Graph Attention Network

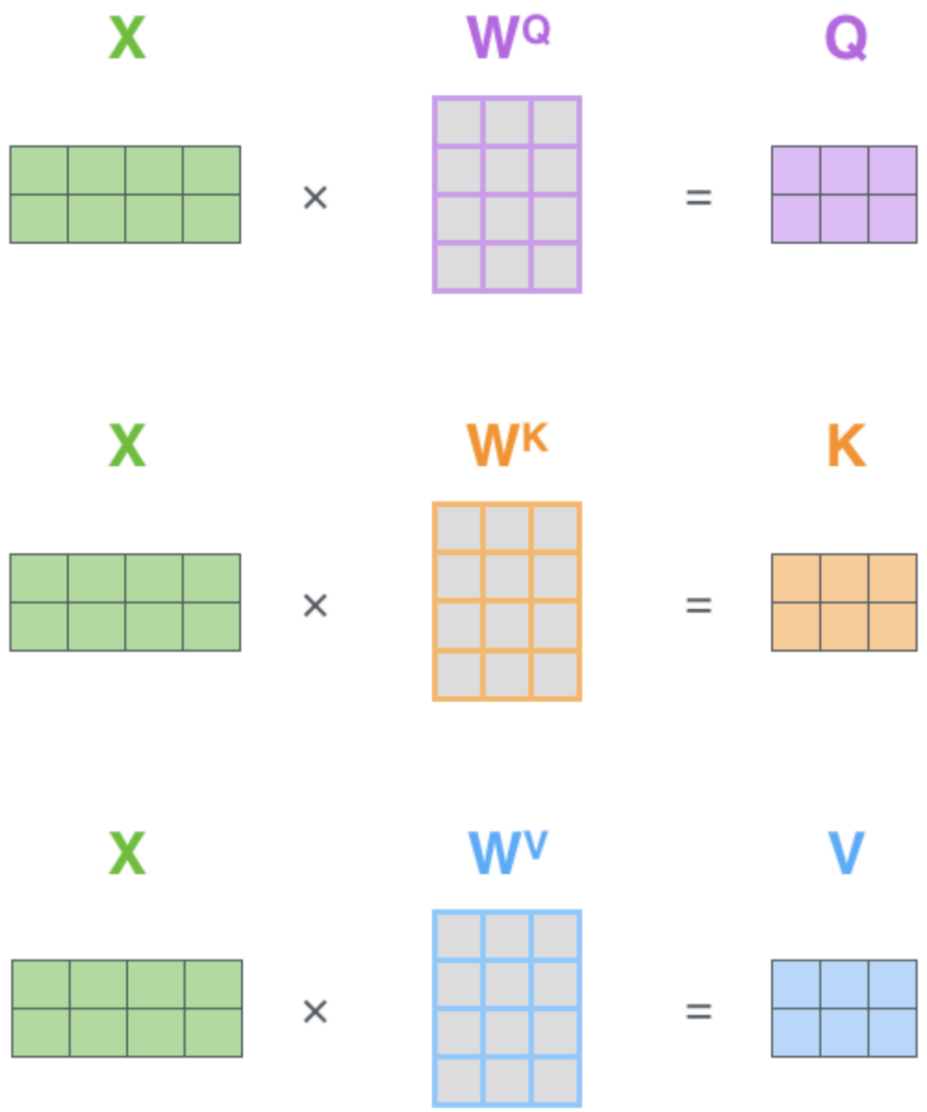
딥러닝 논문읽기 “자연어처리팀”

팀원 :주정헌(발표자), 김은희, 황소현, 신동진, 문경언

2021.02.21

Contents

- Attention-mechanism
- Graph Neural Network
- Graph Attention Network
- Inductive-learning & Transductive-learning
- Experiments
- Q & A



Attention-mechanism

Dictionary 자료형을 이용한 알고리즘
(유사도, 검색알고리즘)

1. Query(질의) : vector, matrix, tensor
학습에 사용될 입력 데이터

2. Key(키) : vector, matrix, tensor
Query와 비교할 대상 데이터

3. Value(값) : vector, matrix, tensor
Key가 갖고 있는 숨은 값들

Attention-mechanism

Query * Key

Similar function * Value

Query

Apple

Key	Similar function	Value	Similarity
Amazon	0.31	AWS, Amazon prime	0.35
Google	0.28	Youtube, Android	0.20
Facebook	0.46	Instagram, Whatsapp	0.53
Tesla	0.19	Model S, Model X	0.11
Nvidia	0.04	Geforce, ARM	0.15

Attention score == Similar function

$$A(q, K, V) = \sum_i \frac{e^{q \cdot k_i}}{\sum_j e^{q \cdot k_j}} v_i$$

Output = 0.35 + 0.20 + 0.53 + 0.11 + 0.15

Becomes: $A(Q, K, V) = softmax(QK^T)V$

$[|Q| \times d_k] \times [d_k \times |K|] \times [|K| \times d_v]$

softmax row-wise

$= [|Q| \times d_v]$

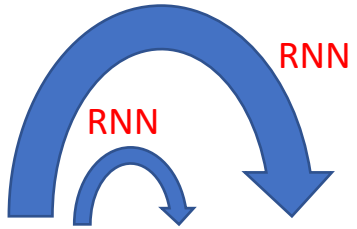
Attention-mechanism

- Document classification based RNN

Attention : Query와 Key의 유사도를 구한 후 value와 weighted sum

KEY, VALUE : RNN의 hidden-state

QUERY : 학습될 문맥벡터(context vector)



Similarity function???

Query : c

[0.12, 0.35, 0.041, 0.05, 0.711]

input	key	value	Attention score	document	classes
x1	h1	h1	a1 = h1 * c	Feature = a1 + a2 + a3 + a4 + a5	class1
x2	h2	h2	a2 = h2 * c		class2
x3	h3	h3	a3 = h3 * c		class3
x4	h4	h4	a4 = h4 * c		class4
x5	h5	h5	a5 = h5 * c		class5

Feature 는 문서의 특성을 갖고 있다고 여긴다 !!

