그래프 클러스터링 개발

김송우*·이지윤**·정유라**
*연세대학교 미래캠퍼스 컴퓨터정보통신공학부
**연세대학교 미래캠퍼스 디지털헬스케어학부

Developing Graph Clustering

Song-WOO KIM*, JI-YUN LEE** and Yu-Ra JEONG**
*Computer Engineering, Yonsei University Mirae Campus
**Digital Healthcare, Yonsei University Mirae Campus

KEY WORDS: 알고리즘; 클러스터링; 그래프의 연결성; 유의미한 클러스터링;

ABSTRACT: 본 논문은 그래프 클러스터링시, Maximal Clique를 기반으로한 새로운 클러스터링 방법론을 제시하고, F-measure 방법으로 Ground Truth와 비교하여 분석한다. 먼저, 주어진 유전자 데이터 셋을 이용하여 Maximal Clique를 구한다. 이후, 중복으로 집계된 clique를 삭제한 후, 그 결과를 초기 클러스터로 정의한다. 이때, 한 개의 점만으로 된 그룹은 클러스터링 되지 않았다고 정의한다. 다음으로, 삭제하는 과정에서 클러스터링 되지 않아 남겨진 한 개의 노드들에 대해 각각 클러스터에 포함되도록 클러스터링 방법론을 제시한다. 제안된 방법을 기반으로 3가지의 클러스터링 방법에 대한 결과를 도출하고, Ground Truth와 비교하여 제안한 클러스터링 방법에 대해 분석한다.

1.서론

그래프란 노드(정점) 세트와 노드 쌍 사이를 연결하는 엣지 세트로 구성된 구조이다.[1] 클러스터링이란 균일하지 않은 데이터를 그룹화하고 그룹들을 식별하며 데이터 분류를 하는 것이다. 이 과정의 결과값으로 나온 그룹을 클러스터라고 한다. 그래프 클러스터링 기법[2]은 말 그대로 데이터를 그래프 형태로 클러스터링 하는 것을 의미한다. 이 기법은 그래프의 엣지 구조를 고려하여 각 클러스터 내에는 많은 엣지가 있고 클러스터 사이에는 상대적으로 적은 엣지가 있도록 그래프의 노드을 클러스터로 그룹화하는 작업이다. 그래프 데이터에는 두 가지 형태의 클러스터링 기법이 있다. 첫 번째, 점 클러스터링은 그래프의 점을 엣지 가중치 또는 엣지 거리를 기반으로 조밀하게 연결된 영역 그룹으로 클러스터링 하는 것이다. 두 번째, 엣지 클러스터링 그래프를 클러스터링 할 개체로 취급하고 유사도를 기반으로 이러한 개체를 클러스터링한다.

그래프 클러스터링 엣지 기법 중 Density-Based Method[3]가 있다. Density-Based Method는 아래 식을 기반으로 그래프의 밀도의 정의를 확인할 수 있다. (V= 그래프 내의 꼭짓점의 개수, E = 엣지의 개수)

$$D(G) = (2|E|) / (|V|(|V|-1))$$

그리고 D(G)=1인 그래프의 경우, complete graph이며 Clique이다. 해당 개념을 이용한 클러스터링 기법은 Maximal Clique Algorithm, Clique Percolation Algorithm 있다.

Jaccard[4]란 두 세트 간의 유사도를 확인하는 기준이며 아래 식을 통해 두 그래프의 간의 여러 neighbor를 비교함으로써 그래프 간의 유사성을 확인할 수 있다. (N(x) = 점 x의 neighbor들의 집합)

$$S(x,y)= (|N(x)\cap N(y)|)/(|N(x)\cup N(y)|)$$

---

[1] Satu Elisa Schaeffer, Graph clustering, Computer Science Review, Volume 1, Issue 1, (2007)

[2] Aggarwal, C.C. Graph Clustering. In: Sammut, C., Webb, G.I. (eds) Encyclopedia of Machine Learning. Springer, Boston, MA. (2011)

[3] Hao Li, Xiaojie Liu, Tao Li, Rundong Gan, A novel density-based clustering algorithm using nearest neighbor graph, Pattern Recognition, Volume 102, (2020)

[4] Niwattanakul, Suphakit, et al. "Using of Jaccard coefficient for keywords similarity." Proceedings of the international multiconference of engineers and computer scientists. Vol. 1. No. 6. (2013)

그리고 density=1인 경우, 각 세트의 크기가 클수록 jaccard 값이 커진다.

