**Clustering and WordEmbedding**

**2015058204 오현택**

**Language**

- Python 3.61

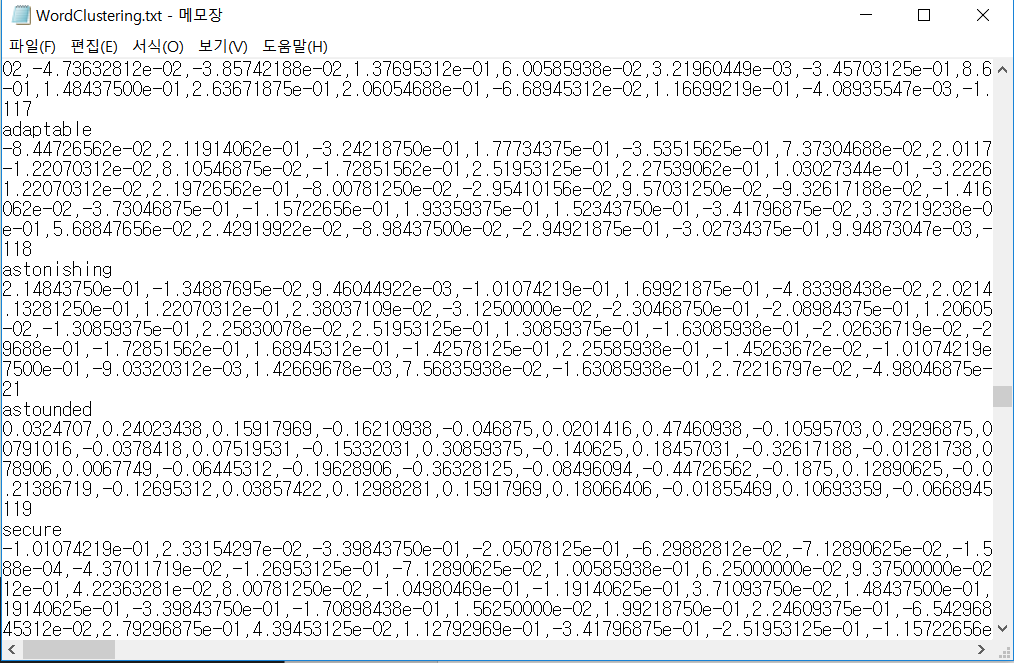
**Specification**

- GoogleNews corpus로 학습된 300차원의 Word2Vec Embedding vector가 실제로 단어의 의미에 맞게 분포되었는지 확인하기. Cosine similarity와 Euclidean similiarity로 Complete Link Clustering을 진행 후, 각각의 cluster에 대하여 Entropy 계산을 통하여 Information Gain값을 구해 Clustering 결과에 대해 분석해본다.

**Implementation**

- 우선, 분석이 제대로 이루어지지 않았음을 먼저 밝히겠다. 핵심은 Entropy 분석을 통하여 clustering 적합성 등을 논하는 데에 의의가 있지만, 당장 과제의 난이도에 막혀 clustering을 쉽게 구현하는데에만 중점을 두었던 것 같다. Clustering을 행렬을 update하면서 구현할지, 아니면 트리나 B트리 형태로 만들지도 생각해 보았지만 아무래도 이 경우들에는 구현의 어려움과 제약조건 등이 많았다. 그래서 최근에 프로그래밍 대회 등에서 Kruskal 알고리즘 등을 사용할 때 Union-find라는 자료구조를 자주 사용하였는데 , 이 자료구조로 set 개념을 쉽게 구현할 수 있어서 이 알고리즘 방식으로 clustering을 구현하면 쉽겠다고 생각했다.

실제로 막막했던 기간에 비하면 clustering을 진행하는 코드 작성 과정은 어렵지 않고막힘없이 순탄했다. 코드와 알고리즘을 잠깐 설명하자면, Word2Vec 벡터들 338개의 모든 쌍(Binomial(338,2))에 대해 similarity를 계산 후, 그 값에 따라 높은 값부터 낮은 값까지 내림차순 정렬을 한다. 그리고 입력 받은 threshold similarity 까지 complete link clustering을 시행하는데, complete link clustering을 해주려면 두 cluster 집합 A,B의 distance중 젤 먼 원소끼리의 distance, 즉 min값을 사용하므로 두 집합의 모든 원소쌍에 대한 distance정보가 visitedList에 들어있어야 한다. 그리고 이 조건을 만족할 때 clustering을 해주는 과정을 위에서 말한 union-find 개념을 이용한다. 배열(리스트)를 사용하여 한 집합인지 아닌지를 나타내는 방법인데, 동일한 cluster이면 cluArr 클래스 배열에 들어있는 자기 클래스의 cluNum(=cluster Number)를 동일하게 바꿔준다. 처음에 굳이 1001부터 할당하는 이유는 clustering 이후 다시 번호를 1부터 깔끔하게 지정해주기 위함이다.



위의 그림은 cosine similarity 임계점을 0.2까지로 설정한 후 clustering을 진행한 출력파일의 모습이다. 이렇듯 여차저차 clustering 하는데 까진 문제가 없었지만, WordTopic.txt에서 Group화된 cluster에 맞춰 엔트로피 개별 분석을 하는데에서 어려움이 있었다.

Pointer가 있는 자료구조나 linkedList 들은 같은 집합에 있는 원소들을 다루는데 편하지만, 내가 쓴 방식은 너무나 어려웠다. 한 개의 group을 리스트 하나로 set을 만들어 처리하기는 쉬웠지만, 두 개의 group을 대상으로 서로 탐색하고 값을 가져오고 하는 방식은 너무나도 어렵고 불리할 뿐더러 시간복잡도도 O(n^4)까지 올라가게 된다. 비록 제대로 된 분석은 막혀서 실시하지 못했지만 처음부터 큰 디자인이 중요하다는 걸 다시금 알게된 과제였다.