Paper Review # 2

Distributed Representations of Words and Phrases and their Compositionality

(T. Mikolov et al., 2013)

Keywords :
Skip-gram, # Hierarchical Softmax,
Negative Sampling # Subsampling

Seunghan Lee
Department of Statistics & Data Science

Contents

Distributed Representations of Words and Phrases and their Compositionality

Tomas Mikolov
Google Inc.
Mountain View
mikolov@google.com

Ilya Sutskever
Google Inc.
Mountain View
ilyasu@google.com

Kai Chen
Google Inc.
Mountain View
kai@google.com

Greg Corrado
Google Inc.
Mountain View
gcorrado@google.com

Jeffrey Dean
Google Inc.
Mountain View
jeff@google.com

Abstract

The recently introduced continuous Skip-gram model is an efficient method for learning high-quality distributed vector representations that capture a large number of precise syntactic and semantic word relationships. In this paper we present several extensions that improve both the quality of the vectors and the training speed. By subsampling of the frequent words we obtain significant speedup and also learn more regular word representations. We also describe a simple alternative to the hierarchical softmax called negative sampling.

An inherent limitation of word representations is their indifference to word order and their inability to represent idiomatic phrases. For example, the meanings of “Canada” and “Air” cannot be easily combined to obtain “Air Canada”. Motivated by this example, we present a simple method for finding phrases in text, and show that learning good vector representations for millions of phrases is possible.

1. Introduction

- **Skip-gram model** :
  - **(1) EFFICIENT** method of learning **(2) DISTRIBUTED** vector representation
  - Able to capture the semantics of words! … ex) King – Man + Woman = Queen
- Proposes an **extension of Skip-gram model**
  - Improvement in both “quality of vector representation” & “training speed”
  - **1) Subsampling of frequent words**
  - **2) Negative Sampling**
- Limitations of word representation : “Inability to represent idiomatic phrases”

1. Introduction

- **Skip-gram model** :
  - **(1) EFFICIENT** method of learning **(2) DISTRIBUTED** vector representation
  - Able to capture the semantics of words! … ex) King – Man + Woman = Queen
- Proposes an **extension of Skip-gram model**
  - Improvement in both “quality of vector representation” & “training speed”
  - **1) Subsampling of frequent words**
  - **2) Negative Sampling**
- Limitations of word representation : “Inability to represent idiomatic phrases”

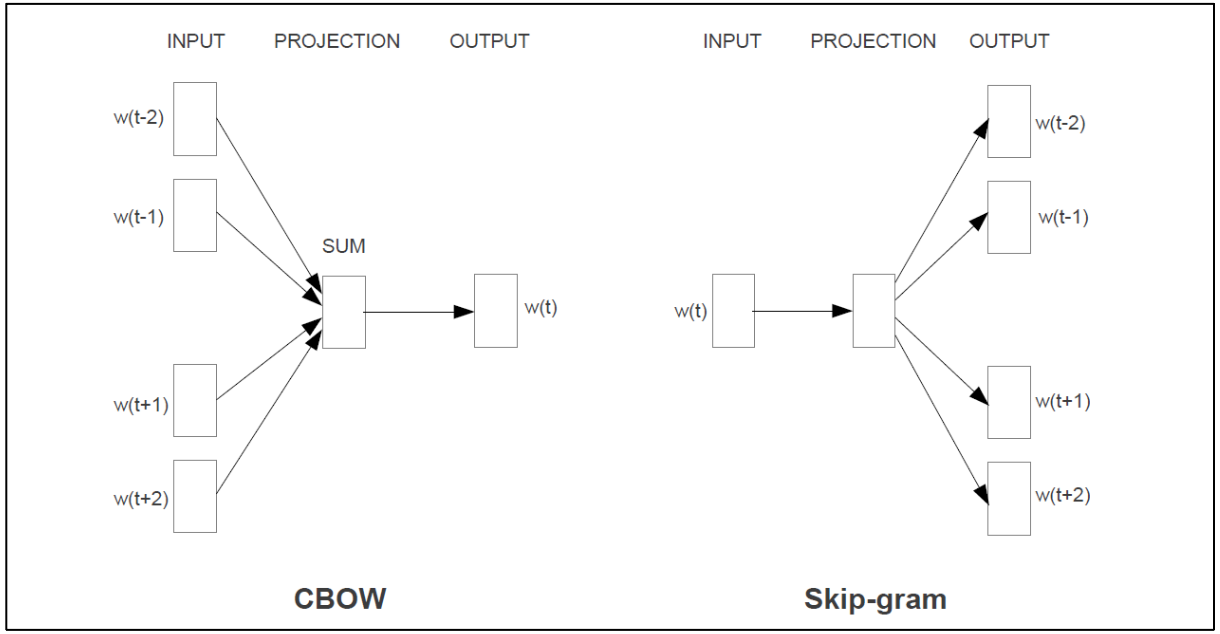
1. Introduction

- **Skip-gram model** :
  - **(1) EFFICIENT** method of learning **(2) DISTRIBUTED** vector representation
  - Able to capture the semantics of words! … ex) King – Man + Woman = Queen
- Proposes an **extension of Skip-gram model**
  - Improvement in both “quality of vector representation” & “training speed”
  - **1) Subsampling of frequent words**
  - **2) Negative Sampling**
- Limitations of word representation : “Inability to represent idiomatic phrases”

2. Skip-gram Model

(1) Brief review of word2vec models

- **CBOW** : predicting the "**current word**", based on the context
- **Skip-gram** : predicting "**words within a certain range**" of the current word, given current word.