Tsutomu[5] et al.는 생물 의학 관계 그래프에서 촘촘하게 연결된 그래프 즉, density가 1인 경우인 clique가 유전자 추출분야에서 중요하다고 주장하였다. Palla et al.은 최대 clique 개수를 구하는 Maximal Clique Algorithm [6] 을 제안하였다. 또한 Qinghua Wu et al.[7]은 Maximal Clique Problem은 수많은 분야에서 응용되고 있음을 밝혔다. 본 논문에서는 Clique의 연결성의 중요성을 인식하여 중복 제거한 Maximal Clique를 초기 클러스터로 정의하였다. 제거하는 과정에서 클러스터링 되지 않고 한 개의 점들이 남겨지게 된다. 남겨진 점들에 대해서는 다른 클러스터에 jaccard 유사도 측정 방식을 기반으로 클러스터링 되도록 알고리즘을 정의하였다. 이후, 각각의 알고리즘에 대해서 F1-score 방법으로 Ground Truth와 비교하였고, 도출된 결과에 대해서 분석하였다.

2. 클러스터링 알고리즘

2.2 The M1-Graph algorithm

본 논문에서는 M1-Graph algorithm에 관련된 정의와 접근법을 소개한다. 해당 알고리즘은 연결성이 데이터로 주어진 경우 주로 사용된다. M1-Graph의 구현 방식은 초기 클러스터를 구성하고, 클러스터에 포함되지 않은 데이터를 클러스터링하는 방법에 대해 제안하는 두 가지 순서로 이루어져 있다.

첫 번째 순서는 Maximal Clique를 바탕으로 한 알고리즘이다. 여러 점이 있고 이러한 점들 간의 연결성에 대한 데이터가 있다고 가정하자. 먼저, 주어진 유전자 데이터에 대해 Maximal Clique를 구한다. 크기가 큰 clique 부터, 이 clique에 포함된 유전자 데이터를 중복해서 포함하고 있는 clique를 제거하였다. 그렇게 삭제한 결과를 초기 클러스터로 정의한다. 여기서 Maximal Clique와의 차이점은 모든 클러스터에 **Fig 1**.과 같이 중복값이 없다는 것이다. 예를 들어, A, B, C 라는 점이 있다. 이 점들은 {A, B}, {A, C}, {A, D}, {B, C} 로 연결된 점들이다. 그렇다면, size-2 clique는 {A, B}, {A, C}, {A, D}, {B, C}이다. size-3은 {A, B, C}로 클러스터가 된다. 이때, 점 D는 클러스터가 되지 않았다고 정의한다. 첫 번째 단계에서는 {A, B, C}라는 클러스터로 결정이 되고 클러스터가 되지 않은 점 D가 남는다. 이렇게 클러스터링이 완료가 되면 지금까지 클러스터된 모든 클러스터는 density가 1이다. 이때, 중복을 완전히 제거시키면, 클러스터의 크기가 1개만 있는 것이 많아진다. 우리는 이를 클러스터가 되지 않았다고 정의한다.

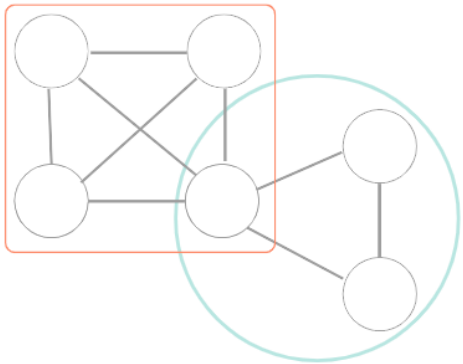

Fig 1. Maximal Clique의 중복에 대한 예시

두 번째 순서는 트리와 유사한 형태로 진행된다. 클러스터가 되지 않은 것들을 트리의 형태로 진행하며 병합한다. 먼저 클러스터가 되지 않은 것을 루트 노드로 지정한다. 루트 노드의 여러 neighbor 중 가장 큰 edge jaccard 값을 지정하여 루트 노드와 연결한다. 그 다음 다시 연결된 점 기준으로 여러 neighbor 중 가장 큰 edge jaccard 값을 찾는다. 이 때, 선택된 점이 2개 이상의 클러스터에 들어가 있는 점일 때 그 클러스터의 모든 점들을 병합한다. 위의 과정에서 이전의 edge jaccard 값과 같거나 낮아질때, 클러스터의 병합을 멈추거나 모든 neighbor가 클러스터에 포함이 되어 있다면 멈춘다.

