었음을 기억하라. ([26], [8])



**Figure 1.** 새로운 모델 아키텍쳐이다. CBOW 아키텍쳐는 문맥을 바탕으로 현재 단어를 예측하고, Skip-gram은 현재 단어를 고려하여 주변 단어를 예측한다.

**3.1 Continuous Bag-of-Words Model**

첫번째로 제시하는 모델은 feedforward NNLM과 유사하다. 비선형 은닉층이 제거되고, 모든 단어에 대해 투사층이 공유된다. (Projection matrix 뿐만 아니라) 그러므로, 모든 단어는 같은 위치에 투사된다. (그들의 벡터는 평균화된다.) 우리는 이 아키텍쳐를 bag-of-words 모델이라고 부른다. 그 역사에서 단어의 순서는 투사에 영향을 미치지 않기 때문이다. 더욱이, 우리는 또한 미래의 단어를 사용한다. 우리는 input에서 4개의 전, 후 단어를 가지고 log-linear classifier를 만듦으로써 다음 섹션에서 도입된 태스크에 대해 최고의 성능을 얻었다. 이때 training criterion은 현재 단어, 즉 가운데 단어를 알맞게 분류하는 것이다. 학습 복잡도는 다음과 같다.

(4)

우리는 이 모델을 추후에 CBOW로 표기한다. 표준 bag-of-words 모델과 달리, 그것은 맥락의 연속적인 분산 표현을 사용하기 때문이다. 모델 아키텍쳐는 Figure 1 과 같다. 입력층과 투사층 사이의 가중치 행렬은 모든 단어 위치에 대해 NNLM에서와 같은 방식으로 공유됨을 주목하라.

**3.2 Continuous Skip-gram Model**

두번째 아키텍쳐는 CBOW와 유사하지만, 문맥을 바탕으로 현재 단어를 예측하는 대신, 같은 문장 내 또 다른 단어를 바탕으로 단어의 classification을 최대화한다. 더 정확히, 우리는 입력으로 각각의 현재 단어를 사용하고, continuous projection layer와 함께 log-linear classifier에 넣는다. 그리고 현재 단어 앞 뒤로 특정 범위 내의 단어들을 예측한다. 우리는 범위를 늘리는 것이 결과로 나오는 단어 벡터의 퀄리티를 향상시키는 것을 발견했다. 하지만, 그것은 컴퓨팅 복잡도를 늘리기도 했다. 더 먼 거리의 단어들은 보통 가까운 것보다는 현재 단어와 덜 관련이 있기 때문에, 우리는 training example에서 그러한 단어들을 덜 샘플링함으로써 거리가 먼 단어에 더 적은 가중치를 부여한다.

이 아키텍쳐의 트레이닝 복잡도는 아래에 비례하다.

(5)

C는 단어의 최대 거리이다. 그러므로, 만약 C=5를 선택하면, 우리가 랜덤하게 고를 각각의 학습 단어 R개는 [1, C] 범위 내에 놓인다. 그리고, 맞는 라벨로서 현재 단어의 앞, 뒤 R개의 단어를 사용한다. 이것은 우리가 R\*2개의 단어를 분류하도록 요구한다. 이때의 입력은 현재 단어이고, 각각 R+R개의 단어가 출력으로 나오게 된다. 이후 실험에서, 우리는 C=10으로 사용한다.

