[1]:
```python
from sentence_transformers import SentenceTransformer
import numpy as np

# 문장 임베딩
model = SentenceTransformer('all-MiniLM-L6-v2')
words = [
    "Paris",
    "Berlin",
    "Beijing",
    "Seoul"
]
embeddings = model.encode(words)

# 첫 번째 문장을 Query로, 나머지를 Key로 설정
query = embeddings[0].reshape(1, -1)
keys = np.vstack(embeddings)

# dot product 기반 유사도 계산
scores = query @ keys.T
print("Dot product 유사도:", scores)
```

[그림1] 사용단어와 코딩 예시

```
Dot product 유사도: [[1.         0.5681463  0.5181803  0.51473486]]
```

[그림2] “Paris”를 기준으로 각 단어와의 내적 의미 유사도

1. 문장 임베딩 및 dot product 유사도 실험결과: "Paris"를 기준으로 한 내적 유사도

"Paris"와 다른 단어(예: "Beijing", "Seoul")의 내적 유사도가 0.5점대로 나온 이유는 이 단어들이 모두 수도라는 공통된 의미를 가지기 때문입니다.

- 예시:
  - "Paris"와 "Seoul"은 모두 국가의 수도이자 문화적 중심지입니다.
  - 문장 임베딩 모델은 단어의 맥락을 벡터로 변환하는데, 비슷한 맥락을 가진 단어는 벡터 공간에서 가까이 위치합니다.
  - 내적 값이 높을수록 두 벡터가 비슷한 방향과 크기를 가진다는 의미로, "수도"라는 공통 특성이 반영된 결과입니다.

2. 내적(dot product)의 기하학적 정의

내적은 두 벡터의 방향과 크기를 함께 고려하는 계산법입니다.

간단한 예시

- 벡터 A = (3, 4), 벡터 B = (1, 2)라면:
  내적 = (3×1) + (4×2) = 3 + 8 = 11
- 기하학적 의미: 두 벡터가 같은 방향일수록, 크기가 클수록 내적 값이 커집니다.

3. 코사인 유사도 vs 내적 유사도 차이점

구분 코사인 유사도 내적 유사도

계산법	벡터의 각도만 고려 (방향)	벡터의 각도 + 크기 모두 고려
사용처	문서 비교 (길이 무관)	주의력 메커니즘(Attention)
예시	"고양이" vs "강아지" (유사)	"고양이" vs "고양이!!!!" (크기가 다름)

- 코사인 유사도: 두 벡터의 방향만 비교합니다. 예를 들어 "고양이"와 "강아지"는 방향이 비슷해 유사도가 높습니다.
- 내적 유사도: 방향뿐 아니라 벡터의 크기도 반영합니다. "고양이"와 "고양이!!!!"는 같은 방향이지만, "!!!!"가 추가된 문장은 벡터 크기가 더 커서 내적 값이 높아집니다.

4. Attention에서 내적을 사용하는 이유

주의력 메커니즘(Attention)은 벡터의 크기가 중요한 정보로 작용하기 때문에 내적을 사용합니다.

- 예시:
  - "중요한 단어"는 벡터 크기가 큽니다(예: "화재!!!").
  - 내적을 사용하면 크기가 큰 벡터가 더 강하게 반영되어, 모델이 중요한 단어에 집중할 수 있습니다.
- 코사인 유사도의 한계: 크기를 무시하기 때문에 "화재"와 "화재!!!"를 구별하지 못합니다.

"시험 공부할 때 '매우 중요한 개념'은 크게 표시하고, '덜 중요한 개념'은 작게 표시한다고 생각해보세요. 내적은 이 '크기 표시'까지 고려해서 유사도를 계산하는 거예요!"

5. 요약

1. 내적은 방향 + 크기를 모두 반영하므로, Attention처럼 "중요도"를 구별해야 할 때 유용합니다.
2. 코사인 유사도는 순수히 의미적 유사성(방향)만 비교할 때 적합합니다.
3. "Paris" 실험에서 내적 유사도가 높게 나온 것은 단어들이 공통된 의미(수도)를 가지면서 벡터 크기도 비슷하기 때문입니다.