NLP

NATURAL LANGUAGE PROCESSING

학부연구생 김지현

CONTENT

밑바닥부터 시작하는 딥러닝2

1 수학과 파이썬 복습

* Numpy 라이브러리로 파이썬에서 행렬 표현

(1) 벡터와 행렬

그림 1-1 벡터와 행렬의 예

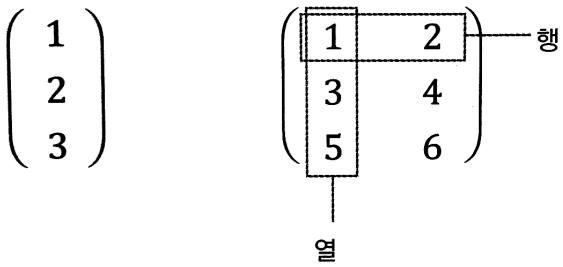
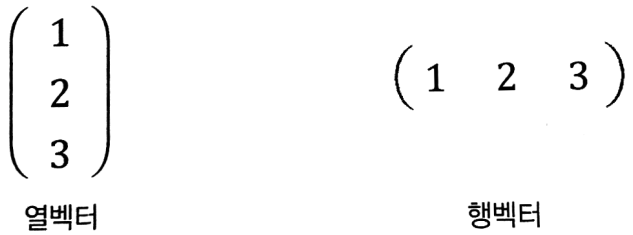


그림 1-2 벡터의 표현법



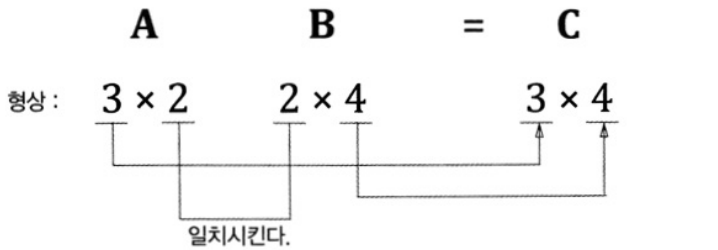
(2) 벡터의 내적과 행렬의 곱

: 두 벡터가 얼마나 같은 방향을 향하고 있는지를 나타낸다.

$$\mathbf{x} \cdot \mathbf{y} = x_1 y_1 + x_2 y_2 + \cdots + x_n y_n$$

(3) 행렬의 곱과 형상 확인

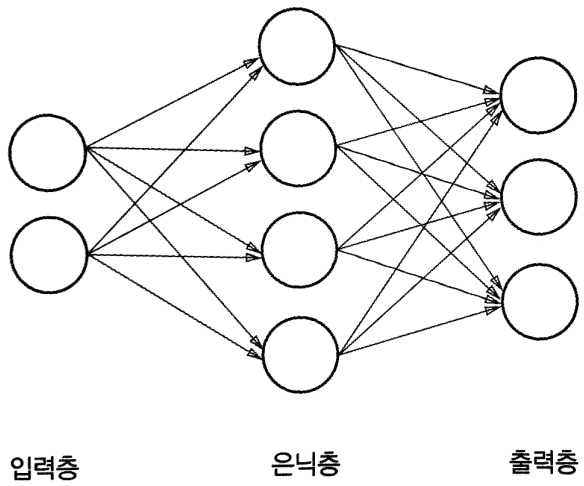
그림 1-6 형상 확인: 행렬의 곱에서는 대응하는 차원의 원소 수를 일치시킨다.



참고: '100 numpy exercise' 사이트

신경망 (완전연결계층)

그림 1-7 신경망의 예



신경망이 수행하는 계산의 수식

$$h_1 = x_1 w_{11} + x_2 w_{21} + b_1$$

h1: 은닉층 중 첫 번째 뉴런
(x1, x2) : 입력층의 데이터
w11, w21 : 가중치
b1 : 편향

(1) '가중치의 합'으로 계산
(2) 행렬의 곱으로 한꺼번에 계산

은닉층의 뉴런들

$$(h_1, h_2, h_3, h_4) = (x_1, x_2) \begin{pmatrix} w_{11} & w_{12} & w_{13} & w_{14} \\ w_{21} & w_{22} & w_{23} & w_{24} \end{pmatrix} + (b_1, b_2, b_3, b_4)$$

2 자연어 처리 개요

한국어와 영어 등
우리가 평소에 쓰는 말

자연어란

자연어처리

자연어를 처리하는 분야,
즉 우리의 말을 컴퓨터에게 이해시키기 위한 분야

컴퓨터가 우리의 말을 알아듣게
만드는 것

본질적인 문제

추구하는 목표

사람의 말을 컴퓨터가 이해하도록 만들어서,
컴퓨터가 우리에게 도움이 되는 일을 수행하게 하는 것

Programming Language

컴퓨터가 이해할 수 있는 언어?

'프로그래밍 언어'나 '마크업 언어' 이러한 언어들은 모든 코드의 의미를 고유하게 해석할 수 있도록 문법이 정의되어 있다. 컴퓨터는 정해진 규칙에 따라서 코드를 해석하기 때문에 컴퓨터가 이해할 수 있는 일반적인 프로그래밍 언어는 기계적이고 고정적이다. 즉 딱딱한 언어이다.

Natural Language

영어나 한국어는?

반면 영어나 한국어는 부드러운 언어이다. 똑같은 의미의 문장도 여러 형태로 표현 가능하고. 문장의 뜻이 애매할 수 있으며 의미나 형태가 유연하다. 세월이 흐르면서 새로운 말이나 새로운 의미가 생겨나고 있던 것이 사라지기도 하는데 이는 자연어는 부드럽기 때문이다.

Now what

컴퓨터에게 자연어를 이해시키려면?

컴퓨터에게 자연어를 이해시키기란 평범한 방법으론 매우 어려운 도전이다. 만약 그 난제를 해결할 수 있다면 수많은 사람들에게 컴퓨터에게 일을 시킬 수 있다.
대표적인 예 - 검색 엔진, 기계 번역, 질의응답 시스템, IME(입력기 전환), 문장 자동요약, 감정분석

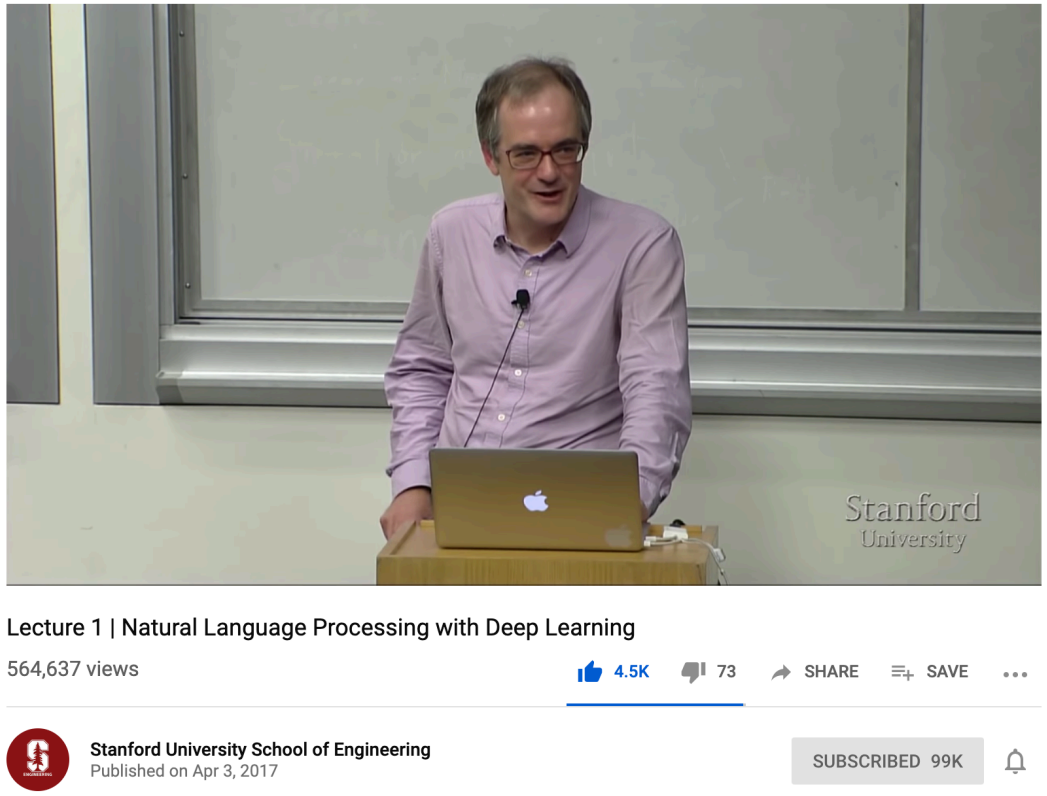
2 자연어와 단어의 분산 표현

컴퓨터에게 **단어의 의미** 이해시키는 것이 중요하다

단어의 의미를 잘 파악하는 표현 방법 3가지 기법

- 시소러스를 활용한 기법
- 통계 기반 기법
- 추론 기반 기법(word2vec)

Reference



스탠퍼드 대학교의
'CS224d: Deep Learning for Natural Language Processing' 수업

링크: https://www.youtube.com/watch?v=OQQ-W_63UgQ&list=PL-IaZsQsQJOS7dv4EMBq_tDMZ9re6dY2B

2 자연어와 단어의 분산 표현

(1) 시소러스

그림 2-1 동의어의 예: "car", "auto", "automobile" 등은 "자동차"를 뜻하는 동의어다.

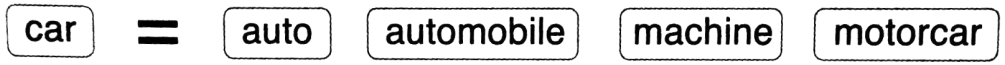
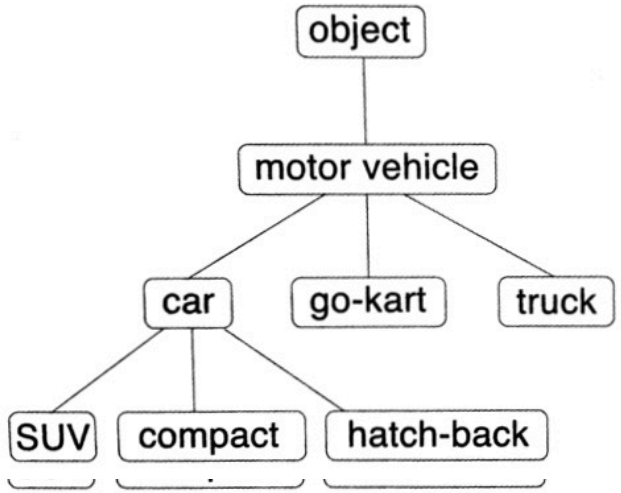


그림 2-2 단어들을 의미의 상 · 하위 관계에 기초해 그래프로 표현한다(문헌 [14]를 참고하여 그림).