**4 Results**

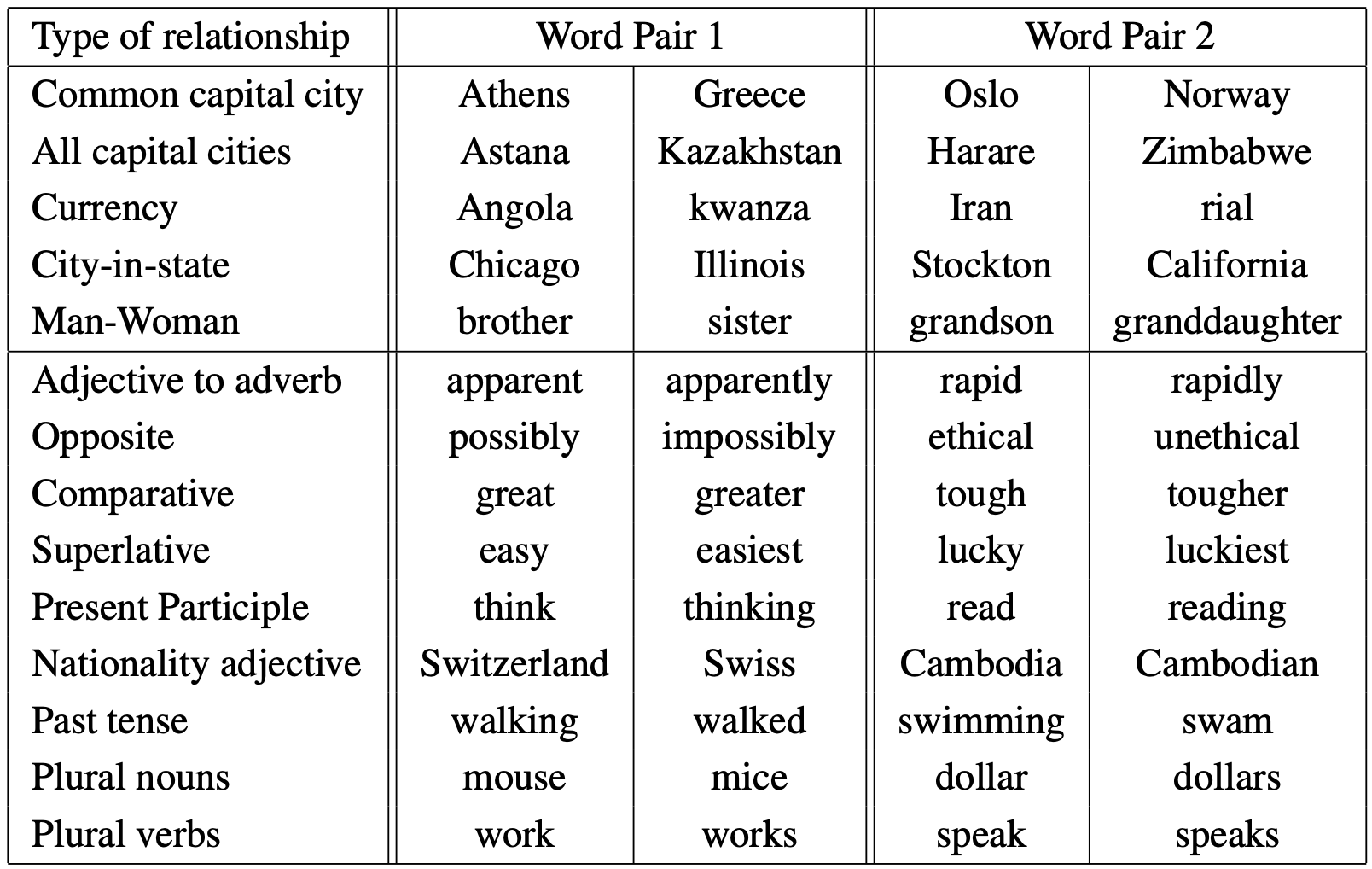
다른 버전의 단어 벡터 퀄리티를 비교하기 위해, 이전의 논문들은 보통 example words와 그들의 가장 비슷한 단어들을 보여주는 테이블을 사용했고, 그것들을 직관적으로 이해했다. 단어 “France”는 Italy와 비슷하고 아마도 일부 다른 나라들과 비슷하다는 것을 보여주기 쉬울지라도, 더 복잡한 유사도 태스크에서 그러한 벡터들을 적용하는 것은 훨씬 더 어렵다. 우리는 단어들 간의 많은 다른 유형의 유사도가 있을 수 있다는 이전의 관찰을 따른다. 예를 들어, “small”이 “smaller”와 유사하다는 맥락에서,“big”은 “bigger”와 유사하다. 관계의 또 다른 유형의 예시로는 “big”-“biggest”와 “small”-“smallest”와 같은 단어쌍이 있다. ([20]) 우리는 다음 질문에서와 같은 관계를 갖는 단어쌍을 추가적으로 표기한다. - “biggest”가 “big”과 비슷하다는 맥락에서, “small”과 비슷한 단어는 무엇인가?

놀랍게도, 이러한 질문들은 단어들의 벡터 표현을 가지고 단순한 선형대수적 연산을 수행함으로써 답변 가능하다. “Biggest”가 “big”과 비슷하다는 맥락에서 “small”과 비슷한 단어를 찾기 위해, 우리는 간단하게 벡터 X=vector(“biggest”) - vector(“big”) + vector(“small”) 를 계산하고, 그것을 질문에 답으로 사용한다. (우리는 이 탐색 동안 입력의 질문 단어들을 없앤다.) 단어 벡터들이 잘 훈련될 때, 이 방법을 사용해 맞는 답, 즉, “smallest”를 찾는 것이 가능하다.

마지막으로, 우리는 고차원의 단어 벡터를 대량의 데이터셋에 대해 학습시킬 때, 결과로 나오는 벡터가, 이를 테면 city와 그 city가 속한 country(France - Paris, Germany - Berlin)와 같은 단어들 간의 아주 미묘한 맥락적 관계에 답이 될 수 있음을 발견했다. 이러한 맥락적 관계를 포함하는 단어 벡터는 기계 번역, 정보 탐색, 질의 응답과 같이 많은 현존하는 NLP 응용을 향상시킬 수 있다. 그리고 아마도 아직은 만들어지지 않은, 다른 추후의 응용도 가능하게 할 것이다.

**4.1 Task Description**

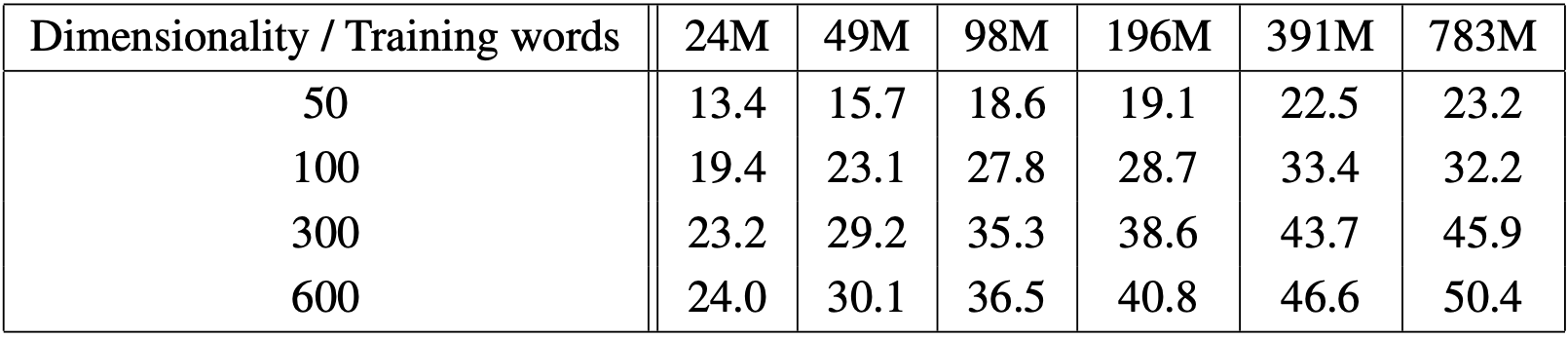
단어 벡터의 퀄리티를 측정하기 위해, 우리는 다섯 종류의 시멘틱 질문과 아홉 가지의 신택틱 질문을 포함하는 종합적인 테스트셋을 정의한다. 각각의 카테고리에서 두 가지의 예시는 Table 1에서 볼 수 있다. 종합적으로, 8,869개의 semantic question과 10,675개의 syntatic