Efficient Estimation of Word Representations in Vector Space (T Mikolov et al. , 2013)

2. Skip-gram Model

(2) Training Skip-gram

- How? “Maximize the **average log-probability**”

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

  - c : size of the training context
  - Larger c (= more training examples) $\rightarrow$ higher accuracy, slower speed

2. Skip-gram Model

(2) Training Skip-gram

- How? “Maximize the **average log-probability**”

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

  - c : size of the training context
  - Larger c (= more training examples) $\rightarrow$ higher accuracy, slower speed

- Softmax Function

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

  - v_w : input
  - v'_w : output
  - W : number of words (vocab)

2. Skip-gram Model

(2) Training Skip-gram

- How? “Maximize the **average log-probability**”

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

  - c : size of the training context
  - Larger c (= more training examples) $\rightarrow$ higher accuracy, slower speed

- Softmax Function

$$p\left(w_O \mid w_I\right)=\frac{\exp\left({v'_{w_O}}^{\top} v_{w_I}\right)}{\sum_{w=1}^{W}\exp\left({v'_w}^{\top} v_{w_I}\right)}.$$ $\rightarrow$ Impractical ! Too many words! ($10^5 \sim 10^7$ terms)

  - v_w : input
  - v'_w : output
  - W : number of words (vocab)

2. Skip-gram Model

(2) Training Skip-gram

- How? “Maximize the **average log-probability**”

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

  - c : size of the training context
  - Larger c (= more training examples) $\rightarrow$ higher accuracy, slower speed

- Softmax Function

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

$\rightarrow$ Impractical ! Too many words! ($10^5 \sim 10^7$ terms)

$\nabla \log p\left(w_O \mid w_I\right)$ is proportional to W

  - v_w : input
  - v'_w : output
  - W : number of words (vocab)

2. Skip-gram Model

(3) Problem of standard softmax

$\nabla \log p\left(w_O \mid w_I\right)$ is proportional to W

Proof

Inefficient to use all the W words!

- softmax function : $\boxed{\hat{y}_i = P(i \mid c) = \frac{\exp\left(u_i^T v_c\right)}{\sum_{w=1}^{W} \exp\left(u_w^T v_c\right)}}$

  where u_i and v_j are the column vectors of embedded matrix

  (let $U = \left[u_1, u_2, \ldots, u_k, \ldots u_W\right]$ be a matrix composed of u_k column vectors)

- loss : Cross Entropy Loss : $\boxed{J = -\sum_{i=1}^{W} y_i \log(\hat{y}_i)}$

  where y : one-hot encoded vector & $\hat{y}$: softmax prediction

2. Skip-gram Model

(3) Problem of standard softmax

$\nabla \log p(w_O \mid w_I)$ is proportional to W

Proof

$$
\begin{aligned}
J &= -\sum_{i=1}^{W} y_i \log\left(\frac{\exp\left(u_i^T v_c\right)}{\sum_{w=1}^{W} \exp\left(u_w^T v_c\right)}\right) \\
&= -\sum_{i=1}^{W} y_i \left[u_i^T v_c - \log\left(\sum_{w=1}^{W} \exp\left(u_w^T v_c\right)\right)\right] \\
&= -y_k \left[u_k^T v_c - \log\left(\sum_{w=1}^{W} \exp\left(u_w^T v_c\right)\right)\right]
\end{aligned}
$$

2. Skip-gram Model

(3) Problem of standard softmax

$\nabla \log p(w_O \mid w_I)$ is proportional to W

Proof

$$J = -\sum_{i=1}^{W} y_i \log\left(\frac{\exp(u_i^T v_c)}{\sum_{w=1}^{W} \exp(u_w^T v_c)}\right)$$
$$= -\sum_{i=1}^{W} y_i \left[u_i^T v_c - \log\left(\sum_{w=1}^{W} \exp(u_w^T v_c)\right)\right]$$
$$= -y_k \left[u_k^T v_c - \log\left(\sum_{w=1}^{W} \exp(u_w^T v_c)\right)\right]$$

➡

$$\frac{\partial J}{\partial v_c} = -\left[u_k - \frac{\sum_{w=1}^{W} \exp(u_w^T v_c) u_w}{\sum_{x=1}^{W} \exp(u_x^T v_c)}\right]$$
$$= \sum_{w=1}^{W} \left(\frac{\exp(u_w^T v_c)}{\sum_{x=1}^{W} \exp(u_x^T v_c)} u_w\right) - u_k$$
$$= \sum_{w=1}^{W} (\hat{y}_w u_w) - u_k$$

2. Skip-gram Model

(3) Problem of standard softmax

$\nabla \log p\left(w_O \mid w_I\right)$ is proportional to W

Proof

$$
\begin{aligned}
J &= -\sum_{i=1}^{W} y_i \log\left(\frac{\exp\left(u_i^T v_c\right)}{\sum_{w=1}^{W} \exp\left(u_w^T v_c\right)}\right) \\
&= -\sum_{i=1}^{W} y_i \left[u_i^T v_c - \log\left(\sum_{w=1}^{W} \exp\left(u_w^T v_c\right)\right)\right] \\
&= -y_k \left[u_k^T v_c - \log\left(\sum_{w=1}^{W} \exp\left(u_w^T v_c\right)\right)\right]
\end{aligned}
$$

$\Rightarrow$

$$
\begin{aligned}
\frac{\partial J}{\partial v_c} &= -\left[u_k - \frac{\sum_{w=1}^{W} \exp\left(u_w^T v_c\right) u_w}{\sum_{x=1}^{W} \exp\left(u_x^T v_c\right)}\right] \\
&= \sum_{w=1}^{W} \left(\frac{\exp\left(u_w^T v_c\right)}{\sum_{x=1}^{W} \exp\left(u_x^T v_c\right)} u_w\right) - u_k \\
&= \boxed{\sum_{w=1}^{W} \left(\hat{y}_w u_w\right) - u_k}
\end{aligned}
$$

How to make it more efficient?

2. Skip-gram Model

(3) Problem of standard softmax

$\nabla \log p(w_O \mid w_I)$ is proportional to W

Proof

Efficient ways of updating the model :

1. Hierarchical Softmax

2. Negative Sampling

$$J = -\sum_{i=1}^{W} y_i \log\left(\frac{\exp(u_i^T v_c)}{\sum_{w=1}^{W}\exp(u_w^T v_c)}\right)$$

$$= -\sum_{i=1}^{W} y_i \left[u_i^T v_c - \log\left(\sum_{w=1}^{W}\exp(u_w^T v_c)\right)\right]$$

$$= -y_k \left[u_k^T v_c - \log\left(\sum_{w=1}^{W}\exp(u_w^T v_c)\right)\right]$$

$$\frac{\partial J}{\partial v_c} = -\left[u_k - \frac{\sum_{w=1}^{W}\exp(u_w^T v_c)u_w}{\sum_{x=1}^{W}\exp(u_x^T v_c)}\right]$$

$$= \sum_{w=1}^{W}\left(\frac{\exp(u_w^T v_c)}{\sum_{x=1}^{W}\exp(u_x^T v_c)}u_w\right) - u_k$$

$$= \boxed{\sum_{w=1}^{W}(\hat{y}_w u_w) - u_k}$$

How to make it more efficient?

