Efficient Estimation of Word Representations in Vector Space

& Distributed Representations of Words and Phrases and their Compositionality

최희정

Efficient Estimation of Word Representations in Vector Space

1 Introduction

1.1 Goals of the 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.

- 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.
  ex) simple algebraic operations: vector("King")- vector("Man")+ vector("Woman")= "Queen"

- In this paper, we try to maximize accuracy of these vector operations by developing new model architectures that preserve the linear regularities among words.

- Moreover, we discuss how training time and accuracy depends on the dimensionality of the word vectors and on the amount of the training data.

1 Introduction

1.2 Previous Work

- Neural Network Language Model(NNLM)
  : feedforward neural network with a linear projection layer and a non-linear hidden layer was used to learn jointly the word vector representation and a statistical language model

- Word vectors are first learned using neural network with a single hidden layer. The word vectors are then used to train the NNLM. In this work, we directly extend this architecture.

- However, these architectures were significantly more computationally expensive.

2 Model Architectures

- In this paper, we focus on distributed representations of words learned by neural networks.

- To compare different model architectures, we define first the computational complexity of a model as the number of parameters that need to be accessed to fully train the model. Next, we will try to maximize the accuracy, while minimizing the computational complexity.

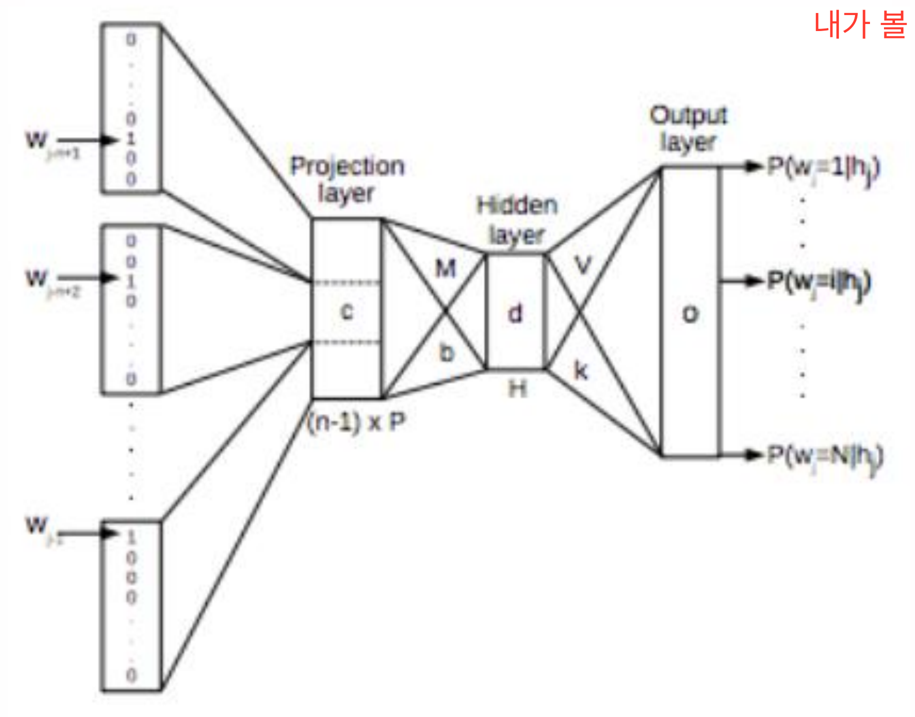
- the computational complexity: $O = E \times T \times Q$
  where E is number of the training epochs, T is the number of the words in the training set and Q is defined further for each model architecture.

- All models are trained using stochastic gradient descent and backpropagation

2 Model Architectures

2.1 Feedforward Neural Net Language Model (NNLM)

: 현재 단어 이전의 N개 단어의 one-hot encoding 벡터에 대한 확률분포 출력



NNLM Structure

내가 볼 N개의 단어, 5개가 있으면, 4개를 보는게 N이고, 1개를 보는 것.

- 구성: Input Layer(N), Projection Layer(D), Hidden Layer(H), Output Layer(V) corpus의 모든 단어

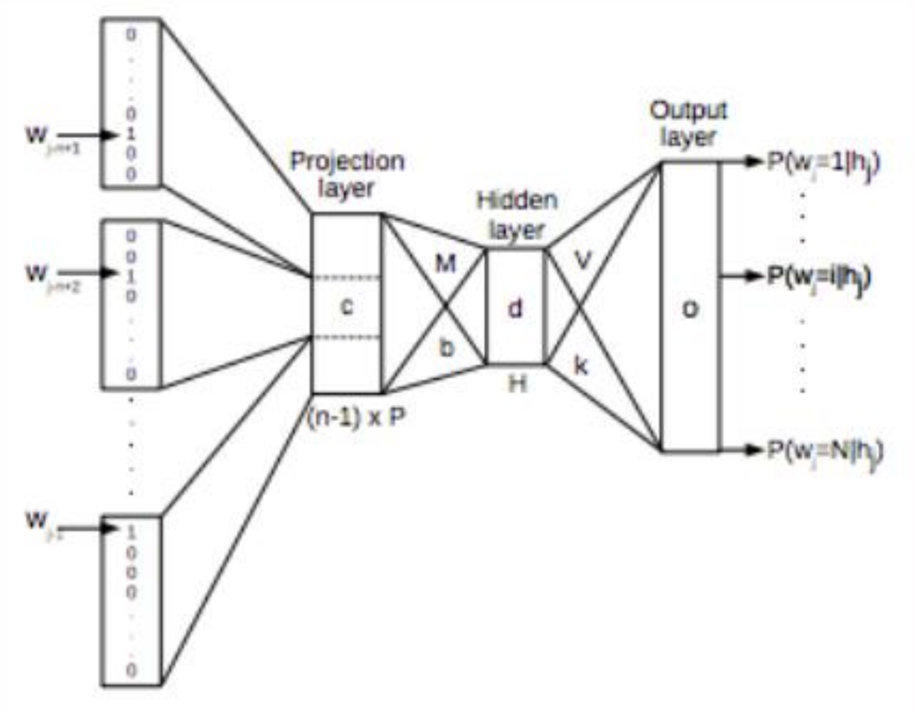
- Input Layer: 현재 단어 이전의 N개의 단어를 V개의 사전단어를 기반으로 한 one-hot encoding 벡터 입력
- Projection Layer: N개의 input에 대해 Projection Matrix(V×D)로 projection
- Hidden Layer: 각 단어의 probability distribution 계산

- 최적화: input & output 차이로부터 error 계산 후, back-propagation를 통한 weight 최적화

2 Model Architectures

2.1 Feedforward Neural Net Language Model (NNLM)

N개의 단어, D차원의 프로젝션 레이어,



NNLM Structure

- computational complexity
  : $Q = N \times D + N \times D \times H + H \times V$

