Graph Attention Networks

2020. 09. 11

Data Mining & Quality Analytics Lab.

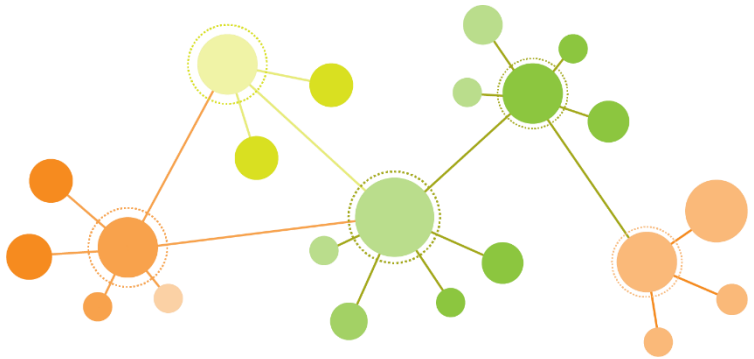
강현규

목차

01 | Introduction

Data Mining Quality Analytics

- 발표자 소개
  - 강현규 (Hyeongyu Kang)
    - ✓ Data Mining & Quality Analytics Lab.
    - ✓ M.S. Student (2019.03 ~ Present)
    - ✓ 2021.02 졸업 예정
  - Research Interest
    - ✓ Machine learning / Deep learning
    - ✓ Natural Language Processing / Graph Neural Networks

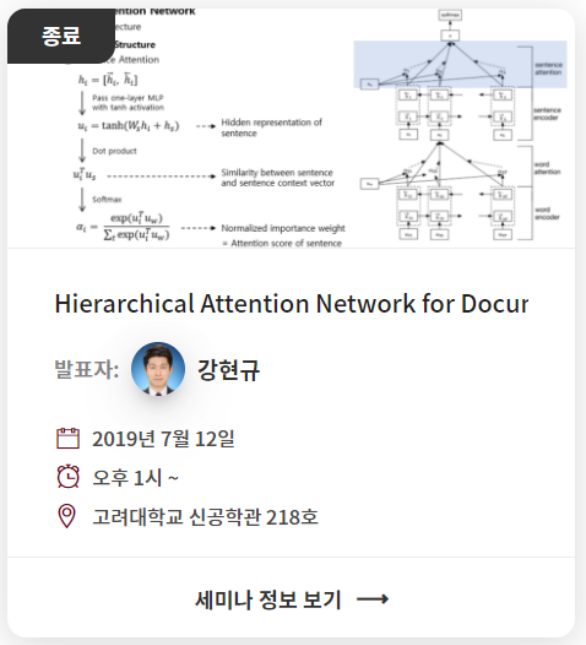


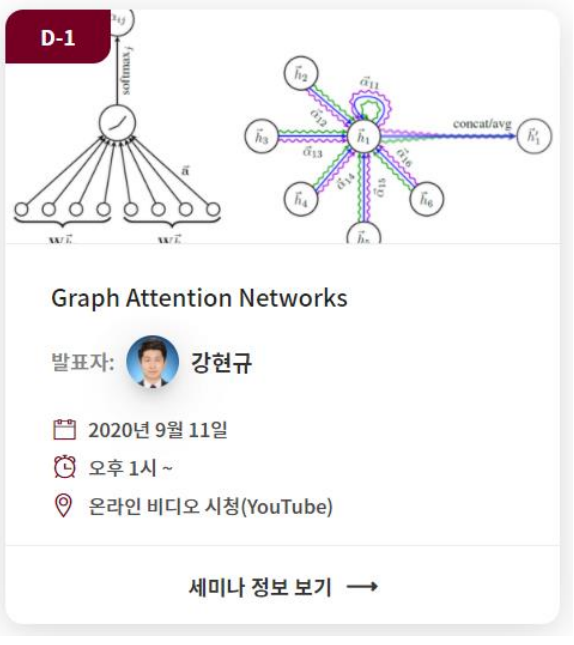
Hyeongyu Kang
(강현규)

M.S. Student
Industrial Management
Engineering,
Korea University

Data Mining Quality Analytics

- Seminar Topic
  - Deep learning
    - ✓ 깊고 복잡한 신경망 구조 -> End-to-End learning
  - Attention
    - ✓ 모델이 집중해서 학습해야 하는 곳 까지도 모델이 학습
    - ✓ Explainability + model performance





Data Mining Quality Analytics

- Seminar Topic
  - ✓ Graph Neural Networks

Graph Neural Networks

Node-Edge로 구성된 그래프 데이터의 구조를 학습하는 딥러닝 모델

Data Mining Quality Analytics

- Seminar Topic
  - Graph Attention Networks
    - ✓ Graph Neural Networks + Attention

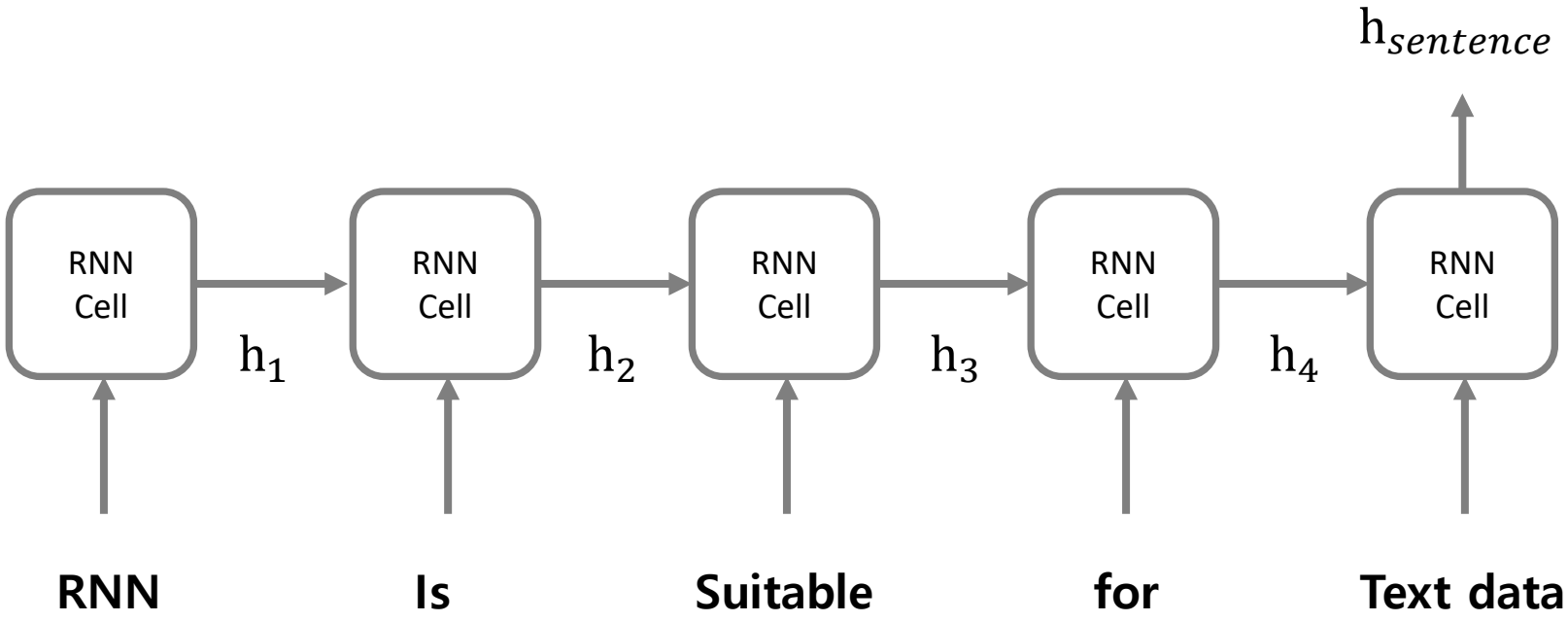
중요 Node에 가중치를 부여하는 어텐션 메커니즘을 사용하여

Graph Attention Networks

Node-Edge로 구성된 그래프 데이터의 **구조를 학습하는 딥러닝 모델**

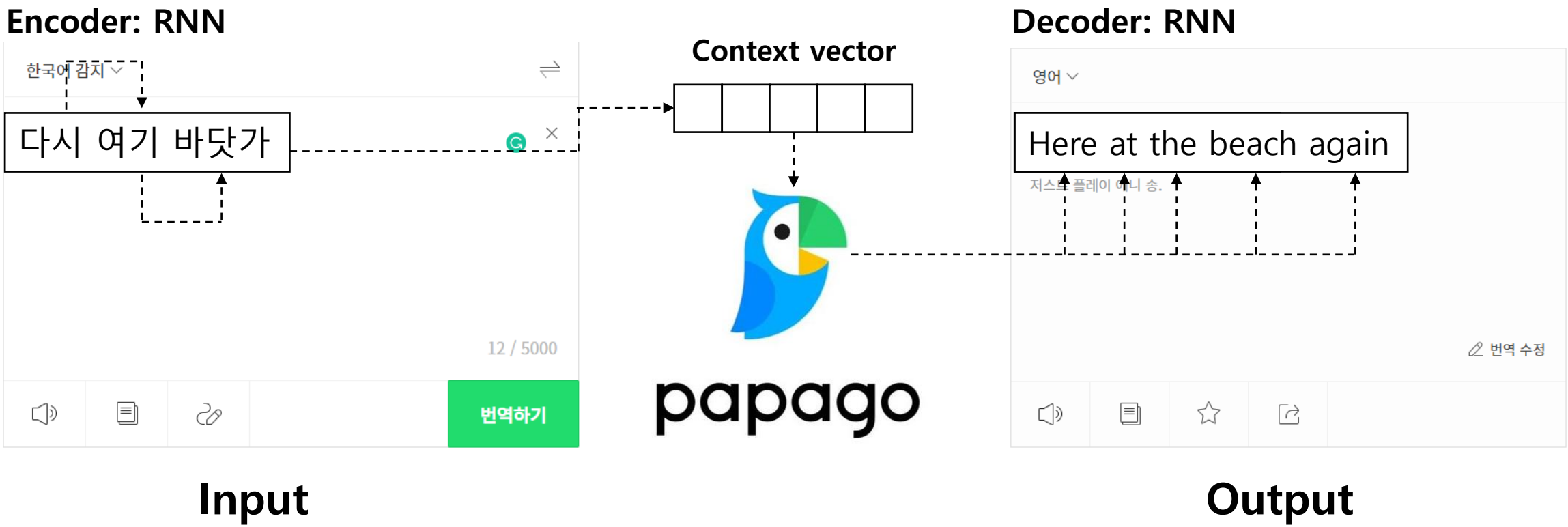
Data Mining Quality Analytics

- Text data & Recurrent Neural Network (RNN)
  - RNN은 sequential data의 학습에 적합한 신경망 모델
  - Text data는 co-occurence와 sequential pattern을 고려한 분석이 필요



Data Mining Quality Analytics

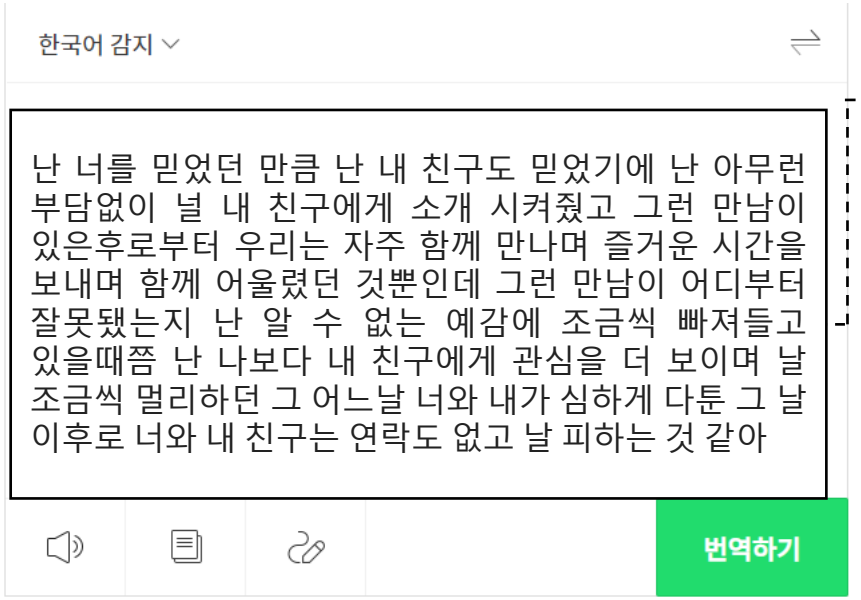
- Seq2seq
  - RNN encoder + RNN decoder
  - Machine translation



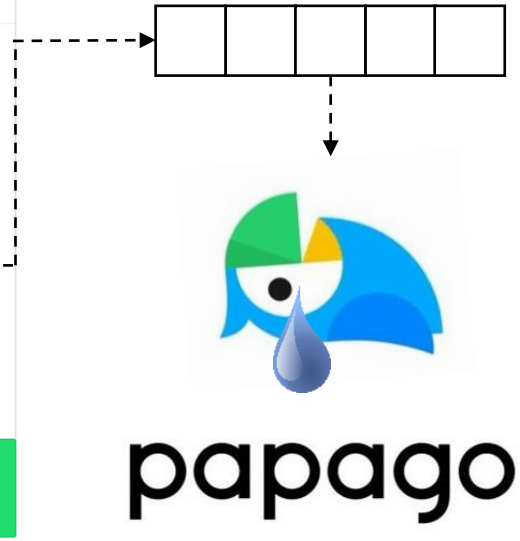
Data Mining Quality Analytics

- Seq2seq
  - Long term dependency
  - Vanishing/Exploding gradient

Encoder: RNN



Context vector



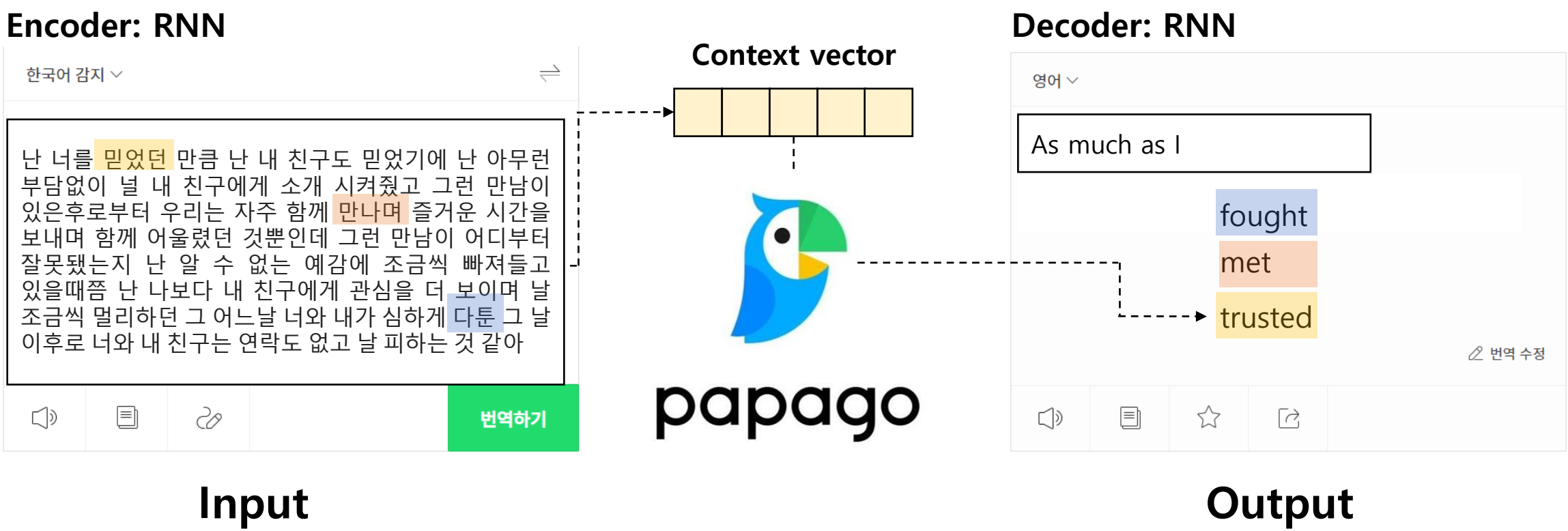
Decoder: RNN

Input

Output

Data Mining Quality Analytics

- Seq2seq with Attention
  - Relieve the encoder from the burden of having to encode all information into a fixed length vector



Data Mining Quality Analytics

- Key, Query, Value
  - Dictionary 자료 구조
  - Query와 Key가 일치하면 Value를 return

Query: 유재석

Key	Value
이효리	천옥
엄정화	만옥
제시	은비
화사	실비
유재석	지미 유

부캐 Dictionary

Output: 지미 유

Data Mining Quality Analytics

- Key, Query, Value
  - Dictionary 자료 구조
  - Query와 Key가 일치하면 Value를 return

```
① def Similarity(Query, Key)
        if Query = Key:
                return 1
        else:
                return 0
```

Query: 유재석

Key	Sim	Value
이효리	0	천옥
엄정화	0	만옥
제시	0	은비
화사	0	실비
유재석	1	지미 유

부캐 Dictionary

Data Mining Quality Analytics

- Key, Query, Value
  - Dictionary 자료 구조
  - Query와 Key가 일치하면 Value를 return

```
① def Similarity(Query, Key)
      if Query = Key:
          return 1
      else:
          return 0
```

```
② def SimXValue(Sim,Value)
      output = Sim X int(value)
      return output
```

Query: 유재석

Key	Sim	Value	SimXValue
이효리	0	천옥	0
엄정화	0	만옥	0
제시	0	은비	0
화사	0	실비	0
유재석	1	지미 유	지미 유

부캐 Dictionary

Data Mining Quality Analytics

- Key, Query, Value
  - Dictionary 자료 구조
  - Query와 Key가 일치하면 Value를 return

```
① def Similarity(Query, Key)
      if Query = Key:
          return 1
      else:
          return 0
```

```
② def SimXValue(Sim,Value)
      output = Sim X int(value)
      return output
```

Query: 유재석

Key	Value
이효리	천옥
엄정화	만옥
제시	은비
화사	실비
유재석	지미 유

부캐 Dictionary

SimXValue
0
0
0
0
지미 유

Result
지미 유

```
③ def Result(outputs)
      return sum(outputs)
```

- Key, Query, Value
  - Dictionary 자료형의 결과 리턴 과정
    - ① Similarity(key, value)
      - ✓ Key와 value의 유사도를 계산한다
    - ② SimXValue(sim, value)
      - ✓ 유사도와 value를 곱한다
    - ③ Result(outputs)
      - ✓ 유사도와 value를 곱한 값의 합을 리턴한다

$$result = \sum_{i} similarity(key, query) * value$$

```
① def Similarity(Query, Key)
        if Query = Key:
            return 1
        else:
            return 0
```

```
② def SimXValue(Sim,Value)
        output = Sim X int(value)
        return output
```

```
③ def Result(outputs)
        return sum(outputs)
```

