그래프 임베딩 방법의 정확도-메모리 트레이드오프 분석

조성운 [1], Masoud Reyhani Hamedani[2], 김상욱 [3*]
한양대학교 컴퓨터소프트웨어학과 [1] 석사과정, [2] 연구조교수, [3] 교수

[1]seongun, [2]masoud, [3]wook@hanyang.ac.kr

Analyzing the Trade-off Between Accuracy and Memory Efficiency in Graph Embedding Methods

Seong-Un Cho[1], Masoud Reyhani Hamedani [2], Sang-Wook Kim[3*]
[1,2,3]Dept. of Computer Science, Hanyang University

요 약

일부 최신 그래프 연구는 정확도 향상에만 초점을 맞추거나 정확도보다 메모리 효율성을 우선적으로 고려한다. 그러나 효과적인 그래프 임베딩 방법은 정확도와 메모리 효율성을 동시에 제공해야 한다. 따라서 본 연구에서는 15 개의 최신 방법을 대상으로, 8 개의 실세계 데이터셋에서 정확도와 메모리 효율성 간의 트레이드오프를 비교 분석한다. 이를 통해 실세계 어플리케이션에서 효과적인 그래프 임베딩 방법 선택 시, 정확도와 메모리 효율성을 종합적으로 고려할 수 있는 실질적인 가이드를 제공한다. 실험 결과, 메모리 제한 환경에서는 그래프 압축 기반 방법이, 높은 정확도가 요구되는 어플리케이션에서는 유사도 기반 방법이 가장 효과적임을 확인하였다.

1. 서론

그래프 임베딩 방법은 그래프의 각 노드를 임베딩 스페이스에서 저차원 벡터로 표현하여 노드의 의미 정보와 커뮤니티 구조를 보존하는 것을 목표로 한다. 이렇게 얻어진 저차원 벡터 표현은 Link Prediction [1–2], Node Classification [3–4], 그래프 복원 [5], 추천 시스템 [6], 단어 유사성 분석 [7], 문서 분류 [7] 등 다양한 태스크에서 활용될 수 있다. 대규모 그래프 데이터가 점점 더 다양한 도메인에서 생성됨에 따라, 그래프 임베딩의 실제 적용 가능성을 평가하는 데 정확도와 메모리 효율성 간의 트레이드오프를 이해하는 것이 필수적이다. 예를 들어, 추천 시스템이나 소셜 네트워크 분석에서는 대규모 데이터를 효과적으로 처리하기 위해 메모리 효율성이 중요한 역할을 한다.

일부 최신 연구는 메모리 효율성을 간과한 채 정확도 향상에만 초점을 맞추는 반면, 다른 연구들은 정확도보다 메모리 효율성을 우선적으로 고려하는 데 중점을 두고 있다. 예를 들어, GELTOR [5]와 DUPLEX [8]는 각각 Learning to Rank 와 Hermitian 행렬 기반의 이중 Graph Attention Network 인코더를 임베딩 과정에 적용하여 정확도를 개선하려고 시도한다. 반면, MILE [9]과 GELSumm [10]은 임베딩 과정 전에 그래프 구조를 압축하여 메모리 효율성을 향상시키는 데 초점을 맞추고 있다. 그러나 효과적인 그래프 임베딩 방법은 높은 정확도와 최적화된 메모리 효율성을 동시에 제공해야 한다. 따라서 임베딩 방법을 평가하기 위해서는 정확도와 메모리 효율성을 동시에 고려하는 것이 중요하다.

본 연구에서는 다양한 크기의 실세계 데이터셋 8 개를 활용하여, 15 개의 최신 그래프 임베딩 방법의 정확도와 메모리 효율성 간의 트레이드오프를 비교 분석한다. 이를 위해 두 가지 주요 머신러닝 태스크인 Link Prediction 과 Node Classification 의 실험을 진행한다.

