머신러닝의 자연어 처리기술(I)

2016. 7.

김홍배

"言語と画像の表現学習", 慶應義塾大学, 野口裕貴

"Word2vec from scratch", KAIST, 이진표

목차

사진으로 연애이야기를 만드는 AI

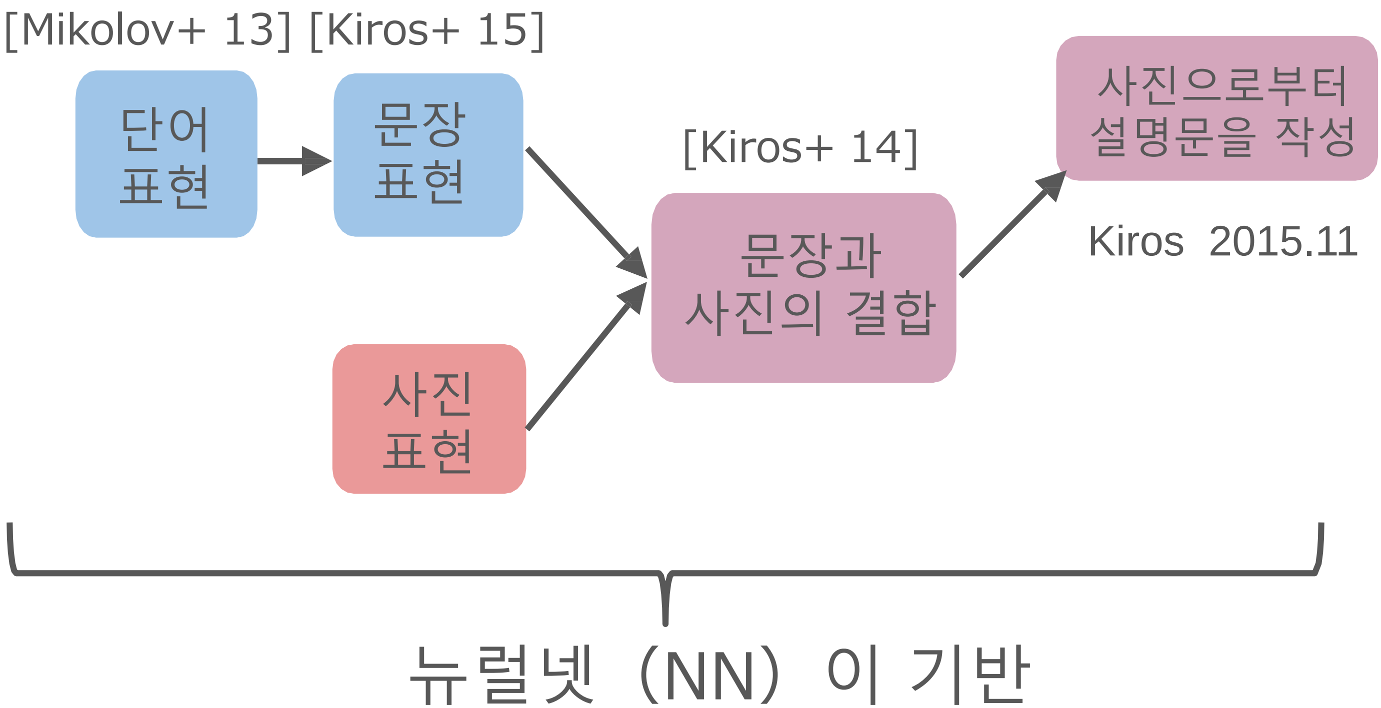
Generated story about image
Model: Romantic Novels

"We men were in a tense position at the end of the meeting. And i looked up at my best friend.

Of course, i had no intention of letting him go. I don't know what else to say, but he is also the most beautiful man you ever meet."

「회의 마지막에 우리들은 긴장된 분위기였다. 그리고 나는 친구를 쳐다봤다.
물론 나는 그가 가도록 놔둘 의도는 없었다. 달리 할말을 모르겠지만,
그는 가장 아름다운 남자다」

Neural Storyteller에서 사용되는 기술



오늘의 주제

- NN이 파워풀한 이유중 하나 : 분산표현
- NN에 언어 및 사진을 학습시키는 방법

분산표현 [Hinton+ 1986]

1986년、Geoffrey Hinton이 뉴론이 어떻게 개념을 나타내는가를 설명하기 위해서 분산표현 (distributed representation)을 제안

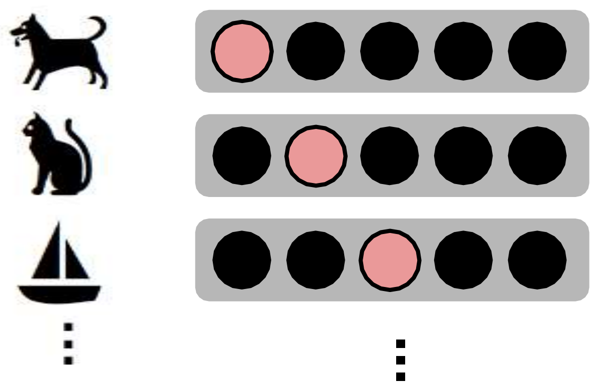
목차

1. 개요
2. **분산표현의 개요**
3. 자연어처리의 개요
4. 언어의 벡터표현
5. encoder-decoder 모델
6. 멀티모달모델

국소표현 vs 분산표현

국소표현

1개 뉴론의 작동으로 1개의 개념을 나타냄

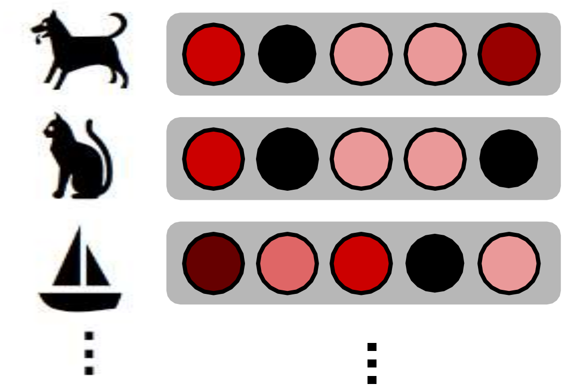


[1, 0, 0, 0, 0]
[0, 1, 0, 0, 0]
[0, 0, 1, 0, 0]
⋮

벡터형태로 나타내서 **one-hot vector**

분산표현

복수 뉴론의 작동으로 1개의 개념을 나타냄.



