YBIGTA NLP Study

Distributed Representation and Compositionality of Words and Phrases

발제자
윤민주

목차
Contents

1. 논문 소개
   논문 개요 / 주요 개념
2. 기본 방식과 튜닝
   Skip-gram / Phrase-tokenizing
3. 성능 비교
   HS / NCE / NEG / Subsampling / Phrase

1. 논문 소개

1-1. 논문의 개요

- 개요
  - 초기 Word2Vec에 튜닝 기법이 추가된 버전
- 목적
  - Skip-gram 모델의 Vector representation quality 향상
- 방법
  - Negative sampling
  - Subsampling frequent words
  - Phrase learning

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

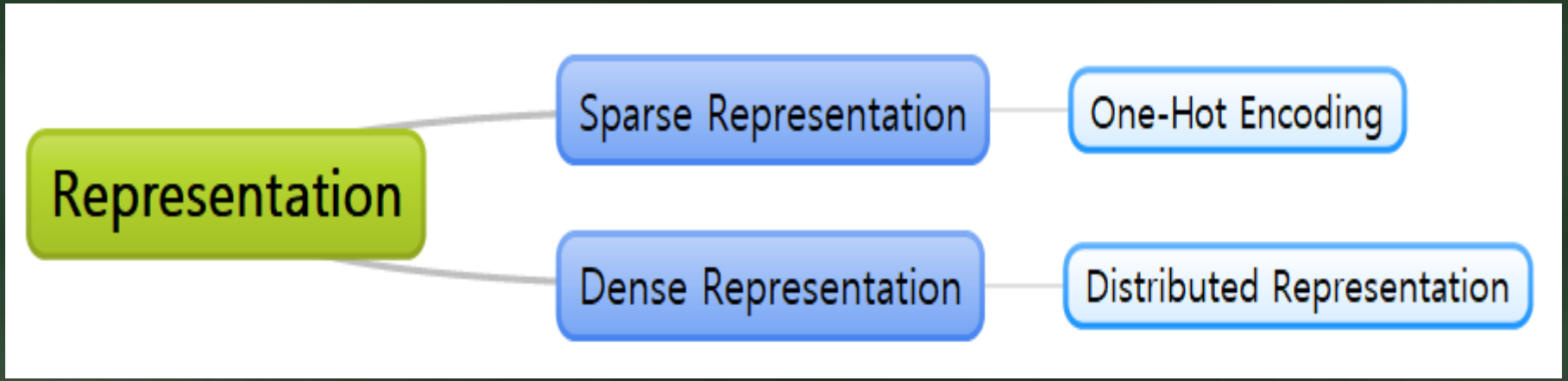
Distributed representations of words in a vector space help learning algorithms to achieve better performance in natural language processing tasks by grouping similar words. One of the earliest use of word representations dates back to 1986 due to Rumelhart, Hinton, and Williams [13]. This idea has since been applied to statistical language modeling with considerable success [1]. The follow up work includes applications to automatic speech recognition and machine translation [14, 7], and a wide range of NLP tasks [2, 20, 15, 3, 18, 19, 9].

Recently, Mikolov et al. [8] introduced the Skip-gram model, an efficient method for learning high-quality vector representations of words from large amounts of unstructured text data. Unlike most of the previously used neural network architectures for learning word vectors, training of the Skip-gram model (see Figure 1) does not involve dense matrix multiplications. This makes the training extremely efficient: an optimized single-machine implementation can train on more than 100 billion words in one day.

The word representations computed using neural networks are very interesting because the learned vectors explicitly encode many linguistic regularities and patterns. Somewhat surprisingly, many of these patterns can be represented as linear translations. For example, the result of a vector calculation vec("Madrid") - vec("Spain") + vec("France") is closer to vec("Paris") than to any other word vector [9, 8].

