

# Word2Vec & t-SNE

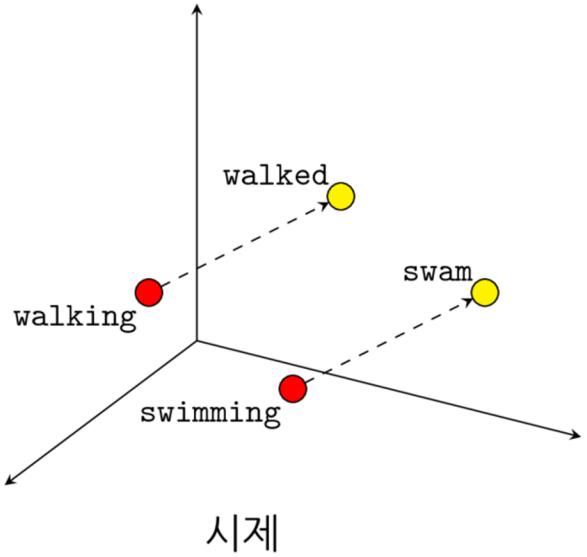
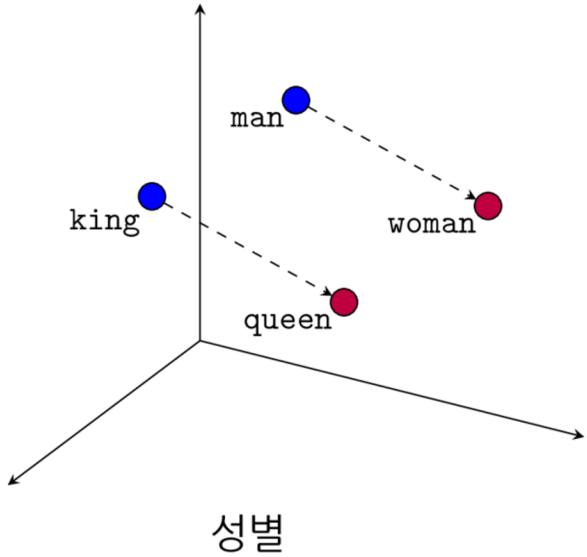
# Word2Vec

- Word Embedding 중 하나
  - Word Embedding이란?  
단어가 가진 의미를 그대로 보존하면서 의미와 맥락을 고려하여 단어를 벡터로 표현하는 것  
즉, 개별 단어가 가진 속성, 단어의 의미를 그대로 보존하면서  
실수값으로 채워진 벡터로 표현하는 것
- Word2Vec은 단어와 단어 사이의 관계를 벡터 사이의 연산관계로 환원하여 측정할 수 있도록 하며, 단어 사이의 의미 관계에 대해 쉽게 일반화 할 수 있다.

# Word2Vec 평가방식

단어 임베딩은 2가지에 의해 평가 받는다.

- 유사한 벡터를 가진 단어가 과연 유사한 의미를 갖는지 여부
- 단어를 재현하는 공간에서 거리가 얼마나 의미가 있는지 여부



Spain	Madrid
Italy	Rome
Germany	Berlin
Turkey	Ankara
Russia	Moscow
Canada	Ottawa
Japan	Tokyo
Vietnam	Hanoi
China	Beijing

