FBS(Friend-Base-Clustering) Algorithms for Graph Data

2018253034 이정연

2018253070 배재익

2018251039 고명석

Document Sections

Ⅰ. Background 1

Ⅱ. Algorithm Approach  2

Ⅲ. Implementing an Algorithms 4

Ⅳ. Result 5

Ⅴ. Conclusion 6

Background

그래프(Graph)는 정점(vertex, 혹은 node)과 이를 연결하는 간선(edge)들로 이루어져 있다. 간선은 두 정점이 연관성을 가진다는 것을 의미하므로, 특정 정점의 차수(degree)가 클수록 해당 그래프 내에서 더 많은 정점과의 연관 관계를 가지고 있다고 볼 수 있다. 이를 바탕으로 차수가 가장 큰 정점은 자신이 속한 그래프에서 중심성이 제일 높다고 해석하는 지표가 연결 중심성(degree centrality)이다. 본지에선 여기에서 더 나아가, 한 정점 자신의 차수 뿐 아니라, 그 정점과 인접한 정점의 차수 또한 그래프 내의 중심성을 매기는 데 중요한 척도가 된다고 해석했다. 이러한 생각을 바탕으로, 연결 중심성 계산 방식을 수정하여 중심이 되는 정점과 그와 가까운 정점들을 클러스터링하는 FBC(Friend Base Clustering) 알고리즘을 개발했다.

이번 프로젝트의 목적은 제공된 Ground-truth 데이터와 비교하여 클러스터링 결과를 평가하기 위해 f-score를 사용한다. 이때 각 출력 클러스터와 각 Ground-truth 클러스터 간의 최대 f-score를 기준으로 한다. 이번 프로젝트에서 만든 새로운 알고리즘과 기존의 과제(assignment 5, assignment 6)에서 구현한 두 개의 이전 알고리즘 간의 평균 f-score를 비교하고 직접 구현한 알고리즘이 성능이 좋은 알고리즘인지를 확인해본다.

Algorithm Approach

이 알고리즘에서, 각 정점에 대해 계산되는 depth 개념을 고려한다. Depth는 naïve degree centrality 대신 사용되는 변형된 연결 중심성으로, 인접한 정점의 중심성을 함께 고려하기 위한 지표이다. 알고리즘은 두 단계로 구성된다. 첫 단계는 depth를 계산하는 단계이고, 두 번째 단계는 각 정점의 depth에 따라 클러스터를 나누는 단계이다.

첫 번째 단계에서, 각 정점의 depth는 처음에는 1이며, 매 반복 회차마다 증가분이 계산되어 더해진다. 이 증가분의 계산 방법은 [Figure 1]를 참고하라. 알고리즘의 반복 횟수가 증가함에 따라, 각 정점의 depth 또한 단조 증가하게 된다.

depth의 증분은 인접 정점의 depth의 합을 반복 횟수의 제곱으로 나누어 반복 횟수가 무한히 증가하는 경우 depth의 증분은 0으로 수렴한다. 따라서, 반복이 진행됨에 따라 depth의 증분은 단조 감소하고, 이 알고리즘은 언젠간 종료함이 보장된다. 모든 정점의 depth 증분이 알고리즘에서 정의한 임계 값()이하라면 반복을 종료한다.

두 번째 단계에서는 정점의 depth가 알고리즘 동작 인수(parameter)로부터 계산된 cutoff-value를 넘지 못하는 정점들과 그렇지 못한 정점들을 따로 분류하여 클러스터링한다. 이 cutoff-vlaue는 동작 인수로 지정된 quantile을 토대로 계산되며, 모든 정점의 depth 사이의 비율(0~1 사이의 값)이다. 예컨대, 동작 인수로 지정된 quantile 이 0.25인 경우 cutoff-value는 depth를 하위 25%와 상위 75%로 분리하는 값이며, 하위 25%에 해당하는 depth를 가진 정점들과 상위 75%에 해당하는 정점들 사이의 간선이 삭제된다. 이후 남겨진 간선들의 연결성을 토대로 클러스터를 만든다.