Data Mining Quality Analytics

- Key, Query, Value in Attention
  - Attention: Query와 key의 유사도를 계산한 후 value의 가중합을 계산하는 과정
  - Attention score: Value에 곱해지는 가중치
  - Considerations
    - ✓ Key, Query, Value = Vectors (Matrix/Tensor)
    - ✓ Similarity function

$$output = \sum_{i} similarity(key, query) * value$$

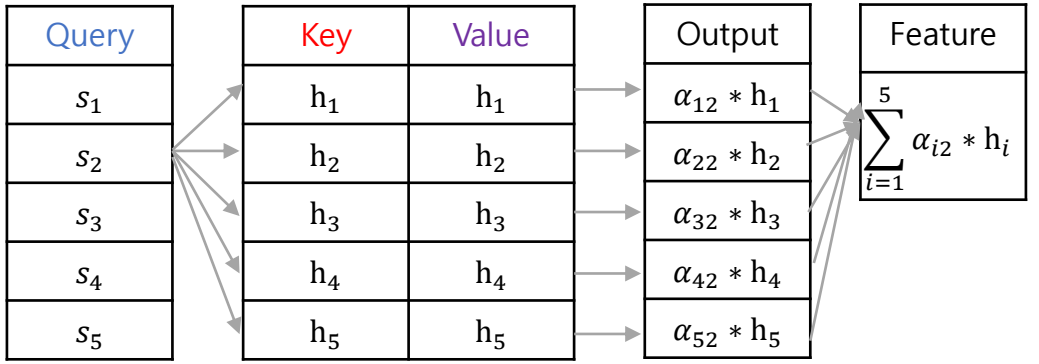
Dictionary 자료 구조

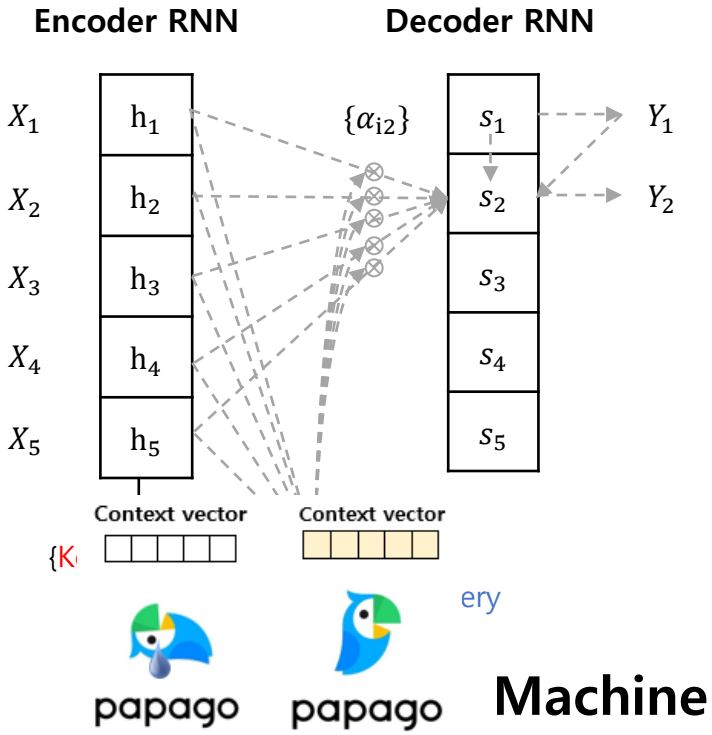
$$A(q, K, V) = \sum_{i} softmax(f(K, q))\, V$$

Attention

Data Mining Quality Analytics

- Attention in Seq2seq Machine Translation
  - Attention: Query와 key의 유사도를 계산한 후 value의 가중합을 계산하는 과정
  - Key, Value = Hidden states of encoder h_i
  - Query = Hidden state of decoder s_2
  - Feature = Context vector at time step 2





$$\{\alpha_{i2}\} = softmax(f(\mathrm{h}_i, s_2))$$

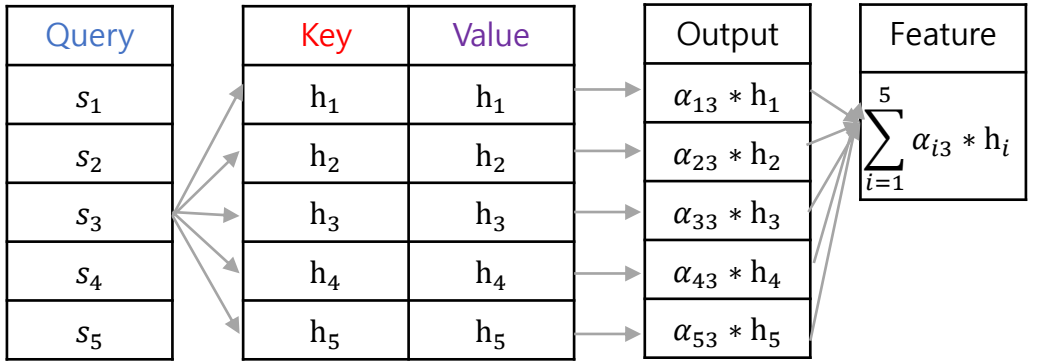
$$feature = \sum_{i=1}^{5} \alpha_{i2} * \mathrm{h}_i$$

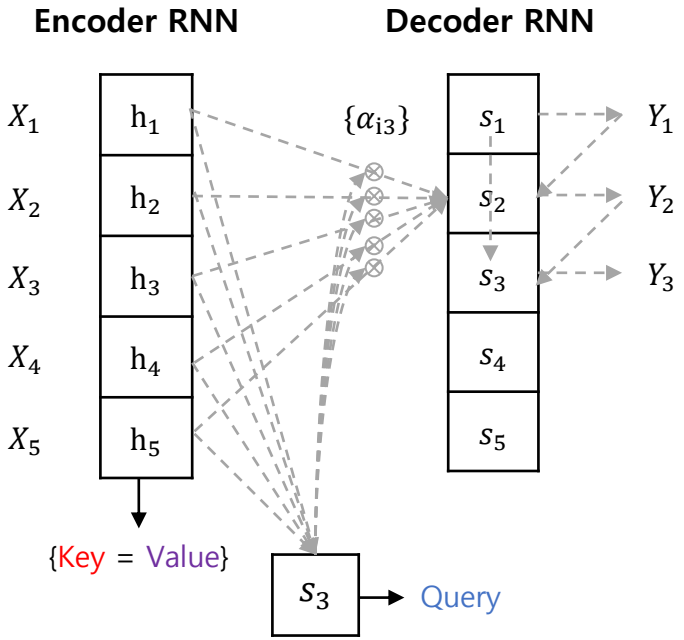
$$A(q, K, V) = \sum_{\mathrm{i}} softmax(f(K, q))\, V$$

Machine Translation

Data Mining Quality Analytics

- Attention in Seq2seq Machine Translation
  - Attention: Query와 key의 유사도를 계산한 후 value의 가중합을 계산하는 과정
  - Key, Value = Hidden states of encoder h_i
  - Query = Hidden state of decoder s_3
  - Feature = Context vector at time step 3





$$\{\alpha_{i3}\} = softmax(f(h_i, s_3))$$

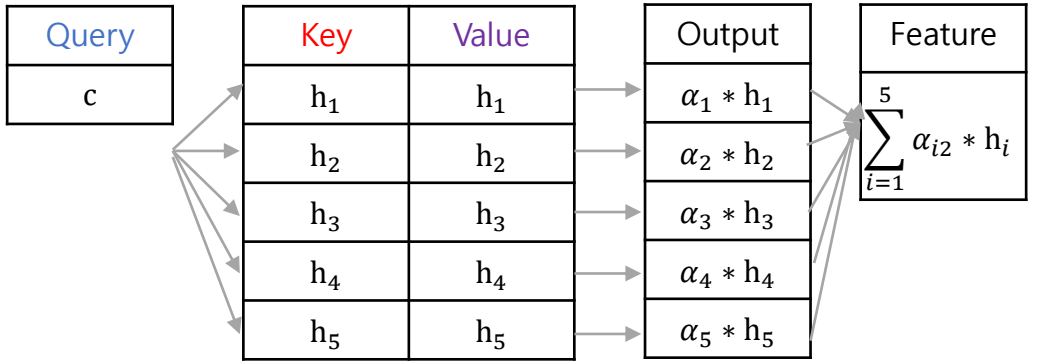
$$feature = \sum_{i=1}^{5} \alpha_{i3} * h_i$$

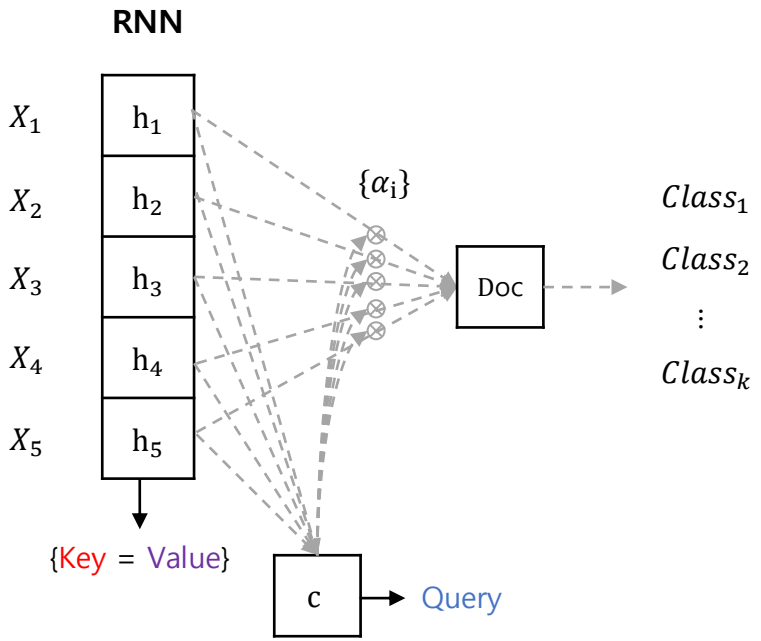
$$A(q, K, V) = \sum_{i} softmax(f(K, q))\, V$$

Machine Translation

Data Mining Quality Analytics

- Attention in RNN-based Document Classification
  - Attention: Query와 key의 유사도를 계산한 후 value의 가중합을 계산하는 과정
  - Feature = Document vector
  - Key, Value = Hidden states of RNN h_i
  - Query = Learnable parameter vector c (context vector)





$$\{\alpha_i\} = softmax(f(h_i, c))$$

$$feature = \sum_{i=1}^{5} \alpha_i * h_i$$

$$A(q, K, V) = \sum_{i} softmax(f(K, q))\, V$$

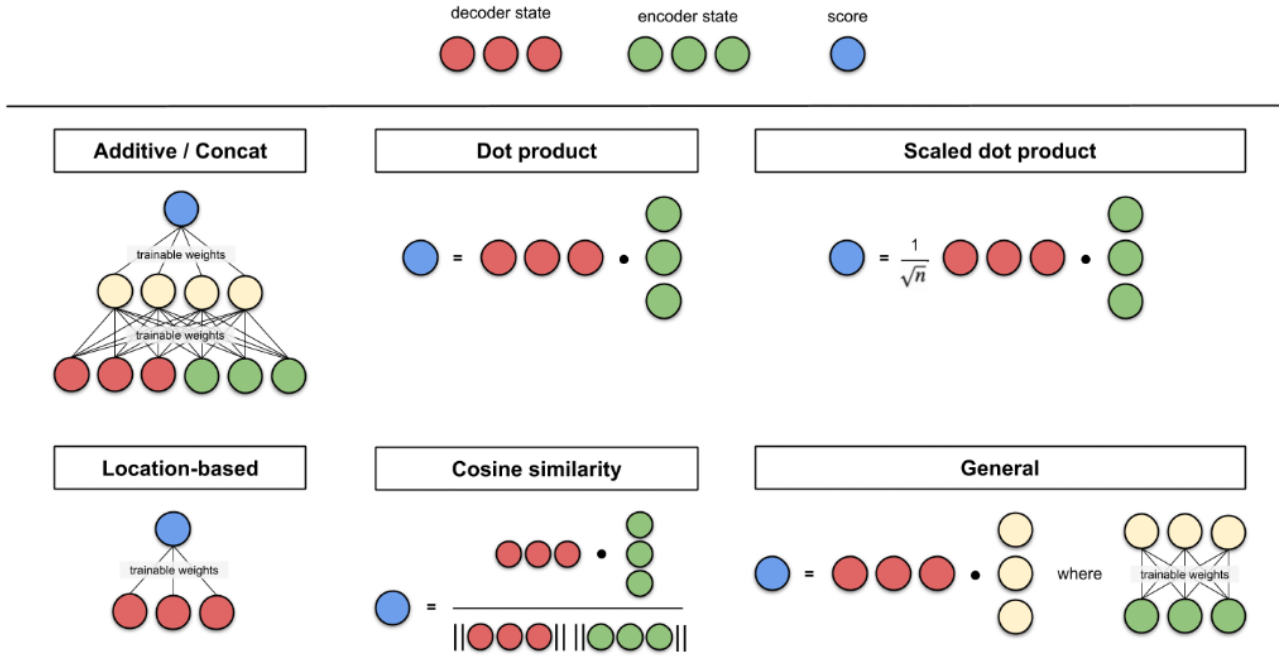
Document Classification

Data Mining Quality Analytics

- Similarity Function (Alignment model)
  - Vector간 유사도를 계산하는 다양한 방법을 similarity function으로 사용 가능
  - Bahdanau Attention (2014), Graph Attention Network (2018) -> Additive / Concat
  - Luong (2015) -> 다양한 similarity function을 제시
  - Transformer (2017) -> Scaled-dot product

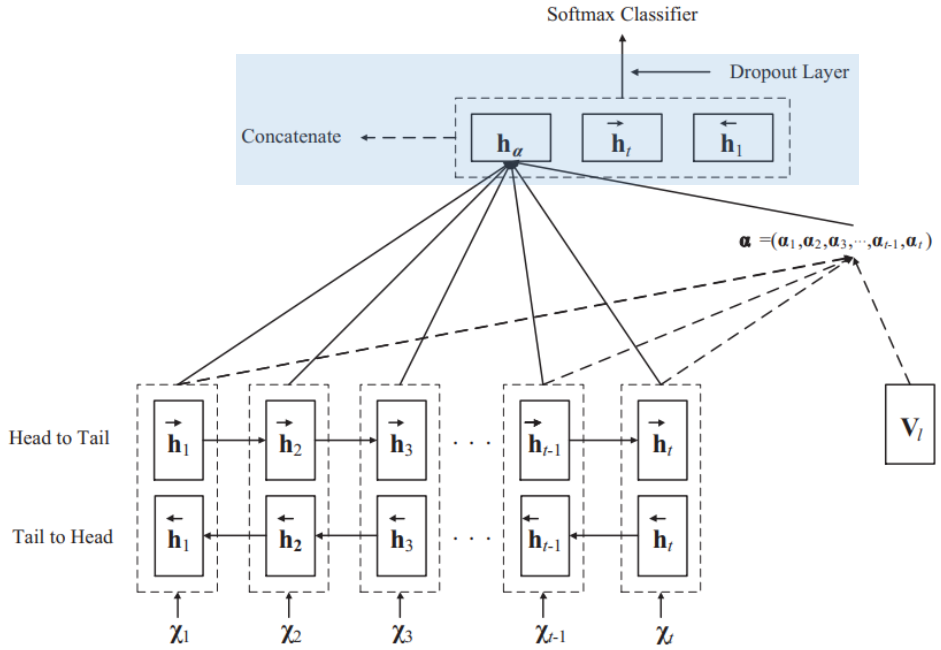
$$A(q, K, V) = \sum_i softmax(f(K, q))\, V$$

$$f(Key, Query) = score$$

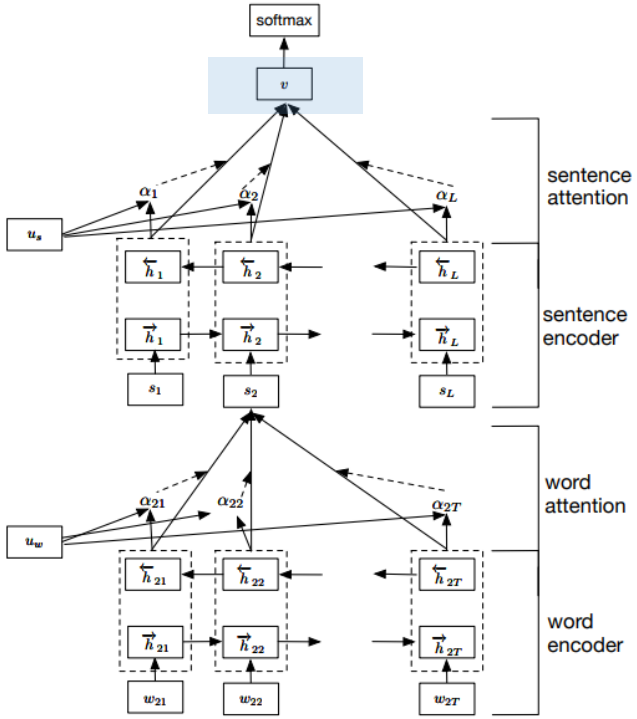


Data Mining Quality Analytics

- Feature Representation by RNN-based Network
  - Bi-RNN with Attention
  - Hierarchical Attention Network (2016)



Bidirectional RNN with Attention



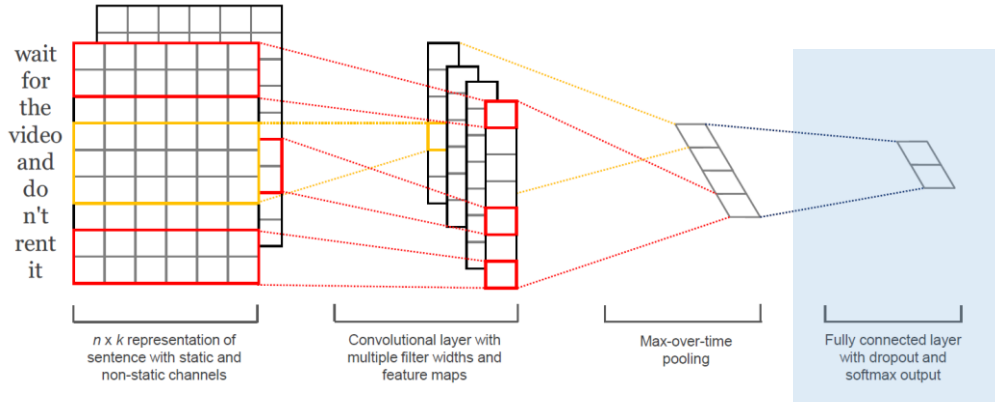
Hierarchical Attention Network

Liu, Tengfei, et al. "Recurrent networks with attention and convolutional networks for sentence representation and classification." *Applied Intelligence* 48.10 (2018): 3797-3806.

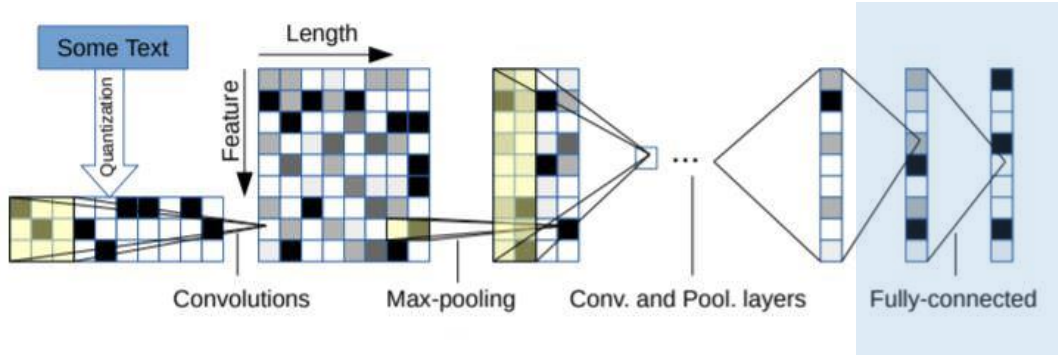
Yang, Zichao, et al. "Hierarchical attention networks for document classification." *Proceedings of the 2016 conference of the North American chapter of the association for computational linguistics* 2016.

Data Mining Quality Analytics

- Feature Representation by CNN-based Network
  - TextCNN (2014)
  - Character-level CNN (2015)



TextCNN



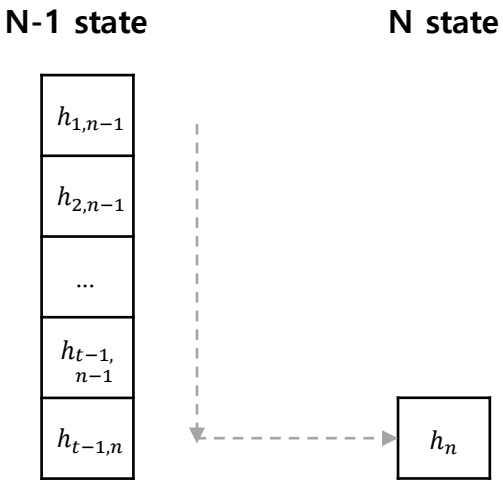
Character-level CNN

Kim, Yoon. "Convolutional neural networks for sentence classification." *arXiv preprint arXiv:1408.5882* (2014).

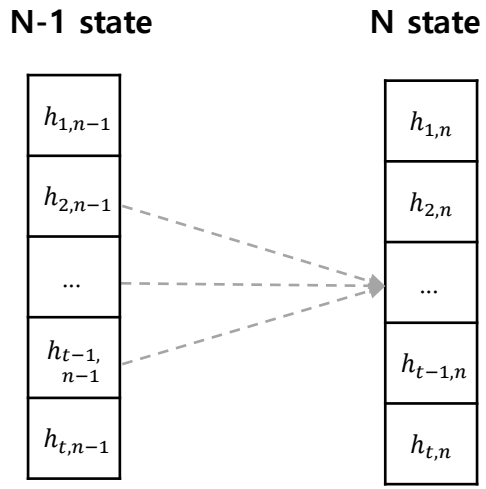
Zhang, Xiang, Junbo Zhao, and Yann LeCun. "Character-level convolutional networks for text classification." *Advances in neural information processing systems*. 2015.