1-2. Representation

- 대상의 속성을 표현하는 방법
- Vector density에 따라 2가지로 구분

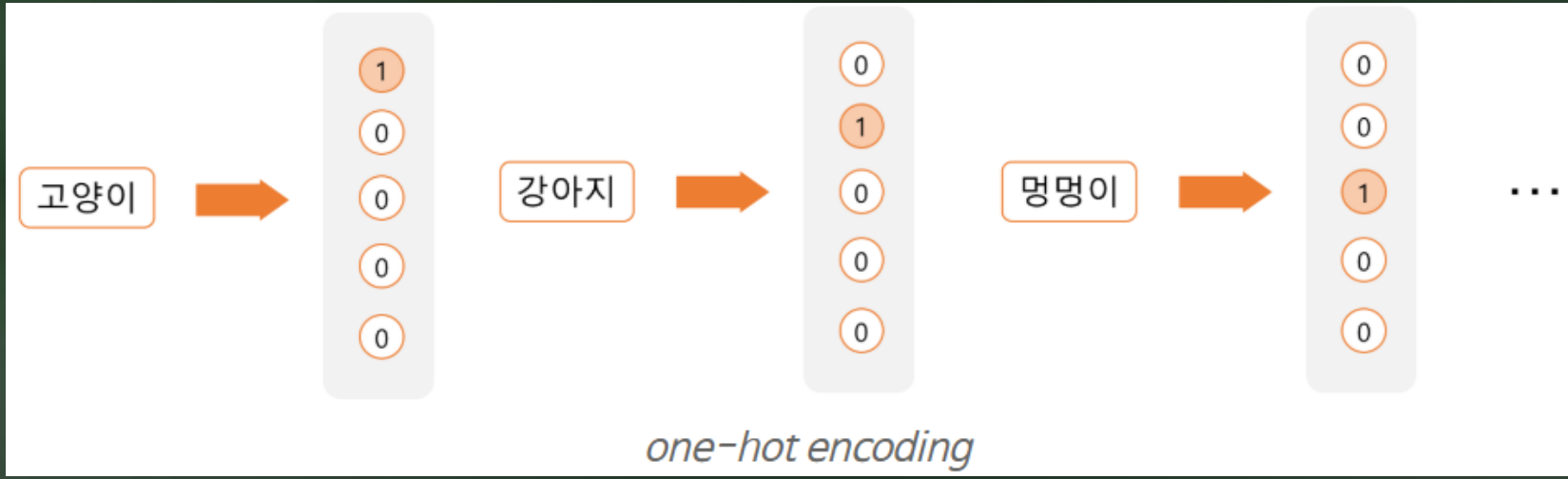


1-2. Representation

- Sparse Representation
  - 대부분의 성분값 = 0
  - 가장 간단한 방식이며, 전통적으로 자주 사용
  - N개의 단어 → N-Dimensional Vector Space
    → Sparse Representation

1-2. Representation

- One-hot encoding
  - 해당 index에는 1, 나머지 index에는 0을 부여
  - N개의 단어를 표현하기 위해서는 N차원 이상이 필요
  - 단어와 단어 간의 관계가 나타나지 않음



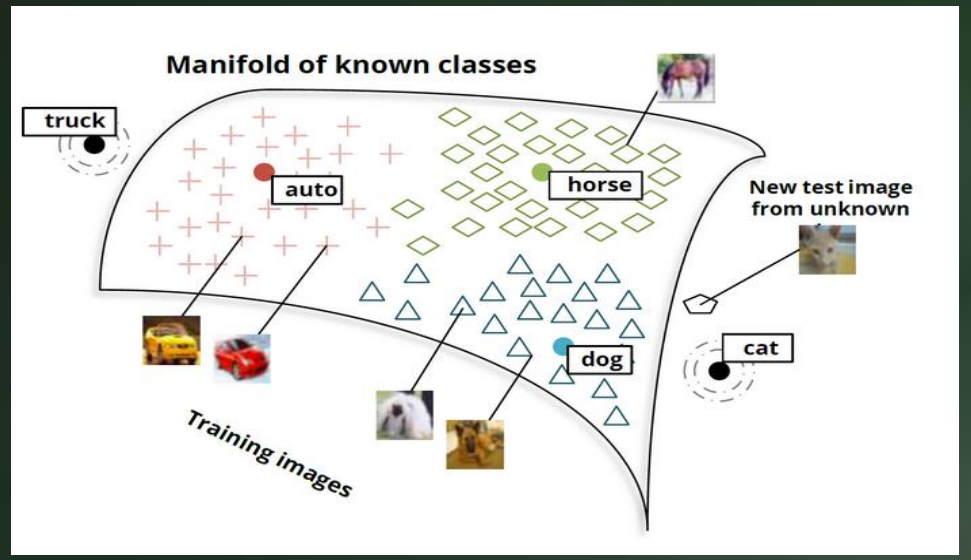
one-hot encoding

1-2. Representation

- Dense Representation
  - 벡터의 차원을 지정
  - 지정한 개수에 맞추어 대상의 속성을 표현
  - 모든 차원이 값을 가짐(Dense)