Attention-mechanism

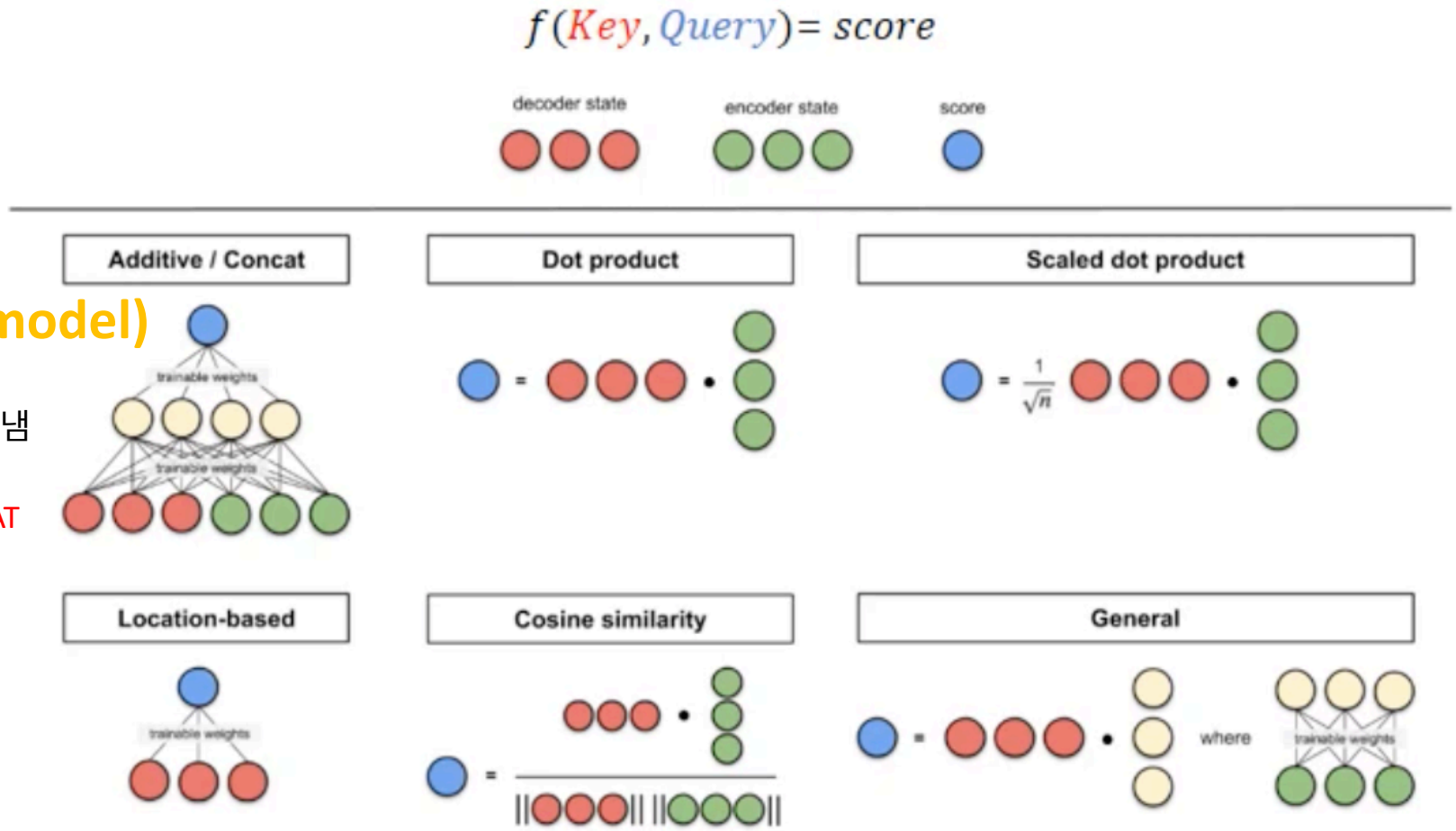
Similarity function (Alignment model)

Alignment는 벡터간의 유사도를 구하는 방법을 나타냄

Additive / concat = Bahdanua attention(2014), GAT

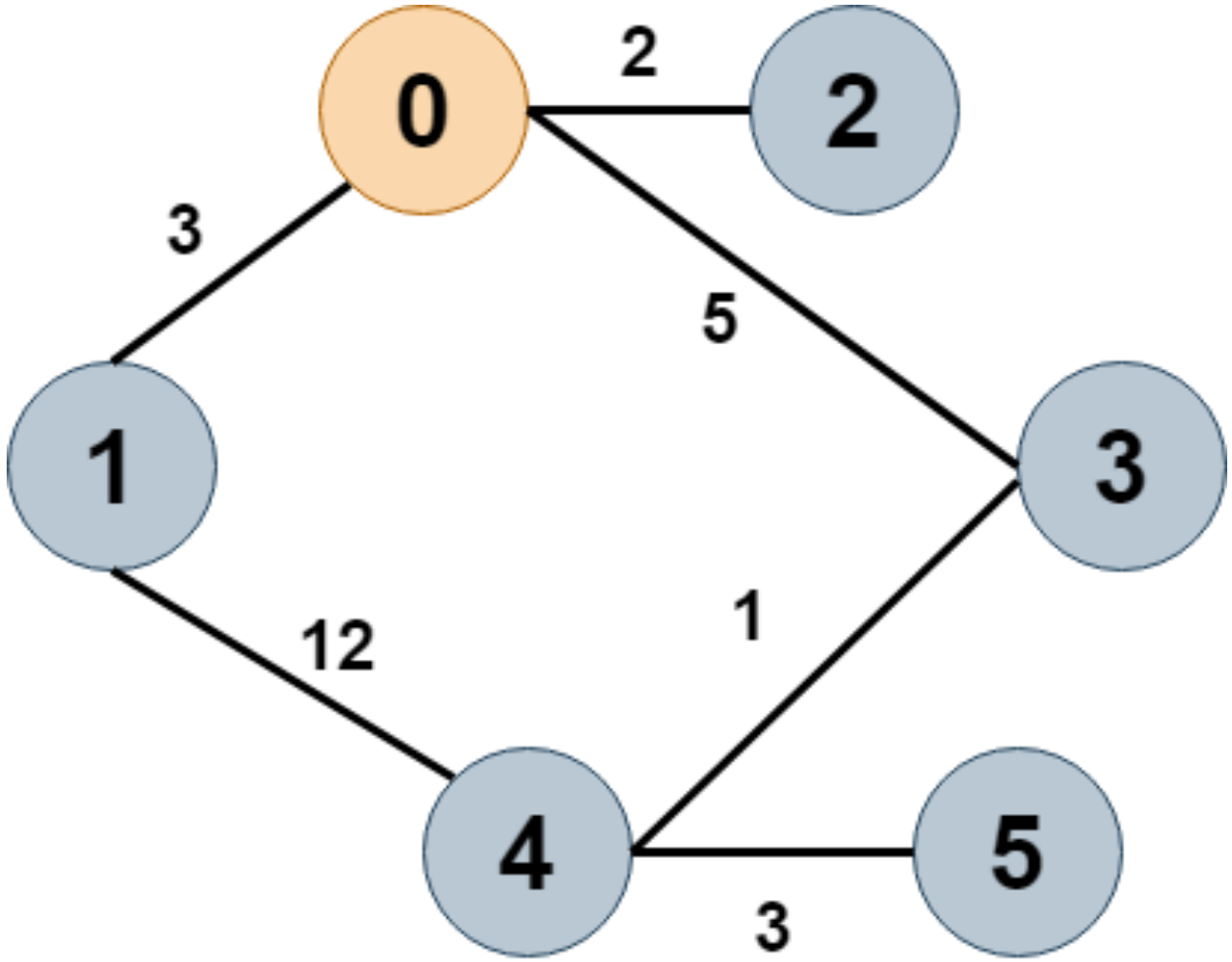
Scaled-dot product = Transformer(2017)