이러한 분석은 실세계 어플리케이션에 적합한 임베딩 방법을 선택하기 위한 분석적 참고자료로 활용될 수 있다. 광범위하게 수행된 실험 결과를 통해 다음과 같은 결론을 제시한다: 1) 메모리가 제한되거나 메모리 효율성이 중요한 환경에서는 그래프 압축 기반 방법이 적합하다. 2) 높은 정확도가 중요한 경우, 유사도 기반 방법이 가장 적합하며, 그 다음으로 Neural Network 기반 방법이 적합하다. 3) 본 논문은 정확도와 메모리 효율성 간의 트레이드오프를 분석하는 데 중점을 두었지만, 일부 임베딩 방법은 특정 그래프 유형에만 적용 가능하여 적용성이 제한적이다. 따라서 실세계 어플리케이션에 적합한 임베딩 방법을 선택할 때 환경적 제약과 데이터셋의 특성을 모두 고려해야 한다.

2. 임베딩 방법들

BoostNE [11]는 Multi-level approach 를 기반으로 행렬 분해를 반복적으로 수행하여 잔여 정보를 점진적으로 학습한다. DUPLEX [8]는 Hermitian 행렬 기반의 이중 Graph Attention Network 인코더를 활용해 학습한다. DeepWalk [1]는 랜덤 워크를 연결하여 그래프를 노드 시퀀스로 변환하고 언어 모델의 Skip-gram [12]을 적용한다. DWNS [6]는 DeepWalk 의 확장으로 적대적 학습 [6]을 활용하며, 학습 과정은 두 플레이어 간의 게임으로 구성된다. FREDE [4]는 Singular Value Decomposition 기반의 Frequent Direction 행렬 스케칭 [13]을 Random walk With Restart (RWR) 유사도 행렬에 적용한다. GELSumm [10]은 Degree Preserving Graph Summarization [14]를 활용하여 그래프를 압축하고, 압축된 그래프에 DeepWalk 를 적용한뒤 원래 크기로 복원한다. GELTOR [5]는 임베딩 공간에서 내적으로 계산된 임베딩 벡터의 순위를, AdaSim* 점수 [5]를 기반으로 계산된 그래프 노드의 순위와 일치하도록 조정한다. Newton 의 만유인력 이론에서 영감을 받아 설계된 GravityVAE [15]는 그래프 변분 자동 인코더를 활용하여 학습한다. MILE [9]은 Structural Equivalence Matching [16]과 Normalized Heavy Edge Matching [17]을 활용해 그래프를 압축하고, 압축된 그래프에서 학습한 임베딩을 원래 그래프로 복원한다. NetMF [18]는 DeepWalk 기반으로 저차원 연결 행렬을 Singular Value Decomposition 로 근사화하여 고차원 관계를 학습한다. Node2vec [2]은 DeepWalk 의 확장으로 랜덤 워크에서 특정 노드를 재방문하거나 새로운 노드를 방문할 가능성을 통제하여 편향된 랜덤 워크를 수행한다. NodeSketch [19]는 Hamming space 에서 잠재 벡터를 생성하며 임베딩 과정에 Recursive Sketching 을 적용한다. REFINE [20]은 skip-gram 모델에 직교 제약 조건을 추가하고, 랜덤화 블록 QR 분해를 통해 잠재 벡터를 학습한 뒤 그래프 확산 방법을 활용하여 임베딩을 생성한다. SigMaNet [21]은 스펙트럼 기반 Graph Convolutional Network 를 확장한 방법이며 Hermitian Laplacian 을 활용한다. VERSE [3]는 그래프에서 RWR 로 얻은 노드 간 유사성 분포와 임베딩 공간에서 내적을 통해 계산된 벡터 간 유사성 분포 사이의 교차 엔트로피를 최소화한다.

3. 실험

3.1 설정. 본 장에서는 데이터셋, 파라미터 설정, 머신러닝 태스크, 그리고 실험 환경에 대해 설명한다.

