1. Introduction

1.2 Goal of the Paper

- 성능이 좋은 word vector 를 학습할 수 있는 기술을 소개한다.
- word vector 중 의미가 유사한 단어들은 가까이 위치하고 있을 뿐만 아니라, multiple degrees of similarity 를 보유하고 있다.
- word vector 사이의 연산도 가능하다 : vector("King") - vector("Man") + vector("Woman") = vector("Queen")
- 이번 논문에서는 word vector 의 정확도를 최대화할 수 있는 모델을 소개한다.

1.2 Previous Work

- 단어를 vector 로 나타내는 데에는 긴 역사가 있다.
- Neural Network Language Model (NNLM) : linear projection 을 하는 feedforward NN
- NNLM 의 새로운 구조
    1. 한 개의 hidden layer 를 보유하는 NN 을 사용해서 word vector 를 학습
    2. 학습된 결과를 이용해서 NNLM 을 학습시킨다(?)
- word vector 를 이용해서 많은 NLP 문제들에 적용할 수 있다는 것을 알 수 있었다.
- 하지만, 기존의 방식들은 계산 비용이 매우 크다.(computationally expensive)

2. Model Architectures

- LDA, LSA 등 연속적인 단어표현을 예측하는 모델들이 존재한다.
- 이번 논문에서는 distributed representation 을 사용한다.
    - LSA 는 linear regularities 를 유지하는 성능이 떨어진다.
        - vector("King") - vector("Man") + vector("Woman") = vector("Queen") : 이러한 연산을 하는 성능
    - LDA 는 큰 데이터셋에 대해서 계산하는 비용이 매우 크다.
- 전체적인 성능을 끌어 올리면서, 연산 비용을 최소화한다.

- 이번 논문에서 제안한 training complexity O = E × T × Q
  - E : 학습 epoch 수 (보통 3-50)
  - T : training set 에 있는 단어의 수 (~ 1,000,000,000 개)
  - Q : 각 모델에서 별도로 정의한다.
  - compuatational complexity 는 학습된 전체 모델을 구성하는 파라미터의 수로 정의한다.
- SGD 와 backpropagation 으로 학습을 진행한다.

2.1 Feedforward Neural Net Language Model (NNLM)

- input, projection, hidden, output layer 로 구성된다.
- input layer
  - N 개의 이전 단어들이 1-of-V coding(one-hot-vector)으로 인코딩된다. (V 는 전체 단어의 수)
  - 보통 N = 10 으로 지정
  - projection matrix 를 통해 input 이 projection layer 로 전달된다.
- projection layer
  - N × D dimensionality
  - 상대적으로 저렴한 비용의 연산이다.
  - D = 500 ~ 2000
- hidden layer
  - projection layer 와 hidden layer 사이의 연산에서 상대적으로 큰 비용이 발생한다.
  - hidden layer size H = 500 ~ 1000
- output layer
  - V 차원의 결과를 제공한다.
- 계산 비용 computational complexity Q = N × D + N × D × H + H × V
  - 가장 큰 비용을 차지하는 부분 : H × V
    - hierarchical softmax 를 사용하면 비용이 log(V) 까지 줄어들 수 있다.
  - 비용을 위와 같이 줄여도 N × D × H 가 다음으로 큰 비용을 차지한다.

2.2 Recurrent Neural Net Language Model (RNNLM)

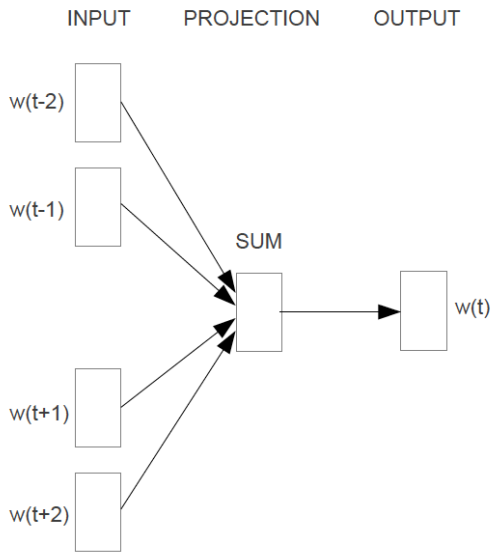
- Feedforward NN 의 몇 가지 단점을 극복하고자 사용 : 입력의 길이를 명시하지 않아도 된다.
- input, hidden, output layer 로 구성된다.
    - hidden layer 를 사용해서 short term memory 를 유지할 수 있도록 한다.
        - 이전 hidden layer 의 정보와 새로운 input 에 대해서 새로운 hidden layer 를 생성하기 때문에, 과거의 정보를 어느정도 유지할 수 있다.
- 계산 비용 computational complexity $Q = H \times H + H \times V$
    - $H \times V$ 는 $H \times \log(V)$ 로 축소될 수 있다. (hierarchical softmax)
    - 대부분의 비용은 $H \times H$ 에서 온다.