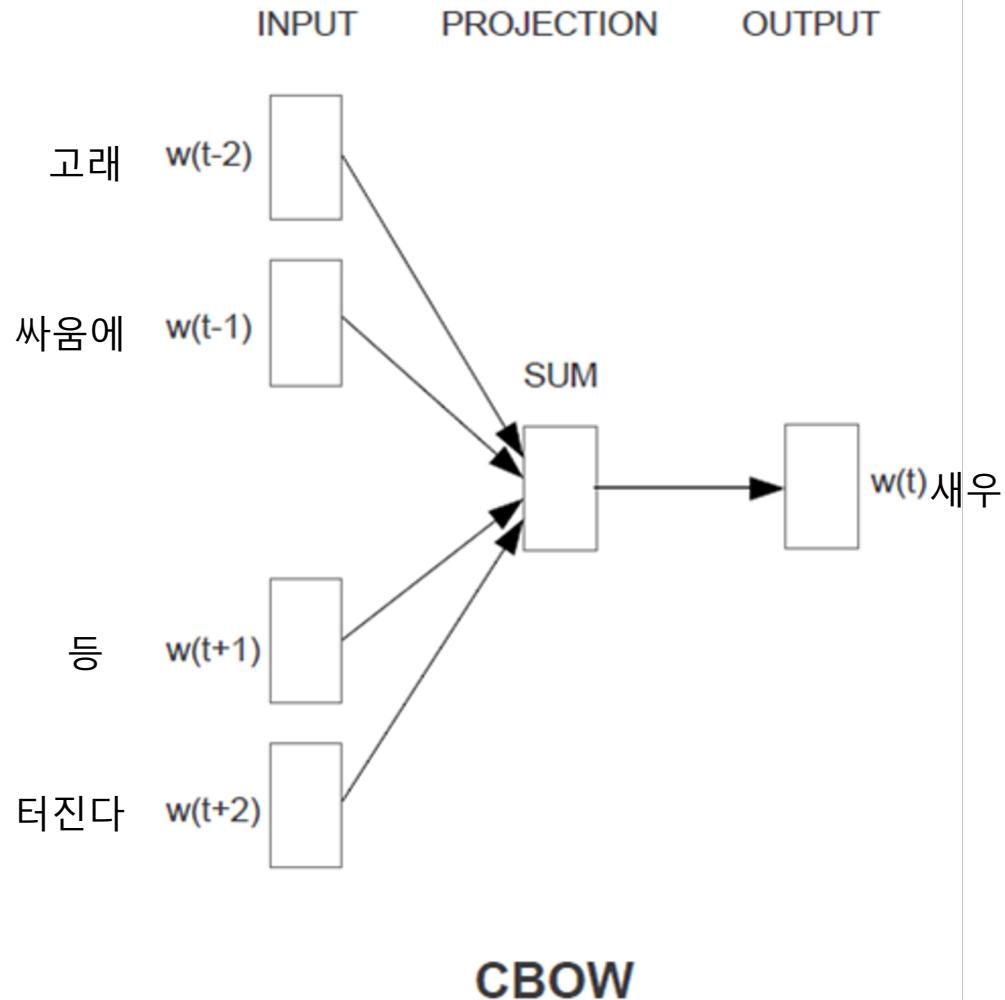
국가-수도

# Word2Vec 방법

- CBOW  
주변 단어로 중심으로 중심 단어를 예측하도록 모델을 구축
- Skip-gram  
중심 단어로 주변 단어를 예측하도록 모델을 구축  
반복학습을 더 많이 하기에 더 정확한 경우가 많아 최근에는 Skip-gram이 더 널리 쓰이고 있다.