Others = Luong(2015) et al



출처 : https://www.youtube.com/watch?v=NSjpECvEf0Y (강현규, 김성범)

Graph Neural Network

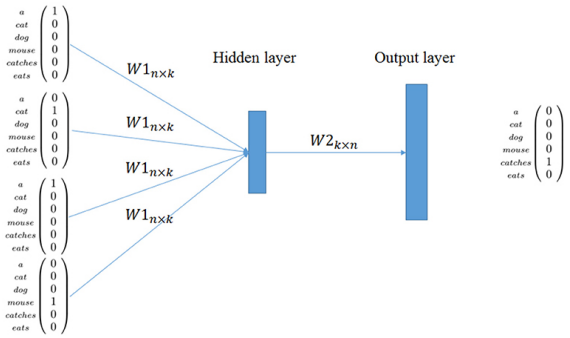


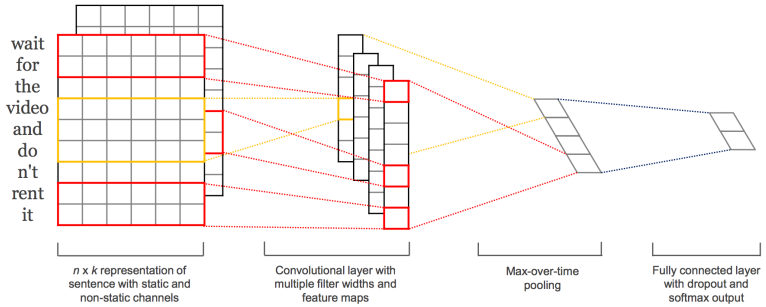
- Node(vertex) : 데이터 그 자체의 특성값
- Edge : 데이터들 사이의 관계를 나타내는 값

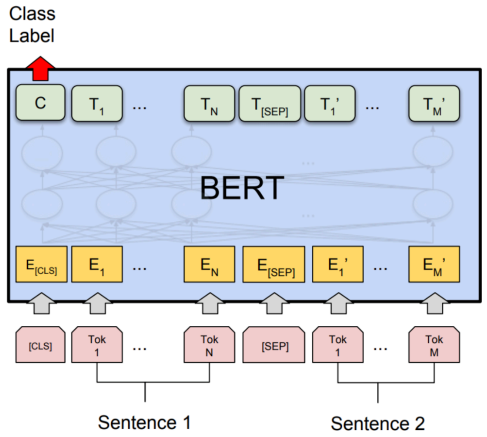
Graph Neural Network (부록)

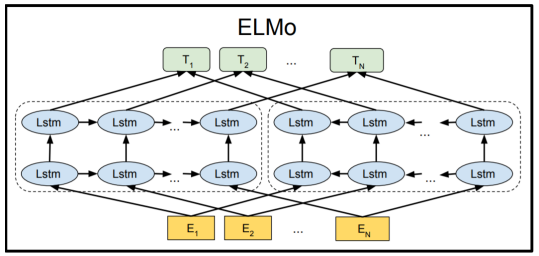
- Language modeling

NLP 분야에서는 문장 혹은 단어들 간의 시퀀스(관계) 를 수치화하여 다차원으로 표현될 수 있는 값을 Euclidean space에 표현할 수 있게 한다. 이를 언어모델링이라 표현하고 이렇게 나온 값들을 Representation 혹은 Embedding이라 부른다.



