Word Embedding

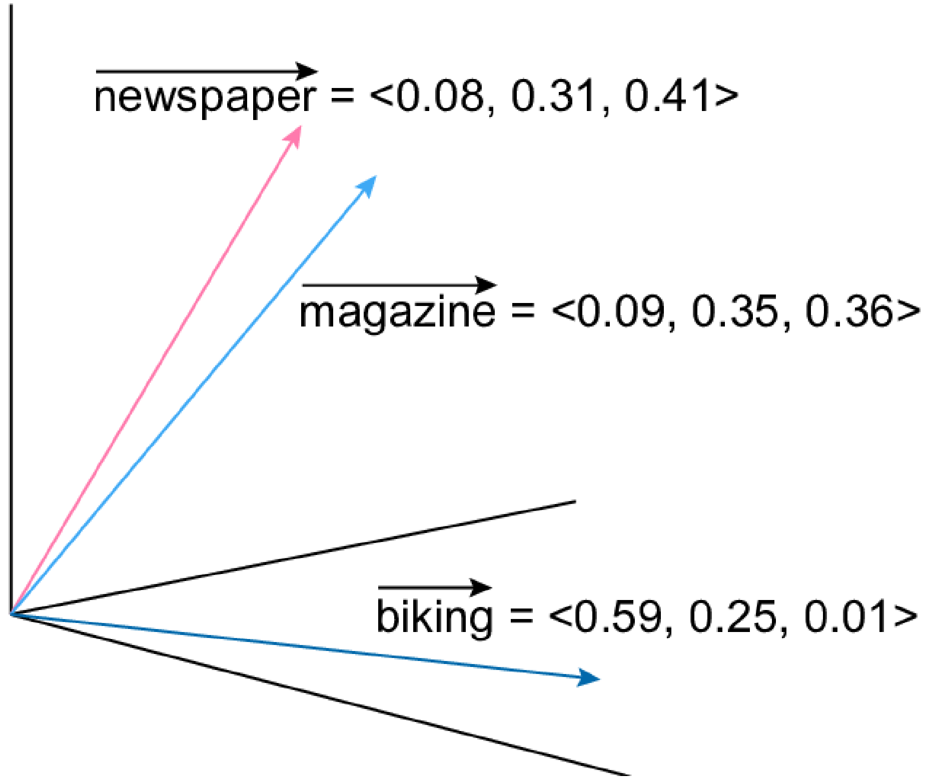
POSTECH
POHANG UNIVERSITY OF SCIENCE AND TECHNOLOGY

Word Embedding

INTRODUCTION

Word Vector

- What is the word vectors and why do we use them?
  - Word vector: A mapping of discrete words into vectors
  - We need **vector representations** for vector space models!



Word Vector

- One hot vector (Sparse representation)
  - Size of vector = |V|, where |V| is the size of vocabulary
  - “0” for all dims except for a single “1” for a specific dim to uniquely identify the word.

```
                  Paris
         Rome                                   word V
Rome   = [1, 0, 0, 0, 0, 0, ..., 0]
Paris  = [0, 1, 0, 0, 0, 0, ..., 0]
Italy  = [0, 0, 1, 0, 0, 0, ..., 0]
France = [0, 0, 0, 1, 0, 0, ..., 0]
```

Limitation of one hot vector

- Difficult to represent relations between words
  - For example, impossible to represent similarity
    - $(w_{Rome})^T w_{Paris} = 0$ (inner product = orthogonal)
  - In addition, impossible to distinguish homonyms (동음이의어)