# Word2Vec - CBOW

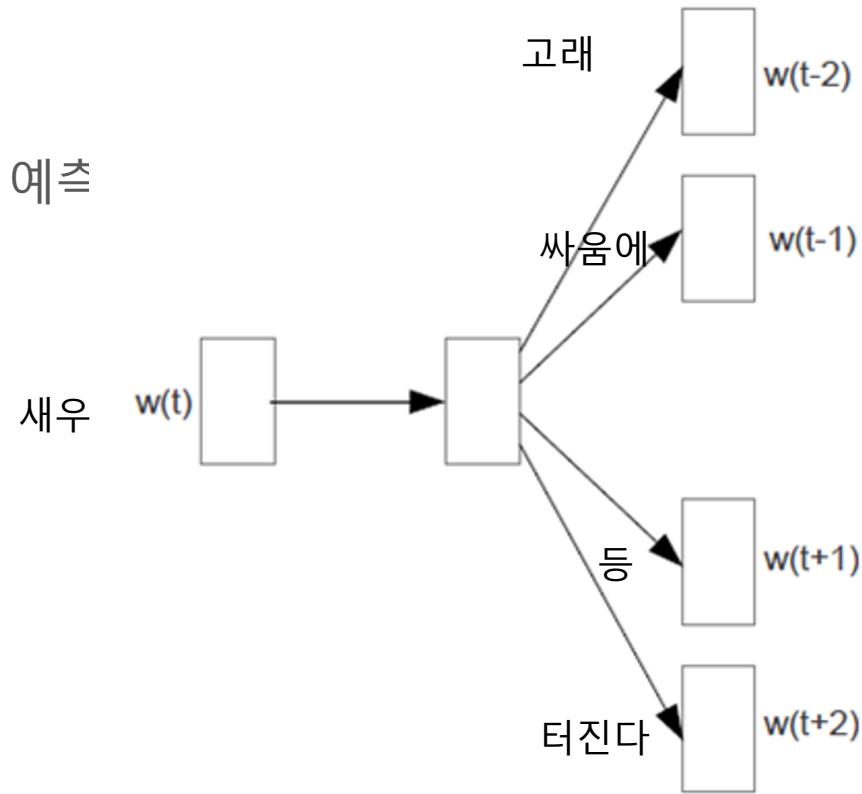
여러 단어의 맥락이 주어진 상태에서  
이후에 나올 단어를 예측하는 방식



INPUT PROJECTION OUTPUT

# Word2Vec - Skip-gram

특정 단어로부터 주변에 올 수 있는 단어를 예측



Skip-gram

# Word2Vec - Parameter

```
from gensim.models import Word2Vec

word2vec_model = Word2Vec(
    word2vec_corpus,
    size=100,
    alpha=0.025,
    window=5,
    min_count=5,
    sg=0,
    negative=5)
```

그 외에 신경써야 할 arguments에 대하여 알아봅니다. size는 임베딩 벡터의 크기입니다. 아주 작은 수준만 아니라면 벡터의 차원이 커진다고 학습의 경향이 달라지진 않습니다. 적당히 큰 숫자면 충분합니다. alpha는 learning rate입니다. 기본값 쓰셔도 됩니다. window는 스냅샷의 크기입니다. "a little, (cat), sit, on"의 windows는 2입니다. 앞 뒤로 고려하는 단어의 개수입니다. Word2Vec에서는 2 ~ 5 정도면 비슷한 경향을 보입니다. min count는 데이터에서 등장하는 단어의 최소빈도수입니다. Word2Vec은 자주 등장하지 않은 단어에 대해서는 제대로 학습이 이뤄지지 않습니다. 또한 min\_count가 작으면 모델에 지나치게 많은 단어가 포함되어 모델의 크기가 매우 커집니다. sg는 skip-gram 옵션입니다. Word2Vec의 구조를 skip-gram, cbow 두 가지로 설명하는데, 이는 포스트에서 언급하지 않았습니다. negative는 negative samples의 개수입니다. 이 역시 기본값 쓰셔도 됩니다. 클수록 학습속도가 느려집니다. 하지만 지나치게 작으면 학습이 제대로 이뤄지지 않습니다.

# t-SNE

word2vec을 사용하여

생성된 벡터들 사이의 연산을 통해 표시된 단어 유사도를

t-SNE 알고리즘을 이용하여 시각화할 수 있다.

t-SNE는 벡터 시각화를 위해 사용되는 알고리즘으로,

고차원 공간에서 유사한 두 벡터가

저차원 공간에서도 유사하도록 차원을 축소하여

단어들 간의 유사도를 그림으로 보여준다.