[0.5, 0.0, 1.0, 1.0, 0.3]

[0.5, 0.0, 1.0, 1.0, 0.0]

[0.2, 0.9, 0.5, 0.0, 1.0]

⋮

분산표현

개념을 **특징의 조합**으로 나타냄.

계수

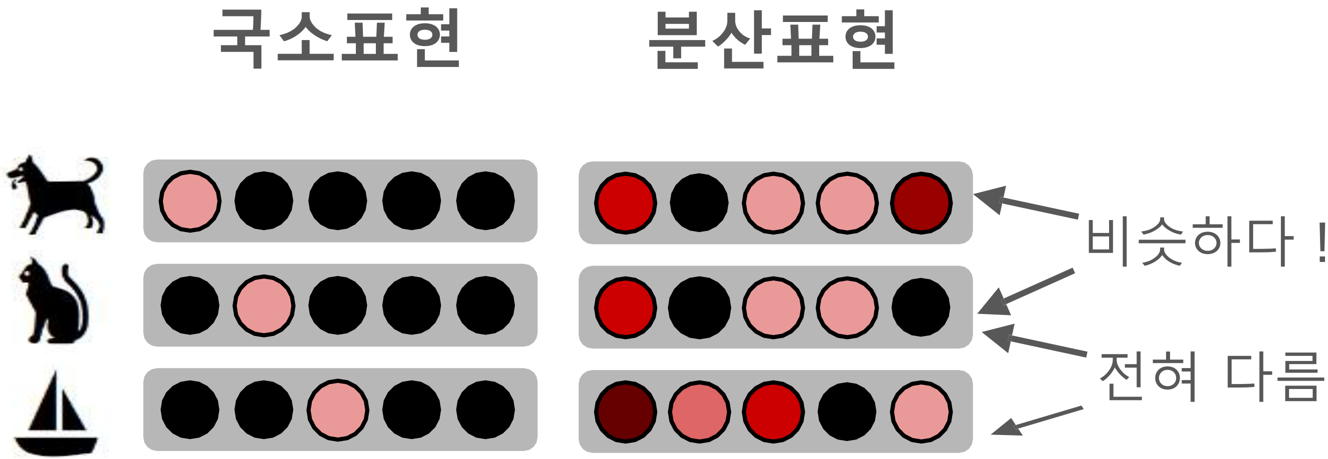
 = 1·펫 + 1 · 멍멍 + 0 · 야옹 + 0.1 · 타는것 + 0.1 · 바다

 = 1·펫 + 0 · 멍멍 + 1 · 야옹 + 0 .0· 타는것 + 0 · 바다

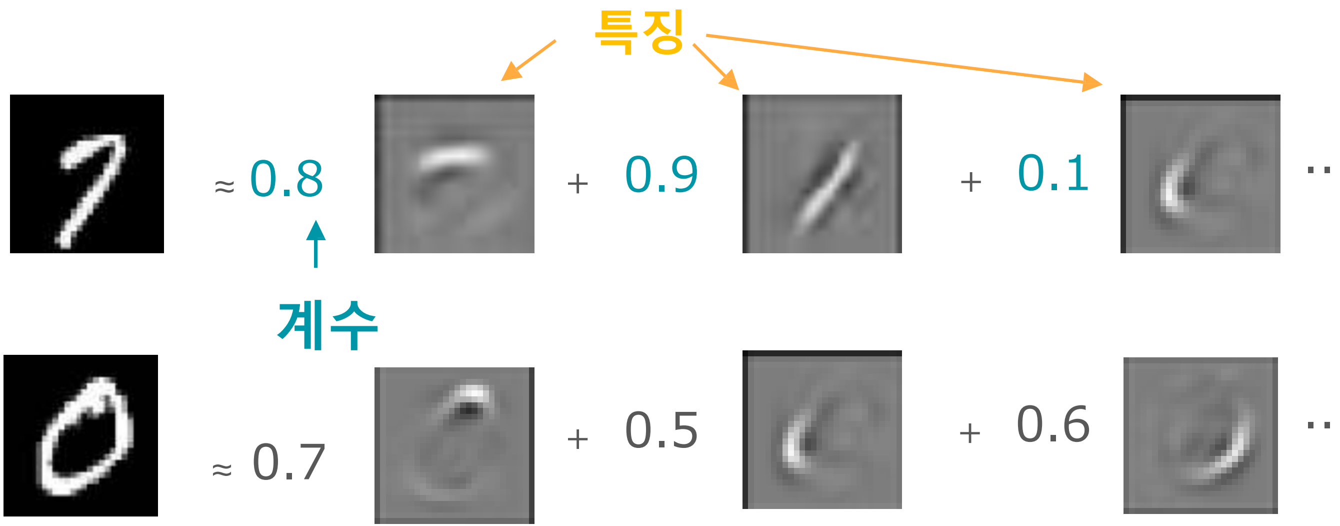
 = 0·펫 + 0 · 멍멍 + 0 · 야옹 + 0.9 · 타는것 + 0.8 · 바다

특징

개념의 유사



문자인식의 분산표현



딥뉴럴네트워크

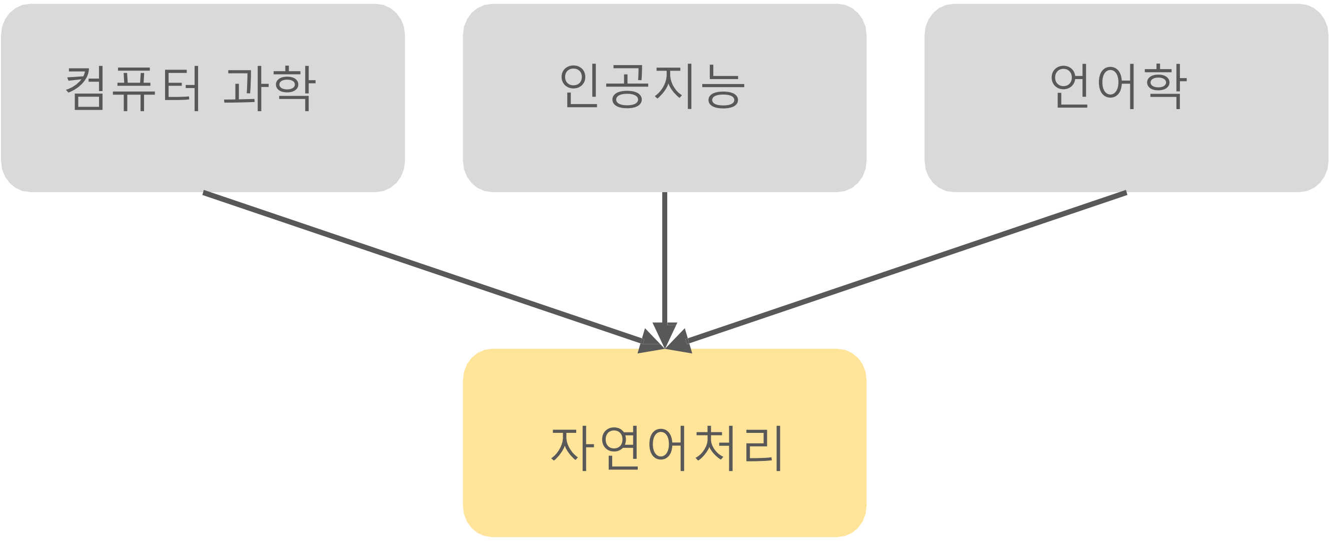
음성인식과 **사진인식**을 시작으로 다양한 분야에서 성과를 만들어내고 있다.