이러한 트리 형태의 병합을 시도한 이유는 jaccard 값이 동일할 때 어떤 점을 선택해야 할지에 대한 문제점을 해결해주기 때문이다. 만약, 비교할 여러 점의 jaccard 값이 동일하다면 여러 점를 모두 후보로 두어 위와 같은 로직에 따라 각 노드들이 멈출 때까지 진행한다. 각 후보들이 지나친 노드들의 jaccard 평균이 높은 것을 선택한다.

---

[5] Matsunaga, T., Yonemori, C., Tomita, E. *et al.* Clique-based data mining for related genes in a biomedical database. *BMC Bioinformatics* 10, 205 (2009)

[6] Palla, G., et al., "Uncovering the overlapping community structure of complex networks in nature and society", Nature (2005)

[7] Qinghua Wu, Jin-Kao Hao,A review on algorithms for maximum clique problems, European Journal of Operational Research, Volume 242, Issue 3, Pages 693-709 (2015)

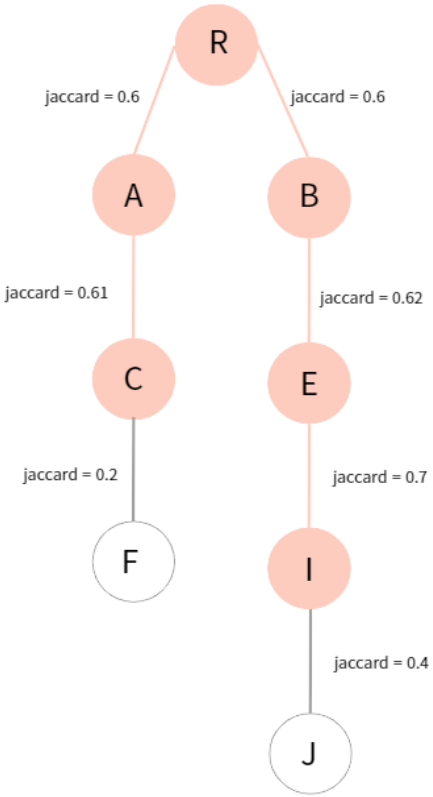

Fig 2. M-Graph의 jaccard를 이용한 Tree구조에 대한 예시

Jaccard를 이용한 트리 구조에 대한 방법에 대해 **Fig 2**이라는 예시를 들어 설명하자면, 점 R은 M1-graph 알고리즘 정의에 따라 클러스터 되지 않은 노드이다. 점 R의 여러 neighbor 중 jaccard 값이 가장 큰 것을 고른다. 하지만, 점 R의 jaccard 최고 값이 0.6으로 동일하다. 이러한 경우에는, A, B 모두 R 노드와 연결한다. 점 A를 기준으로 먼저 시작하자면, 점 A에서의 여러 neighbor 중 가장 큰 jaccard 값이 0.61인 점 C가 선택되어 A와 C를 연결한다. 이때, 만약 점 C가 clique를 통해 묶여 클러스터가 2개 이상이라면, 점 C가 들어가있는 클러스터를 모두 같이 병합한다. 점 C의 여러 neighbor 중에서 가장 큰 jaccard 값이 0.2라면, A-C의 jaccard값인 0.61보다 작기 때문에 {R, A, C}로 클러스터가 된다. 위와 같은 방법으로 진행하면 {R, B, E, I} 로 클러스터가 진행된다. 이 두 개 중에 지금까지 진행했던 jaccard의 평균이 높은 것으로 선택된다. **Fig 2.**를 보면 {R,A,C}의 클러스터에서는 R-A의 jaccard는 0.6이고 A-C의 jaccard는 0.61이다. 이에 따라, {R, A, C}의 클러스터의 jaccard값의 평균은 (0.6 + 0.61) / 2 = 0.605이고 {R,B,E,I}의 클러스터의 jaccard 평균도 앞의 과정처럼 하면, (0.6 + 0.62 + 0.7) / 3 = 0.64이므로 {R,B,E,I}가 결국 하나의 클러스터가 된다.

