AI School 6기 8주차

Word2Vec 기초이론

Word2Vec 실습

AI School 6기 8주차

Word2Vec 기초이론

One-hot encoding

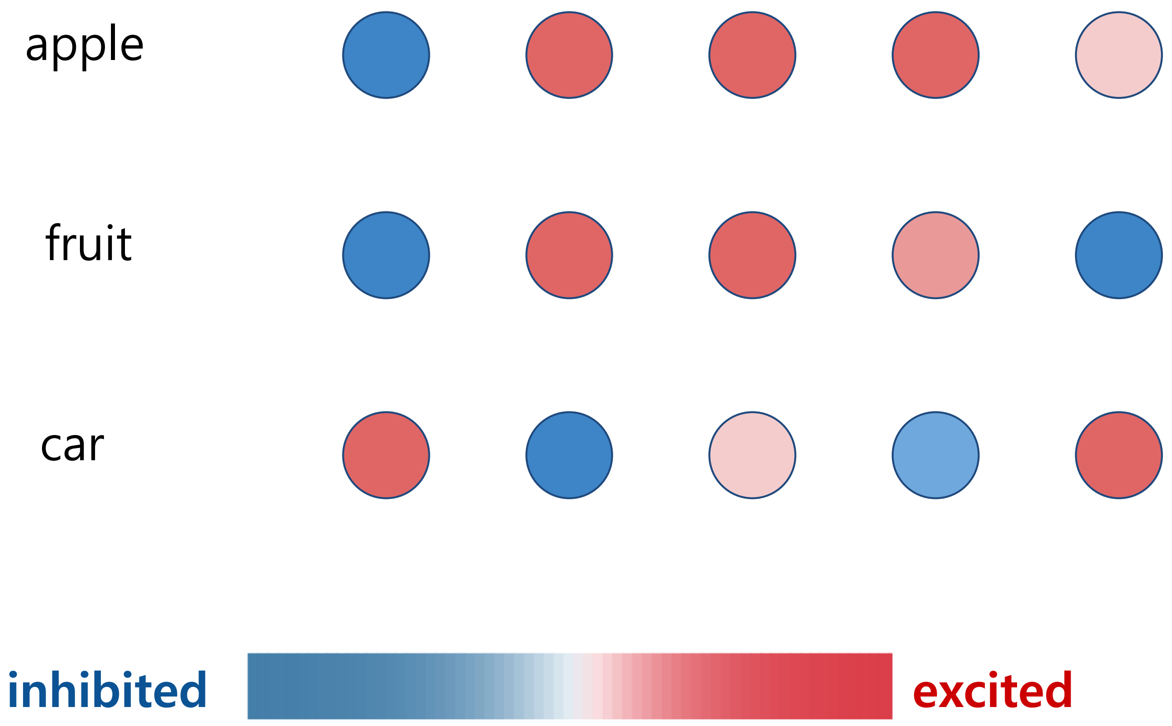
apple = [0 0 0 0 1 0 0 0 … 0 0 0]

fruit = [0 0 0 0 0 0 0 0 … 0 1 0]

car = [0 0 0 1 0 0 0 0 … 0 0 0]

Distributed Representation

- Word is represented as continuous level of activations



Word embedding

- Distributional hyphothesis (Harris et al., 1954)

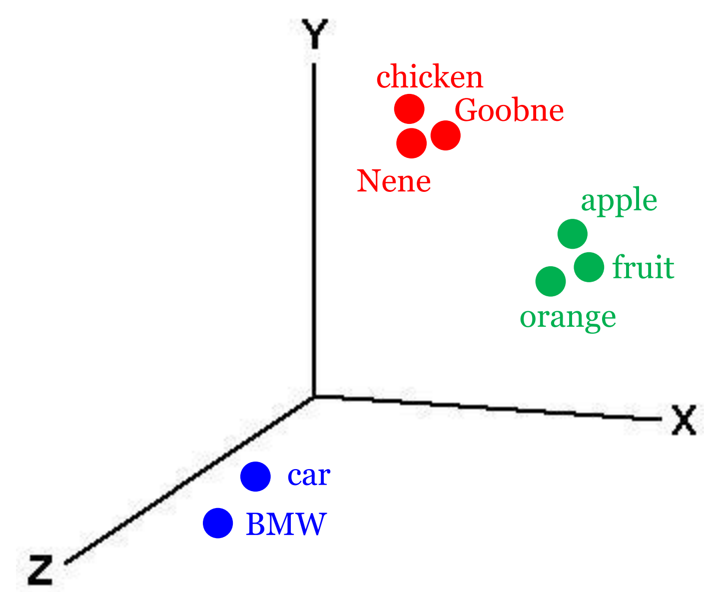
“Words that occur in the same contexts tend to have similar meanings”

I eat an apple every day.

I eat an orange every day.

I eat a chicken every day.

I like driving my car to work.



Word2vec

Two original papers published in association with word2vec by Mikolov et al. (2013)

Efficient Estimation of Word Representations in Vector Space

Tomas Mikolov
Google Inc., Mountain View, CA
tmikolov@google.com

Kai Chen
Google Inc., Mountain View, CA
kaichen@google.com

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

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

Abstract

We propose two novel model architectures for computing continuous vector representations of words from very large data sets. The quality of these representations is measured in a word similarity task, and the results are compared to the previously best performing techniques based on different types of neural networks. We observe large improvements in accuracy at much lower computational cost, i.e. it takes less than a day to learn high quality word vectors from a 1.6 billion words data set. Furthermore, we show that these vectors provide state-of-the-art performance on our test set for measuring syntactic and semantic word similarities.

Citation: 11001

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

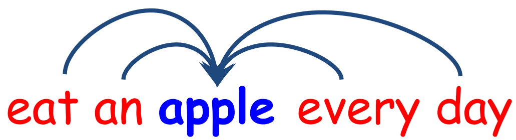
Abstract

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.

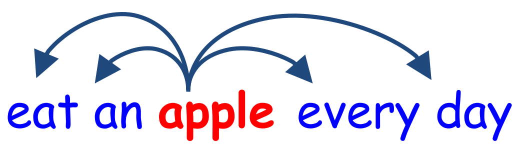
Citation: 13531

Word2vec

Method 1: continuous bag-of-word (CBOW)

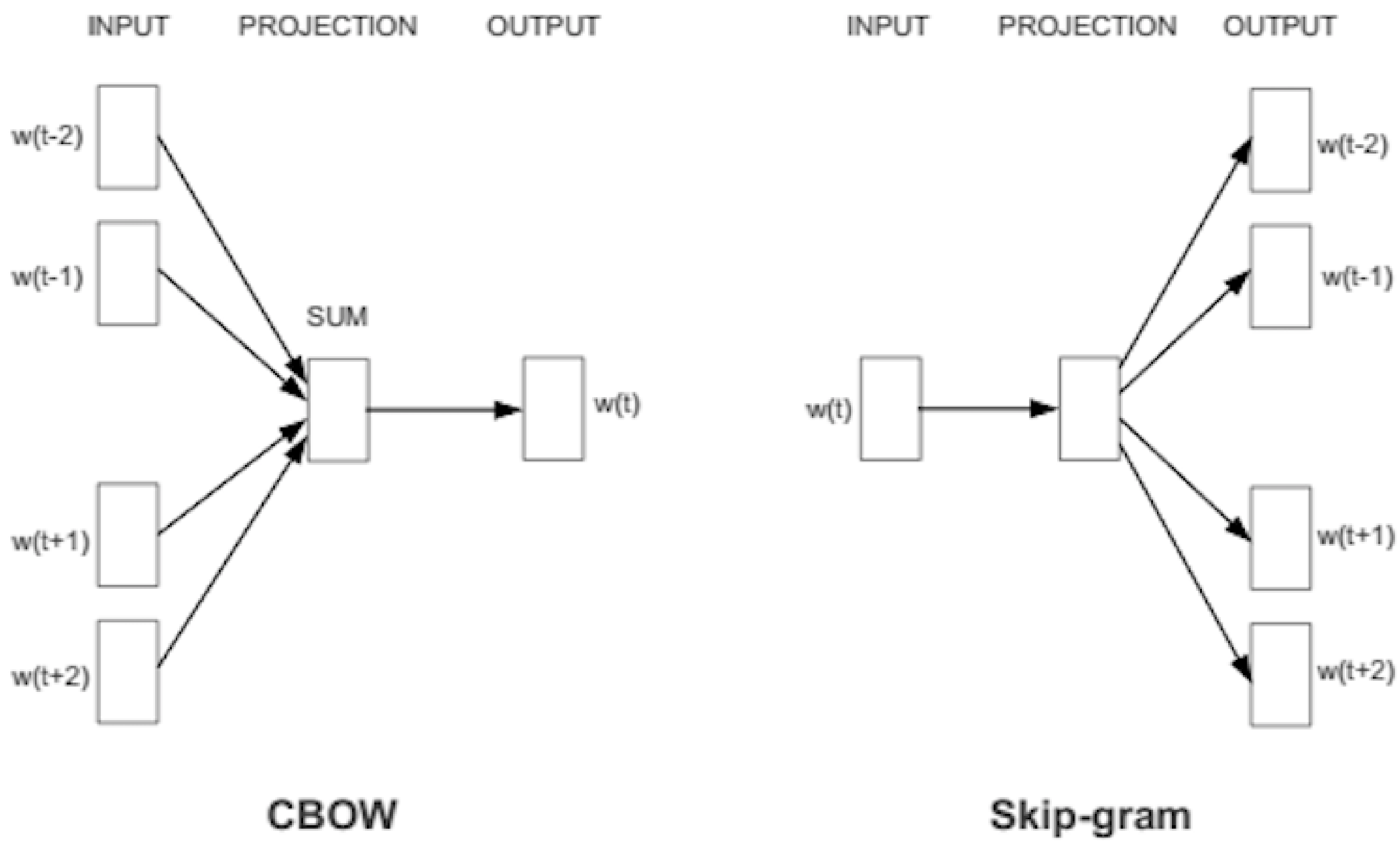


Method 2: skip-gram (SG)



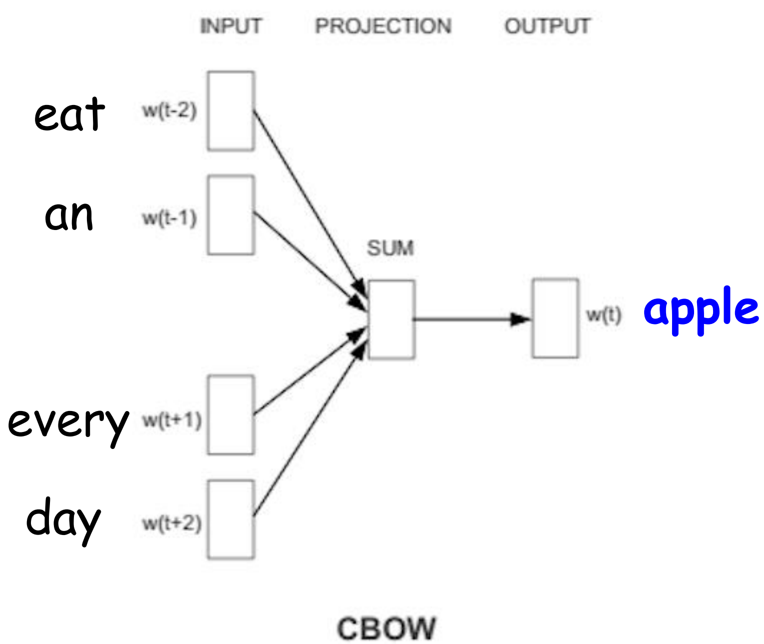
Word2vec

- Connect **words** and their **context**



Word2vec: CBOW

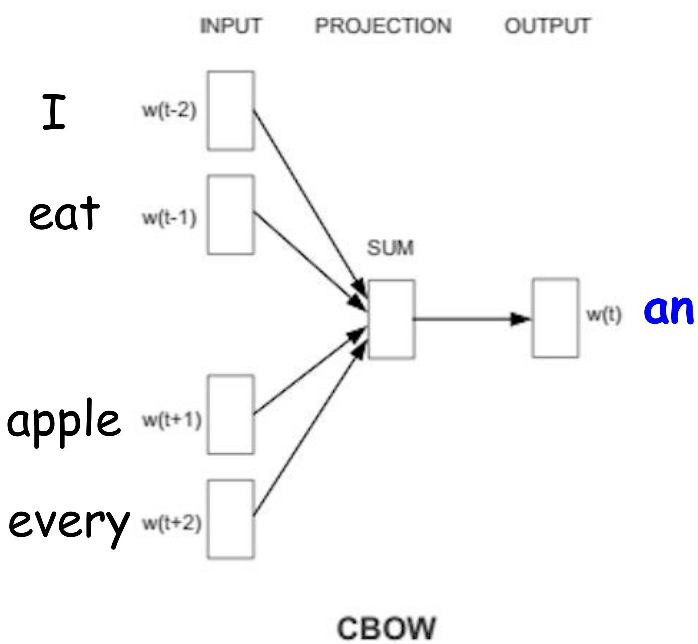
- Predict **target words** based on **context words**



I eat an **apple** every day.

Word2vec: CBOW

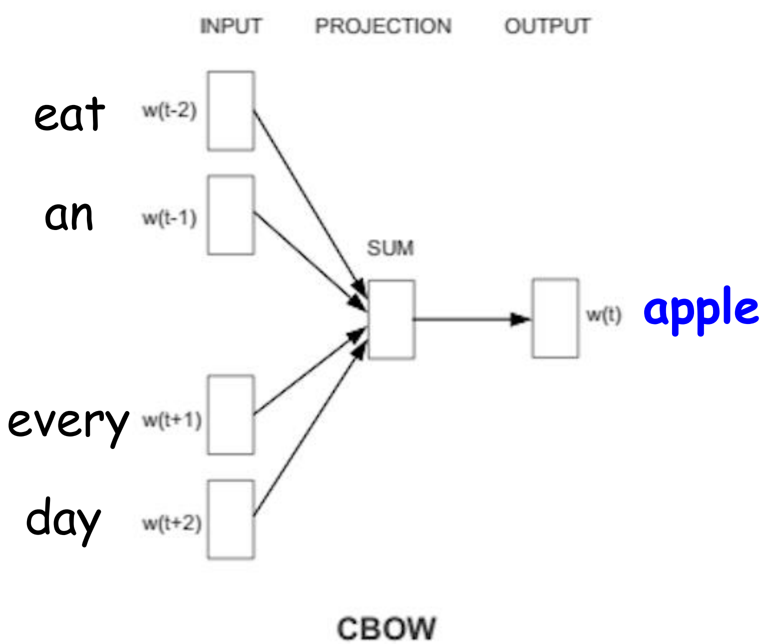
- Predict **target words** based on **context words**



I eat an apple every day.

Word2vec: CBOW

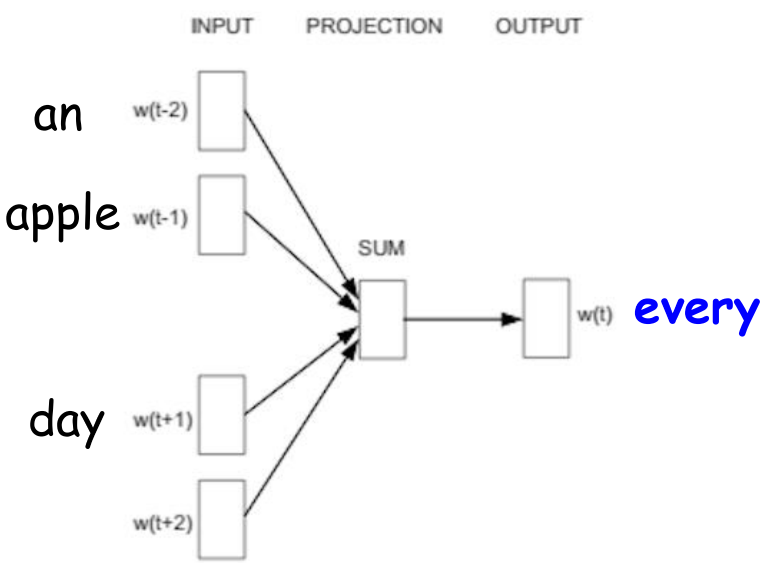
- Predict **target words** based on **context words**



I eat an **apple** every day.