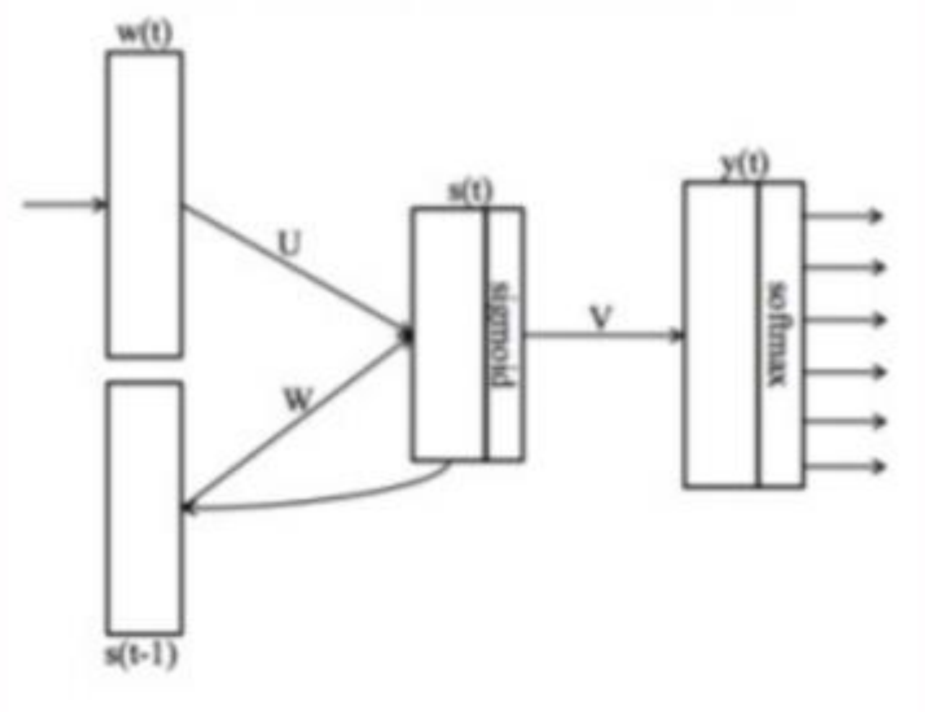
- dominating term = $H \times V$
  (∵ 사전은 가능한 모든 단어 포함, 약 천만개의 단어)

- dominating term after hierarchical softmax = $N \times D \times H$
  (∵ hierarchical softmax: V -> $\log_2 V$)

2 Model Architectures

2.2 Recurrent Neural Net Language Model (RNNLM)

: NNLM에 RNN을 적용한 model (Projection Layer 대신 Hidden Layer에 recurrent한 연결 존재)



RNNLM Structure

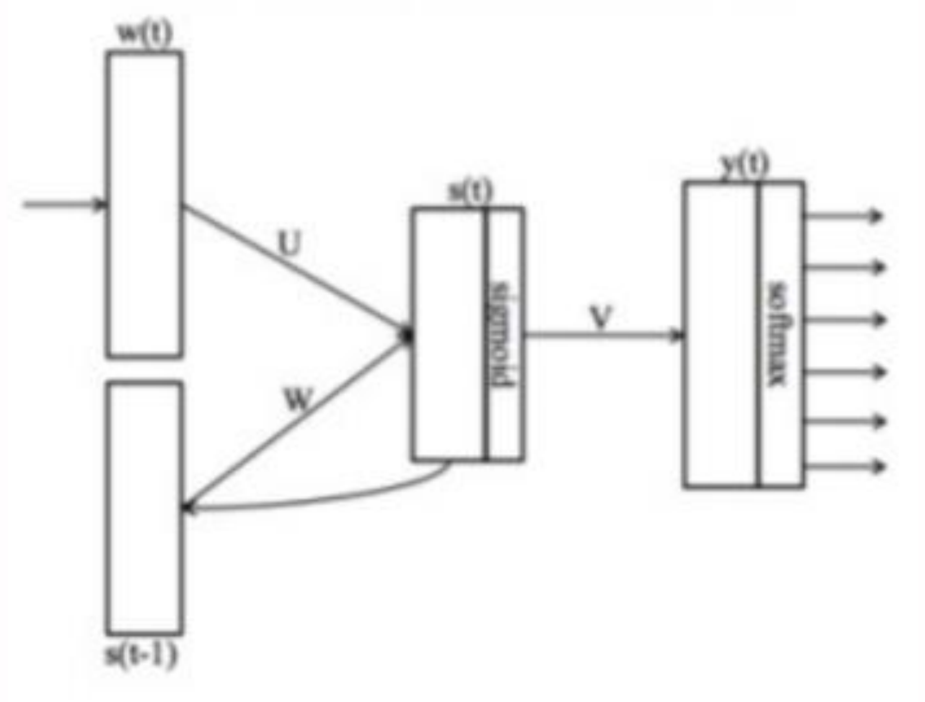
- 구성: Input Layer(N), Hidden Layer(H), Output Layer(V)

- Input Layer: N개의 단어개수 지정없이 순차적으로 단어 입력
- Hidden Layer: 각 단어의 embedding과 이전 Hidden Layer를 input으로 학습해 이전 단어를 고려한 probability distribution 계산

- 최적화: input & output 차이로부터 error 계산 후, back-propagation를 통한 weight 최적화

2 Model Architectures

2.2 Recurrent Neural Net Language Model (RNNLM)



RNNLM Structure

- computational complexity

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

  히든레이어 차원

- dominating term = $H\times V$

  (∵ 사전은 가능한 모든 단어 포함, 약 천만개의 단어)

- dominating term after hierarchical softmax = $H\times H$

  (∵ hierarchical softmax: V -> $\log_2 V$)

2 Model Architectures

2.3 Parallel Training of Neural Networks

- To train models on huge data sets, we have implemented several models on top of a large-scale distributed framework called DistBelief.

- The framework allows us to run multiple replicas of the same model in parallel, and each replica synchronizes its gradient updates through a centralized server that keeps all the parameters.