정리하자면, M1-graph의 수도코드는 아래와 같다:

1) Maximal Clique를 기반으로, 각각의 점이 클러스터에 중복해서 포함하지 않도록 제거한 clique를 초기 클러스터로 설정한다. 이때, 클러스터 되지 않은 점들을 루트 노드로 설정한다.
2) - 각각의 루트 노드에서 여러 neighbor 중 가장 큰 edge jaccard 값을 가지는 neighbor를 선택한 후, 이를 자식 노드로 생성한다.
   - jaccard 값이 동일한 경우, 모든 경우를 나열하고, jaccard의 평균이 가장 높은 노드들의 집합을 얻는다.
   - 모든 neighbor가 클러스터에 포함이 되어있다면, 멈춘다.
3) 다음 자식 노드에서 해당 노드의 neighbor와의 jaccard 값이 부모 노드와의 jaccard 값보다 크다면 3) 과정을 반복한다. 같거나 작아진다면 그 때의 자식 노드를 리프 노드로 설정한다.
4) 노드들의 집합을 하나의 클러스터로 병합한다. 이때, 2개 이상의 클러스터에 포함되어 있다면, 그 클러스터를 포함하여 하나의 클러스터로 병합한다.

2.3 The M1-graph의 한계점

M1-graph는 클러스터링 되지 않은 한 점으로부터 트리 구조로 진행될 때, 고려할 jaccard 값으로 클러스터 내부의 점을 포함하지 않았다. **Fig 3.**로 예를 들어보면, 점 R은 클러스터링이 되지 않은 점이고 점 A는 클러스터링이 되어 clique 안에 들어가있는 점 중 하나라고 하자. R의 여러 neighbor 중에서 가장 큰 jaccard 값을 가진 점 A이다. A가 포함된 클러스터 안에 있는 점들은 jaccard를 계산할 때, A의 neighbor에서 배제가 된다. 즉, jaccard 값 계산은 노드 E, I는 배제가 된다. 이러한 조건 때문에, 트리는 같은 클러스터에 있는 노드를 선택할 수 없다. 즉, 트리를 구성하는 과정에서 클러스터가 다른 자식 노드들로 구성할 확률이 커지게 된다. 각각 다른 클러스터가 하나의 클러스터로 계속해서 병합되게 되어, 결국에는 한 개의 클러스터가 계속해서 비대해지는 결과를 초래하게 된다. 실제로 유전자 쌍 데이터를 입력하였을 때 가장 크기가 큰 클러스터의 크기가 1479개가 나왔음을 확인하였다.

2.4 The M2-graph 알고리즘

위와 같은 M1-graph의 알고리즘의 한계점을 개선하고자 M2-graph를 고안하였다. M1-graph에서는 클러스터 안에 들어가 있는 점들은 jaccard 값을 구할 때 배제하였다. 본 논문에서는 하나의 클러스터에 노드가 몰리는 현상의 원인이 클러스터 내부의 점은 jaccard 값을 계산할 때, 고려하지 않았기 때문이라고 판단했다. 이러한 것들을 개선하고자, M2-graph에서는 jaccard 값을 구할 때, 클러스터 내부의 점도 포함이 된다. M2-graph는 jaccard 값을 통해 다시 동일 클러스터로 돌아왔다면, 계속 진행하는 것이 무의미 판단하였다. 이에 따라, 이미 트리 구조에 있는 점이 어떠한 클러스터에 들어가 있는데도 불구하고, 클러스터에 들어가 있는 다른 한 점을 또 만난다면 종료한다는 조건을 추가한다. Fig 3. 을 보면, R이라는 한 점에서 시작한다. 이때, R은 클러스터링이 되지 않은 점이다. R의 neighbor에서 jaccard 값이 가장 큰 점은 A이다. M1-graph와 같이 계속해서 진행하다가 점 I를 만났다. 점 I는 점 A가 들어가 있는 클러스터에 속한다. 따라서, 같은 클러스터를 두 번 만났기 때문에 종료한다. 이러한 종료조건이 M1-graph와의 차이점이다.

정리하자면, M2-graph의 수도코드는 아래와 같다:

1) Maximal Clique를 기반으로, 각각의 점이 클러스터에 중복해서 포함하지 않도록 제거한 clique를 초기 클러스터로 설정한다. 이때, 클러스터 되지 않은 점들을 루트 노드로 설정한다.
2) - 각각 루트 노드에서 여러 neighbor 중 가장 큰 edge jaccard 값을 가지는 neighbor를 선택한 후, 이를 자식 노드로 생성한다.
   - jaccard 값이 동일한 경우, 모든 경우를 나열하고, jaccard의 평균이 가장 높은 노드들의 집합을 얻는다.
3) 다음 자식 노드에서 해당 노드의 neighbor와의 jaccard 값이 부모 노드와의 jaccard 값보다 크다면 3) 과정을 반복한다.
   - jaccard 값이 같거나 작아진다면 그 때의 자식노드를 리프 노드로 설정하고 반복을 종료한다.
   - 가장 큰 jaccard 값을 가지는 자식 노드가 트리 내의 노드들 중에 속해 있는 클러스터에 포함된다면 반복을 종료한다.
   - 모든 neighbor가 클러스터에 포함이 되어있다면, 멈춘다.
4) 노드들의 집합을 하나의 클러스터로 병합한다. 이때, 2개 이상의 클러스터에 포함되어 있다면, 그 클러스터를 포함하여 하나의 클러스터로 병합한다.