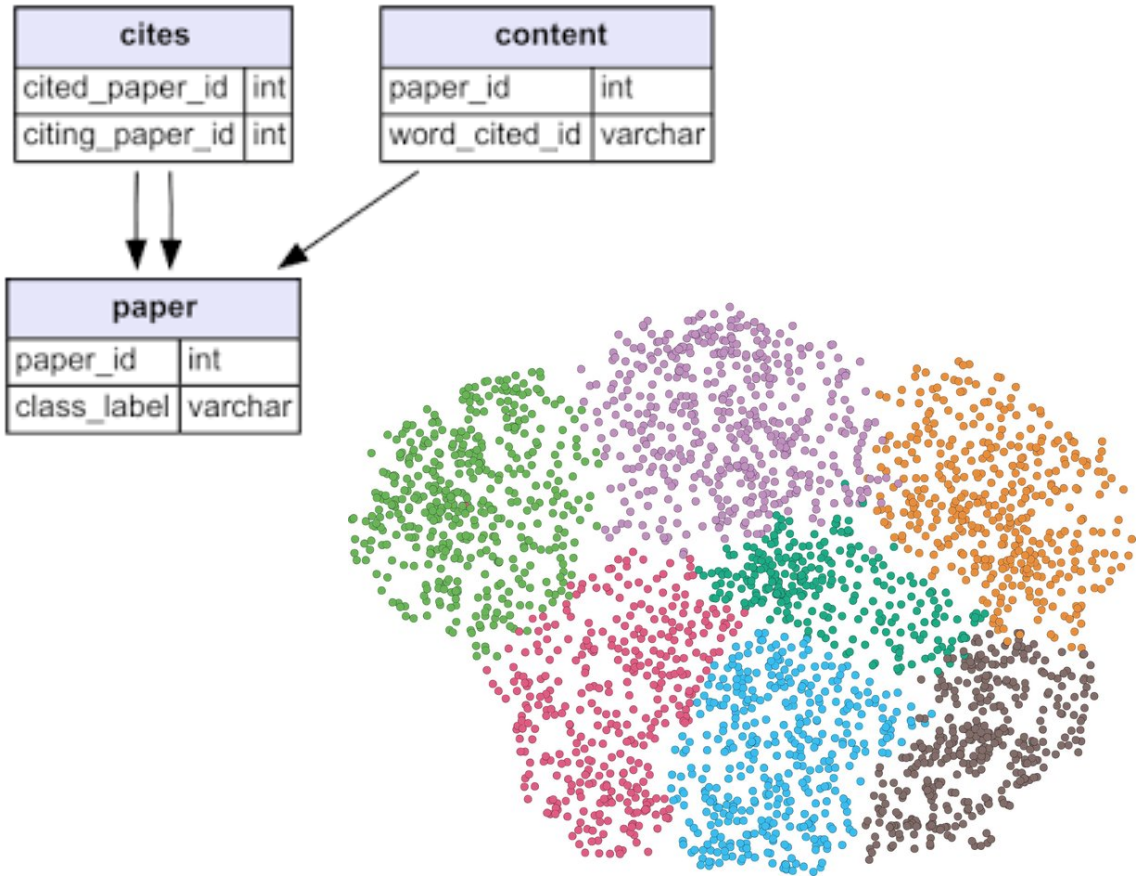




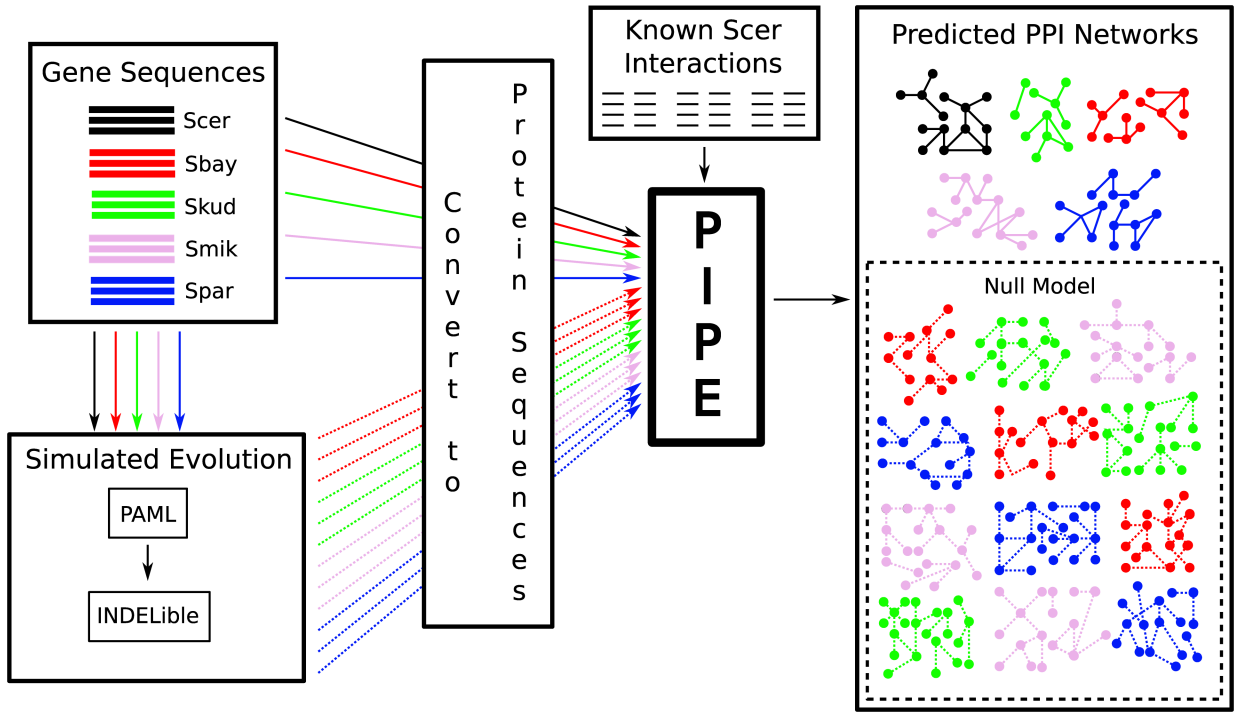


Graph Neural Network

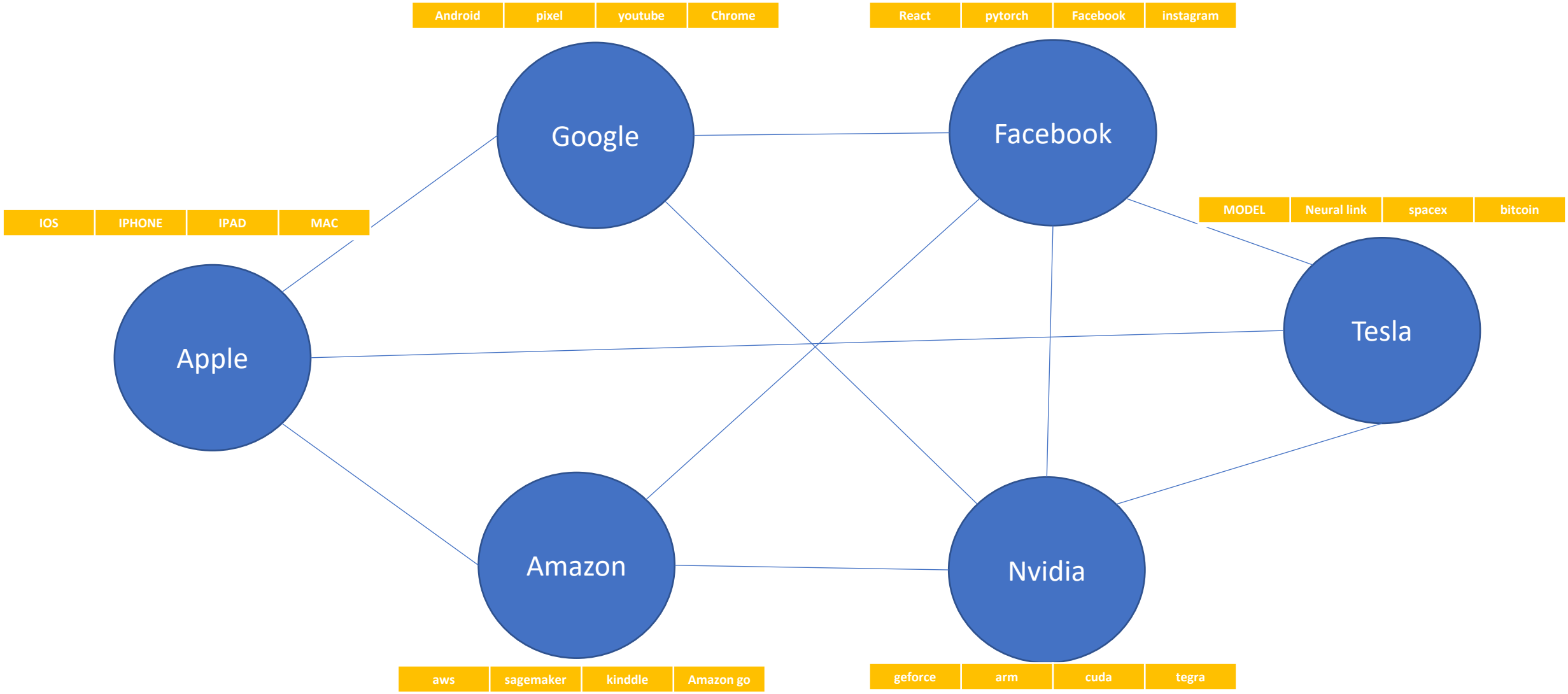
CORA



PPI(protein-protein interaction)