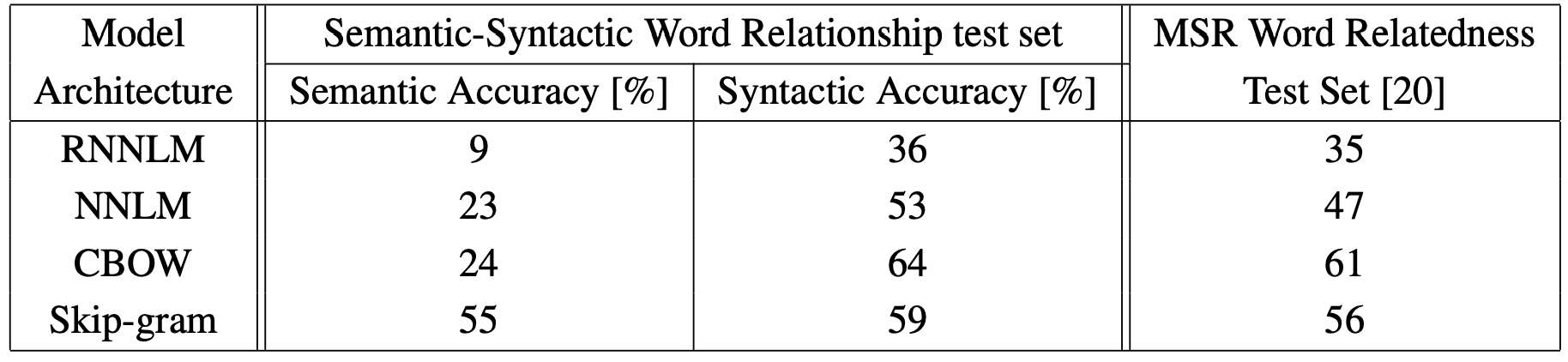
**Table 1.** Semantic-Syntatic Word Relationship test set에 포함된 5가지의 semantic question과 9가지의 syntactic question.

question이 존재한다. 각 카테고리의 질문들은 두 개의 단계로 만들어졌다. 첫번째로, 비슷한 단어 쌍 리스트는 수동으로 만들었다. 다음으로, 큰 질문 리스트는 두 단어 쌍을 연결시킴으로써 형성된다. 예를 들어, 우리는 68개의 큰 미국 도시와 그들이 속한 주의 리스트를 만들었고, 랜덤으로 두 단어 쌍을 집음으로써 2.5K 개의 질문을 형성했다. 우리는 테스트셋에 오직 single token word만 포함시켰고, 그러므로 New York과 같은 multi-word entity는 존재하지 않는다.

우리는 모든 질문 유형에 대한 전체적인 정확도를 측정하고, 각 질문 타입을 독립적으로도 측정하였다. 질문은 위의 방법으로 계산된 벡터와 가장 가까운 단어가 질문의 맞는 답변과 정확히 일치할 때에만 맞게 답변 되었다고 가정했다. 이것은 또한 100% 정확도를 달성하는 것이 불가능해보임을 의미한다. 현재의 모델은 word morphology에 대한 어떠한 입력 정보도 갖고 있지 않기 때문이다. 하지만, 우리는 특정 응용에 대한 단어 벡터의 유용성이



**Table 2.** Semantic-Syntatic Word Relationship test set의 subset에서의 정확도. 제한된 vocabulary를 갖는 CBOW 아키텍쳐에서의 단어 벡터를 사용하였다. 가장 자주 사용된 30k개의 단어를 포함하는 question들만 사용되었다.



**Table 3.** 640차원의 단어 벡터를 가지고, 같은 데이터에서 훈련시킨 모델을 사용해 아키텍쳐를 비교하였다. 정확도는 우리의 Semantic-Syntatic Word Relationship test set과 [20] 에서 사용된 syntactic relationship test set을 사용하여 측정되었다.

이 정확도 행렬과 긍정적으로 관계되어야 한다고 믿는다. 추후의 발전은 특히 syntatic question에서 단어 구조에 대한 정보를 포함시킴으로써 달성 가능할 것이다.