# 진행사항

```
def preprocess(doc_list):
    okt = Okt()
    print(okt.tagset)
    #result_doclist = []
    result_dict = []
    for index in range(0, len(doc_list)):
        remove_file_enc = re.compile(r'<[^>]+>')
        remove_special_char = re.compile(r"[^가-힣^.^?^.!]")

        # 한글, 기본 기호를 제외한 문자 제거하기(일단, 전체 문서가 한국어로만 되어 있다고 가정한다.) 이때, 특수기호 및 \t, \n, 구두점도 같이 제거된다.
        text = remove_file_enc.sub('', doc_list[index])
        text = remove_special_char.sub(' ', text)

        # 형태소 분석하기 (이때, norm = true, stem = true)
        morphs_texts= okt.pos(text, norm=True, stem=True)

        # stopwords 사전
        stop_words = [".","","","","!","가까스로","가령","각","각각","각자","각종","갖고말하자면","같다","같이","개의치않고","거니와","거바","거의","것","것과 같아","것들","계다가","개우디"]

        result = []
        for token, tag in morphs_texts:
            # stopwords 제외 and 김탄사, 조사, 어미 제외
            #print(token, tag)
            if token not in stop_words and tag in ['Noun', 'Verb', 'Adjective', 'Determiner', 'Adverb', 'Conjunction', 'Suffix']:
                result.append(token)

        result_dict.append(result)
    print(index)
```

# 진행사항

---

```
1  #-*- coding: utf-8 -*-  
2  
3  import pickle  
4  with open('/home/hyeyoung/NKDB/data/total_morphs_list.txt', 'rb') as f:  
5      corpus = pickle.load(f) # 단 한줄씩 읽어옴  
6  
7  # word2vec 모델 학습  
8  from gensim.models import word2vec  
9  
10 data = corpus  
11 model = word2vec.Word2Vec(data, size = 100, window = 5, min_count=5, workers = 5, sg=0) #CBOW  
12 # 100차원 벡터,  
13 # 출현 빈도는 5개 미만은 제외  
14 # 분석 방법론은 CBOW을 선택  
15  
16 model.save("/home/hyeyoung/NKDB/model/CBOW_model.model")
```

---

# 진행사항

```
from gensim.models import Word2Vec
from sklearn.manifold import TSNE
import matplotlib.pyplot as plt

model = Word2Vec.load("/home/hye young/NKDB/model/CBOW_model.model")

def tsne_plot(model):
    labels = []
    tokens = []

    for word in model.wv.vocab:
        tokens.append(model[word])
        labels.append(word)

    tsne_model = TSNE(perplexity=40, n_components=2, init='pca', n_iter=2500, random_state=23)
    new_values = tsne_model.fit_transform(tokens)

    x = []
    y = []
    for value in new_values:
        x.append(value[0])
        y.append(value[1])

    plt.figure(figsize=(16, 16))
    for i in range(len(x)):
        plt.scatter(x[i], y[i])
        plt.annotate(labels[i],
                    xy=(x[i], y[i]),
                    xytext=(5, 2),
                    textcoords='offset points',
                    ha='right',
                    va='bottom')
    plt.show()

tsne_plot(model)
```

# 결과

```
(py37) hyeyoung@dwb:~/NKDB/NKDB/topicmodel$ python makeDTM2.py  
TfidfModel(num_docs=13820, num_nnz=23963180)  
(py37) hyeyoung@dwb:~/NKDB/NKDB/topicmodel$ █
```