데이터셋. CoCit [3]은 데이터 마이닝, 데이터베이스 및 기계 학습 분야에서 작성된 논문의 인용 그래프로, 노드 라벨은 출판된 학회를 나타낸다. Cora [5]는 컴퓨터 과학 분야 논문의 인용 그래프이며, 노드 라벨은 연구 주제를 나타낸다. Epins [22]는 Epinions 웹사이트에서 소비자 리뷰로 구성된 신뢰 네트워크다. Last [22]는 Last.fm 웹사이트에서 소비자 선호도를 기반으로 한 신뢰 네트워크다. Pokec [23]은 슬로바키아의 온라인 소셜 네트워크로, 방향성이 있는 친구 관계를 포함한다. Google [24]은 2002 년에 Google 에서 공개된 웹 그래프다. Live [5]는 LiveJournal 웹사이트의 블로거들 간의 사회적 우정 그래프다. VK [3]는 러시아 소셜 네트워크에서 2017 년에 추출된 데이터다. 본 실험에서는 <표 1>에 요약된 8 개의 실세계 데이터셋을 사용하였다

<표 1> 데이터셋 통계

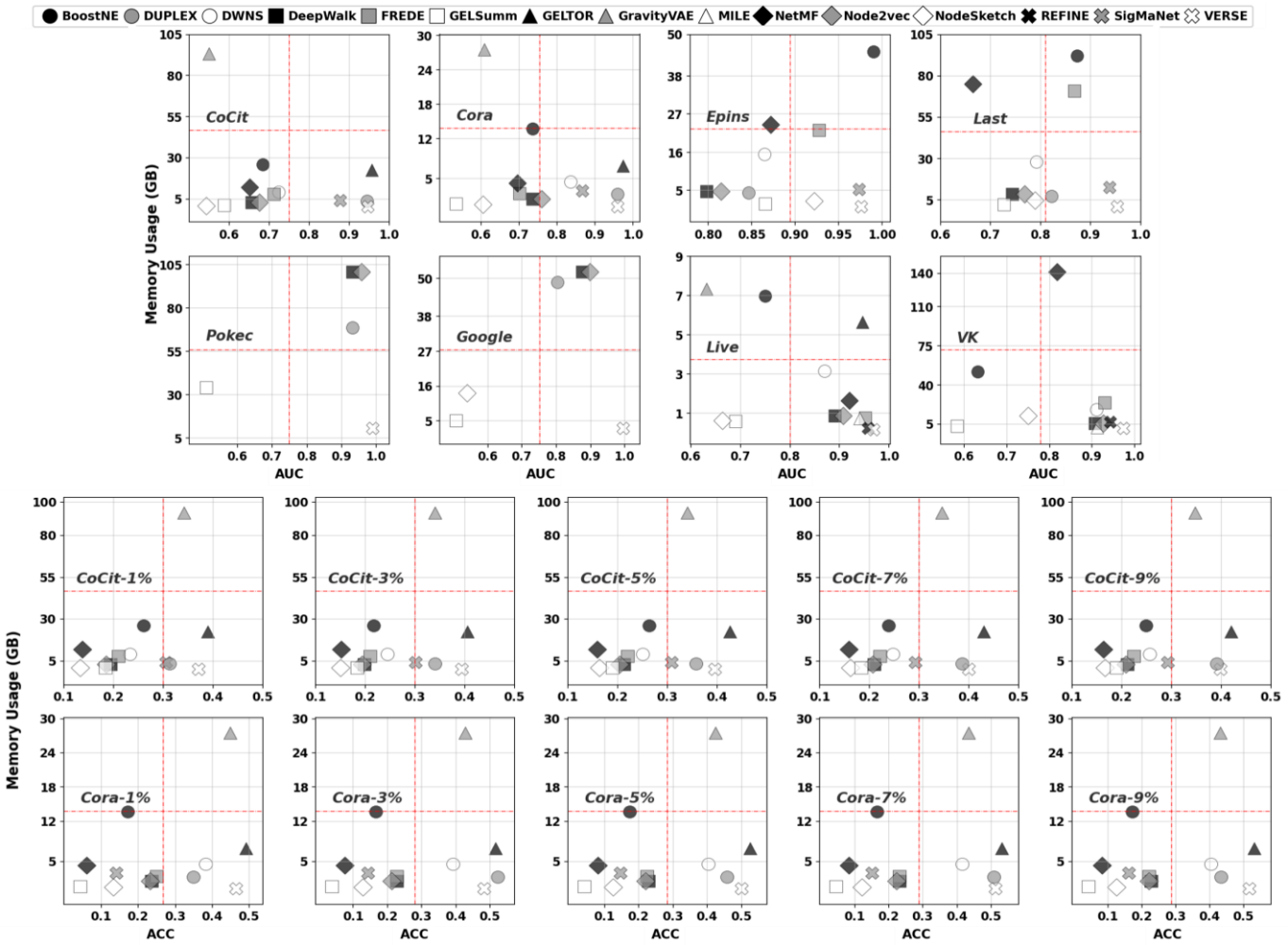
Datasets	|V|	|E|	#Labels	Type
CoCit	44K	195K	Full	Directed
Cora	23K	92K	Full	Directed
Epins	76K	406K	-	Directed
Last	136K	1.7M	-	Directed
Pokec	1.6M	31M	-	Directed
Google	876K	5.1M	-	Directed
Live	12K	160K	-	Undirected
VK	79K	2.7M	-	Undirected