Word2vec: CBOW

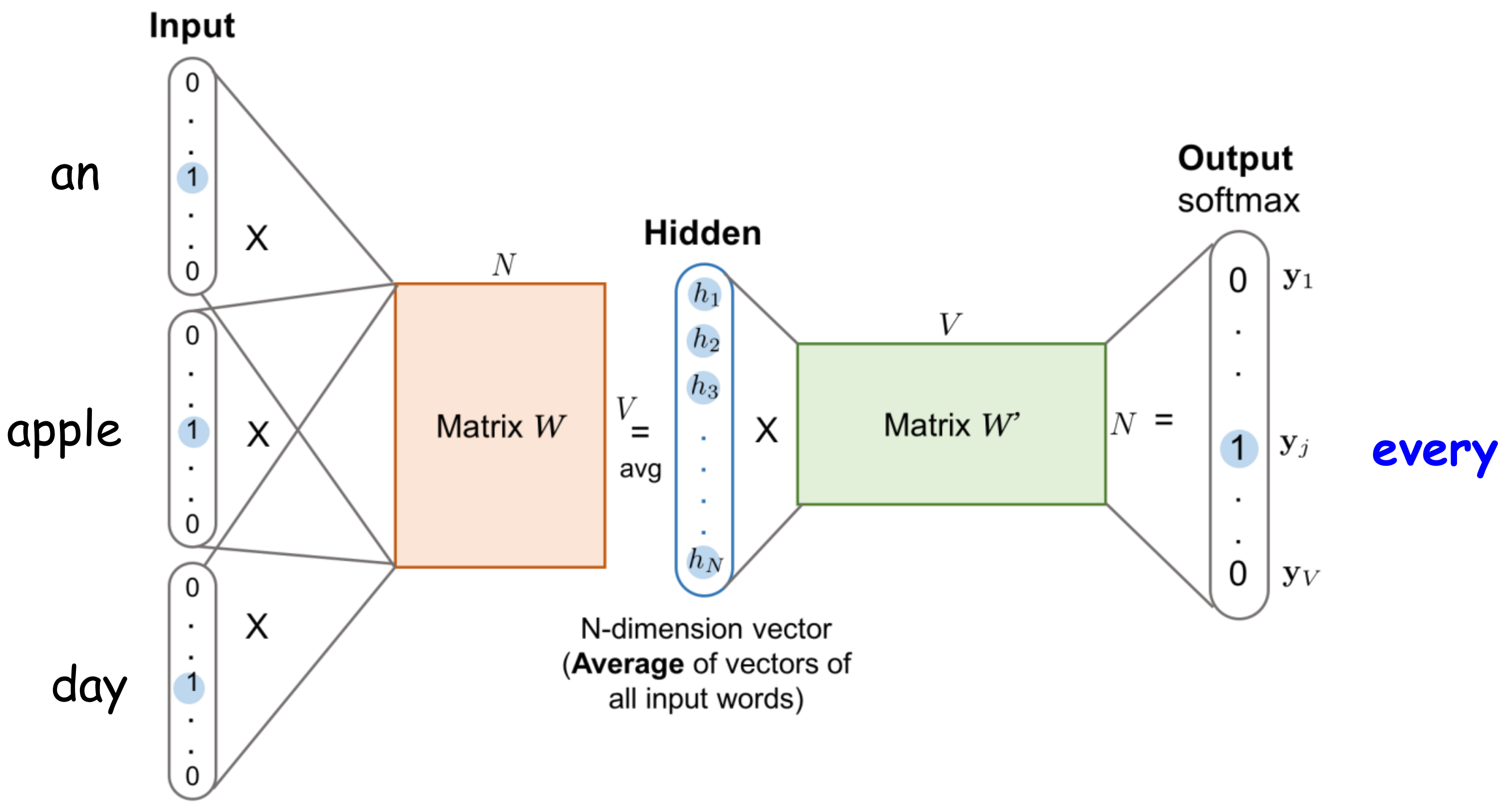
- Predict **target words** based on **context words**



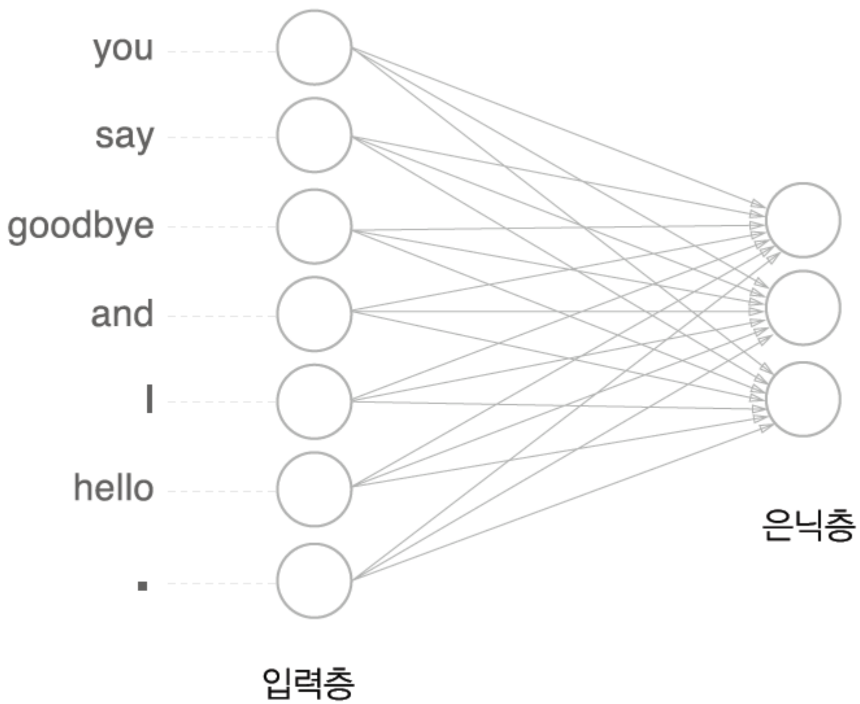
CBOW

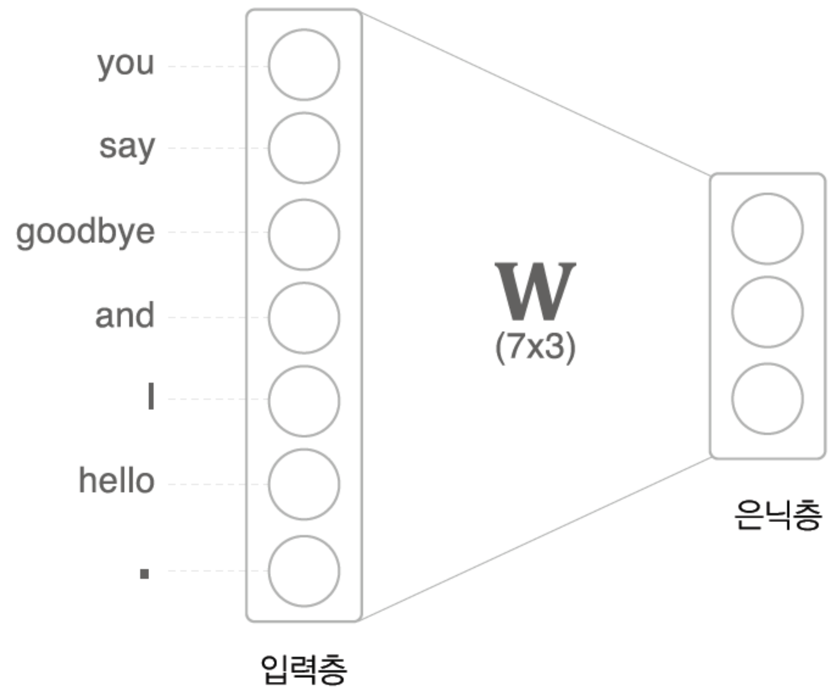
I eat an apple **every** day.