최근에는 **자연어처리**에도 활용되고 있다.

목차

자연어처리 (NLP)

컴퓨터과학, 인공지능과 언어학이 합쳐진 분야



자연어 처리 업무

쉬움 중간 어려움

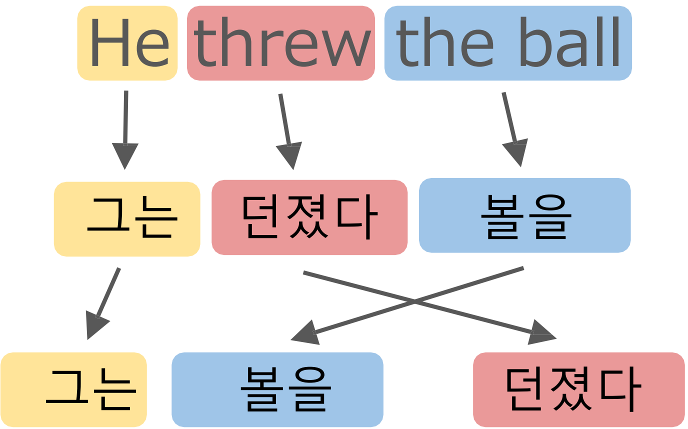
- 스펠링 체크
- 키워드 검사
- 유사어 감지

- 웹사이트 및 서류의 형태 해석
- 구문해석 etc.

- 기계번역
- 감정분석
- 질의응답시스템

기계번역

문구기반 번역의 예



단어의 모호성

감정분석

문장으로부터 감정을 판단

긍정적　「매우 **재밌다**. 아무리 놀아도 **실증나지않는다.**」

→ **0.86**

부정적　「설치하지마. 데이터만 **낭비**한다.」

→ **-0.68**

질문응답시스템(QA 시스템)

closed-domain – 정해진 분야의 질문에 응답

「라마는 무슨 과？」 → 「낙타과」

open-domain – 어떠한 질문에도 응답

「왜 나는 결혼을 못하나？」 → 「...」

자연어 처리의 어려움

언어・상황・환경・지각 지식의 학습 및 표현의 복잡함
→　Rule 기반만으로는 무리인가 ?

DNN은 분산표현의 장점으로 인해
모호하지만 풍부한 정보를 얻을 수 있다.
→　단어의 벡터화로부터 시작

목차

단어의 국소표현

고양이	[1, 0, 0, 0, 0]
개	[0, 1, 0, 0, 0]
사람	[0, 0, 1, 0, 0]
⋮	⋮

이것만 가지고는 단어의 의미를 전혀 알 수 없다.

→ 단어의 **의미를 파악하는 벡터**를 갖고 싶다.

분포가설 [Harris 1954, Firth 1957]

"You shall know a word by the company it keeps"
- J. R. Firth

비슷한 문맥을 가진 단어는 비슷한 의미를 갖는다.

현대의 통계적 자연어 처리에서 획기적인 발상

Count-based vs Predictive methods

분포가설에 기반한 방법은 크게 2종류로 나눈다.

- count-based methods
  - 예 : SVD (LSA)、HAL、etc.
  - 단어, 문맥 출현횟수를 세는 방법
- predictive methods
  - 예 : NPLM、word2vec、etc.
  - 단어에서 문맥 또는 문맥에서 단어를 예측하는 방법

Count-based vs Predictive methods

이번에는 이중에서 3개만 중점적으로

- count-based methods
  - 예 : **SVD (LSA)**、HAL、etc.
  - 단어, 문맥 출현횟수를 세는 방법
- predictive methods
  - 예 : **NPLM**、**word2vec**、etc.
  - 단어에서 문맥 또는 문맥에서 단어를 예측하는 방법

문맥(context)의 정의

- **문맥창**(다음 슬라이드)
- 자신 이외의 ○○중에 나타나는 단어
    - **문장**
    - **단락**
    - **문서**

문맥창

크기 $2k+1$의 단어열에 대해

기준단어(April)주변의 단어가 문맥

$k=4$ 의 예

기준

It was a bright cold day in April, and the clocks were striking thirteen.

전 4 단어 후 4 단어

문맥

단어문맥행렬(co-occurance matrix)

예 : k=1 의
문맥창으로 한 경우

$|V|$ 는 어휘수

I enjoy technology.
I like eating.
I like to sleep.

"like" 앞에 "I"가 두번 출현
"like" 뒤에 "eating"이 한번 출현

	I	enjoy	technology	like	eating	to	sleep	.
I	0	1	0	2	0	0	0	0
enjoy	1	0	1	0	0	0	0	0
technology	0	1	0	0	0	0	0	1
like	**2**	0	0	0	**1**	**1**	0	0
eating	0	0	0	1	0	0	0	1
to	0	0	0	1	0	0	1	0
sleep	0	0	0	0	0	1	0	1
.	0	0	1	0	1	0	1	0

$|V|$ (rows) × $|V|$ (columns)

단어문맥행렬

technology	0	1	0	0	0	0	0	1
like	2	0	0	0	1	1	0	0
eating	0	0	0	1	0	0	0	1

어휘수

각행을 단어벡터로 사용

그러나 단어가 많아지면,

벡터도 커진다(수십만 차원도 됨)

조밀한 벡터

고차원 벡터의「가장 중요한 정보」를 유지하면서

저차원・조밀한 벡터로 압축하고 싶다.