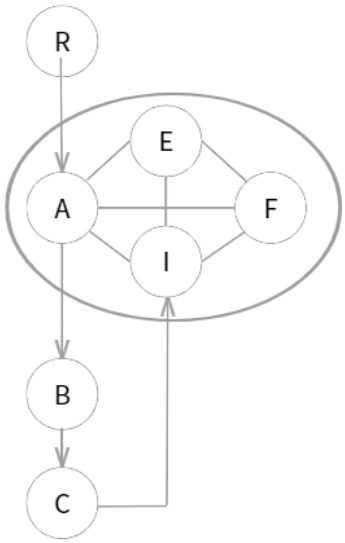

Fig 3. M2-Graph의 종료 조건

2.5 The M2-graph 한계점

M2-graph는 이미 트리 구조에 있는 점이 어떠한 클러스터에 들어가 있는데도 불구하고 그 클러스터에 포함된 다른 한 점을 만난다면 종료하는 조건을 추가했다. 주어진 유전자 데이터를 이용하여 결과를 확인였을 때, 가장 긴 클러스터의 크기가 59개로써, M1-graph 알고리즘의 결과처럼 하나의 클러스터가 비대해지는 결과는 개선되었다. 하지만, f1-score는 M1-graph에 비해 약 0.06 낮아짐을 확인하였다.

2.6 The M3-graph 알고리즘

M3-graph는 clique에다가 클러스터가 되지 않은 점들을 포함하는 방식으로 유의미한 값을 찾아내기 위해 다른 접근 방식을 시도 해봤다. 초기 클러스터에 포함되지 않은 노드를 클러스터링 하는 방식은 M2-graph의 알고리즘과 유사하다. 하지만, 크기가 2 이상인 클러스터를 만나면 멈추는 종료조건을 추가하였다는 점에서 M2-graph와 차이가 있다.

정리하자면, M3-graph의 수도코드는 아래와 같다:

1) Maximal Clique를 기반으로, 각각의 점이 클러스터에 중복해서 포함하지 않도록 제거한 clique를 초기 클러스터로 설정한다. 이때, 클러스터 되지 않은 점들을 루트 노드로 설정한다.
2) - 각각 루트 노드에서 여러 neighbor 중 가장 큰 edge jaccard 값을 가지는 neighbor를 선택한 후, 이를 자식 노드로 생성한다.
   - jaccard 값이 동일한 경우, 모든 경우를 나열하고, jaccard의 평균이 가장 높은 노드들의 집합을 얻는다.
   - 모든 neighbor가 클러스터에 포함이 되어있다면, 멈춘다.
3) 클러스터가 되지 않은 점의 neighbor들 중 가장 큰 edge jaccard 값을 가지는 점을 클러스터에 추가한다. 이때, 점이 2개 이상의 클러스터에 포함되어 있다면, 그 클러스터도 하나의 클러스터로 병합한다.
4) - 클러스터가 되지 않은 것(클러스터가 1개만 있는 것)이 두 개 이상의 노드로 구성된 클러스터를 만난 경우 멈춘다.
   - edge jaccard의 값이 이전 jaccard 값보다 같거나 크다면 2)를 계속하고 작다면, 멈춘다.
   - 이미 트리 구조에 있는 점이 어떠한 클러스터에 들어가 있는데 그 클러스터에 들어가 있는 다른 점을 또 만난다면 멈춘다.
5) jaccard의 값이 동일한 경우에는 모든 경우를 다 나열하여서, jaccard의 평균이 가장 높은 경우를 클러스터로 지정한다.

2.7 M3-graph의 한계점

M3-graph는 클러스터 되지 않은 점을 clique에다가 넣어 유의미한 값을 찾아내기 위한 새로운 접근 방식으로 도전해봤지만, 결국 f1-score는 M1-graph보다 0.08이나 낮아졌고 M2-graph에 비해 0.02이 낮게 나왔다.