파라미터 설정. 2 장에서 설명한 15 개의 임베딩 방법을 실험에 활용하였다. 모든 임베딩 방법은 저자들이 제공한 공개 소스 코드와 권장된 최적의 파라미터 설정을 적용하였으며 임베딩 차원은 128 로 통일하여 일관된 조건에서 실험을 진행하였다.

머신러닝 태스크. 1) Link Prediction 에서는 누락된 링크를 예측하는 것을 목표로 하며 2) Node Classification 에서는 노드의 라벨을 예측하는 것을 목표로 한다.

실험 환경. 본 실험은 Intel i9-10980XE CPU, 256GB RAM, RTX A6000 GPU(48GB 메모리)를 갖춘 Ubuntu 22.04 환경에서 진행되었다. 모든 임베딩 방법은 다섯 번 실행한 후, 결과의 평균값을 최종 정확도와 메모리 사용량으로 기록하였다.

3.2 Link Prediction. 각 데이터셋의 전체 링크 중 10%를 무작위로 제거하고 나머지는 연결된 상태로 유지한다. 제거된 링크는 양성 테스트 샘플(Test+)로 사용되며, 그래프에 남아 있는 링크는 양성 훈련 샘플(Train+)로 간주된다. 이후, 그래프에서 존재하지 않는 링크 중 제거된 링크 수와 동일한 개수(|Test+|)를 선택하여 음성 테스트 샘플(Test−)을 생성한다. Test−와 Train+ 모두에 포함되지 않는 링크 중 동일한 수를 무작위로 선택하여 음성 훈련 샘플(Train−)로 사용한다. 각 링크 (a, b)의 특징 벡터는 해당 노드 a 와 b 의 임베딩 벡터에 대해 Hadamard 연산을 적용하여 로지스틱 회귀 입력으로



(그림 1) Link Prediction (위)과 Node Classification(아래) 결과

사용하여 Area Under the Curve(AUC)를 통해 평가한다. 모든 방법의 AUC 와 메모리 사용량은 그림 1 의 상단에 표시되어 있으며, 각 데이터셋에 대해 별도의 하위 그림이 할당되어 있다.

3.3 Node Classification. 그래프의 노드 라벨을 예측하는 태스크로 라벨이 지정된 데이터셋(CoCit 및 Cora)에서만 수행된다. 각 데이터셋에서 노드 중 1%, 3%, 5%, 7%, 9%를 무작위로 선택하여 테스트 세트로 사용하고 나머지 노드는 훈련 세트로 활용한다. 각 노드의 임베딩 벡터는 로지스틱 회귀 모델의 입력으로 사용되며 훈련 세트의 데이터를 활용해 모델을 학습시킨다. 이후 테스트 세트의 노드 라벨을 예측하여 성능을 평가한다. 모델의 성능은 Macro-F1 점수(ACC)를 통해 측정되며, 이는 모든 클래스의 균형 잡힌 성능을 평가할 수 있는 지표로, 임베딩 방법이 노드의 의미적 정보를 얼마나 잘 학습했는지 확인할 수 있다. 모든 방법의 ACC 와 메모리 사용량은 그림 1 의 하단에서 확인할 수 있으며, CoCit 과 Cora 데이터셋에 대해 각각 노드 비율(1%, 3%, 5%, 7%, 9%)을 기준으로 진행한 실험 결과를 별도의 하위 그림으로 나타냈다.