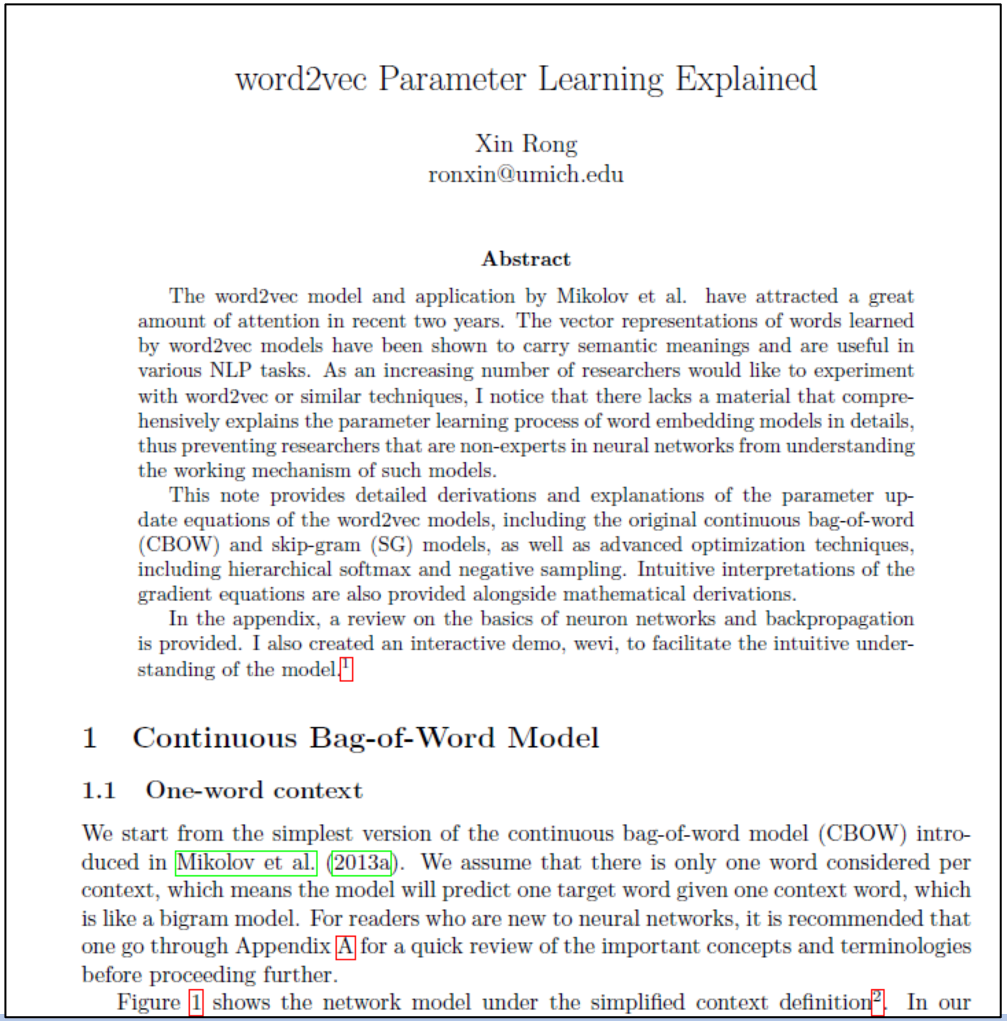
3. Efficient Ways of updating the model

For more about two methods…

Word2vec Parameter Learning Explained

(X. Rong, 2016)

https://arxiv.org/abs/1411.2738



word2vec Parameter Learning Explained

Xin Rong
ronxin@umich.edu

Abstract

The word2vec model and application by Mikolov et al. have attracted a great amount of attention in recent two years. The vector representations of words learned by word2vec models have been shown to carry semantic meanings and are useful in various NLP tasks. As an increasing number of researchers would like to experiment with word2vec or similar techniques, I notice that there lacks a material that comprehensively explains the parameter learning process of word embedding models in details, thus preventing researchers that are non-experts in neural networks from understanding the working mechanism of such models.

This note provides detailed derivations and explanations of the parameter update equations of the word2vec models, including the original continuous bag-of-word (CBOW) and skip-gram (SG) models, as well as advanced optimization techniques, including hierarchical softmax and negative sampling. Intuitive interpretations of the gradient equations are also provided alongside mathematical derivations.

In the appendix, a review on the basics of neuron networks and backpropagation is provided. I also created an interactive demo, wevi, to facilitate the intuitive understanding of the model.[1]

1 Continuous Bag-of-Word Model

1.1 One-word context

We start from the simplest version of the continuous bag-of-word model (CBOW) introduced in Mikolov et al. (2013a). We assume that there is only one word considered per context, which means the model will predict one target word given one context word, which is like a bigram model. For readers who are new to neural networks, it is recommended that one go through Appendix A for a quick review of the important concepts and terminologies before proceeding further.

Figure 1 shows the network model under the simplified context definition[2]. In our

3. Efficient Ways of updating the model

(1) Hierarchical Softmax

Hierarchical probabilistic neural network language model (F. Morin and Y.Bengio, 2005)

- uses binary tree representation
- instead of evaluating W output nodes, only need to evaluate $\log_2(W)$ nodes
- define a random walk that assigns probabilities to words

