http://www.jsebs.org
pISSN 2288-3908
eISSN 2765-3846

The Journal of Society for e-Business Studies
Vol.26, No.3, August 2021, pp.67-80
https://doi.org/10.7838/jsebs.2021.26.3.067

# 인접성 벡터를 이용한 트리플 지식 그래프의 임베딩 모델 개선

## Improving Embedding Model for Triple Knowledge Graph Using Neighborliness Vector

조새롬(Sae-rom Cho)*, 김한준(Han-joon Kim)**

### 초 록

그래프 표현 학습을 위한 노드 임베딩 기법은 그래프 마이닝에서 양질의 결과를 얻는 데 중요한 역할을 한다. 지금까지 대표적인 노드 임베딩 기법은 동종 그래프를 대상으로 연구되었기에, 간선 별로 고유한 의미를 갖는 지식 그래프를 학습하는 데 어려움이 있었다. 이러한 문제를 해결하고자, 기존 Triple2Vec 기법은 지식 그래프의 노드 쌍과 간선을 하나의 노드로 갖는 트리플 그래프를 학습하여 임베딩 모델을 구축한다. 하지만 Triple2Vec 임베딩 모델은 트리플 노드 간 관련성을 단순한 척도로 산정하기 때문에 성능을 높이는데 한계를 가진다. 이에 본 논문은 Triple2Vec 임베딩 모델을 개선하기 위한 그래프 합성곱 신경망 기반의 특징 추출 기법을 제안한다. 제안 기법은 트리플 그래프의 인접성 벡터(Neighborliness Vector)를 추출하여 트리플 그래프에 대해 노드 별로 이웃한 노드 간 관계성을 학습한다. 본 논문은 DBLP, DBpedia, IMDB 데이터셋을 활용한 카테고리 분류 실험을 통해, 제안 기법을 적용한 임베딩 모델이 기존 Triple2Vec 모델보다 우수함을 입증한다.

### ABSTRACT

The node embedding technique for learning graph representation plays an important role in obtaining good quality results in graph mining. Until now, representative node embedding techniques have been studied for homogeneous graphs, and thus it is difficult to learn knowledge graphs with unique meanings for each edge. To resolve this problem, the conventional Triple2Vec technique builds an embedding model by learning a triple graph having a node pair and an edge of the knowledge graph as one node. However, the Triple2 Vec embedding model has limitations in improving performance because it calculates the relationship between triple nodes as a simple measure. Therefore, this paper proposes a feature extraction technique based on a graph convolutional neural network to improve the Triple2Vec embedding model. The proposed method extracts the neighborliness vector

본 연구는 2018년도 정부(교육부)의 재원으로 한국연구재단의 지원을 받아 수행된 기초연구사업(No. NRF-2018R1D1A1A02086148)이며, 또한 과학기술정보통신부 및 정보통신기술진흥센터의 대학 ICT 연구센터지원사업의 연구결과로 수행되었음(IITP-2021-2018-0-01417).

* First Author, Master, School of Electrical and Computer Engineering, University of Seoul(csrom0128@gmail.com)
** Corresponding Author, Professor, School of Electrical and Computer Engineering, University of Seoul (khj@uos.ac.kr)

Received: 2021-07-06, Review completed: 2021-08-10, Accepted: 2021-08-18

of the triple graph and learns the relationship between neighboring nodes for each node in the triple graph. We proves that the embedding model applying the proposed method is superior to the existing Triple2Vec model through category classification experiments using DBLP, DBpedia, and IMDB datasets.

**키워드** : 지식 그래프, 노드 임베딩, 트리플 그래프, 합성곱 신경망, 그래프 특징 추출, 머신러닝
Knowledge Graph, Node Embedding, Triple Graph, Convolutional Network, Graph Feature Extraction, Machine Learning

# 1. 서 론

최근 그래프에 머신러닝 기법을 적용하기 위한 그래프 임베딩 기법이 활발히 연구되고 있다. 대표적인 그래프 임베딩 기법은 GNN[17], Node2Vec[4], DeepWalk[14] 등이 있으며, 이들은 소셜 네트워크와 같이 노드(Node)와 간선(Edge) 종류가 동일한 동종 그래프(Homogeneous Graph)를 대상으로 연구되었다[3]. 하지만 실세계에서는 다양한 노드와 간선 종류를 갖는 지식 그래프(Knowledge Graph)가 주로 활용됨에 따라 지식 그래프에 대한 표현 학습의 중요성이 대두되게 되었다. 이에 Metapath2Vec [1], R-GCN[18] 등 지식 그래프의 임베딩 기법이 연구되었지만, 결과적으로 주어진 그래프를 노드 단위로 임베딩하기 때문에 간선 별로 고유한 의미를 가진 노드 벡터를 얻지 못하는 한계점을 가진다.

이후에 등장한 Triple2Vec[2]은 지식 그래프의 노드 쌍과 간선을 하나의 노드로 간주하는 가중 트리플 그래프를 생성하여 각 간선의 의미를 구분한 트리플 노드 임베딩 모델을 구축한다. 하지만 Triple2Vec은 트리플 그래프의 노드 간 관계를 단순화된 척도로 계산하기 때문에 지식 그래프의 특징을 구체적으로 반영하지 못하고 임베딩 모델의 성능 한계를 초래한다는 문제점이 있다.