1-2. Representation

- Distributed Representation
  - 하나의 정보(대상)가 여러 차원에 분산되어 표현
  - Distributional hypothesis
    - 비슷한 위치(문맥)에 등장하는 단어쌍의 의미는 서로 유사할 것
    - "The meaning of a word is its use in the language"
      - L. Wittgenstein –



1-3. Compositionality

- Combine words meaningfully by an element-wise addition of their vector representations.
  - Linear structure
  - ex) vec("한국") + vec("항공사") = vec("대한항공")

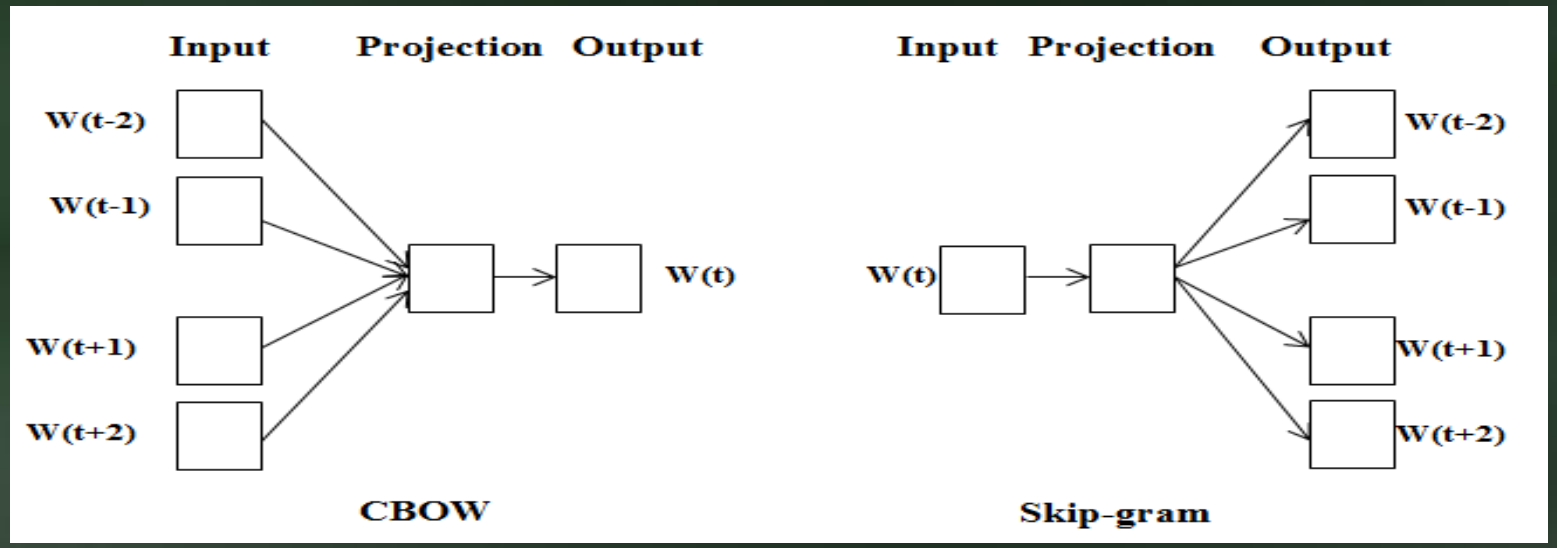
- Skip-gram 모델에 의한 학습 결과

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

2. 기본 방식과 튜닝

2-1. Skip-gram

- Word2Vec
  - CBOW: 주변단어 → 중심단어
  - Skip-gram: 중심단어 → 주변단어



2-1. Skip-gram

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

  * $w_{t-c}, \cdots, w_{t-1}, w_{t+1}, \cdots w_{t+c}$

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

  - v는 w의 벡터 표현이며, v'은 output을 의미

2-1. Skip-gram

- $$p(w_O|w_I) = \frac{\exp\left({v'_{w_O}}^{\top} v_{w_I}\right)}{\sum_{w=1}^{W} \exp\left({v'_w}^{\top} v_{w_I}\right)}$$: 총 W개의 항
- $$\frac{1}{T}\sum_{t=1}^{T} \sum_{-c \le j \le c, j \neq 0} \log p(w_{t+j}|w_t)$$