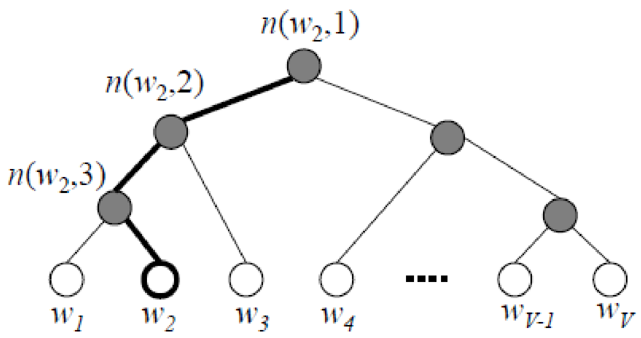
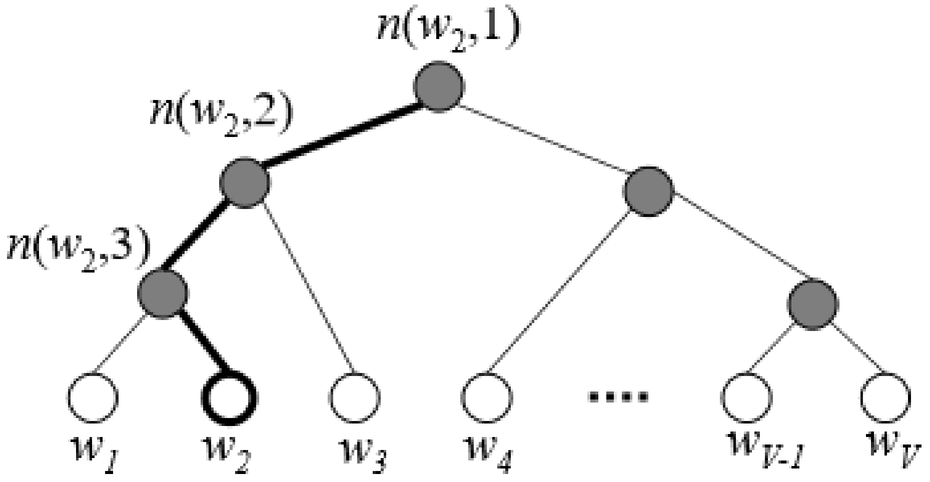


Figure 4: An example binary tree for the hierarchical softmax model. The white units are words in the vocabulary, and the dark units are inner units. An example path from root to w_2 is highlighted. In the example shown, the length of the path $L(w_2) = 4$. $n(w, j)$ means the j-th unit on the path from root to the word w.

3. Efficient Ways of updating the model

(1) Hierarchical Softmax

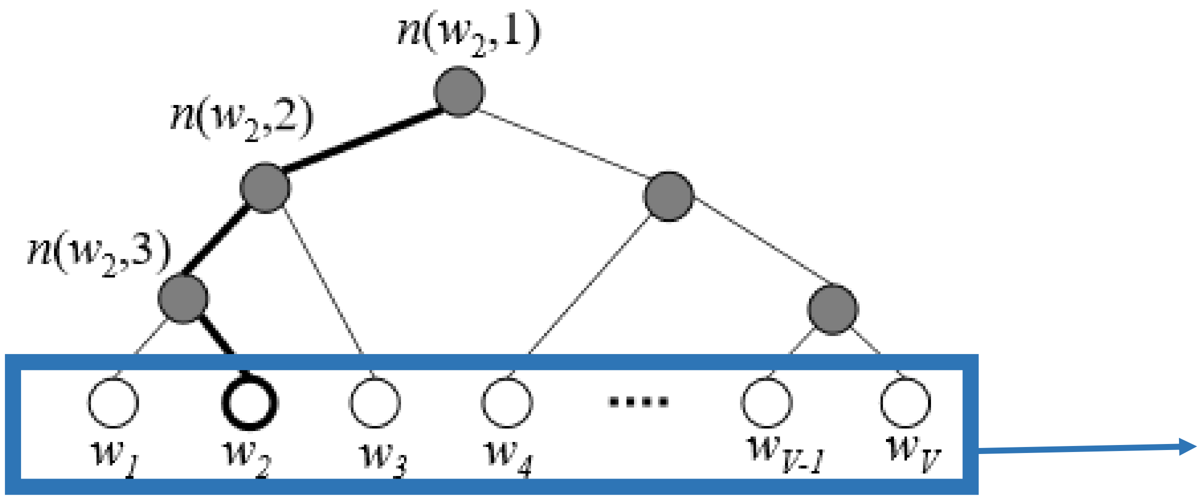
Binary Tree Structure & Notation



3. Efficient Ways of updating the model

(1) Hierarchical Softmax

Binary Tree Structure & Notation



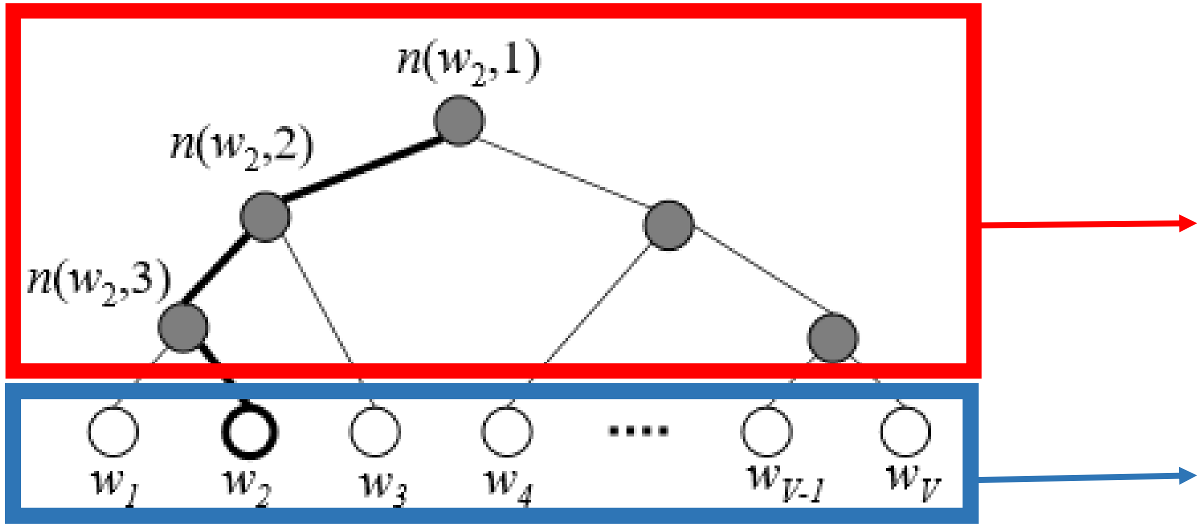
Terminal node (total : V-1)

(each represents one vocabulary)

3. Efficient Ways of updating the model

(1) Hierarchical Softmax

Binary Tree Structure & Notation



Inner Unit (total : V-1)

