



# 10. 자연어 처리를 위한 임베딩

## 10.1 임베딩 기법

### 10.1.1 희소 표현 기반 임베딩

- **희소 표현 (Sparse Representation)**

대부분의 값이 0으로 채워져 있는 벡터 표현. 원-핫 인코딩이 대표적.

- **원-핫 인코딩**

- 단어를 N차원의 벡터로 표현. 특정 단어에 해당하는 위치에만 1, 나머지는 0.예:

`[calm, fast, cat]` -> `fast` 는 `[0, 1, 0]`.

- **단점:**

1. 단어 간 관계성(유의어, 반의어 등) 없음 → 벡터들이 직교(독립적).
2. 차원의 저주: 단어 개수만큼 차원이 증가. 예: 단어 10만 개 → 벡터 차원 10만.

- **대안:** 신경망 기반 단어 벡터화 기법

예: Word2Vec, GloVe, FastText.

---

### 10.1.2 횡수 기반 임베딩

- **카운터 벡터 (Count Vector)**

- 단어 출현 빈도를 기반으로 벡터화.
- 구현: 사이킷런의 `CountVectorizer()` 사용.
  - 문서 토큰화 → 빈도 계산 → 벡터 변환.

- **TF-IDF (Term Frequency-Inverse Document Frequency)**

- 단어의 중요도를 계산하여 가중치 부여.
- 구성 요소:
  - **TF:** 특정 문서 내 단어 출현 빈도.

- **IDF**: 단어가 여러 문서에 등장하는 빈도에 따라 가중치 조정.
  - 공통 단어(a, the 등)의 가중치 감소.
  - 로그 스무딩 적용: 분모에 1 추가.
- 사용 사례:
  - 검색 엔진, 키워드 분석, 검색 결과 순위 결정.

### 10.1.3 예측 기반 임베딩

- **Word2Vec**

- 신경망 기반 임베딩. 문맥 내 단어 관계성을 학습.
- **방법론**:
  1. **CBOW (Continuous Bag of Words)**: 문맥으로 중심 단어를 예측. 예: "calm  
cat on the sofa" → 중심 단어 "slept" 예측.
  2. **Skip-Gram**: 중심 단어로 주변 단어를 예측. 예: 중심 단어 "slept" → "calm",  
"on" 등 문맥 단어 예측.

- **패스트텍스트 (FastText)**

- Word2Vec 단점 보완 (사전에 없는 단어 처리).
- **n-그램 사용**: 단어를 n-그램으로 분리해 학습. 예: "Deep Learning Book" → ["Deep  
Learning", "Learning Book"] .

### 10.1.4 횡수/예측 기반 임베딩

- **GloVe (Global Vectors for Word Representation)**

- 횡수 기반(LSA) + 예측 기반(Skip-Gram) 방식 결합.
- 단어 간 글로벌 동시 발생 확률을 활용.
- 단어 간 통계적 관계 표현.

## 10.2 트랜스포머와 어텐션

### 10.2.1 어텐션 메커니즘

- **어텐션**:

인코더와 디코더 간 정보를 효율적으로 전달. 중요한 벡터에 집중.

- 소프트맥스 함수로 가중치 계산 → 컨텍스트 벡터 생성.

- **과정:**

1. **어텐션 스코어:** 인코더 은닉 상태와 디코더 은닉 상태의 관련성 계산.
  2. **시간 가중치 계산:** 소프트맥스 적용.
  3. **컨텍스트 벡터 생성:** 시간 가중치와 은닉 상태의 가중합.
  4. **디코더 은닉 상태 갱신:** 컨텍스트 벡터, 이전 은닉 상태로 출력 계산.
- 

### 10.2.2 트랜스포머

- 어텐션 메커니즘 확장. 인코더와 디코더를 여러 층으로 구성.

- **인코더:**

- **셀프 어텐션:** 단어 간 관계 계산.
- **전방향 신경망:** 셀프 어텐션 결과 전달.

- **디코더:**

1. 셀프 어텐션.
  2. 인코더-디코더 어텐션.
  3. 전방향 신경망.
- 

### 10.2.3 seq2seq 모델

- **목적:** 입력 시퀀스 → 출력 시퀀스 변환 (예: 번역).

- **특징:**

- 입력, 출력 시퀀스 길이 달라도 가능.
- 번역처럼 입력-출력 간 의미가 중요할 때 사용.