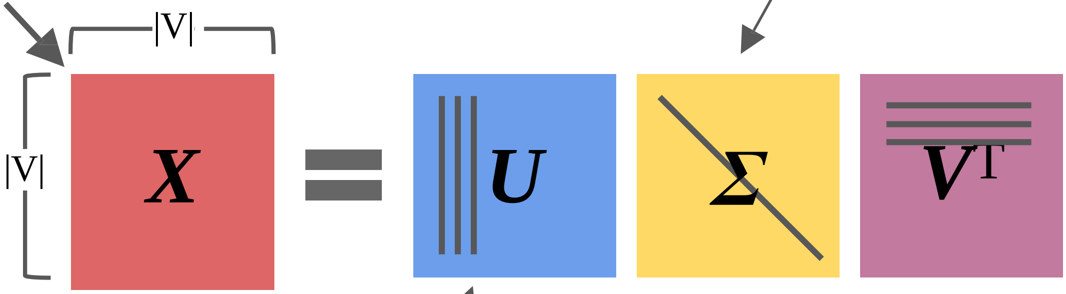
(e.g. 수십만 차원 → 수백 차원)

→ **특이치분해（Singular Value Decomposition, SVD)**

특이치분해（SVD）

대각선 값은 XX^T 와 X^TX 의 고유치
(크기순)

단어문맥행렬



각열은 XX^T 의
고유벡터

대응하는 고유치
의 크기순으로
나열됨.

각행은 X^TX 의
고유벡터

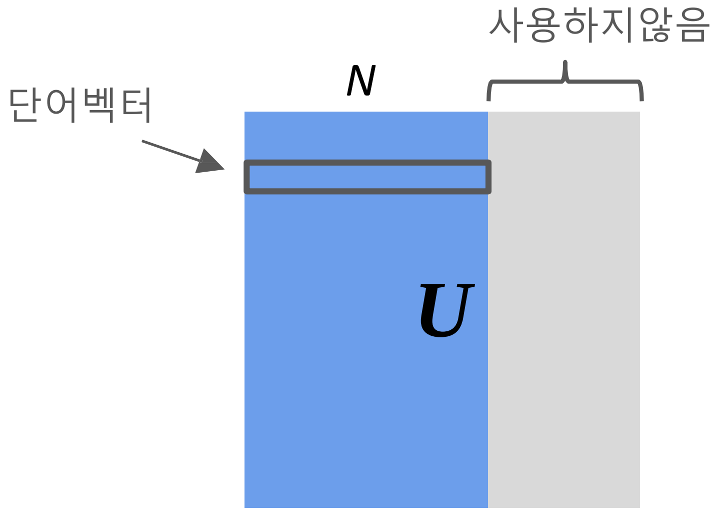
$$\Sigma = \begin{bmatrix} \sigma_1 & & & \\ & \sigma_2 & & \\ & & \ddots & \\ & & & \sigma_n \end{bmatrix}$$

$\sigma_1 > \sigma_2 > \ldots\ldots > \sigma_n$

앞쪽이 중요하고
뒤로 갈 수록 중요도가 낮아짐

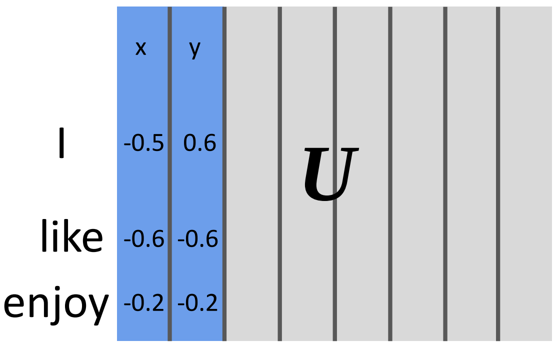
단어벡터

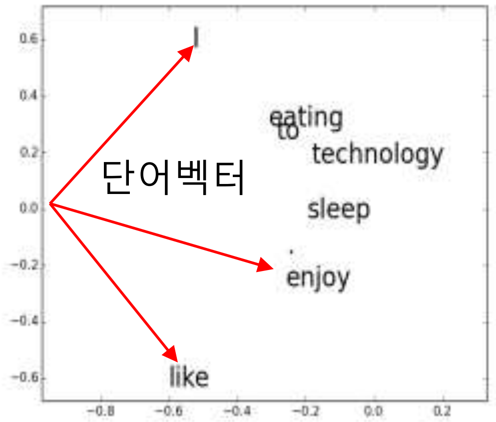
- U 의 각행을 **단어벡터**로 사용하되
- 중요도가 높은 앞쪽 고유치(N개)에 해당하는 U 의 앞쪽 N열까지 사용



단어벡터

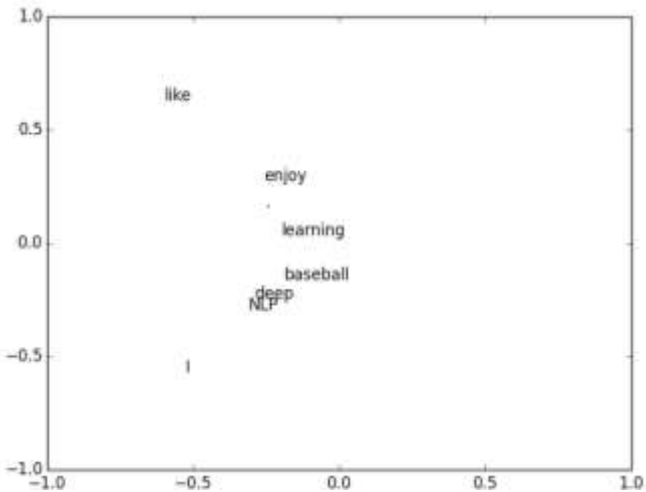
처음 2열까지 가시화 예





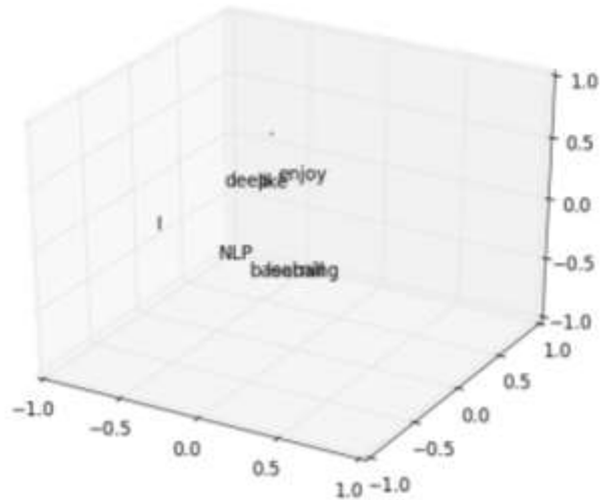
단어벡터 예

K = 2



like
enjoy
learning
baseball
deep
NLP
I
1.0
0.5
0.0
−0.5
−1.0
−1.0
−0.5
0.0
0.5
1.0