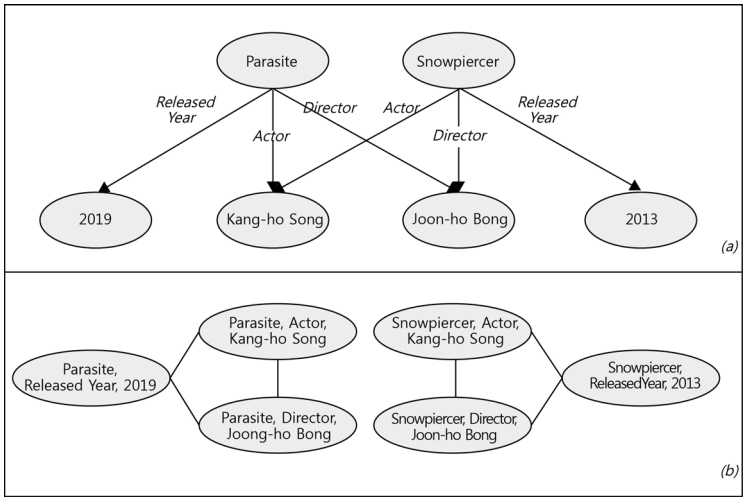
이에 본 논문은 Triple2Vec 임베딩 모델을 개선하기 위한 그래프 합성곱 신경망 기반의 특징 추출 기법을 제안한다. 제안 기법은 트리플 그래프의 연쇄적인 이웃 노드 간 관계 정보를 학습한 인접성 벡터(Neighborliness Vector)를 추출하여 지식 그래프의 구조와 의미 정보 학습에 더욱 유용한 가중 트리플 그래프를 생성한다. 본 논문이 제안하는 그래프 임베딩 모델을 Weighted Triple2Vec이라 명명하며, DBLP, DBpedia, IMDB 지식 그래프 데이터셋을 활용한 카테고리 분류 실험을 통해 제안하는 Weighted Triple2Vec 모델이 우수함을 입증한다.

# 2. 관련 연구

## 2.1 Triple2Vec

### 2.1.1 트리플 지식 그래프

지식 그래프 $G=(V_G, E_G)$는 객체와 객체 간 관계를 노드와 간선으로 표현한 그래프이다. <Figure 1(a)>는 영화 정보를 지식 그래프 형식으로 나타낸 예시이다. 이때, $V_G$는 노드의 집합으로 특정 객체(예: Parasite, Snowpiercer)와 해당 속성을 나타내는 속성 객체(예: Kang-ho




〈Figure 1〉 Knowledge Graph vs. Triple Knowledge Graph

Song, 2019 등)를 포함한다. $E_G$는 간선의 집합을 의미하며 객체 간의 관계(예: Released Year, Actor 등)를 뜻한다. 지식 그래프는 1개의 노드가 1개 이상의 간선을 갖는 특성이 있다. 따라서 지식 그래프의 간선에 따른 노드 간 의미를 구분하기 위해서는 지식 그래프의 노드와 간선을 트리플 형식으로 재구성해야 한다.

트리플 지식 그래프(Triple Knowledge Graph)는 지식 그래프의 노드를 연결된 간선에 따라 트리플 형식의 노드로 재구성한 그래프로서, 본 논문은 이를 간략히 트리플 그래프라고 칭한다. 트리플 노드는 특정 객체 노드를 '주어(Subject)'로 보고, 해당 노드의 속성 정보를 가진 노드는 '목적어(Object)', 두 노드를 연결하는 간선은 관계의 속성 종류를 설명하는 '술어(Predicate)' 역할을 수행한다고 간주한다. 여기서, '술어'는 '주어'와 '목적어'간의 관계를 설명하는 요소로서, 동사, 형용사, 명사 등의 품사를 가진다. 요약하면, 트리플 노드는 ('주어', '술어', '목적어')의 구조를 가지며 간단하게 $(s, p, o)$의 구성으로 표현한다. <Figure 1(b)>는 <Figure 1(a)>를 트리플 그래프로 변환한 것으로서, 지식 그래프의 노드 쌍과 간선을 트리플 노드(예: Parasite, ReleasedYear, 2019; Snowpiercer, Actor, Kang-ho Song) 등)로 재구성하고, 트리플 노드 간 $s$ 또는 $o$가 같으면 간선을 연결하는 과정을 거쳐 생성한다.

트리플 그래프는 지식 그래프에 비해 크기가 크다는 단점이 있다. 이러한 특징은 트리플 그래프를 학습할 때 많은 시간을 소요하게 하므로, 트리플 그래프를 효과적으로 학습하기 위해서는 트리플 그래프에 적절한 가중치를 부여하는 것이 바람직하다.

### 2.1.2 Triple2Vec의 학습 과정

Triple2Vec은 크게 3단계 학습 과정을 거친다. 1단계는 가중 트리플 그래프를 만드는 과정이다. 트리플 그래프 $G_T=(V_T, E_T, w)$는 트리플 노드 쌍의 관련성을 나타내기 위해 간선에 가중치 $w$를 부여한다. Triple2Vec은 $w$를 구하

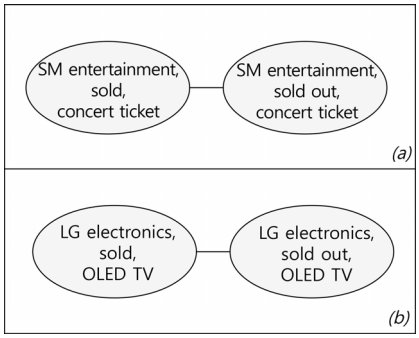
기 위해 트리플 노드 간 동시출현 확률을 기반으로 유사도를 측정한다[15]. 2단계는 가중 트리플 그래프의 경로를 추출하는 과정이다. 단어를 임베딩하기 위해서는 말뭉치가 필요한 것처럼, 그래프를 임베딩하기 위해서는 단절된 그래프 경로(Truncated Graph Walks)가 필요하다[2]. 그래프의 경로는 특정 노드에서 출발하여 연결된 간선을 따라가며 구성하며, 경로를 구성하는 노드는 가중 트리플 그래프의 간선에 부여된 가중치 $w$가 높을수록 우선하여 추출한다. 3단계는 추출한 경로를 학습하는 과정이다. Triple2Vec은 추출한 경로를 Skip-gram 모델[12]로 학습하여 트리플 그래프에 대한 임베딩 모델을 구축한다. 생성한 임베딩 모델은 지식 그래프의 노드 연결 구조와 노드가 갖는 간선의 의미 정보를 모두 내포한다.