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

3 New Log-linear Models

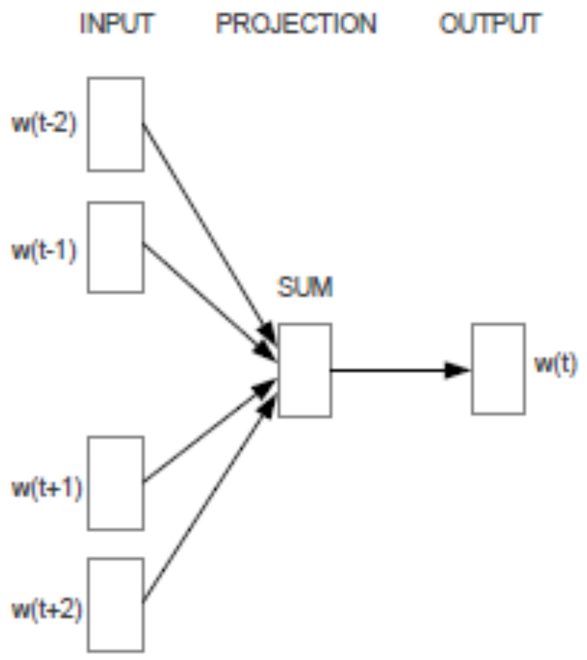
- In this section, we propose two new model architectures for learning distributed representations of words that try to minimize computational complexity. The main observation from the previous section was that most of the complexity is caused by the non-linear hidden layer in the model.

- The new architectures directly follow those proposed in our earlier work, 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

3 New Log-linear Models

3.1 Continuous Bag-of-Words Model (CBOW)

: 주어진 단어에 대해 앞뒤로 N/2개씩 총 N개의 단어를 Input으로 사용하여, 주어진 단어를 맞추는 모델
NNLM과 유사하고 단어순서가 projection에 영향을 미치지 않는 모델



CBOW

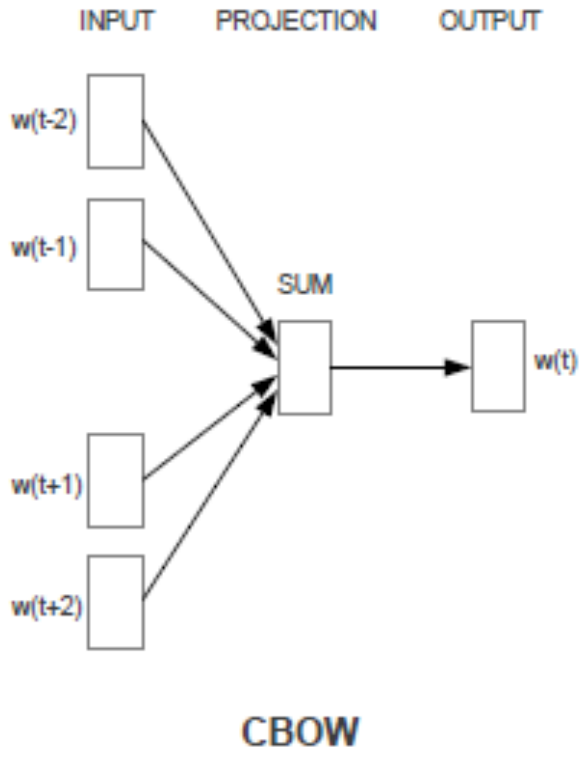
- 구성: Input Layer, Projection Layer(D), Output Layer(V)

- Input Layer: N개의 one-hot encoding 벡터 입력
- Projection Layer: 공통 Projection Matrix(V×D)로 projection한 후, 평균 벡터에 Weight Matrix (D×V)를 곱해 Output Layer로 보냄

- 최적화: output의 softmax 결과와 input의 차이로부터 error 계산 후, back-propagation를 통한 weight 최적화

3 New Log-linear Models

3.1 Continuous Bag-of-Words Model (CBOW)



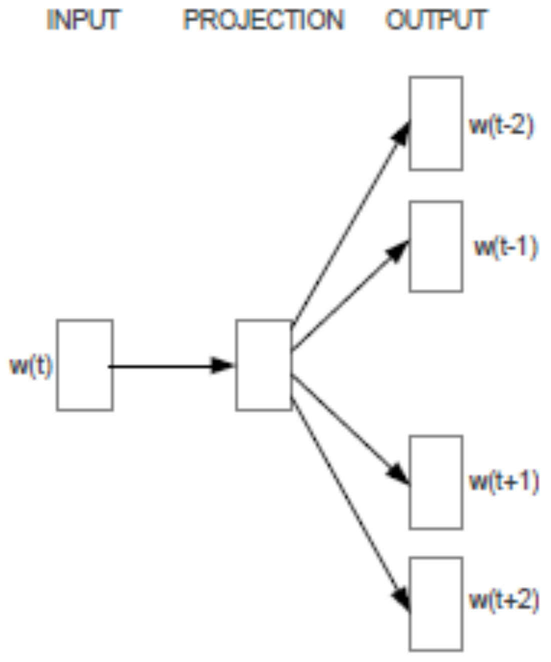
- computational complexity
  : $Q = N \times D + D \times \log_2 V$

- It uses continuous distributed representation of the context.

3 New Log-linear Models

3.2 Continuous Skip-gram Model

: 주어진 단어 하나를 가지고 같은 문장 내 특정범위의 주위 단어들의 등장 여부를 유추하는 모델
주위 단어들은 가까운 단어일수록 높은 확률로 샘플링해서 결정



Skip-gram

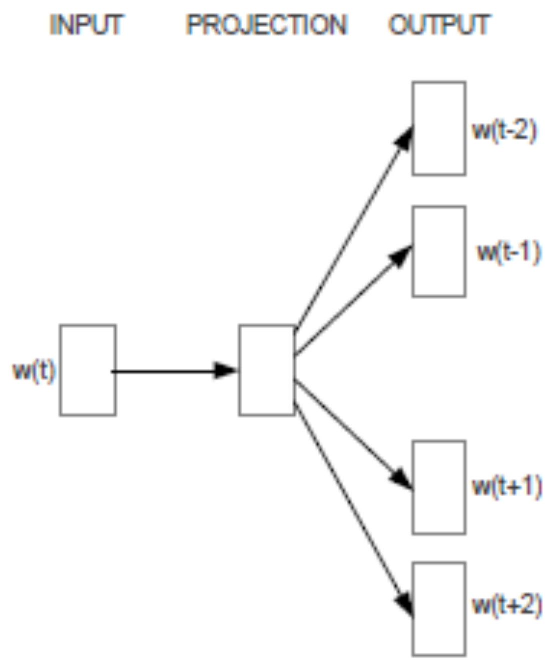
- 구성: Input Layer, Projection Layer(D), Output Layer(V)

- Input Layer: 1개의 단어 one-hot encoding 벡터 입력
- Projection Layer: Projection Matrix(1×D)로 projection 후, Weight Matrix (D×V) 곱해서 Output Layer로 보냄
- Output Layer: 2R개의 단어에 대한 등장 확률 출력

- 최적화: output 확률이 word의 확률분포와 동일할 때까지 학습

3 New Log-linear Models

3.2 Continuous Skip-gram Model



Skip-gram

- computational complexity
  : $Q = C \times (D + D \times \log_2 V)$

- C개 단어에 대한 $D + D \times \log_2 V$ 연산

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.