영희가 철수에게 미안하다고 **사과**하면서
나무에서 갓 딴 맛있는 **사과**를 주었습니다

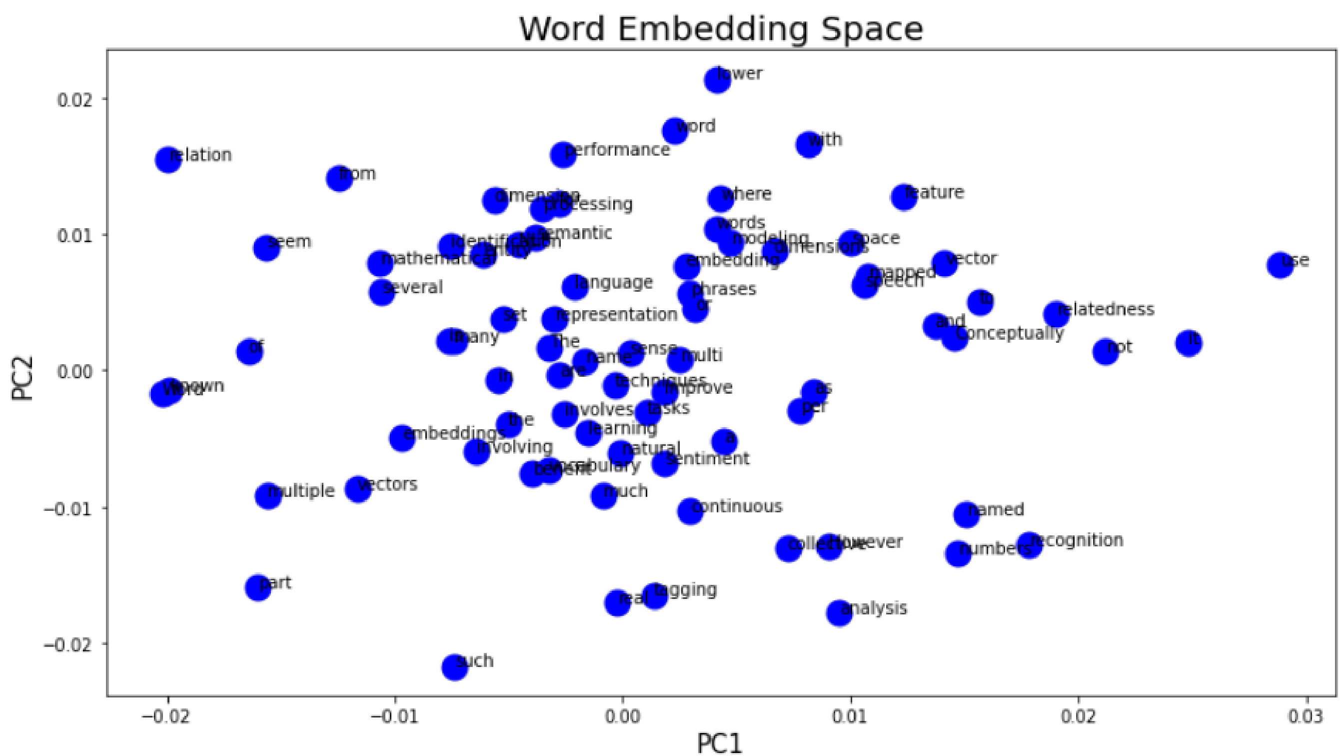
- 사과$_1$ = [0, 0, 0, 1, 0, 0, …, 0]
- 사과$_2$ = [0, 0, 0, 1, 0, 0, …, 0]
- 사과$_1$ = 사과$_2$

Limitation of one hot vector

- Computational inefficiency
  - Curse of dimensionality
    - Redundant space (0-valued)
    - The more words exist, the larger dimensions are needed,
      - leading to high computational cost.
  - No semantic information on words
    - Can model understand what the word means?

Word Embedding

- Word embedding vector
    - Representing words to "dense vector" (continuous space representation)

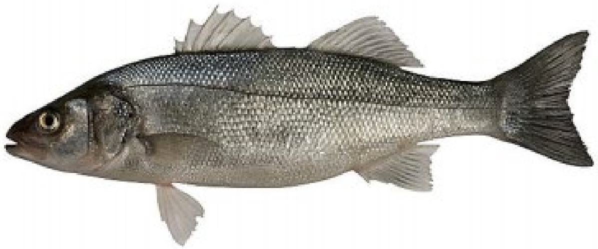


Word Embedding

- Embedding?
  - (Machine Learning)
    a mapping of a discrete (categorical) variable to a vector of continuous numbers [Toward Data Science)