### 2.1.3 Triple2Vec의 한계점

Triple2Vec은 지식 그래프를 트리플 그래프로 변환하여 지식 그래프에 대한 임베딩 모델을 구축한다. 이러한 Triple2Vec의 핵심 아이디어는 지식 그래프의 구조와 의미 정보를 저차원 공간에 투영시키기에 유용하다. 그러나 Triple2Vec이 사용하는 트리플 그래프의 노드 간 가중치 계산 방식은 임베딩 모델의 성능 저하를 초래하는 2가지의 문제점을 가진다.

첫째, 가중치 측정 대상의 단순화로 인한 신뢰성 저하이다. 먼저 Triple2Vec은 트리플 그래프의 노드 간 관계를 $p$의 관계로 정의한다[2]. 이는 $p$가 지식 그래프 간선에 해당하는 술어로서, 지식 그래프에서 트리플 노드를 구성하는 기준이 되기 때문이다. Triple2Vec은 트리플 노드 쌍 $(s, p_i, o)$와 $(s, p_j, o)$가 있을 때, 동일한 $s$와 $o$가 사용된 트리플 노드는 문맥적으로 비슷한 술어를 사용하고 높은 연관성을 가진다고 간주한다. 이러한 가정하에 트리플 노드 간 간선에 부여하는 가중치는 동일한 $s$와 $o$를 사용하는 트리플 노드 쌍의 경우에 한하여 산출한 $(p_i, p_j)$의 동시출현 확률을 기반으로 한다. 하지만 해당 척도는 $s$ 또는 $o$ 하나만 일치한 트리플 노드 쌍의 관련성을 고려하지 않으므로 그 가중치의 신뢰도를 저하시킨다.

둘째, 가중치 계산 방법의 일반화로 인한 정보의 손실이다. Triple2Vec은 트리플 그래프의 간선에 노드 간 연관성을 표현하는 가중치를 부여하기 위해 모든 $p$쌍에 대한 동시출현 행렬(Co-occurrence Matrix)을 구성한다. 그리고 트리플 노드 쌍 $(s, p_i, o)$와 $(s, p_j, o)$가 있을 때 간선의 가중치는 동시출현 행렬의 i번째 행과 j번째 행의 코사인 유사도(Cosine Similarity)이다[15]. 해당 계산 방식은 트리플 노드 쌍에 사용된 술어의 관계만 파악하면 가중치를 쉽게 계산할 수 있다는 장점이 있지만, 트리플 노드에 사용된 $s$와 $o$에 상관없이 동일한 $(p_i, p_j)$를 가지면 가중치는 모두 같은 값으로 측정된다는 문제점이 있다.




〈Figure 2〉 Two Pairs of Triple Nodes with the Same Predicate Relationship

예를 들어, <Figure 2(a)>, <Figure 2(b)>는 트리플 노드 쌍의 술어 관계가 각각 'sold'와 'sold out'으로 동일하다. 그러나 트리플 노드를 구성하는 $s$와 $o$를 살펴보면, 각 노드 쌍은 서로 다른 연관성을 지닌다. 예를 들어, <Figure 2(a)>의 트리플 노드 쌍은 $s$가 연예기획사인 'SM entertainment'이고, $o$는 'sold' 또는 'sold out'된 'concert ticket'이다. 이 경우, 'concert ticket'은 판매량 대비 매진 비율이 높으므로 트리플 노드 쌍의 술어 관계는 높은 연관성을 가진다. 반면, <Figure 2(b)>의 트리플 노드 쌍은 $s$가 전자제품을 파는 대기업인 'LG electronics'이고 $o$는 기업의 전자제품인 'OLED TV'로, 전자제품은 많은 판매량에 비해 매진될 확률이 낮으므로 연관 있는 술어 관계라고 보기 어렵다. 즉, 동일한 술어 관계를 갖더라도 객체의 특성에 따라 트리플 노드 쌍은 다른 연관성을 가지지만 Triple2Vec은 이를 일반화하여 동일하게 간주함으로써 트리플 노드 간 구체적인 관계의 의미를 손실한다.

Triple2Vec 기법의 가중치 계산 방식은 지식 그래프의 특징을 적절하게 반영하지 못한다. 트리플 그래프의 노드 간 가중치는 임베딩 학습을 위한 입력 데이터의 구성을 좌우하므로, 결국 이러한 문제점은 임베딩 모델의 성능 저하를 야기한다.

### 2.2 GCN

합성곱 신경망(Convolutional Network)을 통해 특징을 추출하는 CNN[9]은 동일한 크기의 필터를 이용한 합성곱 연산을 통해 효과적으로 지역 정보를 학습한다. 마찬가지로 그래프에 합성곱 연산을 적용하는 기법인 GCN(Graph Convolutional Network)[8] 또한 인접한 노드들의 정보를 집약하여 그래프의 노드 임베딩을 수행한다. GCN은 그래프를 인접행렬과 특징행렬로 표현하여 합성곱 연산을 수행한다. 인접행렬 $A \in \boldsymbol{R}^{N \times N}$은 $N$개의 노드를 가진 그래프의 노드 간 연결 유무를 표현한 행렬이다[17]. 그래프의 i번째 노드와 j번째 노드가 연결되어 있으면 $A_{i,j}$는 1값을, 반대의 경우는 0값을 가진다. 이때, 인접행렬은 종류에 따라 이진값이 아닌 가중값을 가질 수 있다[8]. 특징행렬 $F \in \boldsymbol{R}^{N \times C}$은 노드를 $C$차원의 특징으로 나타낸 행렬이다. 예를 들어, 논문 인용 그래프(Citation Graph)와 같이 한 노드가 논문을 나타낸다면, 특징행렬의 열은 논문에 쓰인 단어 집합이 된다. GCN은 준비된 인접행렬과 특징행렬을 합성곱층에 통과시키며 가중치 행렬과의 합성곱 연산을 수행하고, 합성곱 연산을 거듭하며 노드의 연쇄적인 이웃 정보를 학습한다.