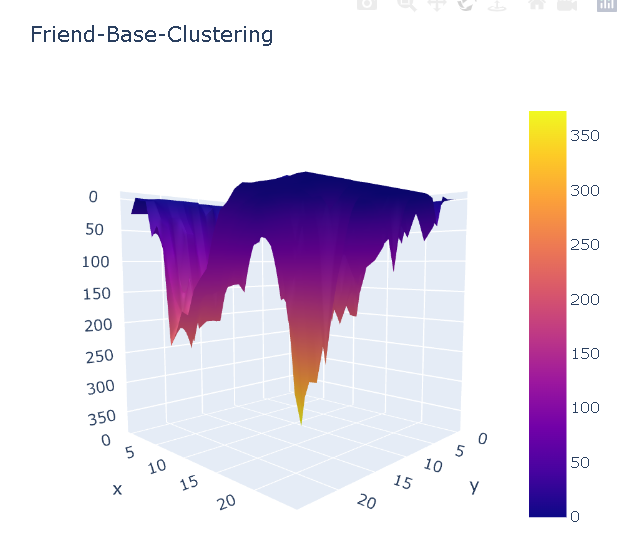
**Figure 1.**

depth 증분 계산 공식



**Figure 2.**

depth와 Percentile을 이용한 분할 예시 (임의 데이터, Parameter = 0.25)



1st Quartile(25%)에 해당하는 값

Implementing an Algorithms

**FBC algorithm - Pseudo Code**

1: **depth**: value of each node that calculated in algorithm (Start at 1)

2: **iteration**: number of repeat-util phrase executed

3: **repeat**

4: **for** each node **do**

5: increment := (sum of adjacent node’s depth) / (square of iterations);

6: depth := depth + increment;

7: **end for**

8: **until** (maximal increased depth of all nodes < 0.1)

9: value\_for\_slice := Percentile of depth (based on parameter);

10: **for** each edge in graph **do**

11: **if** value\_for\_slice is between two node’s depth

12: remove edge;