**4.2 Maximization of Accuracy**

우리는 단어 벡터를 학습시키기 위해 Google News corpus를 사용했다. 이 코퍼스는 약 6B token을 포함한다. 우리는 vocabulary 사이즈를 1M개의 가장 자주 사용된 단어로 제한했다. 명백하게, 우리는 시간 제약이 있는 최적화 문제에 직면한다. 더 많은 데이터를 사용하는 것과, 고차원의 단어 벡터를 사용하는 것 모두 정확도를 높일 것이라고 예상되기 때문이다. 가능한 한 좋은 결과를 가능한 한 빠르게 얻기 위한 최고의 모델 아키텍쳐를 측정하기 위해, 우리는 첫번째로 학습 데이터의 일부를 학습시킨 모델에 검증을 진행했다. 이때의 데이터는 가장 자주 사용된 30k vocabulary로 제한되었다. 단어 벡터의 차원을 다르게 선택하고, 학습 데이터의 양을 늘린 CBOW 아키텍쳐를 사용한 결과는 Table 2와 같다.

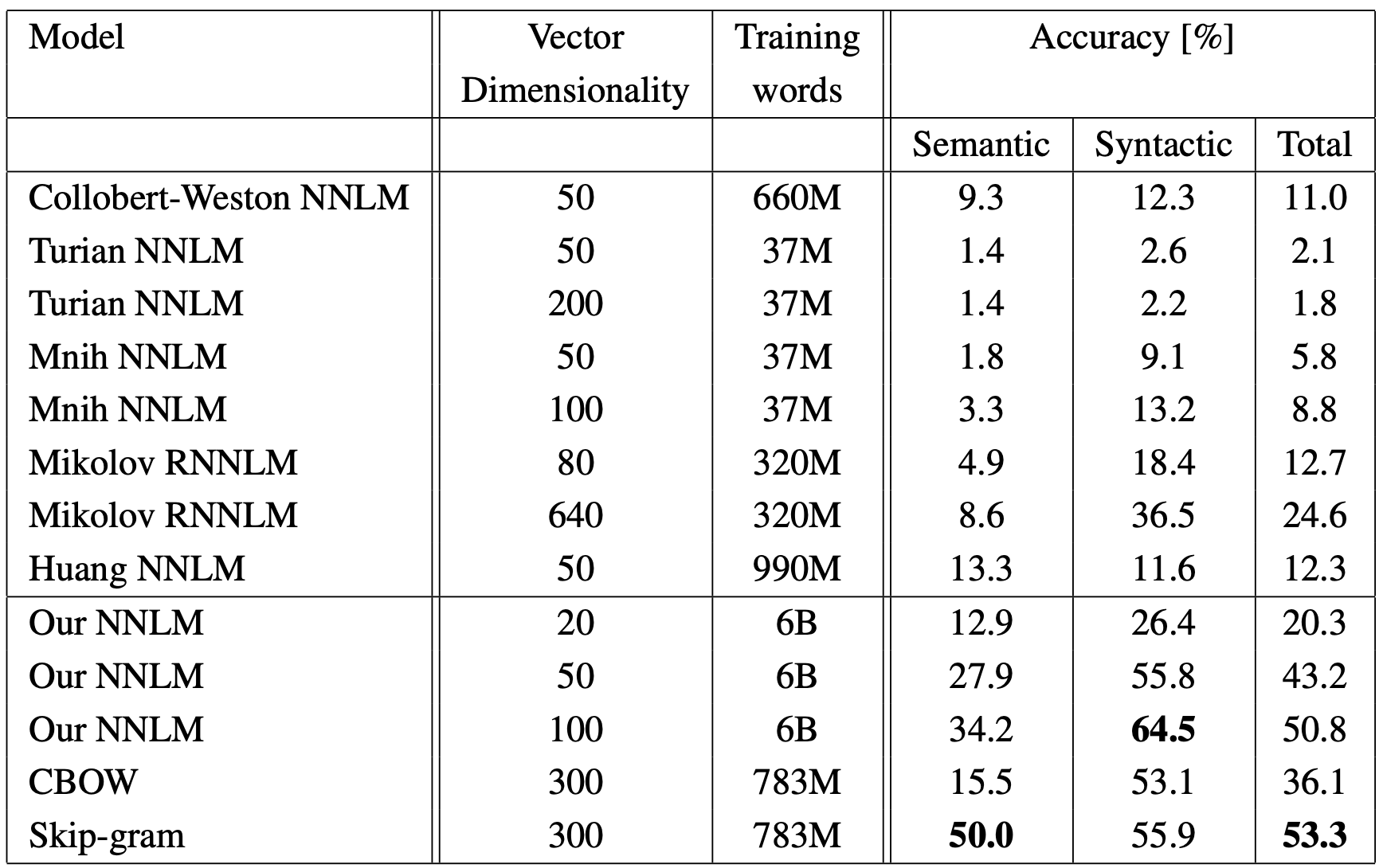
몇 개의 지점 다음에, 차원을 더 추가하거나, 학습 데이터를 더 추가하는 것은 향상이 잘 이루어지지 않는다는 것을 볼 수 있다. 그래서, 우리는 벡터의 차원과 학습 데이터의 양을 모두 늘렸다. 이 관찰은 사소해보일지라도, 상대적으로 많은 양의 데이터에 대해 불충분한 사이즈를 갖는 단어 벡터를 학습시키는 것이 현재 대중적이라는 것을 반드시 인지하라. (예를 들면, 50-100) 방정식 4를 고려할 때, 학습 데이터의 양을 두 배로 증가시키는 것은 벡터 사이즈를 두 배로 늘리는 것만큼 계산 복잡도가 증가하는 결과를 낳는다.