Word Embedding

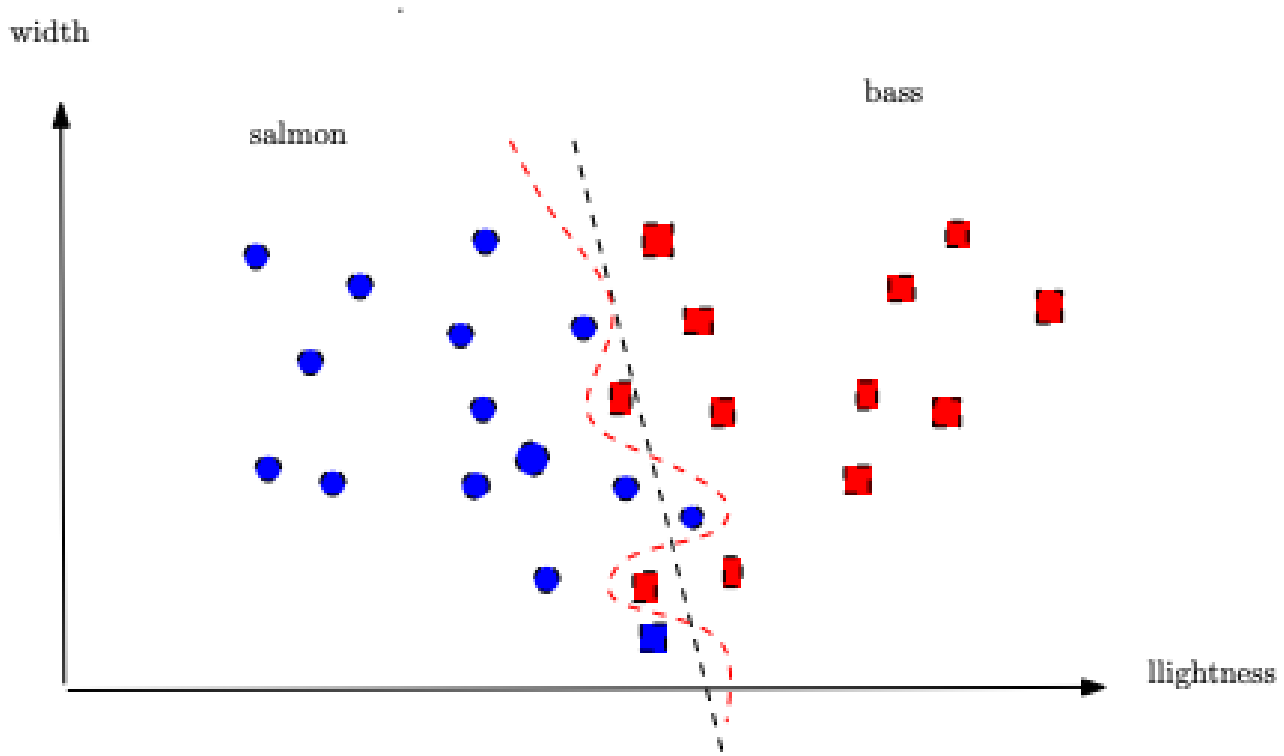
- Embedding? (Design method)
  - Simple example: representing salmon and bass
    - Manual design method → Use features
      – 크기, 너비, 밝기, 지느러미의 수 ... (Dimension)
      - 자질(Attribute) – 50cm, 12cm, 10, 4 ... (Component)



생선$_1$: [40, 12, 8, ...]
생선$_2$: [50, 15, 5, ...]
생선$_3$: [47, 10, 7, ...]
생선$_4$: [42, 15, 14, ...]
생선$_5$: [55, 19, 12, ...]