사람이 직접 단어의 의미를 정의하는 방식
(표준국어대사전처럼 시소러스 형태의 사전)

시소러스란? 유의어 사전

가장 유명한 시소러스는 WordNet

그러나 문제점들.. 多

2 자연어와 단어의 분산 표현

(2) 통계 기반 기법

말뭉치(corpus) 이용

: 말뭉치란 대량의 텍스트 데이터이다.

단어의 분산 표현

'단어'도 벡터로 표현할 수 있을까?

분포 가설(distributional hypothesis)

단어의 의미는 주변 단어에 의해 형성된다

2 자연어와 단어의 분산 표현

(2) 통계 기반 기법

단어 자체는 의미가 없고 그 단어가 사용된 **'맥락(context)'**이 의미를 형성한다.

통계 기반 기법은 맥락에 주목한다

그림 2-3 윈도우 크기가 2인 '맥락'의 예. 단어 "goodbye"에 주목한다면, 그 좌우의 두 단어(총 네 단어)를 맥락으로 이용한다.

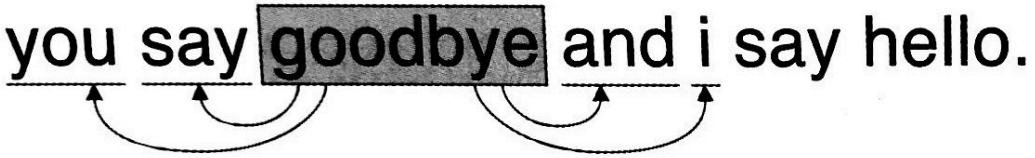


그림 2-4 단어 "you"의 맥락을 세어본다.

'맥락'이란 특정 단어를 중심에 둔 그 주변 단어를 말하고
맥락의 크기를 (주변 단어를 몇 개나 포함할지)를 '**윈도우 크기**(window size)'라고 한다

2 자연어와 단어의 분산 표현

그림 2-5 단어 "you"의 맥락에 포함되는 단어의 빈도를 표로 정리한다.

	you	say	goodbye	and	i	hello	.
you	0	1	0	0	0	0	0

"you"라는 단어를 [0,1,0,0,0,0,0]이라는 벡터로 표현

마찬가지로 say에 대해서도

"say"라는 단어는 벡터 [1,0,1,0,1,1,0]

그림 2-6 단어 "say"의 맥락에 포함되는 단어의 빈도를 표로 정리한다.

	you	say	goodbye	and	i	hello	.
say	1	0	1	0	1	1	0

동시 발생 행렬!

co-occurrence matrix

그림 2-7 모든 단어 각각의 맥락에 해당하는 단어의 빈도를 세어 표로 정리한다.

	you	say	goodbye	and	i	hello	.
you	0	1	0	0	0	0	0
say	1	0	1	0	1	1	0
goodbye	0	1	0	1	0	0	0
and	0	0	1	0	1	0	0
i	0	1	0	1	0	0	0
hello	0	1	0	0	0	0	1
.	0	0	0	0	0	1	0

각 행은 단어를 표현한 벡터가 됨.
이 표는 행렬의 형태를 띄고 있기 때문에 동시발생 행렬 이라고 한다

벡터 간 유사도

코사인 유사도(cosine similarity)

두 벡터 x = (x1, x2, x3, ... xn) 과 y = (y1, y2, y3, ... yn) 코사인 유사도

$$\text{similarity}(\mathbf{x},\mathbf{y}) = \frac{\mathbf{x}\cdot\mathbf{y}}{\|\mathbf{x}\|\|\mathbf{y}\|} = \frac{x_1y_1 + \cdots + x_ny_n}{\sqrt{x_1^2 + \cdots + x_n^2}\sqrt{y_1^2 + \cdots + y_n^2}}$$

[식 2.1]

Python Code

자연어처리 딥러닝 구현 코드 예제

LEARN MORE

구현 예시1. 말뭉치로부터 동시발생 행렬을 만들어주는 함수

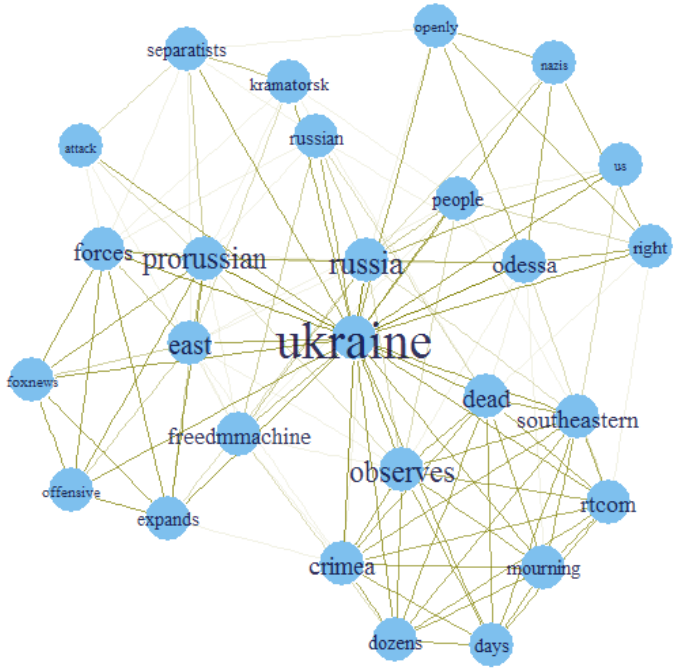
```
def create_co_matrix(corpus, vocab_size, window_size=1):
    corpus_size = len(corpus)
    co_matrix = np.zeros((vocab_size, vocab_size), dtype=np.int32)

    for idx, word_id in enumerate(corpus):
        for i in range(1, window_size + 1):
            left_idx = idx - i
            right_idx = idx + i

            if left_idx >= 0:
                left_word_id = corpus[left_idx]
                co_matrix[word_id, left_word_id] += 1

            if right_idx < corpus_size:
                right_word_id = corpus[right_idx]
                co_matrix[word_id, right_word_id] += 1

    return co_matrix
```

구현 예시2. 유사 단어의 랭킹을 표시 해주는 함수

```
def most_similar(query, word_to_id, id_to_word, word_matrix, top=5):
    #(1) 검색어를 꺼낸다.
    if query not in word_to_id:
        print('%s(을)를 찾을 수 없습니다.' % query)
        return
    print('\n[query] ' + query)
    query_id = word_to_id[query]
    query_vec = word_matrix[query_id]

    #(2) 코사인 유사도 계산
    vocab_size = len(id_to_word)
    similarity = np.zeros(vocab_size)
    for i in range(vocab_size):
        similarity[i] = cos_similarity(word_matrix[i], query_vec)

    # (3) 코사인 유사도를 기준으로 내림차순으로 출력
    count = 0
    for i in (-1 * similarity).argsort():
        if id_to_word[i] == query:
            continue
        print(' %s: %s' % (id_to_word[i], similarity[i]))

        count += 1
        if count >= top:
            return
```

동시발생 행렬의 문제점

car? the? drive?



통계 기반 기법 개선

점별 상호정보량(Pointwise Mutual Information(PMI))

PMI는 확률 변수 x와 y에 대해 다음 식으로 정의됨

$$\mathrm{PMI}(x,y) = \log_2 \frac{P(x,y)}{P(x)P(y)} \quad \textbf{[식 2.2]}$$

P(x): x가 일어날 확률
P(y: y가 일어날 확률
P(x, y): x와 y가 동시에 일어날 확률

PMI 값이 높을수록 관련성이 높다

$$\mathrm{PMI}(x,y) = \log_2 \frac{P(x,y)}{P(x)P(y)} = \log_2 \frac{\frac{C(x,y)}{N}}{\frac{C(x)}{N}\frac{C(y)}{N}} = \log_2 \frac{C(x,y)\cdot N}{C(x)C(y)} \quad \textbf{[식 2.3]}$$

-> 동시발생 행렬로부터 PMI 구하기

통계 기반 기법 개선

동시발생 횟수 관점일때 vs PMI 관점일때

$$\text{PMI}(\text{“the”},\text{“car”}) = \log_2 \frac{10 \cdot 10000}{1000 \cdot 20} \approx 2.32 \qquad \text{[식 2.4]}$$

$$\text{PMI}(\text{“car”},\text{“drive”}) = \log_2 \frac{5 \cdot 10000}{20 \cdot 10} \approx 7.97 \qquad \text{[식 2.5]}$$

PMI를 이용하면 car은 the보다 drive와 관련성이 더 강해졌다

why? 단어가 단독으로 출현하는 횟수가 고려되었기 때문

구현 예시3. 동시발생 행렬을 PPMI 행렬로 변환하는 함수

```python
def ppmi(C, verbose=False, eps=1e-8):
    M = np.zeros_like(C, dtype=np.float32)
    N = np.sum(C)
    S = np.sum(C, axis=0)
    total = C.shape[0] * C.shape[1]
    cnt = 0

    for i in range(C.shape[0]):
        for j in range(C.shape[1]):
            pmi = np.log2(C[i, j] * N / (S[j]*S[i] + eps))
            M[i, j] = max(0, pmi)

            if verbose:
                cnt += 1
                if cnt % (total/100) == 0:
                    print('%.1f%% 완료 ' % (100*cnt/total))
    return M
```

```
동시발생 행렬
[[0 1 0 0 0 0 0]
 [1 0 1 0 1 1 0]
 [0 1 0 1 0 0 0]
 [0 0 1 0 1 0 0]
 [0 1 0 1 0 0 0]
 [0 1 0 0 0 0 1]
 [0 0 0 0 0 1 0]]
--------------------------------------------------
PPMI
[[0.    1.807 0.    0.    0.    0.    0.   ]
 [1.807 0.    0.807 0.    0.807 0.807 0.   ]
 [0.    0.807 0.    1.807 0.    0.    0.   ]
 [0.    0.    1.807 0.    1.807 0.    0.   ]
 [0.    0.807 0.    1.807 0.    0.    0.   ]
 [0.    0.807 0.    0.    0.    0.    2.807]
 [0.    0.    0.    0.    0.    2.807 0.   ]]
```

