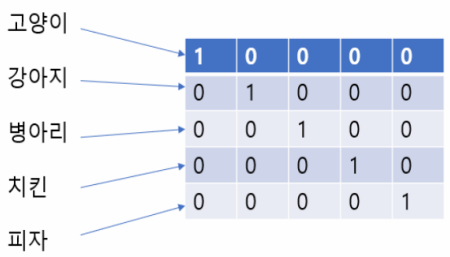
Embedding이란? (Word Embedding)

- 앞에서 살펴보았던 vectorization의 치명적인 단점은 바로 단어나 문장들 사이의 관계에 대해서 설명하지 못한다는 것입니다. 다음과 같은 예시를 들어보겠습니다.



- one hot encoding을 통해 각 5개의 토큰들이 고유의 벡터를 갖게 되었습니다. 하지만 뭔가 이상하지 않나요? 사람이 보기에는 5개의 토큰들이 너무나도 명확하게 구분 됩니다. 지금까지 살펴 본 벡터화 방법들은 단어의 중요도나 문서 안에서의 중요도는 구분 할 수 있지만 단어 사이의 유사도는 구별할 수 없었습니다.

↑ 단어간의 그저 유사도는 '0'이됨.

이 때 사용되는 것이 Embedding 기법이며 word2vec을 비롯한 다양한 임베딩 기법들이 존재합니다. 대략적인 아이디어는 다음과 같습니다.

의미가 유사한 토큰들은 가깝게 임의의 차원에 뿌려보자!!

- 위의 예시처럼 비슷한 의미를 내포하고 있는 토큰들은 서로 가깝게, 그렇지 않은 토큰들은 서로 멀리 뿌리도록 하는 것이 임베딩의 목적입니다. 검색 시스템, 감성 분석 등에서는 훌륭한 임베딩을 수행하는 것이 전체 문제 해결에 많은 영향을 줍니다.

- 임베딩 또한 하나의 모델을 의미하며 훈련이 필요합니다. 데이터가 충분하고 시간이 많으면 소지한 데이터에 특화된 임베딩 모델을 학습시킬 수 있습니다. 보통은 pre_trained embedding model을 가져와서 사용합니다.

1.Keras Embedding Layer

- 기본적으로 가장 쉽고 빠르게 네트워크 모델에 임베딩 층을 주입할 수 있는 방식입니다. 이 방법은 무작위로 특정 차원으로 입력 벡터들을 뿌린 후 학습을 통해 가중치들을 조정해 나가는 방식입니다. 즉, 단어 사이의 관계를 반영하는 방법이 아닙니다.
- 적용은 아래와 같이 매우 간단하게 keras 코드로 구현할 수 있습니다.

```
model = Sequential()
model.add(Embedding(vocab_size, 128, input_length = max_len))
```

2. word2vec

- word2vec의 핵심 아이디어는 "친구를 보면 그 사람을 알 수 있다"입니다. 주변 단어와의 관계를 통해 해당 단어의 의미적 특성을 파악합니다. 자세한 내용은 다음 링크를 참조해 주세요.(word2vec)
- word2vec embedding matrix를 kears의 embedding에 주입하는 과정은 다음과 같습니다.

1. 구글의 사전 훈련된 word2vec bin 파일을 다운로드 합니다.(다운로드 링크)

↓

2. gensim 모듈과 bin파일을 활용해 word2vec 모델을 로드합니다.

```
import gensim
word2vec = gensim.models.KeyedVectors.load_word2vec_format('GoogleNews-vectors-negative300.bin.gz', binary = True)
```

↓

3. vocabulary에 있는 토큰들의 벡터를 가져와 embedding matrix에 저장합니다.

```
embedding_matrix = np.zeros((vocab_size, 300)) #300차원의 임베딩 매트릭스 생성

for index, word in enumerate(vocabulary): #vocabulary에 있는 토큰들을 하나씩 넘겨줍니다.
    if word in word2vec: #넘겨 받은 토큰이 word2vec에 존재하면(이미 훈련이 된 토큰이라는 뜻)
        embedding_vector = word2vec[word] #해당 토큰에 해당하는 vector를 불러오고
        embedding_mxtrix[i] = embedding_vector #해당 위치의 embedding_mxtrix에 저장합니다.
    else:
        print("word2vec에 없는 단어입니다.")
        break
```

↓

4. keras embedding layer에 embedding_matrix를 가중치로 주어 이용합니다.

```
model = Sequential()
model.add(Embedding(vocab_size, 300,weights = [embedding_matrx], input_length = max_len))
```