- Cost of computing = $$\nabla \log p(w_O|w_I) \propto W$$
  - W의 개수만큼 연산을 하게 되어 비효율적

2-1. Skip-gram

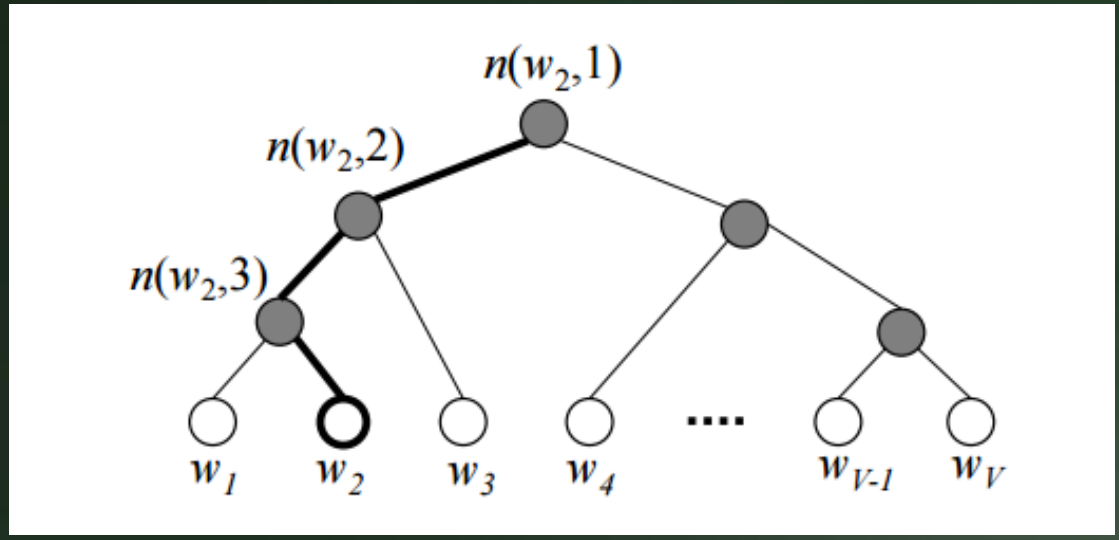
- Cost of computing 문제
  - Hierarchical Softmax
  - Negative Sampling
  - Subsampling of the frequent words

2-2. Hierarchical Softmax

- 모델 구조를 Full binary tree로 변경
  - 해당되는 node의 weight을 곱한다.
  - 합의 법칙 → 곱의 법칙

- 모든 word 고려x
  → cost of computation 감소
  - 평균적으로 $\log_2(W)$에 비례

- 빈번하지 않은 단어에 적합



2-2. Hierarchical Softmax

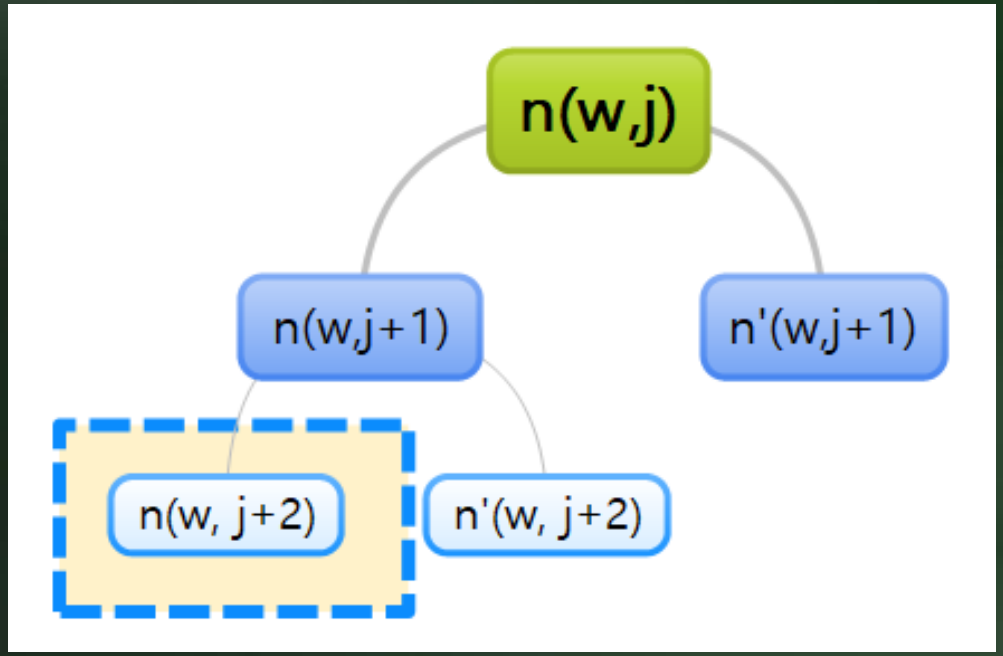
- $$p(w = w_O) = \prod_{j=1}^{L(w)-1} \sigma\left([\![n(w, j+1) = \mathrm{ch}(n(w, j))]\!] \cdot {\mathbf{v}'_{n(w,j)}}^T \mathbf{h}\right)$$