## 3. 제안 기법

본 논문은 트리플 그래프의 노드 임베딩 모델을 개선하기 위한 트리플 그래프의 특징 추출 기법을 제안한다. 트리플 그래프는 지식 그래프의 구조를 재구성한 형태로, 트리플 그래프의 특징을 추출하는 것은 곧 지식 그래프의 특징을 추출하는 것을 의미한다. 트리플 그래프의 간선에 부여된 가중치는 임베딩 학습을 위한 입력 데이터인 그래프의 경로 집합의 구성을 결정하게 된다. 따라서 지식 그래프를 임베딩하기 위해서는 양질의 가중 트리플 그래프가 요구된다.

### 3.1 트리플 그래프의 인접성 벡터

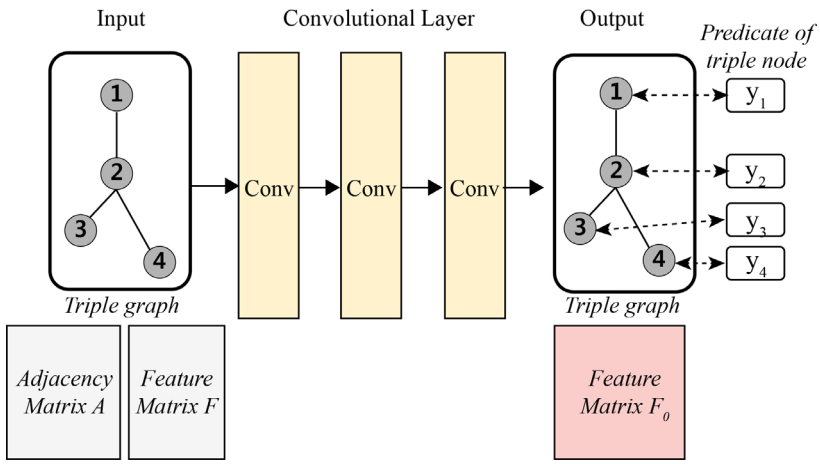
<Definition 1> **인접성 벡터(Neighborliness Vector)**는 특정 트리플 노드에 대해 연쇄적으로 인접한 이웃 트리플 노드의 관계 정보를 집약적으로 내포한 특징 벡터이다.

지식 그래프의 특징을 효과적으로 반영하는 가중 트리플 그래프를 생성하기 위해서는 트리플 그래프의 2가지 특징을 반영해야 한다. 첫 번째는 모든 그래프가 갖는 특성으로서, 노드가 갖는 연쇄적인 연관성을 파악해야 한다. 그래프의 간선은 관련 있는 노드를 연결하는 역할을 한다. 따라서 노드 간 연관성을 파악하기 위해서는 단순히 특정 노드에 이웃한 노드를 파악하는 것이 아니라, 이웃 노드에 연쇄적으로 연결된 노드들까지 학습해야 한다. 두 번째는 트리플 노드가 가지는 고유한 의미를 파악해야 한다. 트리플 노드는 2.1.3절에서 다루었듯이 이웃 노드들과 맺는 연관성을 통해 의미하는 바가 달라진다. 즉, 트리플 노드의 의미를 파악하기 위해서는 관련 있는 노드들과 이루는 구체적인 관계를 파악해야 한다.

따라서 본 논문에서는 GCN을 활용하여 연쇄적으로 인접한 트리플 노드를 학습하고, 이를 집약한 정보인 인접성 벡터(Neighborliness Vector)를 추출하고자 한다. GCN은 본래 동종 그래프를 대상으로 학습하기 때문에 지식 그래프는 학습이 어렵지만[8], 본 논문은 지식 그래프를 동일한 하나의 간선을 갖는 트리플 그래프로 변환함으로써 해당 문제점을 해결하였다.

### 3.2 인접성 벡터 추출 과정

제안하는 트리플 그래프의 인접성 벡터 추출 기법의 아키텍처는 <Figure 3>으로 나타낸다. 제안 기법의 입력 데이터는 트리플 그래프의 인접행렬과 특징행렬이다. 초기에 생성하는 특징행렬은 트리플 노드의 특징을 나타내야 한다. 트리플 노드의 의미는 이웃 노드들과 맺는 연관성을 통해 추정할 수 있으므로, 각 트리플 노드에 대해 이웃한 $p$들과 갖는 관계의 비중을 구해야 한다.




〈Figure 3〉 Neighborliness Vector Extraction Process of Triple Graph

〈Algorithm 1〉 Algorithm of Generating Feature Matrix of Triple Graph

**Input :** Triple Graph $G_T$
**Output :** Feature Matrix $F$

1: $G_T = \{V_T, E_T\}$
2: **for all** $v = (s, p, o) \in V_T$ **do**
3: let $N(v)$ be a set of neighbor nodes of $v$
4: **for all** $v' = (s', p', o') \in N(v)$ **do**
5: add $p'$ to $\mathrm{Pred}_{neighbor}(v)$
6: **end for**
7: **if** $p$ not in $\mathrm{Pred}_{tot}$ **then**
8: add $p$ to $\mathrm{Pred}_{tot}$
9: **end if**
10: **end for**
11: N = number of $V_T$
12: P = number of $\mathrm{Pred}_{tot}$
13: let F be a new N X P matrix
14: **for all** $v \in V_T$ **do**
15: **for all** $p_i \in \mathrm{Pred}_{tot}$ **do**
16: **if** $p_i \in \mathrm{Pred}_{neighbor}(v)$ **then**
17: $Relatedness(v, p_i) = \frac{number\ of\ p_i \in \mathrm{Pred}_{neighbor}(v)}{number\ of\ \mathrm{Pred}_{neighbor}(v)}$
18: **else**
19: $Relatedness(v, p_i) = 0.0$
20: **end if**
21: $F_{v,p_i} = Relatedness(v, p_i)$
22: **end for**
23: **end for**
24: **return** $F$