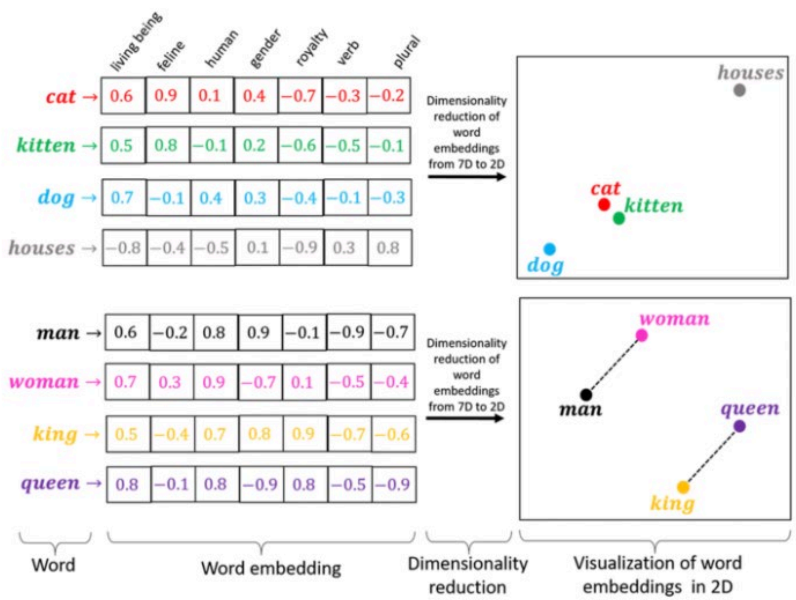
Word Embedding

- Embedding?



Word Embedding

- Word embedding
  - Representing words with features
    - For example,
    - Semantic and/or syntactic information
    - Statistics (word frequency)



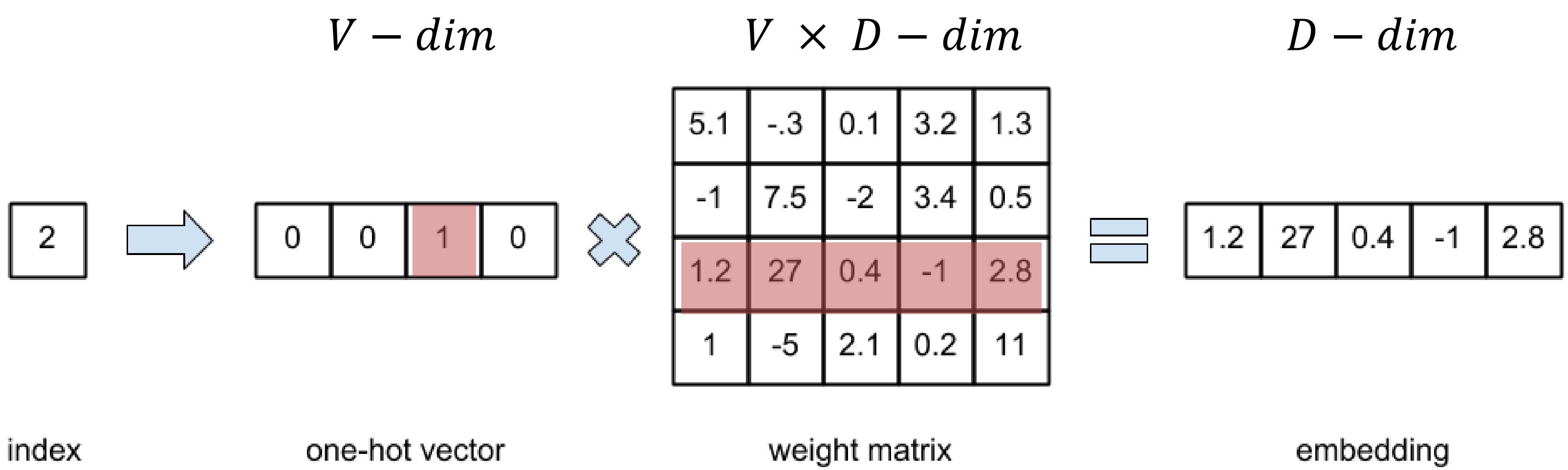
Word Embedding

- Limitation of feature-based construction
  - No standard consensus on what to extract for features
  - High human cost due to manual construction

- In neural approach
  - word vectors can be represented as model weights (trainable parameters)

Word Embedding

- Word embedding vectors



$$emb = X \times \boxed{W}$$

$$\text{where } X \in \mathbb{R}^{L \times V} \text{ is a set of sequences,}$$
$$\text{and } W \in \mathbb{R}^{V \times D} \text{ is a trainable weight matrix.}$$

실습 1

실습

- Goal
  - Pytorch에서 trainable embedding layer를 생성하여 단어가 주어졌을 경우 해당하는 embedding vector로 변환

- Steps
  1. Train data에서 dictionary 형태의 vocabulary 만들기
  2. nn.Embedding() 모듈을 활용하여 embedding layer 생성
  3. Weight 확인 및 word embedding 결과 vector 확인

torch.nn.Embedding

- Practices:
  - Step1: 주어진 단어에 대한 embedding 출력
  - Step2: 주어진 문장에 대한 embedding 출력
  - Step3: 주어진 Batch에 대한 embedding 출력