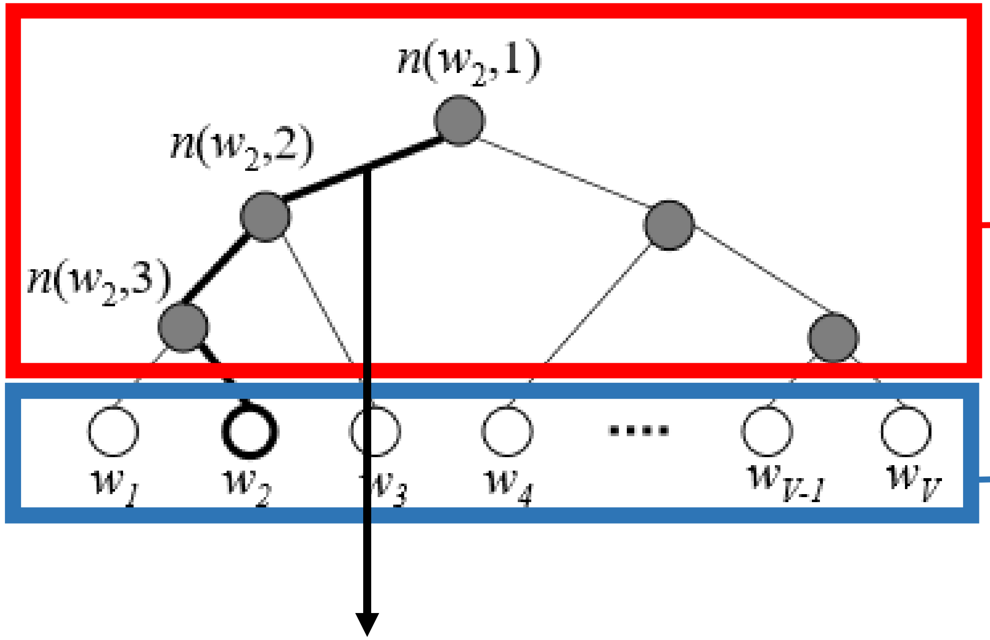
Terminal node (total : V-1)

(each represents one vocabulary)

3. Efficient Ways of updating the model

(1) Hierarchical Softmax

<u>**Binary Tree Structure & Notation**</u>



Inner Unit (total : V-1)

Terminal node (total : V-1)

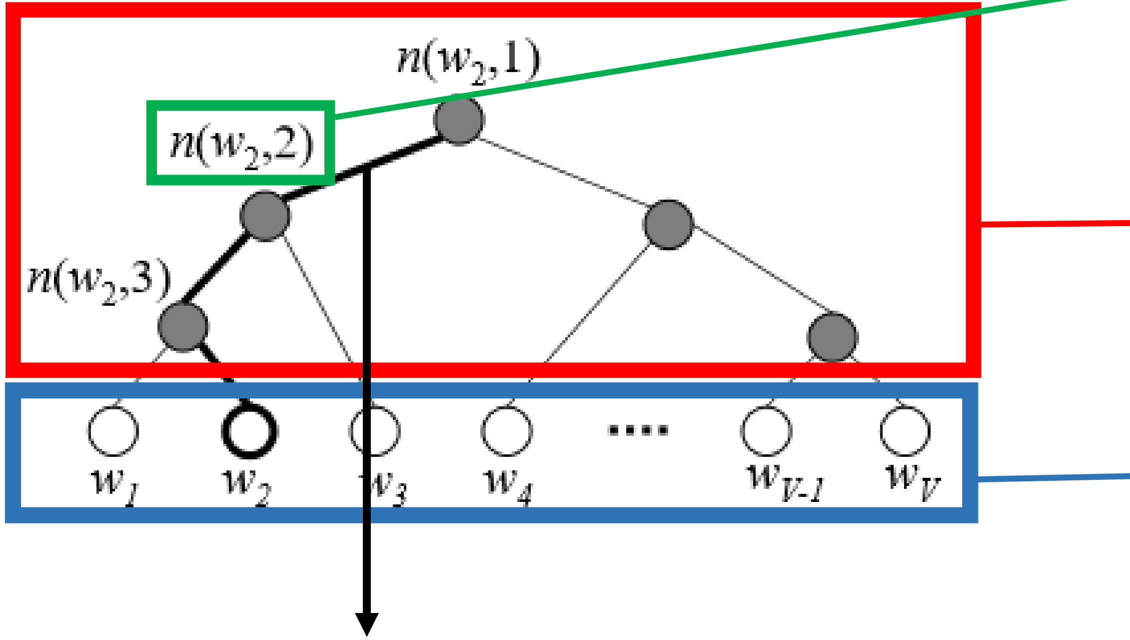
(each represents one vocabulary)

$L(w_2) = 4$: length of "root unit" to w_2

3. Efficient Ways of updating the model

(1) Hierarchical Softmax

Binary Tree Structure & Notation



$n(w, j)$: j^{th} unit from "root unit" to word "w"

Inner Unit (total : V-1)

Terminal node (total : V-1)

(each represents one vocabulary)

$L(w_2) = 4$: length of "root unit" to w_2

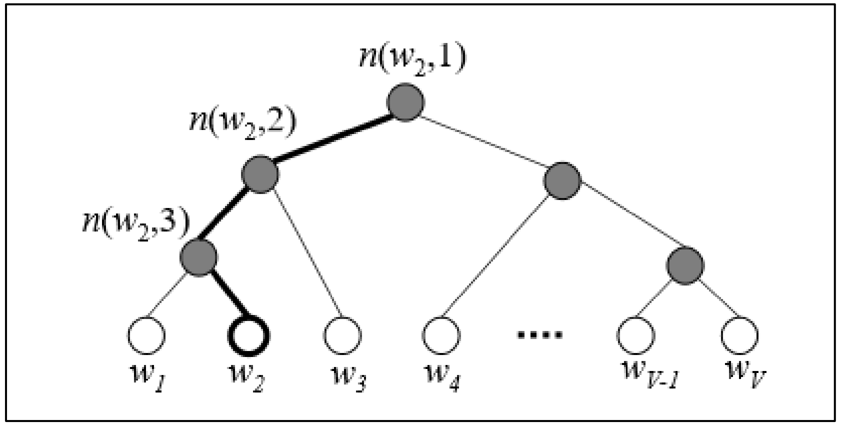
3. Efficient Ways of updating the model

(1) Hierarchical Softmax

Hierarchical Softmax (vs Standard Softmax)

- NO vector representation of each word (= terminal node)

  (Instead, **every inner node has its representation**)



