Chapter 2 - 자연어와 단어의 분산 표현

2.1 자연어 처리란

- 한국어와 영어 등 우리가 평소에 쓰는 말을 [자연어]라고 합니다.
- [자연어처리(NLP)]는 '우리의 말을 컴퓨터에게 이해시키기 위한 기술(분야)'입니다.

2.1.1 단어의 의미

- 우리의 말은 [문자]로 구성되며, 말의 의미는 [단어]로 구성됩니다.
- 단어는 말하자면 의미의 최소단위인 셈이죠.
- 컴퓨터에게 '단어의 의미'를 이해시키는 방법은 크게 3가지가 있습니다.
  1) 시소러스를 활용한 기법
  2) 통계 기반 기법
  3) 추론 기반 기법
- 가장 먼저, 사람의 손으로 만든 [시소러스]를 이용하는 방법이 있습니다.
- 그 다음으로, 통계 정보로부터 단어를 표현하는 [통계 기반 기법]이 있습니다.
- 마지막으로, 신경망을 활용한 [추론 기반 기법]이 있습니다.

2.2 시소러스

- 시소러스란 '뜻이 같은 단어(동의어)'를 의미합니다.
- 시소러스에서는 단어 사이의 '상위와 하위' 혹은 '전체와 부분' 등,
  더 세세한 관계까지 정의해둔 경우가 있습니다.

2.2.1 WordNet

- WordNet을 사용하면 유의어를 얻거나 '단어 네트워크'를 이용할 수 있습니다.

2.2.2 시소러스의 문제점

- 시대 변화에 대응하기 어렵다
  : 시소러스를 사람이 수작업으로 끊임없이 갱신해야 합니다.
- 사람을 쓰는 비용은 크다.
- 단어의 미묘한 차이를 표현할 수 없다.

2.3 통계 기반 기법

- 통계 기반 기법을 살펴보면서 우리는 말뭉치를 이용할겁니다.
- 말뭉치란 간단히 말하면 대량의 텍스트 데이터입니다.
- 결국 말뭉치란 텍스트 데이터에 지나지 않습니다만, 그 안에 담긴 문장들은 사람이 쓴겁니다.
- 자연어 처리에 사용되는 말뭉치에는 텍스트 데이터에 대한 추가 정보가 포함되는 경우가 있습니다.
  예컨대 텍스트 데이터의 단어 각각에 '품사'가 레이블링될 수 있습니다.

2.3.1 파이썬으로 말뭉치 전처리하기

- [전처리]란 텍스트 데이터를 단어로 분할하고 그 분할된 단어들을 단어 ID 목록으로 변환하는 일입니다.
- 파이썬 내장 함수인 딕셔너리를 사용하면 단어를 가지고 단어 ID를 검색하거나, 반대로 단어 ID를 가지고 단어를 검색할 수 있습니다.

2.3.2 단어의 분산 표현

- '단어의 의미'를 정확하게 파악할 수 있는 벡터 표현입니다.
- 이를 [단어의 분산 표현]이라고 말합니다.

2.3.3 분포 가설

- 단어의 의미는 주변 단어에 의해 형성된다는 것이 전제
- 이를 [분포 가설]이라고 함. 즉, 단어 자체에는 의미가 없고, 그 단어가 사용된 '맥락'이 의미를 형성한다는 것.
- [맥락]이란 특정 단어를 중심에 둔 그 주변 단어. 맥락의 크기(주변 단어를 몇 개나 포함할지)를 '윈도우 크기'라고 합니다.

2.3.4 동시발생 행렬

- [통계 기반 기법] : 그 주변에 어떤 단어가 몇 번이나 등장하는지 세는 것
- 모든 단어 각각의 맥락에 해당되는 단어의 빈도를 세어 표를 정리한 벡터의 집합, 행렬은 [동시 발생 행렬]이라고 한다.

2.3.5 벡터 간 유사도

- 단어 벡터의 유사도를 나타낼 때는 [코사인 유사도]를 자주 이용합니다.
- [코사인 유사도]에는 L2 노름이 나오는데, 어쨌든 [코사인 유사도]의 핵심은 벡터를 정규화하고 내적을 구하는 것입니다.
- 그런데 [코사인 유사도]는 인수로 제로 벡터가 들어오면 '0으로 나누기' 오류가 발생해버립니다.
- 이 문제를 해결하는 전통적인 방법은 나눌 때 분모에 작은 값을 더해주는 것입니다.
- [코사인 유사도] 값을 -1에서 1 사이의 값이기에, 0.7 정도는 비교적 높다(유사성이 크다)고 말할 수 있습니다.