2.3 Parallel Training of Neural Networks

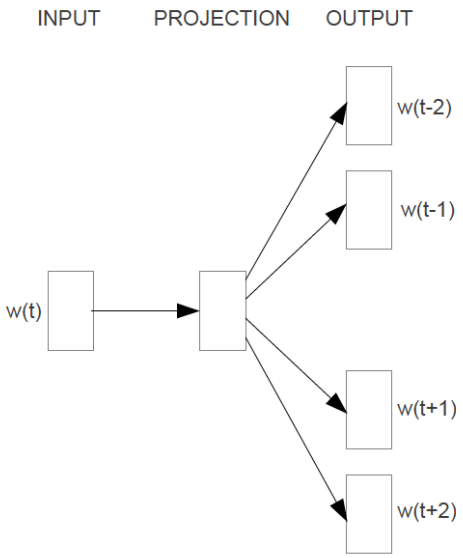
- DistBelief : large-scale distributed framework
    - 동일한 모델을 병렬처리 할 수 있다.
    - gradient update 를 중앙 서버를 통해서 하게 된다. (모든 파라미터를 보유)
    - Adagrad : mini-batch asynchronous gradient descent
- 해당 프레임워크를 사용하면 100 개 이상의 서로 다른 CPU 에서 동시에 학습이 가능하다.

3. New Log-linear Models

- 계산 비용을 최소화하면서 distributed word vector 를 학습할 수 있는 2 가지 모델을 제안
- NNLM, RNNLM 에서는 대부분의 비용이 non-linear hidden layer 에서 발생했다.
    - 위의 방식만큼 정밀한 word vector 가 아닐 수 있지만, 더 효율적으로 데이터를 처리할 수 있는 방법을 제공한다.



CBOW



Skip-gram

CBOW 와 Skip-gram

3.1 Continuous Bag-of-Words Model

- feedforward NNLM 과 유사하지만, non-linear hidden layer 가 모두 제거되고 projection layer 가 모든 단어들과 공유된다.
- 전체 word vector 의 평균값을 사용하게 된다. = bag-of-words model
  - 단어의 순서가 projection 에 영향을 주지 않는다. (그냥 평균을 사용하기 때문에)
- 이래의 단어들도 사용한다.
  - 4 개의 과거 단어들과 4 개의 미래 단어들을 입력으로 사용한다.
  - 입력들을 이용해서 중간 단어를 classify 하는 것이 목표이다.
- 계산 비용 training complexity $Q = N \times D + D \times \log(V)$
- 해당 모델을 CBOW 라고 부른다.
  - 연속적인 context 를 사용하기 때문이다.

3.2 Continuous Skip-gram Model

- CBOW 와 다르게, 중간 단어를 이용해서 주변 단어를 예측하는 방식이다.
- 중간 단어를 log-linear classifier 에 입력으로 제공한다.

        - 현재 단어의 일정 범위 내에 있는 이전/이후의 단어를 예측한다.
        - 범위를 증가시키면 word vector 의 성능이 좋아지지만, 계산 비용이 증가한다.
        - 또한, 현재 단어와 멀리 떨어진 단어는 해당 단어와 관련이 있을 가능성이 낮기 때문에, 낮은 weight 를 제공한다.
    - 계산 비용 training complexity Q = C × (D + D × log(V))
        - C : 단어 사이의 최대 거리
        - 이후 실험에서는 C = 10 을 사용한다.

4. Results

- word similarity
    - big 와 biggest 의 관계처럼 small 에 대응하는 단어는 뭘까?
        - vector("biggest") - vector("big") + vector("small") = vector("smallest")
- 고차원 벡터를 학습하면 단어들 사이의 semantic relationship 을 학습할 수 있다.
    - city-country 등의 관계를 학습할 수 있다.

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

Examples of five types of semantic and nine types of syntactic questions

4.1 Task Description

- Test set of 5 types of semantic questions, 9 types of syntactic questions.