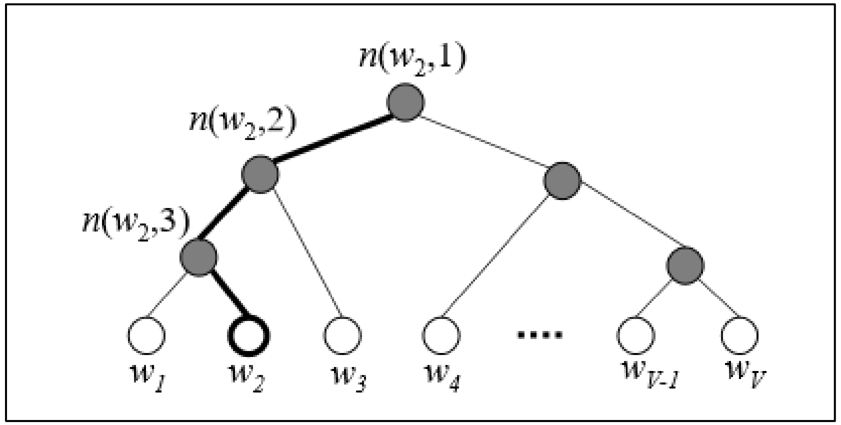
3. Efficient Ways of updating the model

(1) Hierarchical Softmax

Hierarchical Softmax (vs Standard Softmax)

- NO vector representation of each word (= terminal node)

  (Instead, **every inner node has its representation**)



- Probability of output word, becoming w_O :

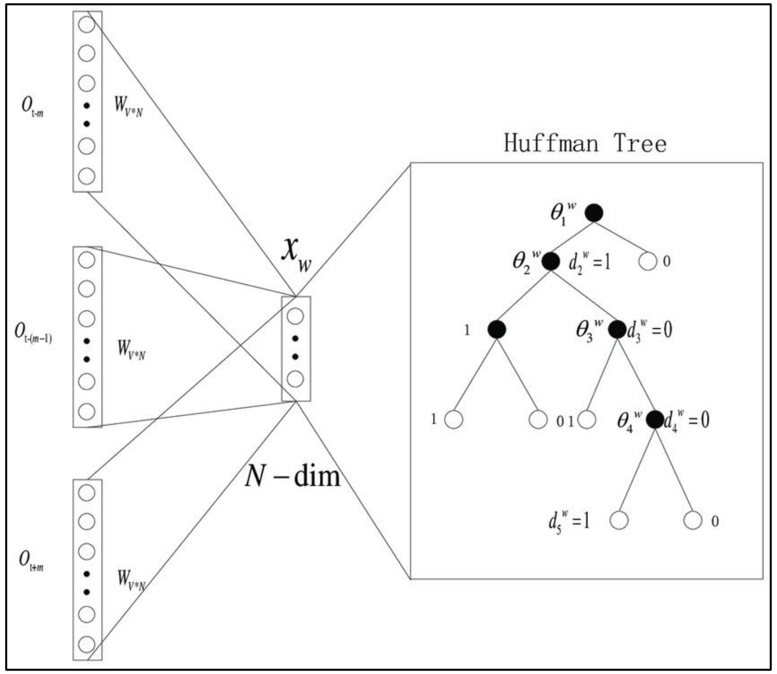
$$p\left(w = w_O\right) = \prod_{j=1}^{L(w)-1} \sigma\left(\ll n(w, j+1) = \mathrm{ch}(n(w,j)) \gg \cdot \mathbf{v}'_{n(w,j)} \mathbf{h}\right)$$

(1) Hierarchical Softmax

- Probability of output word, becoming w_O :

$$p\left(w=w_O\right)=\prod_{j=1}^{L(w)-1} \sigma\left(\ll n(w, j+1)=\operatorname{ch}(n(w, j)) \gg \cdot \mathbf{v}_{n(w, j)}^{\prime}{ }^T \mathbf{h}\right)$$

- $\operatorname{ch}(n)$: "left" child node of unit n
- $\mathbf{v}'_{n(w,j)}$: vector representation of inner unit $n(w,j)$
- $\mathbf{h}$: output vector of hidden layer
  - Skip-gram) $\mathbf{h}=\mathbf{v}_{w_I}$
  - CBOW) $\mathbf{h}=\frac{1}{C}\sum_{c=1}^{C}\mathbf{v}_{w_c}$
- $\ll x \gg = \begin{cases} 1 & \text{if } x \text{ is true} \\ -1 & \text{otherwise} \end{cases}.$



https://www.researchgate.net/figure/Figure1-Hierarchical-Softmax-CBOW-Model-Schematic-The-word-vector-matrix_fig1_329395807

3. Efficient Ways of updating the model

(2) Negative Sampling

- alternative to hierarchical softmax : NCE (Noise Contrastive Estimation)
    - good model = able to differentiate data from noise, using logistic regression
    - simplified NCE = **Negative Sampling**
- Negative sampling : use k negative samples (= wrong answers)

3. Efficient Ways of updating the model

(2) Negative Sampling

- alternative to hierarchical softmax : NCE (Noise Contrastive Estimation)
  - good model = able to differentiate data from noise, using logistic regression
  - simplified NCE = **Negative Sampling**
- Negative sampling : use k negative samples (= wrong answers)

Context Word	Word	Is it a Target word?
Grape	Juice	1
..	King	0
..	Earphone	0
..	Kindergarten	0
..	Pencil	0

→ Positive sample

Negative Samples (K=4)

(small data) K=5~20

(large data) K=2~5

3. Efficient Ways of updating the model

(2) Negative Sampling

- (Standard) Objective function

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

Maximize…

$$\log p(w_O \mid w_I) = \underline{\left({v'_{w_O}}^\top v_{w_I}\right) - \log \sum_{w=1}^{W} \exp\left({v'_w}^\top v_{w_I}\right)}$$

3. Efficient Ways of updating the model

(2) Negative Sampling

- (Standard) Objective function

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

Maximize…

$$\log p\left(w_O \mid w_I\right)=\underline{\left({v'_{w_O}}^{\top} v_{w_I}\right)-\log \sum_{w=1}^{W} \exp\left({v'_w}^{\top} v_{w_I}\right)}$$

- (Proposed) instead of $\log p\left(w_O \mid w_I\right)$…

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

3. Efficient Ways of updating the model

(2) Negative Sampling

- (Standard) Objective function

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

Maximize…

$$\log p(w_O \mid w_I) = \underline{\left({v'_{w_O}}^\top v_{w_I}\right) - \log \sum_{w=1}^{W} \exp\left({v'_w}^\top v_{w_I}\right)}$$