3 자연어처리 최근 동향

1. Intent matching
사용자가 원하는 것 파악 (머신러닝 모델 이용)

2. Entities
built in system entity 이용

3. Dialog control
대화의 흐름을 구성

Scalar Lab

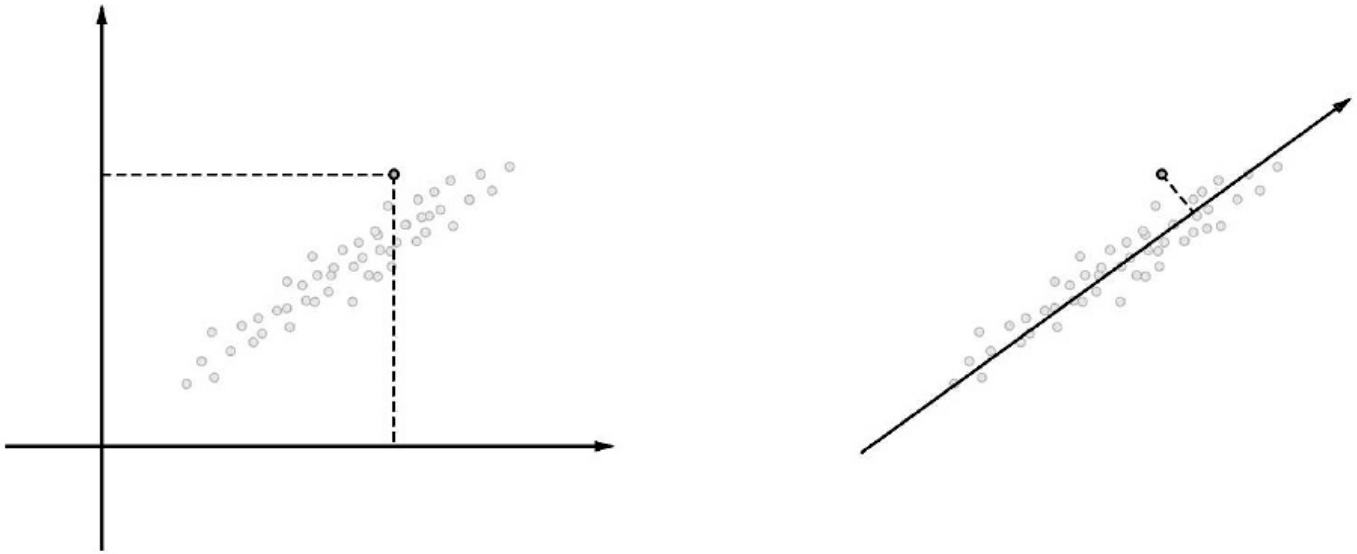
Word2vec

2가지 Architecture 모델 수학적으로 이해하기

review (+ 차원감소)

통계기반기법

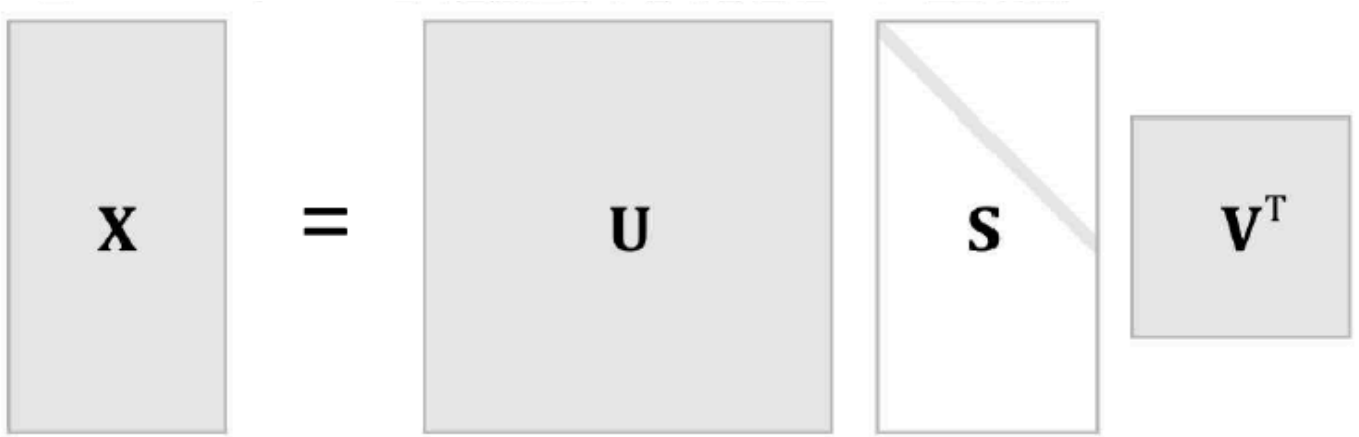
- 분포가설 / 단어의 분산 표현
- 윈도우 크기
- 동시발생 행렬 / PPMI 행렬
- **차원 감소**



특이값 분해

Singular Value Decomposition(SVD)

$$X = USV^T$$



U, V : 직교행렬 (orthogonal matrix)
S : 대각행렬 (diagonal matrix)

특이값 분해

Singular Value Decomposition(SVD)

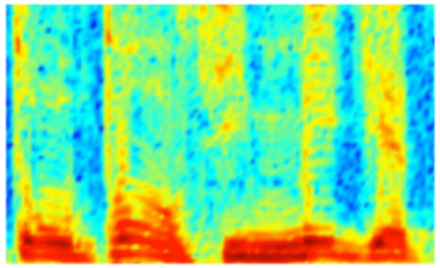
$$X = USV^T$$

U, V : 직교행렬 (orthogonal matrix)
S : 대각행렬 (diagonal matrix)

Word Embeddings

: vector representations of words

AUDIO



Audio Spectrogram

DENSE

IMAGES

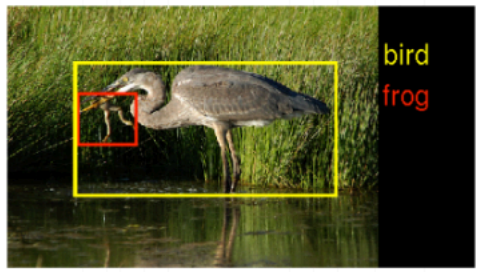
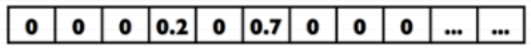


Image pixels

DENSE

TEXT



Word, context, or
document vectors

SPARSE

Problems with this discrete representation

The vast majority of rule-based and statistical NLP work regards words as atomic symbols: hotel, conference, walk

In vector space terms, this is a vector with one 1 and a lot of zeroes

[0 0 0 0 0 0 0 0 0 0 1 0 0 0 0]

Dimensionality: 20K (speech) – 50K (PTB) – 500K (big vocab) – 13M (Google 1T)

Word meaning is defined in terms of vectors

We will build a dense vector for each word type, chosen so that it is good at predicting other words appearing in its context

… those other words also being represented by vectors … it all gets a bit recursive

$$\textit{linguistics} = \begin{pmatrix} 0.286 \\ 0.792 \\ -0.177 \\ -0.107 \\ 0.109 \\ -0.542 \\ 0.349 \\ 0.271 \end{pmatrix}$$

출처: Stanford University NLP lecture2

추론 기반 기법

Word2vec

단어 임베딩(word embedding)을 생성하는 관련된 모델들

Tomas Mikolov
Google Inc.

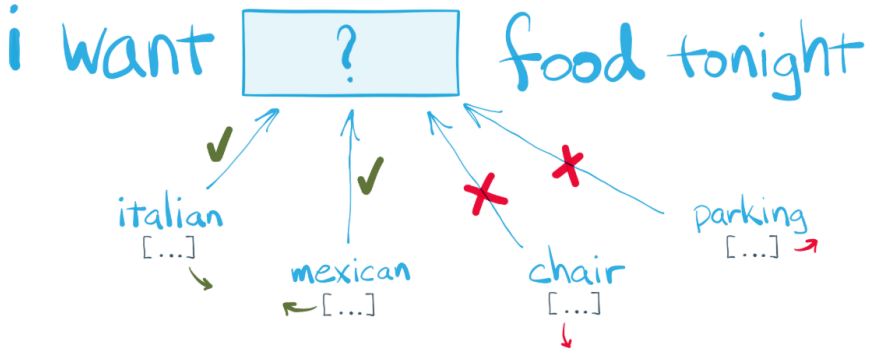
단순한 2층 layer 신경망 모델

- **CBOW** (continuous bag of words)
- **Skip-gram**

Q Word2vec은 단어를 표현하는 방법을 어떻게 학습하는 것일까?

단어의 주변을 보면 그 단어를 안다. You shall know a word by the company it keeps.
– 언어학자 J.R. Firth (1957)

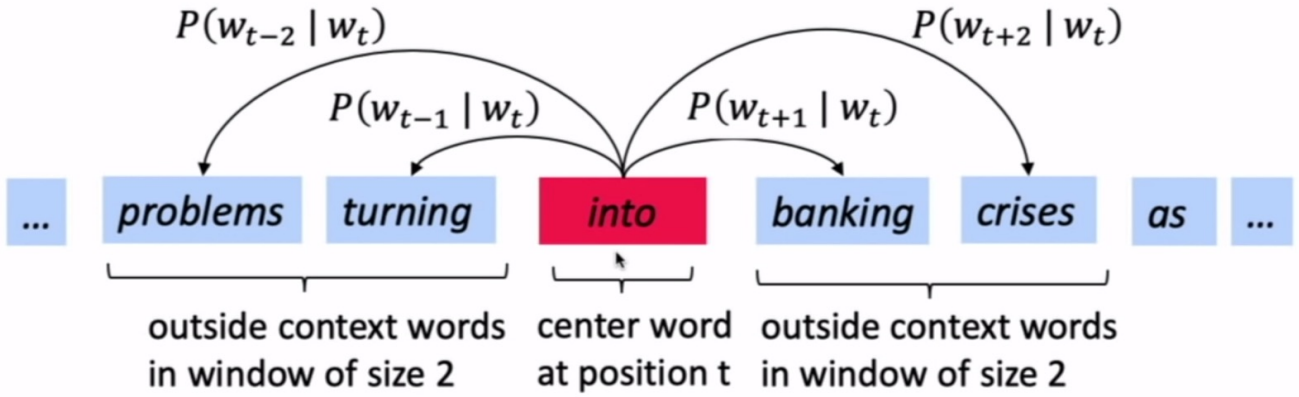
빈칸에 어떤 단어가 들어갈 수 있을까?



