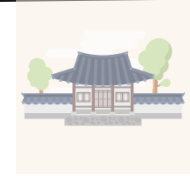
집현전 초급반 허치영

A Neural Probabilistic Language Model

Bengio, Y., Ducharme R., Vincent P. & Jauvin C. (2003)

Contents

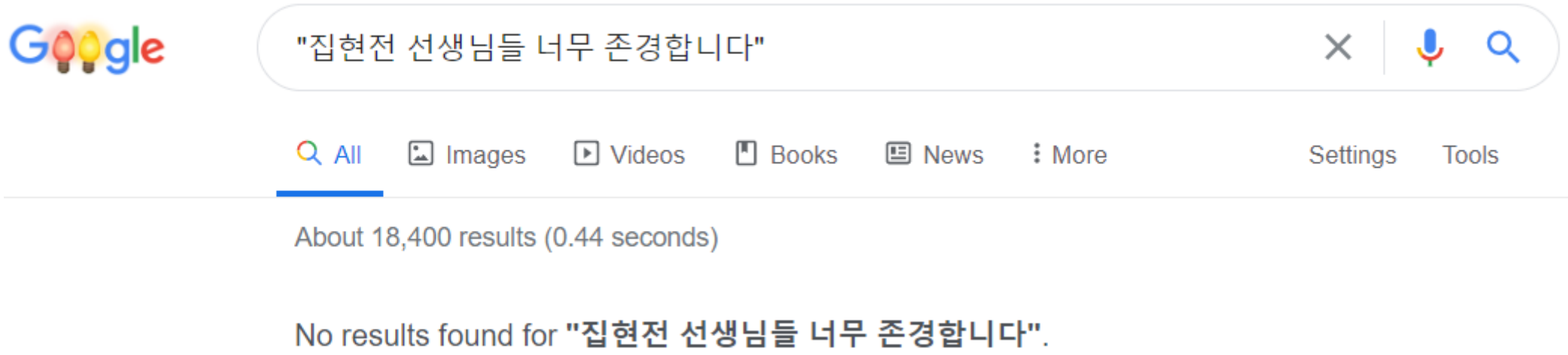


1. Abstract
2. Introduction
3. Model Architecture
4. Experiment & Results
5. Reference

Abstract

기존 Statistical Language Model의 문제점

- Curse of dimensionality
  - Training dataset에서 등장하지않은 word sequence가 test 과정에서 등장할 경우가 많기 때문



Curse of Dimensionality

- Discrete random variables(word sequence)간의 joint distribution 계산할 때 흔히 발생
  - ex) Vocab size=100,000인 단어 중 10개의 연속된 단어의 결합 분포는 $100{,}000^{10} - 1 = 10^{50} - 1$개의 parameter 필요하게 됨

1. Introduction

A fundamental problem that makes language modeling and other learning problems difficult is the *curse of dimensionality*. It is particularly obvious in the case when one wants to model the joint distribution between many discrete random variables (such as words in a sentence, or discrete attributes in a data-mining task). For example, if one wants to model the joint distribution of 10 consecutive words in a natural language with a vocabulary V of size 100,000, there are potentially $100\,000^{10} - 1 = 10^{50} - 1$ free parameters. When modeling continuous variables, we obtain generalization more easily (e.g. with smooth classes of functions like multi-layer neural networks or Gaussian mixture models) because the function to be learned can be expected to have some local smoothness properties. For discrete spaces, the generalization structure is not as obvious: any change of these discrete variables may have a drastic impact on the value of the function to be estimated, and when the number of values that each discrete variable can take is large, most observed objects are almost maximally far from each other in hamming distance.