- We follow previous observation that there can be many different types of similarities between words.
  : simple algebraic operations & search in the vector space for the word closest to X measured by cosine distance
  ex) syntactic: vector("biggest")- vector("big")+ vector("small")= "smallest"
        semantic: vector("Paris")- vector("France")+ vector("Germany")= "Berlin"

4 Results

4.1 Task Description

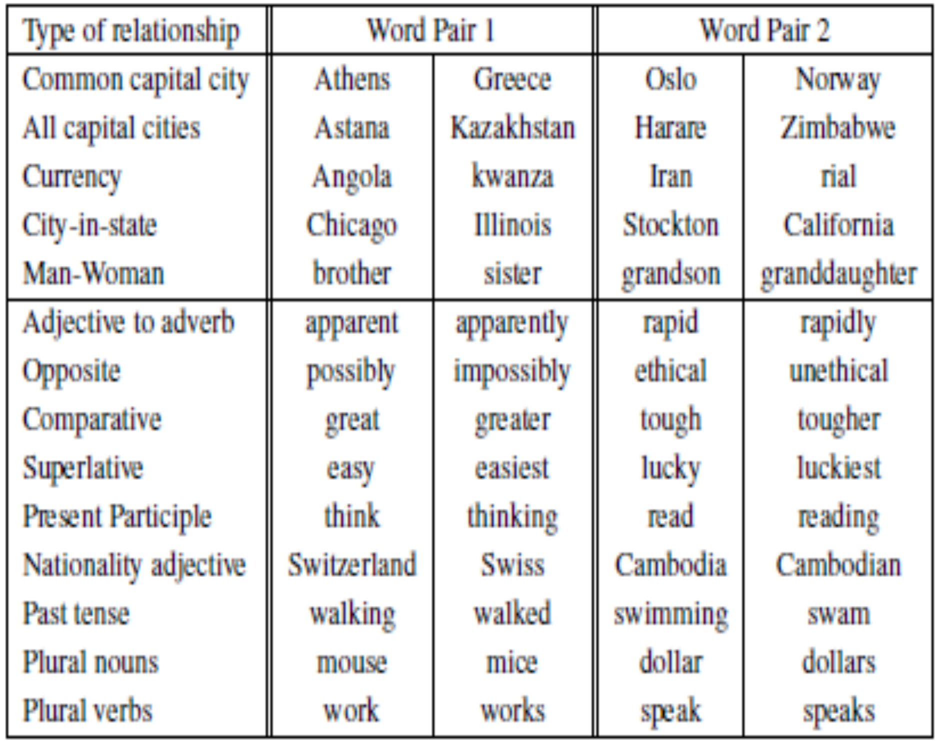


Table 1: *Examples of five types of semantic and nine types of syntactic questions in the Semantic-Syntactic Word Relationship test set.*

Type of relationship	Word Pair 1		Word Pair 2	
Common capital city	Athens	Greece	Oslo	Norway
All capital cities	Astana	Kazakhstan	Harare	Zimbabwe
Currency	Angola	kwanza	Iran	rial
City-in-state	Chicago	Illinois	Stockton	California
Man-Woman	brother	sister	grandson	granddaughter
Adjective to adverb	apparent	apparently	rapid	rapidly
Opposite	possibly	impossibly	ethical	unethical
Comparative	great	greater	tough	tougher
Superlative	easy	easiest	lucky	luckiest
Present Participle	think	thinking	read	reading
Nationality adjective	Switzerland	Swiss	Cambodia	Cambodian
Past tense	walking	walked	swimming	swam
Plural nouns	mouse	mice	dollar	dollars
Plural verbs	work	works	speak	speaks

- To measure quality of the word vectors, we define a comprehensive test set that contains five types of semantic questions, and nine types of syntactic questions.

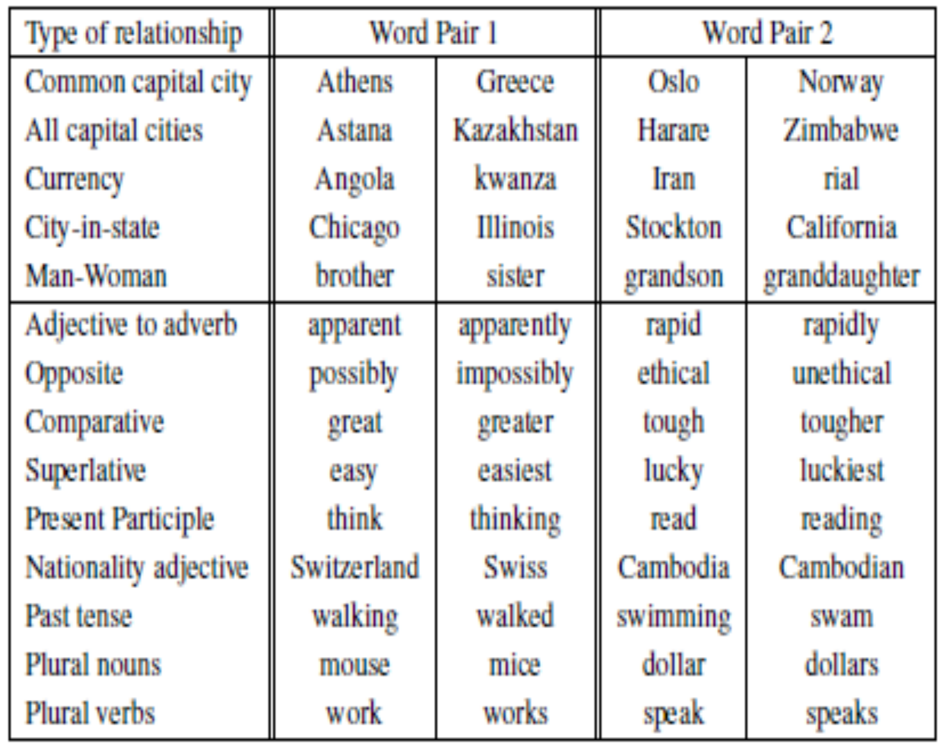
- Overall, there are 8869 semantic and 10675 syntactic questions.

- The questions in each category were created in two steps: first, a list of similar word pairs was created manually. Then, a large list of questions is formed by connecting two word pairs.