<Algorithm 1>은 트리플 그래프의 특징을 반영한 초기 특징행렬의 생성 과정을 보여준다. 트리플 그래프는 트리플 노드와 간선으로 구성되며(Line 1), 각 트리플 노드 $v \in V_T$는 이웃 노드 집합 $N(v)$을 가진다(Line 2-3). 트리플 노드는 이웃 노드에 따라 서로 다른 술어 관계를 맺으며, 전체 트리플 그래프에 사용된 각 술어에 대해 특정 노드와 이루는 비중을 계산하기 위해서는 트리플 노드의 이웃 술어 집합인 $\mathrm{Pred}_{neighbor}(v)$(Line 4-6)와 전체 술어 집합인 $\mathrm{Pred}_{tot}(v)$을 정의해야 한다(Line 7-10).

$$F_{v,p'} = Relatedness(v, p') \tag{1}$$

$$= \begin{cases} \dfrac{n(p' \in \mathrm{Pred}_{neighbor}(v))}{n(\mathrm{Pred}_{neighbor}(v))} & \text{if } p' \in \mathrm{Pred}_{neighbor}(v), \\ 0.0 & otherwise \end{cases}$$

그리고 초기화된 트리플 그래프의 특징행렬인 $F \in \boldsymbol{R}^{|V_T| \times |\mathrm{Pred}_{tot}|}$ (Line 11-13)의 값 $F_{v,p'}$는 $v \in V_T$, $p' \in \mathrm{Pred}_{tot}$을 만족하며 수식 (1)을 적용하여 특정 노드 $v$에 대해 이웃한 술어 $p'$와의 관련성(Relatedness)을 계산한다. 관련성은 특정 트리플 노드에 대해 연관 있는 술어의 중요도로서 이웃한 트리플 노드들을 구성하는 전체 술어 빈도수 $n(\mathrm{Pred}_{neighbor}(v))$ 대비 각 술어의 출현 빈도 $n(p' \in \mathrm{Pred}_{neighbor}(v))$을 구한다. 각 트리플 노드에 대해 $\mathrm{Pred}_{neighbor}(v)$에 속한 이웃한 술어들과의 관련성을 모두 계산하면 최종적으로 특징행렬 $F$가 생성되고(Line 14-24), $F$의 행벡터는 특정 트리플 노드를 이웃한 술어들과 맺는 관련성으로 나타내게 된다.

트리플 그래프의 인접행렬과 특징행렬은 합성곱층을 거치며 가중치 행렬과의 합성곱 연산을 수행한다. 초기의 특징행렬 $F$는 특정 트리플 노드에 대해 근접한 트리플 노드와의 연관성을 나타내지만, 합성곱층을 거치면서는 멀리 이웃한 트리플 노드의 특징까지 학습한다. 최종적으로 출력되는 행렬은 $F_o \in \boldsymbol{R}^{|V_T| \times |\mathrm{Pred}_{tot}|}$ 이며, 각 트리플 노드는 연쇄적으로 연관된 술어와의 조합으로 설명할 수 있다. 이때, 트리플 노드의 카테고리는 해당 트리플 노드의 $p$로, 연관된 술어들의 조합이 $p$를 설명할 수 있도록, 교차 엔트로피 손실함수(Cross Entropy Loss

Function)를 통해 가중치 행렬을 업데이트한다.

### 3.3 인접성 벡터 기반의 Weighted Triple 2Vec

3.3절에서는 트리플 그래프의 인접성 벡터를 이용하여 지식 그래프 표현에 적합한 가중 트리플 그래프를 생성하고, 이를 활용한 Weighted Triple2Vec 임베딩 모델 구축 과정을 소개한다. 본 논문에서는 3.2절에서 추출한 특징행렬 $F_o$을 사용하여 가중 트리플 그래프를 생성한다. $F_o$의 행벡터는 각 트리플 노드에 대해 연쇄적인 이웃 관계 정보를 집약한 특징 벡터를 의미한다. 트리플 노드 간 가중치는 트리플 노드에 해당하는 $F_o$의 행벡터 간 코사인 유사도이다.

생성한 가중 트리플 그래프는 Triple2Vec 임베딩 기법과 동일하게 학습한다. 트리플 노드 간 가중치는 그래프의 경로 집합을 구성하는데 우선순위 역할을 하며, 생성한 경로 집합은 Skip-gram 모델로 학습한다. 트리플 그래프로부터 추출한 특징의 의미가 훼손되는 것을 억제하기 위해, 제안 기법은 단일 은닉층을 가지는 Skip-gram 기반의 임베딩 알고리즘을 사용하였다. 최종적으로 구축되는 Weighted Triple2Vec 임베딩 모델은 지식 그래프의 연쇄적인 연결 구조와 노드와 간선이 갖는 의미 정보를 모두 내포한다.

## 4. 실험 방법 및 결과

### 4.1 실험 데이터

실험에 사용하는 트리플 그래프는 3개의 실세계 데이터셋을 대상으로 한다.

- DBLP: DBLP는 컴퓨터 과학 분야의 서지 정보를 나타낸다. 특정한 논문 객체는 4가지(저자, 키워드, 출간년도, 저널 이름) 관계로 연결된 속성 객체를 가진다. 추출한 3,565개의 논문 객체는 학회에 따라 5개의 카테고리로 분류된다.
- DBpedia: DBpedia는 위키피디아 사이트에 등록된 사전 객체 정보를 나타낸다. 사전 객체는 개별적인 특성에 따라 다양한(장르, 프로듀서, 언어 등) 속성 객체를 가진다. 추출한 4,329개의 사전 객체는 10개의 카테고리로 분류된다.
- IMDB: IMDB는 영화에 대한 정보를 나타낸다. 본 논문에서 사용한 1,000개의 영화 객체는 최대 36가지(감독, 배우, 작가, 평점 등)의 관계로 이루어진 속성 객체를 가지며, 15개의 영화 장르로 구분된다.

<Table 1>은 본 실험에 사용된 3개의 데이터셋에 포함된 객체들의 카테고리 정보를 보여준다. <Table 2>는 그 데이터셋들로부터 생성된 트리플 그래프에 대한 구조 정보를 요약한 것이다. 트리플 그래프는 트리플 형식의 노드를 가지며, 트리플 노드는 특정 객체와 속성 객체, 그리고 이들을 연결하는 간선으로 구성된다. 본 논문에서 사용한 컴퓨터 프로세서의 사양은 Intel Core i7-8700 CPU이며 실험 환경을 고려하여 트리플 그래프의 노드 개수는 최대 30,000개에서 40,000개 사이로 제한하였다. 데이터셋에서 추출한 트리플 개수가 50,000개를 넘는 IMDB의 경우에는 객체에 대한 전체 트리플 중 30,000개를 무작위로 선택하여 트리플 그래프를 생성하였으며 이때, 각 객체는 30개 내외의 트리플 노드를 가지도록 제어하였다.

