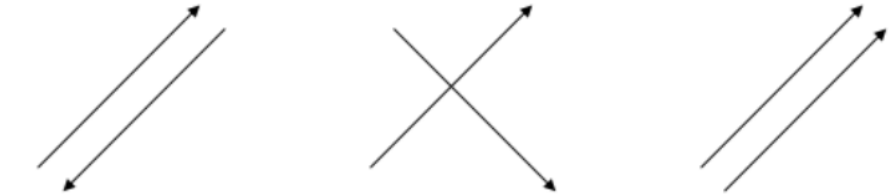
코사인 유사도(Cosine Similarity)

- 두 벡터 간의 코사인 각도를 이용하여 구할 수 있는 두 벡터의 유사도
- 두 벡터의 방향이 동일한 경우에는 1의 값을 가진다.
  - 90도의 각을 이루면 0
  - 180도로 반대의 방향을 가지면 -1
- 즉, 코사인 유사도는 -1 이상 1 이하의 값을 가진다.
- 값이 1에 가까울수록 유사도가 높다고 판단할 수 있다.



$$similarity = cos(\Theta) = \frac{A \cdot B}{||A||\ ||B||} = \frac{\sum_{i=1}^{n} A_i \times B_i}{\sqrt{\sum_{i=1}^{n}(A_i)^2} \times \sqrt{\sum_{i=1}^{n}(B_i)^2}}$$

- 식 끝에 있는 시그마 식은 벡터의 크기와 벡터의 dot production을 구하는 공식
- 문서의 유사도를 구하는 경우 문서 단어 행렬이나 TF-IDF 행렬이 각각의 특징 벡터 A, B 가 된다.
  - 예시
  - 문서 1 : 저는 사과가 좋아요
  - 문서 2 : 저는 바나나가 좋아요
  - 문서 3 : 저는 바나나가 좋아요 저는 바나나가 좋아요
  - 문서 딘어 행렬

	바나나	사과	저는	좋아요
문서1	0	1	1	1
문서2	1	0	1	1
문서3	2	0	2	2

```
문서 1과 문서2의 유사도 : 0.67
문서 1과 문서3의 유사도 : 0.67
문서 2과 문서3의 유사도 : 1.00
```

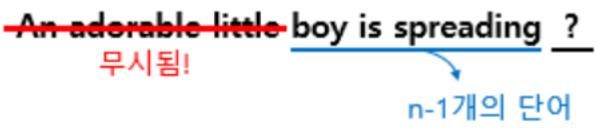
- 문서 1과 문서 2의 코사인 유사도와 문서 1과 문서 3의 코사인 유사도가 같음
- 문서 2와 문서 3의 코사인 유사도는 1이 나온다.
- 문서 3은 문서 2에서 모든 단어의 빈도 수가 1씩 증가함
  - 한 문서 내의 모든 단어의 빈도수가 동일하게 증가하는 경우 기존 문서와 코사인 유사도의 값이 1이다.
  - 예를 들어 문서 A와 문서 B, 문서 C가 있을 때 A와 B는 같은 주제의 문서이고 문서 C는 다른 주제의 문서일때
    - 문서 A와 C의 문서의 길이는 거의 동일하고 문서 A와 B의 길이는 문서 B가 2배 길 경우
    - 유클리드 거리로 유사도를 연산한다면 문서 A가 문서 B보다 문서 C와 더 유사하다고 결과가 출력될 수 있다.
    - 이는 유사도 연산에 문서의 길이가 영향을 받음을 시사함
    - 이 경우 코사인 유사도가 해결책이 된다.
    - 코사인 유사도는 벡터의 방향(패턴)에 영향을 받기 때문

N-gram 언어 모델

- N-gram 언어 모델은 단어가 나올 확률을 구할 때 기준 단어의 앞 단어를 전부 포함해서 카운트하는 것이 아닌 앞 단어 중 임의의 개수만 포함해서 카운트하는 것
    - 이렇게 하면 해당 단어의 시퀀스를 카운트할 확률이 높아진다. -> 해당 단어를 추출할 확률이 높아진다 라는 개념
- 이때 임의의 개수를 정하기 위한 기준을 위해 사용하는 것이 N-Gram이다.
- N은 n개의 연속적인 단어 나열을 의미한다.
- 가지고 있는 코퍼스에서 n개의 단어 뭉치 단위로 끊어서 이를 하나의 토큰으로 간주
- An adorable little boy is spreading smiles이 있을 때, 각 n에 대해서 n-gram을 전부 구해보면 다음과 같다.

