AiDA Lab.
Artificial Intelligence & Data Analysis Laboratory

AI 고급 : 자연어 처리 과정(정규)

11. 텍스트 유사도

Seokhwan Yang

텍스트 유사도 계산

- **텍스트 유사도란?**
  - **문장 간의 의미가 서로 얼마나 유사한지 측정하는 방법**
  - **어떻게 측정(인지)하는가?**
    - 사람
      - 무의식 중에 두 개의 문장에 동일한 단어나 의미상 비슷한 단어들이 얼마나 분포되어 있는지 직감적으로 파악
    - 컴퓨터
      - 사람과 동일한 방법으로 두 문장 사이의 유사도를 계산
      - 임베딩 기법을 통해 문장에 포함된 각 단어들의 벡터를 구한 다음 벡터 간의 거리를 계산함으로써 단어 간 유사도 계산
      - 문장은 단어의 묶음이므로 각 단어들의 벡터를 하나의 벡터로 묶어서 문장 간의 유사도 계산

n-gram 유사도

- n-gram 유사도
  - n-gram: 주어진 문장에서 n개의 연속적인 단어 시퀀스(단어 나열)를 의미함
  - 문장에서 n개의 단어를 토큰으로 사용
  - 이웃한 단어의 출현 횟수를 통계적으로 표현하여 텍스트의 유사도를 계산함
  - 구현 방법이 쉬우며 학습용 말뭉치의 품질만 좋으면 괜찮은 성능을 보여 줌
  - 단어의 출현 빈도에 기반한 유사도를 계산하므로 이를 통해 논문 인용 및 도용 정도의 조사에 활용
  - 두 문장 A, B에서

$$similarity = \frac{tf(A,B)}{tokens(A)}$$

tf: term frequency, 두 문장 A와 B에서 동일한 토큰의 출현 빈도
tokens: 해당 문장에서의 전체 토큰의 수
→ 전체 토큰 중에서 두 문장 A, B에 동일한 토큰이 얼마나 있는지 비율로 표현한 수식

코사인 유사도

- 코사인 유사도
  - 단어, 문장을 벡터로 표현하고 두 벡터 간 코사인 각도를 이용하여 유사도 측정
  - 일반적으로 벡터의 크기가 중요하지 않을 때 그 거리를 측정할 때 사용
  - 코사인 유사도는 벡터의 크기와 상관없이 안정적인 결과를 가짐
    - n-gram의 경우, 동일한 단어가 문서 내에 자주 등장하면 유사도 결과에 좋지 않은 영향
  - 코사인 유사도는 다양한 차원에서 적용 가능하여 실무에서 많이 활용 중
  - 두 문장 A, B에서

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

기타 유사도 계산 방법

- 이 외에도 다양한 벡터 유사도 계산 방법을 활용 가능
  - 맨해튼 거리(Manhattan Distance)
  - 유클리디안 거리(Euclidean Distance)
  - Infinity Norm
  - 자카드 유사도(Jaccard Similarity)
  - 기타 등등...

기타 유사도 계산 방법

- 맨해튼 거리(Manhattan Distance)
  - L1 norm을 사용한 거리 계산 방법. L1거리(L1 Distance)라고도 부름
  - 두 벡터의 각 차원 별 값의 차이의 절대값을 모두 합한 값
  - $d_{L1}(w, v) = \sum_{i=1}^{d} |w_i - v_i|, where\ w, v \in \mathbb{R}^d$