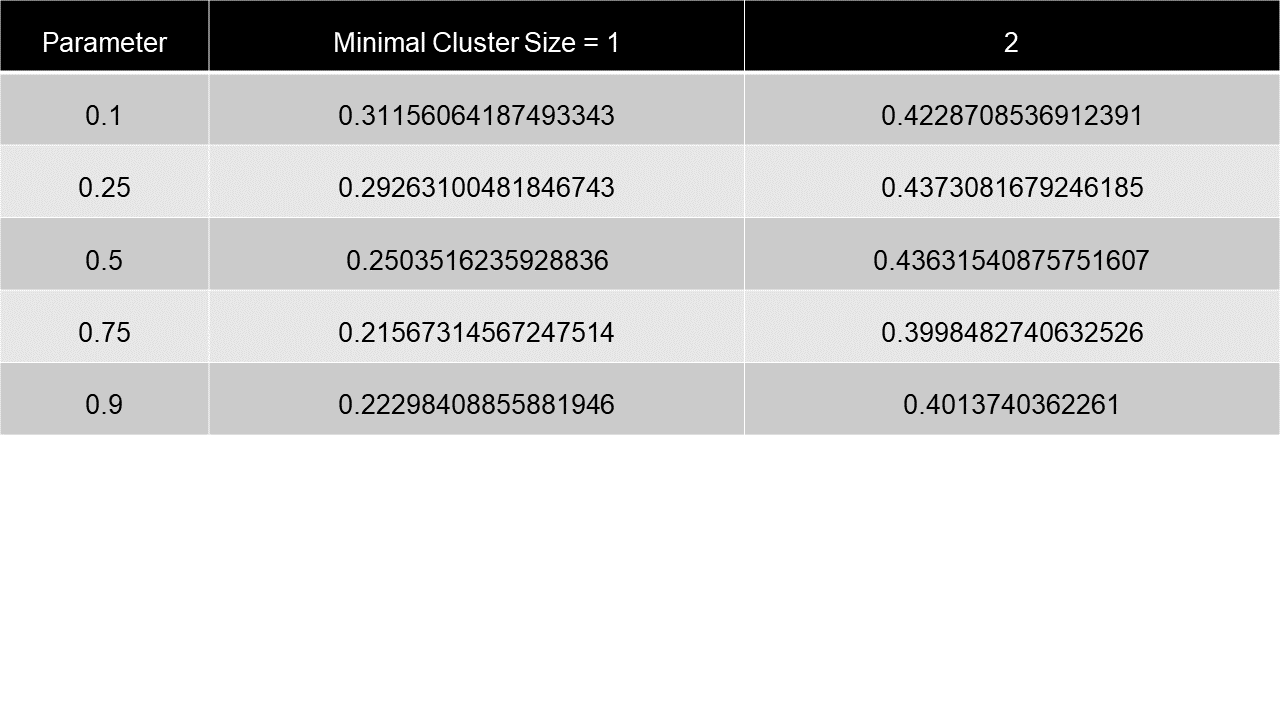
13: **end if**

14: **end for**

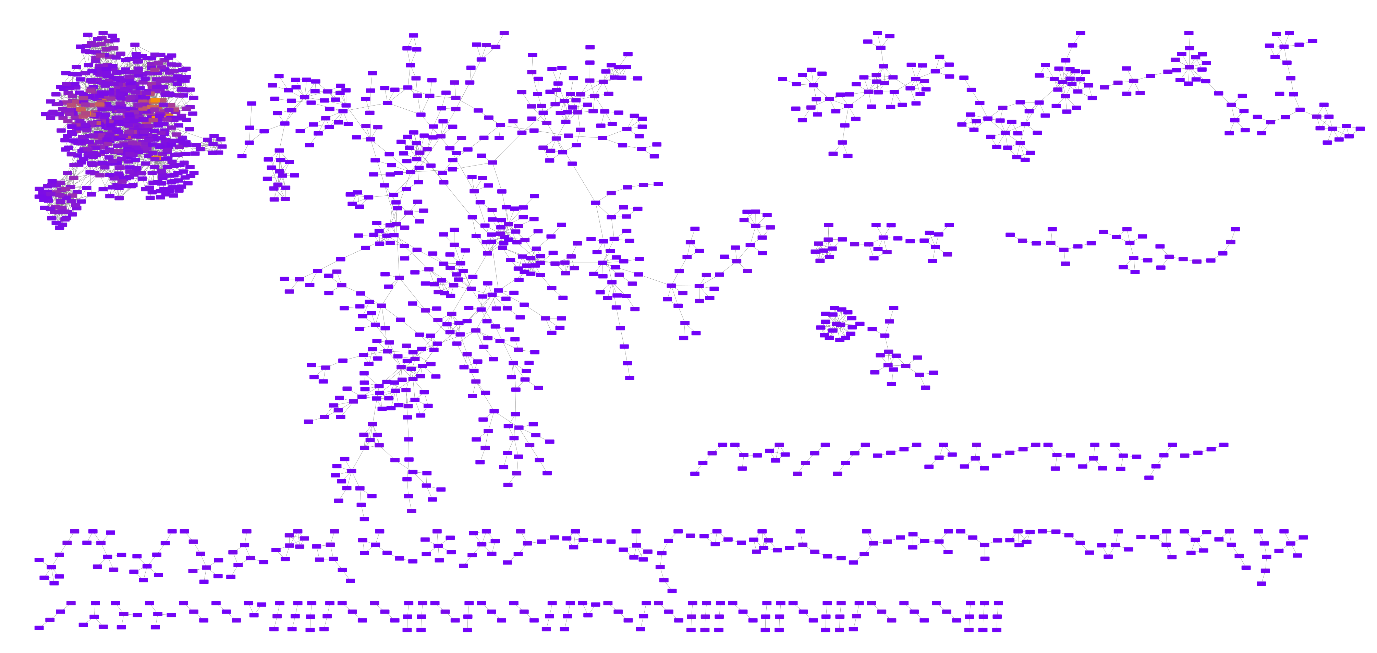
Result

클러스터링 결과를 Ground-truth 데이터와 비교하여 f-score로 확인해본 결과, Assignment5의 f-score는 0.36정도이고 Assignment6은 0.24정도로 결과값이 나오는 걸 확인해 볼 수 있다

**Parameter와 Minimal Cluster Size에 따른 FBC의 f-score 결과값**



**Output**



\* Cytoscape를 활용하여 Clustering 결과를 출력한 모습. (파랑에서 주황으로 갈수록 큰 depth값을 가짐.)

Conclusion

1. Strength

1. 반복 프로세스가 단순하기 때문에 빠르게 연산이 가능하다.

2. naïve degree centrality를 이용한 cutoff 방식에 비해 더 상식에 부합하는 결과를 기대할 수 있다.

3. 간선이 하나인 정점도 간선이 많은 정점 옆에 있을 경우 큰 클러스터에 포함이 된다.

1. Weakness
2. 모든 정점이 비슷한 차수를 가질 때 부정확하게 클러스터링이 될 수 있다.
3. 충분한 density가 보장되지 않으면 부적절하게 클러스터링이 된다.
4. 주어진 입력 그래프에 적응적이지 않다. 다시 말해 동작 인수를 데이터에 맞도록 조정하는 작업이 필요하다.
5. 역할분담
6. 고명석 : 아이디어 구상, 코드 구현, 보고서 전체적 작성 및 종합
7. 배재익 : 아이디어 구상, 코드 구현, 보고서 pseudo code 작성
8. 이정연 : 아이디어 구상, 코드 구현, 결과 그래프 작성