- L(w): root에서 w라는 leaf에 가는 path의 길이
- n(w,i): w까지 가는 path에서 만나는 i번째 노드
- ch(node): node의 임의이 한 자식(child)
- [[x]]: if x == True: [[x]] = 1, else: [[x]] = -1
- v': node를 지날 때 곱하는 weight vector
- h: hidden layer의 값 벡터
- $\sigma(x) = {}^{1}/_{(1+\mathrm{e}^{-x})}$: sigmoid function

2-2. Hierarchical Softmax

- [[n(w, j+1) = ch(n(w, j)]]
  - 이전 노드에서 좌측으로 이동: 1
    우측으로 이동: -1

- Sigmoid 함수에 의해
  복잡한 summation 없이 sum=1
  - $\sigma(v_n^T v_{w_i}) + \sigma(-v_n^T v_{w_i}) = 1$

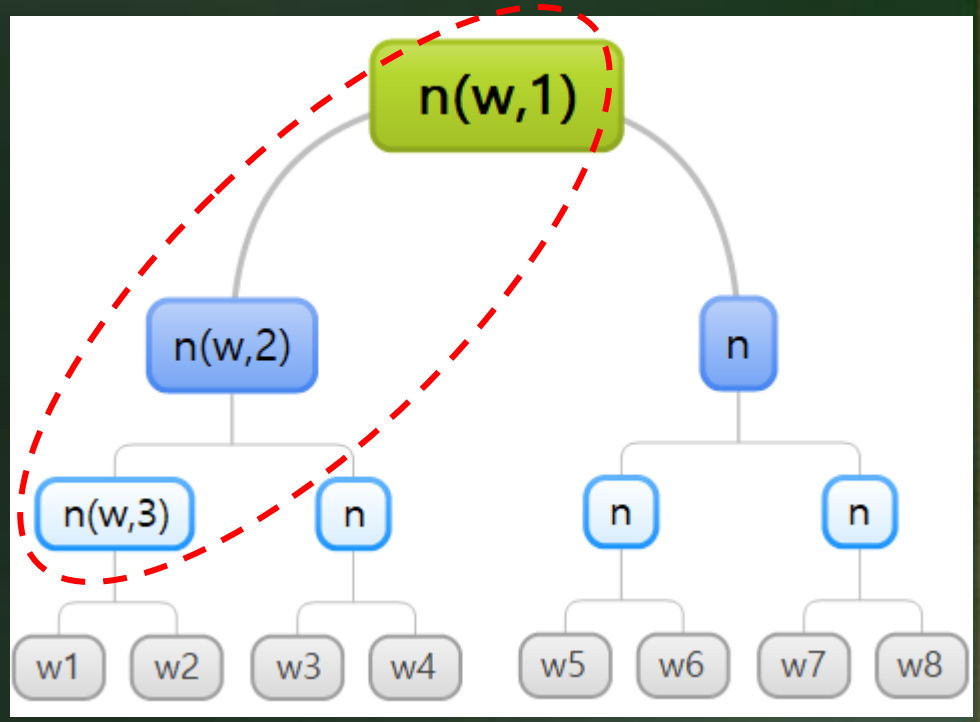


* n′: 지나지 않은 노드

2-2. Hierarchical Softmax

- 계산량 = N x $\log_2(W)$
  - N: projection Layer의 크기