Table 2와 Table 4에서 제시된 것과 같이, 우리는 SGD와 backpropagation을 사용하여 3개의 training epoch을 사용했다. 우리는 시작 학습률을 0.025로 했고, 마지막 에포크의 끝부분에서 0에 가까워지도록, 학습률을 선형적으로 감소시켰다.

**4.3 Comparison of Model Architectures**

첫번째로, 우리는 같은 640 차원의 단어 벡터, 같은 훈련 집합을 사용한 단어 벡터로부터 만들어진 다른 모델 아키텍쳐를 비교한다. 추후 연구에서, 우리는 Semantic-Syntactic Word Relationship의 전체 질문 집합을 사용한다. 즉, 제한되지 않은 30k vocabulary를 모두 사용한다. 우리는 또한 단어들 간의 문법적 유사도에 초점을 맞춘 [20] 에서 사용한 테스트셋의 결과를 포함했다.

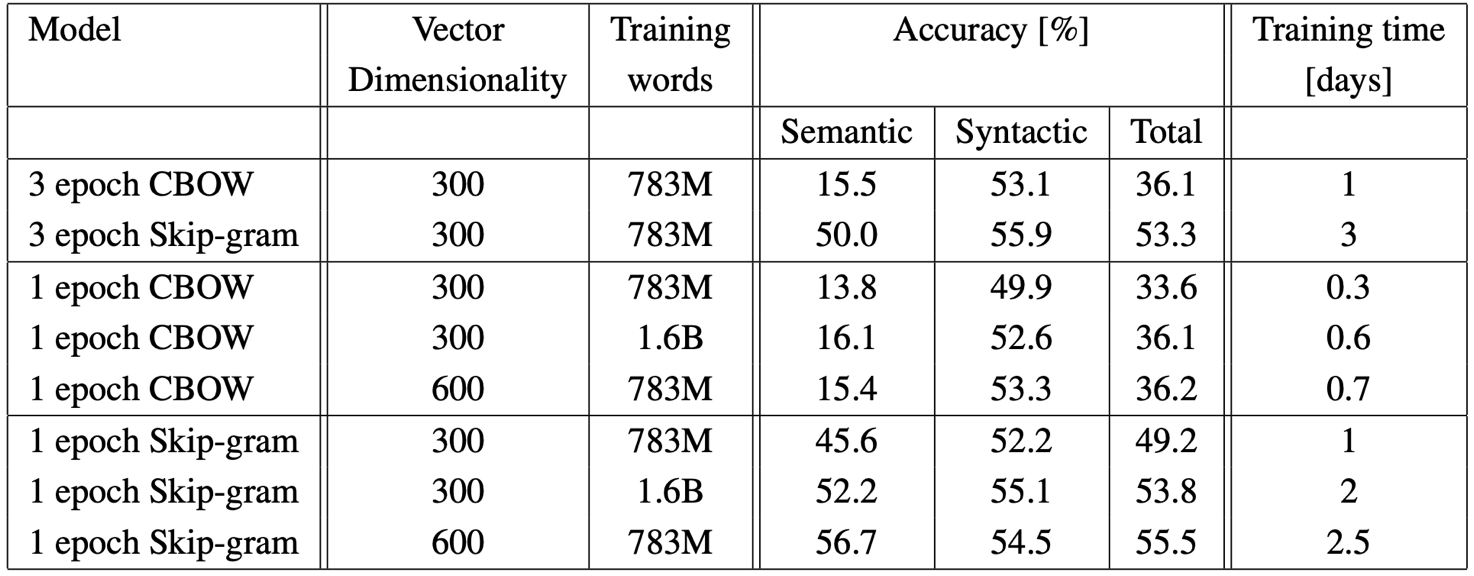
학습 데이터는 몇 가지 LDC corpora를 포함하고, 자세한 사항은 [18]에서와 같이 묘사된다. (320M 개의 단어, 82K 개의 vocabulary). 우리는 이전에 학습시킨, 하나의 CPU에서 학습에 약 8주가 소요된 RNNLM과의 비교를 위해 이러한 데이터를 사용했다. 우리는 8개의 이전 단어를 사용해서, DistBelief parallel training ([6])와 같이 640개의 hidden unit을 갖는 feedforward NNLM을 학습시켰다. (그러므로, NNLM은 RNNLM보다 더 많은 파라미터를 갖는다. 투사층의 사이즈가 640\*8이기 때문이다.)



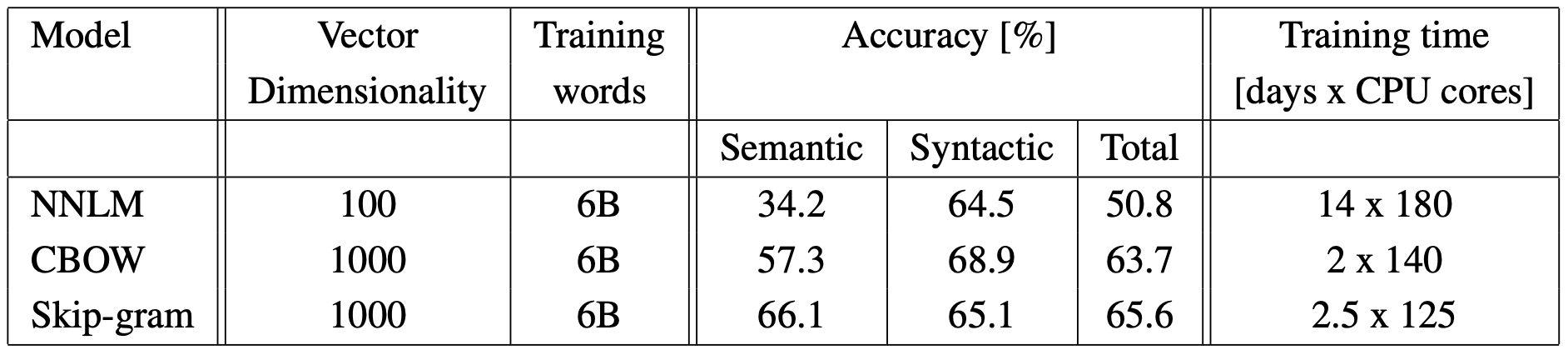
**Table 4.** Semantic-Syntatic Word Relationship test set에서 공개적으로 이용 가능한 단어 벡터, 그리고 우리 모델에서 만든 단어 벡터의 비교. 모든 vocabulary를 사용하였다.