Graph Neural Network

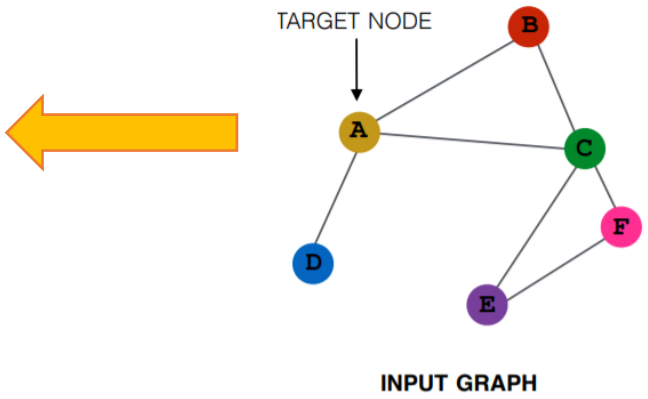


Graph Neural Network

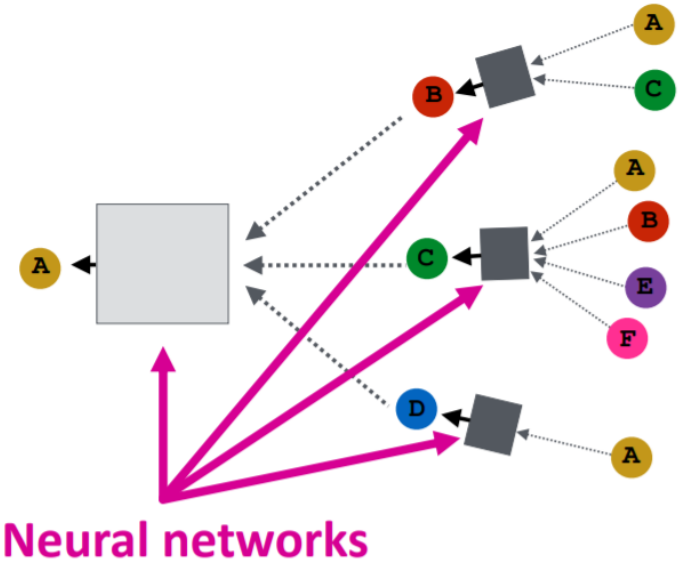
Adjacency Matrix

0	1	0	1	0	1
1	0	1	0	1	0
0	1	0	1	1	1
1	0	1	0	1	0
0	1	1	1	0	1
1	0	1	0	1	0

Graph data

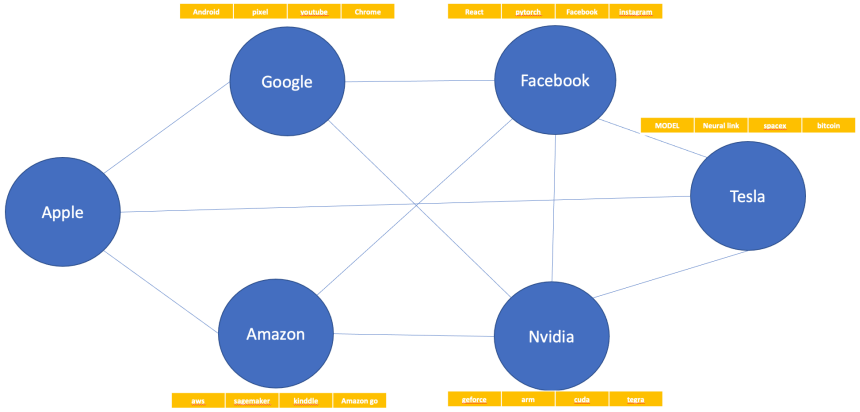


GNN Layer



1:apple, 2:google, 3:facebook 4: tesla, 5:nvidia, 6:amazon

Graph Neural Network



- Aggregation

a1 = aggregate(Android | pixel | youtube | Chrome , aws | sagemaker | kinddle | Amazon go)

인접 노드의 임베딩 벡터를 요약한다.

$$a_v^{(k-1)} = aggregate^k(\{h_u^{(k-1)}, \forall u \in N(v)\})$$

- Combine

h1 = combine(a1, IOS | IPHONE | IPAD | MAC)

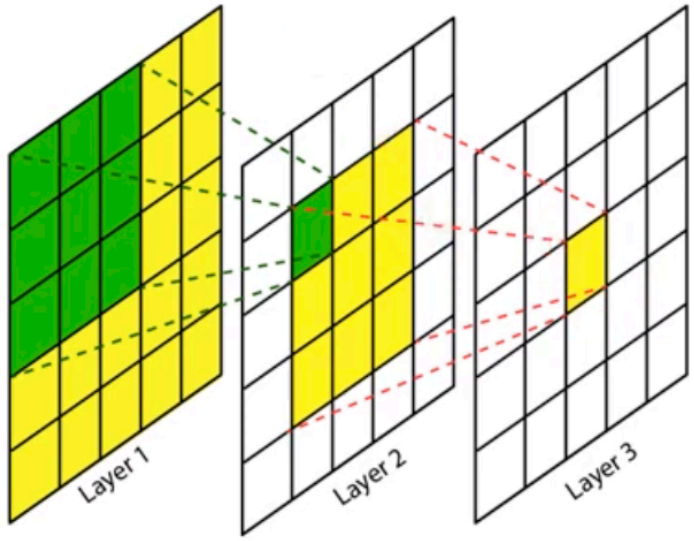
h1 = h1 | h2 | h3 | h4

$$h_1^{(k)} = combine^k(a_1^{(k-1)},$$  $$)$$

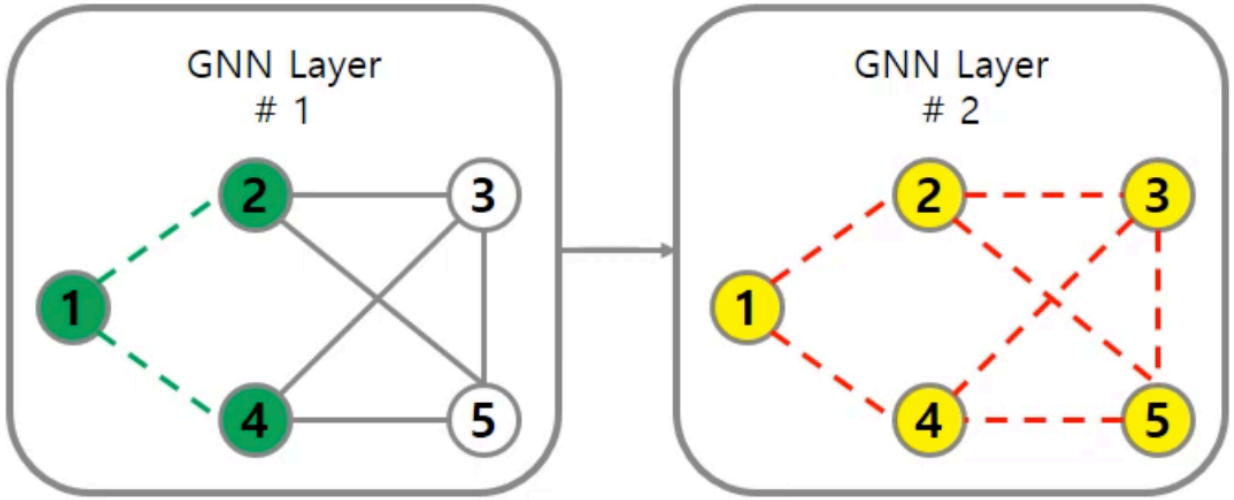
- Readout

h1 | h2 | h3 | h4

Graph Neural Network



Convolutional Neural Networks



Graph Neural Networks

출처 : https://www.youtube.com/watch?v=NSjpECvEf0Y (강현규, 김성범)