Word2vec: CBOW

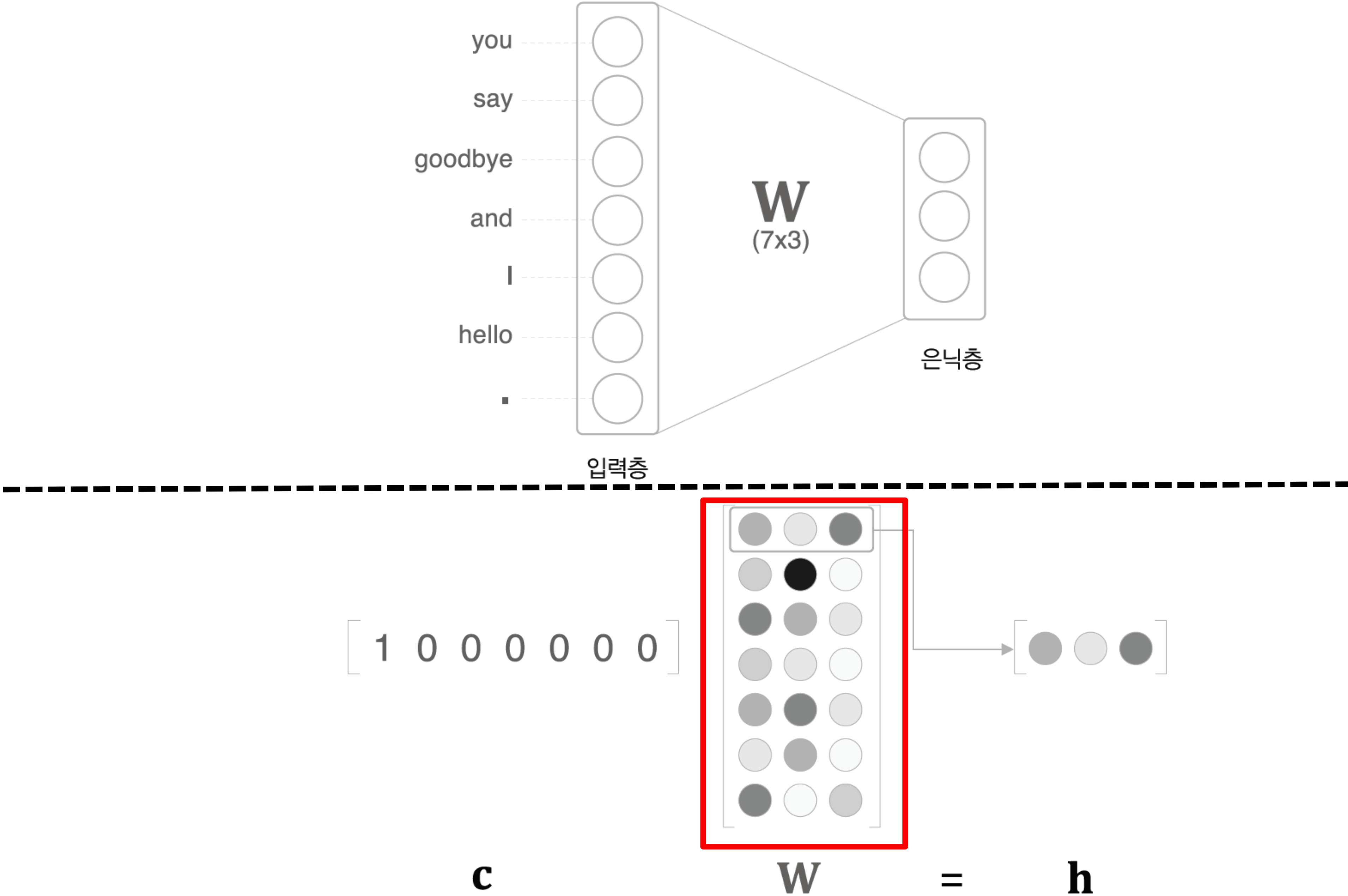


Word2vec: CBOW

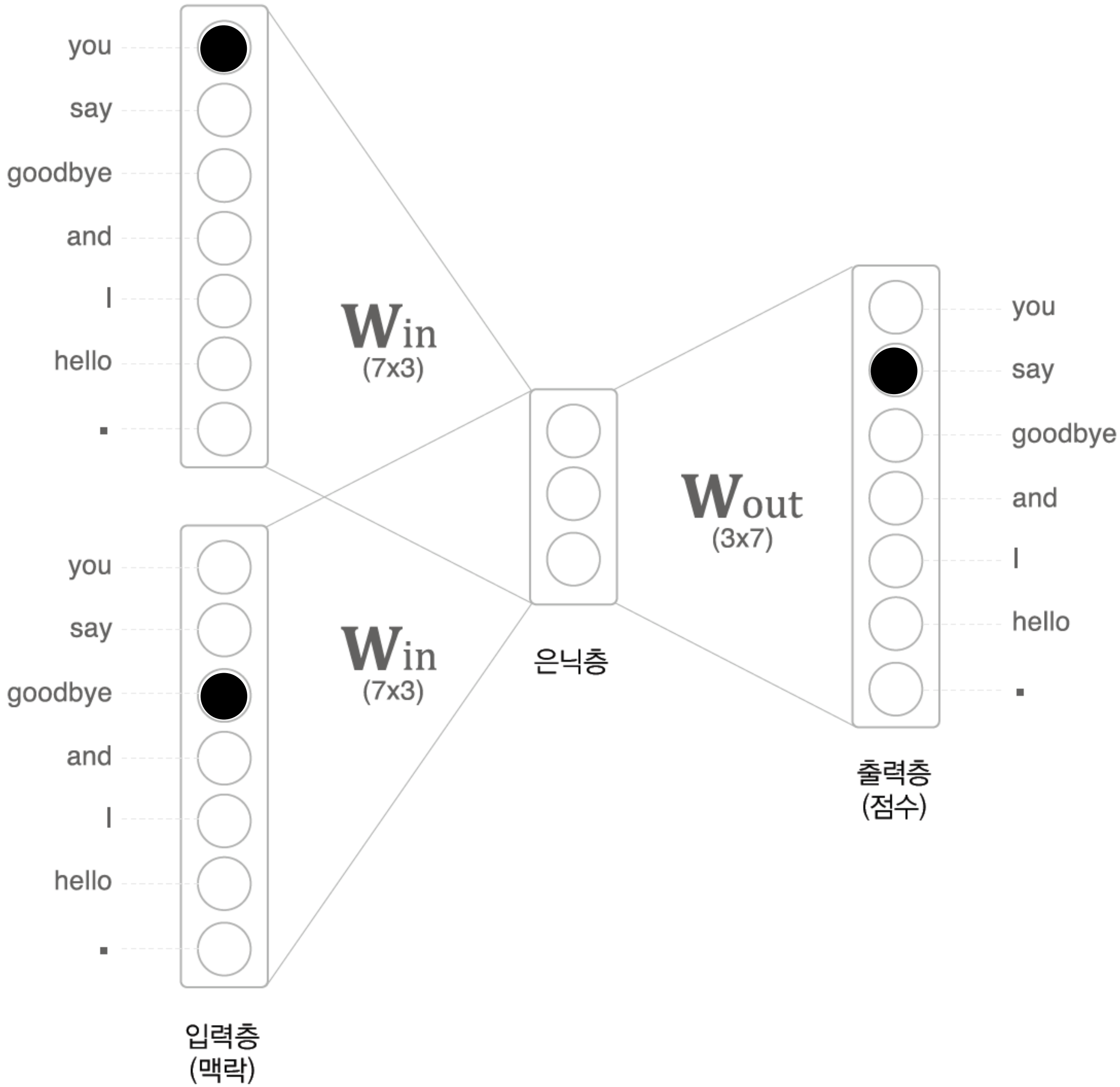




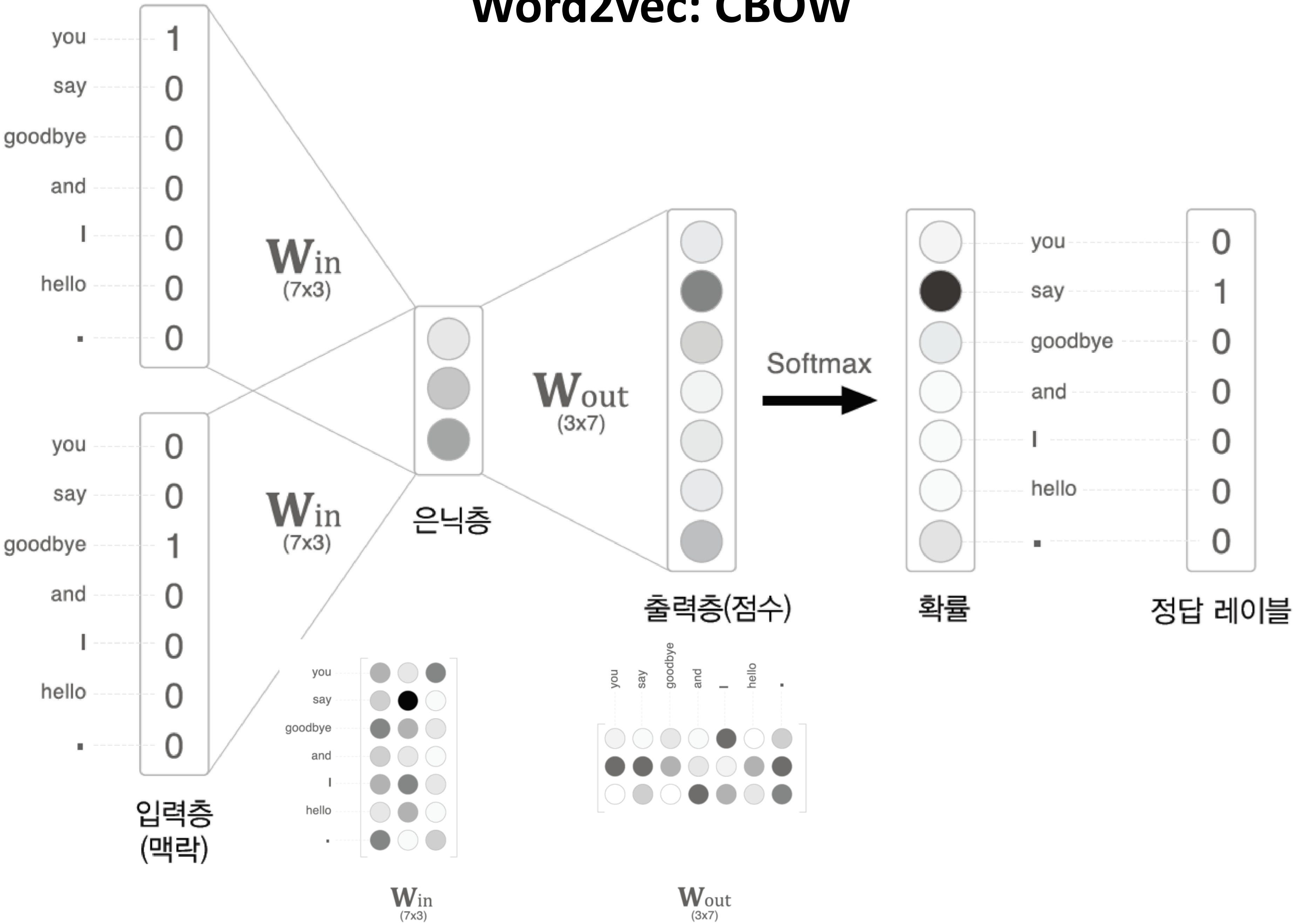
Word2vec: CBOW



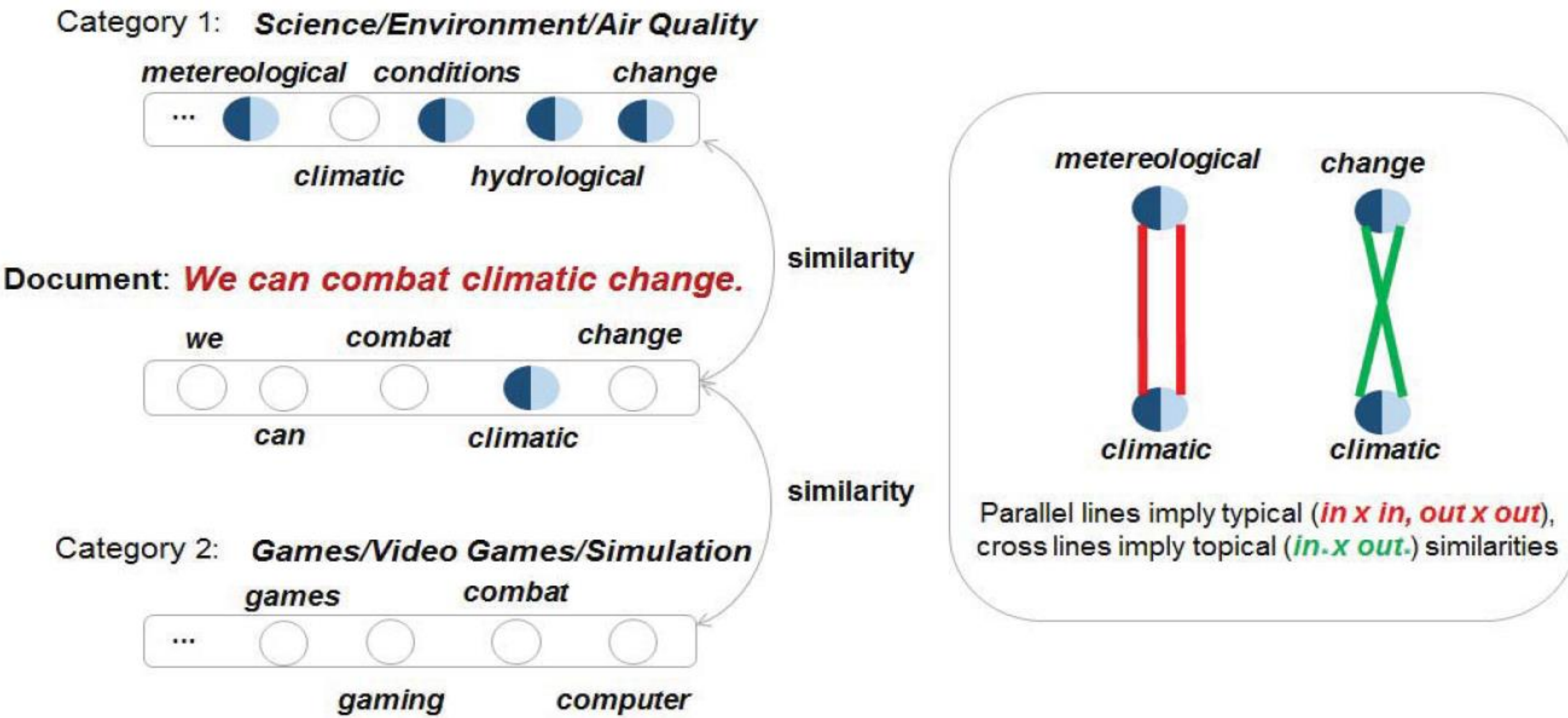
Word2vec: CBOW



Word2vec: CBOW



Dual Word Embeddings

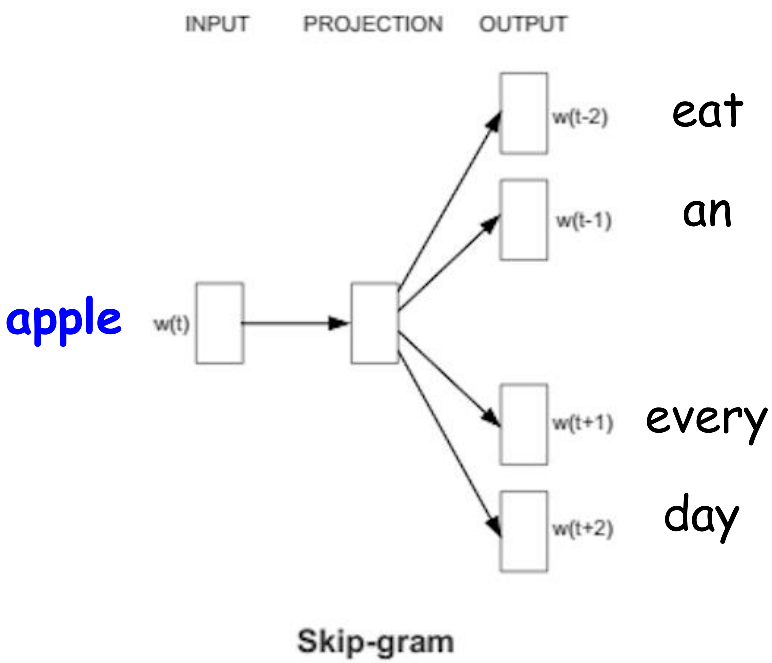


Word2vec: CBOW

말뭉치	맥락(contexts)	타깃
you say goodbye and I say hello .	you, goodbye	say
you say goodbye and I say hello .	say, and	goodbye
you say goodbye and I say hello .	goodbye, I	and
you say goodbye and I say hello .	and, say	I
you say goodbye and I say hello .	I, hello	say
you say goodbye and I say hello .	say, .	hello

Word2vec: Skip-gram

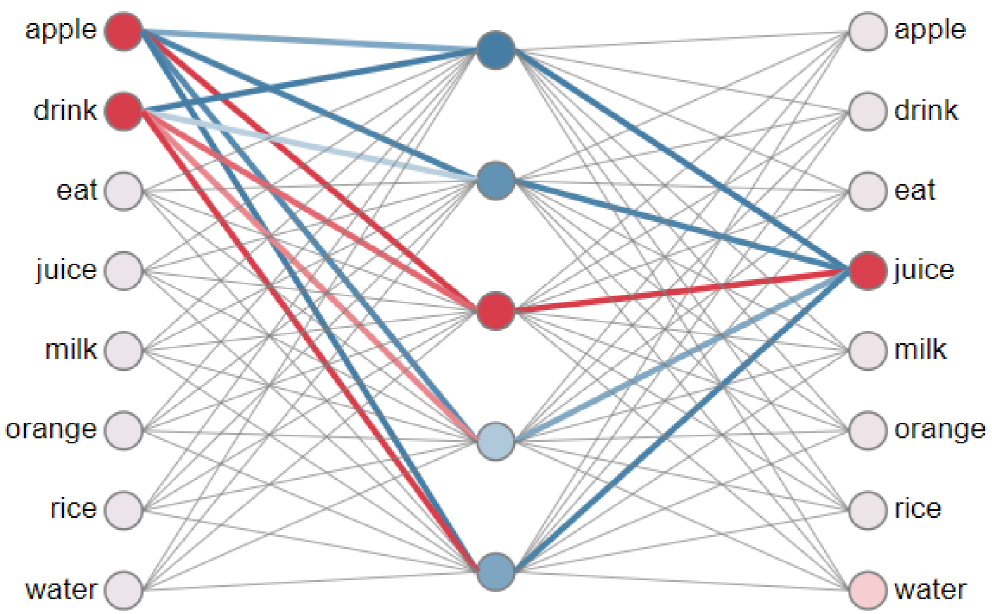
- Predict **context words** based on **target words**

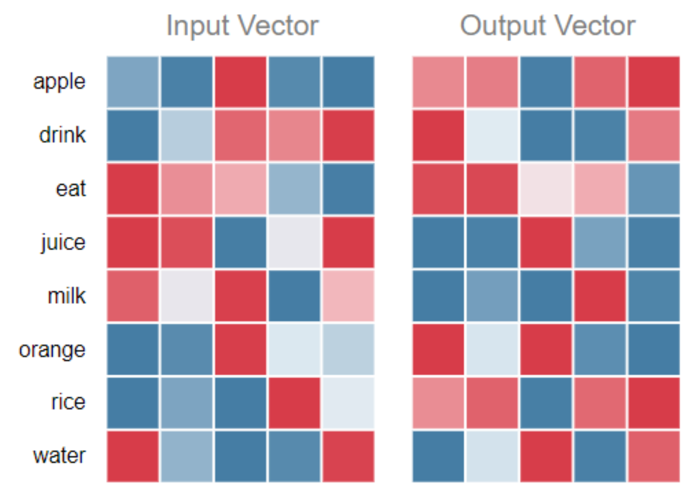


I eat an **apple** every day.