Table 3에서, RNN으로 만든 ([20]에서처럼) 단어 벡터는 대개 syntatic question에서 잘 작동하는 것을 볼 수 있다. NNLM 벡터는 RNN보다 상당히 더 좋은 성능을 보인다. 이것은 놀라운 결과는 아니다. RNNLM의 단어 벡터가 비선형 은닉층에 직접적으로 연결되기 때문이다. CBOW 아키텍쳐는 syntatic task에서 NNLM보다 더 잘 작동하고, semantic task에서도 비슷하게 작동한다. 마지막으로, Skip-gram 아키텍쳐는 CBOW 모델보다 syntactic task에서 약간 더 낮지만, NNLM보단 여전히 더 낫다. 그리고, semantic part에서는 모든 다른 모델들보다 훨씬 더 괜찮았다.

다음으로, 우리는 하나의 CPU를 사용해 학습된 우리 모델을 검증하고, 공개적으로 이용 가능한 단어 벡터에 대한 결과와 비교했다. Table 4를 통해 확인 가능하다. CBOW 모델은 Google News 데이터의 서브셋으로 약 하루 동안 학습되었다. Skip-gram 모델의 학습 시간은 약 3일이었다.



**Table 5.** 같은 데이터에 대해 1 에폭만 학습시킨 모델과 3 에폭 동안 학습 시킨 모델의 비교. 정확도는 Semantic-Syntatic data set 전체를 사용하여 측정되었다.



**Table 6.** DistBelief 분산 프레임워크를 사용해 학습시킨 모델 간의 비교. 1,000 차원의 벡터를 사용한 NNLM의 학습이 완성시키는 데에 너무 오래 걸렸음을 인지하라.

추후의 보고되는 실험 결과에 대해, 우리는 에폭을 한 번만 수행했다. (다시, 우리는 학습률을 학습의 마지막에 0에 가깝도록 선형적으로 감소시켰다.) Table 5에서 볼 수 있듯이, 한 에폭을 사용해 두 배 많은 데이터를 모델에 학습시키는 것은 같은 데이터를 3 에폭 동안 학습시키는 것보다 비교할만한, 혹은 더 좋은 결과를 제시한다. 그리고 약간의 속도 향상도 이뤄냈다.

**4.4 Large Scale parallel Training of Models**

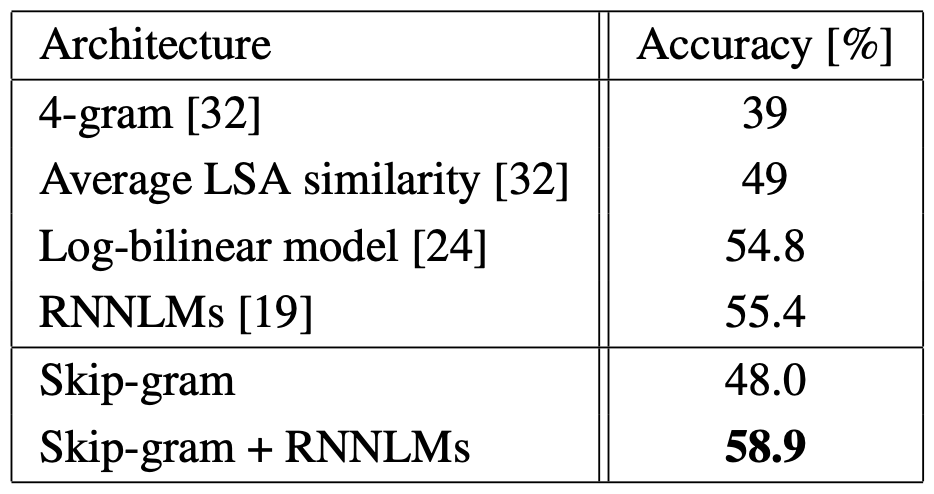
일찍이 언급했듯이, 우리는 DistBelief라고 불리는 분산된 프레임워크에서 다양한 모델을 구현했다. 아래에서 mini-batch asynchronous gradient descent와 adaptive learning rate procedure (Adagrad, [7])를 적용하여 Google News 6B data set에서 학습시킨 몇 가지 모델의 결과를 보인다. 우리는 학습 기간 동안 50~100개의 모델 replica를 사용했다. CPU 코어의 개수는 추정치이다. 데이터 센터 머신이 다른 태스크와 공유되고, 사용량은 꽤나 변동될 수 있기 때문이다. 분산 프레임워크의 오버헤드 때문에, CBOW 모델과 Skip-gram 모델의 사용량은 그들의 single-machine implementation보다 서로에 훨씬 더 가까움을 기억하라. 결과는 Table 6에서 제시된다.