# 결과 CBOW

```
associate_word2 = model.wv.most_similar(positive=["문재인", "북한"], negative=["남한"], topn=1)
print(associate_word2)
print(associate_word2[0])
print()

print(model.wv.most_similar(positive=['서울', '일본'], negative=['한국']))
print()

print(model.wv.most_similar(positive=['왕', '남자'], negative=['여자']))

print(model.wv.most_similar(positive=["김대중", "북한"], negative=["남한"], topn=1))
```

```
[py37] hyeyoung@dwb:~/NKDB/NKDB/topicmodel$ python associated_word2.py
[('당 선 인', 0.8088319897651672), ('전 대 통령', 0.7138012647628784), ('팅 선 자', 0.6486227512359619), ('행 정 부', 0.6470855474472046), ('정 부', 0.6306924819946289), ('총 리', 0.6302433609962463), ('대 통', 0.6064052581787109), ('대 통령', 0.5949649214744568), ('통령', 0.5851585865020752), ('힐 러 리', 0.5835388898849487)]
('당 선 인', 0.8088319897651672)

[('이 명 박', 0.6136453151702881)]
('이 명 박', 0.6136453151702881)

[('도 쿄', 0.5622795820236206), ('김 기 석', 0.4846559762954712), ('유 키', 0.46907907724380493), ('삿 포 로', 0.458246111869812), ('유 키 오', 0.4318702220916748), ('리 저', 0.4273390769958496), ('나 고 야', 0.42546963691711426), ('김 학 준', 0.4245833456516266), ('요 시 히 코', 0.4175368547439575), ('나 카 가 와', 0.4157060384750366)]

[('리 자 오 싱', 0.5661404132843018), ('타 오', 0.5602253675460815), ('즈 췈', 0.5486589670181274), ('양 제 츠', 0.5369718074798584), ('러 우', 0.5308992266654968), ('류 치', 0.5222364068031311), ('랑 광 례', 0.5204468369483948), ('명 젠 주', 0.5190800428390503), ('탕 자 쉬', 0.5154823660850525), ('쑹', 0.5126700401306152)]
('노 무 현', 0.6484166383743286)
```

# 결과 Skip-gram

```
associate_word2 = model.wv.most_similar(positive=["문재인", "북한"], negative=["남한"], topn=1)
print(associate_word2)
print(associate_word2[0])
print()

print(model.wv.most_similar(positive=['서울', '일본'], negative=['한국']))
print()

print(model.wv.most_similar(positive=['왕', '남자'], negative=['여자']))

print(model.wv.most_similar(positive=["김대중", "북한"], negative=["남한"], topn=1))
```

```
[('당선인', 0.8027415871620178), ('박근혜', 0.7920651435852051), ('전대통령', 0.7618861794471741), ('벼락', 0.7545623779296875), ('푸틴', 0.7466790676116943), ('취임', 0.7452720403671265), ('트럼프', 0.7361042499542236), ('메드베데프', 0.734591543674469), ('이명박', 0.7343828678131104), ('조지부시', 0.7302359938621521)]
('당선인', 0.8027415871620178)
```

```
[('박근혜', 0.7102257013320923)]
('박근혜', 0.7102257013320923)
```

```
[('한울', 0.6155502796173096), ('오사카', 0.612023115158081), ('법문사', 0.6115644574165344), ('도쿄', 0.5941550731658936), ('과학사', 0.5795176029205322), ('이수광', 0.5784896612167358), ('조이선교회', 0.574654407501221), ('견지동', 0.5753879547119141), ('란코브', 0.5709304809570312), ('정정호', 0.5684270858764648)]
```

```
[('우왕', 0.5969898700714111), ('쑨수센', 0.5745290517807007), ('자오커스', 0.572936475276947), ('청궈펑', 0.571307897567749), ('정찌광', 0.5698022842407227), ('명홍', 0.5696694254875183), ('후청', 0.5663501024246216), ('왕래', 0.5653870701789856), ('천스쥐', 0.5623260736465454), ('펑후', 0.5618948936462402)]
('노무현', 0.7035806179046631)]
```

# 추후 할 일 및 질문

## ▶ 추후 할 일

gridcv 사용해서 적합한 파라미터 찾아 학습 잘 시키기

t-SNE 사용해 시각화 하기

## ▶ 질문

word embedding 성능 비교 이외에도 있는가?

- 유사한 벡터를 가진 단어가 과연 유사한 의미를 갖는지 여부
- 단어를 재현하는 공간에서 거리가 얼마나 의미가 있는지 여부

형태소 분석 userDictionary 어떻게 할것인가?

Thank you