Data Mining Quality Analytics

- Attention to Self-Attention
  - RNN-based Network
    - ✓ Sequential data → Parallel computing X
    - ✓ Calculation time and complexity ↑
    - ✓ Vanishing gradient / Long term dependency
  - CNN-based Network
    - ✓ Long path length between long-range dependencies



RNN-Based



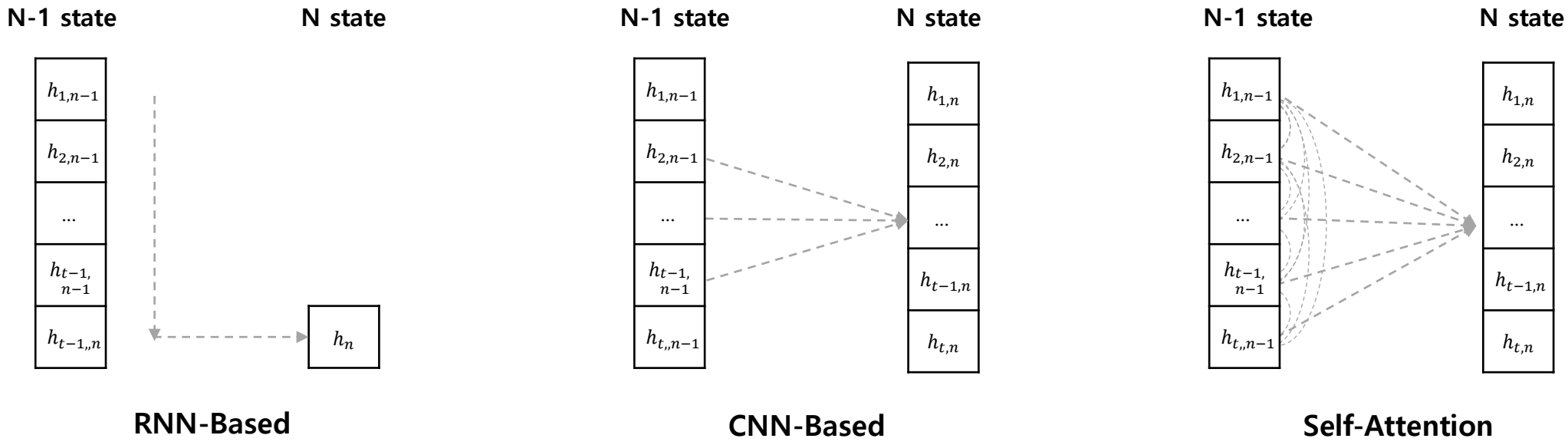
CNN-Based

Data Mining Quality Analytics

- Self-Attention
  - RNN, CNN 구조를 사용하지 않고 attention만을 사용하여 feature representation
    - ✓ Key = Query = Value = Hidden state of word embedding vector
    - ✓ Scaled dot-product attention
    - ✓ Multi-head attention

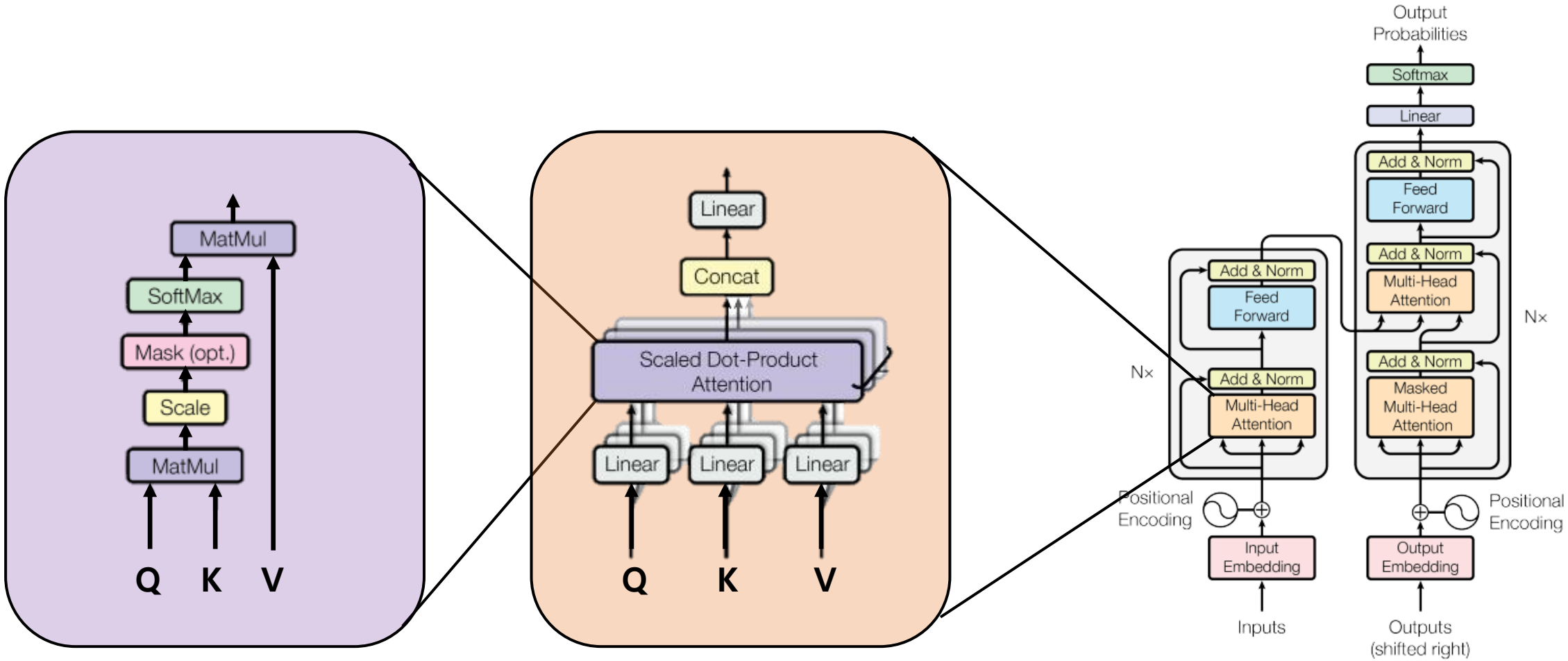
$$A(q, K, V) = \sum_{i} softmax(f(K, q))\, V$$

$$Key = Query = Value$$



Data Mining Quality Analytics

- Transformer

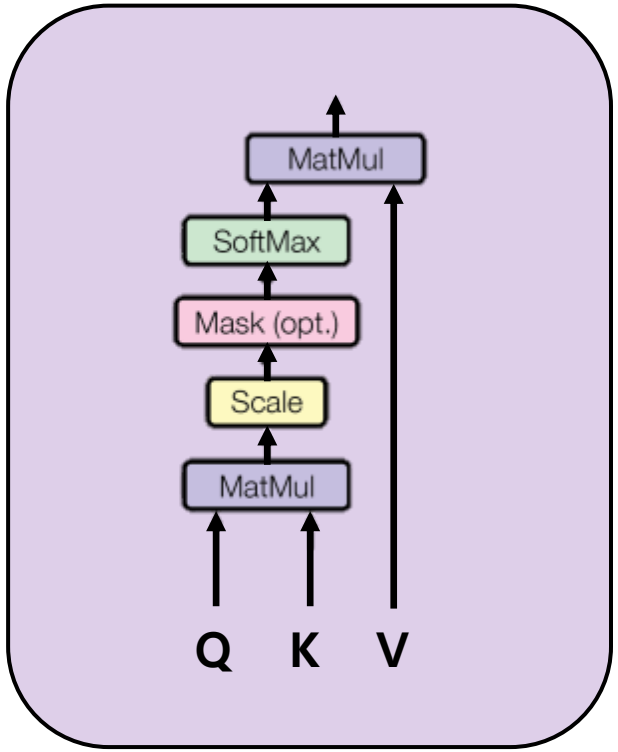


Scaled Dot-Product Attention **Multi-Head Attention** **Transformer**

Vaswani, Ashish, et al. "Attention is all you need." *Advances in neural information processing systems*. 2017..

Data Mining Quality Analytics

- Transformer
  - Scale-dot product attention (Self-Attention)
    - ✓ Key = Query = Value = Hidden state of word embedding vector (X)
    - ✓ Similarity function = Dot-product



Scaled Dot-Product Attention

Generalized Attention Form

$$A(q, K, V) = \sum_{i} softmax(f(K, q))\, V$$

① **MatMul**

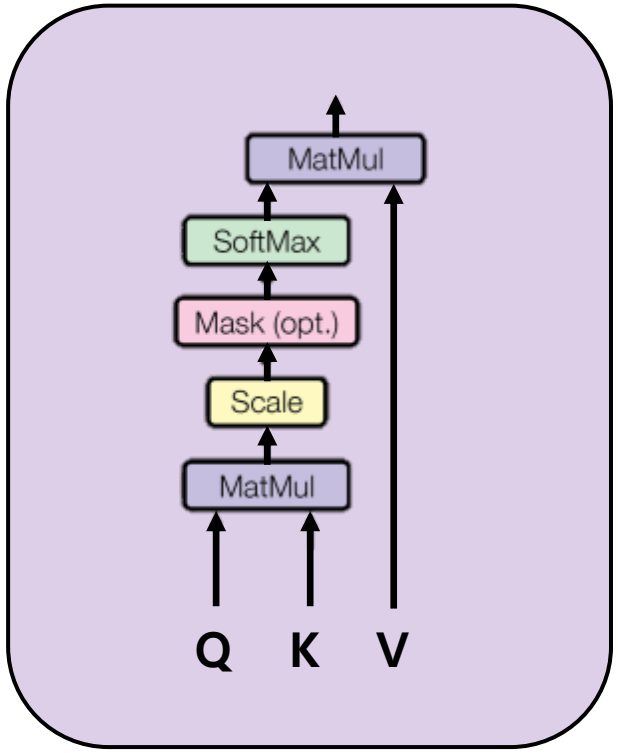
$$f(K, Q) = QK^{T} \quad (K = XW^{K}, Q = XW^{Q}, V = XW^{V})$$

While the two are similar in theoretical complexity, dot-product attention is much faster and more space-efficient in practice, since it can be implemented using highly optimized matrix multiplication code. (…)

Vaswani, Ashish, et al. "Attention is all you need." *Advances in neural information processing systems*. 2017..

Data Mining Quality Analytics

- Transformer
  - Scale-dot product attention (Self-Attention)
    - ✓ Similarity function: Scaled-dot product



Scaled Dot-Product Attention

Generalized Attention Form

$$A(q, K, V) = \sum_i softmax(f(K, q))\, V$$

① MatMul

② Scale

$$f(K, Q) = QK^T \quad (K = XW^K, Q = XW^Q, V = XW^V)$$

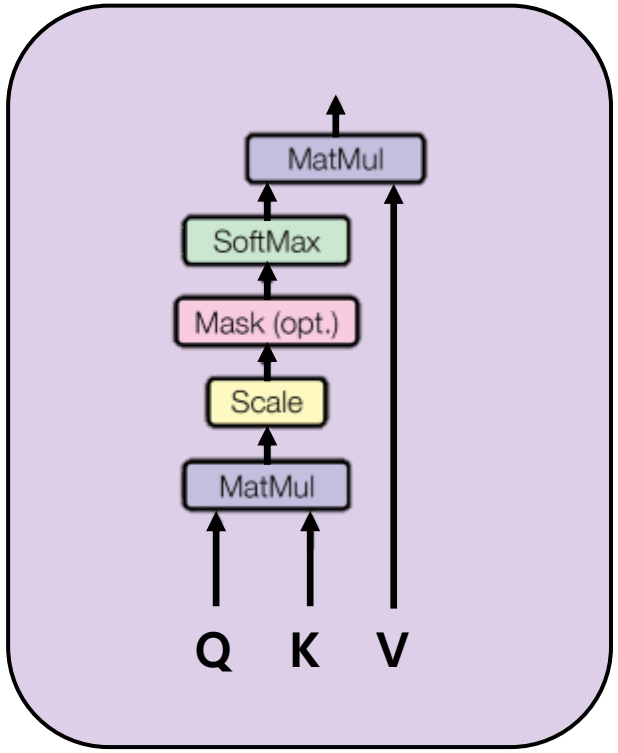
$$\frac{QK^T}{\sqrt{d_k}}$$

We suspect that for large values of dk, the dot products grow large in magnitude, pushing the softmax function into regions where it has extremely small gradients . To counteract this effect, we scale the dot products (...)

Vaswani, Ashish, et al. "Attention is all you need." *Advances in neural information processing systems*. 2017..

Data Mining Quality Analytics

- Transformer
  - Scale-dot product attention (Self-Attention)
    - ✓ Weight-sum of value vectors



Scaled Dot-Product Attention

Generalized Attention Form

$$A(q, K, V) = \sum_{i} softmax(f(K, q))\, V$$

① **MatMul** $f(K, Q) = QK^{T}$ $(K = KW^{K}, Q = QW^{Q}, V = QW^{V})$

② **Scale** $\frac{QK^{T}}{\sqrt{d_k}}$

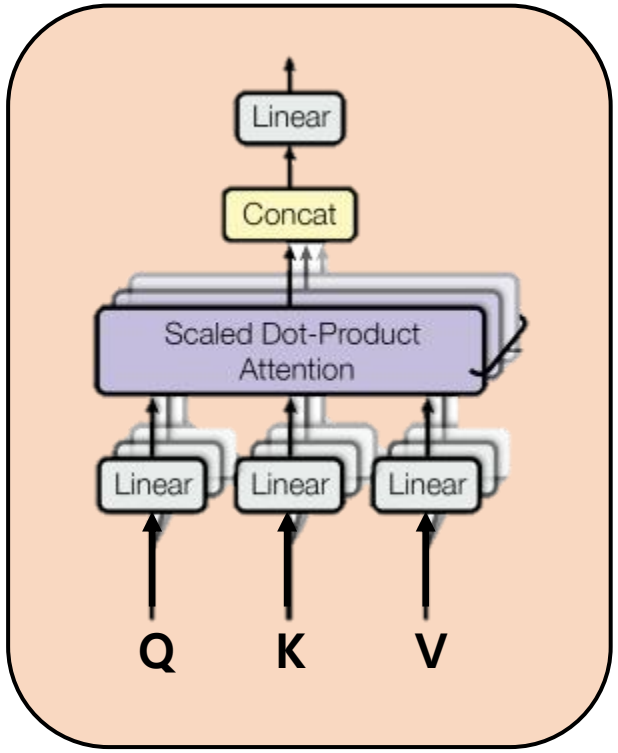
③ **Softmax** $softmax(\frac{QK^{T}}{\sqrt{d_k}})$

④ **MatMul** $softmax(\frac{QK^{T}}{\sqrt{d_k}})V$

Vaswani, Ashish, et al. "Attention is all you need." *Advances in neural information processing systems*. 2017..

Data Mining Quality Analytics

- Transformer
  - Multi-head Attention
    - ✓ Learning diverse input features



Multi-Head Attention

Self-Attention

$$SA(q, K, V) = softmax(\frac{QK^T}{\sqrt{d_k}})V$$

① **Linear** $Q` = QW_i^Q \quad K` = KW_i^K \quad V` = VW_i^V \quad (i = 1 \dots h)$

② **Self-Atten** $\text{head}_i = SA(Q`, K`, V`)$

③ **Concat** $[\text{head}_1, \text{head}_2, \dots, \text{head}_h]$

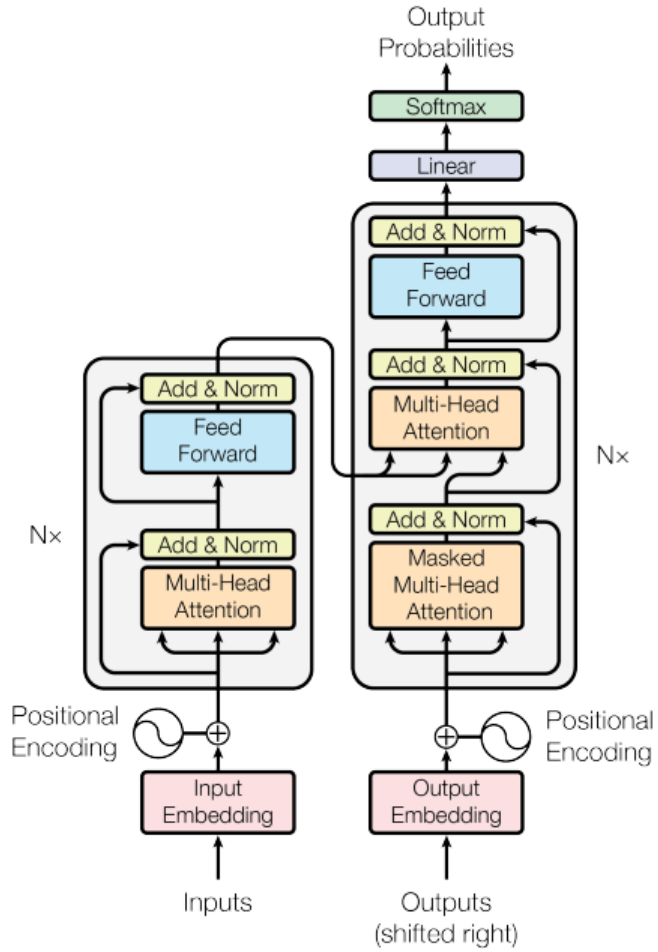
④ **Linear** $[\text{head}_1, \text{head}_2, \dots, \text{head}_h]W^O = \boldsymbol{MultiHead(Q, K, V)}$

Vaswani, Ashish, et al. "Attention is all you need." *Advances in neural information processing systems*. 2017..

Data Mining Quality Analytics

- Transformer
  - Contributions

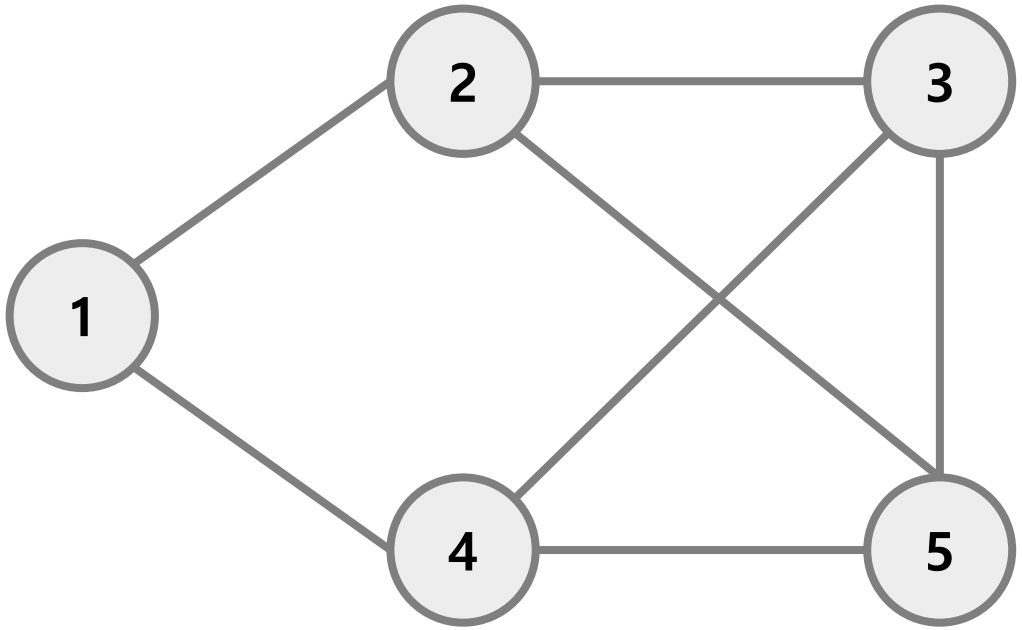
- One is the total computational complexity per layer. (…)
- Another is the amount of computation that can be parallelized, (…)
- The third is the path length between long-range dependencies in the network. Learning long-range dependencies is a key challenge in many sequence transduction tasks. (…)
- As side benefit, self-attention could yield more interpretable models.