Graph Attention Network

1) Nodes 들 간의 linear transform
2) Nodes들 간의 Attention
3) Activation -> Softmax -> Attention score
4) Hidden-states와 Aij를 weighted sum
5) Self-attention을 이용한 multi-head attention
6) 새로운 hidden-state로 target node의 feature 학습

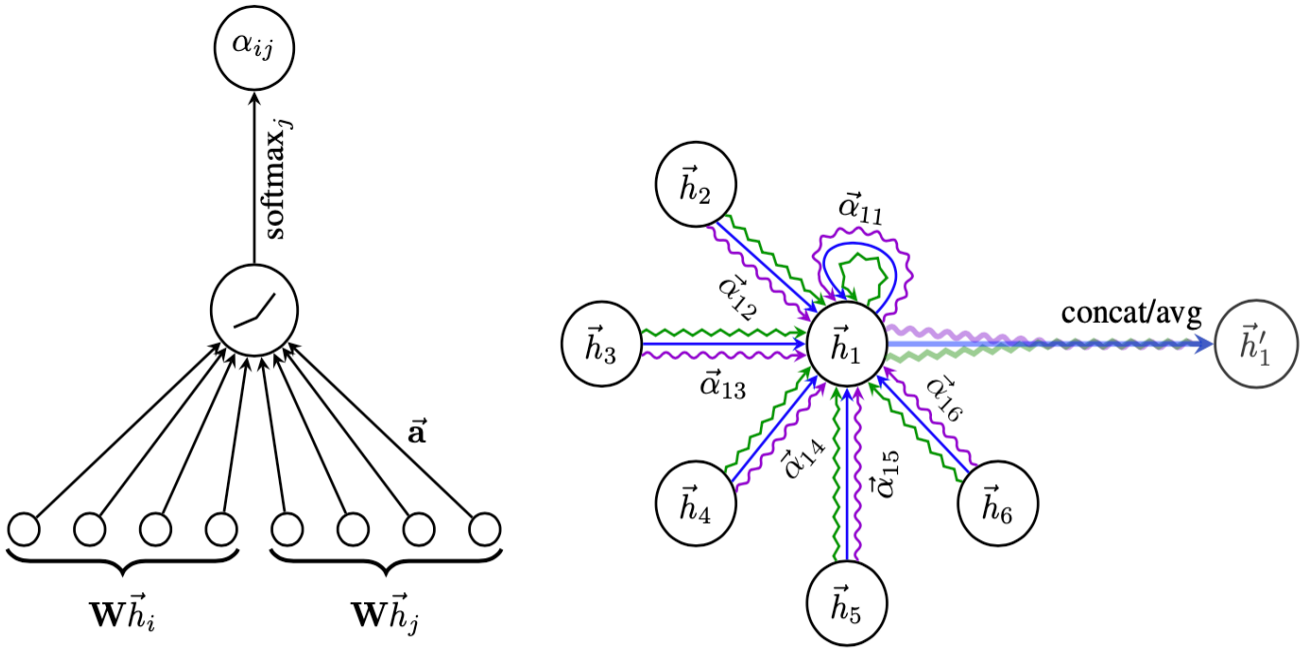
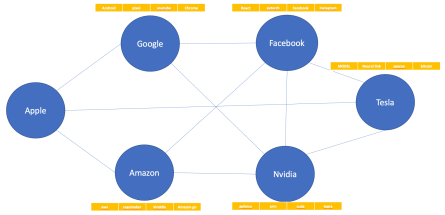


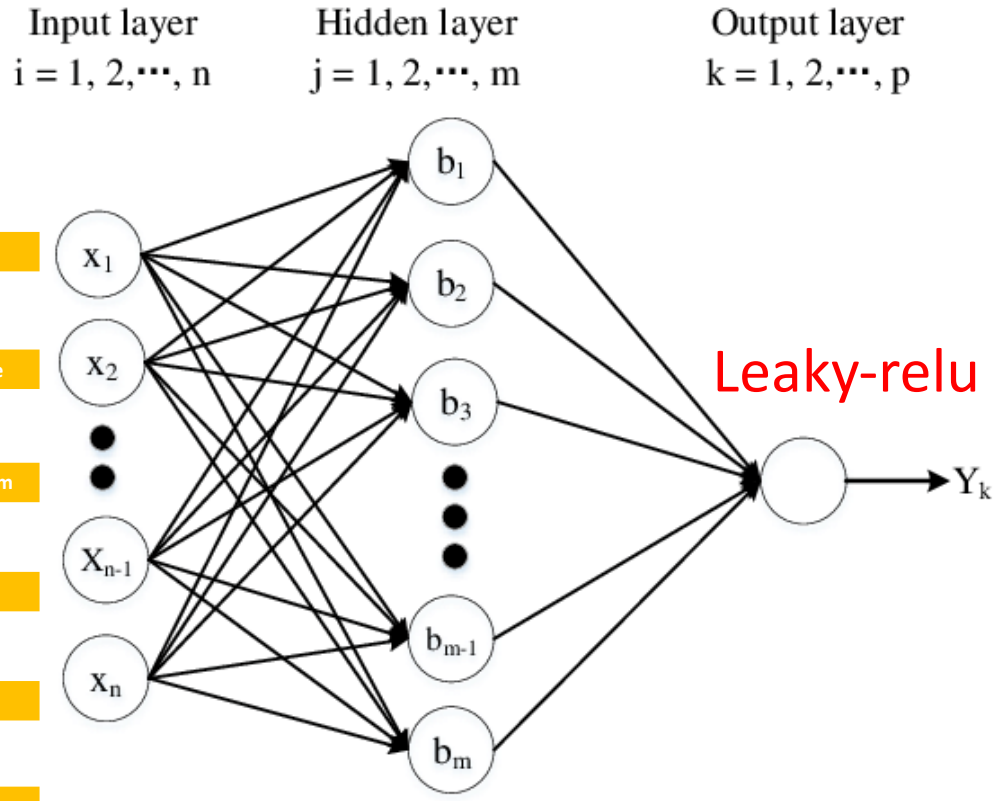
Figure 1: **Left:** The attention mechanism $a(\mathbf{W}\vec{h}_i, \mathbf{W}\vec{h}_j)$ employed by our model, parametrized by a weight vector $\vec{\mathbf{a}} \in \mathbb{R}^{2F'}$, applying a LeakyReLU activation. **Right:** An illustration of multi-head attention (with $K = 3$ heads) by node 1 on its neighborhood. Different arrow styles and colors denote independent attention computations. The aggregated features from each head are concatenated or averaged to obtain $\vec{h}'_1$.