Word Embedding

WORD2VEC

Distributional Semantics

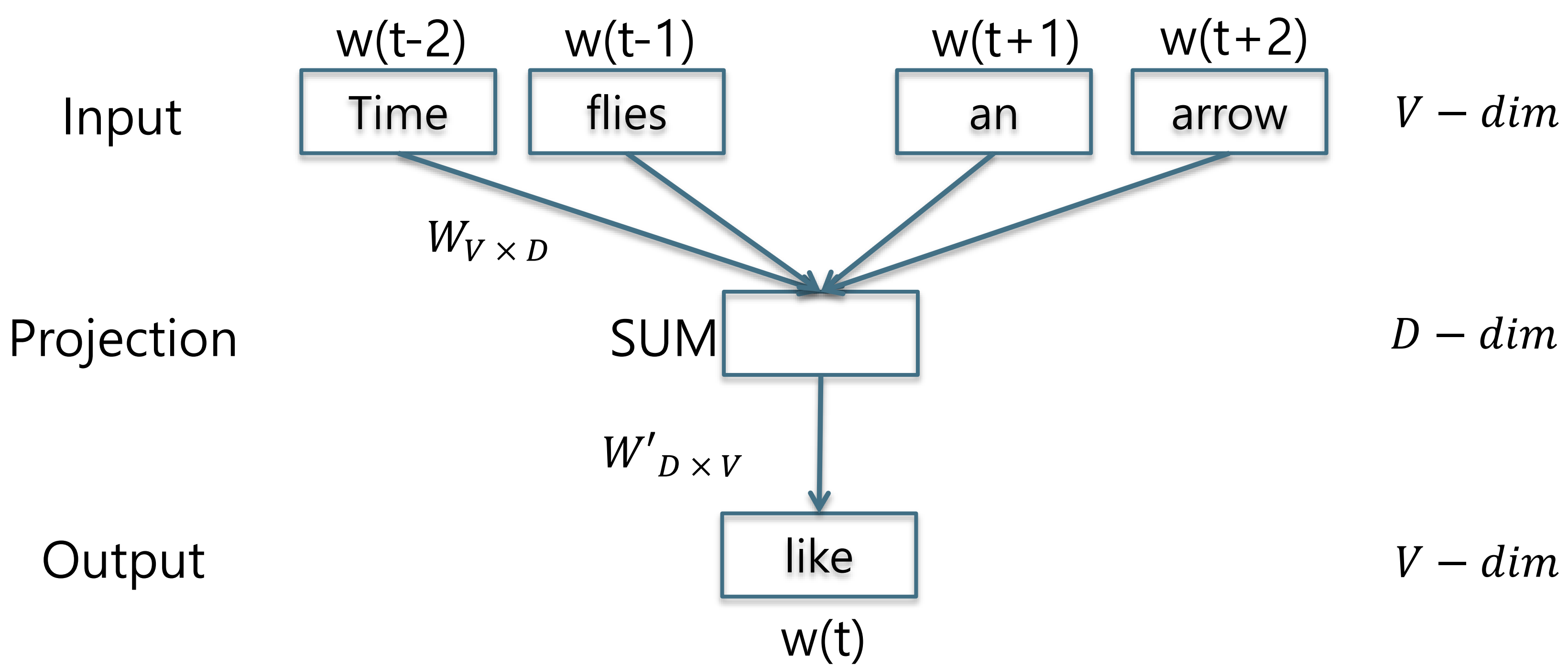
- Distributional Hypothesis
  - Words that are used and occur in the same context tend to purport similar meanings [Harris 1954]
  - Use word co-occurrence information
  - Approaches
    - Word2Vec
    - Fasttext
    - GloVe

Distributional Semantics

- Word2Vec
  - A two-layer neural network for word embeddings
  - Based on Distributional Hypothesis
    - Similar words highly occur in the same (similar) context
  - Training method
    - CBOW (Continuous Bag-Of-Words)
    - Skip-gram