〈Table 1〉 Entity Categories of Datasets

| | DBLP category | DBpedia category | IMDB category |
|---|---|---|---|
| 1 | Artificial Intelligence | Band | Action |
| 2 | Information Systems | Book | Adventure |
| 3 | Distributed Computing | Chemical Compound | Comedy |
| 4 | Computer Software | Flowering Plant | Crime |
| 5 | Computation Theory | Historic Place | Drama |
| 6 | - | OfficeHolder | Family |
| 7 | - | School | Fantasy |
| 8 | - | SoccerClub | History |
| 9 | - | TelevisionShow | Horror |
| 10 | - | VideoGame | Musical |
| 11 | - | - | Mystery |
| 12 | - | - | Romance |
| 13 | - | - | SciFi |
| 14 | - | - | Thriller |
| 15 | - | - | Western |

〈Table 2〉 Summary of Triple Graphs

| Dataset | DBLP | DBpedia | IMDB |
|---|---|---|---|
| Entity | 3,565 | 4,329 | 1000 |
| Edge Type | 4 | 190 | 36 |
| Triple | 37,141 | 38,988 | 55,145 |
| Triple Graph Node | 37,141 | 38,988 | 30,000 |
| Triple Graph Edge | 2,251,865 | 2,302,593 | 1,395,066 |
| Class | 5 | 10 | 15 |

## 4.2 파라미터 설정

- Triple2Vec: [2]의 Triple2Vec 모델을 구축 환경과 동일하게, 본 연구의 트리플 그래프의 각 노드에 대한 경로 개수(Number of Walks)는 10, 경로 길이(Walk Length)는 100, Skip-gram 모델이 학습하는 노드에 대한 윈도우 크기(Window Size)는 10으로 설정하였다. 최종적으로 생성되는 임베딩 모델은 128차원이다.
- Weighted Triple2Vec: Weighted Triple2Vec은 2단계의 학습 과정을 거친다. 첫 단계는 트리플 그래프의 특징 추출 과정이다. 본 논문에서는 StellarGraph 라이브러리를 활용하여 4개의 합성곱 층을 만들고 Dropout 비율을 0.5로 설정하여 GCN 모델을 구축하였다. 우리는 4.1.1절에서 언급한 CPU만을 사용하여 Full-Batch Gradient Descent를 사용하여 GCN 모델을 생성하였으며, 이때 사용하는 트리플 노드는 트리플 그래프의 노드를 7.5:2.5의 비율로 각각 훈련 데이터셋(Training Dataset)과 검증 데이터셋(Validation Dataset)으로 나누었다. 다음 단계는 트리플 그래프의 임베딩 모델을 구축하는 과정으로서, 이와 관련한 파라미터는 Triple2Vec 모델과 동일하게 설정하였다.

## 4.3 실험 방법

본 논문은 Triple2Vec 모델과 Weighted Triple2Vec 모델의 성능을 평가하기 위해 임베딩 모델을 활용한 트리플 노드의 카테고리 분류 실험을 진행하였다. 이때, 분류 실험에 사용하는 트리플 노드의 카테고리는 트리플 노드를 구성하는 객체 노드의 카테고리와 동일하다. 분류 모델[7]의 학습을 위한 트리플 노드는 트리플 그래프에 사용된 노드를 대상으로 하며, 8:2의 비율로 훈련 데이터셋(Training Dataset)과 테스트 데이터셋(Testing Dataset)으로 나누어

사용하였다. 본 논문은 평가에 사용되는 데이터 편중을 방지하기 위해 5겹 교차 검증(Cross Validation)을 적용하였다[10]. 분류 모델에 사용한 머신러닝 알고리즘은 scikit-learn에서 제공하는 로지스틱 회귀(Logistic Regression)[16], 서포트 벡터 머신(Support Vector Machine)[6], 의사 결정 나무(Decision Tree)[19], k-근접 알고리즘(k-Nearest Neighbor)[20], 나이브 베이즈(Naive Bayes)[13], 랜덤 포레스트(Random Forest)[11], AdaBoost[5]를 포함한다. 분류 모델의 성능 평가 척도는 정확도(Accuracy)와 F1-스코어(F1-score)이며, 5회 반복수행하여 이들에 대한 표준오차를 구하였다.

### 4.4 실험 결과

<Table 3>은 Triple2Vec의 임베딩 모델과 Weighted Triple2Vec의 임베딩 모델을 활용한 7가지 분류 모델의 정확도 및 F1-스코어를 비교한 표이다. 분류 모델의 정확도를 비교해보면, Triple2Vec 임베딩 모델과 Weighted Triple2 Vec 모두 서포트 벡터 머신 분류 모델에서 가장 높은 성능을 보이는 것을 확인할 수 있다. 이때, DBLP 데이터셋에 대한 SVM 분류 모델의 경우는 Weighted Triple2Vec 임베딩 모델을 사용했을 때, Triple2Vec 임베딩 모델을 사용했을 때보다 정확도가 56% 향상된 것을 볼 수 있다. 이는 Triple2Vec 임베딩 모델이 학습을 수행할 때, 트리플 노드에 사용된 술어의 종류가 적으면 트리플 노드 간 관계 학습이 어렵고 임베딩 모델의 성능 저하를 초래함을 의미한다. 반면, Weighted Triple2Vec은 각 트리플 노드의 연쇄적인 관계 정보를 내포한 인접성 벡터를 학습에 사용함으로써 데이터셋의 특성에 영향을 받지 않고 트리플 임베딩 모델의 성능을 크게 개선한다.