Transformer

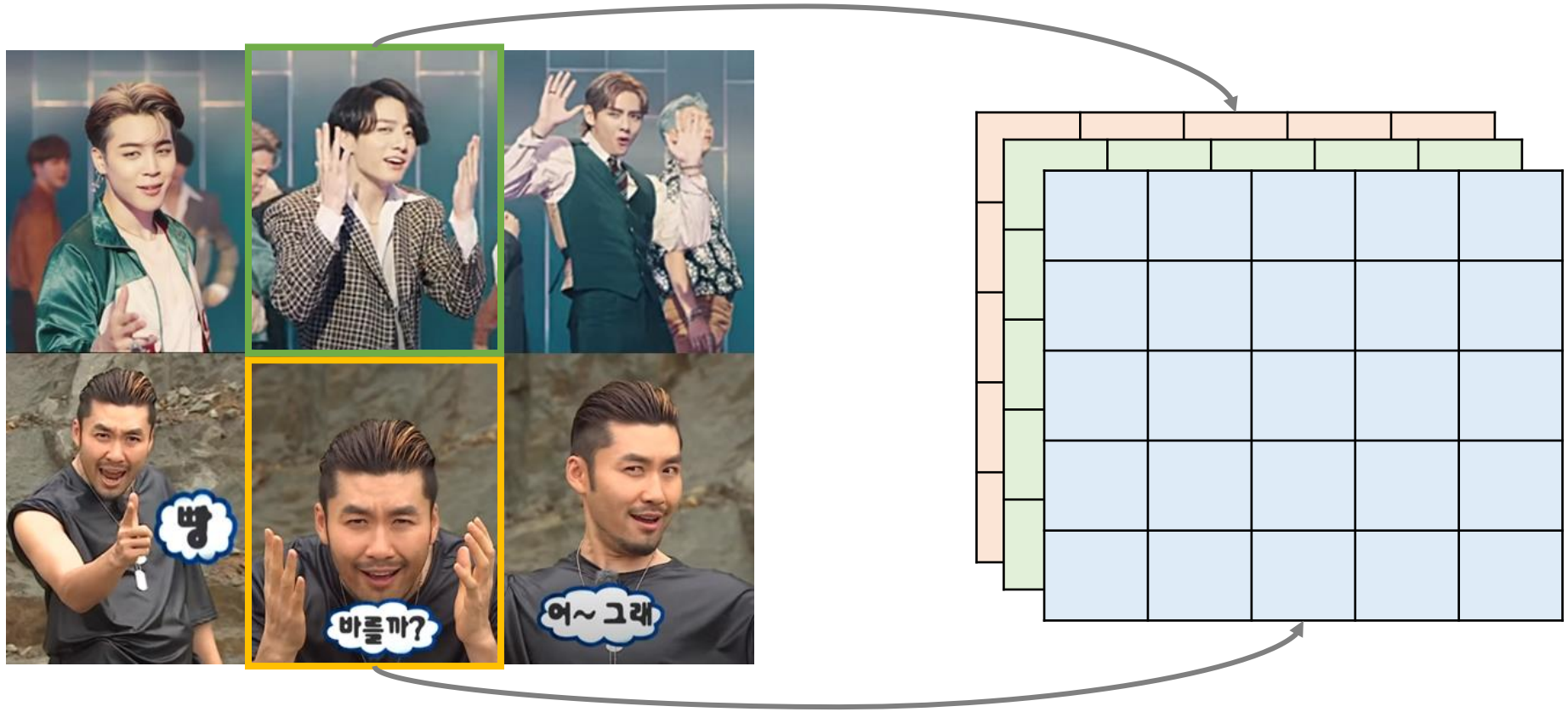
Vaswani, Ashish, et al. "Attention is all you need." *Advances in neural information processing systems*. 2017..

- Graph Data Structure
  - Node = Vertex
    - ✓ Represent elements of a system
  - Edge
    - ✓ Relationship or interaction between nodes



Data Mining Quality Analytics

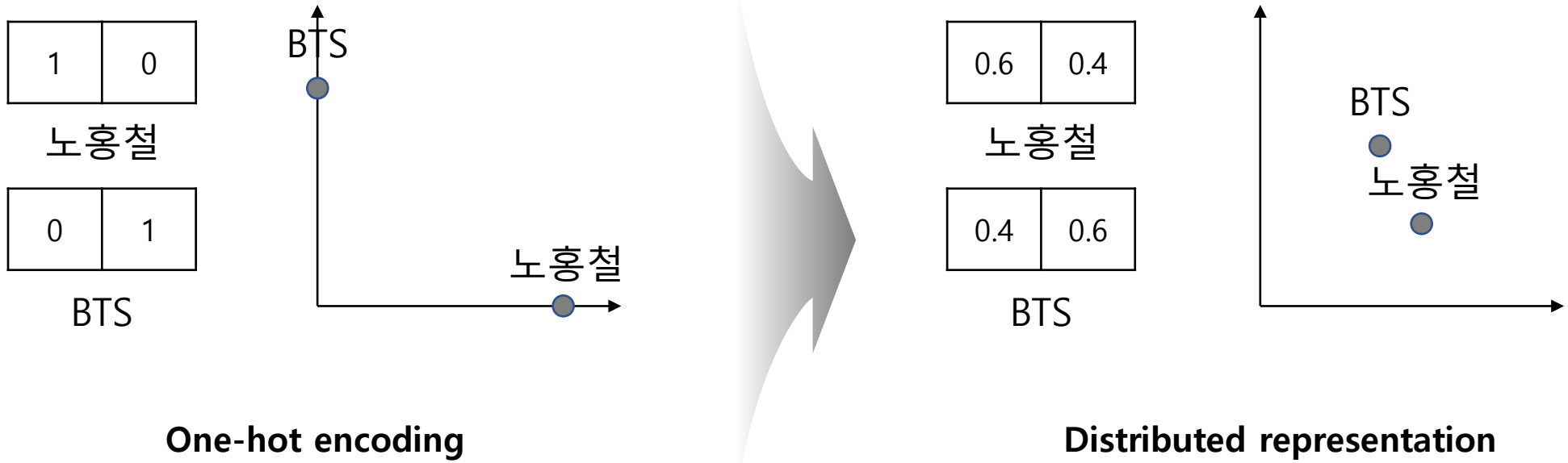
- Graph Data Structure
  - Image data: Euclidean space



[3 X W X H] dimension

Data Mining Quality Analytics

- Graph Data Structure
  - Text data: Euclidean space



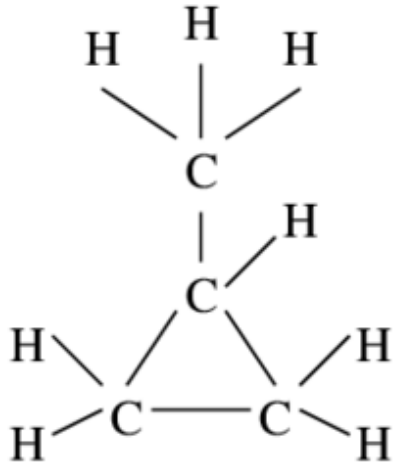
One-hot encoding

Distributed representation

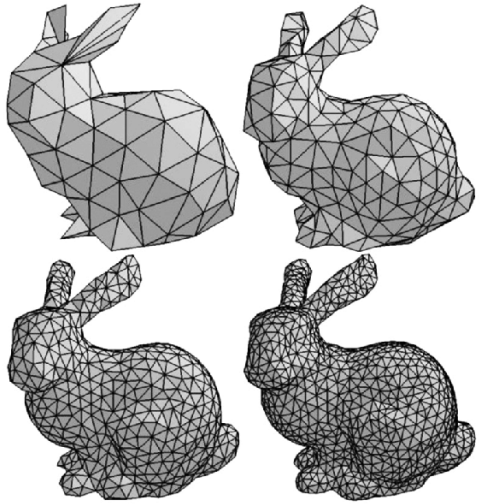
Data Mining Quality Analytics

- Graph Data Structure
  - Graph data: Non-Euclidean Space

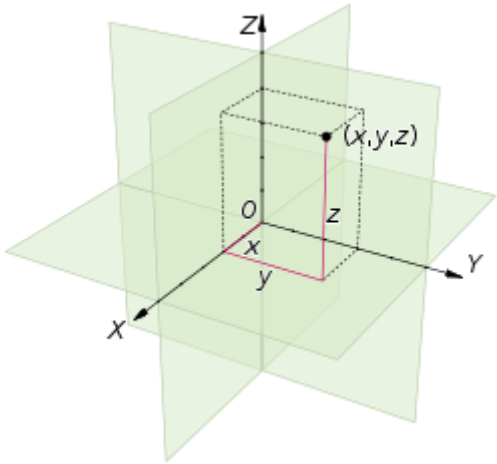
Social Network



Molecular Graph



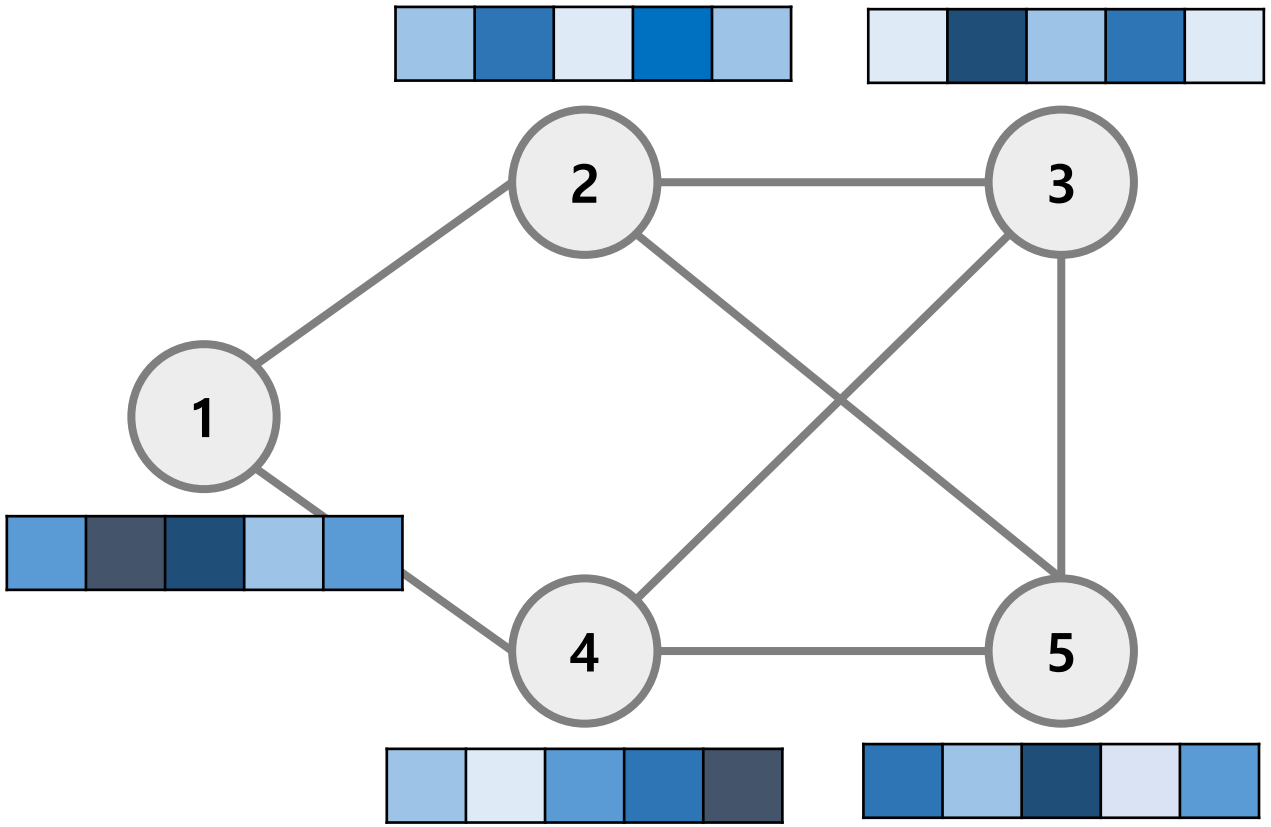
3D Mesh



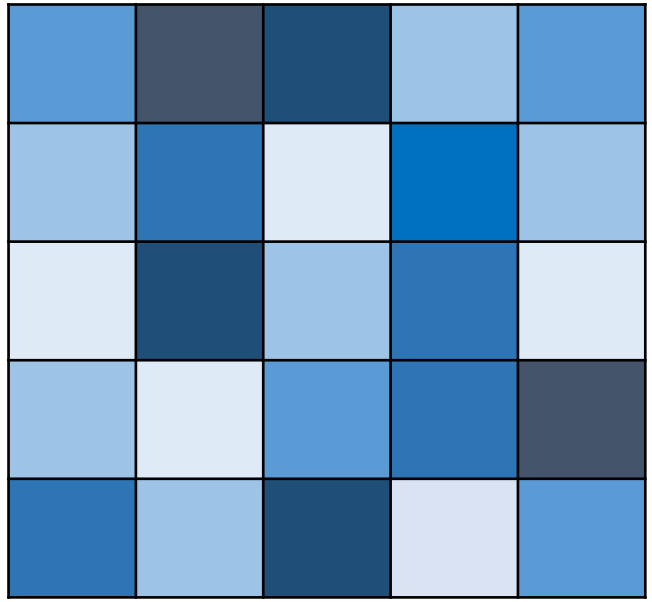
Euclidean Space

Data Mining Quality Analytics

- Matrix Representation of Graph
  - Node-Feature matrix
    - ✓ N by D dimension

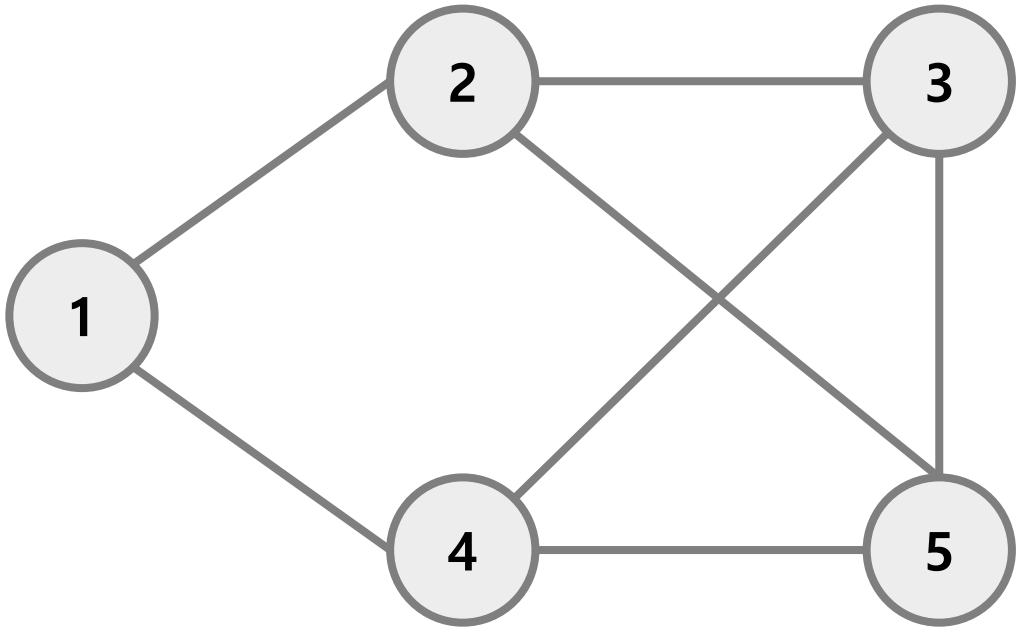


[Node-Feature Matrix]



Data Mining Quality Analytics

- Matrix Representation of Graph
  - Adjacency matrix: Undirected graph
    - ✓ N by N square matrix
    - ✓ Symmetric

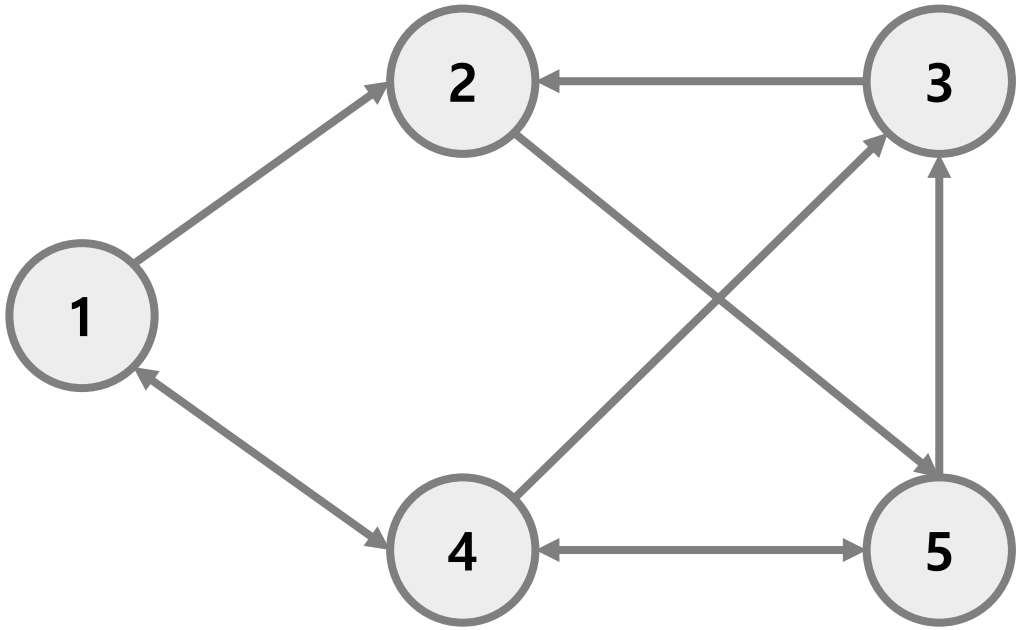


[Adjacency Matrix]

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

Data Mining Quality Analytics

- Matrix Representation of Graph
  - Adjacency matrix: Directed graph
    - ✓ N by N
    - ✓ Asymmetric

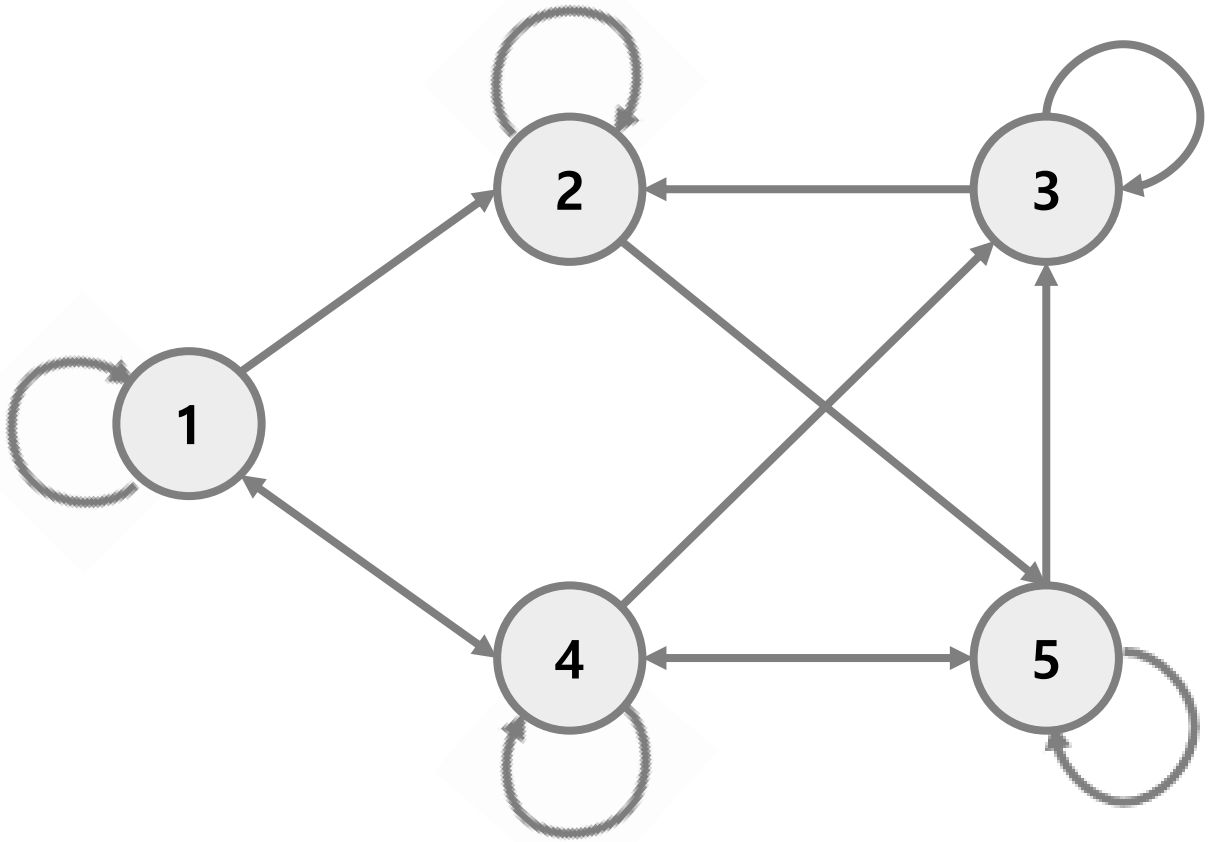


[Adjacency Matrix]