3. 그래프 클러스터링 알고리즘 결과 및 해석

Table 1 각 알고리즘 비교 분석

Algorithm	F1-Score	Number of clusters	Maximum size of the cluster
Assignment 5	0.431	11928	9
Assignment 6	0.600	337	17
M1-graph	0.574	350	1479

M2-graph	0.515	557	59
M3-graph	0.496	663	29

M1-graph의 결과는 350개의 클러스터 개수와 가장 큰 클러스터의 크기는 1479개로 나왔다. f1-score가 0.574로 Assignment 5, M2-graph, M3-graph 보다 높게 나왔지만, Assignment 6보다는 적게 나왔다. M1-graph의 f1-score값이 나오게 된 이유는 한 클러스터 안에 많은 클러스터가 한 번에 들어가 있기 때문이라고 판단된다.

M2-graph의 결과는 클러스터의 개수는 557개, 클러스터의 최대 크기는 59개로 나온다. f1-score는 0.515로 Assignment 5에 비해 높고 Assignment 6, M1-graph에 비해 낮다. 하지만, Assignment 6와 비교하여 M2-graph는 데이터 손실이 없다. 또한, M1-graph의 클러스터 한개에 점들이 몰린다는 한계점을 보완하여서 M1-graph에 비해 상대적으로 균등하게 클러스터링 되었다.

M3-graph의 결과는 663개의 클러스터 개수를 가지고 클러스터의 가장 큰 크기는 27이다. f1-score는 0.496으로 Assignment 5보다는 높고 나머지보다는 낮다.

Assignment 6 가 가장 높게 f1-score가 높게 나왔다. 그 이유는 데이터의 손실이 있기 때문에 Ground-truth와 일치할 확률이 높아지기 때문이라고 판단된다.

4. 결론

본 논문에서는 M1-graph, M2-graph, M3-graph 순으로 각 알고리즘의 한계점을 분석한 후, 각 문제점들을 개선하기 위해 시도하였다.

M1, M2, M3-graph는 Assignment 6보다 f1-score는 낮지만, 데이터 손실은 없다는 장점이 있다. Assignment 5, Assignment 6은 동일 Jaccard 값이 발생되었을 때, 논리적인 척도로 클러스터링을 할 수 없었다는 단점이 있다. 본 논문에서 제시한 3개의 알고리즘은 이러한 문제점을 해결해준다. 후보들의 엣지 데이터를 바탕으로 그것들의 Jaccard 평균을 구하여 더 높은 Jaccard 평균을 가진 후보가 연쇄적인 유사성이 높다는 것을 확인할 수 있다. 이에 근거하여 후보군들의 첫 번째 노드가 동일 Jaccard를 가진 경우, 논리적으로 클러스터링을 할 수 있다.

Density가 1인 clique부터 시작을 하여 엣지 데이터를 바탕으로 클러스터링을 진행하기 때문에 더욱 더 유의미한 클러스터링을 할 수 있다.

M1- graph에서는 한 클러스터 안에 많은 노드가 한 번에 들어가있는 한계점이 있었다. M1-graph에서 클러스터링 되지 않은 한 점으로부터 트리 구조로 진행될 때, 고려할 jaccard 값으로 클러스터 내부의 점을 포함하지 않았다. 따라서, 트리를 구성하는 과정에서 클러스터가 다른 자식 노드들로 구성할 확률이 커지게 되었다. M1-graph의 F1-score가 0.574로 Assignment 5, M2-graph, M3-graph의 F1-score보다 개선된 결과를 보였다. 하지만, 초기 클러스터에 포함되지 않은 노드를 클러스터링 하는 과정에서 특정 클러스터가 비대해지는 결과를 보였다. 이에 따라 M1-graph 결과가 유의미하다고 단언할 수 없다.

M2-graph는 M1-graph의 한계점을 보완하기 위해 해당 알고리즘에서 jaccard 값을 구할 때, 클러스터 내부의 점도 포함이 되는 조건을 추가하였다. M2-graph의 클러스터 최대 크기가 59개라는 것을 통해 M1-graph보다 균등하게 클러스터 되었다는 것을 확인할 수 있다. 또한 M2-graph는 데이터 손실이 없음에도 불구하고, 데이터 손실이 난 Assignment 6와 클러스터의 최대 크기가 1479개인 M1-graph를 제외하면, F1-Score 가 가장 높다. 따라서, 데이터 손실이 없고 데이터가 몰려있지 않았음에도 불구하고 F1-Score가 가장 높기 때문에 유의미한 값들이 클러스터링이 되었다고 판단된다.