**4.5 Microsoft Research Sentence Completion Challenge**

Microsoft Sentence Challenge는 최근에 LM과 다른 NLP 기술의 발전을 위한 task로서 도입되었다. ([32]) 이 태스크는 1,040 개의 문장을 포함한다. 각 문장에서 한 단어는 누락되어 있고, 목표는 주어진 다섯 개의 괜찮은 선택지 중에서, 문장의 나머지 부분과 가장 잘 맞는단어를 고르는 것이다. N-gram 모델, LSA-based model([32]), log-bilinear model([24])를 포함한 몇 가지 기술의 퍼포먼스는 이미 이 데이터셋에 대해 주어졌다. 현재 이 벤치마크에서 55.4%의 정확도를 보유한 state-of-the-art를 달성한 모델은 순환 신경망의 결합이다.

우리는 이 태스크에서 Skip-gram 아키텍쳐의 성능을 측정했다. 첫번째로, 우리는 [32]에서 제공된 50M 개의 단어를 가지고 640 차원의 모델을 학습시켰다. 그 다음에, 우리는 입력에서 알려지지 않은 단어를 사용함으로써 테스트셋 각각의 문장의 점수를 계산하였고, 문장의 모든 주변 단어를 예측했다. 마지막 문장 점수는 이러한 각각의 예측의 총합이다. Sentence score를 사용하여, 우리는 가장 가능성 있는 문장을 골랐다.

이 새로운 결과와, 몇 가지 이전의 결과의 간략한 요약은 Table 7에서 제시되었다. Skip-gram 모델은 그것 자체로는 LSA similarity보다 더 나은 성능을 보이지는 않았지만, 이 모델로 계산한 점수는 RNNLM으로 얻은 점수와 상호보완적이다. 가중 조합은 58.9%의 정확도로,

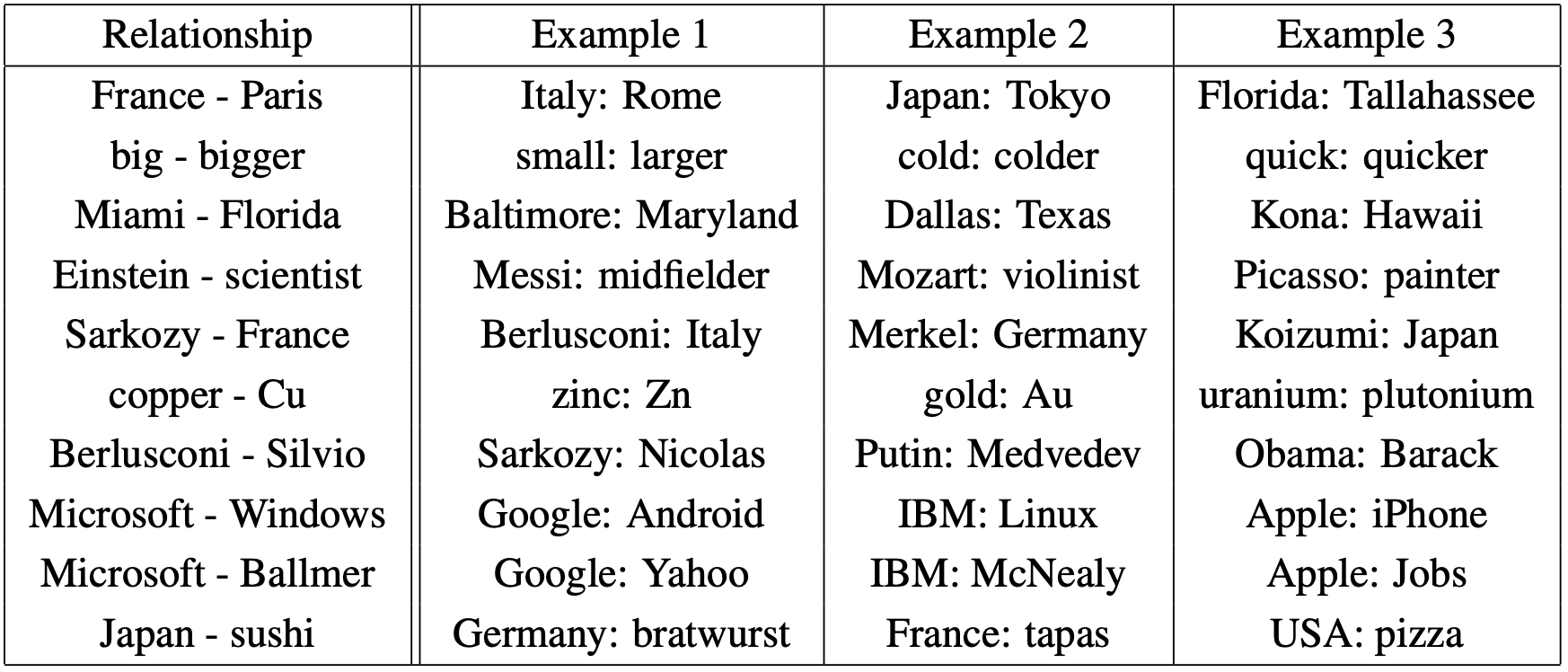


**Table 7.** Microsoft Sentence Completion Challenge에서의 모델 조합과 비교

새로운 state-of-the-art를 이끌어낸다. (Development part of the set에서 59.2%, test part of the set에서 58.7%)