Graph Attention Network



m	a12	m	a14	m	a16
a21	0	a23	0	a25	0
0	a32	0	a34	a35	a36
a41	m	a43	m	a45	m
m	a52	a53	a54	m	a56
a61	m	a63	m	a65	m

IOS	IPHONE	IPAD	MAC
Android	pixel	youtube	Chrome
React	pytorch	Facebook	instagram
MODEL	Neural link	spacex	bitcoin
geforce	arm	cuda	tegra
aws	sagemaker	kinddle	Amazon go



Leaky-relu

1-layered Feed-Forward

Inductive-learning & Transductive-learning

- 귀납법(Induction)

개별적 사실을 근거로 확률적으로 일반화된 추론 방식

1) 참새는 날 수 있다.

2) 독수리는 날 수 있다.

- 모든 새는 날 수 있다.(일반화된 추론)

- 연역법(Deduction)

기존 전제(혹은 가설)이 참이면 결론도 반드시(necessarily) 참인 추론 방식

1) 소크라테스는 사람이다.

2) 모든 사람은 죽는다.

- 소크라테스는 죽는다.(일반화된 사실)

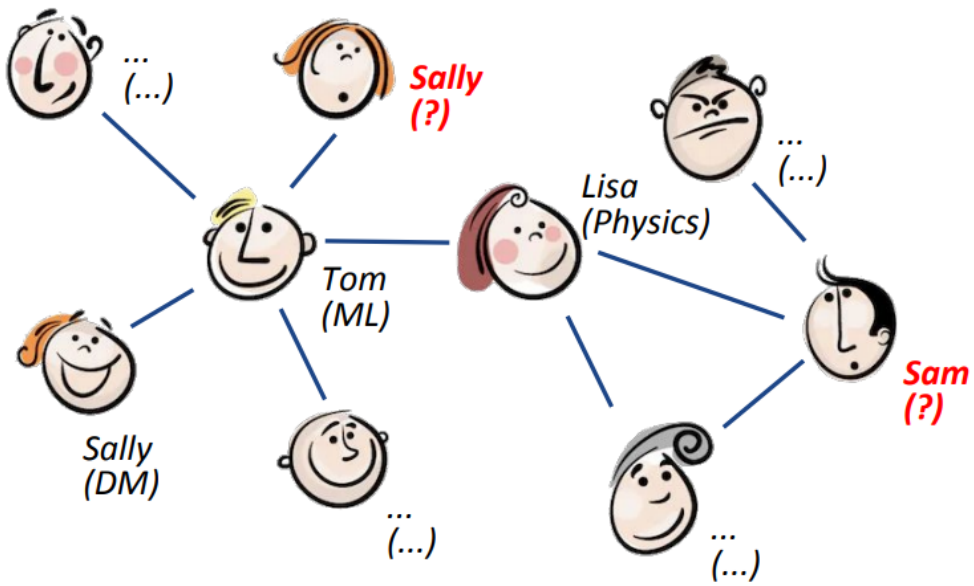
Inductive-learning & Transductive-learning

- **트랜스덕티브(Transductive learning)**

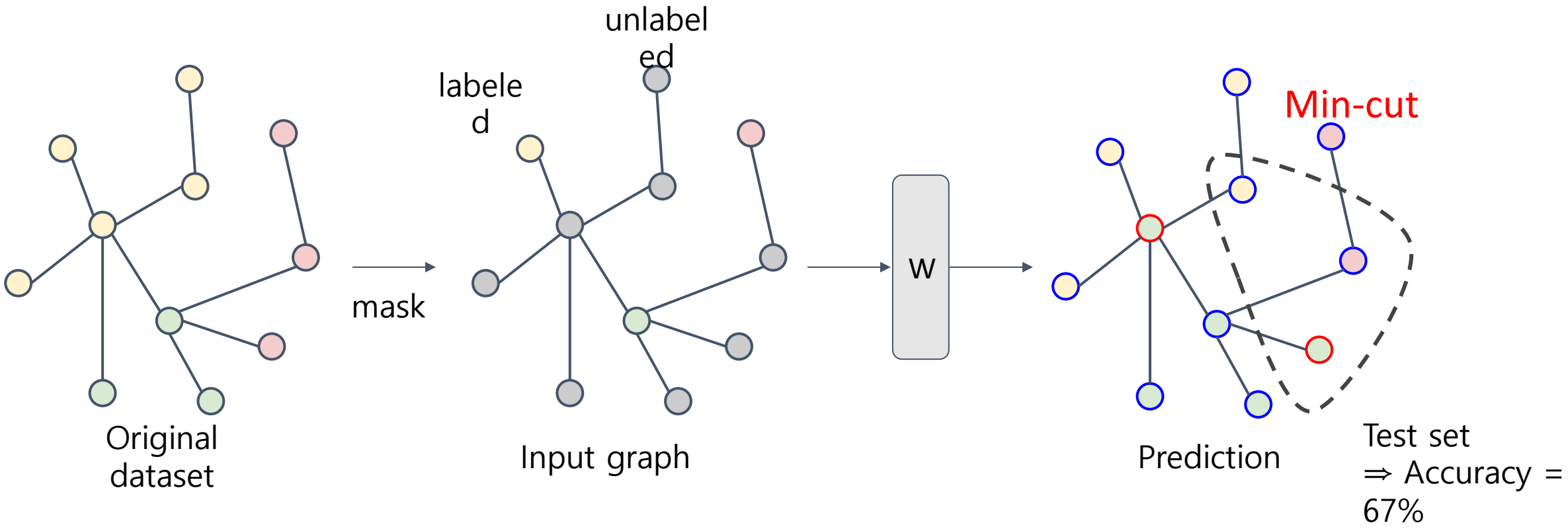
일반화된 추론/사실을 문제로 삼지 않고 중간단계의 문제를 푼다.

1) 연역적으로 “소크라테스는 (반드시) 죽는다” 라는 문제 해결법보다 귀납법이 더 낫다.
2) 특히 문제를 해결하는데 필요한 데이터가 제한적이면 “중간단계의 문제”를 직접 푼다.

- **Node Classification:** (Semi-supervised Learning)
  - Predict research area of unlabeled authors



Inductive-learning & Transductive-learning



Inductive-learning & Transductive-learning

Table 1: Summary of the datasets used in our experiments.