Introduction

Long-term Dependency

- 기존의 Statistical model
  - Conditional Probability : $\hat{P}(w_1^T) = \prod_{t=1}^{T} \hat{P}(w_t|w_1^{t-1})$
  - 이전에 등장한 모든 단어들의 조건부 확률의 곱으로 표현함

- N-gram model
  - Conditional Probability : $\hat{P}(w_t|w_1^{t-1}) \approx \hat{P}(w_t|w_{t-n+1}^{t-1})$
  - 앞선 n-1개의 단어만으로 Statistical model의 조건부확률 근사
  - Sequence내 더 가까운 단어일수록 통계적으로 더 의존적이기 때문

- Problems
  - N-gram sequence가 training dataset에 등장하지 않을 경우 zero probability 부여하게 됨
    - Simple answer : Smaller context (작은 n-gram) 사용 (ex: back-off/smoothed trigram)
  - 장기의존성 포착 불가

> Long-term dependency 예시

한국에 한평생동안 살아온 그는 단 한번도 해외여행을 다녀오거나 다른 언어를 배울 기회가 없어 오직 ____밖에 구사할 줄 모른다.

Abstract

기존 Statistical Language Model 문제 해결시도

- Distributed representation을 이용하자
  - 처음 등장하는 Word sequence라도 이미 문장에 등장했던 sequence내의 단어와 비슷하면 높은 확률을 부여하여 Generalization 획득

EX)
The cat is walking in the bedroom
A dog is running in a room

Abstract

A goal of statistical language modeling is to learn the joint probability function of sequences of words in a language. This is intrinsically difficult because of the **curse of dimensionality**: a word sequence on which the model will be tested is likely to be different from all the word sequences seen during training. Traditional but very successful approaches based on n-grams obtain generalization by concatenating very short overlapping sequences seen in the training set. We propose to fight the curse of dimensionality by **learning a distributed representation for words** which allows each training sentence to inform the model about an exponential number of semantically neighboring sentences. The model learns simultaneously (1) a distributed representation for each word along with (2) the probability function for word sequences, expressed in terms of these representations. Generalization is obtained because a sequence of words that has never been seen before gets high probability if it is made of words that are similar (in the sense of having a nearby representation) to words forming an already seen sentence. Training such large models (with millions of parameters) within a reasonable time is itself a significant challenge. We report on experiments using neural networks for the probability function, showing on two text corpora that the proposed approach significantly improves on state-of-the-art n-gram models, and that the proposed approach allows to take advantage of longer contexts.

Introduction

Focus of this paper

- 바로 앞의 1~2 단어보다 더 멀리 있는 information 포착하기
- 단어간 유사성 찾기

There are at least two characteristics in this approach which beg to be improved upon, and that we will focus on in this paper. First, it is not taking into account contexts farther than 1 or 2 words,[1] second it is not taking into account the "similarity" between words. For example, having seen the sentence "`The cat is walking in the bedroom`" in the training corpus should help us generalize to make the sentence "`A dog was running in a room`" almost as likely, simply because "`dog`" and "`cat`" (resp. "`the`" and "`a`", "`room`" and "`bedroom`", etc...) have similar semantic and grammatical roles.

Proposed Approach

- Vocabulary의 각 단어를 실수 공간에서 distributed word feature vector로 표현
  - 각 feature vector는 적은 수의 특징으로 벡터 공간의 한 점에 mapping됨
  - Feature vector는 이전의 의미적 지식을 활용해 초기화 가능
- Word sequence의 joint probability function을 해당 sequence내 단어들의 feature vector로 표현
  - Probability function은 주어진 이전단어의 다음 단어의 conditional probability의 곱으로 나타냄
  - Function의 parameter는 training data의 log-likelihood를 최대화하며 반복해서 조정
- Word feature vector와 probability function의 parameter 동시에 학습

Introduction

Previous Work

- Decomposing Joint probability as product of Conditional probability
  - 2000년에 제안된 방법
  - $\hat{P}(Z_1 = z_1, \cdots, Z_n = z_n) = \prod_i \hat{P}(Z_i = z_i | g_i(Z_{i-1} = z_{i-1}, Z_{i-2} = z_{i-2}, \cdots, Z_1 = z_1))$
  - $g_i()$는 주어진 이전의 Z를 이용해서 Z_i의 조건부 분포를 표현하는 parameter를 계산함