- (Proposed) instead of $\log p(w_O \mid w_I)$…

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

Distinguish "target word"(=positive) from "draws from the noise distribution"(=negative), using logistic regression

3. Efficient Ways of updating the model

(2) Negative Sampling

Updating Equation (1) hidden-output

$$\frac{\partial E}{\partial \mathbf{v}'_{w_j}{}^T \mathbf{h}} = \begin{cases} \sigma\left(\mathbf{v}'_{w_j}{}^T \mathbf{h}\right) - 1 & \text{if } w_j = w_O \\ \sigma\left(\mathbf{v}'_{w_j}{}^T \mathbf{h}\right) & \text{if } w_j \in \mathcal{W}_{\text{neg}} \end{cases}$$

$$= \sigma\left(\mathbf{v}'_{w_j}{}^T \mathbf{h}\right) - \boxed{t_j}$$ (label : 1 if positive, 0 if negative)

$$\frac{\partial E}{\partial \mathbf{v}'_{w_j}} = \frac{\partial E}{\partial \mathbf{v}'_{w_j}{}^T \mathbf{h}} \cdot \frac{\partial \mathbf{v}'_{w_j}{}^T \mathbf{h}}{\partial \mathbf{v}'_{w_j}} = \left(\sigma\left(\mathbf{v}'_{w_j}{}^T \mathbf{h}\right) - t_j\right) \mathbf{h}$$

$$\mathbf{v}'_{w_j}(\text{ new }) = \mathbf{v}'_{w_j}(\text{ old }) - \eta\left(\sigma\left(\mathbf{v}'^T_{w_j} \mathbf{h}\right) - t_j\right) \mathbf{h}$$

3. Efficient Ways of updating the model

(2) Negative Sampling

Updating Equation (1) hidden-output

$$\frac{\partial E}{\partial \mathbf{v}'_{w_j}{}^T \mathbf{h}} = \begin{cases} \sigma\left(\mathbf{v}'_{w_j}{}^T \mathbf{h}\right) - 1 & \text{if } w_j = w_O \\ \sigma\left(\mathbf{v}'_{w_j}{}^T \mathbf{h}\right) & \text{if } w_j \in \mathcal{W}_{\text{neg}} \end{cases}$$
$$= \sigma\left(\mathbf{v}'_{w_j}{}^T \mathbf{h}\right) - \boxed{t_j}$$ (label : 1 if positive, 0 if negative)

$$\frac{\partial E}{\partial \mathbf{v}'_{w_j}} = \frac{\partial E}{\partial \mathbf{v}'_{w_j}{}^T \mathbf{h}} \cdot \frac{\partial \mathbf{v}'_{w_j}{}^T \mathbf{h}}{\partial \mathbf{v}'_{w_j}} = \left(\sigma\left(\mathbf{v}'_{w_j}{}^T \mathbf{h}\right) - t_j\right) \mathbf{h}$$

$$\mathbf{v}'_{w_j}(\text{ new }) = \mathbf{v}'_{w_j}(\text{ old }) - \eta\left(\sigma\left(\mathbf{v}'^T_{w_j} \mathbf{h}\right) - t_j\right) \mathbf{h}$$

Computationally Efficient!

No need to be applied to all words!

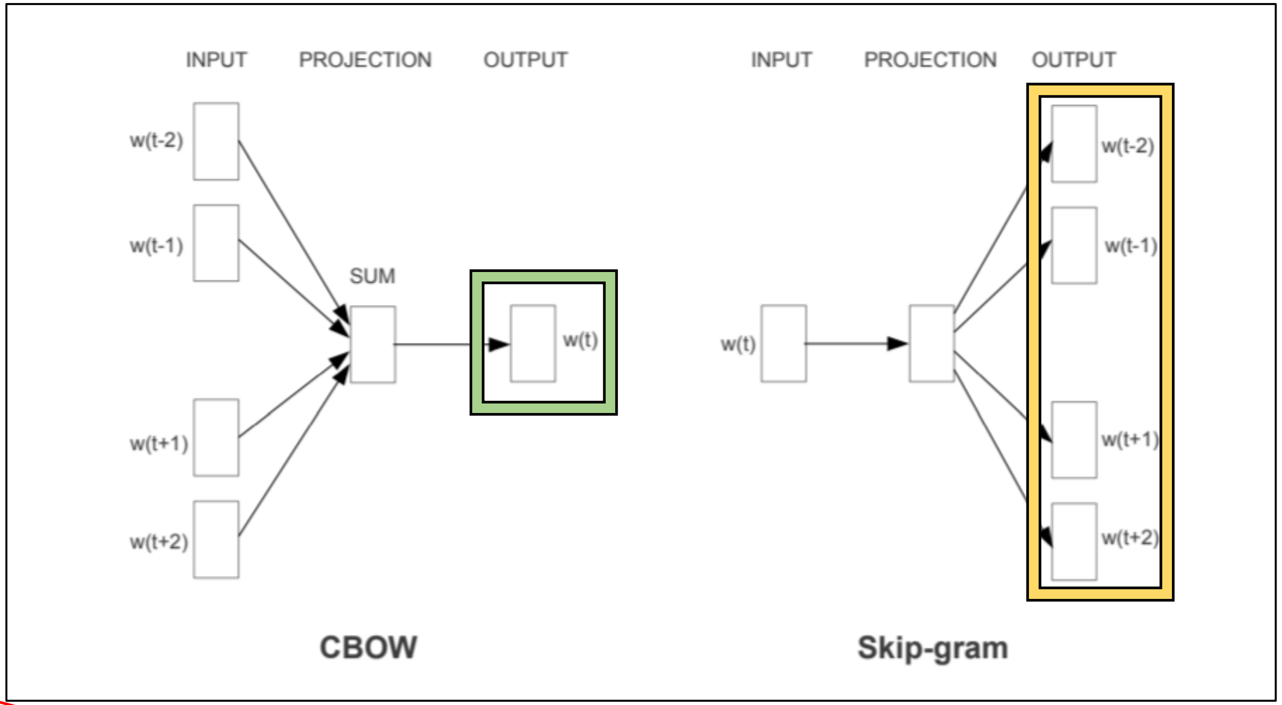
Only K+1 words (1 pos + K neg)