unigrams : an, adorable, little, boy, is, spreading, smiles
bigrams : an adorable, adorable little, little boy, boy is, is spreading, spreading smiles
trigrams : an adorable little, adorable little boy, little boy is, boy is spreading, is spreading smiles
4-grams : an adorable little boy, adorable little boy is, little boy is spreading, boy is spreading smiles

- N-gram을 사용할 때 n이 1인 경우 유니그램, 2일 때는 바이그램, 3일 때는 트라이그램, n이 4이상인 경우 gram 앞에 그대로 숫자를 붙여 명명한다.



$$P(w|\text{boy is spreading}) = \frac{\text{count}(\text{boy is spreading } w)}{\text{count}(\text{boy is spreading})}$$

- N-gram을 통한 언어 모델에서 다음에 나올 단어의 예측은 오직 n-1개의 단어에만 의존한다.
- 즉, n이 4라고 한다면 spreading 다음에 올 단어를 예측하는 것은 n-1에 해당하는 앞의 3개의 단어만을 고려한다.

- 만약 가지고 있는 코퍼스에서 boy is spreading이 1000번 등장하고 boy is spreading insults가 500번 등장하며, boy is spreading smiles가 200번 등장한다면
    - Boy is spreading 다음에 insults가 등장할 확률은 50%이며, smiles가 등장할 확률은 20%이다.
    - 확률적으로 insults가 더 맞다고 판단하게 됨

keyBERT

- keyBERT는 텍스트 임베딩을 형성하는 단계에서 BERT를 사용함

Generate candidate words/phrases

- 공식 튜토리얼에 따르면 N-gram 모델을 사용해 문서의 각 문장을 단어로 쪼개준다.
    - CountVectorizer 등을 사용하여 문서를 토큰화 시킬 수 있음
- 토큰화를 시킨 후 BERT를 통해 각 단어에 대한 임베딩 벡터를 추출 함
    - BERT의 산출물인 임베딩 벡터를 사용

Calculating similarites between document embedding - word/phrase embeddings

- 문서와 가장 유사한 key words/phrases를 찾는 단계
- 문서와 가장 유사한 candidate(후보)는 문서를 나타내기 위한 적절한 키워드라고 간주하는 것
- Candidate keywords/phrases와 문서의 유사도를 계산하기 위해 벡터들 사이의 코사인 유사도를 사용한다.
- 벡터 간 거리를 이용한 단순한 접근 방식이지만 결과가 좋지 않을 때도 있다.
    - 이때 적절한 함수를 활용하면 결과를 높일 수 있음
    - MMR(Maximal Marginal Relevance)
        - MMR이란 텍스트 요약 작업에서 중복성을 최소화하고 결과의 다양성을 극대화한다.
        - 즉, 문서 주제와 관련성이 높은 문장을 선택하면서도 이전에 선택된 문장들에

대한 유사도가 낮은 문장을 선택하여 선택된 문장들 간의 정보 중복 문제를 해결할 수 있는 기법이다.

- 문서 내에서 반복해서 나오는 구문을 제거하는 과정이라고 생각하면 됨
- 즉, tf-idf에서의 idf와 유사한 과정이라고 생각하면 된다.

◆ MMR은 반복해서 나오는 문구의 순위를 멀리 매겨 문제를 해결함

■ MSS(Max Sum Similarity) OR MSD(Max Sum Distance)

◆ candidate-document 간 거리는 최소로 하면서 candidate-candidate 간 거리는 최대로 함으로써 의미적으로 풍부한 키워드 set을 얻는다

◆ 문서와 가장 유사한 words/phrases 2개를 사용해 이 두개의 top_n 단어에서 모든 top_n combination을 취해 코사인 유사도를 구한다.

◆ 여기서 유사도가 가장 낮은 조합을 추출한다.

◆ 즉, 서로 가장 덜 유사한 키워드를 추출하는 것

◆ MSD는 자료가 없음...