빈칸에 들어가기 적합한 단어들과 부적합한 단어들

Word2Vec Overview

- Example windows and process for computing $P(w_{t+j} \mid w_t)$



출처: Stanford University NLP lecture2

확률 관점

CBOW 모델이 하는 일 : 맥락을 주면 타깃 단어가 출현할 확률을 출력하는 것

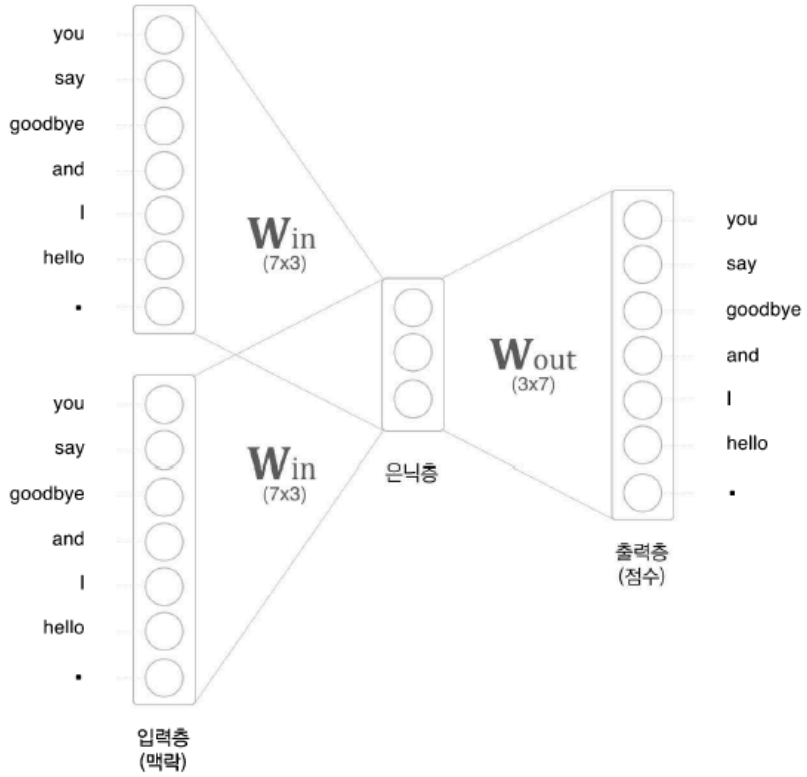
$$w_1 \; w_2 \; \cdots\cdots \; w_{t-1} \; \boxed{w_t} \; w_{t+1} \; \cdots\cdots \; w_{T-1} \; w_T$$

- 맥락 W t-1 과 W t+1 로주어졌을 때 타깃이 Wt가 될 확률

★ $P(w_t \mid w_{t-1}, w_{t+1})$

- 음의 로그 가능도 (negative log likelihood)

$$L = -\frac{1}{T}\sum_{t=1}^{T} \log P(w_t \mid w_{t-1}, w_{t+1})$$



이때의 가중치 매개변수가 우리가 얻고자 하는 단어의 분산 표현!

단어의 분산 표현을 생성하는 두가지 아키텍쳐 모델
CBOW 와 skip grams

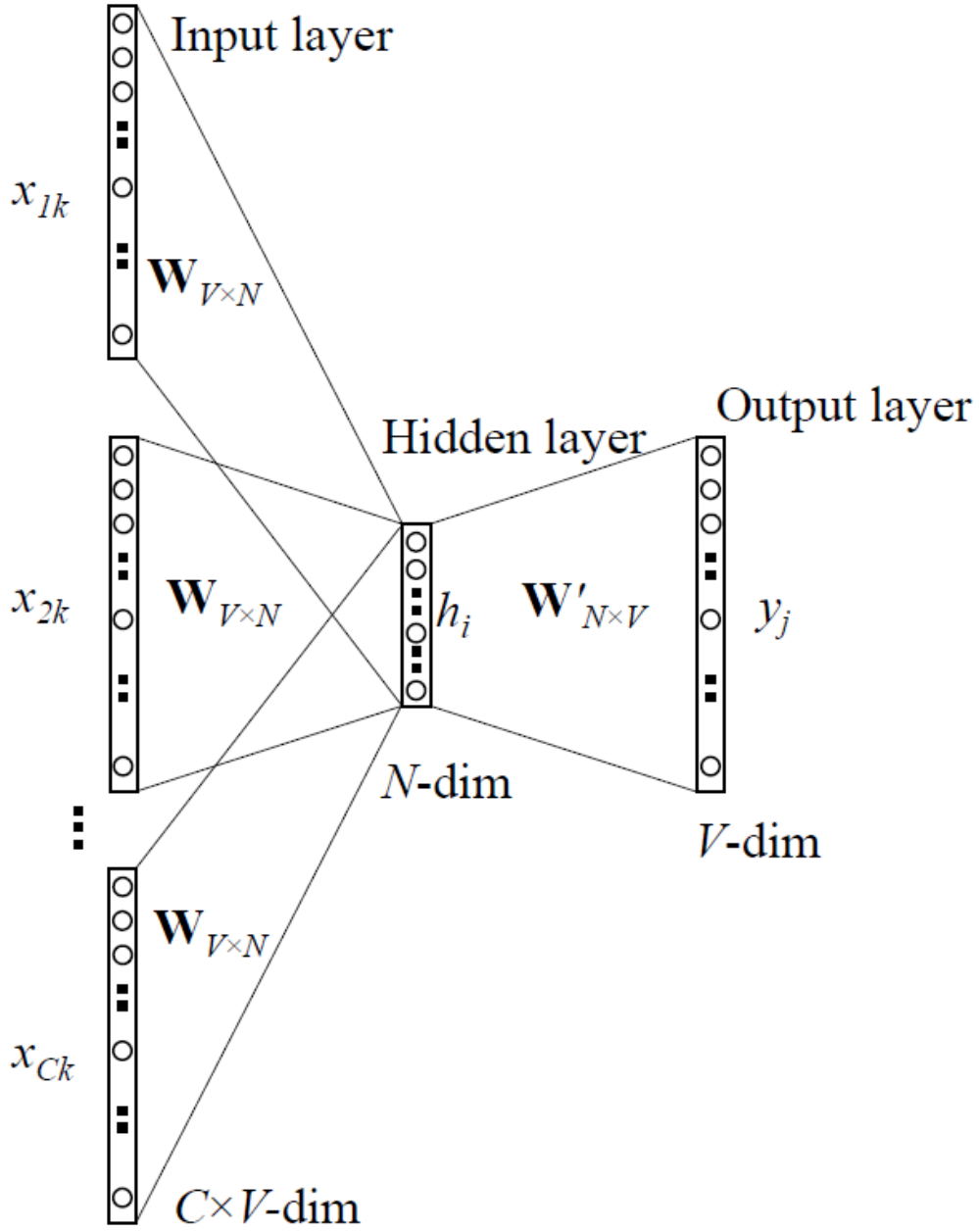
1. continuous bag-of-words architecture

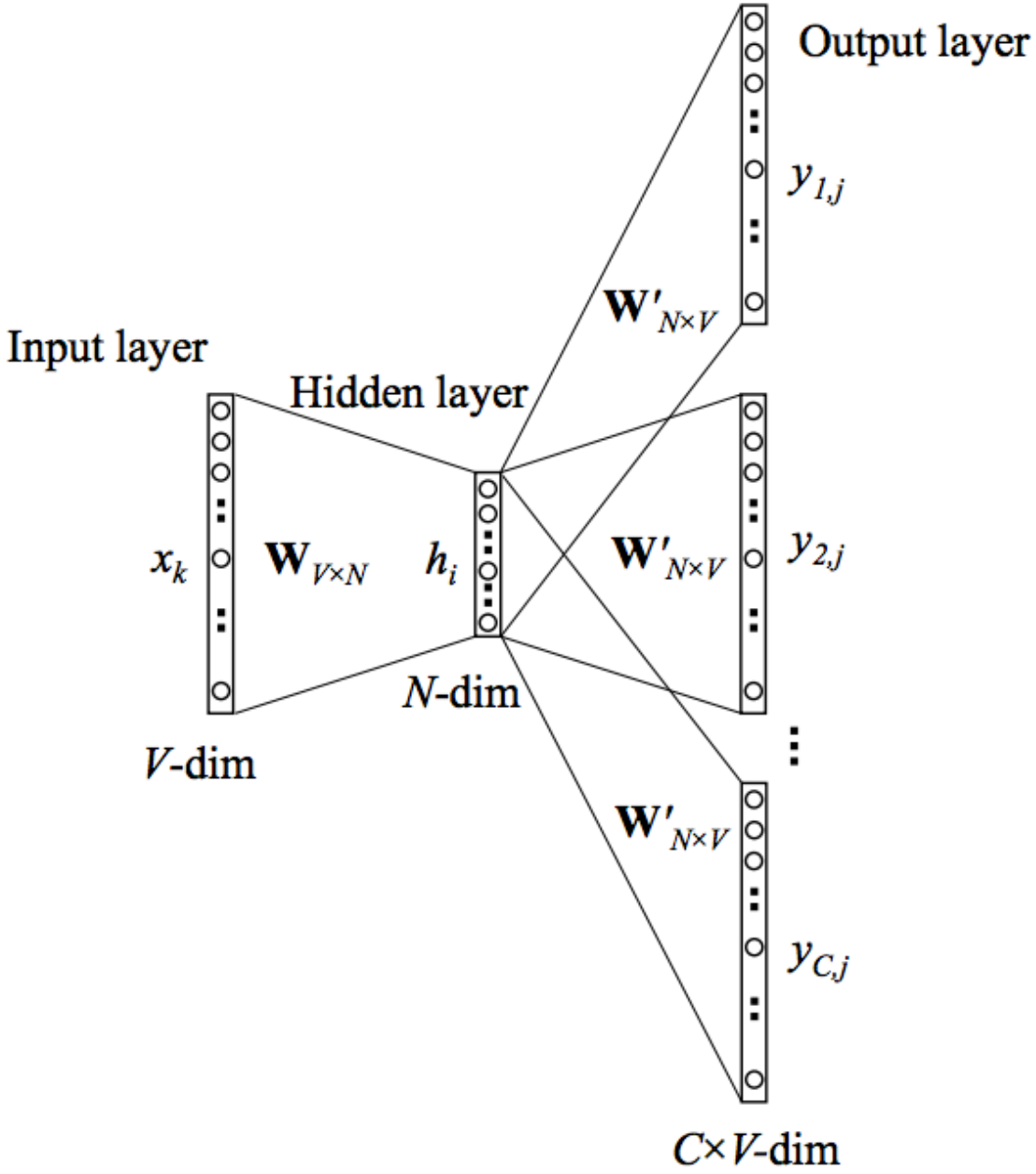
: predicts the current word from a window of surrounding context words

