**Efficient Estimation of Word Representations in Vector Space**

Abstract

Two novel model architecture(for word representation)

Word similarity에 사용, much lower computational cost로 large improvements. (학습 시 하루도 안걸림, 1.6 billion words에 대해)

1. Introduction

이전 NLP => words = atomic unit => but just index in vocab & no similarity

심플 모델 on large dataset > 복잡 모델 on small dataset 이라는 근거하에 학습(N-gram이 그랬다. 어차피 더 복잡해도 많은 데이터 수용 못하게 된다.. <딥러닝 이전>) 하지만 data를 많이 구하기 어렵고 따라서 심플 모델은 발전하기 어렵다. 기술 발전이 필요했다.

딥러닝/머신러닝은 distributed representations of words를 이용(N-gram보다 뛰어남)  
**비슷한 위치에서 등장하는 단어들은 비슷한 의미를 가진다**

1.1 goals of the paper

High-quality word vector를 huge dataset에서 학습하는 것!(vocab에는 수백만 / dataset은 수 billion) / 이전 모델(딥러닝 이전) => 많은 dataset을 소화하지 못한다.(벡터의 차원도 크면 안된다. 약 50~100정도)

본 논문의 representation은 비슷한 단어는 가깝게 위치 / 또한 multiple degrees of similarity를 가짐(다른 논문인데 뭘까? Linguistic Regularities in Continuous Space Word Representations)

Similar ending? 무슨소리..? 뭘까?

Syntactic regularity, semantic regularity를 동시에 만족!

1.2 previous work

Continuous vector로 word representation은 긴 역사(NNLM이 많이 쓰임 과거에) / NNLM이 약가 업그레이드(single hidden layer 추가되어) 우리는 업글된 이 NNLM을 이용

하지만 매우 비싼 연산 과정이 학습 중에 필요하다.

2. model architecture

LSA, LDA => continuous representations of words. (이전의) / 단어 사이의 regularity를 보존하면서 LSA 보다 뛰어나다. LDA는 그리고 연산 비싸다.

주로 평가할 것 1. Computational complexity(모델 파라미터가 얼마나 있는지 확인하면서) 2. Accuracy maximizing(complexity는 줄이면서)

Training complexity => O = E \* T \* Q (EPOCH, 트레이닝 단어 수, Q는 모델의 파라미터 같은데 아직 언급 안됨) / E = 3~50, T = 1 billion / SGD와 BP 이용

2.1 Feedforward Neural Net Language Model (NNLM)

Input, projection, hidden, output layer로 구성

\* input layer => 1-of-V coding으로 인코딩 된다. (V는 Vocab size)  
\* projection layer P(N\*D의 shared projection matrix를 통해)로 project 된다. (N은 그냥 입력 단어 수??) =>이렇게 되면 N개 단어가 각각 D 차원으로 project 되나? (projection layer는 싼 연산)  
\* projection => projection과 hidden layer 사이는 비싼데 그 이유는 500~2000 / 500~1000 개의 unit을 가지기 때문이다. 또한 probability distribution을 모든 단어에 대해 진행한다. => output layer는 V dimension => 위의 Q = N\*D + N\*D\*H + H\*V

Hierarchical version의 softmax를 이용하거나 normalized 되지 않은 모델을 이용해 normalizing 피함?? => 좀더 practical solution => binary tree representation of vocab으로 log2(V)로 unit 줄인다.(output layer에서) => 따라서 N\*D\*H가 지배적으로 된다.

$우리 모델에서는 hierarchical softmax를 사용(Huffman binary tree) => 자주 등장 언어는 짧게 해서 output unit을 줄인다. 그냥 balance tree => log2(V) vs Huffman binary tree => unigram\_perplexity(V)) // 하지만 N\*D\*H가 지배적이라 그렇게 뭐 그런건 아님. => softmax normalization에 더 의존적?

2.2 Recurrent Neural Net Language Model(RNNLM)

RNN => fixed length N을 조절 / projection layer가 없다. => Q = H\*H + H\*V

2.3 Parallel Training of Neural Networks

병렬로 학습

3. New Log-linear Models

Minimize computational complexity! 이전 모델들(NNLM 말고) => hidden layer에서 complexitiy 증가 = 깊어질수록 복잡할수록 매력적 하지만 많은 데이터를 학습하기 위해선 simple model이 필요하고 우리는 그 simple 모델을 업그레이드 시킴

즉, 심플 모델로 continuous word vectors가 학습되고, N-gram NNLM으로 distributed representation of words가 train??

3.1 Continuous Bag-of-Words Model

Feedforward NNLM 모델과 비슷, non-linear hidden layer는 제거되고 projection layer는 모든 words에서 share => 모든 단어가 same position으로 project됨.(vector가 averge됨..?)

단어의 순서는 does not influence the projection!

3.2 다시!

<https://ratsgo.github.io/from%20frequency%20to%20semantics/2017/03/29/NNLM/>

<https://ratsgo.github.io/from%20frequency%20to%20semantics/2017/03/30/word2vec/>

<https://wikidocs.net/22660>

<https://reniew.github.io/21/>

<https://reniew.github.io/22/>

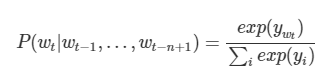
NPLM/NNLM – Bengio(2003)

One-hot => 10만 차원 => 메모리 문제/sparse vector!(비효율) => one-hot vector는 서로 독립/ orthogonal => 단어의 관계, 유사를 알아내기 힘듬 => distributed representations이 해결

Distributed representation

차원을 낮추자!

NNLM은 이걸 이용!

보통 task가 다음 단어 맞추기 => 

이전단어들 주어졌을 때 (n-1개 단어로 n번째 단어를 맞추는 N-gram 모델)=> ‘간다’의 확률을 높이기,, y-wt는 wt라는 단어에 해당하는 score vector(V 차원) 출력 => y-wt에 softmax! => V차원의 확률 벡터

(wt는 one-hot, C는 초기에 랜덤 설정, xt는 차원수) => C는 m\*V 차원 => 이게 projection

=> w-t-1 ~ w-n+1 가 주어졌을 때 wt가 간다일 확률!을 높이자

y-wt라는 score 벡터에 softmax => 확률 벡터 => 가장 높은 확률 단어 vs ‘간다’ 인덱스와 비교해 backpropagation 진행

CBOW -> 블로그

RNNLM

NNLM -> 고정된 개수의 단어를 입력으로 받음

출력이 입력으로 들어가는 대신 teacher forcing을 이용(그 시점에서 ylabel이 들어감.) => 출력이 입력으로 들어가는 경우 한번 잘못 출력할 경우 연쇄적으로 악영향 => 훈련시간 느려짐 그외 과정은 다 같다.