3. Efficient Ways of updating the model

(2) Negative Sampling

Updating Equation (2) input-hidden

$$
\begin{aligned}
\frac{\partial E}{\partial \mathbf{h}} &= \sum_{w_j \in \{w_O\} \cup \mathcal{W}_{\text{neg}}} \frac{\partial E}{\partial \mathbf{v}'^{\,T}_{w_j} \mathbf{h}} \cdot \frac{\partial \mathbf{v}'^{\,T}_{w_j} \mathbf{h}}{\partial \mathbf{h}} \\
&= \sum_{w_j \in \{w_O\} \cup \mathcal{W}_{\text{neg}}} \left(\sigma(\mathbf{v}'^{\,T}_{w_j} \mathbf{h}) - t_j \right) \mathbf{v}'_{w_j} := \mathrm{EH}
\end{aligned}
$$



- **CBOW** : just use EH above
- **Skip-gram** : calculate all the EHs of context words

$$
\mathbf{v}^{(\text{new})}_{w_{I,c}} = \mathbf{v}^{(\text{old})}_{w_{I,c}} - \frac{1}{C} \cdot \eta \cdot \mathrm{EH}^T \qquad \text{for } c = 1, 2, \cdots, C.
$$

4. Subsampling method

- Most frequent words : occur hundreds of millions of times
- Assign different probability of getting sampled for each word

4. Subsampling method

- Most frequent words : occur hundreds of millions of times
- Assign different probability of getting sampled for each word

- **More Frequently** occurred, **Less Probability** of getting sampled

(Probability of getting dropped)

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

where $f(w_i)$: frequency of word i & t : chosen threshold

5. Experiments

"Analogical Reasoning" Task

- Syntactic analogies (ex. quick : quickly = slow : slowly)
- Semantic analogies (ex. Korea : Seoul = Japan : Tokyo)

Method	Time [min]	Syntactic [%]	Semantic [%]	Total accuracy [%]
NEG-5	38	63	54	59
NEG-15	97	63	58	**61**
HS-Huffman	41	53	40	47
NCE-5	38	60	45	53
The following results use 10^{-5} subsampling				
NEG-5	14	61	58	60
NEG-15	36	61	61	**61**
HS-Huffman	21	52	59	55

Table 1: Accuracy of various Skip-gram 300-dimensional models on the analogical reasoning task as defined in [8]. NEG-k stands for Negative Sampling with k negative samples for each positive sample; NCE stands for Noise Contrastive Estimation and HS-Huffman stands for the Hierarchical Softmax with the frequency-based Huffman codes.

6. Learning Phrases

Replacing with a unique phrase

- Lots of words are not a simple composition of the meaning of individual words

  (Apple Pencil != Apple Pencil)
- Instead of treating "**Apple**" and "**Pencil**" apart, treat it as unique token! "**Apple Pencil**"

6. Learning Phrases

Replacing with a unique phrase

- Lots of words are not a simple composition of the meaning of individual words

  (Apple Pencil != Apple Pencil)

- Instead of treating “**Apple**” and “**Pencil**” apart, treat it as unique token! “**Apple Pencil**”

Algorithm

1) find words that **appear frequently together** (& **infrequently in other** contexts)

   (whose score below is above certain threshold)

$$\boxed{\text{score}(w_i, w_j) = \frac{\text{count}(w_i w_j) - \delta}{\text{count}(w_i) \times \text{count}(w_j)}}$$

Discounting Coefficient

2) Replace such words with **unique token (one phrase)**

6. Learning Phrases

Replacing with a unique phrase

- Lots of words are not a simple composition of the meaning of individual words

(Apple Pencil != Apple Pencil)

- Instead of treating "**Apple**" and "**Pencil**" apart, treat it as unique token! "**Apple Pencil**"

Algorithm

1) find words that **appear frequently together** (& **infrequently in other** contexts)

(whose score below is above certain threshold)

$$\boxed{\mathrm{score}(w_i, w_j) = \frac{\mathrm{count}(w_i w_j) - \delta}{\mathrm{count}(w_i) \times \mathrm{count}(w_j)}}$$

Discounting Coefficient

2) Replace such words with **unique token (one phrase)**

6. Learning Phrases

Replacing with a unique phrase

- Lots of words are not a simple composition of the meaning of individual words

  (Apple Pencil != Apple Pencil)

- Instead of treating "**Apple**" and "**Pencil**" apart, treat it as unique token! "**Apple Pencil**"

Algorithm

1) find words that **appear frequently together** (& **infrequently in other** contexts)

   (whose score below is above certain threshold)

$$\boxed{\text{score}(w_i, w_j) = \frac{\text{count}(w_i w_j) - \delta}{\text{count}(w_i) \times \text{count}(w_j)}}$$

Discounting Coefficient

2) Replace such words with **unique token (one phrase)**

Repeat 2x ~ 4x

(+ decreasing threshold)

6. Learning Phrases

Experiment

Best Result : Hierarchical Softmax with Subsampling

Method	Dimensionality	No subsampling [%]	10^{-5} subsampling [%]
NEG-5	300	24	27
NEG-15	300	27	42
HS-Huffman	300	19	**47**

Table 3: Accuracies of the Skip-gram models on the phrase analogy dataset. The models were trained on approximately one billion words from the news dataset.

7. Additive Compositionality

Additive Compositionality

Possible to meaningfully combine words by an element-wise addition

Czech + currency	Vietnam + capital	German + airlines	Russian + river	French + actress
koruna	Hanoi	airline Lufthansa	Moscow	Juliette Binoche
Check crown	Ho Chi Minh City	carrier Lufthansa	Volga River	Vanessa Paradis
Polish zolty	Viet Nam	flag carrier Lufthansa	upriver	Charlotte Gainsbourg
CTK	Vietnamese	Lufthansa	Russia	Cecile De

Table 5: Vector compositionality using element-wise addition. Four closest tokens to the sum of two vectors are shown, using the best Skip-gram model.

Summary

(1) Extension of Skip-gram Model (can also be applied to CBOW)

(2) Improvement in quality & speed using…

- 1) **Negative Sampling**
- 2) **Hierarchical Softmax**
- 3) **Subsampling**

(3) Represent phrases with **single token**

Reference

- **Distributed Representations of Words and Phrases and their Compositionality**

  (T. Mikolov et al., 2013)

  https://arxiv.org/abs/1310.4546

- **Word2vec Parameter Learning Explained**

  (X. Rong, 2016)

  https://arxiv.org/abs/1411.2738

Thank You!

(+ Any Questions?)