2.skip grams architecture

: uses the current word to predict the surrounding window of context words

CBOW Architecture

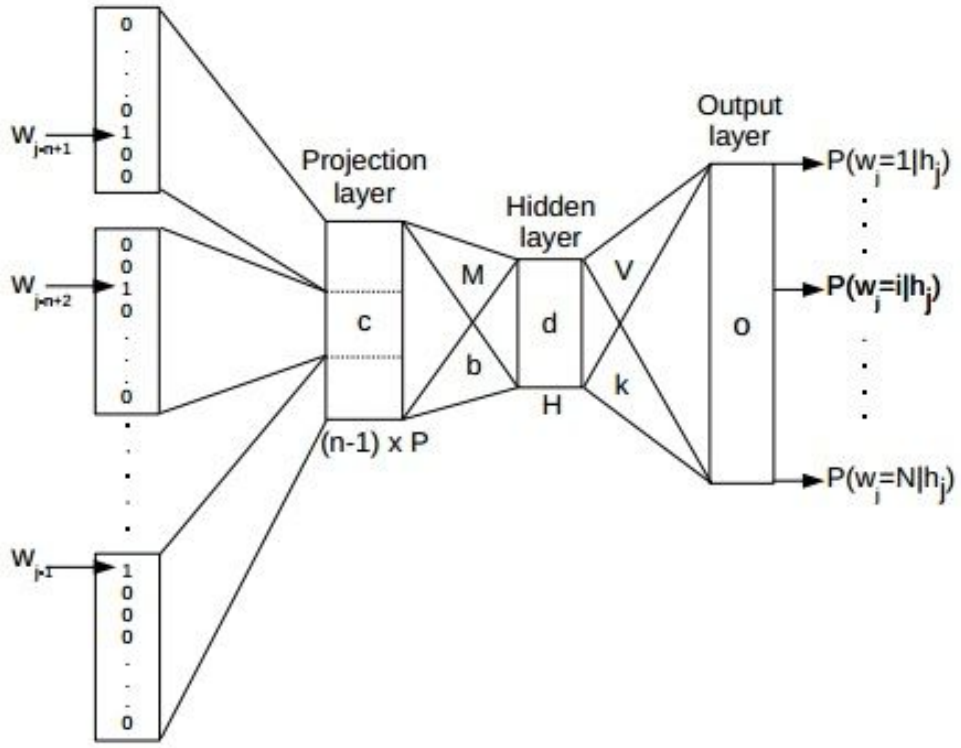




Skip-gram Architecture

모델의 계산복잡도 비교

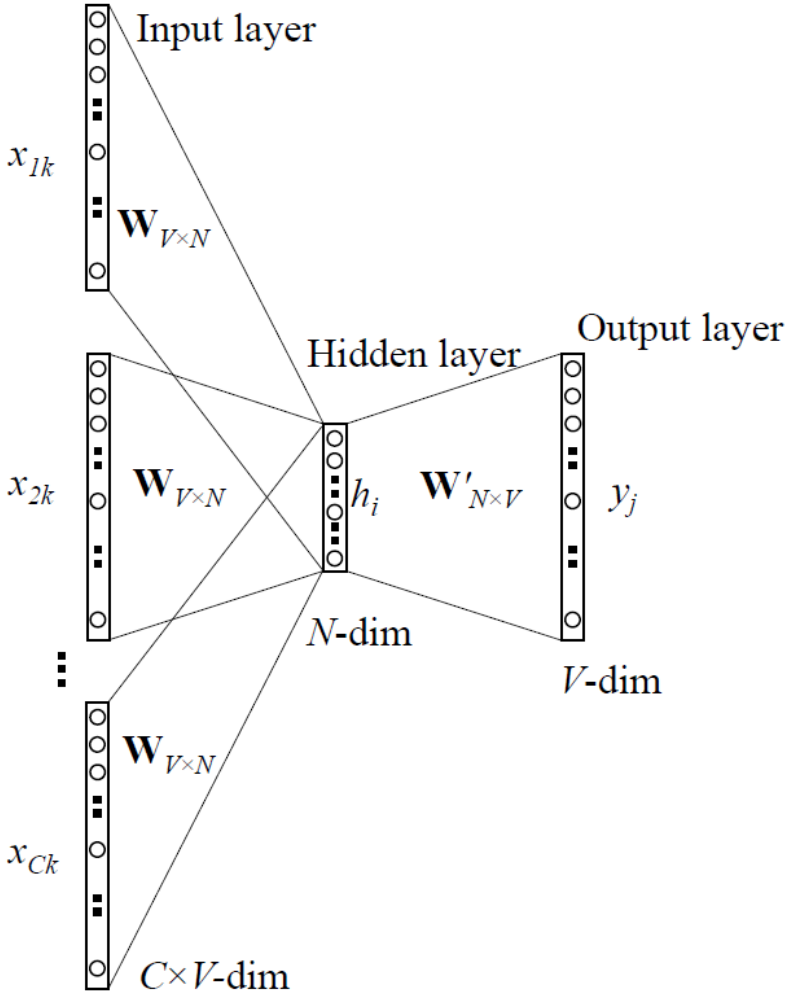
Feed-Forward Neural Net Language Model (NNLM)



- 입력층 (단어들 Projection) - N x P
- 은닉층 (Projection Layer에서 Hidden Layer로) - N x P x H
- 출력층 (Hidden Layer에서 Output Layer로) - H x V

NxP + NxPxH + HxV

ex) N=10, P=500, H=500 일때 O(HxV) = O(**50억**)



CBOW 모델은?

- 입력층(C개 단어 Projection) - C x N
- 은닉층 (Projection Layer에서 Output Layer) - N x V

CxN + NxV
(V를 ln V로 -> **CxN + N x lnV**)

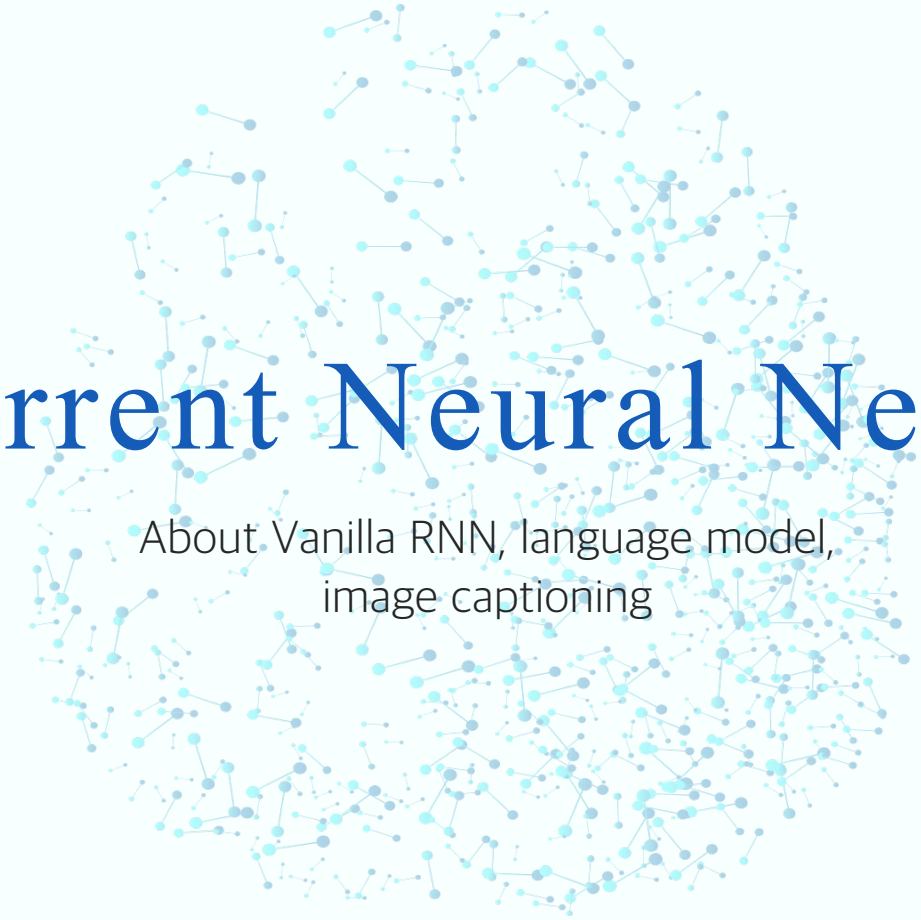
ex) C=10, N=500, V=1,000,000으로 잡아도
O(500 x (10+ln(1,000,000))) = 약 (**10000**)

Next

RNN

Reference

밑바닥부터 시작하는 딥러닝2
Stanford University Lecture 2 | Word Vector Representations: word2vec
Tensorflow.org | Vector representations of words
beomsu kim의 블로그 'word2vec 관련 이론정리'
Dreamgonfly의 블로그 '쉽게 씌여진 word2vec'
于 航의 블로그 https://zhuanlan.zhihu.com/p/43736169