3.4 실험 결과. 먼저, 적용성과 메모리 문제를 설명하고, 그 후에 정확도와 메모리 효율성 간의 트레이드오프에 대한 비교 분석을 진행한다.

적용성 문제. 그래프의 구조적 제약으로 인해 일부 방법은 특정 데이터셋에서 실험이 불가능했다. MILE 은 무방향 그래프에서만 작동 가능하므로 방향 그래프에서는 실험에 포함될 수 없었다. 반대로 DUPLEX 는 방향 그래프에서만 작동 가능하여 무방향 그래프에는 적용할 수 없었다. 마지막으로, REFINE 은 싱크 노드가 포함된 방향 그래프에서는 작동하지 않아 실험에 포함된 6 개의 방향 그래프에서 평가되지 못했다.

메모리 문제. 우리 실험 환경에서 일부 방법은 메모리 초과 문제로 인해 대규모 데이터셋에서 결과를 얻을 수 없었다. GELTOR 와 GravityVAE 는 Epins, Last, Pokec, Google, VK 데이터셋에서 메모리 초과 문제가 발생하였다. Google 데이터셋에서는 BoostNE, DUPLEX, DWNS, NetMF 가, Pokec 데이터셋에서는 위 방법들과 함께 NodeSketch 도 동일한 문제를 보였다.

정확도-메모리 트레이드오프 분석. 그림 1 에서 각 하위 그림은 네 개의 구역으로 나뉘며, 가장 우수한 방법은 낮은 메모리를 사용하면서 높은 정확도를 제공하는 **오른쪽 하단**에 위치한다.

Link Prediction 태스크에서는 GELTOR 와 VERSE 와 같은 유사도 기반 방법이 낮은 메모리 사용량과 높은 AUC 를 보여준다. 반면, NetMF 와 GravityVAE 는 높은 메모리 사용량과 낮은 AUC 를 기록하였다. Google 과 Pokec 과 같은 대규모 데이터셋에서는 메모리 초과로 인해 대부분의 방법이 실험 결과를 기록하지 못하였으나, 그래프 압축 기반 방법인 GELSumm 과 유사도 기반 방법인 VERSE 는 낮은 메모리 사용량을 유지하며 우수한 메모리 효율성을 보였다. 추가적으로, 무방향 그래프 데이터셋에서만 작동 가능한 MILE 은 Live 와 VK 에서 VERSE 다음으로 낮은 메모리를 기록하며 효율성을 입증하였다.

Node Classification 태스크에서도 유사한 경향이 나타났다. GELTOR 와 VERSE 는 낮은 메모리 사용량과 높은 ACC 를 기록하였고, Neural Network 기반 방법인 DUPLEX 는 이들 다음으로 높은 정확도를 보였다. 또한, 그래프 압축 기반 방법인 GELSumm 과 유사도 기반 방법인 VERSE 가 가장 낮은 메모리 사용량을 기록하며 효율성을 보여주었다.

4. 결론

본 연구에서는 15 개의 최신 그래프 임베딩 방법을 대상으로, 다양한 크기와 특성을 가진 8 개의 실세계 데이터셋에서 정확도와 메모리 효율성 간의 트레이드오프를 비교 분석하였다. 실험 결과, 메모리가 제한된 환경에서는 그래프 압축 기반 방법이 적합하며, 높은 정확도가 요구되는 어플리케이션에서는 유사도 기반 방법이 가장 적합했다. 특히, 유사도 기반 방법 중 하나인 VERSE 는 높은 정확도와 낮은 메모리 사용량을 동시에 제공하며, 다양한 환경에서 가장 적합한 그래프 임베딩 방법으로 나타났다.

사사

본 논문은 (1)정부(과학기술정보통신부)의 재원으로 정보통신기획평가원의 지원(No.RS-2022-00155586)과 (2)정부(과학기술정보통신부)의 재원으로 정보통신기획평가원의 지원 (No.RS-2020-II201373, 인공지능대학원지원(한양대학교))을 받아 수행된 연구임.