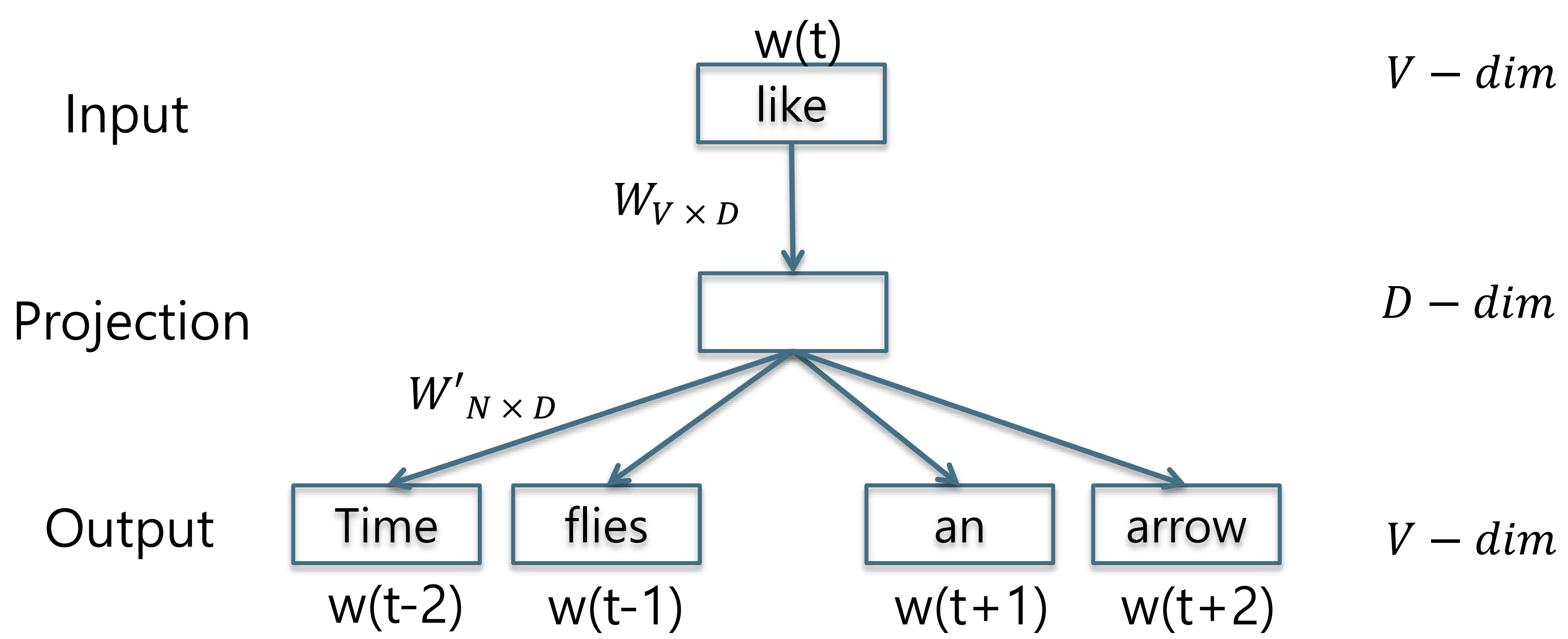
Distributional Semantics

- Continuous Bags of Words (CBOW)
  - Predicting a current (target) word from the surrounding words (context)