- Discovering Word Similarity
  - 1992년에 제안된 방법
  - 단어의 clustering을 학습하여 단어들을 특정 연관성에 따라 class로 similarity 학습
- Vector space Representation
  - LSI (Latent Semantic Indexing)
  - 같은 문서 내 동시 등장 확률을 이용하여 표현함

Difference from Previous Work

- Decomposing Joint probability as product of Conditional probability
  - 이전의 방법은 각 time별로 다른 g() 사용함
  - 논문에서 제안된 방법은 각 time별로 동일한 g() 사용 (Parameter sharing)

- Discovering Word Similarity
  - Class 대신 Distributed feature vector(연속 실수 벡터)를 통해서 단어간 유사도를 표현

Model Architecture

Overall structure

- Objective
  - Training set 외의 경우에서 높은 likelihood얻는 model $f(w_t,\cdots,w_{t-n+1})=\hat{P}(w_t|w_1^{t-1})$ 학습
  - Perplexity 최소화 $\min\left\{\hat{P}(w^t|w_1^{t-1})^{-\frac{1}{t}}\right\}$

- Decompose f() in two parts
  1. |V| x m 크기의 행렬 C에서 feature vector 참조 C(i)
     - C의 각 행은 한 단어의 feature vector 표현
  2. Matrix C로 표현되는 단어에 대한 Probability function
     - $f(i,w_{t-1},\cdots,w_{t-n+1}) = g\big(i,C(w_{t-1}),\cdots,C(w_{t-n+1})\big)$
     - g는 word sequence의 단어의 feature vector를 다음 단어 w_t에 대한 Conditional probability에 mapping
     - g의 출력은 $\hat{P}(w^t = i|w_1^{t-1})$ (i는 행렬 C의 i번째 word feature vector)
     - Function g는 feed-forward 또는 recurrent neural network와 같은 형태로 구현됨

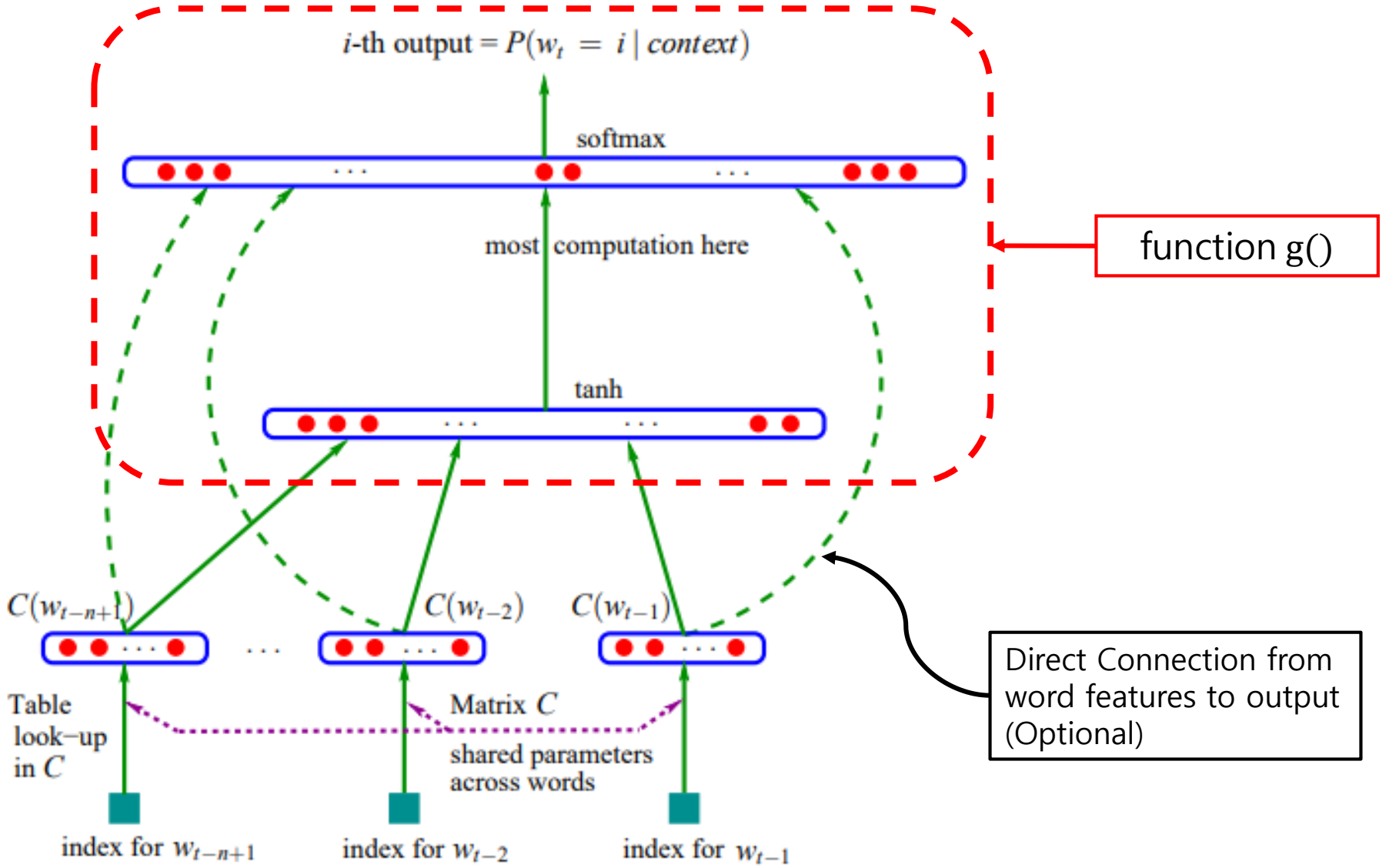
We decompose the function $f(w_t,\cdots,w_{t-n+1}) = \hat{P}(w_t|w_1^{t-1})$ in two parts:

1. A mapping C from any element i of V to a real vector $C(i) \in \mathbb{R}^m$. It represents the *distributed feature vectors* associated with each word in the vocabulary. In practice, C is represented by a $|V| \times m$ matrix of free parameters.

2. The probability function over words, expressed with C: a function g maps an input sequence of feature vectors for words in context, $(C(w_{t-n+1}),\cdots,C(w_{t-1}))$, to a conditional probability distribution over words in V for the next word w_t. The output of g is a vector whose i-th element estimates the probability $\hat{P}(w_t = i|w_1^{t-1})$ as in Figure 1.

$$f(i,w_{t-1},\cdots,w_{t-n+1}) = g(i,C(w_{t-1}),\cdots,C(w_{t-n+1}))$$

Model Architecture



Model Architecture

Overall structure

- Training
  - $L = \frac{1}{T}\sum_t \log f(w_t, w_{t-1}, \cdots, w_{t-n+1}\,;\,\theta) + R(\theta)$
  - Training corpus의 log-likelihood 최대화하며 학습
  - 논문에서 $R(\theta)$는 weight decay penalty, neural network의 weight parameter와 matrix C에 적용 (bias 제외)

- Output layer
  - Softmax 사용
  - $\hat{P}(w_t|w_{t-1}, \cdots w_{t-n+1}) = \frac{e^{y_{w_t}}}{\sum_i e^{y_i}}$

Model Architecture

Structure detail

- y_i & $Parameters$
  - 각 i번째 출력단어에 대한 unnormalized log-probability
  - $y = b + Wx + U\tanh(d + Hx)$
  - 이 때 $x = \left(C(w_{t-1}), C(w_{t-2}), \cdots, C(w_{t-n+1})\right)$, concatenation of the input word features.
  - Parameters : b, W, U, d, H
  - W는 direct connection 없으면 0로 설정
  - Number of parameters : $|V|(1 + nm + h) + h(1 + (n-1)m)$, dominating factor : $|V|(nm + h)$
  - Parameter의 수는 $|V|$에 따라 선형적으로 증가

- Stochastic gradient ascent
  - Training corpus의 t번째 단어로 gradient ascent 반복 수행
  - $\theta \rightarrow \theta + \varepsilon \frac{\partial \log \hat{P}(w_t | w_{t-1}, \cdots, w_{t-n+1})}{\partial \theta}$ (ε : learning rate)