K = 3



enjoy
I
NLP
1.0
0.5
0.0
−0.5
−1.0

좀더 본격적으로

다음은 Brown Corpus 를 사용해보자

- 단어수 : 약 100만
- 어휘수 : 공간으로 나눈 결과, 약 8만
- 우선 단어문맥행렬을 만들어 보려는데....

```
  File "svdwords.py", line 38, in createMatrix
    X = np.zeros((len(d), len(d)))
MemoryError
```

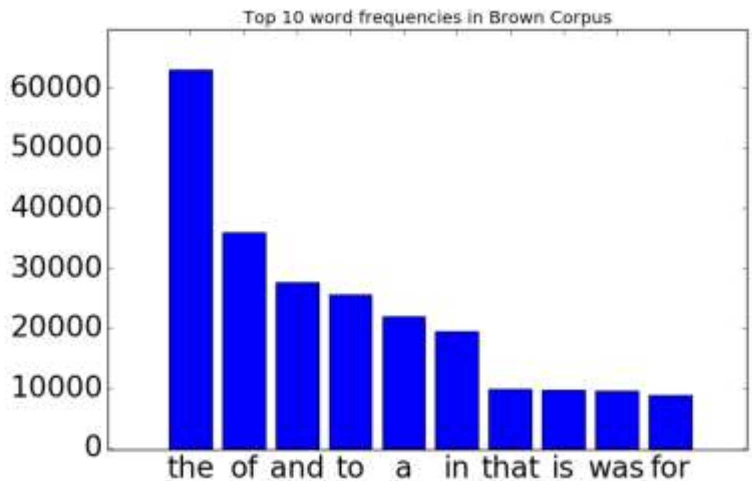
어휘수 줄이기

단어문맥행렬이 너무 크다

(어휘수 8만 → 80,000x80,000)

→ 발생빈도로 1,000번 이하의 단어를 정리

(어휘수 1,000 → 행렬 1,001x1,001)



단어벡터의 가시화

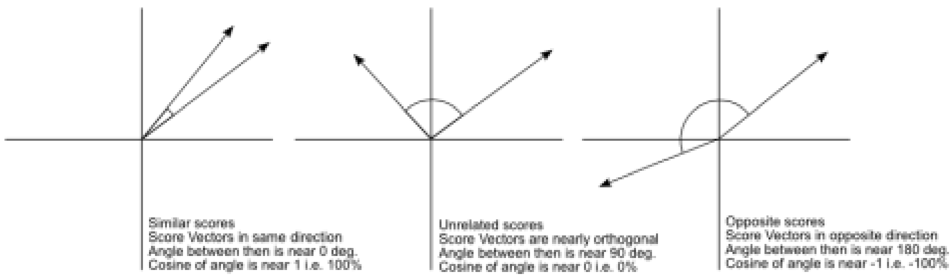
100차원의 벡터
(U의 100열까지 사용)

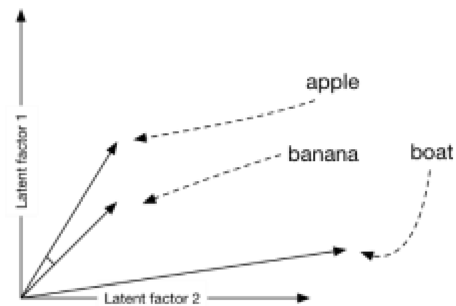
유사한(similarity, correlation, 관련성) 단어

유사도 $= cos(\theta) = \dfrac{w_1 \cdot w_2}{\|w_1\| \, \|w_2\|}$

w_1과 w_2가 유사하면 → 1

w_1과 w_2가 관련없으면 → 0



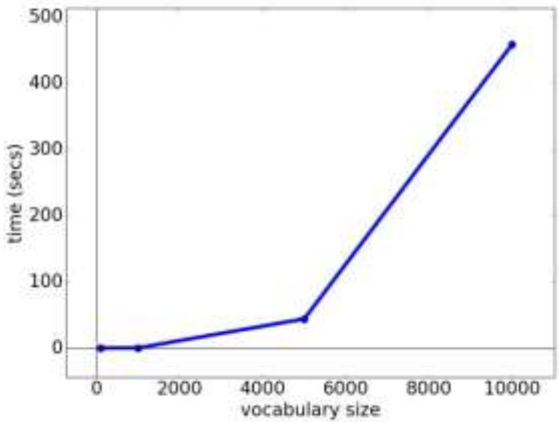


계산량 문제

새로운 텍스트 데이터를 사용할 경우,
단어문맥 행렬을 새롭게 만들어 SVD을 다시 계산

SVD 계산량은 $n \times m$행렬의 경우, $O(mn^2)$ $(n < m)$
→ 즉 어휘수에 한계

실제 어휘수를 늘려본 결과



몇일 소요

↑

100000

뉴럴 확률 언어 모델 [Bengio+ 2003]

NN으로 만들어내는 **언어모델**

→　　언어모델은 또 뭐야？

언어모델

단어열의 문법과 의미가 올바른 정도를 나타내는 확률을 계산하는 모델

$$P_{\mathrm{LM}}(\text{밥을 먹다}) > P_{\mathrm{LM}}(\text{먹다 밥을})$$

응용예 : 단어입력, 스펠링 체크, 기계번역 및 음성인식에 있어서 복수 문장후보의 평가에 사용

음성인식의 예

우리 인류의 궁극적인 꿈
0.023%

우리 여기에 공급적인 com
0.002%

김성훈 교수님 강의 중

n-gram 언어모델

계산량의 한계로 조건을 붙여 확률을 근사화

$$P(w_1, \ldots, w_m) = \prod_{i=1}^{m} P(w_i \mid w_1, \ldots, w_{i-1}) \approx \prod_{i=1}^{m} P(w_i \mid w_{i-(n-1)}, \ldots, w_{i-1})$$

어떤 단어의 출현확률은 이전 (n-1)개의 단어에 의존한다.

n=4 의 경우

...man stood $\underbrace{\text{still as they slowly}}_{\text{조건}}$ walked through the...