놈(norm): 벡터의 크기(magnitude) 또는 길이(length)를 측정하는 방법

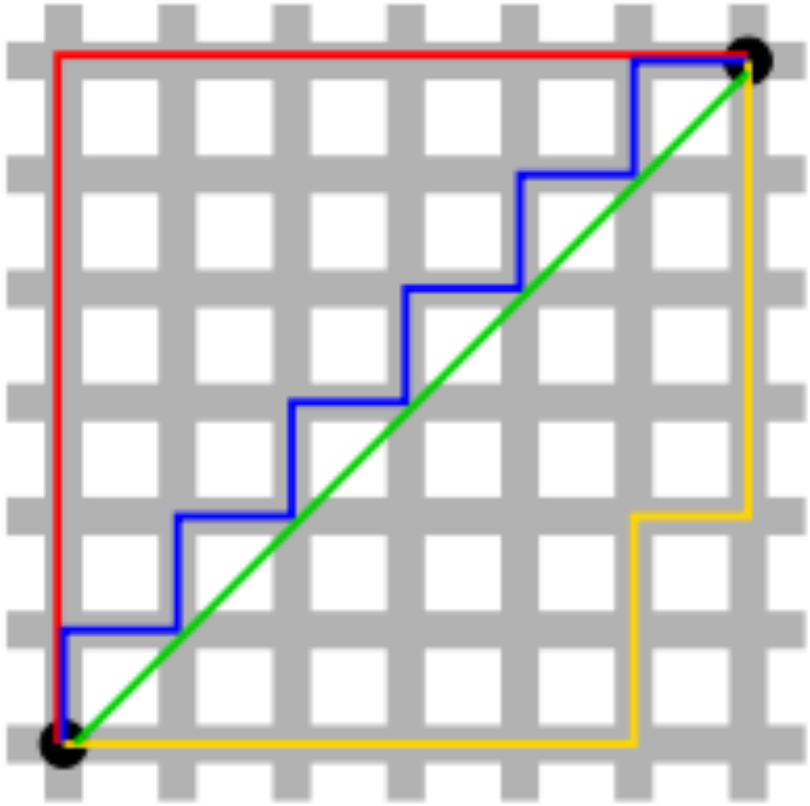
기타 유사도 계산 방법

- 유클리디안 거리(Euclidean Distance)
  - L2 norm을 이용한 거리 계산 방법. L2 거리(L2 Distance)라고도 부름
  - 차원별 값 차이의 제곱의 합에 루트를 취한 형태

- $d_{L2}(w, v) = \sqrt{\sum_{i=1}^{d}(w_i - v_i)^2}, where\ w, v \in \mathbb{R}^d$

기타 유사도 계산 방법

- **L1 거리와 L2 거리 비교**
  - **L1 거리: 초록색 선 외의 거리**
    - L2 거리를 제외한 선의 길이는 모두 같다
  - **L2 거리: 초록색 선**



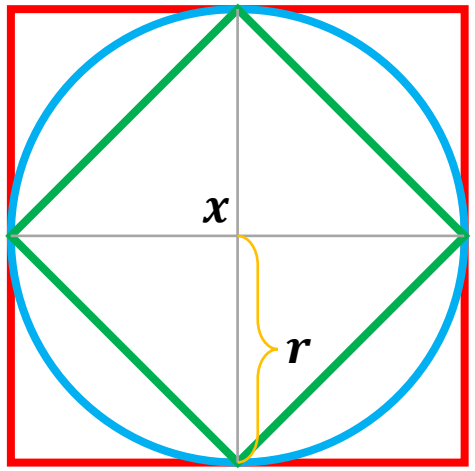
기타 유사도 계산 방법

- **Infinity Norm**
  - L_∞ 거리
  - 차원 별 값의 차이 중 가장 큰 값을 나타냄

- $d_{L_\infty}(w, v) = max(|w_1 - v_1|, |w_2 - v_2|, \ldots, |w_d - v_d|), where\ w, v \in \mathbb{R}^d$

기타 유사도 계산 방법

- 같은 크기(r)를 가진 L_1, L_2, L_∞ 거리를 그림으로 비교하면



GREEN : L1 Norm
BLUE : L2 Norm
RED : Infinity Norm

  - L_1 → L_2 → L_∞으로 갈 수록 벡터 내의 큰 값에 대해 더욱 집중해서 표현
  - 각 거리를 최소화 하도록 최적화를 수행한다면
  - L_2, L_∞로 갈수록 전체 벡터 중에서 큰 값이 작아지도록 최적화를 수행함

기타 유사도 계산 방법

- 자카드 유사도(Jaccard Similarity)
  - 두 집합 간의 유사도를 구하는 방법
  - 분자에는 두 집합의 교집합, 분모에는 두 집합의 합집합
  - 특징 벡터의 각 차원이 집합의 요소가 됨
  - 각 차원의 값이 0 또는 0이 아닌 값이 아니라, 수치 자체에 대해 계산할 때는 각 차원의 숫자에 대해 min, max 연산을 통해 계산함
  - $sim_{jaccard}(w, v) = \frac{|w \cap v|}{|w \cup v|} = \frac{|w \cap v|}{|w| + |v| - |w \cap v|} \approx \frac{\sum_{i=1}^{d} min(w_i, v_i)}{\sum_{i=1}^{d} max(w_i, v_i)}$, $where\ w, v \in \mathbb{R}^d$

기타 유사도 계산 방법

- 문서 간의 유사도 구하기
  - 단어 → 문장 → 문서, 즉 문서는 더 많은 단어의 집합
  - 문서에 대한 특징을 추출하여 문서 간의 유사도를 구함
  - 문서 내의 단어들에 대한 TF, TF-IDF를 구하여 벡터를 구성하고, 이를 활용하여 벡터 사이의 유사도를 계산