Recurrent Neural Network

About Vanilla RNN, language model,
image captioning

학부연구생 김지현

Overview

Recurrent Neural Networks

of Natural Language Processing

RNN 개요

- 기존의 신경망과 비교
- RNN Computation
- 언어 모델
  (Character level language model)

구현(순전파)

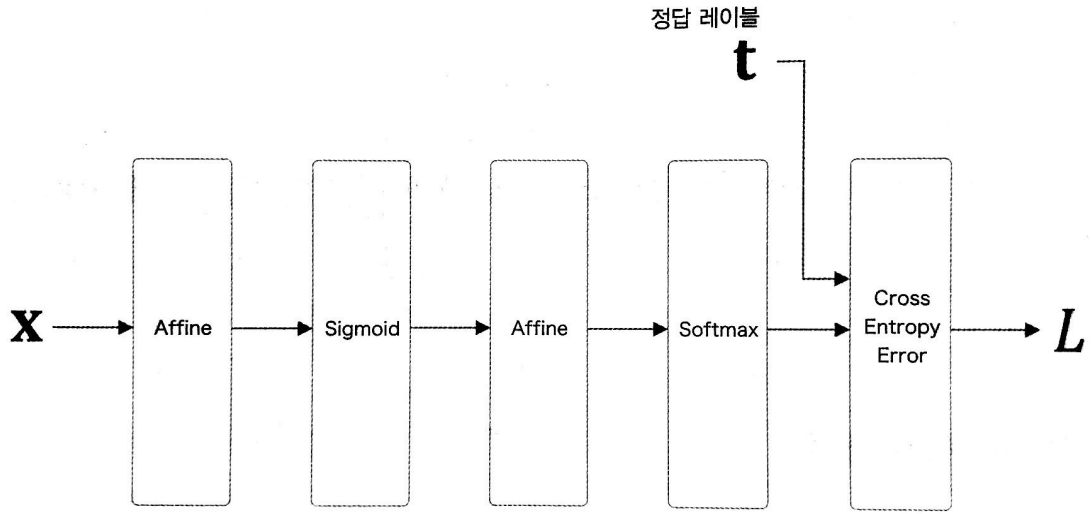
- 밑바닥 / API
- tanh 활성화 함수
- LSTM / GRU ?

image captioning

RNN 개요

기존 신경망?

- feed forward 신경망 (입력층 -> 출력층 한 방향)
- only fixed sized input and output
- 정해진 computational steps (고정된 layer수)



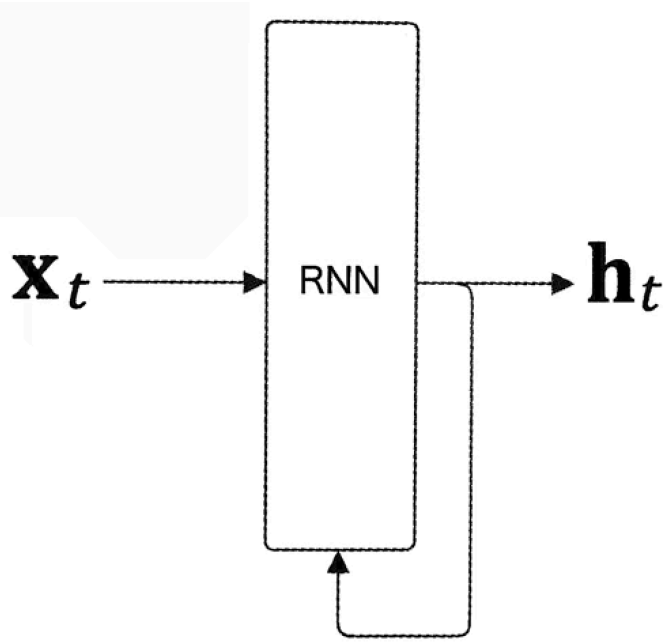
sequence data?

우리의 언어는 지속성(persistence)을 갖고 있다

지속성을 지닌 데이터, 즉 **시계열 데이터를 처리할 수 없다**는 큰 단점

RNN 개요

Recurrent : '순환하다'

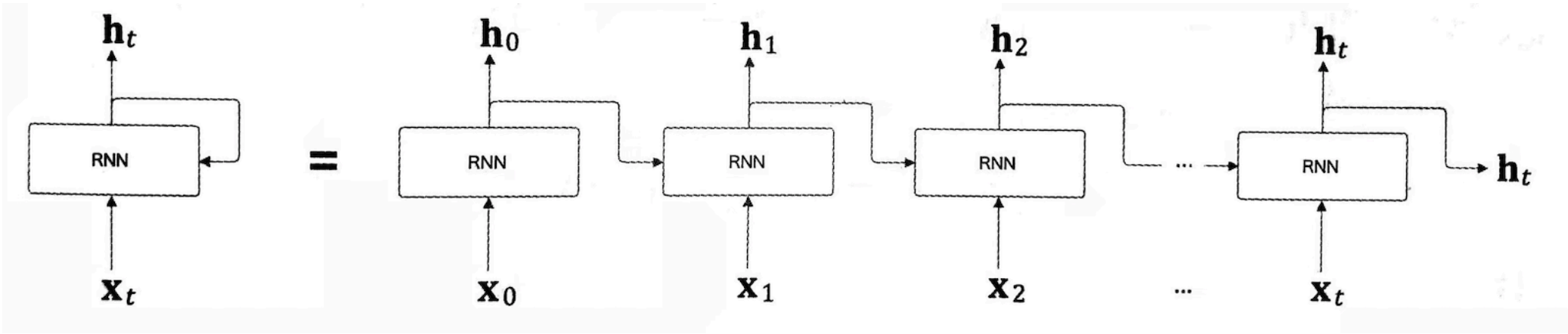


- 시계열 데이터(x0, x1, x2, ⋯, xt)가 RNN 계층에 입력
- 출력이 2개로 분기, 즉 같은 것이 복제되어 분기된 출력이 자기 자신에 입력
- 즉 데이터를 계층 안에서 '순환'시킬 수 있다

순환하는 경로(loop)를 포함하는 RNN 계층

xt: 단어의 분산 표현(word vector)
ht: 은닉 상태 벡터(hidden state vector)

순환 신경망을 펼쳐보면?

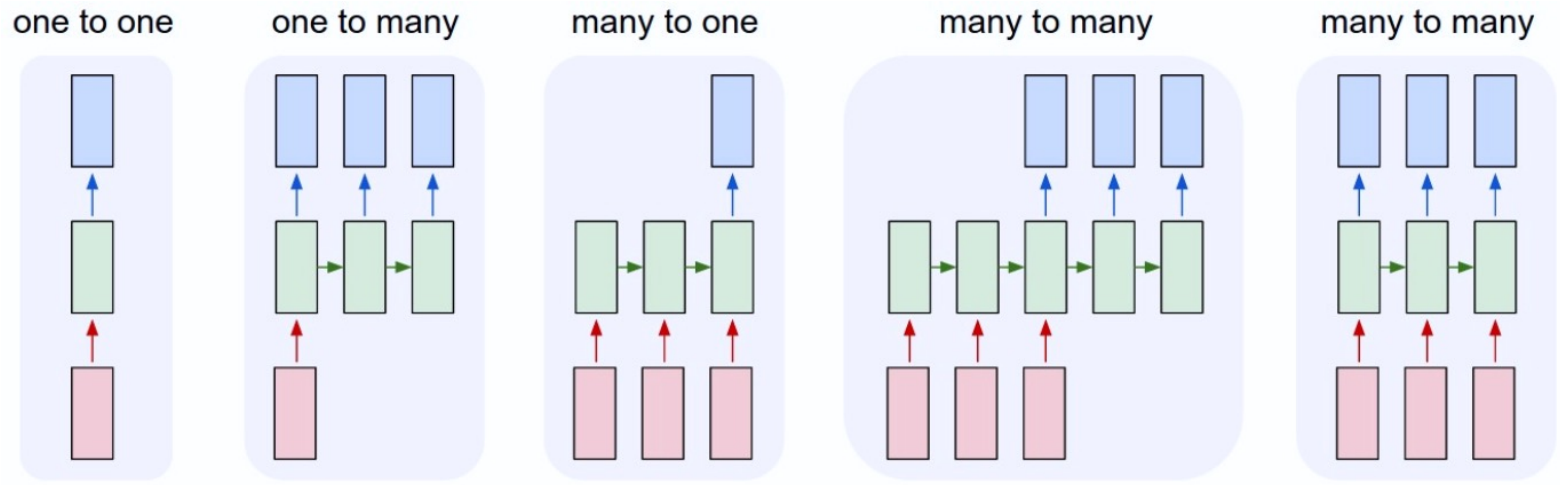


같은 네트워크가 여러 번 복제되어 successor에게 정보 전달하는 체인처럼 연결된 신경망

순환 신경망은 **연속적인**(sequential) **시계열 데이터**(time series) 데이터에 적합한 모델

RNN이 활용되는 분야들

- speech recognition
- language modeling
- translation
- image captioning
- etc



This sentence is sequence of words...

t = 1 t = 2 t = 3 t = 4 . . .