이것을 **n-1차 마코프 과정**이라고 함.

n-gram 언어모델

unigram（n=1）

순서를 전혀 고려하지 않음

P(He plays tennis.)=P(He)*P(plays)*P(tennis)*P(.)

bigram（n=2）

P(He plays tennis.) = P(He)*P(plays|He)*P(tennis|plays)*P(.|tennis)

trigram（n=3）

P(He plays tennis.) = P(He)*P(plays|He)*P(tennis|He plays)*P(.|plays tennis)

…

n-gram로 언어모델링

이와 같이 n-gram의 n을 증가시킨다.
n 을 증가시켜도, 데이터가 충분하면 성능은 좋아진다.

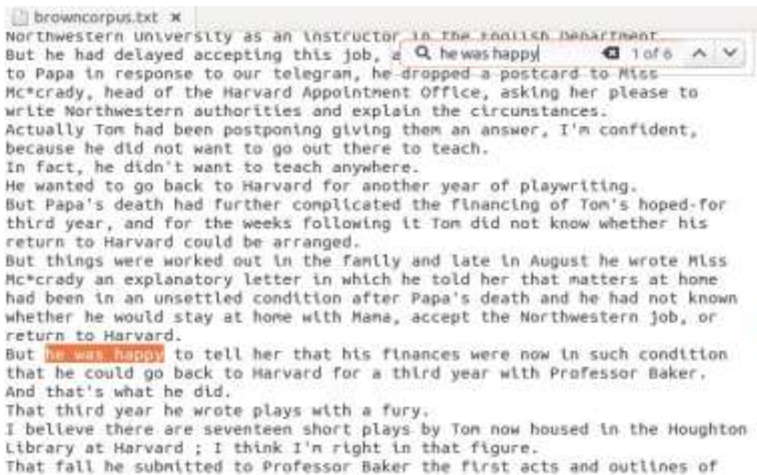
그러나 단어가 가질 수 있는 조합이 $|V|^n$ 로 지수적으로 커진다.

→ 지수적으로 학습데이터가 필요해짐.

n-gram에 있어서 문제점

Brown Corpus를 3-gram으로 언어모델을 만들어보면

“he was happy”는 **6건** 나옴


browncorpus.txt

he was happy 1 of 6

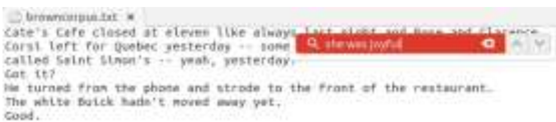
Northwestern University as an instructor in the English Department
But he had delayed accepting this job, a
to Papa in response to our telegram, he dropped a postcard to Miss
Mc*crady, head of the Harvard Appointment Office, asking her please to
write Northwestern authorities and explain the circumstances.
Actually Tom had been postponing giving them an answer, I'm confident,
because he did not want to go out there to teach.
In fact, he didn't want to teach anywhere.
He wanted to go back to Harvard for another year of playwriting.
But Papa's death had further complicated the financing of Tom's hoped-for
third year, and for the weeks following it Tom did not know whether his
return to Harvard could be arranged.
But things were worked out in the family and late in August he wrote Miss
Mc*crady an explanatory letter in which he told her that matters at home
had been in an unsettled condition after Papa's death and he had not known
whether he would stay at home with Mama, accept the Northwestern job, or
return to Harvard.
But he was happy to tell her that his finances were now in such condition
that he could go back to Harvard for a third year with Professor Baker.
And that's what he did.
That third year he wrote plays with a fury.
I believe there are seventeen short plays by Tom now housed in the Houghton
Library at Harvard ; I think I'm right in that figure.
That fall he submitted to Professor Baker the first acts and outlines of

n-gram에 있어서 문제점

"she was joyful"은 **"0"**

→ n-gram 모델이라면 확률 **0%**

→ **Smoothing** 등의 처리를 하는 경우가 있음.



간단한 예
(add-one smoothing)

$$P(x) = \frac{c(x) + 1}{N + |V|}$$

그러나 문제는 완전히 해결되지 않음.

분산표현의 장점

he was happy
she was joyful

→ 유사함 → 한쪽의 확률이 높아지면, 다른 한쪽도 높아짐

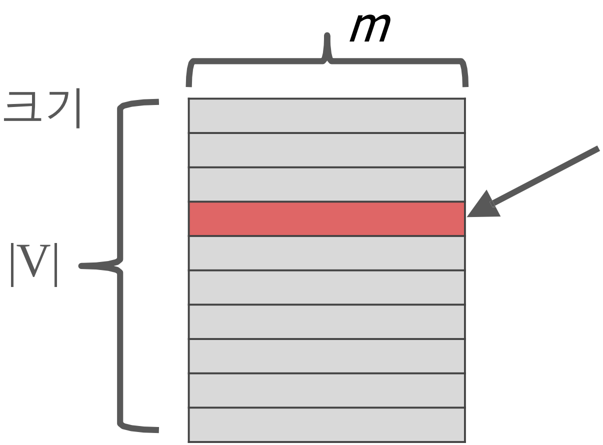
유사성을 고려할 수 있다면, 일반화 능력을 향상시킴

이것은 **분산표현**으로 할 수 있는 일 → NN

임베드행렬(embedding matrix)

단어벡터(임베드 벡터)의 집합

m 은 임베드벡터의 크기
(임의값)



i 행렬이
단어 i 의 백터

이행렬을 NN으로 훈련시키면 ?

뉴럴 확률언어모델(NPLM)

단어열로부터 다음 단어를 예측

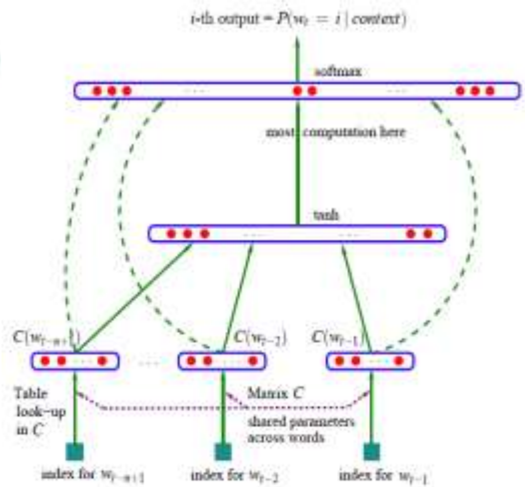
(e.g. Apples are_______________)

$$x = (C_{w_{t-n+1},1}, \ldots, C_{w_{t-n+1},d}, C_{w_{t-n+2},1}, \ldots C_{w_{t-2},d}, C_{w_{t-1},1}, \ldots C_{w_{t-1},d})$$

$$a_k = b_k + \sum_{i=1}^{h} W_{ki} \tanh(c_i + \sum_{j=1}^{(n-1)d} V_{ij} x_j)$$

$$P(w_t = k | w_{t-n+1}, \ldots w_{t-1}) = \frac{e^{a_k}}{\sum_{l=1}^{N} e^{a_l}}$$

$$L(\theta) = \sum_t \log P(w_t | w_{t-n+1}, \ldots w_{t-1})$$



NPLM과 단어벡터

NPLM의 임베드행렬 C의 각행을 단어벡터로서 사용

그러나, NPLM의 첫번째 목적은 언어 모델

단어벡터는 **부산물**

단어벡터를 얻어내는 것에 적합한 방법이 있다면 ?