2.3.6 유사 단어의 랭킹 표시

- argsort() 메서드는 넘파이 배열의 원소를 오름차순으로 정렬. (단, 반환값은 배열의 [인덱스])

2.4 통계 기반 기법 개선하기

2.4.1 상호정보량

- 하지만 단순히 등장 횟수만을 본다면 'cat'는 'drive'보다는 'the'와의 관련성이 훨씬 강하다고 나올 겁니다. 'the'가 고빈도 단어라서 'car'와 강한 관련성을 갖는다고 평가되기 때문이죠.
- 이 문제를 해결하기 위해 [점별 상호정보량(PMI)]이라는 척도를 사용한답니다.
- PMI 값이 높을수록 관련성이 높다는 것을 의미합니다.
- PMI를 이용하면 'car'는 'the'보다 'drive'와의 관련성이 강해집니다.
- 이 PMI에도 한 가지 문제가 있습니다. 바로 두 단어의 동시발생 횟수가 0이면 -inf로 계산된다는 것입니다. 실제로 구현할 때는 양의 정보량(PPMI)를 사용합니다.
- 그러나 PPMI 행렬에도 여전히 큰 문제가 있습니다. 말뭉치의 어휘 수가 증가함에 따라 각 단어 벡터의 차원 수도 증가한다는 문제죠. 예를 들어, 말뭉치의 어휘 수가 10만 개라면 그 벡터의 차원 수도 똑같이 10만이 됩니다.

2.4.2 차원 감소

- [차원 감소]는 문자 그대로 벡터의 차원을 줄이는 방법을 말합니다.
- '중요한 정보'는 최대한 유지하면서 줄이는 게 핵심입니다.
- [그림 2-8]처럼 데이터의 분포를 고려해 '축'을 찾는 일을 수행합니다.
- 원소 대부분이 0인 행렬 또는 벡터를 [희소행렬] 또는 [최소벡터]라 합니다.
- 차원 감소의 결과로 원래의 희소벡터는 원소 대부분이 0이 아닌 값으로 구성된 [밀집 벡터]로 변환됩니다.
- 차원을 감소시키는 방법은 여러 가지입니다만, 우리는 [특잇값 분해(SVD)]를 이용하겠습니다.
- SVD를 수식으로 표현하면, $X = U * S * V^t$로 표현됩니다.
- 이 U 행렬을 [단어 공간]으로 취급할 수 있죠. 또한 S는 대각 행렬로, 그 대각성분에는 '특잇값'이 큰 순서로 나열되어 있습니다. 특잇값이란, 쉽게 말해 '해당 축'의 중요도로 간주할 수 있습니다. 그래섯 [그림 2-10]과 같이 중요도가 낮은 원소를 깎아내는 방법이 차원 축소라고 보면 됩니다.
- 행렬 S에서 특잇값이 작을수록 중요도가 낮다는 것을 의미합니다.
- 이를 우리 문제로 가져와서 '단어의 PPMI 행렬'에 적용해볼까요? 그러면 행렬 X의 각 행에는 해당 단어 ID의 단어 벡터가 저장되어 있으며, 그 단어 벡터가 행렬 U라는 차원이 감소된 벡터로 표현되는 겁니다.
- SVD는 넘파이의 linalg 모듈이 제공하는 svd 메서드로 실행할 수 있습니다.

2.4.4 PTB 데이터셋

- 이번 절에서는 '본격적인' 말뭉치를 이용해보겠습니다. 그 주인공은 바로 펜 트리뱅크(PTB)입니다.
- 이 책에서는 각 문장을 연결한 '하나의 큰 시계열 데이터'로 취급합니다.

2.4.5 PTB 데이터셋 평가

- 이번에는 큰 행렬에 SVD를 적용해야 하므로 고속 SVD를 이용할 것을 추천합니다. 고속 SVD를 이용하려면 sklearn 모듈을 설치해야 합니다.
- 이처럼 단어의 의미 혹은 문법적인 관점에서 비슷한 단어들이 가까운 벡터로 나타났습니다.
- 말뭉치를 사용해 맥락에 속한 단어의 등장 횟수를 센 후 PPMI 행렬로 변환하고, 다시 SVD를 이용

해 차원을 감소시킴으로써 더 좋은 단어 벡터를 얻어냈습니다.