4 Results

4.1 Task Description

Table 1: *Examples of five types of semantic and nine types of syntactic questions in the Semantic-Syntactic Word Relationship test set.*



Type of relationship	Word Pair 1		Word Pair 2	
Common capital city	Athens	Greece	Oslo	Norway
All capital cities	Astana	Kazakhstan	Harare	Zimbabwe
Currency	Angola	kwanza	Iran	rial
City-in-state	Chicago	Illinois	Stockton	California
Man-Woman	brother	sister	grandson	granddaughter
Adjective to adverb	apparent	apparently	rapid	rapidly
Opposite	possibly	impossibly	ethical	unethical
Comparative	great	greater	tough	tougher
Superlative	easy	easiest	lucky	luckiest
Present Participle	think	thinking	read	reading
Nationality adjective	Switzerland	Swiss	Cambodia	Cambodian
Past tense	walking	walked	swimming	swam
Plural nouns	mouse	mice	dollar	dollars
Plural verbs	work	works	speak	speaks

- We evaluate the overall accuracy for all question types, and for each question type separately (semantic, syntactic).

- Question is assumed to be correctly answered only if the closest word to the vector computed using the above method is exactly the same as the correct word in the question; synonyms are thus counted as mistakes.

- We believe that usefulness of the word vectors for certain applications should be positively correlated with this accuracy metric. Further progress can be achieved by incorporating information about structure of words.

4 Results

4.2 Maximization of Accuracy

- We have used a Google News corpus for training the word vectors.
- To estimate the best choice of model architecture for obtaining as good as possible results quickly, we have first evaluated models trained on subsets of the training data, with vocabulary restricted to the most frequent 30k words.
- The results using the CBOW architecture with different choice of word vector dimensionality and increasing amount of the training data are shown in Table 2.

4 Results

4.2 Maximization of Accuracy

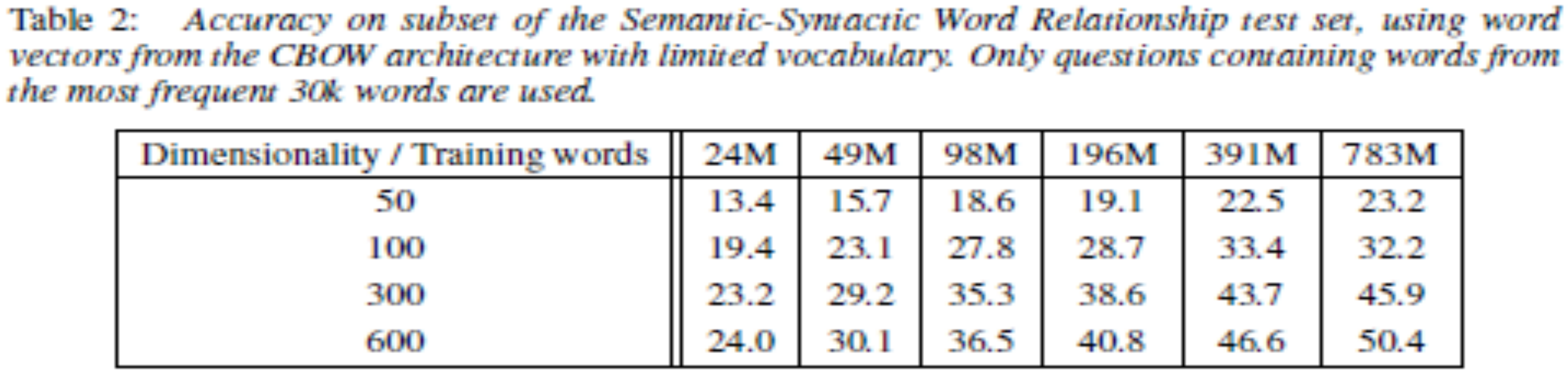


Table 2: *Accuracy on subset of the Semantic-Syntactic Word Relationship test set, using word vectors from the CBOW architecture with limited vocabulary. Only questions containing words from the most frequent 30k words are used.*

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

- It can be seen that after some point, adding more dimensions or adding more training data provides diminishing improvements.

- So, we have to increase both vector dimensionality and the amount of the training data together.

- Increasing amount of training data twice results in about the same increase of computational complexity as increasing vector size twice.

4 Results

4.3 Comparison of Model Architectures

- We compare different model architectures for deriving the word vectors using the same training data and using the same dimensionality of 640 of the word vectors.

- In the further experiments, we use full set of questions in the new Semantic-SyntacticWord Relationship test set, i.e. unrestricted to the 30k vocabulary.

4 Results

4.3 Comparison of Model Architectures

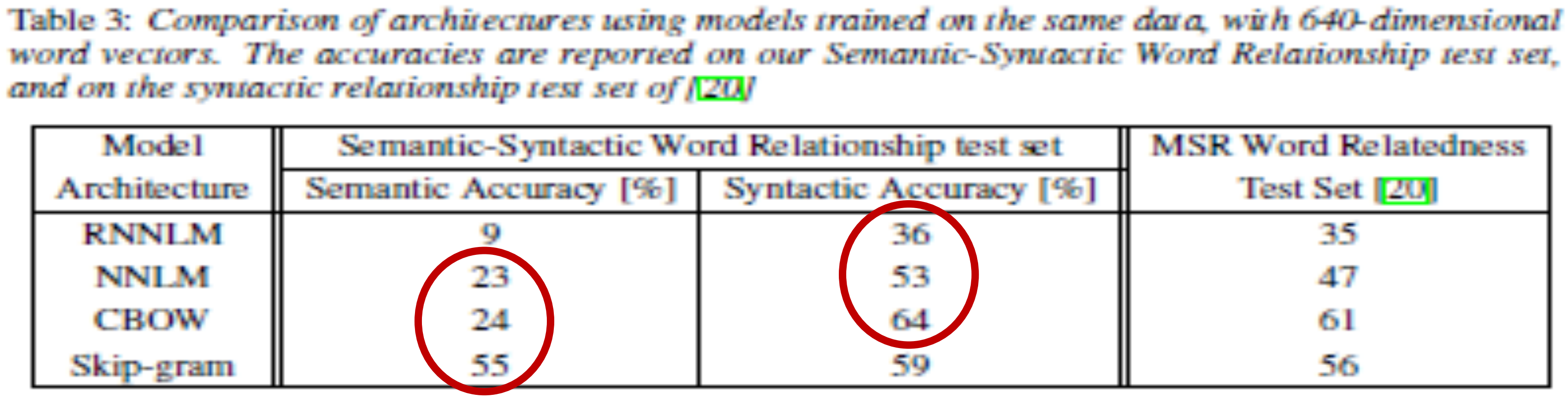


Table 3: *Comparison of architectures using models trained on the same data, with 640-dimensional word vectors. The accuracies are reported on our Semantic-Syntactic Word Relationship test set, and on the syntactic relationship test set of [20]*

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

- The NNLM vectors perform significantly better than the RNN - this is not surprising, as the word vectors in the RNNLM are directly connected to a non-linear hidden layer.