word2vec [Mikolov+ 2013]

CBOW（연속 bag-of-words）모델

- 문맥으로부터 단어를 예측
- 소규모 데이터 셋에 대하여 성능이 좋음

skip-gram 모델

- 단어로부터 문맥을 예측
- 대규모 데이터 셋에 사용됨

skip-gram은 성능이 좋고 빨라서 인기

CBOW (Original)

- **Continuous-Bag-of-word model**

  - Idea: Using context words, we can predict center word

    i.e. Probability(“It is (**?**) to finish” → “**time**”)

    context words (window_size=2)

  - Present word as distributed vector of probability → Low dimension
  - Goal: Train weight-matrix($\boldsymbol{W}$) satisfies below

$$\operatorname{argmax}_W \{Minimize\left(\left|\boldsymbol{time} - softmax\left(pr\left(\boldsymbol{time}\,|\boldsymbol{it}, \boldsymbol{is}, \boldsymbol{to}, \boldsymbol{finish}\right)\right)\right| ; W\right)\}$$

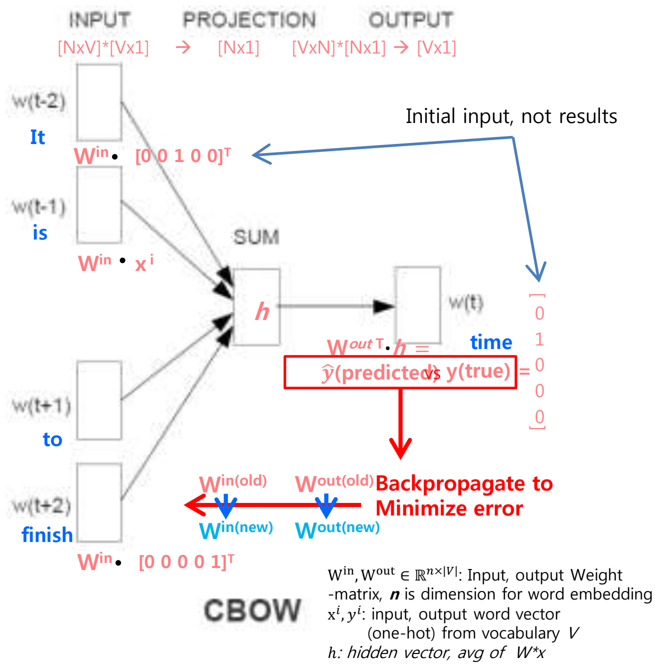
* Softmax(): K-dim vector of x∈ℝ → K-dim vector that has (0,1)∈ℝ

  - Loss-function (using cross-entropy method)

$$E = -\log p(w_t | w_{t-C} .. w_{t+C})$$

CBOW (Original)

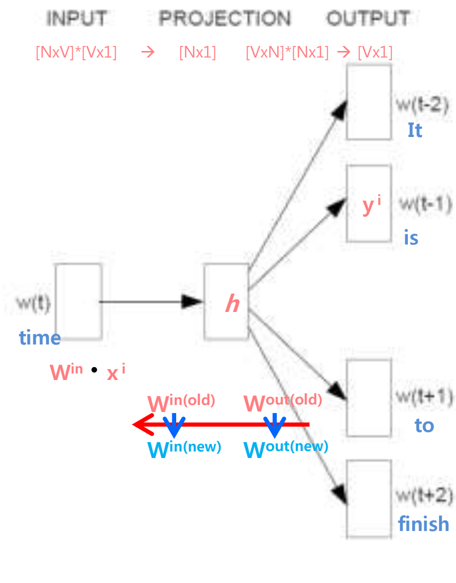
- **Continuous-Bag-of-word model**
  - Input
    - "one-hot" word vector
  - Remove nonlinear hidden layer
  - Back-propagate error from output layer to Weight matrix (Adjust ***W*** s)



Skip-Gram (Original)

- **Skip-gram model**
  - Idea: With center word, we can predict context words
  - Mirror of CBOW (vice versa)

i.e. Probability("**time**" → "It is (**?**) to finish")



Skip-gram

- Loss-function:

$$E = -\log p(w_{t-c}..w_{t+c}|w_t)$$

CBOW: $E = -\log p(w_t|w_{t-c}..w_{t+c})$

Extension of Skip-Gram(1)

- **Hierarchical Soft-max function**
  - To train weight matrix in every step, we need to pass the calculated vector into Loss-Function

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

$\boldsymbol{T}$: *whole step*
$\boldsymbol{c}$: *size of training context(window)*
$\boldsymbol{w_t}, \boldsymbol{w_{t+j}}$: *curent step's word and j − th word* $\boldsymbol{w_t}$

  - Soft-max function
    - Before calculate loss function $(\boldsymbol{E} = -\log \boldsymbol{p}(\boldsymbol{w_{t-C}}..\boldsymbol{w_{t+C}}|\boldsymbol{w_t}))$ calculated vector should normalized as real-number in (0,1)

Extension of Skip-Gram(1)

- **Hierarchical Soft-max function (cont.)**
  - Soft-max function

    (I have already calculated, it's boring ……….)