**5 Examples of the Learned Relationships**

Table 8은 다양한 관계를 따르는 단어를 보인다. 우리는 위에서 언급한 것과 같은 접근법을 따른다. 관계는 두 단어 벡터의 뺄셈으로 정의되고, 결과는 또 다른 단어에 더해진다. 그러므로, 예를 들어서, “Paris” - “France” + “Italy” = “Rome”이다. 비록 명백히 더 많은 개선의 여지가 있을지라도, 표에서 볼 수 있듯이, 정확도는 꽤 괜찮다. (정확한 매치를 가정하는 우리의 정확도 행렬을 사용하는 것을 기억하라. Table 8의 결과는 60% 정도로 점수 매겨질 것이다.) 우리는 훨씬 더 큰 단어 집합에서 학습된 더 고차원의 단어 벡터가 상당히 더 좋은 성능을 낼 것이라고 믿는다. 그리고, 새로운 혁신적인 응용의 시작을 가능하게 할 것이다. 정확도를 올리기 위한 또 다른 방법은 관계의 예시를 한 개보다 더 많이 제공하는 것이다. 관계 벡터를 형성하는 한 개 대신에, 10개의 예시를 사용함으로써, (우리는 개별적인 벡터를 함께 평균 낸다.) 우리의



**Table 8.** 단어 쌍 집합의 예시이다. Table 4에서의 베스트 단어 벡터를 사용하였다. (783M 단어로 학습시킨 300 차원의 Skip-gram 모델이다.)

베스트 모델의 정확도가 semantic-syntatic test에서 절대적으로 약 10% 향상하는 것을 확인했다.

또한, 벡터 연산을 다른 태스크를 해결하기 위해 적용하는 것도 가능할 것이다. 예를 들어, 우리는 단어 리스트의 벡터 평균을 계산하고, 가장 거리가 먼 단어 벡터를 찾아냄으로써 out-of-the-list-words를 골라내는 것에서도 괜찮은 정확도를 얻어냈다. 이것은 특정 인간 지능 테스트에서 인기 있는 종류의 문제이다. 명백하게, 이 기술을 사용하여 만들어낼 수 있는 많은 발견들이 여전히 존재한다.

**6 Conclusion**

이 연구에서, 우리는 문법적, 그리고 문맥적 언어 태스크 모음에서 다양한 모델에 의해 만들어진 단어의 벡터 표현의 퀄리티를 연구했다. 우리는 인기 있는 신경망 모델 (feedforward와 recurrent 둘 다) 과 비교해서, 다양한 단순한 모델 아키텍쳐를 사용함으로써 양질의 단어 벡터를 학습시키는 것이 가능함을 확인했다. 훨씬 더 낮은 계산 복잡도 때문에, 훨씬 더 큰 데이터 셋을 사용하여 매우 정확한 고차원의 단어 벡터를 계산하는 것이 가능하다. DistBelief 분산 프레임워크를 사용하여, CBOW와 Skip-gram 모델을 trillion words의 corpora, 근본적으로는 vocabulary의 사이즈 제한 없이도 훈련시키는 것이 분명 가능할 것이다.

단어 벡터가 이전 state-of-the-art보다 최근에 상당히 좋은 성과를 낸 태스크는 SemEval-2012 Task 2 ([11]) 이다. 공개적으로 이용 가능한 RNN 벡터는 다른 기술들과 함께 사용되어 Spearman’s rank correlation에서 이전 베스트 결과([31])보다 50%의 향상을 이루어냈다. 단어 벡터를 기반으로 한 신경망은 이전에 많은 다른 NLP 태스크에서 적용되었다. 예를 들어서, 감성 분석([12])와, paraphrase detection([28])과 같은 것이 있다. 이러한 응용은 이 논문에서 묘사된 모델 아키텍쳐를 사용하여 더 좋은 결과를 낼 수 있을 거라고 예측할 수 있다.

우리의 ongoing work는 단어 벡터가 성공적으로 automatic extension of facts in Knowledge Bases에 적용될 수 있음을 보인다. 존재하는 사실의 진위성을 입증하기 위해서도 마찬가지이다. 기계 번역 실험의 결과는 매우 전망 있어 보인다. 추후에, 우리의 기술을 Latent Relational Analysis ([30) 등과 비교하는 것도 흥미로울 것이다. 우리는 우리의 종합 test set이, 연구 커뮤니티가 단어 벡터를 검증하기 위한 현존하는 기술을 향상시키는 것을 도울 것이라고 믿는다. 우리는 또한 양질의 단어 벡터가 미래 NLP 응용의 중요한 building block이 될 것이라고 기대한다.

**7 Follow-Up Works**

이 논문의 초기 버전이 쓰여진 다음, 우리는 CBOW와 Skip-gram 아키텍쳐를 둘 다 사용하여 단어 벡터 계산을 위한 single-machine multi-threaded C++ code를 발표했다. 학습 속도는 논문의 초기 버전에서 보고된 것보다 상당히 더 빠르다. 즉, 하이퍼 파라미터를 고르는 시간 당 billion 개의 단어 순서가 상당히 더 빠르다는 것이다. (???) 우리는 또한 100 billion 개 이상의 단어로 학습된, 명명된 entity를 대표하는 1.4M 개의 벡터를 발표했다. 우리의 추후 연구는 다가오는 NIPS 2013 paper에 기재될 것이다. ([21])