- 8,869 semantic and 10,675 syntactic questions
    1. 수작업으로 유사한 단어들을 골라낸다.
    2. 2 개의 단어를 짝을 지어서 큰 질문 리스트를 생성한다.
- 평가(evaluation)
    - 예측한 결과 벡터에 가장 가까운 단어가 질문과 완벽하게 일치하는 경우 정답으로 처리한다.
        - 100%의 정확도는 불가능하다는 뜻이다.
        - 동의어는 오답으로 처리된다.

4.2 Maximization of Accuracy

- Google News 를 사용해서 word vector 를 학습했다.
    - 6B 개의 토큰으로 구성된다.
    - 단어의 개수를 가장 자주 나타나는 100 만개로 제한했다.

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

CBOW 모델에 대한 성능 (차원과 학습 데이터 크기의 관계)

- 어느 순간 이후로 차원을 늘리거나, 학습 데이터를 주가하는 것으로는 나타나는 성능 향상은 적다.
    - 그러므로, 성능 향상을 위해 차원을 늘림과 동시에 학습 데이터를 추가해야 한다.

4.3 Comparison of Model Architectures

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

640 차원의 word vector 로 구성된 동일한 학습 데이터로 학습했을 때의 성능 비교

- 640 dimension 의 word vector 를 사용한 동일한 학습 데이터를 이용해서 서로 다른 모델 구조를 비교한다.

- 학습 데이터는 LDC 말뭉치들을 표함한다.
- 기존에 학습된 NNLM 과 비교한다.
  - 640 개의 hidden units
  - DisBelief 병렬 처리를 통해 학습했다.
  - 이전 8 개의 단어를 입력으로 제공하는 방식으로 학습했다.

- 결과
  - RNN 은 syntactic 문제에 대해서 성능이 더 좋다.
  - NNLM 은 모든 항목에 대해서 RNNLM 에 비해 성능이 좋다.
  - CBOW 는 NNLM 에 비해 syntactic 성능은 좋지만 semantic 성능은 유사하다.
  - Skip-gram 은 CBOW 에 비해 syntactic 성능은 조금 떨어지지만, semantic 성능은 모든 모델들에 비해서 좋다.

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**

다른 word vector 들과 Semantic-Sytactic Word Relationship test set 에 대한 성능 비교

- 공개된 다른 word vector 와 성능을 비교해봤다.
  - Semantic-Syntactic Word Relationship test set 에 대한 성능을 비교했다.
  - 비교에 사용된 CBOW 모델을 Google News 에 대해서 하루 동안 학습되었고, Skip-gram 은 3 일 정도 학습되었다.

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

학습 데이터의 크기와 epoch 의 관계

- 상대적으로 적은 데이터를 여러번의 epoch 를 통해 학습하는 것보다, 많은 데이터를 이용해 1 번의 epoch 만 학습하는 것이 더 좋은 성능을 제공한다는 것을 알 수 있었다.

4.4 Large Scale parallel Training of Models

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

모델에 따른 학습 시간

- Google News 데이터에 대해 학습된 몇 개의 모델 통계를 제공한다.
- DistBelief 를 사용해서 학습했으며, 50~100 개의 복제본을 생성해서 학습했다.
- CBOW 와 Skip-gram 의 CPU 사용량은 거의 비슷하다는 것을 알 수 있다.
- NNLM 의 벡터 차원을 1000 으로 사용하면 연산 시간이 너무 오래 걸려 100 으로만 제한했다.

4.5 Microsoft Research 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**

Microsoft Research Sentence Completion Challenge 의 결과

- Microsoft Research Sentence Completion Challenge 는 하나의 단어가 없는 1,040 의 문장으로 구성된다.
    - 빈 단어에 알맞는 단어를 찾아내는 것이 목표이다.
- Skip-gram 구조를 적용시켜봤다.
    - Skip-gram 만으로는 높은 성능을 제공하지 못한다.
    - RNNLM 과 함께 적용한 경우 58.9%라는 높은 성능을 제공했다.

6. Conclusion

- 벡터 표현에 대한 성능을 다양한 모델을 이용해서 다양한 task 에 대해서 측정했다.
- 단순한 구조로도 뛰어난 성능의 word vector 를 학습할 수 있다는 것을 알 수 있었다.
    - NNLM 과 RNNLM 과 비교해서
- 연산 비용이 매우 작기 때문에, 큰 차원의 큰 데이터셋에 대한 학습도 가능하다.
- SemEval-2012 Task 2 에서 state-of-art 의 성능을 제공했다.