$$
\begin{aligned}
\text{minimize } J &= -\log P(w^{(i-C)}, \ldots, w^{(i-1)}, w^{(i+1)}, \ldots, w^{(i+C)} | w^{(i)}) \\
&= -\log \prod_{j=0, j\neq C}^{2C} P(w^{(i-C+j)} | w^{(i)}) \\
&= -\log \prod_{j=0, j\neq C}^{2C} P(v^{(i-C+j)} | u^{(i)}) \\
&= -\log \prod_{j=0, j\neq C}^{2C} \boxed{\frac{\exp(v^{(i-C+j)T} h)}{\sum_{k=1}^{\boxed{|V|}} \exp(v^{(k)T} h)}} \\
&= -\sum_{j=0, j\neq C}^{2C} v^{(i-C+j)T} h + 2C \log \sum_{k=1}^{|V|} \exp(v^{(k)T} h)
\end{aligned}
$$

Original soft-max function of skip-gram model

Extension of Skip-Gram(1)

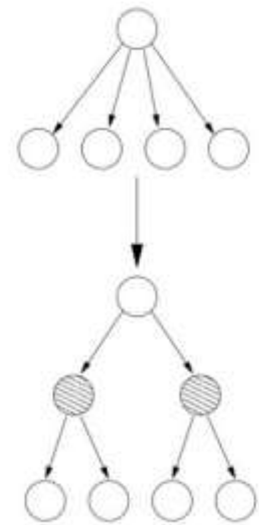
- **Hierarchical Soft-max function (cont.)**
  - Since V is quite large, computing $\log(p(w_o|w_I))$ costs to much
  - **Idea**: Construct **binary Huffman tree** with word

  → Cost: $O(|V|)$ **to** $O(\log|V|)$

  - **Can train Faster!**
  - Assigning
    - Word = $node(w, L(w))$ $by\ random\ walk$

$$p(w|w_I) = \prod_{j=1}^{L(w)-1} \sigma\left([\![n(w, j+1) = \mathrm{ch}(n(w, j))]\!] \cdot {v'_{n(w,j)}}^{\top} v_{w_I}\right)$$

(* details in "Hierarchical Probabilistic Neural Network Language Model ")



Extension of Skip-Gram(2)

Noise Constrastive Estimation

- **Negative Sampling (similar to NCE)**
  - Size(Vocabulary) is computationally huge! → Slow for train
  - Idea: Just sample several negative examples!

    i.e. "Stock boil fish is toy" ???? → negative sample

    - Do not loop full vocabulary, only use neg. sample → **fast**
    - Change the target word as negative sample and learn negative examples → **get more accuracy**
  - Objective function

$$E = -\log p(w_{t-C}..w_{t+C}|w_t) = \underset{\theta}{\operatorname{argmax}} \prod_{(w,c)\in D} P(D=1|w,c,\theta) \prod_{(w,c)\in \tilde{D}} P(D=0|w,c,\theta)$$

$$= -\log \sigma(v^{(i-C+j)} \cdot h) + \sum_{k=1}^{K} \log \sigma(\tilde{v}^{(k)} \cdot h)$$

Extension of Skip-Gram(3)

- **Subsampling**
  - ("Korea", "Seoul") is helpful, but ("Korea", "the") isn't helpful
  - **Idea:** Frequent word vectors (i.e. "the") should not change significantly after training on several million examples.
    - Each word w_i in the training set is discarded with below probability

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

$\mathrm{f(w_i)}: frequncy\ of\ word\ w_i$
$t: chosen\ threshold, around\ 10^{-5}$

    - It aggressively subsamples frequent words while preserve ranking of the frequencies
    - But, this formula was chosen heuristically…

Extension of Skip-Gram

- **Evaluation**
  - **Task**: Analogical reasoning
    - Accuracy test using cosine similarity determine how the model answer correctly.

      i.e. vec(X) = vec("Berlin") – vec("Germany") + vec("France")

      Accuracy = cosine_similarity(vec(X), vec("Paris"))
  - **Model**: skip-gram model(**Word-embedding dimension** = 300)
  - **Data Set**: News article (Google dataset with 1 billion words)
  - **Comparing Method** (w/ or w/o 10^{-5}subsampling)
    - NEG(Negative Sampling)-5, 15
    - Hierarchical Softmax-Huffman
    - NCE-5(Noise Contrastive Estimation)

Extension of Skip-Gram

- **Empirical Results**

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.

  - Model w/ **NEG** outperforms the **HS** on the analogical reasoning task (even slightly better than NCE)
  - The **subsampling** improves the training speed several times and makes the word representations more accurate

Learning Phrases

- **Word base model can not represent idiomatic word**
  - i.e. "Newyork Times", "Larry Page"

- **Simple data driven approach**
  - If phrases are formed based on 1-gram, 2-gram counts

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

$\delta: discounting\ coefficient$
$(Prevent\ too\ many\ phrases\ consisting\ of\ infrequent\ words)$

  - Target words that has high score would meaningful phrase

Learning Phrases

- **Evaluation**
  - **Task**: Analogical reasoning
    - Accuracy test using cosine similarity determine how the model answer correctly **with phrase**
    - i.e. vec(X) = vec("Steve Ballmer") – vec("Microsoft") + vec("Larry Page")

      Accuracy = cosine_similarity(vec(x), vec("Google"))
  - **Model**: skip-gram model(**Word-embedding dimension** = 300)
  - **Data Set**: News article (Google dataset with 1 billion words)
  - **Comparing Method** (w/ or w/o 10^{-5}subsampling)
    - NEG-5
    - NEG-15
    - HS-Huffman

Learning Phrases

- **Empirical Results**

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.

- **NEG-15** achieves better performance than **NEG-5**
- **HS** become the best performing method when **subsampling**
  - This shows that the subsampling can result in faster training and can also improve accuracy, at least in some cases.
- When **training set** = 33 billion, **d**=1000 → **72% (6B → 66%)**
  - **Amount of training set is crucial!**

Additive Compositionality

- **Simple vector addition (on Skip-gram model)**
  - Previous experiments shows *Analogical reasoning (A+B-C)*
  - Vector's values are related logarithmically to the probabilities
  - → Sum of two vector is related to product of context distribution

- **Interesting!**

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.

Conclusion

- **Contributions**
  - Showed detailed process of training distributed representation of words and phrases
  - Can be more accurate and faster model than previous word2vec model by **sub-sampling**
  - **Negative Sampling**: Extremely simple and accurate for frequent words. (not frequent like phrase, **HS** was better)
  - Word vectors can be meaningful by simple vector addition
  - Made a code and dataset as open-source project

Conclusion

- **Compare to other Neural network model**

<Find most similar word>

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

Table 6: Examples of the closest tokens given various well known models and the Skip-gram model trained on phrases using over 30 billion training words. An empty cell means that the word was not in the vocabulary.

- Skip-gram model trained on large corpus outperforms all to other paper's models.