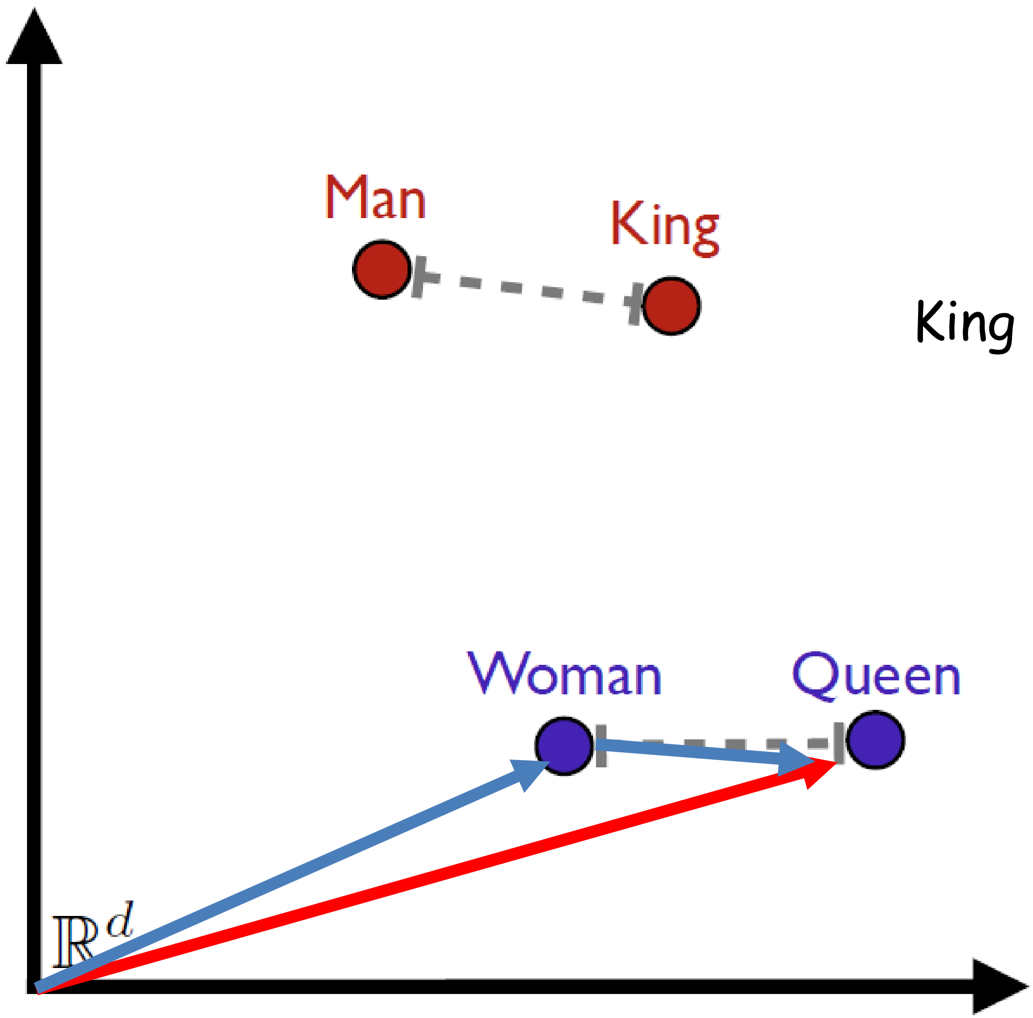
Word2vec visual inspector

https://ronxin.github.io/wevi/



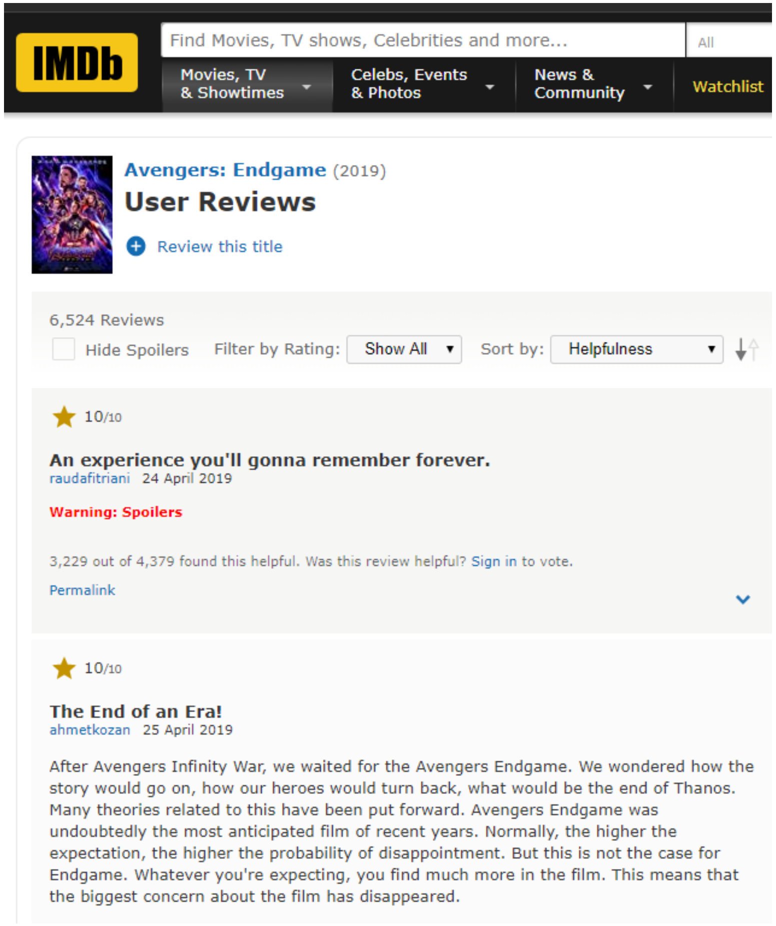


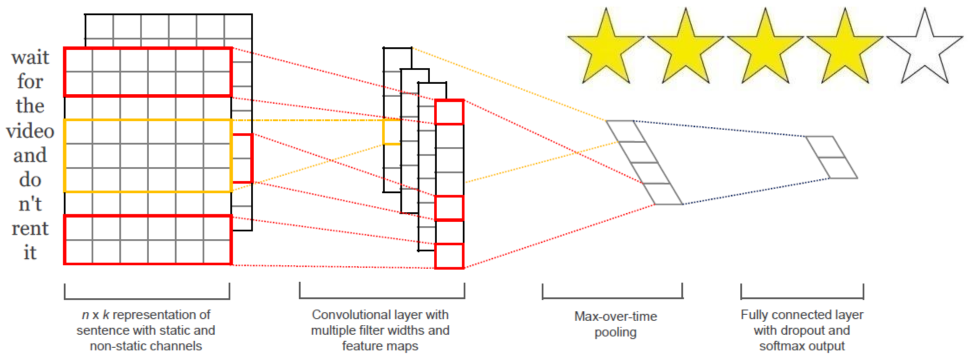
Word analogy



King $\vec{v}$ - man $\vec{v}$ + women $\vec{v}$ ≈ Queen $\vec{v}$

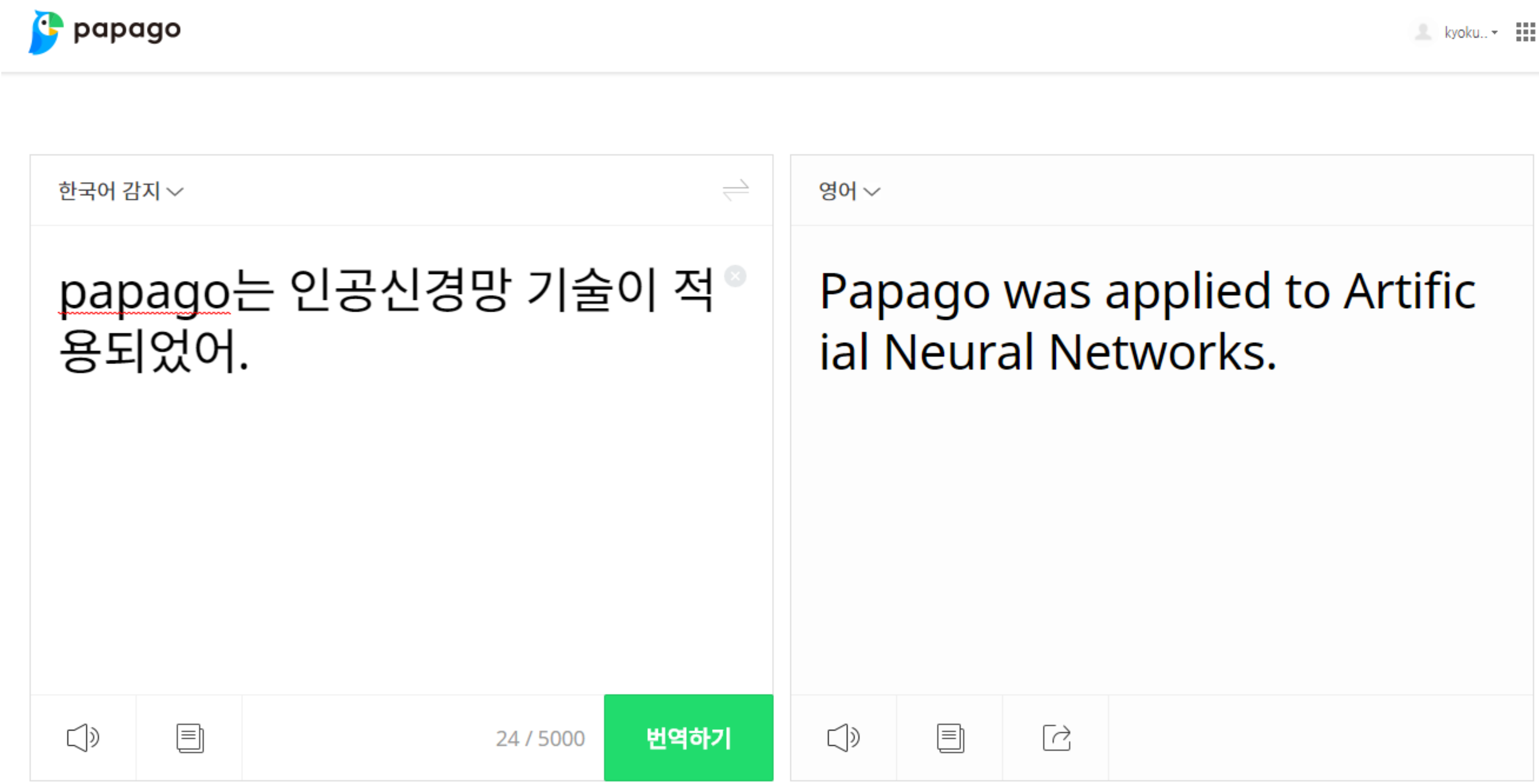
Usage


IMDb
Find Movies, TV shows, Celebrities and more...
All
Movies, TV & Showtimes
Celebs, Events & Photos
News & Community
Watchlist
Avengers: Endgame (2019)
User Reviews
Review this title
6,524 Reviews
Hide Spoilers
Filter by Rating:
Show All
Sort by:
Helpfulness
10/10
An experience you'll gonna remember forever.
raudafitriani 24 April 2019
Warning: Spoilers
3,229 out of 4,379 found this helpful. Was this review helpful? Sign in to vote.
Permalink
10/10
The End of an Era!
ahmetkozan 25 April 2019
After Avengers Infinity War, we waited for the Avengers Endgame. We wondered how the story would go on, how our heroes would turn back, what would be the end of Thanos. Many theories related to this have been put forward. Avengers Endgame was undoubtedly the most anticipated film of recent years. Normally, the higher the expectation, the higher the probability of disappointment. But this is not the case for Endgame. Whatever you're expecting, you find much more in the film. This means that the biggest concern about the film has disappeared.


wait
for
the
video
and
do
n't
rent
it
n x k representation of sentence with static and non-static channels
Convolutional layer with multiple filter widths and feature maps
Max-over-time pooling
Fully connected layer with dropout and softmax output

Usage

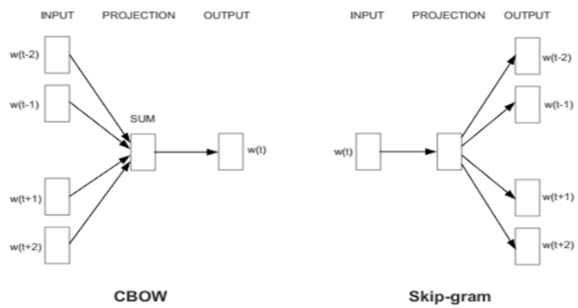
- 기계 번역
  - 예: 파파고를 이용한 외국어 자동 번역 (한국어 → 영어)



Applications

“벤치프레스”

Locally trained Word Embeddings [1]



<word2vec>

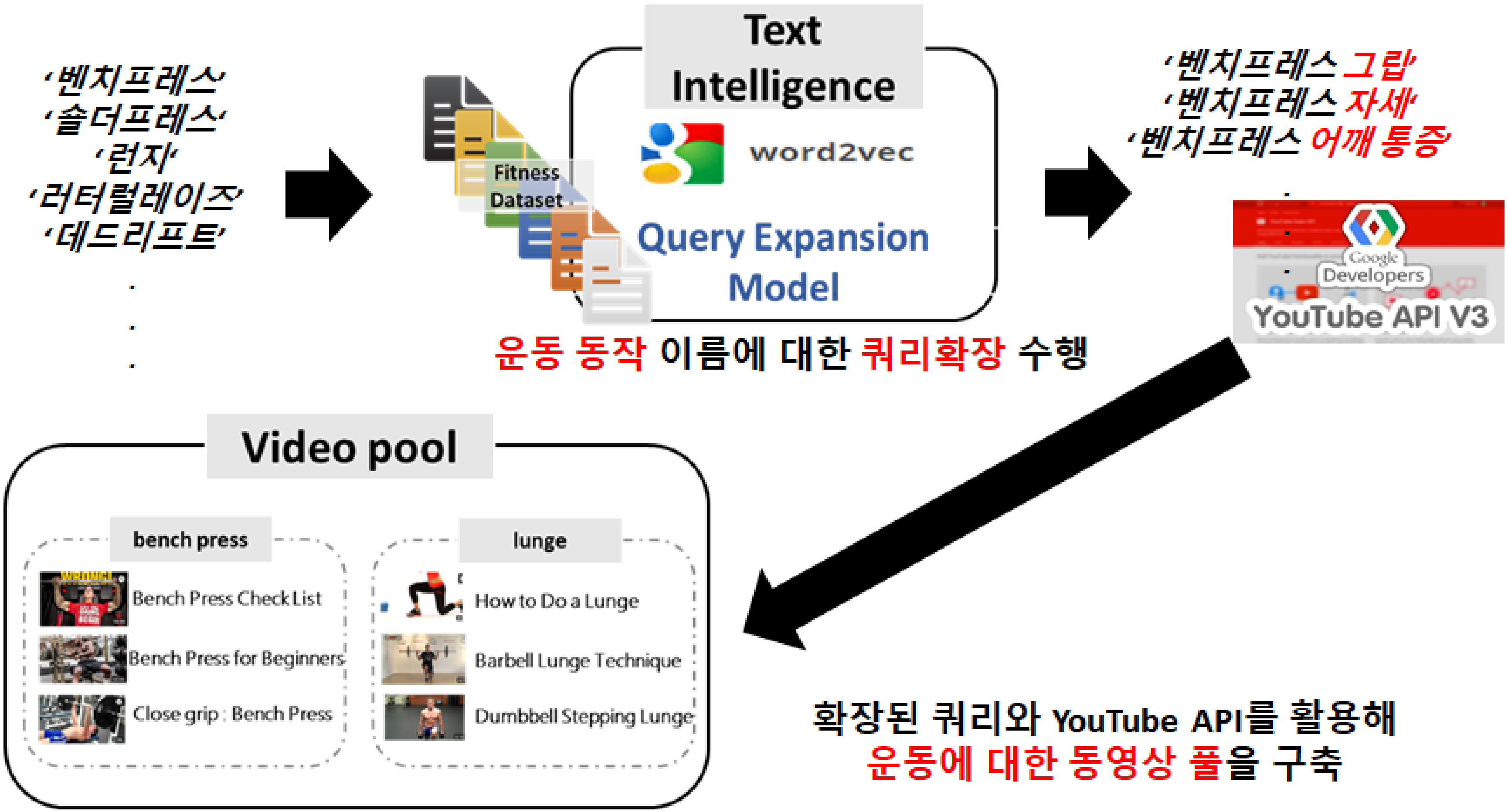
Query Expansion

<Query Expansion Model>

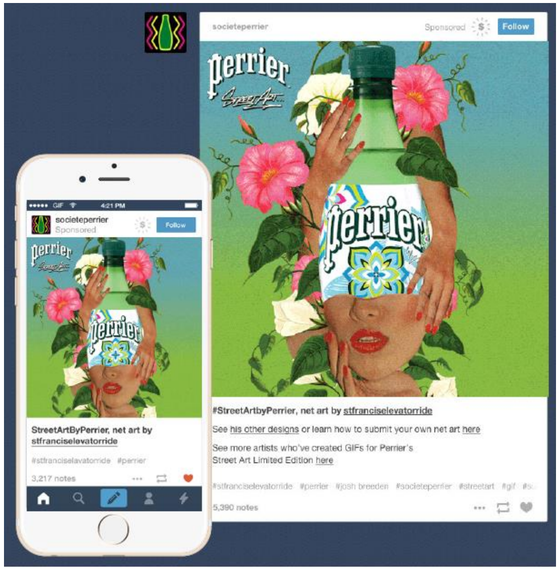
- 분류된 동작을 word2vec과 Query Expansion Model[1]로 처리

[1] Fernando Diaz, Bhaskar, Mitra, Nick Craswell, “Query Expansion with Locally-Trained Word Embeddings”, ACL 2016

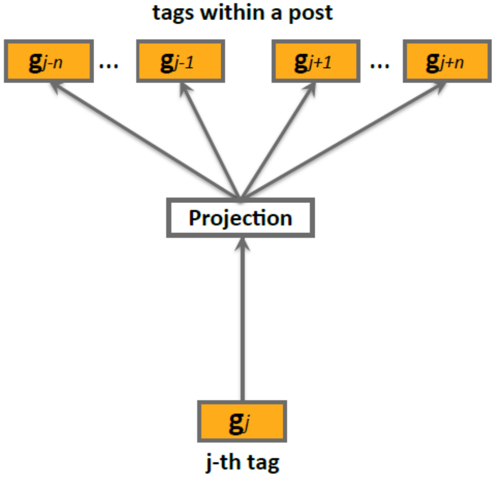
Applications

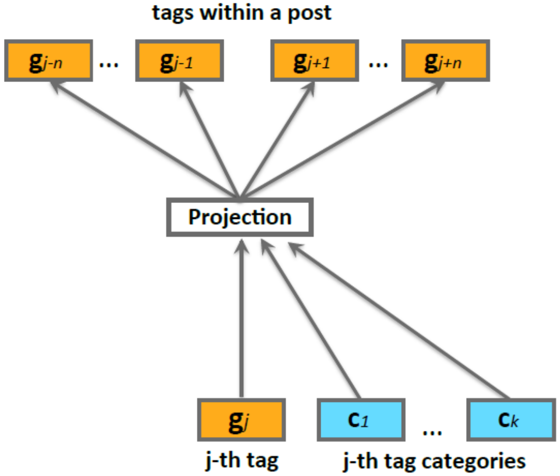


Applications



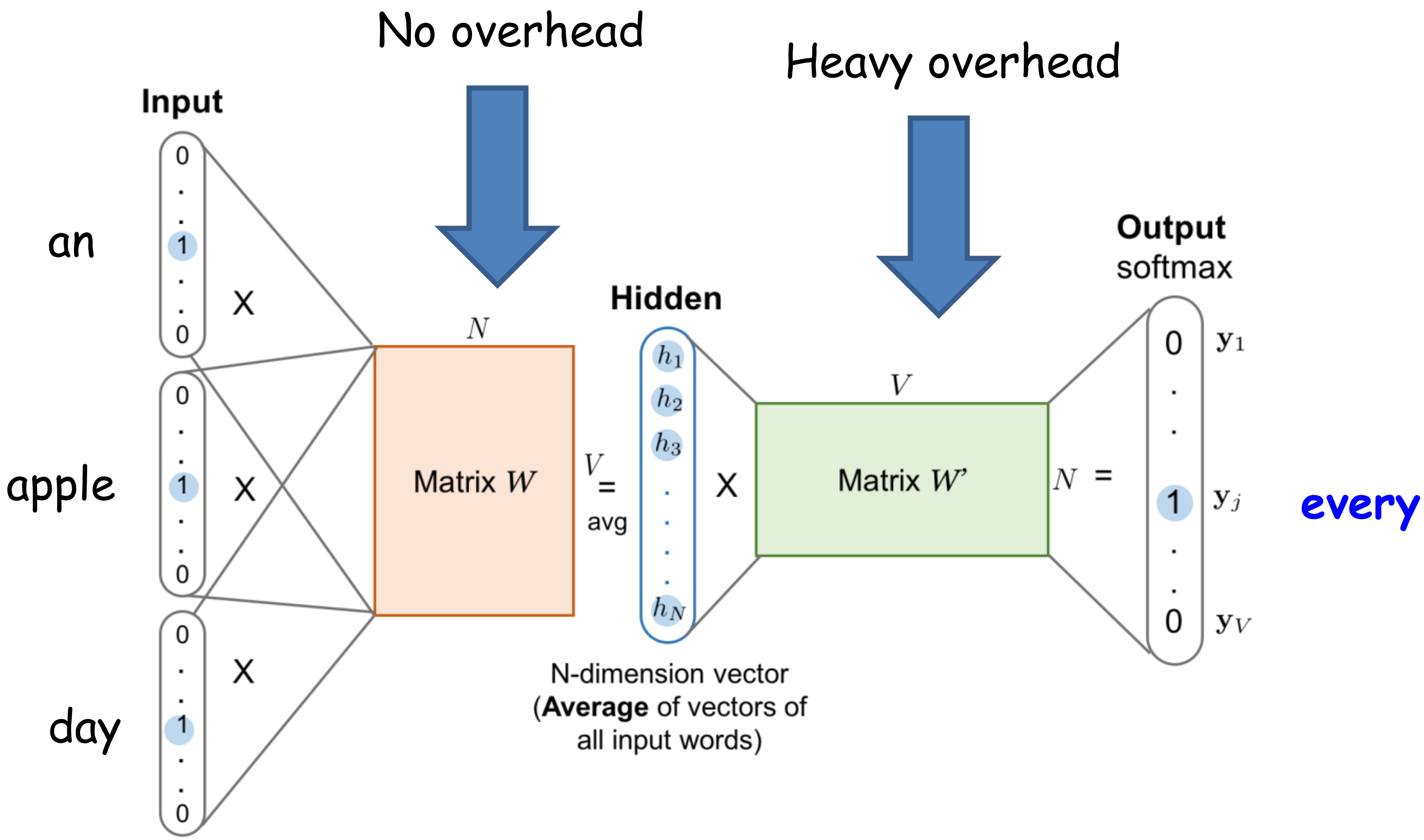
[1]