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

Data Mining Quality Analytics

- Matrix Representation of Graph
  - Adjacency matrix: Directed graph
    - ✓ Adjacency matrix + Identity matrix

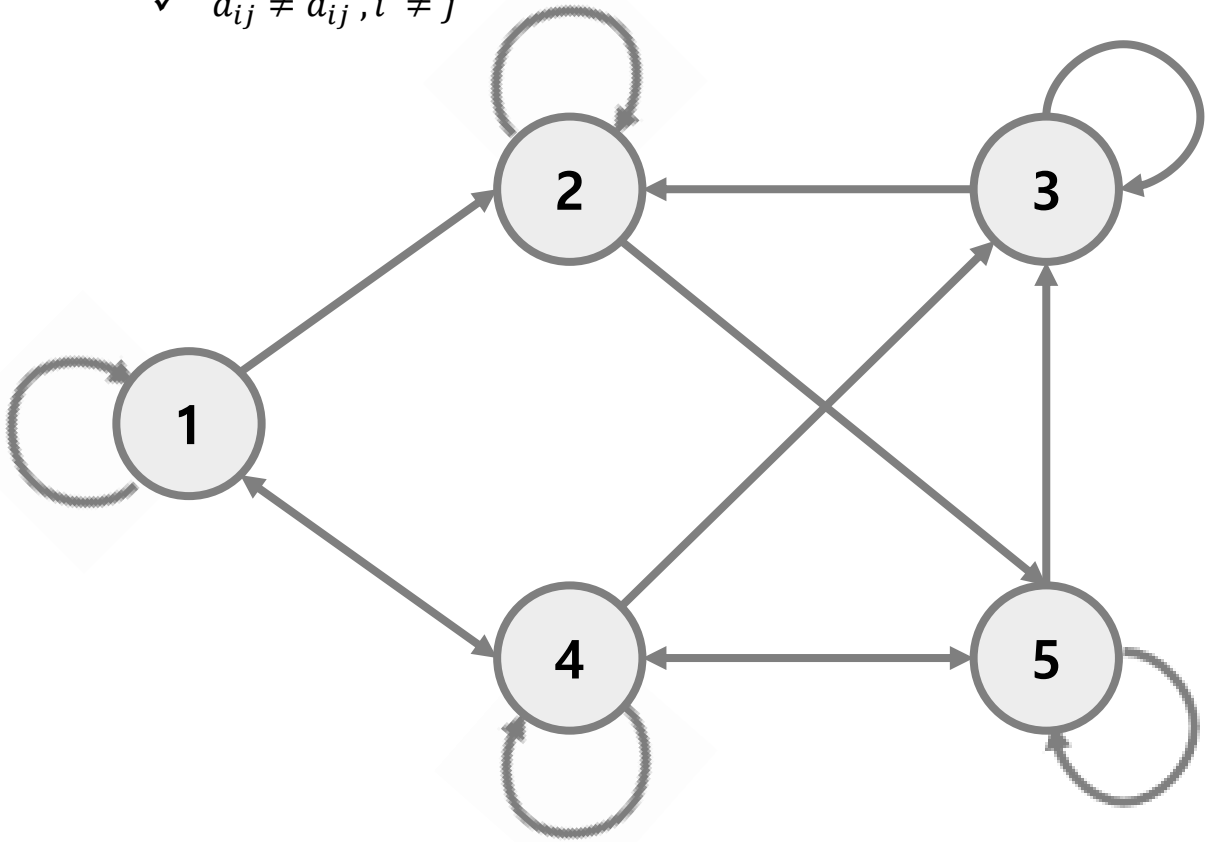


[Adjacency Matrix]

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

Data Mining Quality Analytics

- Matrix Representation of Graph
  - Adjacency matrix: Weighted directed graph
    - ✓ Edge information
    - ✓ $a_{ij} \neq a_{ij}, i \neq j$

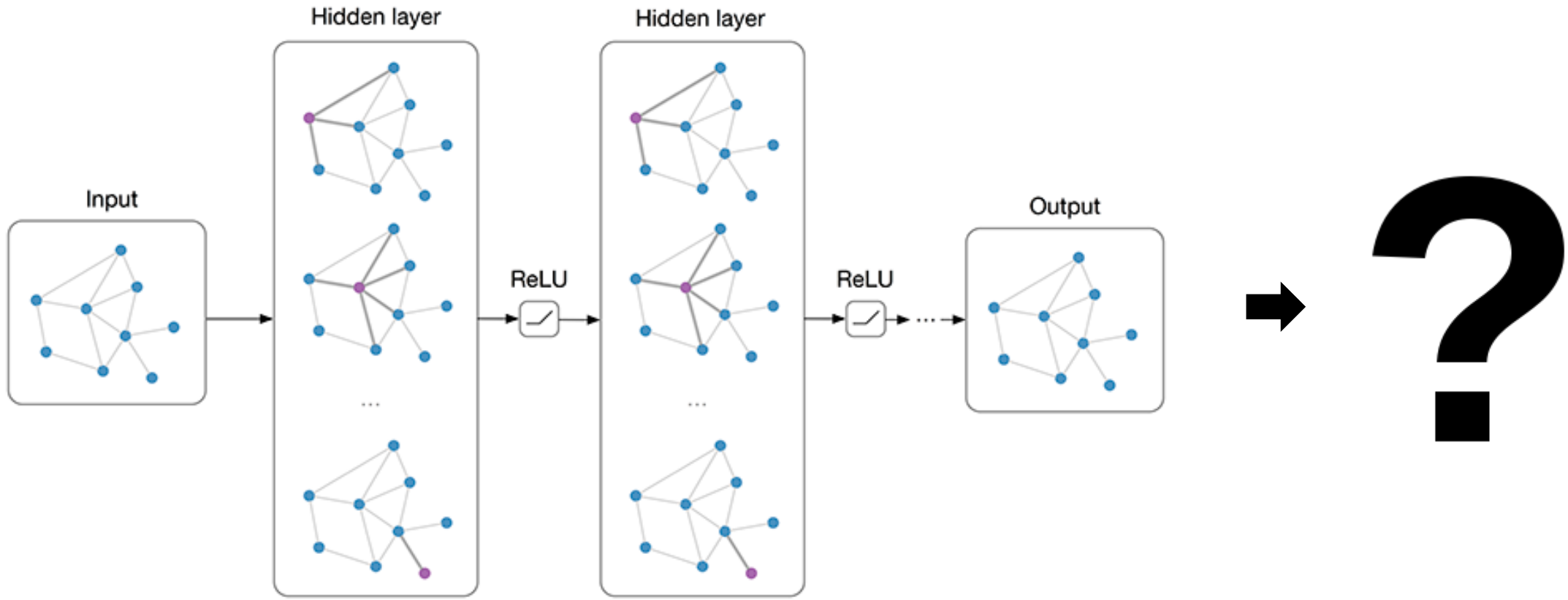


[Adjacency Matrix]

a_{11}	a_{12}	0	a_{14}	0
0	a_{22}	0	0	a_{25}
0	a_{32}	a_{33}	0	0
a_{41}	0	a_{43}	a_{44}	a_{45}
0	0	a_{53}	a_{54}	a_{55}

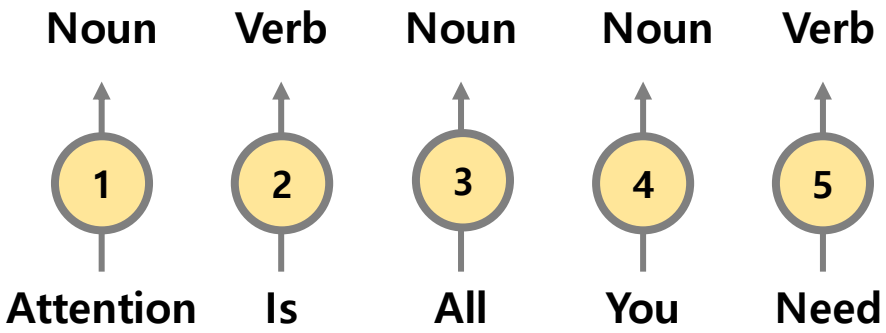
Data Mining Quality Analytics

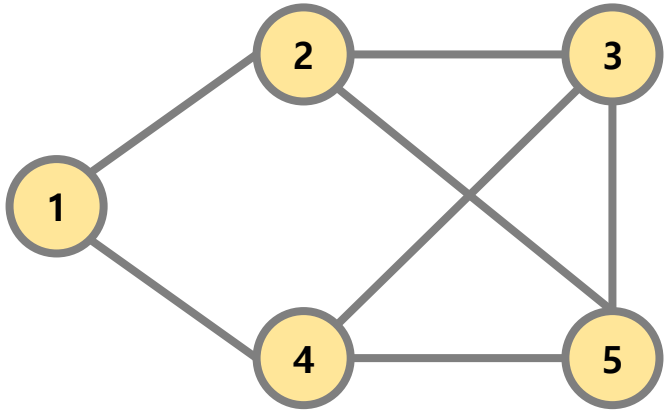
- Graph Neural Networks
  - Neural Networks for learning the structures of graphs



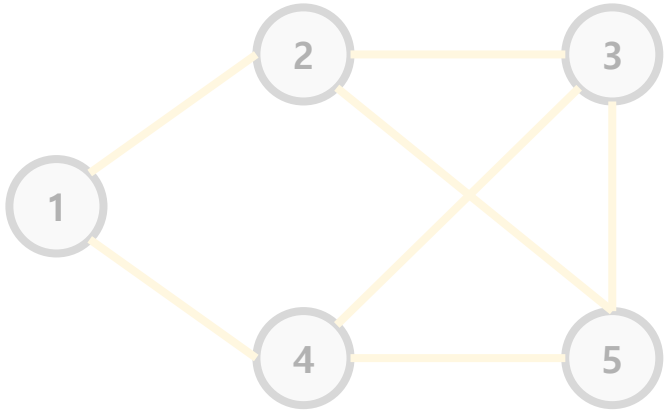
Data Mining Quality Analytics

- GNN Tasks
  - Node Level
  - Edge Level
  - Graph Level

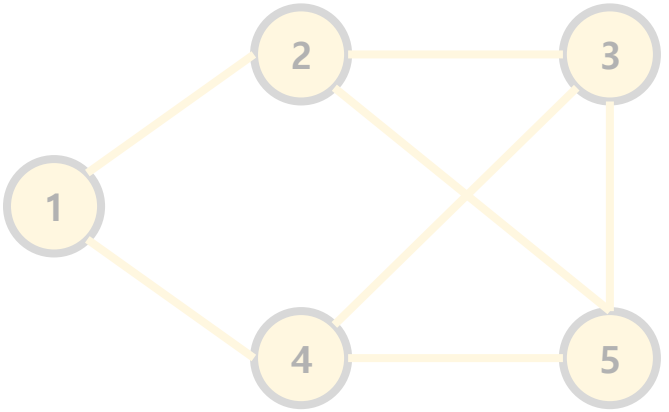




Node Level



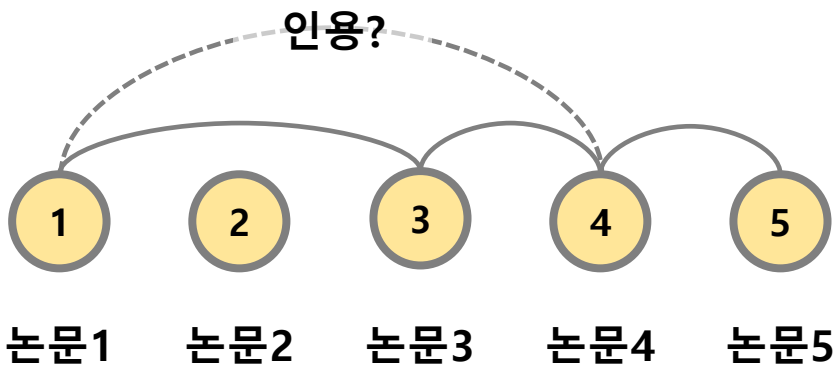
Edge Level

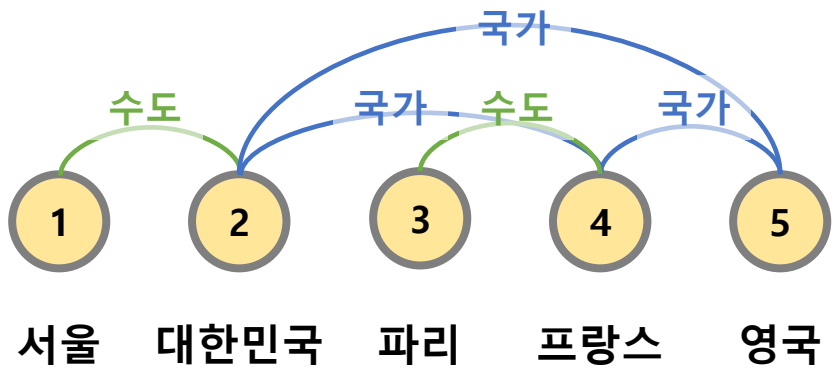


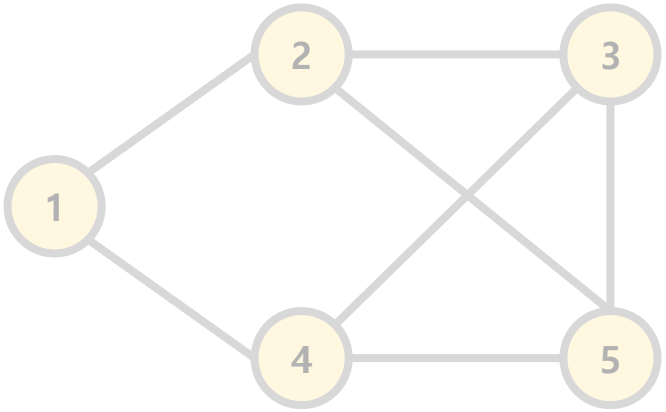
Graph Level

Data Mining Quality Analytics

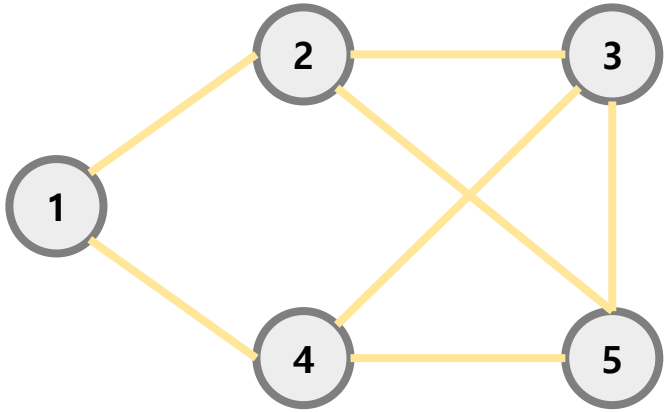
- GNN Tasks
  - Node Level
  - Edge Level
  - Graph Level



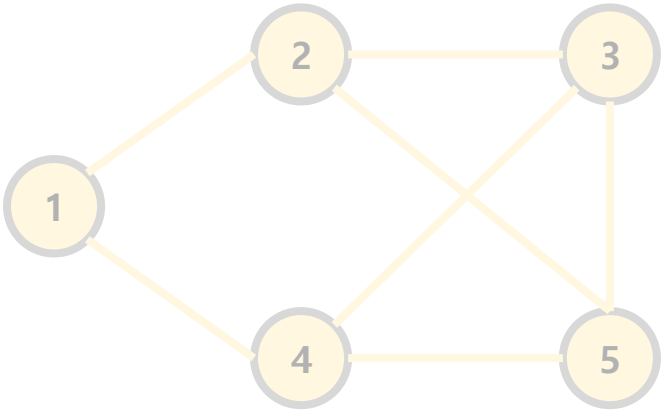




Node Level



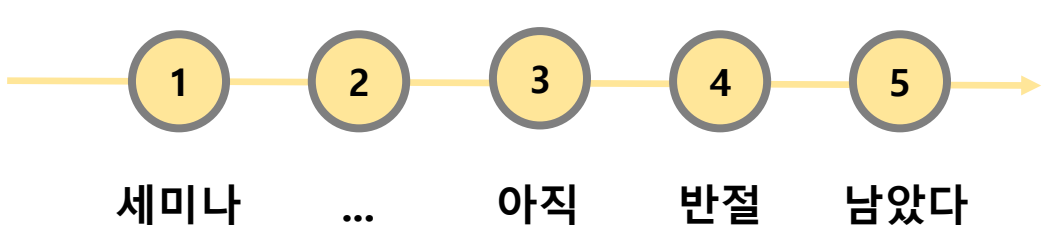
Edge Level

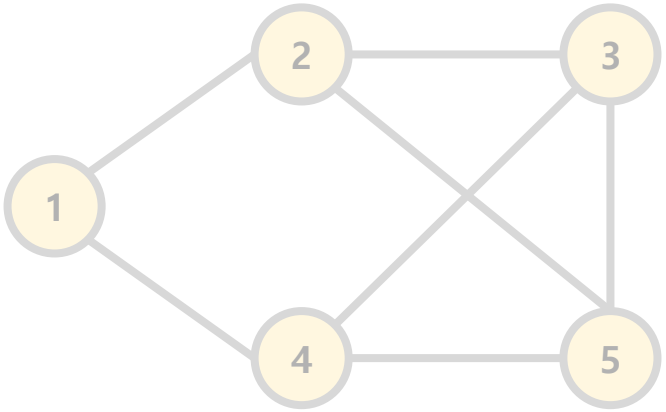


Graph Level

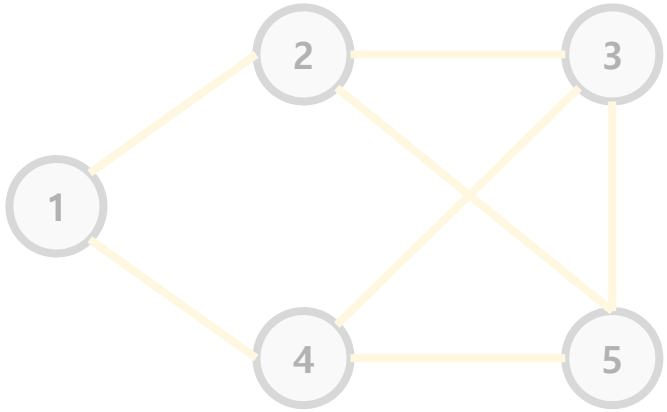
Data Mining Quality Analytics

- GNN Tasks
  - Node Level
  - Edge Level
  - Graph Level

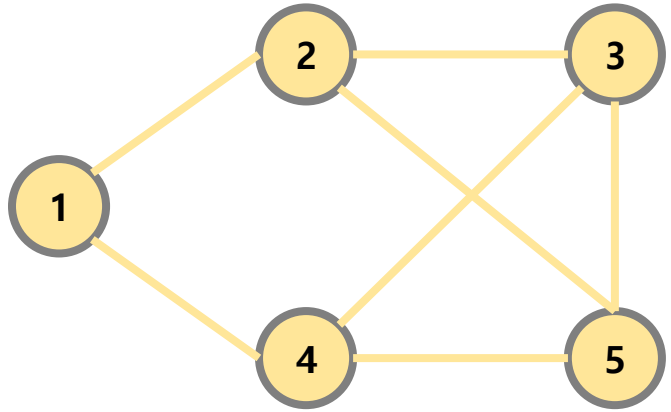




Node Level



Edge Level



Graph Level

Data Mining Quality Analytics

- Graph Task
  - Computer Vision
  - Natural Language Processing
  - Etc.