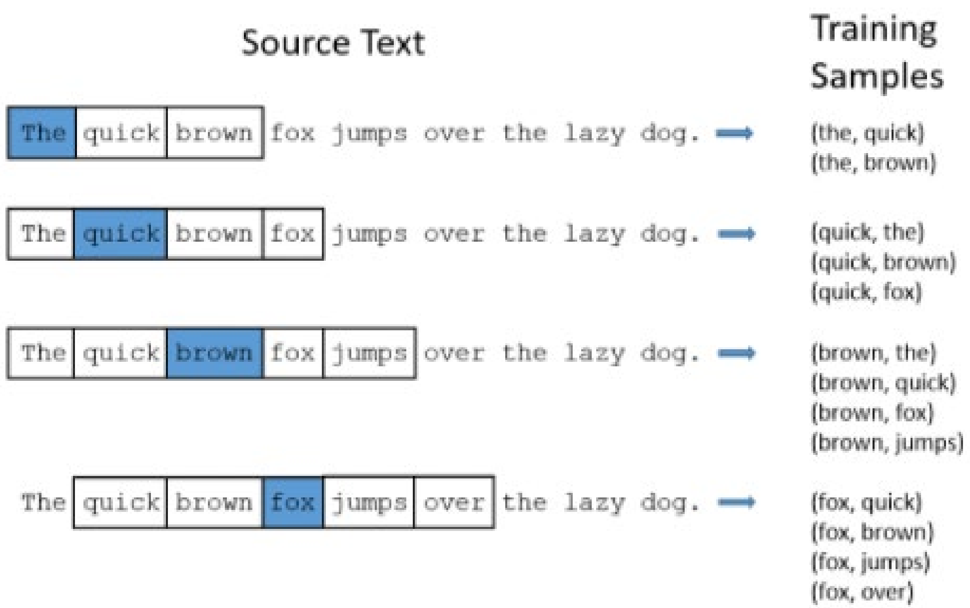
Distributional Semantics

- Skip-gram
  - Predicting the context words from the current word



Distributional Semantics

- Skip-gram



실습 2

Gensim

- Free Python library for statistical semantics
  - https://radimrehurek.com/gensim/index.html

Gensim

- Install Gensim
  - sudo pip install --upgrade gensim
  - conda install gensim

  - More information for install gensim
    - https://radimrehurek.com/gensim/install.html

Practice

- Step 1. Prepare the corpus for training
- Step 2. Train a Word2Vec model
- Step 3. Load the trained model
- Step 4. Get word similarity
- Step 5. Find the word further away from the mean
- Step 6. Find the top-N most similar words
- Step 7. Vector calculation

Practice

- Step 1. Prepare the corpus for training
  - Korean news corpus
    - Crawled from online news site
    - About 430k sentences, 160k morphemes
    - Morphologically segmented (No POS Tags)
    - Word frequency

Frequency	>= 1000	>= 700	>= 500	>= 300	>=100
Unique Word	1612	2175	2891	4250	9196

Practice

- Step 2. Train a Word2Vec model
  - model = gensim.models.Word2Vec(
    vector_size, window, sg, min_count, worker)
    - vector_size: the dimension of word vector, default = 100
    - window: the size of word window, default = 5
    - sg: 0 – CBOW / 1 – skip-gram, default = 0
    - min_count: threshold of word frequency, default = 5
    - worker: the number of thread for training, default = 1

Practice

- Step 2. Train a Word2Vec model
  - model.build_vocab(sentences)
    - sentences: text for training
  - model.train(sentences, total_examples, epochs)
  - model.save($model_name)
    - $model_name: file name of the saved model

Practice

- Step 3. Load the trained model
  - model = gensim.models.Word2Vec.load($model_path)
    - $model_path: location of trained model

Practice

- Step 4. Score the similarity between words
  - model.wv.similarity(*word1*, *word2*)
    - Score the similarity of *word1* and *word2*
- Examples
  - 한국 – 북한: 0.995
  - 노트북 – 컴퓨터: 0.994
  - 일본 – 도쿄: 0.987
  - 자동차 – 휘발유: 0.982
  - 임상실험 – 신약: 0.933
  - 파인애플 – 피자: 0.147

Practice

- Step 5. Find the word further away from the mean of all words.
  - model.wv.doesnt_match(*word_list*)
    - Returns the word further away from the mean of *word_list*

- Examples
  - 소프트웨어 하드웨어 컴퓨터 치약 – 치약
  - 국회 정부 정책 창문 – 창문
  - 버스 지하철 비행기 자가용 - 자가용

Practice

- Step 6. Find the top-N most similar words
  - model.wv.most_similar(positive=[*word*])
  - Print 10 most similar words

Practice

- Step 7. Find the top-N most similar words with combination of words
  - model.wv.most_similar(positive=[*words*], negative=[*words*], topn= *1*)
    - positive / negative: (pos/neg) words for the calculation
    - topn: # of the most similar words

- Example
  - Find the most similar word with the result of [a – b + c]
  - 대통령 – 한국 + 미국: 부시
  - https://word2vec.kr/search

Practice

- More information
  - https://radimrehurek.com/gensim/models/word2vec.html
  - https://radimrehurek.com/gensim/models/keyedvectors.html
  - https://radimrehurek.com/gensim/auto_examples/index.html

실습 3

Practice

- gensim으로 학습된 embedding을 이용한 torch.nn.Embedding 초기화

Q & A