고정된 길이의 벡터가 아닌 연속적인 벡터들을 얼마든지 입력으로 받을 수 있다는 장점

RNN computation

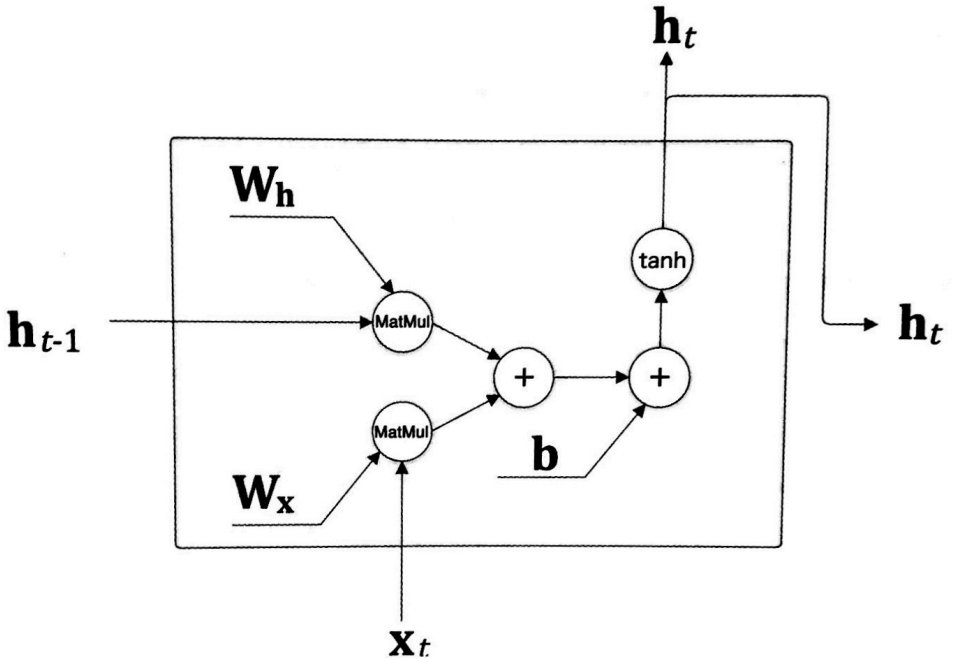
순전파 수식

$$\mathbf{h}_t = \tanh(\mathbf{h}_{t-1}\mathbf{W}_\mathbf{h} + \mathbf{x}_t\mathbf{W}_\mathbf{x} + \mathbf{b})$$

ht : 은닉 상태(hidden state)

가중치 2개

Wx : 입력 x과 행렬 곱
Wh : 이전 RNN 출력과 행렬 곱



언어 모델

언어를 '**확률(Probability)**'로 다루는 언어 모델

$$P(\boldsymbol{x}^{(1)}, \ldots, \boldsymbol{x}^{(T)}) = P(\boldsymbol{x}^{(1)}) \times P(\boldsymbol{x}^{(2)} | \boldsymbol{x}^{(1)}) \times \cdots \times P(\boldsymbol{x}^{(T)} | \boldsymbol{x}^{(T-1)}, \ldots, \boldsymbol{x}^{(1)})$$
$$= \prod_{t=1}^{T} P(\boldsymbol{x}^{(t)} | \boldsymbol{x}^{(t-1)}, \ldots, \boldsymbol{x}^{(1)})$$

간단한 예시)

h e l l o

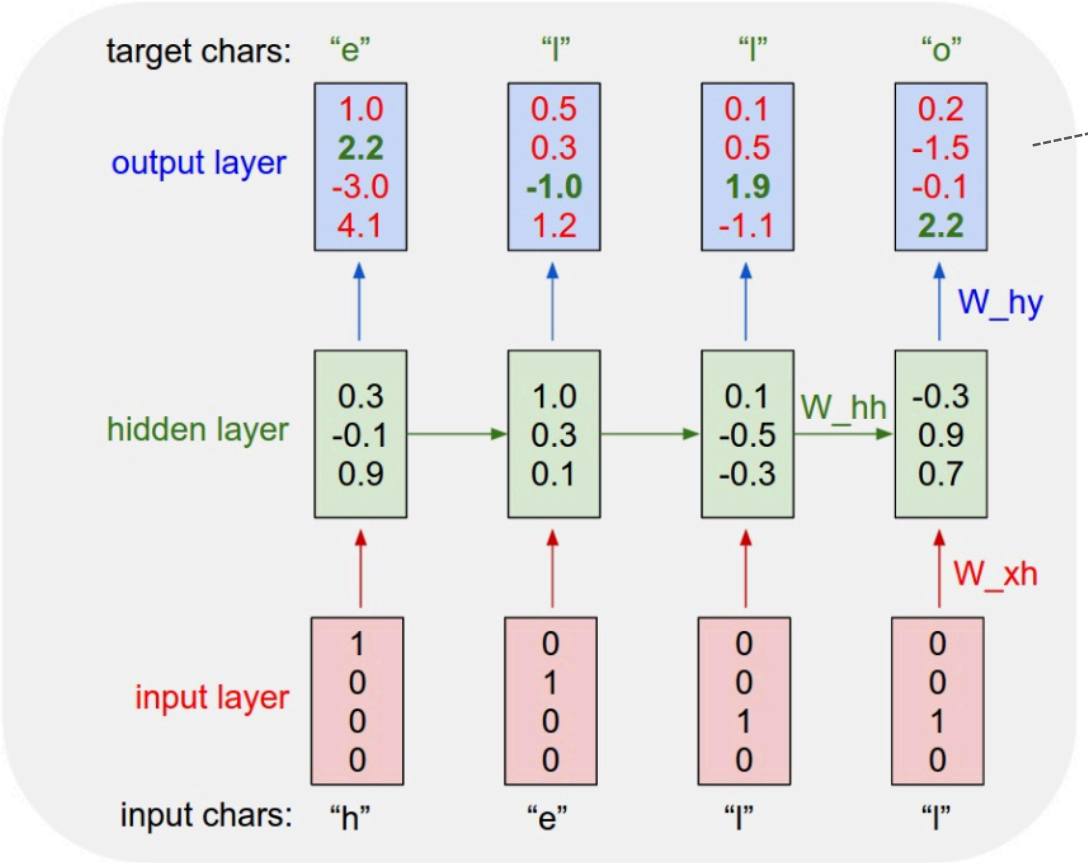
4가지 학습 예시

"h" 뒤에 - "e"
"he" 뒤에 - "l"
"hel" 뒤에 - "l"
"hell" 뒤에 - "o"

} 일 확률이 높도록 학습

언어 모델

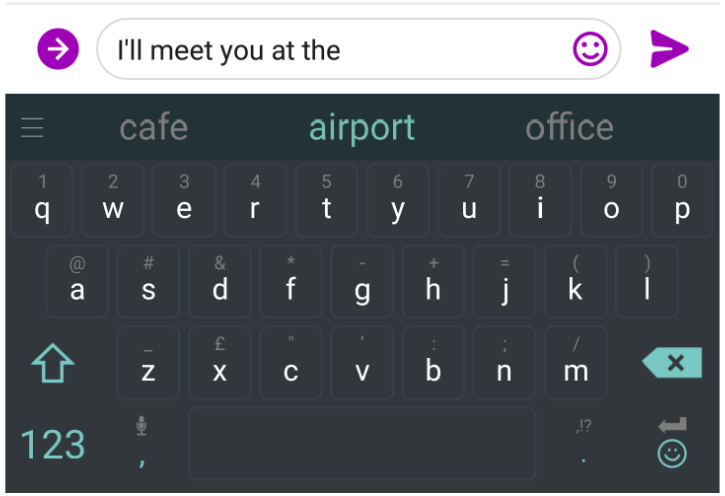
Character-Level Language Models

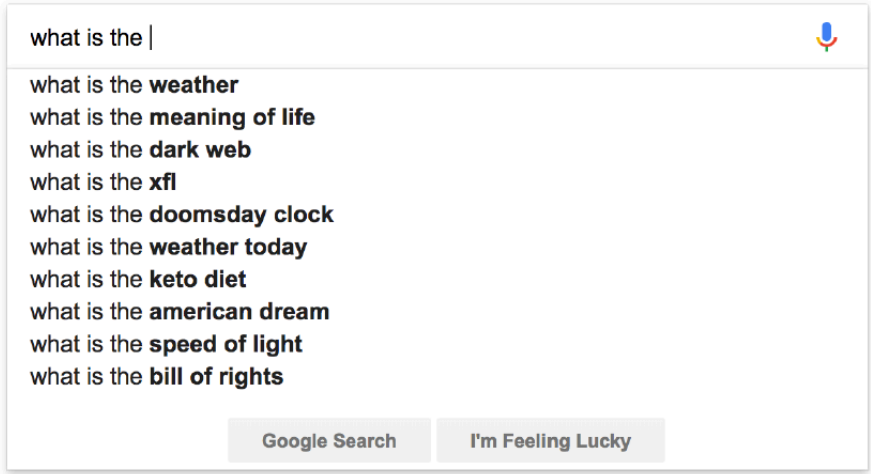


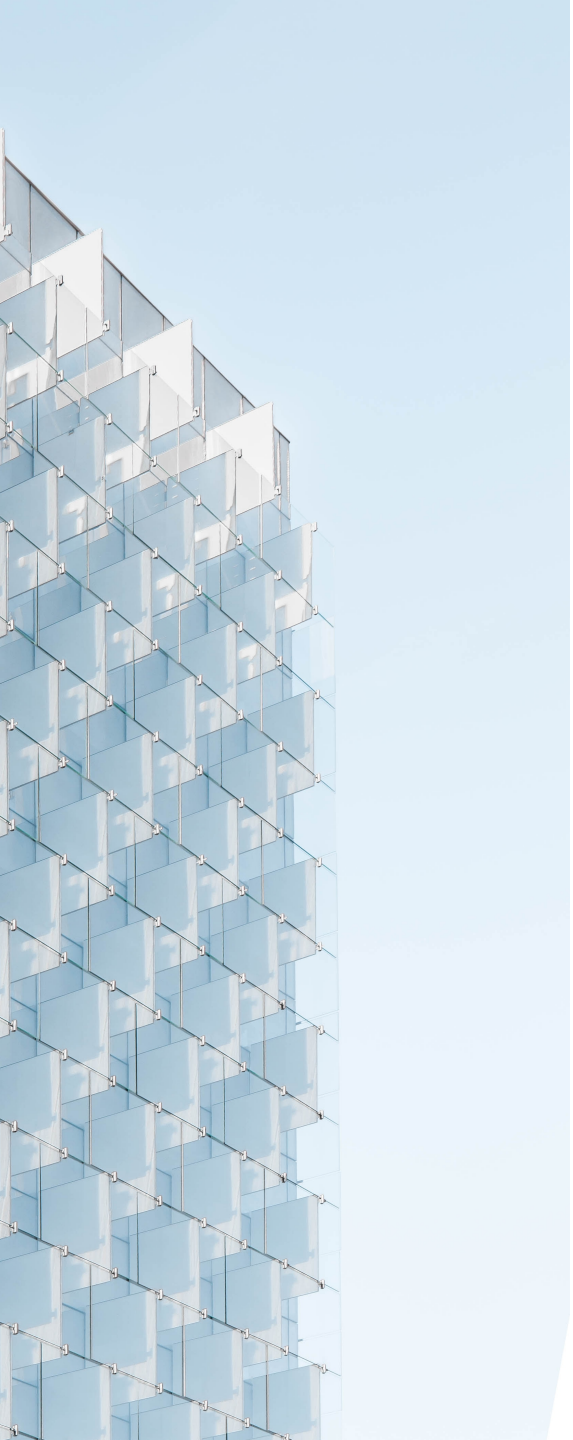
원하는 것
confidence 증가
confidence 감소

An example RNN with 4-dimensional input and output layers, and a hidden layer of 3 units (neurons). This diagram shows the activations in the forward pass when the RNN is fed the characters "hell" as input. The output layer contains confidences the RNN assigns for the next character (vocabulary is "h,e,l,o"); We want the green numbers to be high and red numbers to be low.