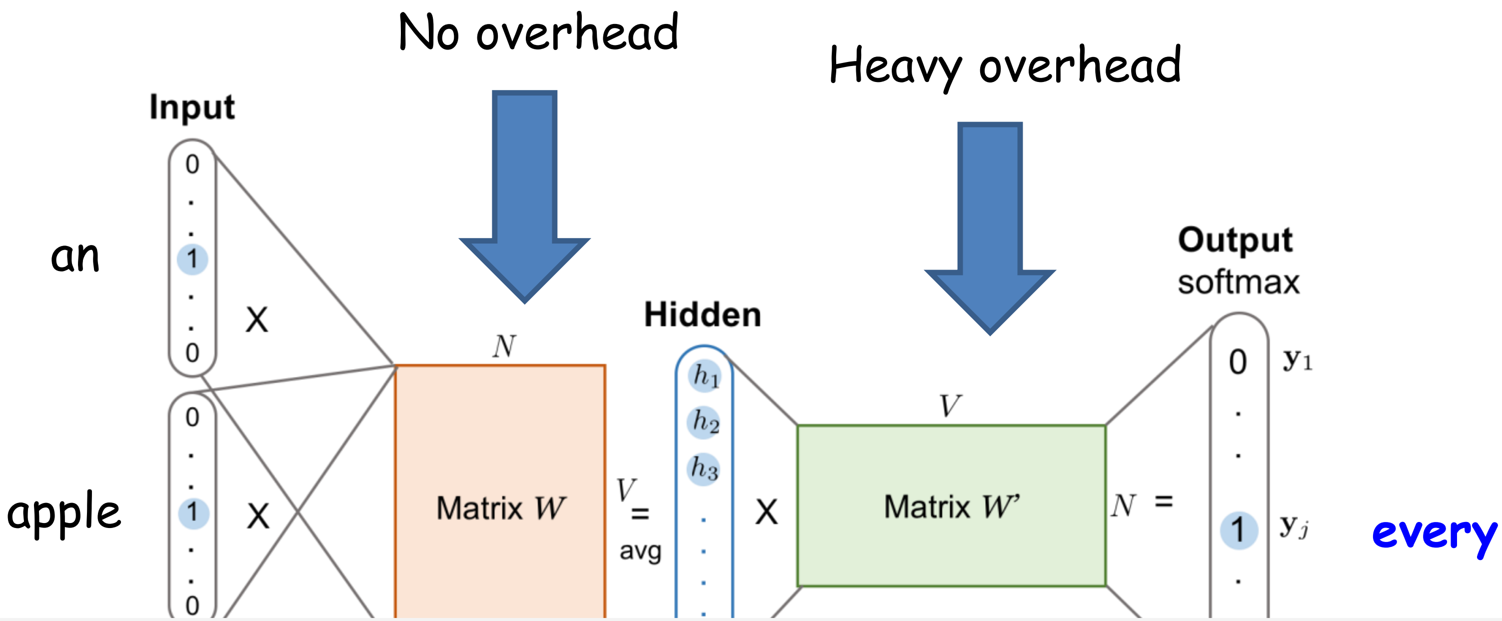


[1] Mihajlo Grbovicy, Vladan Radosavljevicy, Nemanja Djuricy, Narayan Bhamidipatiy, Ananth Nagarajanz, "Gender and Interest Targeting for Sponsored Post Advertising at Tumblr", KDD 2015

Word2vec

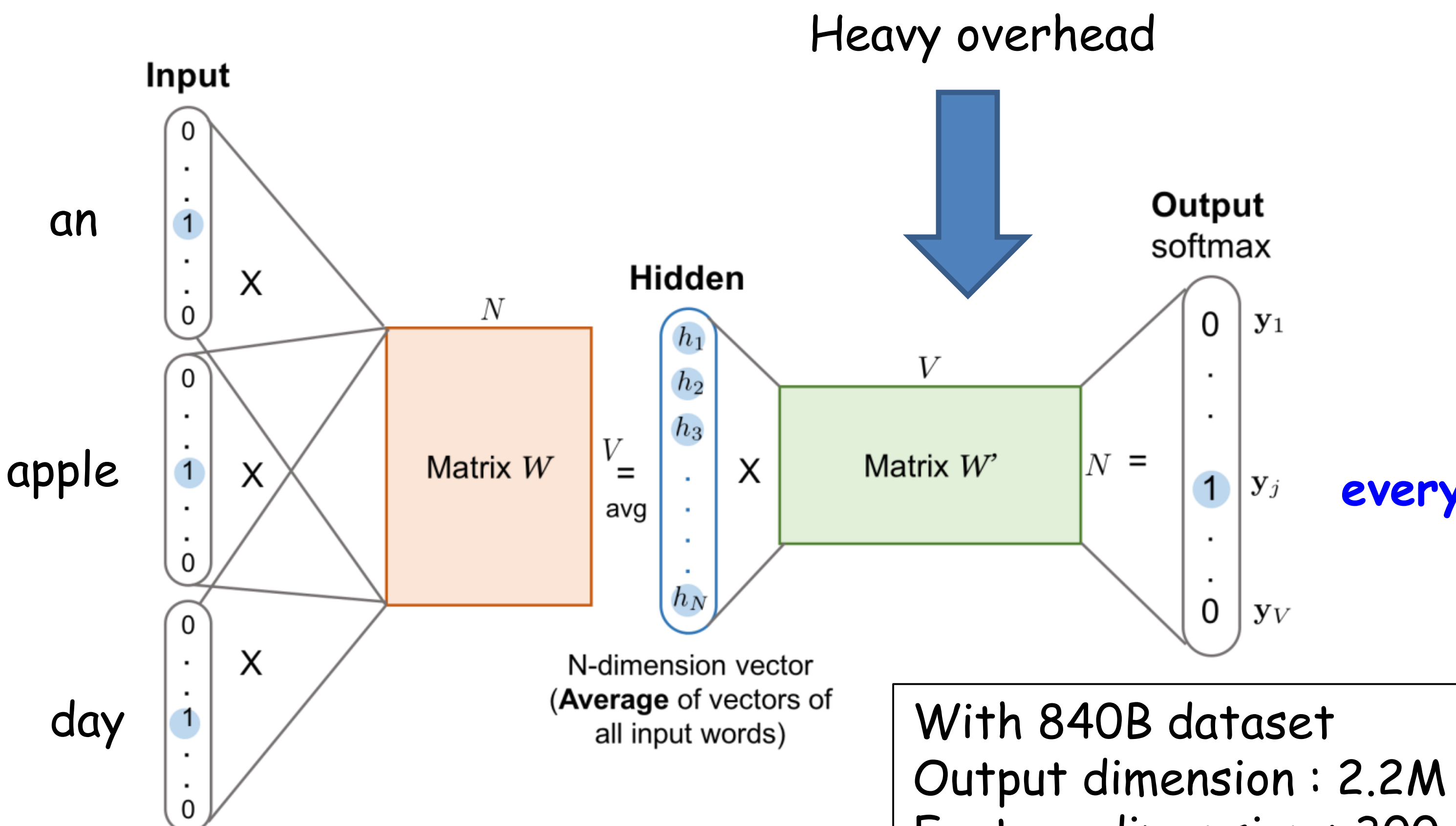

No overhead
Heavy overhead
Input
an
apple
day
X
X
X
N
Matrix W
V
=
avg
Hidden
h_1
h_2
h_3
h_N
X
V
Matrix W'
N =
Output
softmax
0
1
0
y_1
y_j
y_V
every
N-dimension vector
(**Average** of vectors of
all input words)

Word2vec



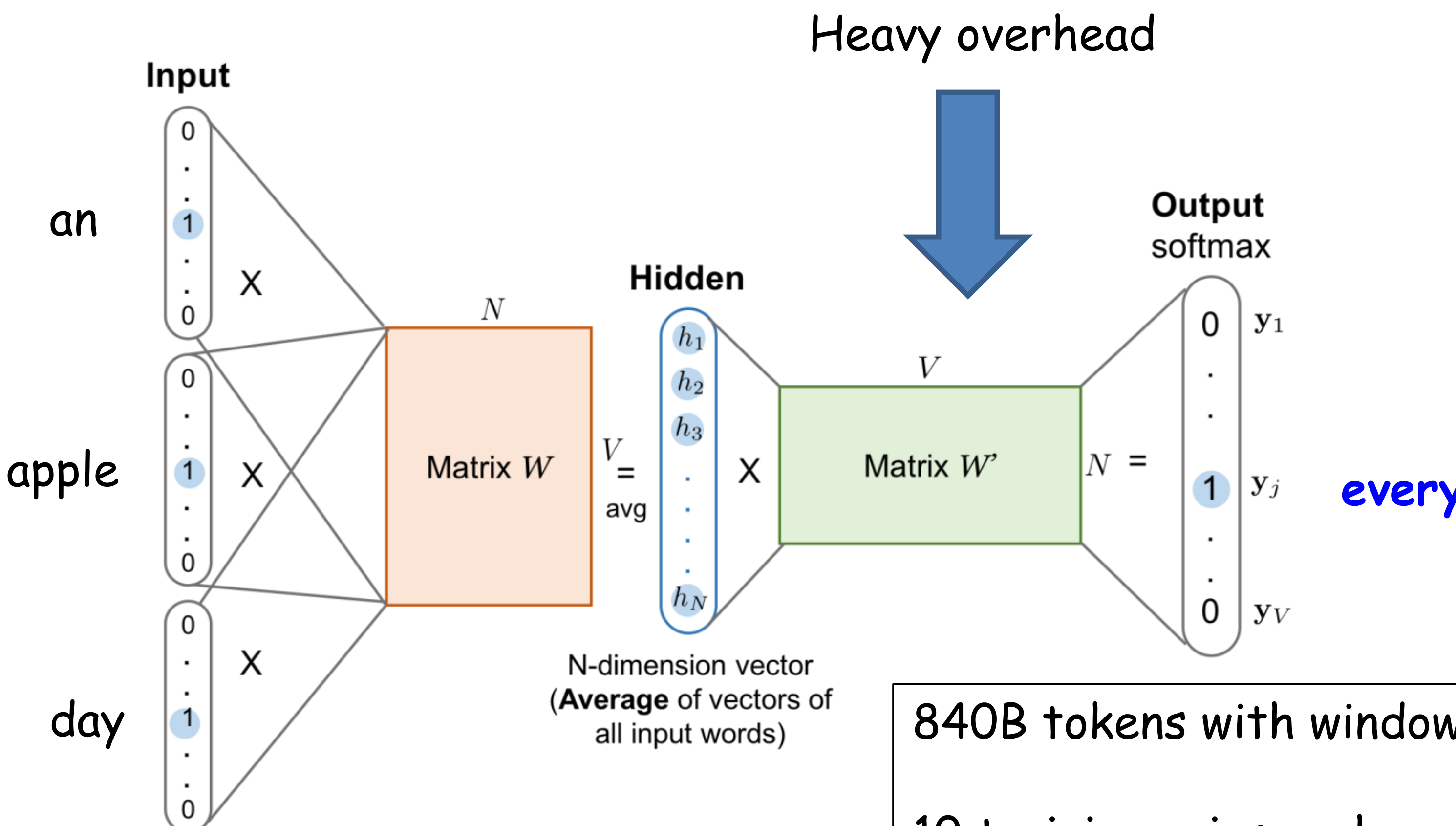
- Wikipedia 2014 + Gigaword 5 (6B tokens, 400K vocab, uncased,
- Common Crawl (42B tokens, 1.9M vocab, uncased, 300d vectors
- Common Crawl (840B tokens, 2.2M vocab, cased, 300d vectors
- Twitter (2B tweets, 27B tokens, 1.2M vocab, uncased, 25d, 50d, 1

Word2vec



With 840B dataset
Output dimension : 2.2M
Feature dimension : 300
W': (2.2M, 300)
->
660M operation to calculate
$y = softmax(W'^T W^k)$

Word2vec

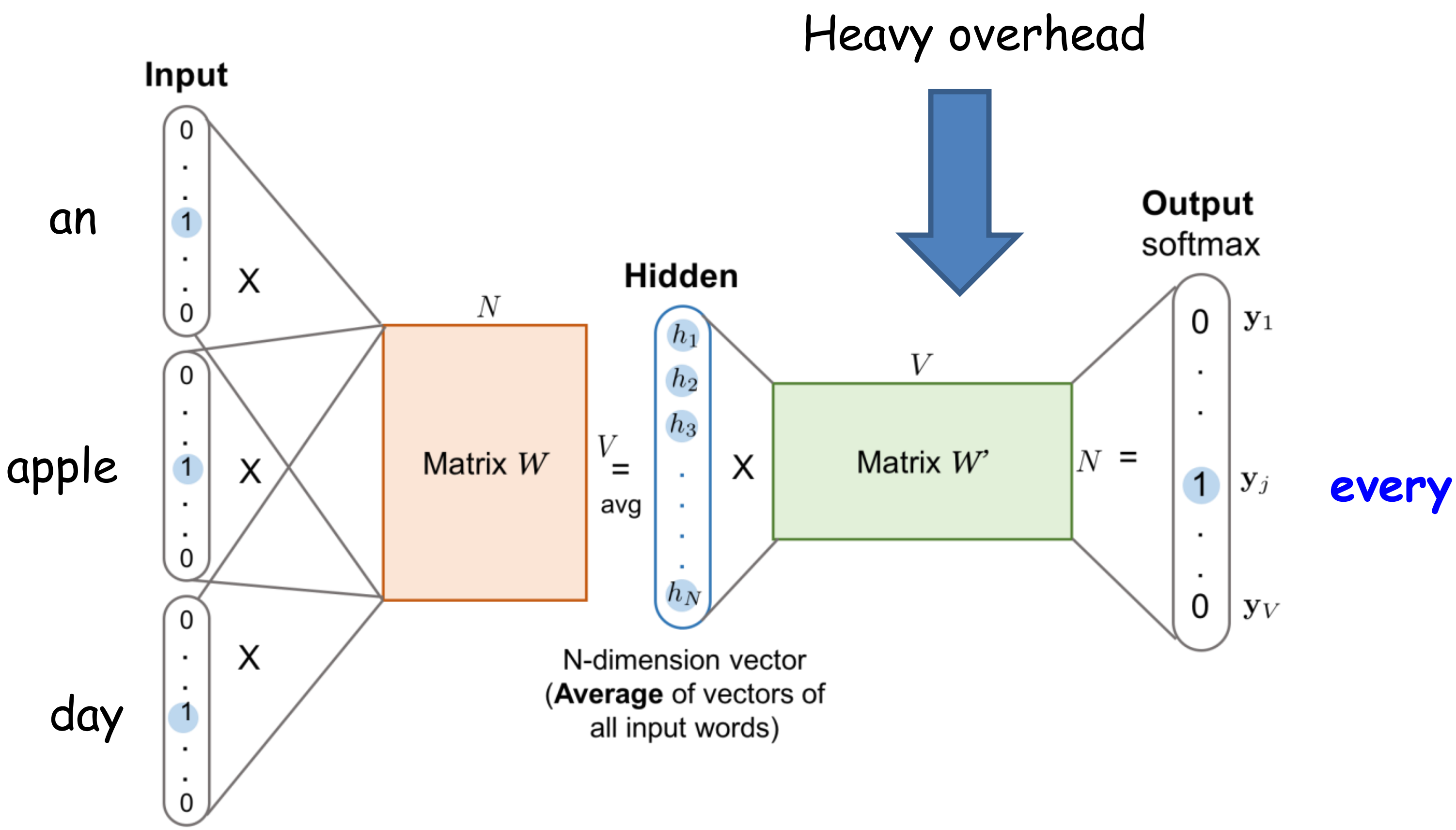


840B tokens with window size 5

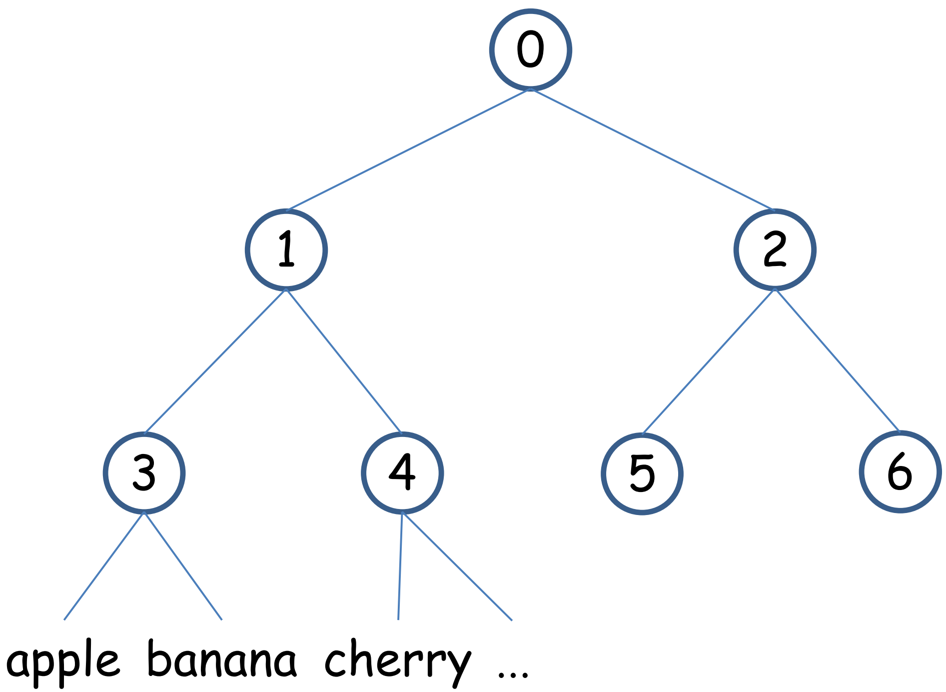
10 training pairs each word

660M x 8.4T operations an epoch

Word2vec: Hierarchical softmax



Word2vec: Hierarchical softmax



With 840B dataset

Output dimension : 2.2M
Feature dimension : 300

Average activated nodes : 21

660M (softmax) → 6.3k (HS)