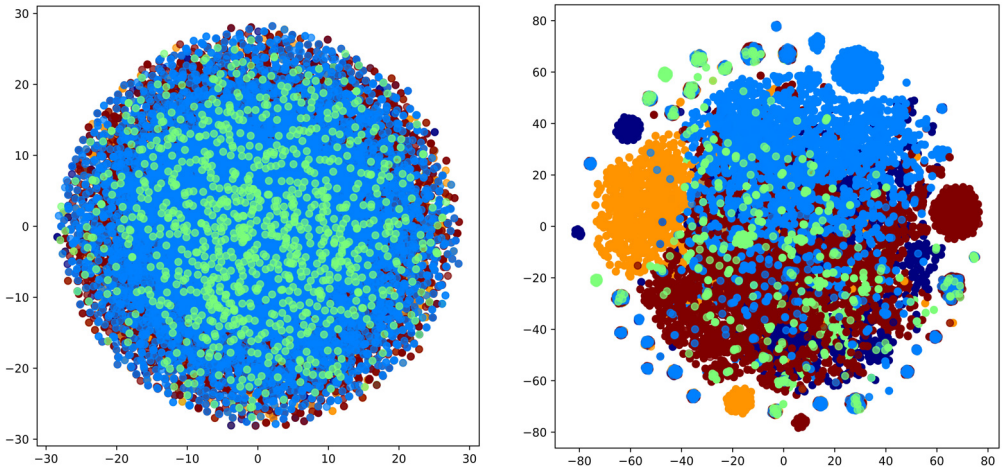
〈Table 3〉 Performance of Classification Models Using Triple Graph Embedding

| Classification Algorithm | Embedding Model | Metric | Dataset1_ DBLP | Dataset2_ DBpedia | Dataset3_ IMDB |
|---|---|---|---|---|---|
| Logistic Regression | Triple2Vec | Accuracy | 0.38±0.00 | 0.51±0.00 | 0.67±0.00 |
| | | F1-score | 0.55±0.00 | 0.55±0.00 | 0.78±0.01 |
| | Weighted Triple2Vec | Accuracy | 0.72±0.00 | 0.92±0.00 | 0.72±0.00 |
| | | F1-score | 0.78±0.00 | 0.92±0.00 | 0.78±0.00 |
| SVM (RBF kernel) | Triple2Vec | Accuracy | 0.38±0.00 | 0.70±0.00 | 0.85±0.04 |
| | | F1-score | 0.55±0.00 | 0.72±0.00 | 0.86±0.03 |
| | **Weighted Triple2Vec** | **Accuracy** | **0.94±0.00** | **0.98±0.00** | **0.94±0.00** |
| | | **F1-score** | **0.94±0.00** | **0.98±0.00** | **0.94±0.00** |
| Decision Tree | Triple2Vec | Accuracy | 0.37±0.00 | 0.41±0.00 | 0.67±0.00 |
| | | F1-score | 0.53±0.00 | 0.45±0.00 | 0.79±0.00 |
| | Weighted Triple2Vec | Accuracy | 0.67±0.00 | 0.49±0.00 | 0.67±0.00 |
| | | F1-score | 0.78±0.00 | 0.55±0.02 | 0.78±0.00 |

〈Table 3〉 Performance of Classification Models Using Triple Graph Embedding (Continued)

| Classification Algorithm | Embedding Model | Metric | Dataset1_ DBLP | Dataset2_ DBpedia | Dataset3_ IMDB |
|---|---|---|---|---|---|
| KNN | Triple2Vec | Accuracy | 0.30±0.00 | 0.61±0.00 | 0.76±0.04 |
| | | F1-score | 0.31±0.00 | 0.62±0.00 | 0.80±0.02 |
| | Weighted Triple2Vec | Accuracy | 0.89±0.00 | 0.95±0.00 | 0.89±0.00 |
| | | F1-score | 0.90±0.00 | 0.95±0.00 | 0.90±0.00 |
| Naïve Bayes | Triple2Vec | Accuracy | 0.37±0.00 | 0.43±0.00 | 0.35±0.09 |
| | | F1-score | 0.48±0.00 | 0.45±0.00 | 0.29±0.11 |
| | Weighted Triple2Vec | Accuracy | 0.72±0.00 | 0.88±0.01 | 0.72±0.00 |
| | | F1-score | 0.76±0.01 | 0.88±0.01 | 0.76±0.01 |
| Random Forest | Triple2Vec | Accuracy | 0.36±0.00 | 0.69±0.00 | 0.76±0.01 |
| | | F1-score | 0.45±0.00 | 0.71±0.00 | 0.81±0.00 |
| | Weighted Triple2Vec | Accuracy | 0.86±0.00 | 0.97±0.00 | 0.86±0.00 |
| | | F1-score | 0.87±0.00 | 0.97±0.00 | 0.87±0.00 |
| AdaBoost | Triple2Vec | Accuracy | 0.37±0.00 | 0.67±0.00 | 0.76±0.01 |
| | | F1-score | 0.52±0.00 | 0.69±0.00 | 0.80±0.01 |
| | Weighted Triple2Vec | Accuracy | 0.85±0.00 | 0.94±0.00 | 0.85±0.00 |
| | | F1-score | 0.86±0.00 | 0.94±0.00 | 0.86±0.00 |

<Figure 4(a)>는 DBLP 트리플 그래프의 Triple2Vec 임베딩 모델을, <Figure 4(b)>는 DBLP 트리플 그래프의 Weighted Triple2Vec 임베딩 모델을 t-SNE 기법을 사용하여 2차원




(a) Triple2Vec (b) Weighted Triple2Vec

〈Figure 4〉 t-SNE based Visualization for Triple2Vec and Weighted Triple2Vec

의 공간으로 시각화 한 것이다. Triple2Vec 임베딩 모델은 서로 다른 카테고리를 가진, 즉 연관이 낮은 트리플 노드가 뒤섞여 있는 형태지만, Weighted Triple2Vec 임베딩 모델은 비교적 같은 카테고리를 가진 유사한 트리플 노드가 모여 있는 것을 확인할 수 있다.