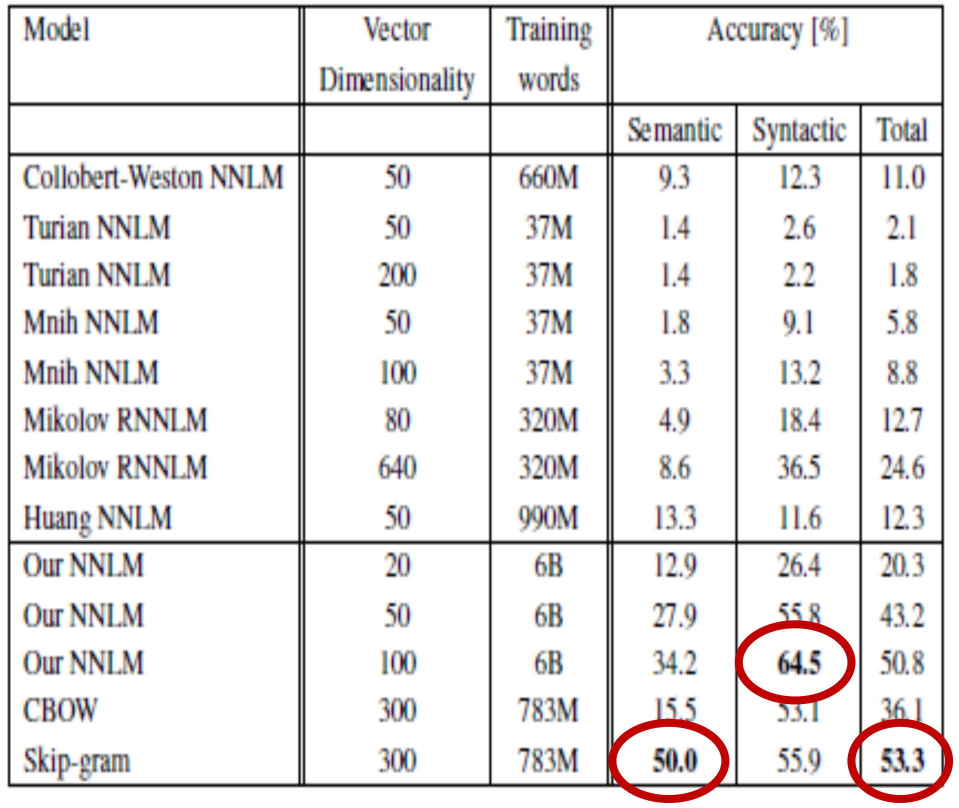
- The CBOW architecture works better than the NNLM on the syntactic tasks, and about the same on the semantic one.

- Finally, the Skip-gram architecture works slightly worse on the syntactic task than the CBOW model (but still better than the NNLM), and much better on the semantic part of the test than all the other models.

4 Results

4.3 Comparison of Model Architectures

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**

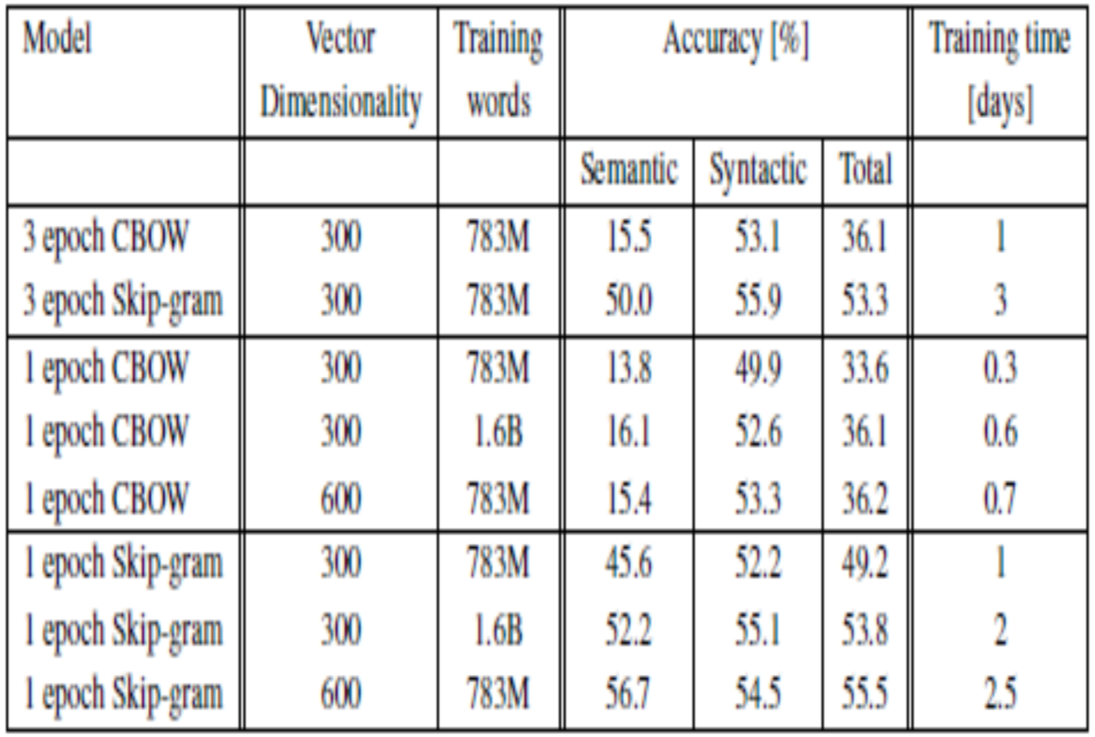
- We evaluated our models trained using one CPU only and compared the results against publicly available word vectors.

- The CBOW model was trained on subset of the Google News data in about a day, while training time for the Skip-gram model was about three days.

4 Results

4.3 Comparison of Model Architectures

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

- Training a model on twice as much data using one epoch gives comparable or better results than iterating over the same data for three epochs, as is shown in Table 5, and provides additional small speedup.