You use Language Models every day!

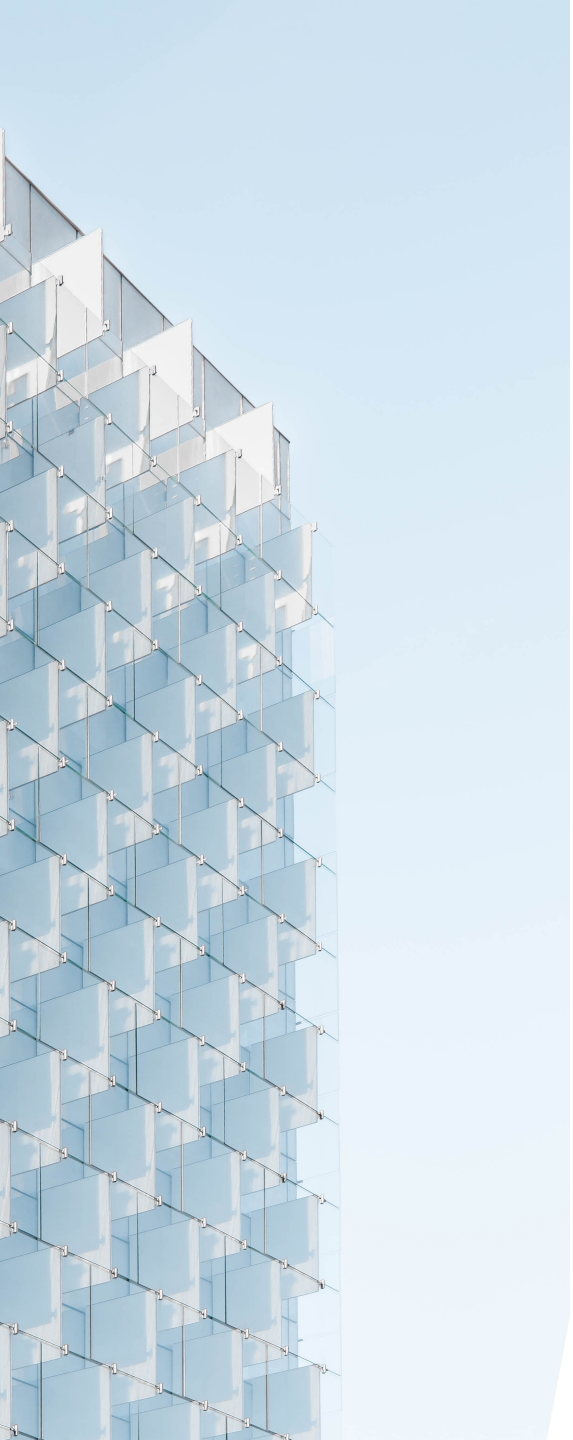






Vanilla RNN (순전파) 구현


```python
class RNN:
    def __init__(self, Wx, Wh, b):
        self.params = [Wx, Wh, b]
        self.grads = [np.zeros_like(Wx), np.zeros_like(Wh), np.zeros_like(b)]
        self.cache = None

    def forward(self, x, h_prev):
        Wx, Wh, b = self.params
        t = np.matmul(h_prev, Wh) + np.matmul(x, Wx) + b
        h_next = np.tanh(t)

        self.cache = (x, h_prev, h_next)
        return h_next

    def backward(self, dh_next):
        Wx, Wh, b = self.params
        x, h_prev, h_next = self.cache

        dt = dh_next * (1 - h_next ** 2)
        db = np.sum(dt, axis=0)
        dWh = np.matmul(h_prev.T, dt)
        dh_prev = np.matmul(dt, Wh.T)
        dWx = np.matmul(x.T, dt)
        dx = np.matmul(dt, Wx.T)

        self.grads[0][...] = dWx
        self.grads[1][...] = dWh
        self.grads[2][...] = db

        return dx, dh_prev
```



Vanilla RNN (순전파) 구현


```
rnn = RNN()
y = rnn.step(x) # x is an input vector, y is the RNN's output vector
```

```
class RNN:
  # ...
  def step(self, x):
    # update the hidden state
    self.h = np.tanh(np.dot(self.W_hh, self.h) + np.dot(self.W_xh, x))
    # compute the output vector
    y = np.dot(self.W_hy, self.h)
    return y
```

```
y1 = rnn1.step(x)
y = rnn2.step(y1)
```

참고)

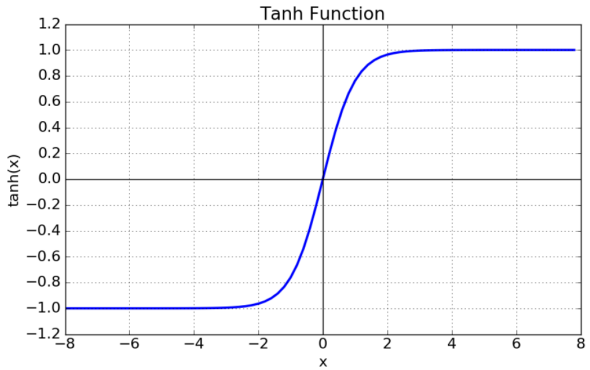
tanh 함수?

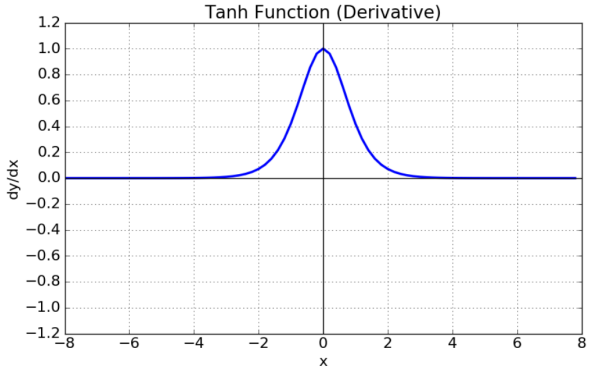
Hyperbolic tangent function

$$tanh(x) = 2\sigma(2x) - 1$$

$$tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$$

$$tanh'(x) = 1 - tanh^2(x)$$





함수의 중심값을 0으로 옮겨 시그모이드의 최적화 과정이 느려지는 문제 해결

하지만 gradient vanishing 문제는 여전히 남아있음

그러나 성능?

Tom was watching TV in his room. Mary came into the room.
.....
.....
.....Mary said hi to ____ ?

Vanilla RNN은 **장기**(long term) **의존 관계**를 잘 학습할 수 없다

Long-Short Term Memory (LSTM) & Gated Recurrent Unit (GRU)

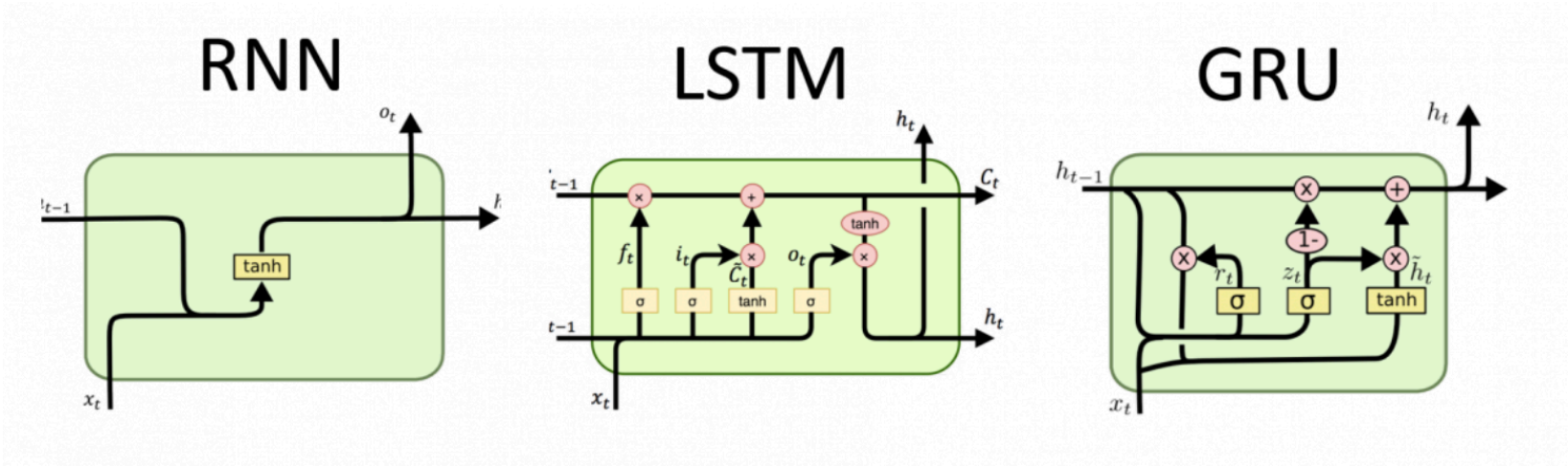


사진 출처: dProgrammer lopez님의 블로그

image captioning

"girl in pink dress is jumping in air."

"black and white dog jumps over bar."

"young girl in pink shirt is swinging on swing."

"man in blue wetsuit is surfing on wave."

링크

https://github.com/tensorflow/tensorflow/blob/r1.13/tensorflow/contrib/eager/python/examples/generative_examples/image_captioning_with_attention.ipynb

Reference

밑바닥부터 시작하는 딥러닝2
Stanford CS224N lecture 6| Language Models and RNNs
Andrej Karpathy의 블로그 “The Unreasonable Effectiveness of RNN”
Google AI Blog “Show and ell: image captioning open sourced in Tensorflow”
Tensorflow github “image_captioning_with_attention”