  - W: 단어의 개수

- Ex) 8개 단어 set에서는...
  - W = 8
  - 지나는 node = 3
  - $\log_2(W) = \log_2(8) = 3$



2-3. Noise-Contrastive Estimation

- Word2Vec에서 사용하는 비용(cost) 계산 알고리즘
- Binary logistic regression probability 모델
  - Logistic regression → data와 noise를 이진 분류
- noise distribution
  → k개의 contrastive(대조되는) words를 sampling
  → 기대값을 추정(Estimation)

2-4. Negative Sampling

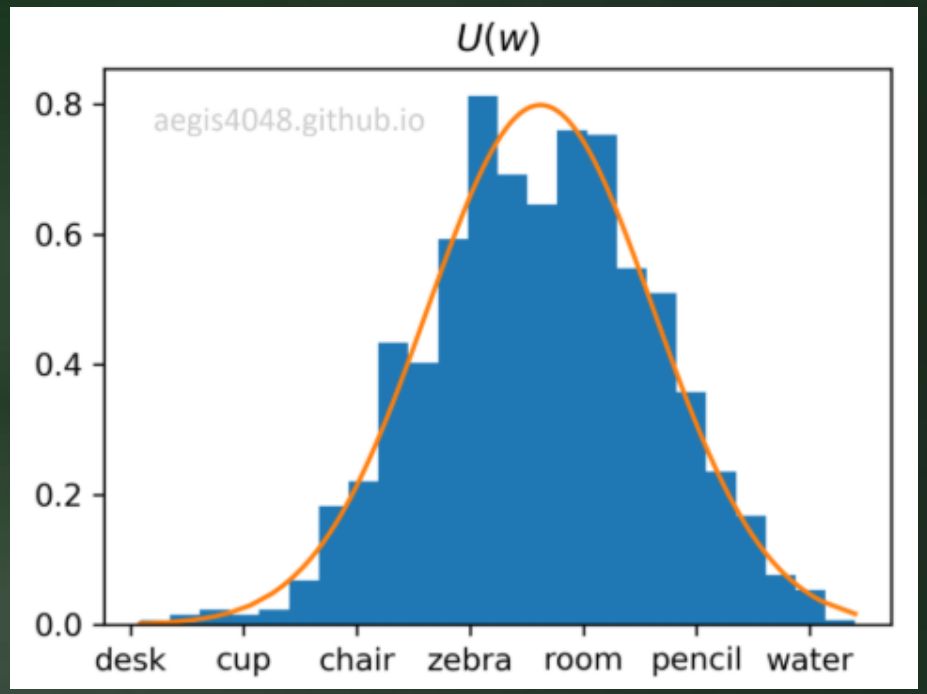
- NCE를 단순화한 버전

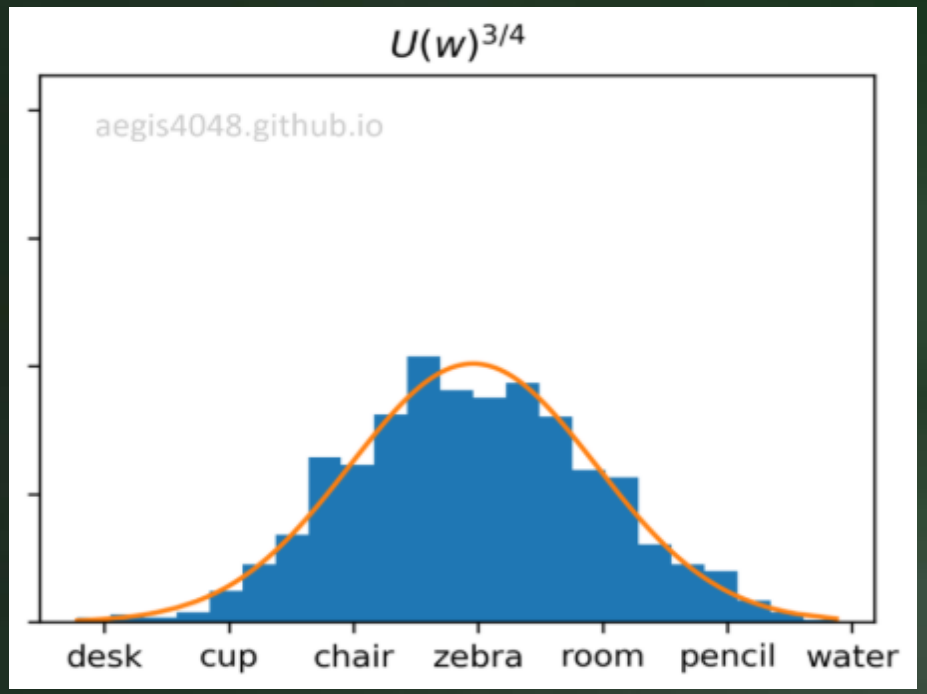
- (word, context) 쌍이 positive인지, negative인지 분류
  - positive sample: (w, c) 쌍이 corpus에 포함되는 경우
  - negative sample: (w, c) 쌍이 corpus에 포함되지 않는 경우