M3-graph는 M2-graph가 M1-graph보다 f1-score가 낮아지는 것을 보고 이를 보완하기 위해 새로운 방법으로 시도한 것이다. M3-graph는 M2-graph보다 더 균등하게 클러스터링 되었지만, f1-score를 보면, 유의미한 값들의 클러스터라고는 말 할 수 없다. 왜냐하면 클러스터되지 않은 점들이 clique를 만나면 바로 종료된다는 조건을 추가함으로써 clique들 간에 이어질 수 없었다. 따라서 두 클러스터 간의 연결성을 확인할 수 없었기 때문에 M3-graph의 F1-Score가 M2-graph의 F1-Score보다 낮게 나온 것으로 추론되었다.

M1, M2, M3-graph에 대해 f1-score, 전체 클러스터의 개수, 클러스터 최대 크기를 기준으로 분석해본 결과, M2-graph가 가장 유의미한 알고리즘이라는 것을 확인할 수 있었다.

5. 후기

5.1 이지윤 후기

본 논문을 작성하면서, graph clustering에 대한 정의와 이해도가 향상할 수 있는 기회가 되었다. 우리 TM팀은 아이디어 회의와 논의 및 토론을 거의 64시간 정도 했다. 그만큼 팀프로젝트를 하면서 자신이 생각한 로직과 그 논리성을 설명하는 것에 시간을 쏟았다. 자신의 생각을 말로 논리적으로 설명하는 것이 처음에는 어려웠지만, 이 프로젝트를 하면서 이러한 점들이 늘었고 커뮤니케이션도 원활히 진행이 되었던 것 같다. 사실 이 프로젝트를 어떻게 시작해야할지 정말 막막하였는데 알고리즘이라는 2학년 과목의 개념에 나오는 B-Tree, B+ -Tree등으로 데이터마이닝 분야에 적용하여 그것을 구현했다는 것이 신기했다. 이 프로젝트를 하면서 특히나, 한 분야에서 독창성 있는 새로운 것을 찾는 것은 어렵다는 것을 느꼈다. 앞으로는 한 분야 말고 여러 분야에 대한 공부를 하는 것이 좋은것을 깨달았다. 모든 분야가 다 연관성이 있다는 것을 느꼈다. 이 프로젝트를 통해 가장 많이 생각했던 것은 '유의미한 값'의 정의는 무엇인가? 이다. 정말 우리에게 골치아팠던 문제이다. 그래서 교수님께서 이에 대해 물어봤었는데 그러한 '유의미한 값'을 찾게 끔 하는 분야가 바이오 인포메틱스라고 하셔서 그 쪽 분야에 대해 흥미도 생겼다. '유의미한 값' 에 대해 만약 이 프로젝트를 진행하지 않았다면 신경도 쓰지 않았던 주제이지만, 이 프로젝트를 통해 추상적인것을 구체적으로 만드는 것도 결국 미래의 과제임을 깨달았다. 또한, 지금까지는 논문을 분석하는 일만 하다가 논문을 쓰는 입장이 되니까 논문을 분석하던 것이 더 잘 이해가 되었다. 이 프로젝트를 PM과 개발을 둘다 한 입장으로서 결국 프로젝트의 핵심은 의사소통임을 깨달았다. 이러한 알고리즘을 결코 혼자 구현할 수 없고 결국 다른 사람과 토론을 하면서 진행되기 때문이다. 마지막으로, 정말 다들

밤을 새며 프로젝트를 한 것에 대해 감사하다는 말을 남기고 싶다.