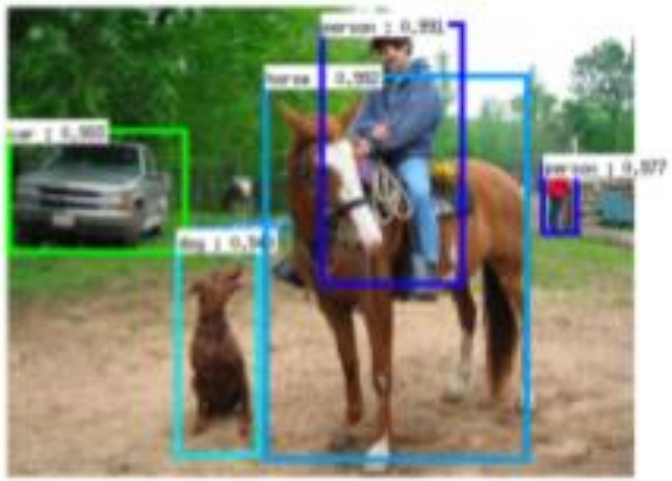
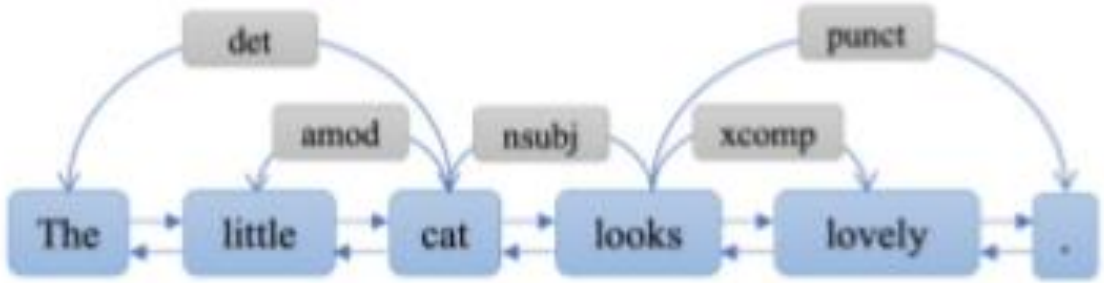


Image Graph

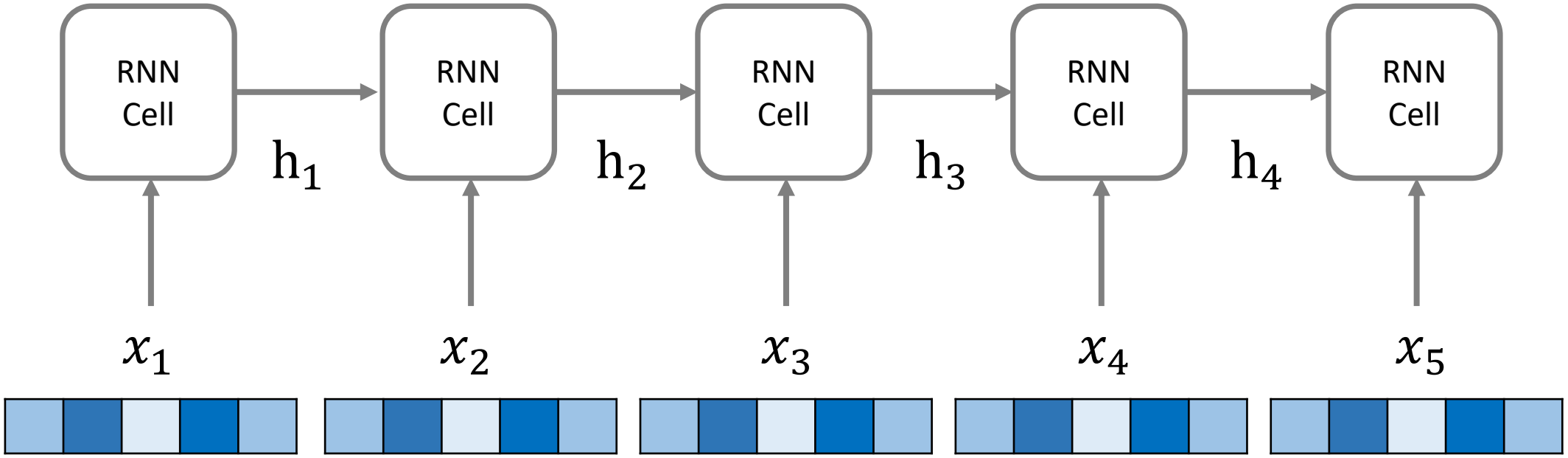


Text Graph

Zhou, Jie, et al. "Graph neural networks: A review of methods and applications." *arXiv preprint arXiv:1812.08434* (2018).

Data Mining Quality Analytics

- GNN Learning Process
  - RNN learning process

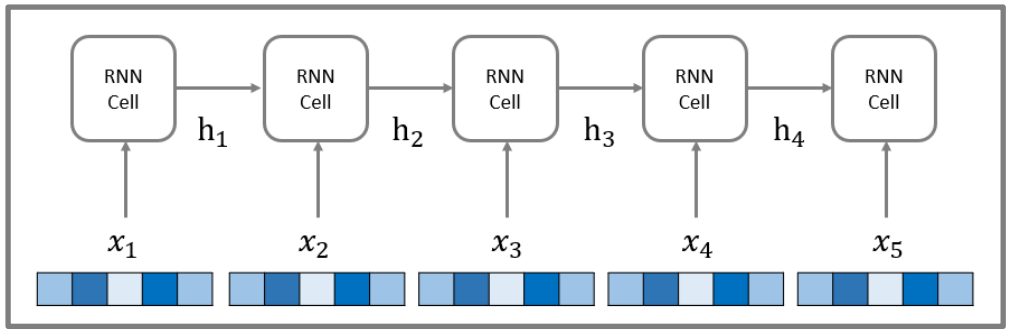


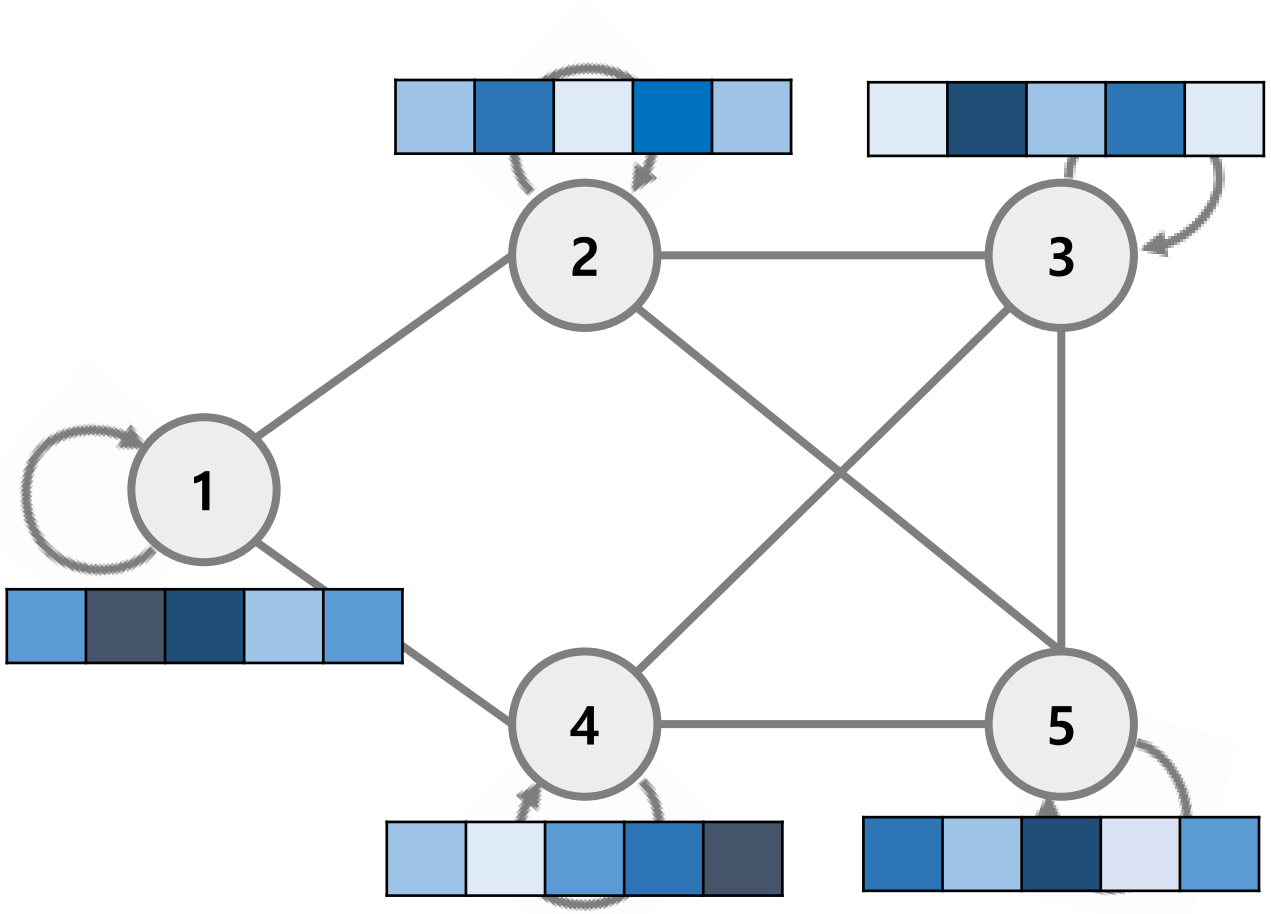
$$h_t = combine(h_{t-1}, x_t)$$

$$combine \in \{RNN, LSTM, GRU\}$$

Data Mining Quality Analytics

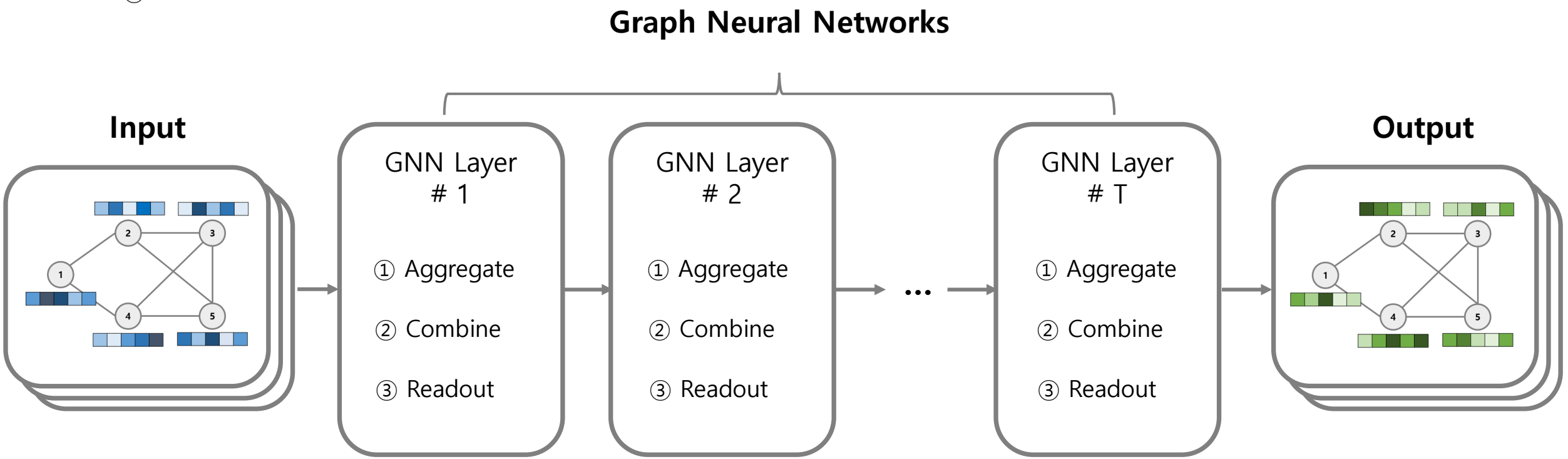
- GNN Learning Process
  - Graph: different with sequential data
    - ✓ No sequences(ordering)
    - ✓ Various graph structures
    - ✓ Multiple in-edge per node
  - Considerations for encoding graphs
    - ✓ Information passes along edges
    - ✓ Information passes in parallel
    - ✓ Target nodes affected by multiple nodes





Data Mining Quality Analytics

- GNN Layer
  - Node feature update reflecting graph structures
    - ① Aggregate / Massage passing
    - ② Combine / Update
    - ③ Readout



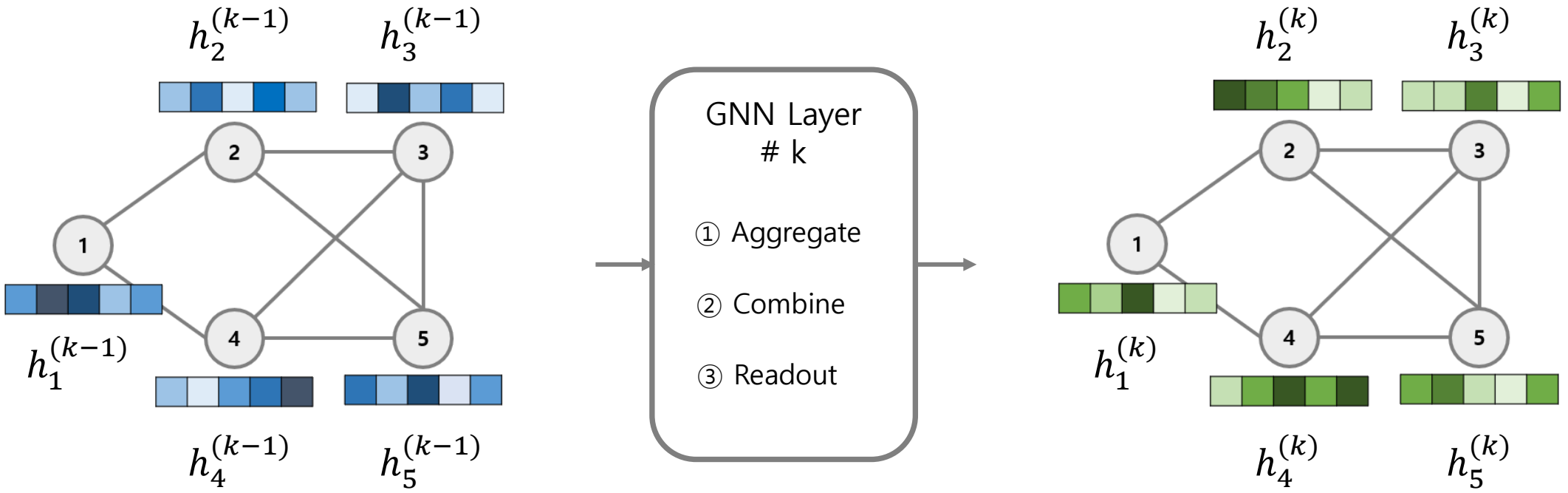
Data Mining Quality Analytics

- GNN Notation
  - $h_v^{(k)}$: hidden embedding node v at kth GNN layer
  - $v = target\ node$
  - $N(v) = neightbor\ nodes\ of\ v$
  - $u = neightbor\ node \in N(v)$

$$graph = G(A, X)$$

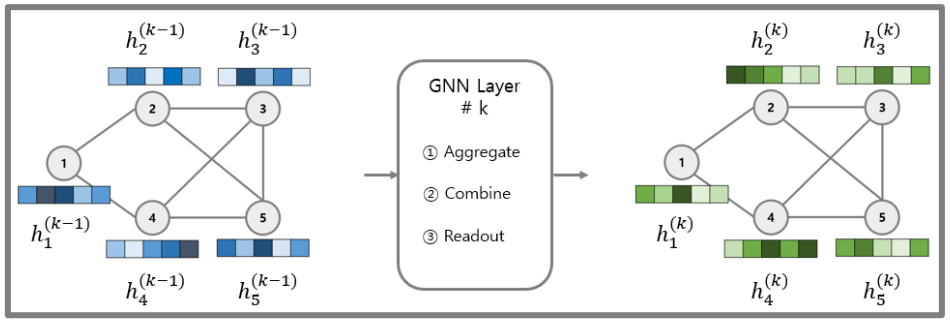
$$X = Node - Feature\ Matrix$$

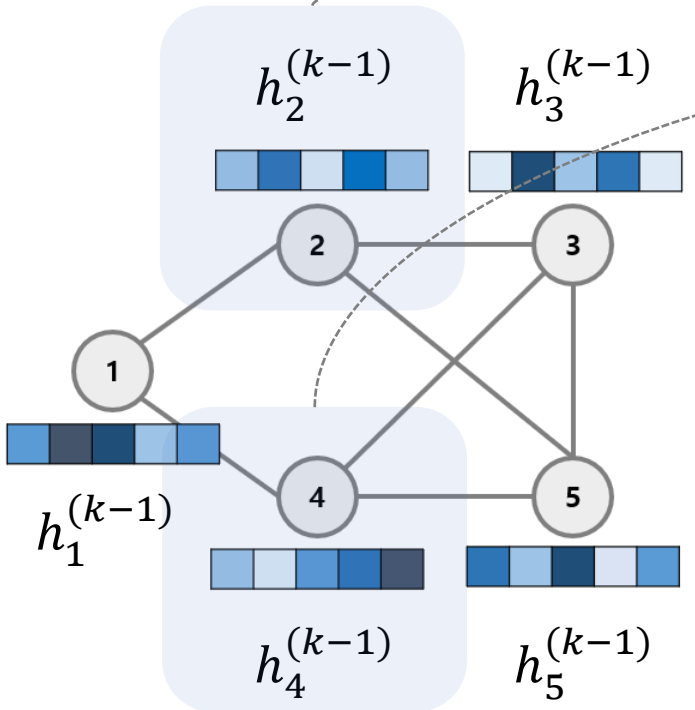
$$A = Adjacency\ Matrx$$



Data Mining Quality Analytics

- GNN: ① Aggregate
  - 타겟 노드의 이웃 노드들의 k-1 시점의 hidden state를 결합



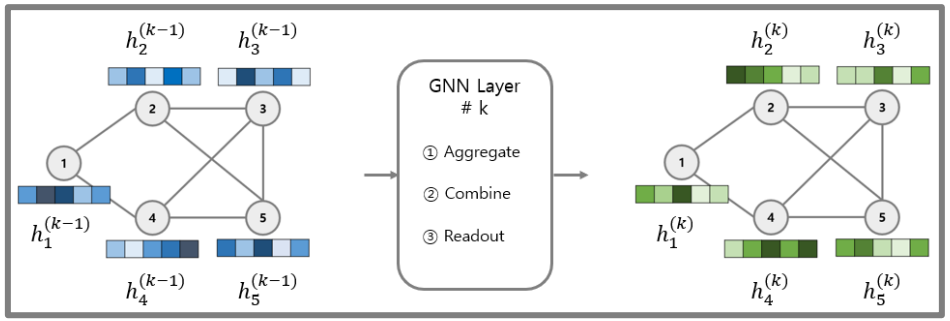


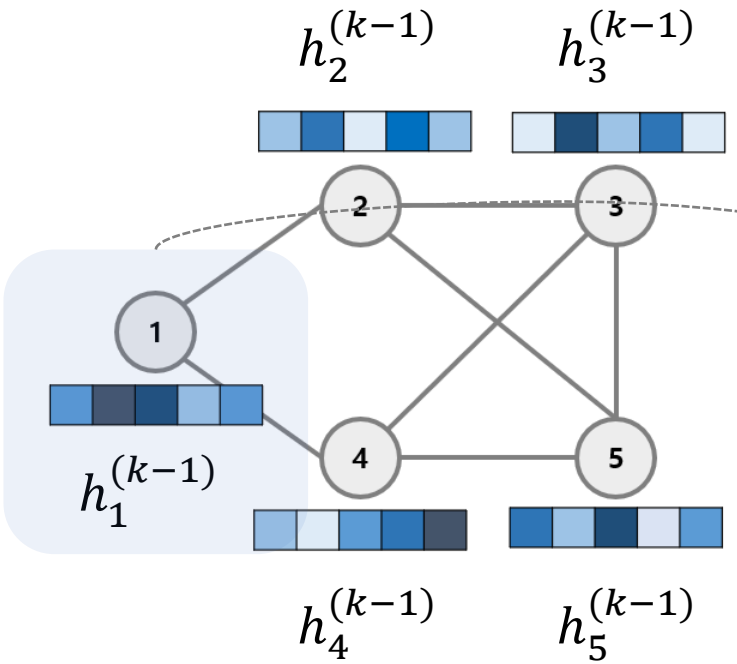
$$a_1^{(k-1)} = aggregate^k(\{ \quad , \quad \})$$