Word2vec: Hierarchical softmax

1. Give every word a binary code (Huffman coding recommended)

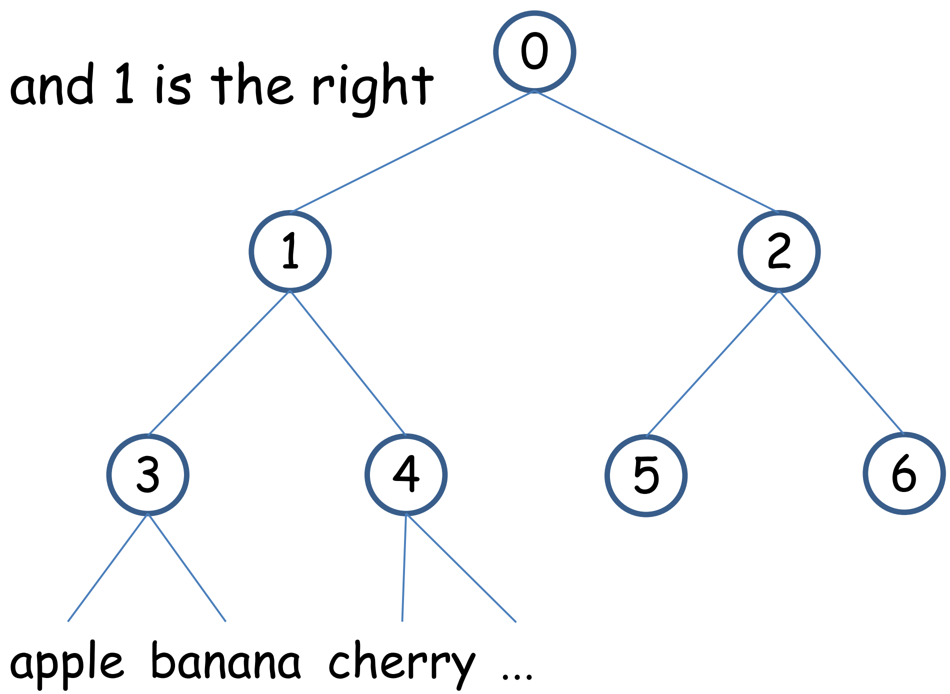
ex) apple : 000
banana : 001
cherry : 010
...

Word2vec: Hierarchical softmax

2. Make a binary tree whose leaf nodes are the words

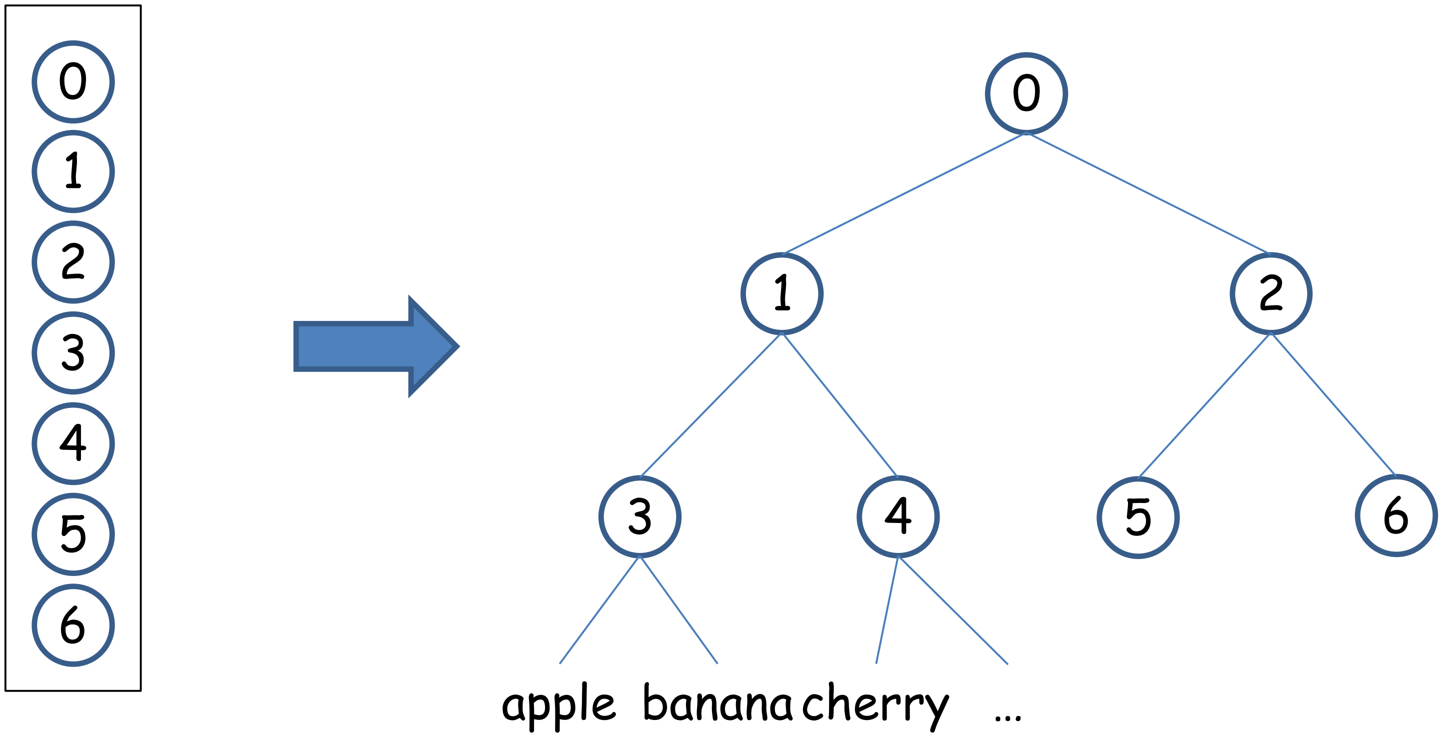
ex) apple : 000
banana : 001
cherry : 010
...

Suppose that 0 is the left and 1 is the right



Word2vec: Hierarchical softmax

3. Assign elements of the final layer of word2vec to the tree's nodes

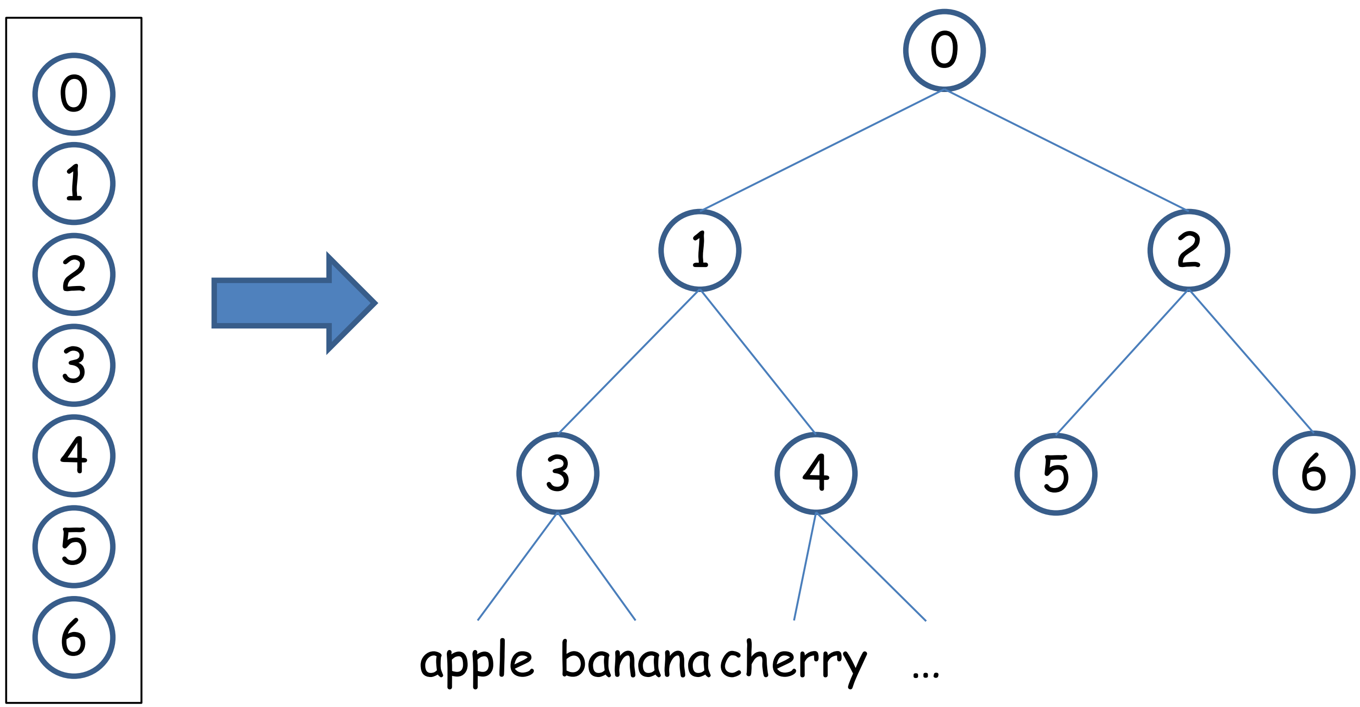


Word2vec: Hierarchical softmax

4. The elements are calculated in the same way of basic softmax, except for activation

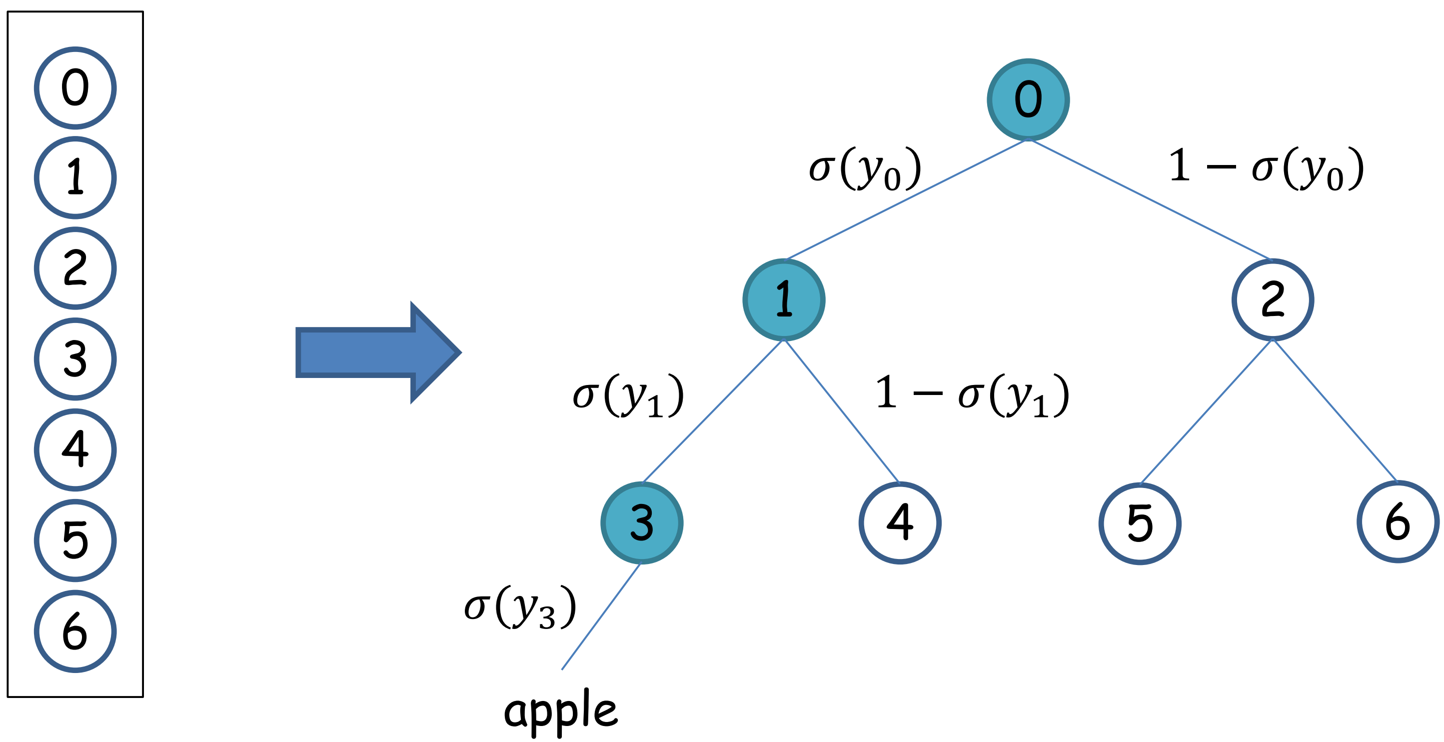
Each node has sigmoid activation function
Instead of softmax