- Noise distribution $P_n(w) = \left(\frac{U(w)}{Z}\right)^{\alpha}$ (unigram)
  - normalized frequency distribution of words raised to the power of α
  - z: normalization factor
  - α: hyper-parameter (typically ¾)

2-4. Negative Sampling

- α=3/4인 경우에 더 smooth한 distribution이 나타남





2-4. Negative Sampling

- 목적 함수의 최대화

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

Positive sample

Negative sample

  - Noise words에 낮은 확률을 할당할 때 목적 함수가 최대가 된다.
  - $\mathbb{E}_{w_i \sim P_n(w)}$ 의 의미
    - w_i comes from the noise distribution $P_n(w)$
    - randomly draws

2-5. Subsampling

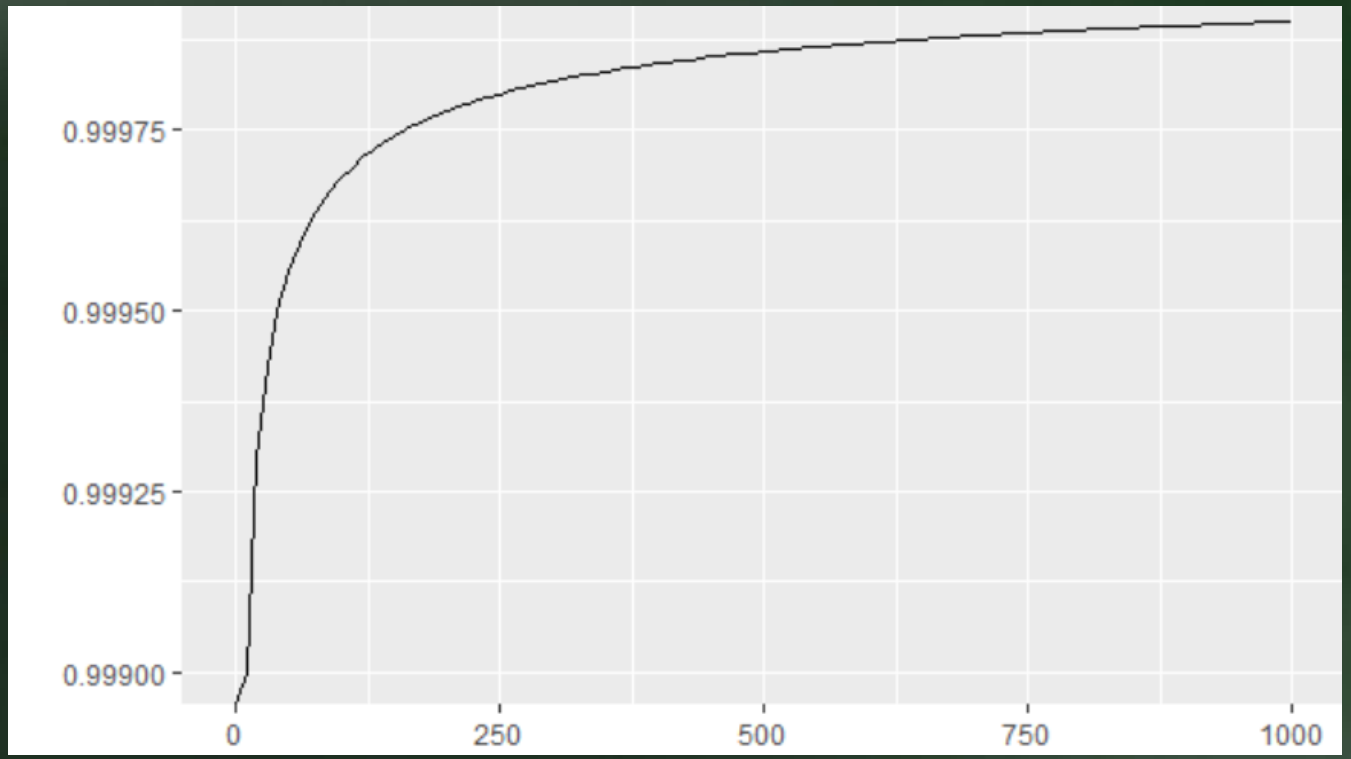
- frequent words
  - in, a, the와 같은 단어로 큰 의미가 없음