## 5. 결 론

본 논문은 지식 그래프의 효과적인 임베딩을 위해 가중 트리플 그래프의 특징 추출 기법을 제안한다. 제안 기법은 그래프 합성곱 신경망을 기반으로 트리플 그래프의 이웃 노드 정보를 중첩한 인접성 벡터를 추출한다. 해당 특징 정보를 내포한 가중 트리플 그래프는 결과적으로 트리플 그래프의 노드 임베딩 모델을 개선하고, 이는 곧 양질의 지식 그래프 임베딩 모델을 얻을 수 있음을 의미한다. 본 논문은 실험을 통해 제안 기법을 적용한 Weighted Triple2 Vec 임베딩 모델의 2가지 우수성을 입증한다. 첫째, Weighted Triple2 Vec 모델은 트리플 그래프의 인접성 벡터를 활용하여 지식 그래프의 노드 간 연결 구조 정보와 트리플 노드가 서로에게 미치는 구체적인 의미 정보를 내포하고 있어 기존의 Triple2 Vec보다 지식 그래프 표현에 더 적합하다. 둘째, Weighted Triple2Vec 모델은 데이터셋의 특성에 영향을 받지 않고 균일하게 우수한 성능을 유지한다.

향후 연구로서, 우리는 트리플 지식 그래프의 방대한 규모로 인한 비효율성 문제를 해결하고, 개선된 Weighted Triple2Vec 모델을 활용한 추천 알고리즘을 개발할 예정이다.

## References

[1] Dong, X., Chawla, N. V., and Swami, A., “metapath2vec: Scalable Representation Learning for Heterogeneous Networks,” In Proc. of Int. Conference on Information and Knowledge Management, pp. 135-144, 2017.

[2] Fionda, V. and Pirro, G., “Triple2Vec: Learning Triple Embeddings fromKnowledge Graphs,” AAAI, 2020.

[3] Gao, Z., Fu, G., and Ouyang, C., “edge2vec: Representation learning using edge semantics for biomedical knowledge discovery,” BMC Bioinformatics, 2019.

[4] Grover, A. and Leskovec, J., “node2vec: Scalable Feature Learning for Networks,” In Proc. of Int. Conference on Knowledge Discovery and Data Mining, pp. 855-864, 2016.

[5] Hastie, T., Rosset, S., Zhu, J., and Zou, H., “Multi-class AdaBoost,” Statistics and Its Interface, Vol. 2, No. 3, pp. 349-360, 2009.

[6] Hearst, M. A., “Support Vector Machines,” IEEE Intelligent Systems, Vol. 13, pp. 18-28, 1998.

[7] Hwang, S. H. and Kim, D. H., “BERT-based Classification Model for Korean Documents,” The Journal of Society for e-Business Studies, Vol. 25, No. 1, 2020.

[8] Kipf, T. and Welling, M., “Semi-Supervised Classification with Graph Convolutional Networks,” Proceedings of the 5th Inter-

national Conference on Learning Representation, 2017.

[9] Krizhevsky, A., Sutskever, I., and Hinton, G. E., "ImageNet Classification with Deep Convolutional Neural Networks," NIPs, 2012.

[10] Lee, S.-E. and Kim, H.-J., "A New Ensemble Machine Learning Technique with Multiple Stacking," The Jounal of Society for e-Business Studies, Vol. 25, No. 3, pp. 1-13, 2020.

[11] Liaw, A. and Wiener, M., "Classification and Regression by randomForest," R News, Vol. 2/3, pp. 18-22, 2002.

[12] Mikolov, T., Sutskever, I., Chen, K., Corrado, G. S., and Dean, J., "Distributed Representations of Words and Phrases and their Compositionality," NIPs, 2013.

[13] Patil, T. R. and Sherekar, S. S., "Performance analysis of Naive Bayes and J48 classification algorithm for data classification," International Journal of Computer Science and Applications, Vol. 6, No. 2, pp. 256-261, 2013.

[14] Perozzi, B., Al-Rfou, R., and Skiena, S., "Deepwalk: OnLine Learning of Social rRpresentations," In Proc. of KDD, pp. 701-710, 2014.

[15] Pirro, G., "Building relatedness explanations from knowledge graphs," ICAR-CNR, 2019.

[16] Pregibon, D., "Logistic Regression Diagnostics," The Annals of Statistics, Vol. 9, No. 4, pp. 705-724, 1981.

[17] Scarsell, F., Gori, M., Tsoi, A. C., Hagenbuchner, M., and Monfardini, G., "The Graph Neural Network Model," IEEE Transactions on Neural Networks, Vol. 20, No. 1, pp. 61-80, 2009.

[18] Schlichtkrull, M., Kipf, T. N., Bloem, P., van den Berg, R., Titov, I., and Welling, M., "Modeling Relational Data with Graph Convolutional Networks," ESWC, pp. 593-607, 2018.

[19] Song, Y.-Y. and Ying, L. U., "Decision tree methods: applications for classification and prediction," Shanghai Arch Psychiatry, Vol. 27, No. 2, pp. 130-135, 2015.

[20] Zhang, M.-L. and Zhou, Z.-H., "ML-KNN: A lazy learning approach to multi-label learning," Pattern Recognition, Vol. 40, No. 7, pp. 2038-2048, 2007.

## 저 자 소 개

조새롬 (E-mail: csrom0128@gmail.com)
2018년 국가평생교육진흥원 컴퓨터공학과 (공학사)
2019년~현재 서울시립대학교 전자전기컴퓨터공학과 (석사과정)
관심분야 데이터 마이닝, 머신러닝, 빅데이터 분석

김한준 (E-mail: khj@uos.ac.kr)
1994년 서울대학교 계산통계학과 (이학사)
1996년 서울대학교 전산과학과 (이학석사)
2002년 서울대학교 컴퓨터공학부 (공학박사)
2002년~현재 서울시립대학교 전자전기컴퓨터공학부 정교수
관심분야 머신러닝, 빅데이터분석, 텍스트마이닝, 데이터베이스, 정보검색