$$S(x) = \frac{1}{1+e^{-x}} = \frac{e^x}{e^x+1}.$$



Word2vec: Hierarchical softmax

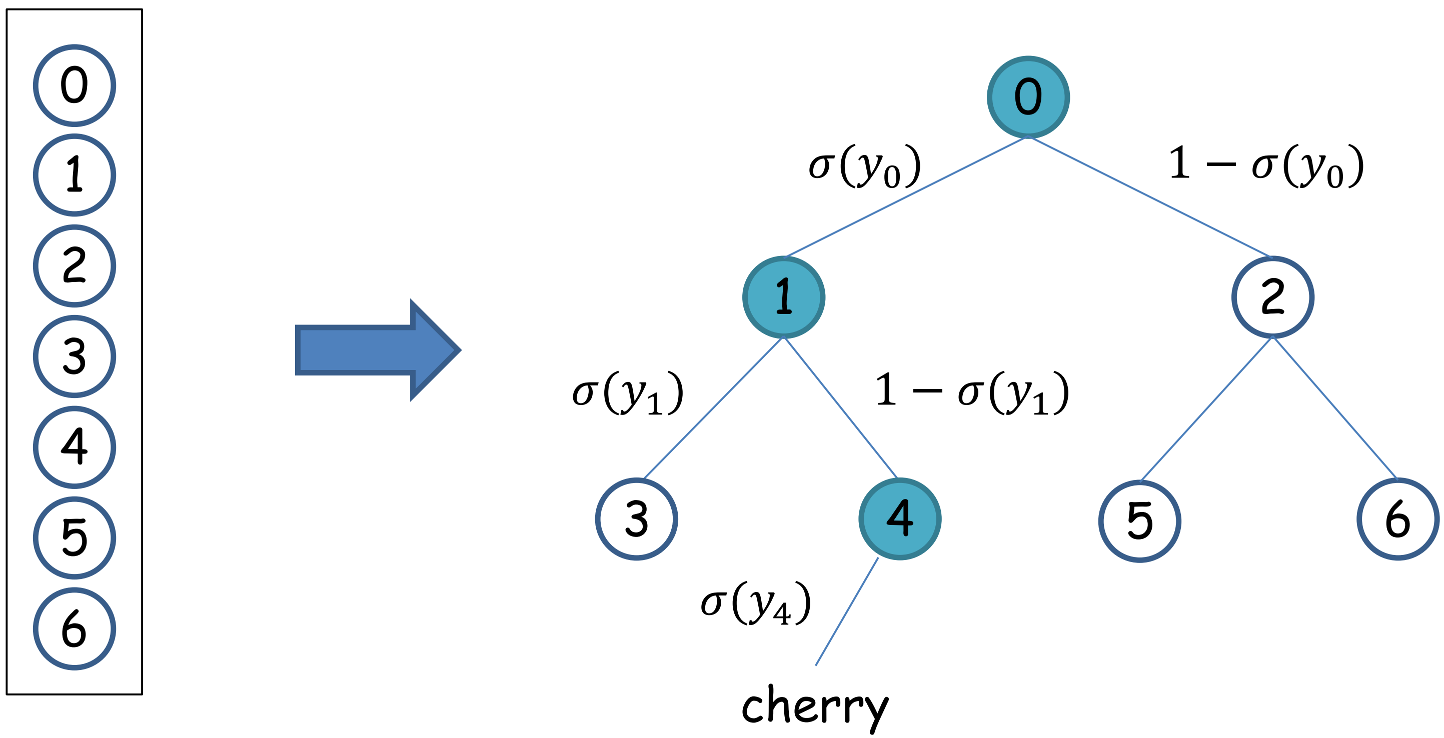
5. The probability of a word is a product of nodes on the way



$$p(apple) = \sigma(y_0)\ \sigma(y_1)\ \sigma(y_3)$$

Word2vec: Hierarchical softmax

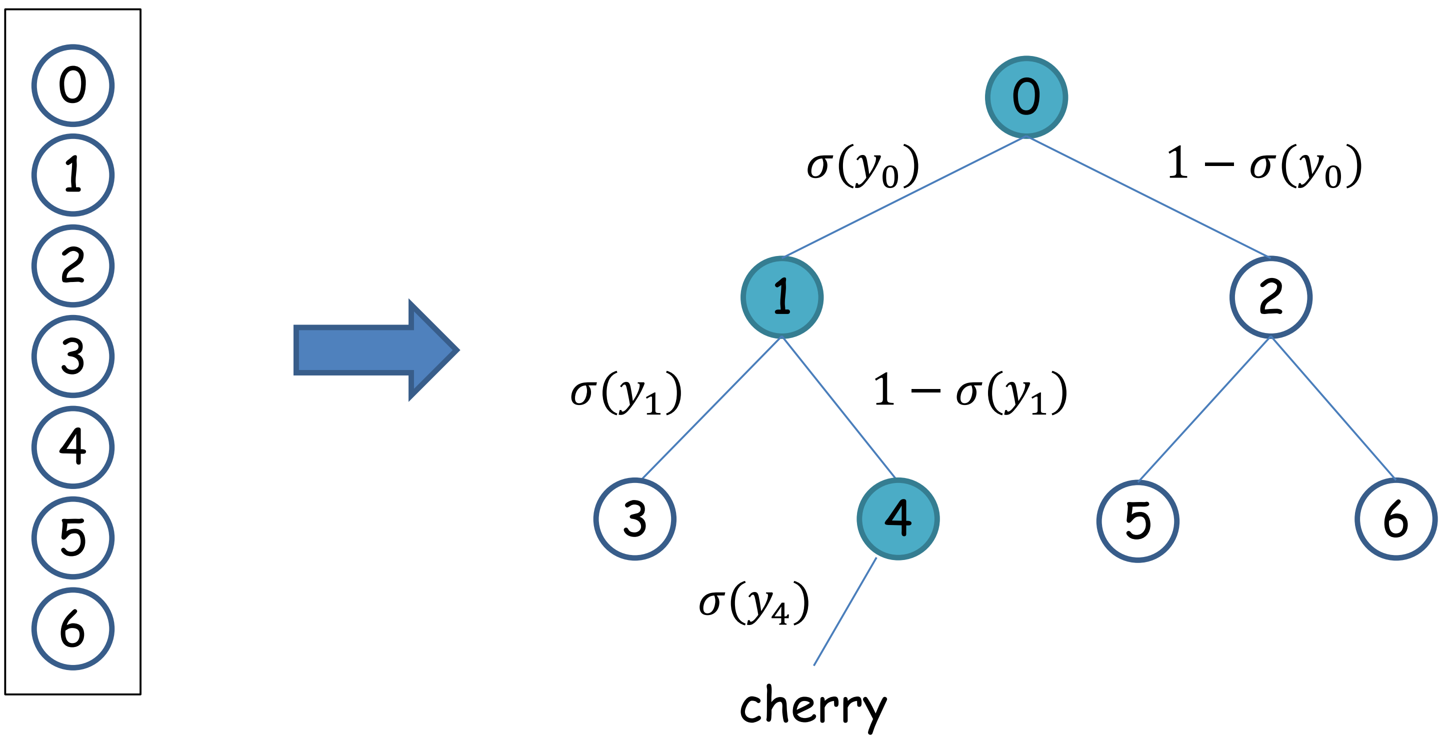
5. The probability of a word is a product of nodes on the way



$$p(cherry) = \sigma(y_0)\,(1-\sigma(y_1))\,\sigma(y_4)$$

Word2vec: Hierarchical softmax

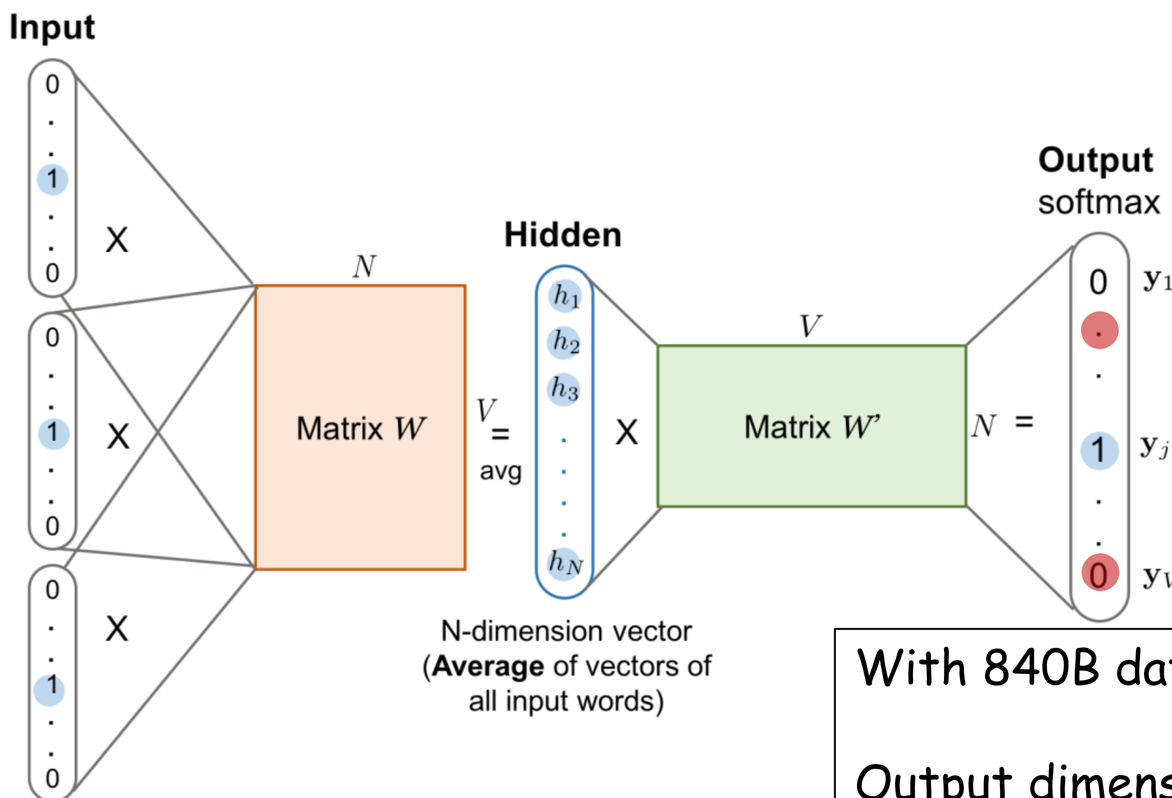
6. Maximize the probability by gradient descent on negative log likelihood



$p(cherry) = \sigma(y_0)\,(1-\sigma(y_1))\,\sigma(y_4)$

Minimize $-\log p(cherry)$

Word2vec: Negative sampling



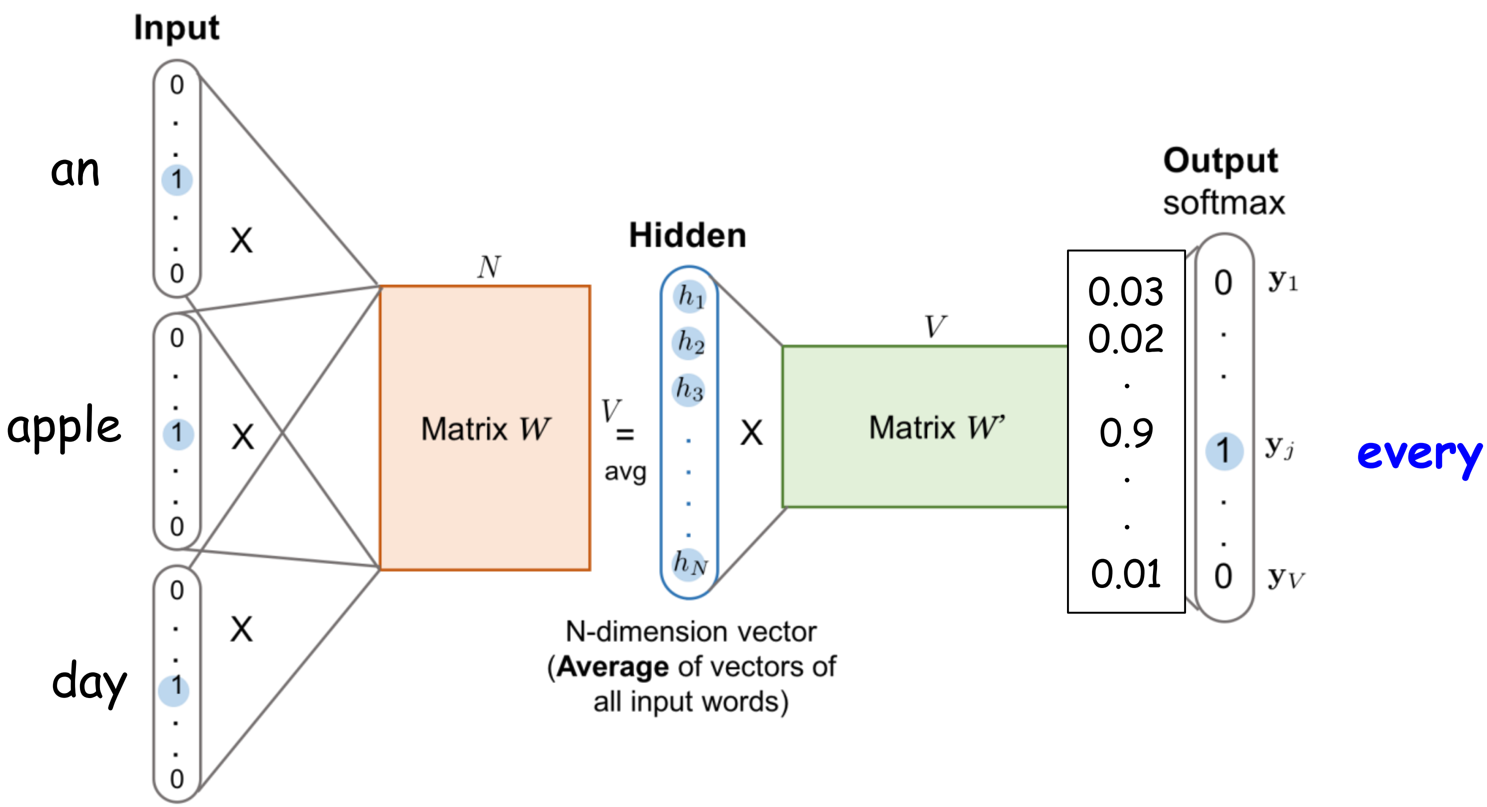
With 840B dataset

Output dimension : 2.2M
Feature dimension : 300
Negative samples : 5

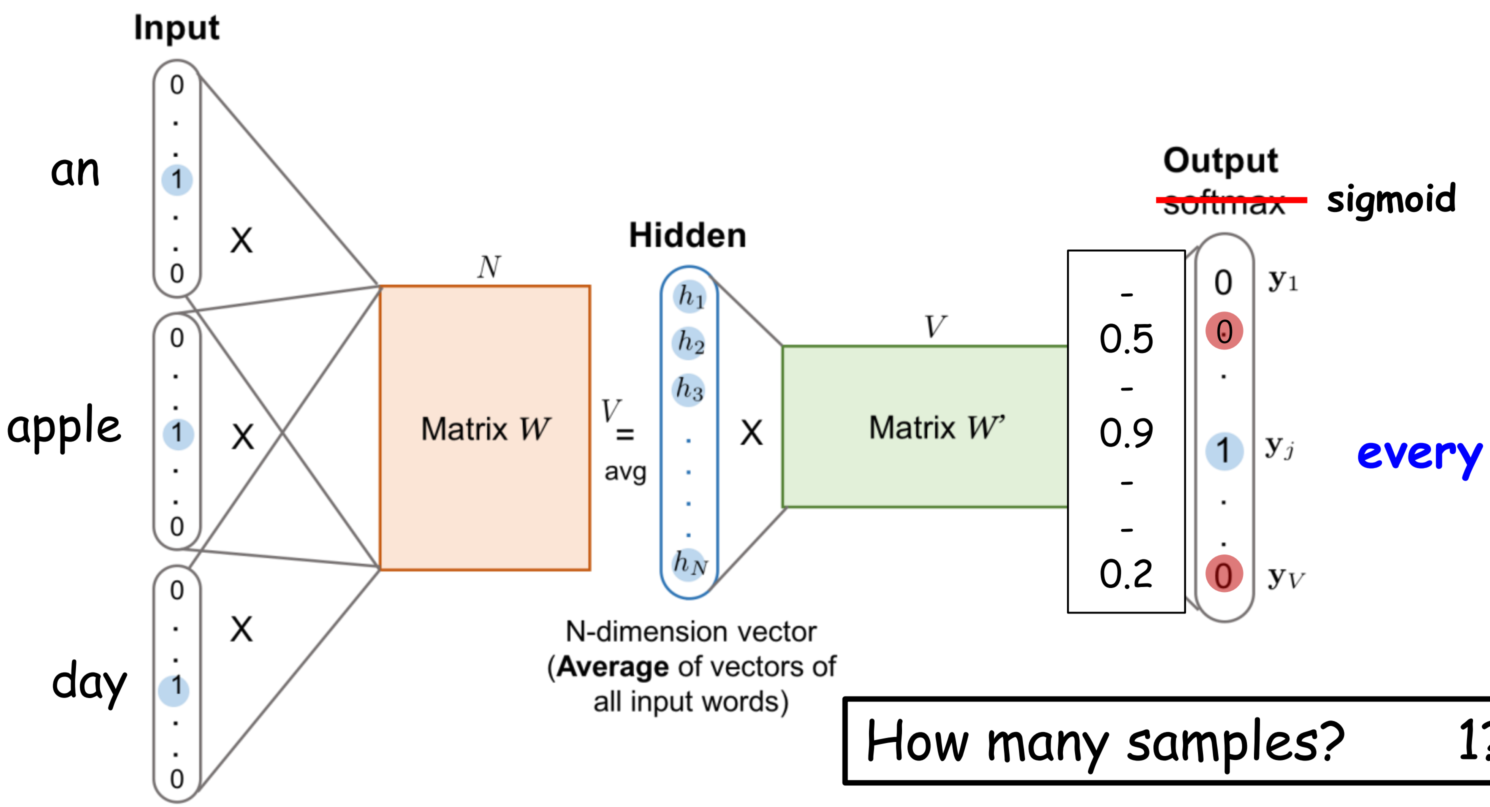
Average activated nodes : 1 + 5

660M (softmax) → 1.8k (neg)
Hierarchical softmax : 6.3k

Word2vec: Negative sampling



Word2vec: Negative sampling



How many samples? 1? 5? 10?