- frequent words를 일정 확률로 제외
  - 벡터 값이 거의 바뀌지 않고 학습 속도가 증가
  - $$P(w_i) = 1 - \sqrt{\frac{t}{f(w_i)}}$$
    - $f(w_i)$: 단어의 빈도
    - t: threshold (일반적으로 10^{-5})

2-5. Subsampling

- threshold = 10^{-5} 일 때,

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



2-6. Learning Phrases

- 관용구의 의미 $\neq \Sigma$ 개별 단어의 의미
  - New + York + Times = New York Times?
  - 롯데 + 자이언츠 = 롯데 자이언츠?
- Phrase를 독립된 token으로 취급
- $$\text{score}(w_i, w_j) = \frac{\text{count}(w_i w_j) - \delta}{\text{count}(w_i) \times \text{count}(w_j)}.$$
  - δ: prevents too many phrases
- 시행을 통해 threshold 값을 점차 감소

3. 성능 비교

3-1. Accuracy

- NEG는 k값이 증가할수록 정확도(성능)가 증가
- Subsampling한 HS가 가장 성능이 우수

  → Subsampling can result in faster training and can also improve accuracy, at least in some cases

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

3-2. Phrase learning

- infrequent phrases의 모델별 학습 비교
  - HS + subsampling이 가장 학습이 잘 됨

	NEG-15 with 10^{-5} subsampling	HS with 10^{-5} subsampling
Vasco de Gama	Lingsugur	Italian explorer
Lake Baikal	Great Rift Valley	Aral Sea
Alan Bean	Rebbeca Naomi	moonwalker
Ionian Sea	Ruegen	Ionian Islands
chess master	chess grandmaster	Garry Kasparov

- Data set과 accuracy
  - 6B words → accuracy = 66%
  - 66B words → accuracy = 72%

3-3. Empirical Result

- HS, NCE, NEG, Subsampling 비교
- Analogy로 검증
  - "Germany" : "Berlin" :: "France" : ?,
  - closest to vec("Berlin") - vec("Germany") + vec("France")
  - Syntactic/Semantic
- Data set
  - 다양한 뉴스 기사(internal Google dataset with one billion words)
  - 5번 이하로 나타난 단어 제외
  - 총 692,000개 단어

3-3. Empirical Result

- NEG가 HS와 NCE보다 성능이 우수
- Subsampling 사용시 학습 속도와 정확도 향상

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

3-4. 선행 모델과의 비교

- 선행 모델과 비교시 training time ↓, quality ↑

Model (training time)	Redmond	Havel	ninjutsu	graffiti	capitulate
Collobert (50d) (2 months)	conyers lubbock keene	plauen dzerzhinsky osterreich	reiki kohona karate	cheesecake gossip dioramas	abdicate accede rearm
Turian (200d) (few weeks)	McCarthy Alston Cousins	Jewell Arzu Ovitz	- - -	gunfire emotion impunity	- - -
Mnih (100d) (7 days)	Podhurst Harlang Agarwal	Pontiff Pinochet Rodionov	- - -	anaesthetics monkeys Jews	Mavericks planning hesitated
Skip-Phrase (1000d, 1 day)	Redmond Wash. Redmond Washington Microsoft	Vaclav Havel president Vaclav Havel Velvet Revolution	ninja martial arts swordsmanship	spray paint grafitti taggers	capitulation capitulated capitulating

감사합니다.