4 Results

4.4 Large Scale Parallel Training of Models

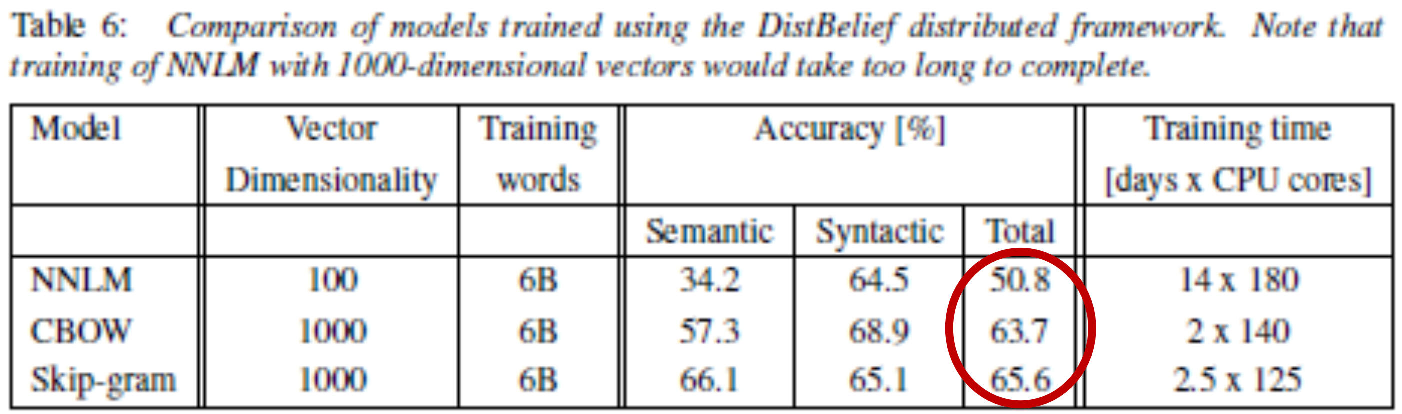


Table 6: *Comparison of models trained using the DistBelief distributed framework. Note that training of NNLM with 1000-dimensional vectors would take too long to complete.*

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

- As mentioned earlier, we have implemented various models in a distributed framework called DistBelief.

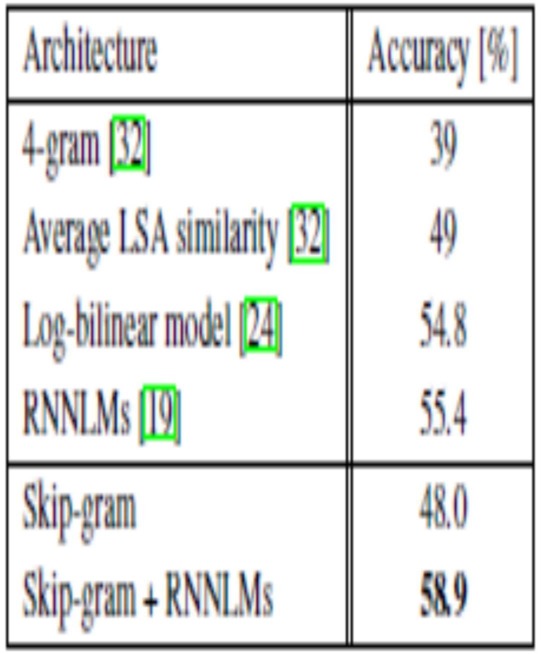
- Below we report the results of several models trained on the Google News 6B data set, with mini-batch asynchronous gradient descent and the adaptive learning rate procedure called Adagrad.

- Note that due to the overhead of the distributed framework, the CPU usage of the CBOWmodel and the Skip-gram model are much closer to each other than their single-machine implementations.

4 Results

4.5 Microsoft Research Sentence Completion Challenge

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

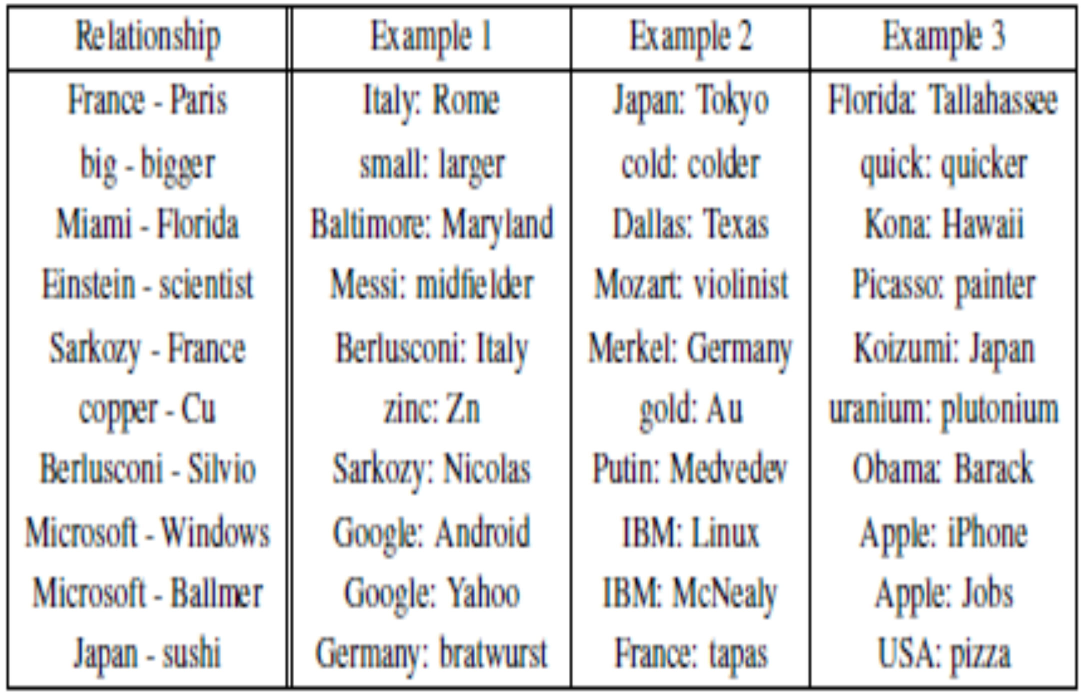


Architecture	Accuracy [%]
4-gram [32]	39
Average LSA similarity [32]	49
Log-bilinear model [24]	54.8
RNNLMs [19]	55.4
Skip-gram	48.0
Skip-gram + RNNLMs	**58.9**

- This task consists of 1040 sentences, where one word is missing in each sentence and the goal is to select word that is the most coherent with the rest of the sentence, given a list of five reasonable choices.
- We have explored the performance of Skip-gram architecture on this task.
- While the Skip-gram model itself does not perform on this task better than LSA similarity, the scores from this model are complementary to scores obtained with RNNLMs, and a weighted combination leads to a new state of the art result 58.9% accuracy.

5 Examples of the Learned Relationships

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

- 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.

6 Conclusion

- 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.

- We also expect that high quality word vectors will become an important building block for future NLP applications.

Distributed Representations of Words and Phrases and their Compositionality

The Skip-gram Model

- The training objective of the Skip-gram model is to find word representations that are useful for predicting the surrounding words in a sentence or a document.