참고문헌

[1] Bryan Perozzi et al., “DeepWalk: Online Learning of Social Representations” In *Proc. of ACM KDD*, 2014, pp. 701–710.

[2] Aditya Grover et al., “node2vec: Scalable Feature Learning for Networks”, In *Proc. of ACM KDD*, 2016, pp. 855–864.

[3] Anton Tsitsulin et al., “VERSE: Versatile Graph Embeddings from Similarity Measures”, In Proc. of WWW, 2018, pp. 539–548.

[4] Anton Tsitsulin et al., “FREDE: Anytime Graph Embeddings”, The VLDB Endowment 14, 6, 2021, pp. 1102–1110.

[5] Masoud Rehyani Hamedani et al., “GELTOR: A Graph Embedding Method Based on Listwise Learning to Rank”, In Proc. of WWW, 2023, pp. 6–16.

[6] Quanyu Dai et al., “Adversarial Training Methods for Network Embedding”, In Proc. of WWW, 2019, pp. 329 339.

[7] Jian Tang et al., “LINE: Large-Scale Information Network Embedding”, In Proc. WWW, 2015, pp. 1067–1077.

[8] Zhaoru Ke et al., “DUPLEX: Dual GAT for Complex Embedding of Directed Graphs”, In Proc. of ICML, 2024, pp. 1–19.

[9] Jiongqian Liang et al., “MILE: A Multi-Level Framework for Scalable Graph Embedding”, In Proc. of AAAI, 2021, pp. 361–372.

[10] Houquan Zhou et al., “A Provable Framework of Learning Graph Embeddings via Summarization”, In Proc. of AAAI, 37, 4, 2023, pp. 4946–4953.

[11] Jundong Li et al., “Multi-Level Network Embedding with Boosted Low-Rank Matrix Approximation”, In Proc. of ASONAM, 2019, pp. 49–56.

[12] Tomas Mikolov, "Efficient estimation of word representations in vector space" *arXiv preprint arXiv:1301.3781* 3781, 2013.

[13] Edo Liberty, "Simple and deterministic matrix sketching" In *Proc of ACM SIGKDD*, 2013, pp. 581–588.

[14] Houquan Zhou et al., “DPGS: Degree-Preserving Graph Summarization” In Proc. of SDM, 2021, pp. 280–288.

[15] Guillaume Salha et al., “Gravity-Inspired Graph Autoencoders for Directed Link Prediction”, In Proc. of ACM CIKM, 2019, pp. 589–598.

[16] George Karypis et al., “A fast and high quality multilevel scheme for partitioning irregular graphs”, SIAM Journal on scientific Computing, 20, 1, 1998, pp. 359–392.

[17] George Karypis el al., “Multilevelk-way Partitioning Scheme for Irregular Graphs”, Journal of Parallel and Distributed computing, 48, 1, 1998, pp. 96–129.

[18] Jiezhong Qiu et al., “Network Embedding as Matrix Factorization: Unifying DeepWalk, LINE, PTE, and node2vec.” In Proc. of ACM WSDM, 2018, pp. 459–467.

[19] Dingqi Yang et al., “NodeSketch: Highly-Efficient Graph Embeddings via Recursive Sketching”, In Proc. of ACM KDD, 2019, pp. 1162–1172.

[20] Hao Zhu et al., “REFINE: Random Range Finder for Network Embedding”, In Proc. of ACM CIKM, 2021, pp. 3682–3686.

[21] Fiorini Stefano et al., “SigMaNet: One Laplacian to Rule Them All”, In Proc. of AAAI, 37, 6, 2023, pp. 7568–7576.

[22] Sheng Zhou et al., “Direction-Aware User Recommendation Based on Asymmetric Network Embedding” ACM Transactions on Information Systems, 40, 2, 2021, pp.1–23.

[23] Shanshan Feng et al., "Role: Rotated lorentzian graph embedding model for asymmetric proximity." IEEE Transactions on Knowledge and Data Engineering, 35, 9, 2022, pp. 9140–9153.

[24] Weiren Yu et al., "SimRank*: Effective and scalable pairwise similarity search based on graph topology." The VLDB Journal, 28, 2019, pp. 401–426.