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

Data Mining Quality Analytics

- GNN: ② Combine
  - k-1 시점 target node의 hidden state와 aggregated information을 사용하여 k 시점의 target node의 hidden state를 update





$$a_1^{(k-1)} = aggregate^k(\{ \quad , \quad \})$$

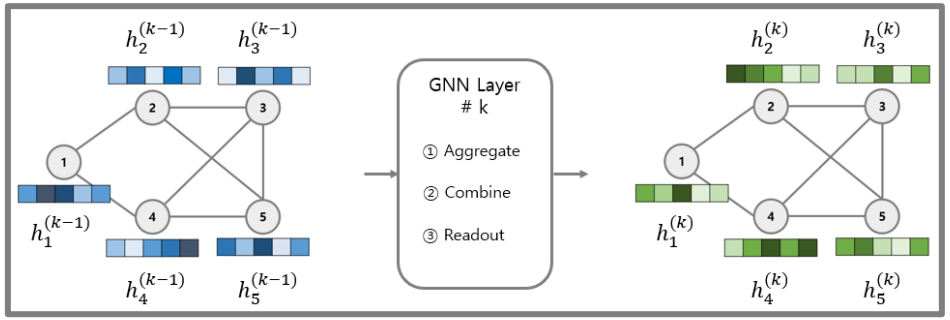
$$h_1^{(k)} = combine^k(a_1^{(k-1)}, \quad)$$

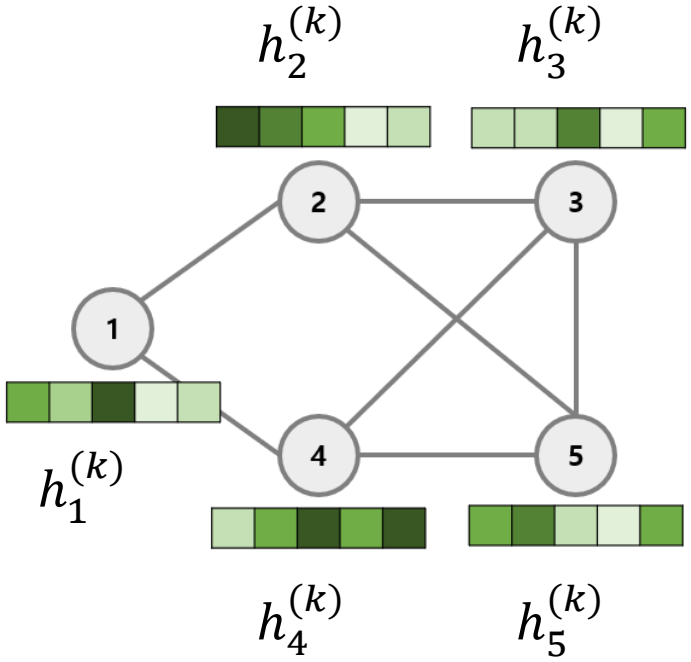
$$=$$

$$h_v^{(k)} = combine^k(a_u^{(k-1)}, h_v^{(k-1)})$$

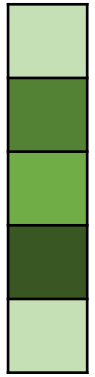
Data Mining Quality Analytics

- GNN: ③ Readout
  - K 시점의 모든 Node들의 hidden state를 결합하여 graph의 hidden state 생성
  - Graph level classification





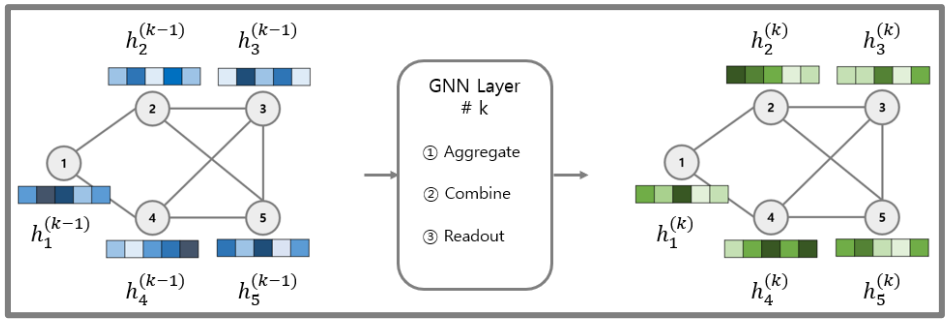
$$h_G^{(k)} = readout^k(\ h_1^{(k)}, h_2^{(k)}, h_3^{(k)}, h_4^{(k)}, h_5^{(k)}\ ) =$$

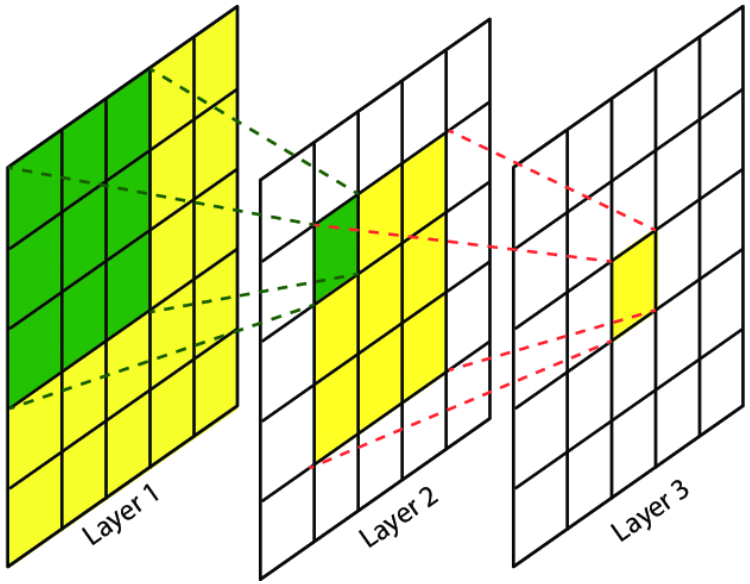


$$h_G^{(k)} = readout^k(h_v^{(k)}, \forall v \in G)$$

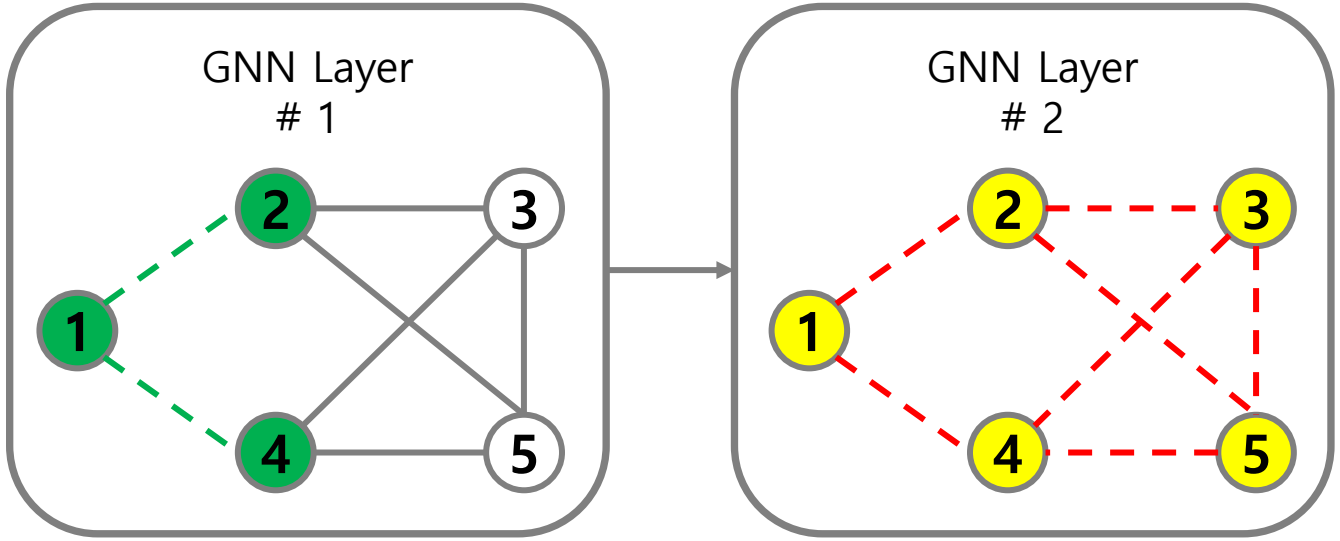
Data Mining Quality Analytics

- Stacking GNN Layer
  - CNN: Increase the receptive filed
  - GNN: Increase hop of the graph





Convolutional Neural Networks



Graph Neural Networks

- GNN: Summary
  - Aggregate
    - ✓ $a_v^{(k-1)} = aggregate^k(\{h_u^{(k-1)}, \forall u \in N(v)\})$
  - Combine
    - ✓ $h_v^{(k)} = combine^k(a_u^{(k-1)}, h_v^{(k-1)})$
  - Readout
    - ✓ For graph level task
    - ✓ $h_G^{(k)} = readout^k(h_v^{(k)}, \forall v \in G)$
  - Stacking GNN Layer
    - ✓ Increase hop of the graph

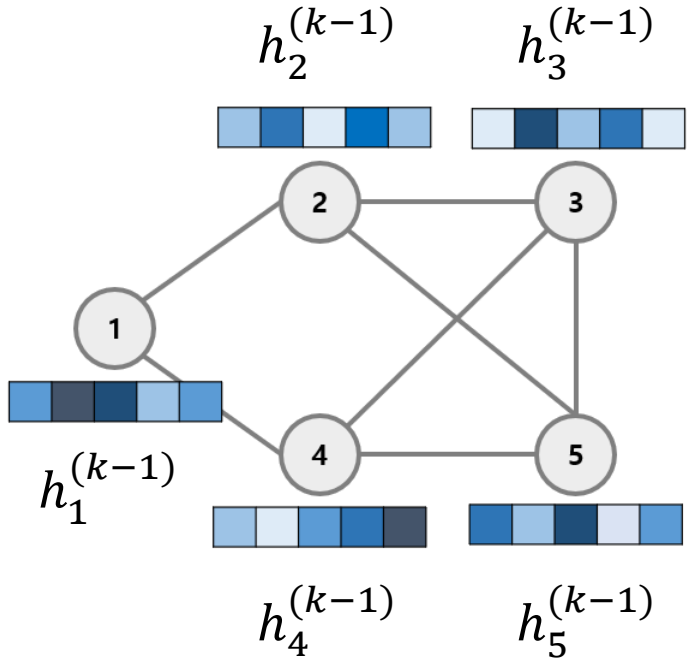
- GNN Variants
  - Aggregate / Combine function의 정의에 따라 다양한 방식의 모델이 존재함
  - Differentiable function

Name	Variant	Aggregator	Updater
Spectral Methods	ChebNet	$\mathbf{N}_k = \mathbf{T}_k(\tilde{\mathbf{L}})\mathbf{X}$	$\mathbf{H} = \sum_{k=0}^{K} \mathbf{N}_k \mathbf{\Theta}_k$
	1^{st}-order model	$\mathbf{N}_0 = \mathbf{X}$ $\mathbf{N}_1 = \mathbf{D}^{-\frac{1}{2}}\mathbf{A}\mathbf{D}^{-\frac{1}{2}}\mathbf{X}$	$\mathbf{H} = \mathbf{N}_0\mathbf{\Theta}_0 + \mathbf{N}_1\mathbf{\Theta}_1$
	Single parameter	$\mathbf{N} = (\mathbf{I}_N + \mathbf{D}^{-\frac{1}{2}}\mathbf{A}\mathbf{D}^{-\frac{1}{2}})\mathbf{X}$	$\mathbf{H} = \mathbf{N}\mathbf{\Theta}$
	GCN	$\mathbf{N} = \tilde{\mathbf{D}}^{-\frac{1}{2}}\tilde{\mathbf{A}}\tilde{\mathbf{D}}^{-\frac{1}{2}}\mathbf{X}$	$\mathbf{H} = \mathbf{N}\mathbf{\Theta}$
Non-spectral Methods	Convolutional networks in [33]	$\mathbf{h}^t_{\mathcal{N}_v} = \mathbf{h}^{t-1}_v + \sum_{k=1}^{\mathcal{N}_v} \mathbf{h}^{t-1}_k$	$\mathbf{h}^t_v = \sigma(\mathbf{h}^t_{\mathcal{N}_v}\mathbf{W}^{\mathcal{N}_v}_L)$
	DCNN	Node classification: $\mathbf{N} = \mathbf{P}^*\mathbf{X}$ Graph classification: $\mathbf{N} = 1^T_N\mathbf{P}^*\mathbf{X}/N$	$\mathbf{H} = f(\mathbf{W}^c \odot \mathbf{N})$
	GraphSAGE	$\mathbf{h}^t_{\mathcal{N}_v} = \text{AGGREGATE}_t(\{\mathbf{h}^{t-1}_u, \forall u \in \mathcal{N}_v\})$	$\mathbf{h}^t_v = \sigma(\mathbf{W}^t \cdot [\mathbf{h}^{t-1}_v \| \mathbf{h}^t_{\mathcal{N}_v}])$
Graph Attention Networks	GAT	$\alpha_{vk} = \frac{\exp(\text{LeakyReLU}(\mathbf{a}^T[\mathbf{W}\mathbf{h}_v \| \mathbf{W}\mathbf{h}_k]))}{\sum_{j\in\mathcal{N}_v}\exp(\text{LeakyReLU}(\mathbf{a}^T[\mathbf{W}\mathbf{h}_v \| \mathbf{W}\mathbf{h}_j]))}$ $\mathbf{h}^t_{\mathcal{N}_v} = \sigma\left(\sum_{k\in\mathcal{N}_v}\alpha_{vk}\mathbf{W}\mathbf{h}_k\right)$ Multi-head concatenation: $\mathbf{h}^t_{\mathcal{N}_v} = \Big\|_{m=1}^{M}\sigma\left(\sum_{k\in\mathcal{N}_v}\alpha^m_{vk}\mathbf{W}^m\mathbf{h}_k\right)$ Multi-head average: $\mathbf{h}^t_{\mathcal{N}_v} = \sigma\left(\frac{1}{M}\sum_{m=1}^{M}\sum_{k\in\mathcal{N}_v}\alpha^m_{vk}\mathbf{W}^m\mathbf{h}_k\right)$	$\mathbf{h}^t_v = \mathbf{h}^t_{\mathcal{N}_v}$
Gated Graph Neural Networks	GGNN	$\mathbf{h}^t_{\mathcal{N}_v} = \sum_{k\in\mathcal{N}_v}\mathbf{h}^{t-1}_k + \mathbf{b}$	$\mathbf{z}^t_v = \sigma(\mathbf{W}^z\mathbf{h}^t_{\mathcal{N}_v} + \mathbf{U}^z\mathbf{h}^{t-1}_v)$ $\mathbf{r}^t_v = \sigma(\mathbf{W}^r\mathbf{h}^t_{\mathcal{N}_v} + \mathbf{U}^r\mathbf{h}^{t-1}_v)$ $\widetilde{\mathbf{h}^t_v} = \tanh(\mathbf{W}\mathbf{h}^t_{\mathcal{N}_v} + \mathbf{U}(\mathbf{r}^t_v \odot \mathbf{h}^{t-1}_v))$ $\mathbf{h}^t_v = (1-\mathbf{z}^t_v)\odot\mathbf{h}^{t-1}_v + \mathbf{z}^t_v \odot \widetilde{\mathbf{h}^t_v}$
Graph LSTM	Tree LSTM (Child sum)	$\mathbf{h}^t_{\mathcal{N}_v} = \sum_{k\in\mathcal{N}_v}\mathbf{h}^{t-1}_k$	$\mathbf{i}^t_v = \sigma(\mathbf{W}^i\mathbf{x}^t_v + \mathbf{U}^i\mathbf{h}^t_{\mathcal{N}_v} + \mathbf{b}^i)$ $\mathbf{f}^t_{vk} = \sigma\left(\mathbf{W}^f\mathbf{x}^t_v + \mathbf{U}^f\mathbf{h}^{t-1}_k + \mathbf{b}^f\right)$ $\mathbf{o}^t_v = \sigma(\mathbf{W}^o\mathbf{x}^t_v + \mathbf{U}^o\mathbf{h}^t_{\mathcal{N}_v} + \mathbf{b}^o)$ $\mathbf{u}^t_v = \tanh(\mathbf{W}^u\mathbf{x}^t_v + \mathbf{U}^u\mathbf{h}^t_{\mathcal{N}_v} + \mathbf{b}^u)$ $\mathbf{c}^t_v = \mathbf{i}^t_v \odot \mathbf{u}^t_v + \sum_{k\in\mathcal{N}_v}\mathbf{f}^t_{vk}\odot\mathbf{c}^{t-1}_k$ $\mathbf{h}^t_v = \mathbf{o}^t_v \odot \tanh(\mathbf{c}^t_v)$
	Tree LSTM (N-ary)	$\mathbf{h}^{ti}_{\mathcal{N}_v} = \sum_{l=1}^{K}\mathbf{U}^i_l\mathbf{h}^{t-1}_{vl}$ $\mathbf{h}^{tf}_{\mathcal{N}_v k} = \sum_{l=1}^{K}\mathbf{U}^f_{kl}\mathbf{h}^{t-1}_{vl}$ $\mathbf{h}^{to}_{\mathcal{N}_v} = \sum_{l=1}^{K}\mathbf{U}^o_l\mathbf{h}^{t-1}_{vl}$ $\mathbf{h}^{tu}_{\mathcal{N}_v} = \sum_{l=1}^{K}\mathbf{U}^u_l\mathbf{h}^{t-1}_{vl}$	$\mathbf{i}^t_v = \sigma(\mathbf{W}^i\mathbf{x}^t_v + \mathbf{h}^{ti}_{\mathcal{N}_v} + \mathbf{b}^i)$ $\mathbf{f}^t_{vk} = \sigma(\mathbf{W}^f\mathbf{x}^t_v + \mathbf{h}^{tf}_{\mathcal{N}_v k} + \mathbf{b}^f)$ $\mathbf{o}^t_v = \sigma(\mathbf{W}^o\mathbf{x}^t_v + \mathbf{h}^{to}_{\mathcal{N}_v} + \mathbf{b}^o)$ $\mathbf{u}^t_v = \tanh(\mathbf{W}^u\mathbf{x}^t_v + \mathbf{h}^{tu}_{\mathcal{N}_v} + \mathbf{b}^u)$ $\mathbf{c}^t_v = \mathbf{i}^t_v \odot \mathbf{u}^t_v + \sum_{l=1}^{K}\mathbf{f}^t_{vl}\odot\mathbf{c}^{t-1}_{vl}$ $\mathbf{h}^t_v = \mathbf{o}^t_v \odot \tanh(\mathbf{c}^t_v)$
	Graph LSTM in [34]	$\mathbf{h}^{ti}_{\mathcal{N}_v} = \sum_{k\in\mathcal{N}_v}\mathbf{U}^i_{m(v,k)}\mathbf{h}^{t-1}_k$ $\mathbf{h}^{to}_{\mathcal{N}_v} = \sum_{k\in\mathcal{N}_v}\mathbf{U}^o_{m(v,k)}\mathbf{h}^{t-1}_k$ $\mathbf{h}^{tu}_{\mathcal{N}_v} = \sum_{k\in\mathcal{N}_v}\mathbf{U}^u_{m(v,k)}\mathbf{h}^{t-1}_k$	$\mathbf{i}^t_v = \sigma(\mathbf{W}^i\mathbf{x}^t_v + \mathbf{h}^{ti}_{\mathcal{N}_v} + \mathbf{b}^i)$ $\mathbf{f}^t_{vk} = \sigma(\mathbf{W}^f\mathbf{x}^t_v + \mathbf{U}^f_{m(v,k)}\mathbf{h}^{t-1}_k + \mathbf{b}^f)$ $\mathbf{o}^t_v = \sigma(\mathbf{W}^o\mathbf{x}^t_v + \mathbf{h}^{to}_{\mathcal{N}_v} + \mathbf{b}^o)$ $\mathbf{u}^t_v = \tanh(\mathbf{W}^u\mathbf{x}^t_v + \mathbf{h}^{tu}_{\mathcal{N}_v} + \mathbf{b}^u)$ $\mathbf{c}^t_v = \mathbf{i}^t_v \odot \mathbf{u}^t_v + \sum_{k\in\mathcal{N}_v}\mathbf{f}^t_{vk}\odot\mathbf{c}^{t-1}_k$ $\mathbf{h}^t_v = \mathbf{o}^t_v \odot \tanh(\mathbf{c}^t_v)$

Zhou, Jie, et al. "Graph neural networks: A review of methods and applications." *arXiv preprint arXiv:1812.08434* (2018).