$\|V\|$	Number of words in Vocabulary
n	Size of input sequence(x)
m	Number of features (=dimension)
h	Number of hidden units

Parameter	Role	Size
b	Output layer bias	$\|V\|$
d	Hidden layer bias	h
U	Hidden-output weight	$\|V\| \times h$
W	Feature-output weight	$\|V\| \times (n-1)m$
H	Hidden layer weight	$h \times (n-1)m$
C	Feature vector	$\|V\| \times m$

Experiment & Results

Datasets

- Brown corpus
  - Total size : 1,181,041 words
  - Training set : first 800,000 words
  - Validation set : 200,000 words
  - Test set : 181,041 words
  - Original Vocabulary size : 47,578 (문장부호 포함, 대소문자 구분)
  - Reduced Vocabulary size : 16,383 (3회 이하 등장 단어 심볼 하나로 합침)

- Associated Press (AP) News (from 1995 to 1996)
  - Training set : 14 million (13,994,528) words
  - Validation set : 1 million (963,138) words
  - Test set : 1 million (963,071) words
  - Original Vocabulary size : 148,721 (문장부호 포함)
  - Reduced Vocabulary size : 17,964 (대문자 소문자로 변환, 희귀 단어와 고유명사 각각 특정 심볼에 mapping)

Experiment & Results

Experiment

- Compared models
  - Interpolated or smoothed trigram model
  - Back-off n-gram models with the Modified Kneser-Ney algorithm
  - Class-based n-gram models

- Details of experiment
  - Test set의 perplexity 비교
  - Initial learning rate : $\varepsilon_0 = 10^{-3}$
  - Learning rate decay : $\varepsilon_t = \frac{\varepsilon_0}{1+rt}$, ($r = 10^{-8}$)
  - Initialized word features randomly
  - Brown Corpus
    - Epochs : 10~20
    - Weight decay penalty : 10^{-4}
  - AP News Corpus
    - Epochs : 5
    - Weight decay penalty : 10^{-5}

Experiment & Results

Results

Table 1 Comparative Results on the Brown Corpus

	n	c	h	m	direct	mix	train.	valid.	test.
MLP1	5		50	60	yes	no	182	284	268
MLP2	5		50	60	yes	yes		275	257
MLP3	5		0	60	yes	no	201	327	310
MLP4	5		0	60	yes	yes		286	272
MLP5	5		50	30	yes	no	209	296	279
MLP6	5		50	30	yes	yes		273	259
MLP7	3		50	30	yes	no	210	309	293
MLP8	3		50	30	yes	yes		284	270
MLP9	5		100	30	no	no	175	280	276
MLP10	5		100	30	no	yes		265	**252**
Del. Int.	3						31	352	336
Kneser-Ney back-off	3							334	323
Kneser-Ney back-off	4							332	321
Kneser-Ney back-off	5							332	321
class-based back-off	3	150						348	334
class-based back-off	3	200						354	340
class-based back-off	3	500						326	**312**
class-based back-off	3	1000						335	319
class-based back-off	3	2000						343	326
class-based back-off	4	500						327	312
class-based back-off	5	500						327	312

N-gram중 가장 성능 좋은 것과 비교했을 때 성능 크게 차이 남

N	Order of the model
C (for class-based model)	Number of word classes
H	Number of hidden units
M	Number of word features
Direct	Whether direct connection
Mix	Whether output of prob mixed with output of trigram

Table 2 Comparative Results on the AP News Corpus

	n	h	m	direct	mix	train.	valid.	test.
MLP10	6	60	100	yes	yes		104	**109**
Del. Int.	3						126	132
Back-off KN	3						121	127
Back-off KN	4						113	119
Back-off KN	5						112	**117**

References

[1] *A Neural Probabilistic Language Model, Bengio et al.*
[2] *https://wikidocs.net/45609*
[2] *https://ratsgo.github.io/from%20frequency%20to%20semantics/2017/03/29/NNLM/*
[3] *Efficient Estimation of Word Representations in Vector Space, Mikolov et al.*