- The objective of the Skip-gram model is to maximize the average log probability

  : $$\frac{1}{T}\sum_{t=1}^{T}\sum_{-c\leq j\leq c, j\neq 0}\log p(w_{t+j}|w_t)$$

\

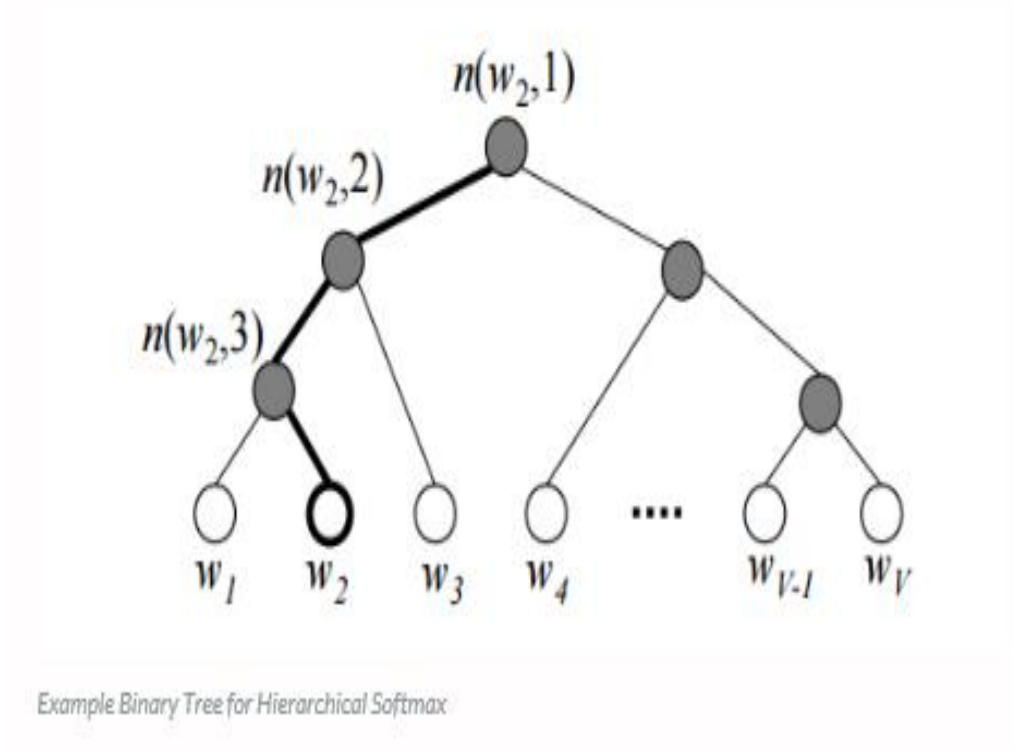
  -> 가정: conditional independence $$P(w_{c-m},\ldots,w_{c-1},w_{c+1},\ldots,w_{c+m}|w_C)=\prod_{j=0,\quad j\neq m}^{2m}P(w_{c-m+j}|w_c)$$

- The basic Skip-gram formulation defines p(wt+j |wt) using the softmax function

  : $$p(w_O|w_I)=\frac{\exp\left({v'_{w_O}}^\top v_{w_I}\right)}{\sum_{w=1}^{W}\exp\left({v'_w}^\top v_{w_I}\right)}$$

The Skip-gram Model

Hierarchical Softmax
: 계산량이 많은 Softmax function 대신 빠르게 계산가능한 multinomial distribution function을 사용하는 방법



Example Binary Tree for Hierarchical Softmax

- 각 단어들을 leaves로 가지는 binary tree 생성

- 해당 단어가 나올 확률 = root에서부터 해당 단어의 leaf로 가는 path에 따른 확률의 곱

- L(w): w라는 leaf에 도달하기 까지의 path의 길이
- n(w, i): root에서부터 w라는 leaf에 도달하는 path 중 i번째 노드
- ch(node): node의 고정된 임의의 한 자식노드
- [[x]]: x가 true일 때 1, false일 때 -1을 반환하는 함수

The Skip-gram Model

Hierarchical Softmax

$$p(w|w_I) = \prod_{j=1}^{L(w)-1} \sigma\left([\![n(w,j+1) = \mathrm{ch}(n(w,j))]\!] \cdot {v'_{n(w,j)}}^{\top} v_{w_I} \right)$$

- 기존 CBOW나 Skip-gram에 있던 Weight Matrix 대신 각각의 internal node에 weight vector 존재

- root에서 단어 *w* 까지 가는 path에 놓여있는 노드의 weight vector와 내적하고, sigmoid 함수를 적용해서 확률화 한 모든 확률의 곱

- 이런 트리형태에서 *L*(*w*) 는 평균적으로 *log*(|*V*|) 에 비례함이 알려져 있으므로 기존|*V*| 개의 summation에서 *log*(|*V*|) 개의 곱으로 computational complexity 감소

- Hierarchical Softmax를 사용하면 [[x]]에 의해 전체 확률에 대한 계산 없이 전체 합을 1로 만들어 줄 수 있어서 multinomial distribution function으로 사용가능

The Skip-gram Model

Negative Sampling

$$\log \sigma({v'_{w_O}}^\top v_{w_I}) + \sum_{i=1}^{k} \mathbb{E}_{w_i \sim P_n(w)} \left[\log \sigma(-{v'_{w_i}}^\top v_{w_I}) \right]$$

- 전체 단어들에 대해 계산을 하는 대신, 일부만 뽑아서 softmax 계산을 하고 normalization을 해주는 방법

- 좌측은 positive sample에 대한 항, 우측은 negative sample들에 대한 항으로 이것을 maximize 하도록 weight을 조정

- 우측은 negative sample이 corpus에 없을 확률을 정의하여 각각을 더하고 log를 취한 형태

- 보통 Negative Sampling에서 샘플들을 뽑는 것은 'Noise Distribution' 을 정의하고 그 분포를 이용하여 단어들을 일정 갯수 뽑아서 사용하는데, 논문에서는 여러 분포를 실험적으로 사용해본 결과 'Unigram Distribution의 3/4승'으로 분포 정의

The Skip-gram Model

Subsampling of Frequent Words

$$P(w_i) = 1 - \sqrt{\frac{t}{f(w_i)}}$$

- 추가적인 방법으로, 논문에서는 'the', 'a', 'in' 등 자주 등장하는 단어들을 확률적으로 제외하여 학습 속도 및 성능 향상

- 각 단어를 p(w)의 확률로 train words에서 제외시키는데, 이 때 t는 빈도가 일정 값 이상일 때만 제외하겠다는 threshold 값인데, 논문에서는 10^-5 의 값을 사용