5.2 김송우 후기
논문을 찾아보면서 수업 시간에서 배운 그래프 마이닝 방법 외에도 많은 방법론과 그 방법론이 어떻게 쓰이는 지 알게 되었다. 그 중에서 Hierarchical based graph mining으로 segmentation을 방법론이 기억에 남았는데, bottom-up 방식과 top-down 방식을 섞어가며 클러스터링 하는 것이 기억에 남았다. 프로젝트를 진행하면서 아이디어를 수업시간에 배운 것을 어떻게 응용해야할지에 대해 고민이 있었다. 어느정도 주어진 데이터에 대해 클러스터링 과정을 처음부터 끝까지 진행해봄으로써 어떤 부분이 클러스터링 할 때 주요 문제점인지, 어떤 부분이 주로 고민하고 집중해야할 부분인지 조금을 알게 되었다. 다음엔 충분한 시간을 가지고 관련 프로젝트를 진행할 때에는 이번 프로젝트를 진행한 경험을 바탕으로 여러 논문을 살펴보며 정리한다면 더 좋은 결과를 낼 수 있을 것이라 생각하게 되었다. 또한, 이번에는 유전자의 관계성만을 가지고 클러스터링을 진행하였다. 이번 데이터마이닝을 수강하면서 여러 numerical 데이터들에 대해 응용방법에 대해서 다양한 방법 또한 배울 수 있었다. 따라서 이러한 Numerical 데이터를 이용하여 클러스터링할 방법에 대해서 고민해볼 수 있는 기회 또한 갖게 된다면 좋은 경험이 될 것 같다고 생각하였다. 마지막으로 프로젝트를 진행하며 설정한 방법론에 대해서 검증하고 맞는지 확인하는 과정에 대해서 팀 프로젝트 입장에서 어떻게 해야할지가 다음 팀 프로젝트에서 진행할 때 고민해야할 부분인 것 같다.

5.3 정유라 후기
본 논문을 작성하기 위해 graph, graph clustering의 정의에 공부를 하며 Density-Based Method, Partition-Based Method, Hierarchical Method에 대해 보다 잘 이해하게 되었다. 논문을 작성할 때, M1, M2, M3-graph에 대해 논의하면서 각 알고리즘의 문제점이 무엇인지, 문제점을 개선하기 위해서는 어떤 조건을 제시해야하는지 알 수 있었다. 또한 문제점에 대해 팀원들과 논의하며 본인이 해당 논문에 대해 잘 이해하고 있는지, 놓치고 있는 부분은 무엇인지에 대해 알게 되면서 나뿐만 아니라 팀원 그리고 제3자가 알고리즘에 대해 잘 이해하게 하려면 어떻게 해야하는지 많이 고민했다. 이러한 고민 끝에 알고리즘에 대한 통찰력이 향상되었다. 논문을 작성하면서 좋은 결과가 무엇인지에 대해 많이 고찰했던 것 같다. 아직 좋은 결과가 무엇인지에 대해 정확히 알지 못하지만 이번 기회에 생각해볼 기회가 있어서 좋았다. 그리고 M2-graph를 바이오 인포매틱스 분야뿐만 아니라 다른 분야에도 적용해보면 좋을 것 같다는 생각이 들었다.

본 연구는 데이터마이닝의 프로젝트로 수행된 연구결과임을 밝히며, 이런 알고리즘을 만들 기회를 주셔서 감사드립니다.

참 고 문 헌

[1] Satu Elisa Schaeffer, Graph clustering, Computer Science Review, Volume 1, Issue 1, (2007)

[2] Aggarwal, C.C. Graph Clustering. In: Sammut, C., Webb, G.I. (eds) Encyclopedia of Machine Learning. Springer, Boston, MA. (2011)

[3] Hao Li, Xiaojie Liu, Tao Li, Rundong Gan, A novel density-based clustering algorithm using nearest neighbor graph, Pattern Recognition, Volume 102, (2020)

[4] Niwattanakul, Suphakit, et al. "Using of Jaccard coefficient for keywords similarity." Proceedings of the international multiconference of engineers and computer scientists. Vol. 1. No. 6. (2013)

[5] Matsunaga, T., Yonemori, C., Tomita, E. *et al.* Clique-based data mining for related genes in a biomedical database. *BMC Bioinformatics* 10, 205 (2009)

[6] Palla, G., et al., “Uncovering the overlapping community structure of complex networks in nature and society”, Nature (2005)

[7] Qinghua Wu, Jin-Kao Hao,A review on algorithms for maximum clique problems, European Journal of Operational Research, Volume 242, Issue 3, Pages 693-709 (2015)

기여도 및 역할

- 이지윤, 김송우, 정유라
  PM
  - 프로젝트 관리
  - 회의록 작성, 회의 진행
  - 노션 제작, 계획 구축
  - 논문 틀 제작, 논문 본문 작성 및 검토(서론,본론, 결론)
  - 아이디어 제공
  - PPT 제작

  개발
  - Maximal Clique Algorithm 모든 cluster에 대한 중복 제거하여 개발
  - Merging By Common Neighbors Based on Maximal Clique 개발
  - M2-graph 수정
  - M3-graph 수정
  - f1-score 개발

자세한 기여도는 Notion에 기록하였습니다.
https://www.notion.so/9a15153e38af4cbca20127abbb5b736a 링크 참고 부탁드립니다.