	Cora	**Citeseer**	**Pubmed**	**PPI**
Task	Transductive	Transductive	Transductive	Inductive
# Nodes	2708 (1 graph)	3327 (1 graph)	19717 (1 graph)	56944 (24 graphs)
# Edges	5429	4732	44338	818716
# Features/Node	1433	3703	500	50
# Classes	7	6	3	121 (multilabel)
# Training Nodes	140	120	60	44906 (20 graphs)
# Validation Nodes	500	500	500	6514 (2 graphs)
# Test Nodes	1000	1000	1000	5524 (2 graphs)

Experiments

- **Inductive learning (GraphSAGE) -Hamilton et al., 2017**
  - ✓ **GraphSAGE-GCN**
    - ✓ **Graph Convolution Network**
  - ✓ **GraphSAGE-mean**
    - ✓ **자신 node와 이웃 node의 평균값 사용**
  - ✓ **GraphSAGE-LSTM**
    - ✓ **이웃 node를 램덤으로 섞은 후 LSTM 방식으로 aggregating**
  - ✓ **GraphSAGE-pool**
    - ✓ **Max/mean을 통해서 이웃을 pooling 하는 방법**

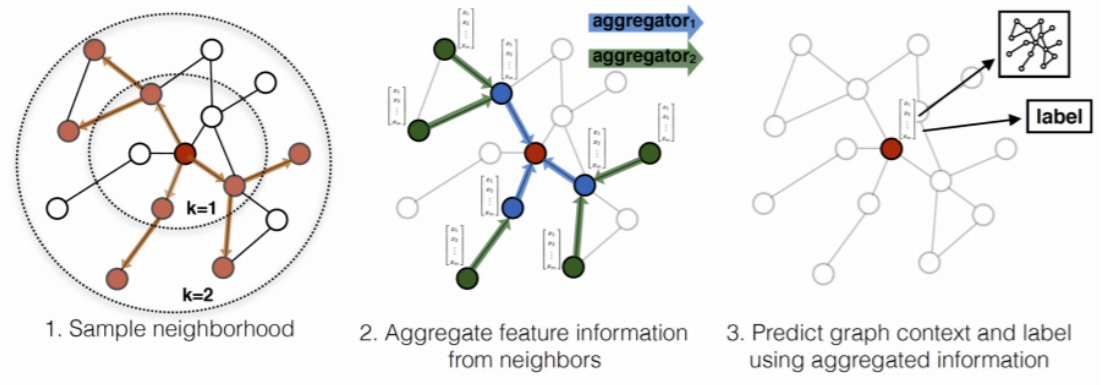


Figure 1: Visual illustration of the GraphSAGE sample and aggregate approach.

Experiments

Table 3: Summary of results in terms of micro-averaged F_1 scores, for the PPI dataset. GraphSAGE* corresponds to the best GraphSAGE result we were able to obtain by just modifying its architecture. Const-GAT corresponds to a model with the same architecture as GAT, but with a constant attention mechanism (assigning same importance to each neighbor; GCN-like inductive operator).

	Inductive
Method	**PPI**
Random	0.396
MLP	0.422
GraphSAGE-GCN (Hamilton et al., 2017)	0.500
GraphSAGE-mean (Hamilton et al., 2017)	0.598
GraphSAGE-LSTM (Hamilton et al., 2017)	0.612
GraphSAGE-pool (Hamilton et al., 2017)	0.600
GraphSAGE*	0.768
Const-GAT (ours)	0.934 ± 0.006
GAT (ours)	$\mathbf{0.973} \pm 0.002$

Experiments

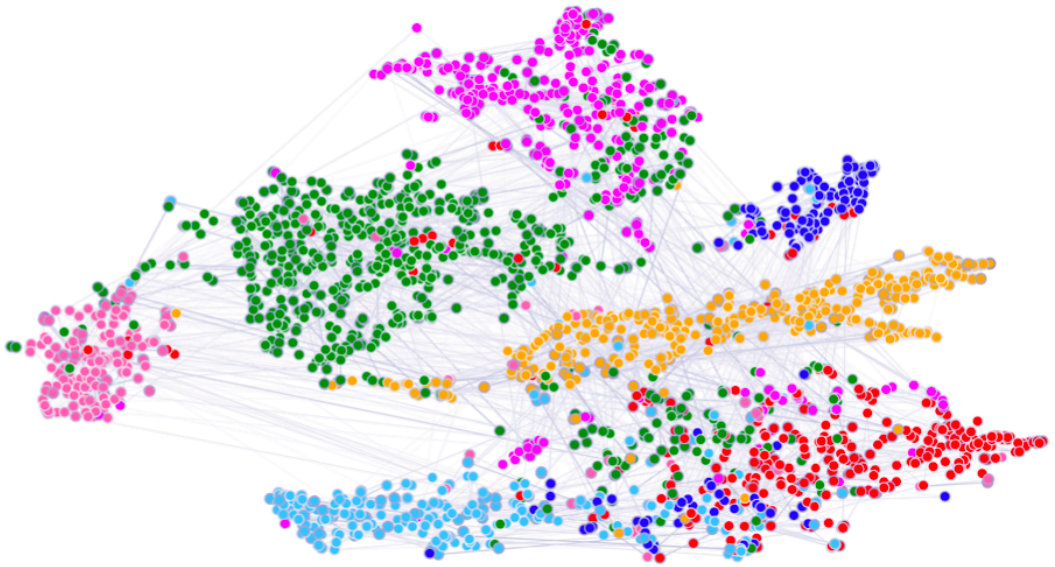


Figure 2: A t-SNE plot of the computed feature representations of a pre-trained GAT model's first hidden layer on the Cora dataset. Node colors denote classes. Edge thickness indicates aggregated normalized attention coefficients between nodes i and j, across all eight attention heads ($\sum_{k=1}^{K} \alpha_{ij}^{k} + \alpha_{ji}^{k}$).

t-SNE

고차원 -> 저차원으로 임베딩 시키는 알고리즘

1. High dimentsion에서 i,j가 이웃 node일 확률

$$p_{ij} = \frac{p_{j|i} + p_{i|j}}{2N}$$

2. low dimentsion에서 i,j가 이웃 node일 확률

$$q_{ij} = \frac{(1 + \|\mathbf{y}_i - \mathbf{y}_j\|^2)^{-1}}{\sum_{k \neq l} (1 + \|\mathbf{y}_k - \mathbf{y}_l\|^2)^{-1}}$$

3. t-SNE 목적함수 (KLD) -> minimize with sgd

$$KL(P||Q) = \sum_{i \neq j} p_{ij} \log \frac{p_{ij}}{q_{ij}}$$

Explainable model

Color = 노드 분류
Edge = 멀티헤드 어텐션 평균값

1개 노드를 updat하는 과정을 알려면 해당 edges의 노드들의 클래스를 관찰할 수 있다.


Q&A

Reference

- GAT : https://arxiv.org/abs/1710.10903

(논문 원문)

- Youtube : https://www.youtube.com/watch?v=NSjpECvEf0Y

(고려대학교 산업공학과 김성범 교수님)

- Books : http://cv.jbnu.ac.kr/index.php?mid=ml

(전북대학교 오일석 교수님)


Thank
you!!