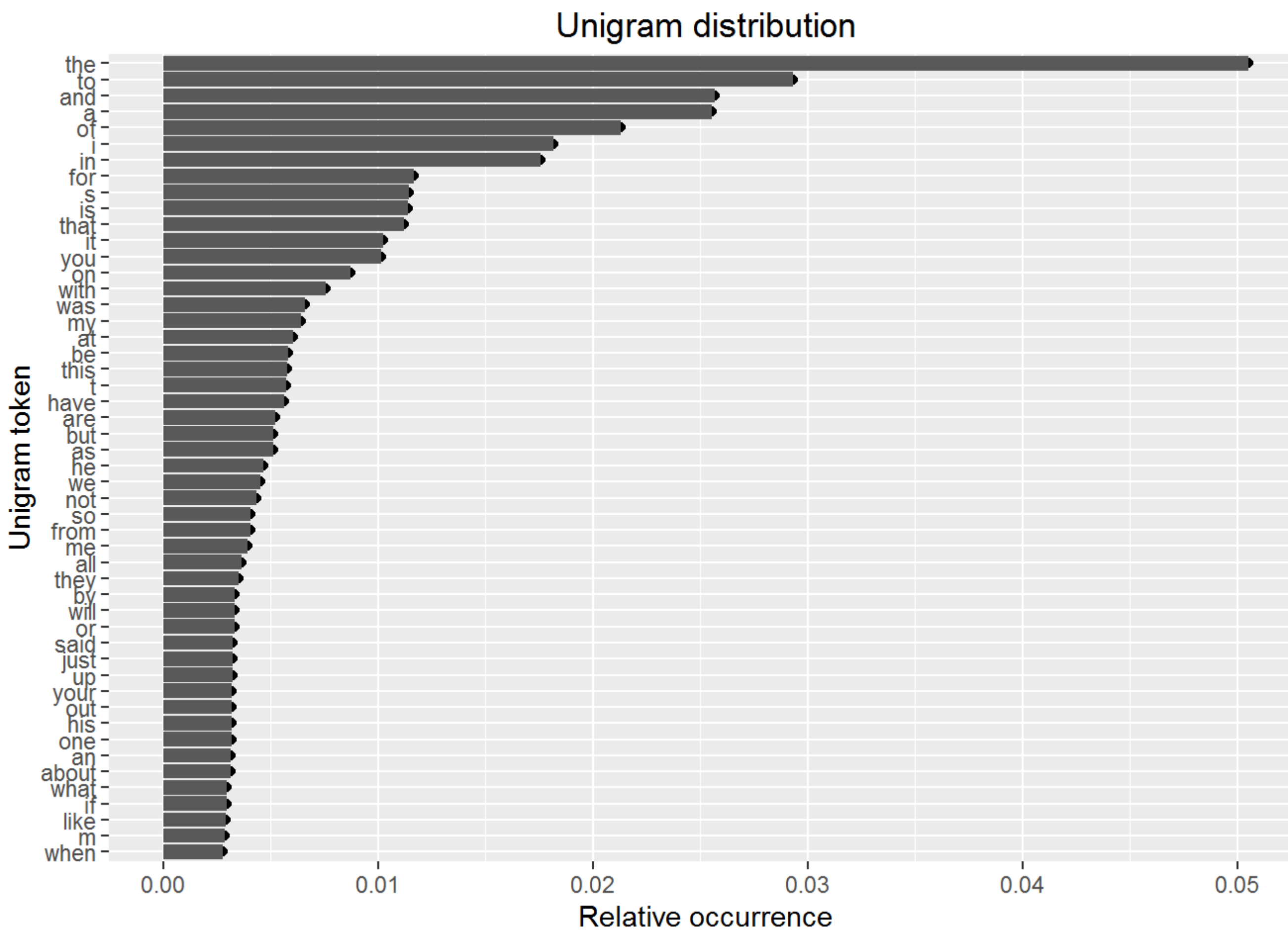
5~15 samples recommended
3~5 samples enough on big corpus

Word2vec: Negative sampling

an

apple

day



How to sample?

Uniformly? Linearly?
With some heuristic function?

(Unigram distribution)^(3/4)

Word2vec: subsampling

The orange is the fruit of the citrus species Citrus × sinensis in the family Rutaceae. It is also called sweet orange, to distinguish it from the related Citrus × aurantium, referred to as bitter orange. The sweet orange reproduces asexually varieties of sweet orange arise through mutations.

Highly frequent words are actually meaningful?

Word2vec: subsampling

~~The~~ orange is ~~the~~ fruit of ~~the~~ citrus species Citrus × sinensis in the family Rutaceae. It is also called sweet orange, to distinguish it from the related Citrus × aurantium, referred to as bitter orange. ~~The~~ sweet orange reproduces asexually varieties of sweet orange arise through mutations.

Discard frequent words with probability

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

AI School 6기 8주차

Word2Vec 실습

Pre-trained Word2Vec

- https://code.google.com/archive/p/word2vec/
- GoogleNews-vectors-negative300.bin.gz download
- 실습 중인 파이썬 파일 (.py)과 같은 경로로 이동

Pre-trained word and phrase vectors

We are publishing pre-trained vectors trained on part of Google News dataset (about 100 billion words). The model contains 300-dimensional vectors for 3 million words and phrases. The phrases were obtained using a simple data-driven approach described in [2]. The archive is available here: GoogleNews-vectors-negative300.bin.gz.

Pre-trained Word2Vec

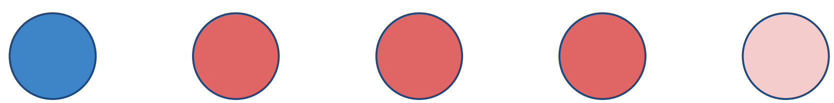
- Gensim 라이브러리를 통한 Word2Vec 모델 loading

```
from gensim.models.keyedvectors import KeyedVectors

model = KeyedVectors.load_word2vec_format("./GoogleNews-vectors-negative300.bin", binary=True)

print(model['apple'])
```

apple



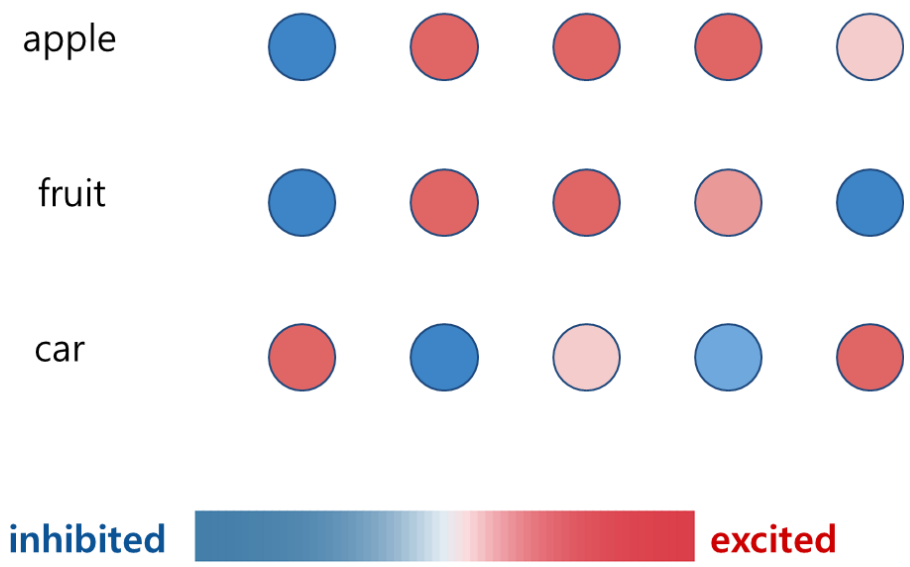
Pre-trained Word2Vec

- Similarity, most similar words

```
from gensim.models.keyedvectors import KeyedVectors

model = KeyedVectors.load_word2vec_format("./GoogleNews-vectors-negative300.bin",
binary=True)

print("similarity between apple and fruit: {}".format(model.similarity("apple", "fruit")))
print("similarity between apple and car: {}".format(model.similarity("apple", "car")))
print(model.most_similar("apple", topn=10))
```



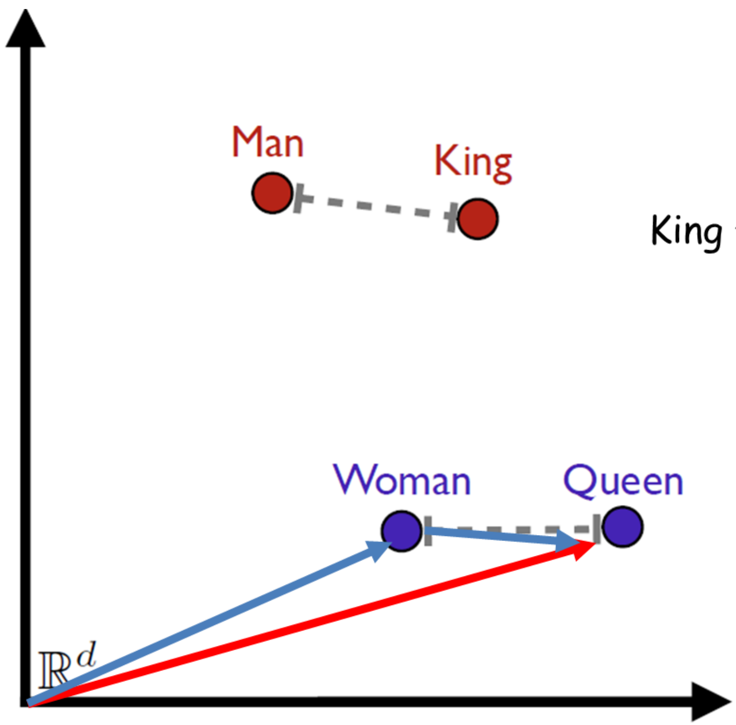
Pre-trained Word2Vec

- Word analogy

```
from gensim.models.keyedvectors import KeyedVectors

model = KeyedVectors.load_word2vec_format("./GoogleNews-vectors-negative300.bin", binary=True)

print(model.most_similar(positive=['king', ‘women'], negative=[‘man'], topn=10))
```



King $\vec{v}$ - man $\vec{v}$ + women $\vec{v}$ ≈ Queen $\vec{v}$

CNN with Pre-trained Word2Vec

- train.py (CNN4TC)

```
...
from gensim.models.keyedvectors import KeyedVectors
import re

tf.flags.DEFINE_string("imdb_pos_data_file", "./data/train/pos/*", "Data source for the positive data.")
tf.flags.DEFINE_string("imdb_neg_data_file", "./data/train/neg/*", "Data source for the negative data.")
tf.flags.DEFINE_string("word2vec", "./data/GoogleNews-vectors-negative300.bin", "Word2vec file with pre-trained embeddings (default: None)")
...
tf.flags.DEFINE_integer("embedding_dim", 300, "Dimensionality of word embedding”)
```

CNN with Pre-trained Word2Vec

- train.py (CNN4TC)

```
...
sess.run(tf.global_variables_initializer())

print("Loading W2V data...")
pre_emb = KeyedVectors.load_word2vec_format(FLAGS.word2vec, binary=True)
pre_emb.init_sims(replace=True)
num_keys = len(pre_emb.vocab)
print("loaded word2vec len ", num_keys)
```

CNN with Pre-trained Word2Vec

- train.py (CNN4TC)

```
...
if FLAGS.word2vec:
    initW = np.random.uniform(-0.25, 0.25, (len(vocab_processor.vocabulary_),
FLAGS.embedding_dim))
    for w in vocab_processor.vocabulary_._mapping:
        arr = []
        s = re.sub('[^0-9a-zA-Z]+', '', w)
        if w in pre_emb:
            arr = pre_emb[w]
        elif w.lower() in pre_emb:
            arr = pre_emb[w.lower()]
        elif s in pre_emb:
            arr = pre_emb[s]
        elif s.isdigit():
            arr = pre_emb['1']
        if len(arr) > 0:
            idx = vocab_processor.vocabulary_.get(w)
            initW[idx] = np.asarray(arr).astype(np.float32)
    print("assigning initW to cnn. len=" + str(len(initW)))
    sess.run(cnn.W.assign(initW))
```

Training Word2vec through gensim

- w2v_train.py

```
from gensim.test.utils import get_tmpfile
from gensim.models import Word2Vec
from gensim.models.word2vec import LineSentence, PathLineSentences

sentences = LineSentence("./data/news1.txt")

model = Word2Vec(sentences, size=100, window=5, min_count=1, workers=4)
model.save("word2vec.model")
```

```
model = Word2Vec.load("word2vec.model")
print(model.wv.most_similar("car", topn=200))
print(len(model.wv.vocab))
```

Word analogy task

- w2v_train.py
- https://code.google.com/archive/p/word2vec/source/default/source에서 questions-words.txt 다운로드

```
model = Word2Vec.load("word2vec.model")
score, predictions = model.wv.evaluate_word_analogies('./data/questions-words.txt')
print(score)
```

Type of relationship	Word Pair 1		Word Pair 2	
Common capital city	Athens	Greece	Oslo	Norway
All capital cities	Astana	Kazakhstan	Harare	Zimbabwe
Currency	Angola	kwanza	Iran	rial
City-in-state	Chicago	Illinois	Stockton	California
Man-Woman	brother	sister	grandson	granddaughter
Adjective to adverb	apparent	apparently	rapid	rapidly
Opposite	possibly	impossibly	ethical	unethical
Comparative	great	greater	tough	tougher
Superlative	easy	easiest	lucky	luckiest
Present Participle	think	thinking	read	reading
Nationality adjective	Switzerland	Swiss	Cambodia	Cambodian
Past tense	walking	walked	swimming	swam
Plural nouns	mouse	mice	dollar	dollars
Plural verbs	work	works	speak	speaks

Word analogy task (Google)

- w2v_train.py

```
model = KeyedVectors.load_word2vec_format("./data/GoogleNews-vectors-negative300.bin", binary=True)
score, predictions = model.evaluate_word_analogies('./data/questions-words.txt')
print(score)
```

Type of relationship	Word Pair 1		Word Pair 2	
Common capital city	Athens	Greece	Oslo	Norway
All capital cities	Astana	Kazakhstan	Harare	Zimbabwe
Currency	Angola	kwanza	Iran	rial
City-in-state	Chicago	Illinois	Stockton	California
Man-Woman	brother	sister	grandson	granddaughter
Adjective to adverb	apparent	apparently	rapid	rapidly
Opposite	possibly	impossibly	ethical	unethical
Comparative	great	greater	tough	tougher
Superlative	easy	easiest	lucky	luckiest
Present Participle	think	thinking	read	reading
Nationality adjective	Switzerland	Swiss	Cambodia	Cambodian
Past tense	walking	walked	swimming	swam
Plural nouns	mouse	mice	dollar	dollars
Plural verbs	work	works	speak	speaks

Training Word2vec through gensim

- Parameters
  - size: 단어 벡터의 차원
  - window size: context 단어 수 / 2
  - min_count: 최소 빈도수 기준, 단어사전에 포함 여부 결정
  - wokers: 스레드 수
  - sg: 1이면 skip-gram 사용
  - hs: 1이면 hierarchical soft, 0이면 negative sampling 사용
  - negative: negative sample의 개수
  - ns_exponent: unigram distribution에 적용될 지수 값
  - cbow_mean: 1이면 context 단어의 평균을 사용, 0이면 합을 사용
  - alpha: learning rate
  - min_alpha: learning rate decay 시에 최소 learning rate
  - max_vocab_size: 단어 사전의 최대 크기
  - iter: epoch 수
  - sorted_vocab: 1이면 사전의 단어들을 빈도수 기준 내림차순 정렬
  - batch_words: batch size

More powerful model!

- w2v_train.py

```
sentences = LineSentence("./data/news1.txt")

model = Word2Vec(sentences, size=300, window=10, min_count=5, workers=4, sg=0,
hs=0, negative=15, ns_exponent=0.75, cbow_mean=1, alpha=0.01, min_alpha=0.0001,
iter=10)
model.save("word2vec.model")
score, predictions = model.wv.evaluate_word_analogies('./data/questions-words.txt')
print(score)
```

More data!

- w2v_train.py

```
sentences = PathLineSentences("./data/1billion/")
#
model = Word2Vec(sentences, size=300, window=10, min_count=5, workers=4, sg=0,
hs=0, negative=15, ns_exponent=0.75, cbow_mean=1, alpha=0.01, min_alpha=0.0001,
iter=3)
model.save("word2vec.model")
print(len(model.wv.vocab))
score, predictions = model.wv.evaluate_word_analogies('./data/questions-words.txt')
print(score)
```

Q&A