Data Mining Quality Analytics

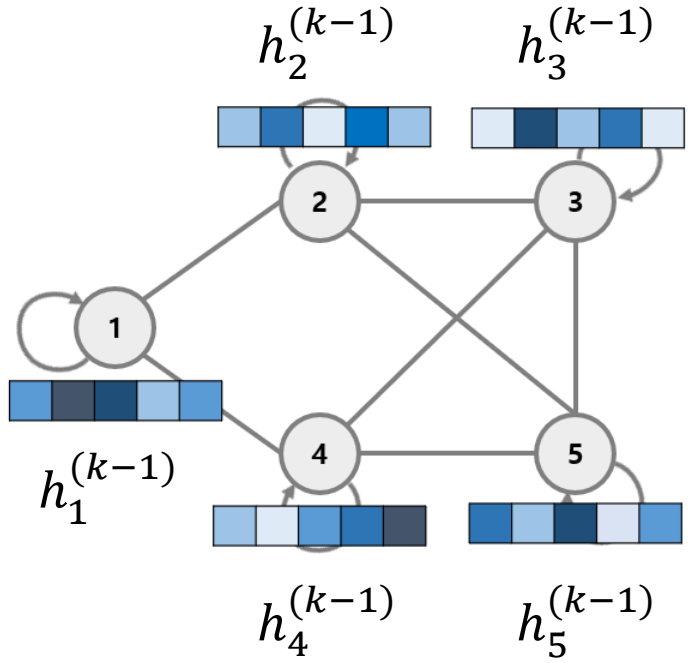
- GNN
  - Aggregate
    - ✓ $a_v^{(k-1)} = \sum_{u \in N(v)} h_u^{(k-1)}$
  - Combine
    - ✓ $h_v^{(k)} = Relu\left(W_{self} h_v^{(k-1)} + W_{neigh} a_v^{(k-1)}\right)$
  - Matrix form
    - ✓ $H^{(t)} = Relu(H^{(t-1)} W_{self} + A H^{(t-1)} W_{neigh} a_v^{(k-1)})$



F. Scarselli, M. Gori, A. C. Tsoi, M. Hagenbuchner, and G. Monfardini, "The graph neural network model," IEEE TNN 2009, vol. 20, no. 1, pp. 61–80, 2009.

Data Mining Quality Analytics

- GNN (Self-loop)
  - When aggregating, self information is needed
  - $W_{self} = W_{neigh} = W$
  - Aggregate
    - ✓ $a_v^{(k-1)} = \sum_{u \in N(v)} h_u^{(k-1)} + h_v^{(k-1)}$
  - Combine
    - ✓ $h_v^{(k)} = Relu(Wa_v^{(k-1)})$
    - ✓ $h_v^{(k)} = Relu(W \sum_{u \in N(v) \cup \{v\}} h_u^{(k-1)})$
  - Matrix form
    - ✓ $H^{(t)} = Relu((A+I)H^{(t-1)}W)$



F. Scarselli, M. Gori, A. C. Tsoi, M. Hagenbuchner, and G. Monfardini, "The graph neural network model," IEEE TNN 2009, vol. 20, no. 1, pp. 61–80, 2009.

Data Mining Quality Analytics

- Graph Convolutional Networks (2016)
  - Amsterdam Univ., CIFAR
  - If graph size is too large
    - ✓ Unstable and sensitive to node degrees
    - ✓ Degree = # of neighbor nodes for each node
  - Normalized aggregate function
    - ✓ $a_v^{(k-1)} = \sum_{u \in N(v) \cup \{v\}} \frac{h_u^{(k-1)}}{\sqrt{|N(v)||N(u)|}}$
  - Combine
    - ✓ $h_v^{(k)} = Relu(\mathrm{W} a_v^{(k-1)})$
  - Matrix form
    - ✓ $H^{(t)} = Relu((D^{-\frac{1}{2}}(A+I)\,D^{-\frac{1}{2}}H^{(t-1)}W)$

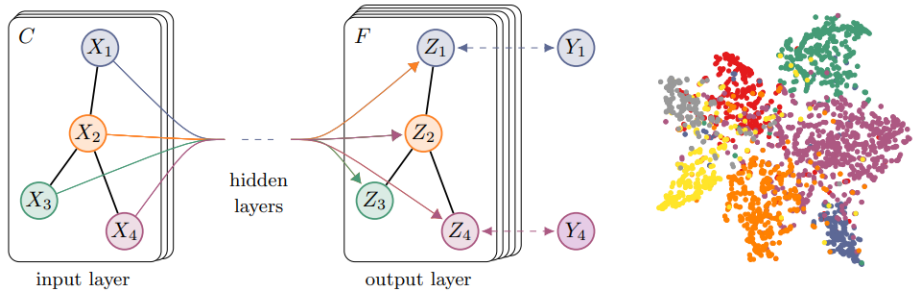
SEMI-SUPERVISED CLASSIFICATION WITH GRAPH CONVOLUTIONAL NETWORKS

Thomas N. Kipf
University of Amsterdam
T.N.Kipf@uva.nl

Max Welling
University of Amsterdam
Canadian Institute for Advanced Research (CIFAR)
M.Welling@uva.nl

ABSTRACT

We present a scalable approach for semi-supervised learning on graph-structured data that is based on an efficient variant of convolutional neural networks which operate directly on graphs. We motivate the choice of our convolutional architecture via a localized first-order approximation of spectral graph convolutions. Our model scales linearly in the number of graph edges and learns hidden layer representations that encode both local graph structure and features of nodes. In a number of experiments on citation networks and on a knowledge graph dataset we demonstrate that our approach outperforms related methods by a significant margin.



(a) Graph Convolutional Network

(b) Hidden layer activations

Kipf, Thomas N., and Max Welling. "Semi-supervised classification with graph convolutional networks." *arXiv preprint arXiv:1609.02907* (2016).

- Gated Graph Neural Networks (2016)
  - Stacking deep layers lead overfitting / vanishing gradient
  - ICLR, Toronto Univ., Microsoft
  - Aggregate
    - ✓ $a_v^{(k-1)} = \sum_{u \in N(v)} \sum h_u^{(k-1)}$
  - Combine
    - ✓ $h_v^{(k)} = GRU(h_v^{(k-1)}, a_v^{(k-1)})$
  - Matrix form
    - ✓ $H^{(t)} = GRU(((A+I)W,\ H^{(t-1)})$

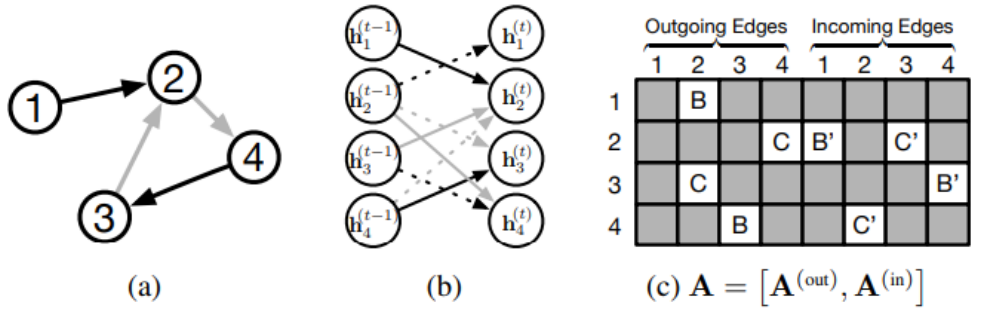
GATED GRAPH SEQUENCE NEURAL NETWORKS

Yujia Li* **& Richard Zemel**
Department of Computer Science, University of Toronto
Toronto, Canada
{yujiali,zemel}@cs.toronto.edu

Marc Brockschmidt & Daniel Tarlow
Microsoft Research
Cambridge, UK
{mabrocks,dtarlow}@microsoft.com

ABSTRACT

Graph-structured data appears frequently in domains including chemistry, natural language semantics, social networks, and knowledge bases. In this work, we study feature learning techniques for graph-structured inputs. Our starting point is previous work on Graph Neural Networks (Scarselli et al., 2009), which we modify to use gated recurrent units and modern optimization techniques and then extend to output sequences. The result is a flexible and broadly useful class of neural net-



Li, Yujia, et al. "Gated graph sequence neural networks." *arXiv preprint arXiv:1511.05493* (2015).

- GraphSAGE (2017)
  - NIPS, Stanford Univ.
  - Consider node importance or ordering
  - Aggregate
    - Mean aggregate
      - ✓ $a_v^{(k-1)} = \sum_{u \in N(v)} \frac{h_u^{(k-1)}}{|N(u)|}$
    - LSTM aggregate (Random permutation of neighbors)
      - ✓ $a_v^{(k-1)} = LSTM(\sum\{W_{agg} h_u^{(k-1)}, \forall u \in N(v)\})$
    - Pooling aggregate
      - ✓ $a_v^{(k-1)} = Pool(\{W_{pool} h_u^{(k-1)}, \forall u \in N(v)\})$
      - ✓ $Pool = element\text{-}wise\ mean\ or\ max$
  - Combine
    - ✓ $h_v^{(k)} = Relu(W[h_v^{(k-1)}, a_v^{(k-1)}])$
  - Residual connection / Skip connection

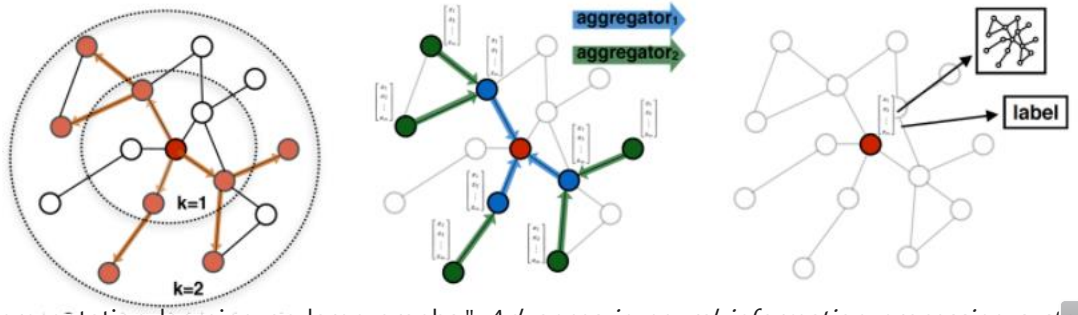
Inductive Representation Learning on Large Graphs

William L. Hamilton* wleif@stanford.edu
Rex Ying* rexying@stanford.edu
Jure Leskovec jure@cs.stanford.edu

Department of Computer Science
Stanford University
Stanford, CA, 94305

Abstract

Low-dimensional embeddings of nodes in large graphs have proved extremely useful in a variety of prediction tasks, from content recommendation to identifying protein functions. However, most existing approaches require that all nodes in the graph are present during training of the embeddings; these previous approaches are inherently *transductive* and do not naturally generalize to unseen nodes. Here we

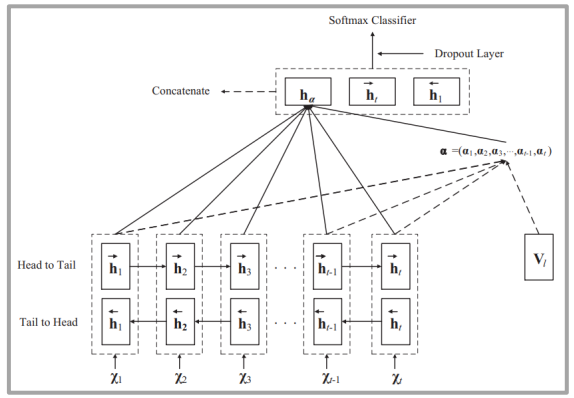


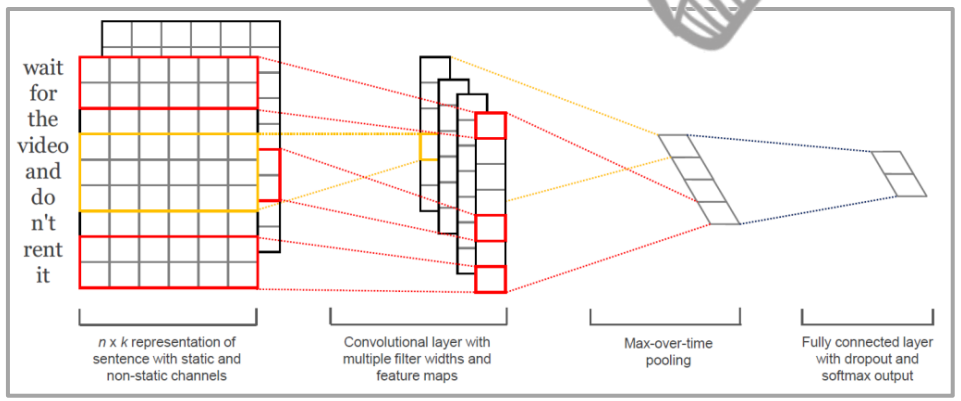
Hamilton, Will, Zhitao Ying, and Jure Leskovec. "Inductive representation learning on large graphs." *Advances in neural information processing systems*. 2017.

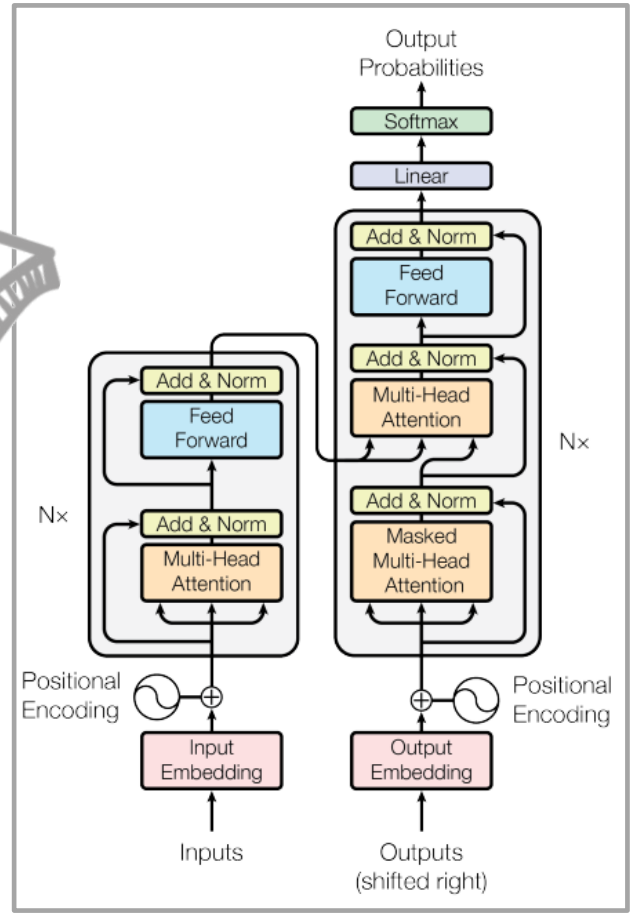
Data Mining Quality Analytics

- GNN Variants
  - Challenges
    - ✓ Self-loop (Vanilla GNN)
    - ✓ Node degrees (GCN)
    - ✓ Node importance (GraphSAGE)
    - ✓ Overfitting / Vanishing gradient (GGNN)

Attention is All You Need!

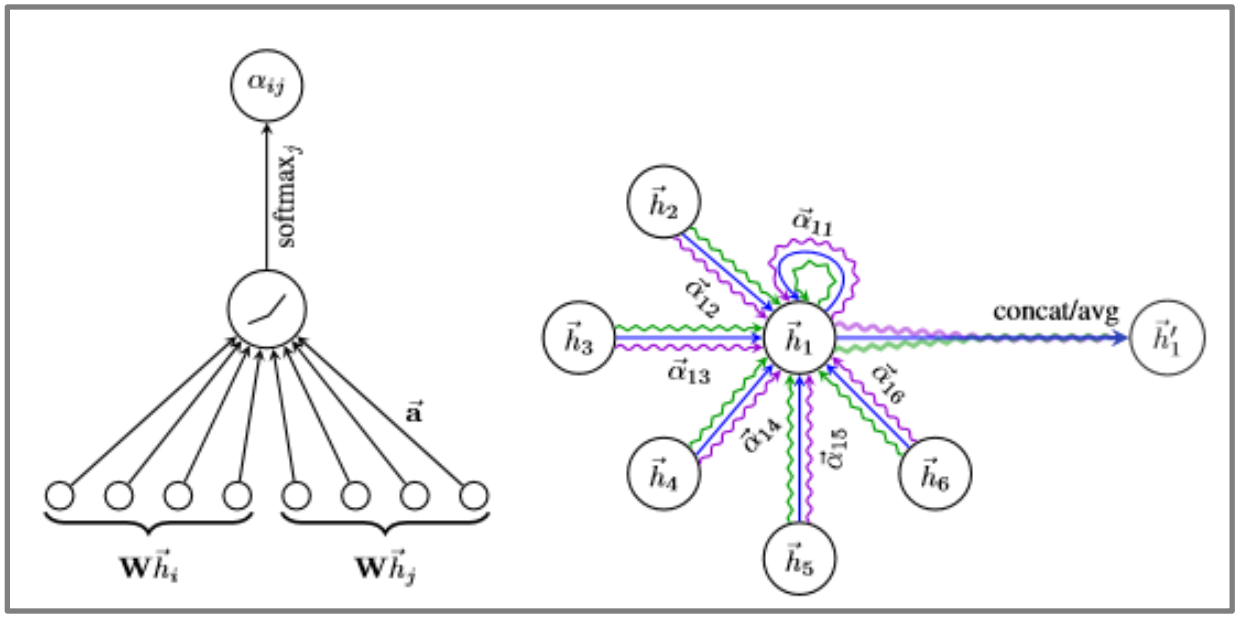






Data Mining Quality Analytics

- Graph Attention Networks: Paper
  - ICLR 2018
  - Cambridge University
  - Veličković, Petar, et al.
  - Recite: 1951



GRAPH ATTENTION NETWORKS

Petar Veličković*
Department of Computer Science and Technology
University of Cambridge
petar.velickovic@cst.cam.ac.uk

Guillem Cucurull*
Centre de Visió per Computador, UAB
gcucurull@gmail.com

Arantxa Casanova*
Centre de Visió per Computador, UAB
ar.casanova.8@gmail.com

Adriana Romero
Montréal Institute for Learning Algorithms
adriana.romero.soriano@umontreal.ca

Pietro Liò
Department of Computer Science and Technology
University of Cambridge
pietro.lio@cst.cam.ac.uk

Yoshua Bengio
Montréal Institute for Learning Algorithms
yoshua.umontreal@gmail.com

ABSTRACT

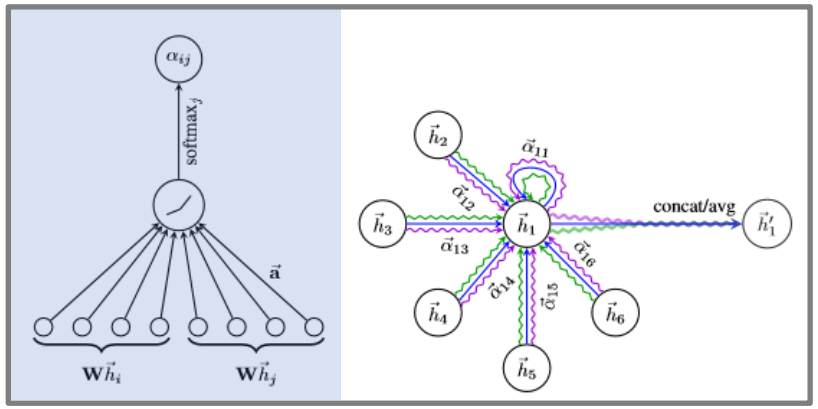
We present graph attention networks (GATs), novel neural network architectures that operate on graph-structured data, leveraging masked self-attentional layers to address the shortcomings of prior methods based on graph convolutions or their approximations. By stacking layers in which nodes are able to attend over their neighborhoods' features, we enable (implicitly) specifying different weights to different nodes in a neighborhood, without requiring any kind of costly matrix operation (such as inversion) or depending on knowing the graph structure upfront. In this way, we address several key challenges of spectral-based graph neural networks simultaneously, and make our model readily applicable to inductive as well as transductive problems. Our GAT models have achieved or matched state-of-the-art results across four established transductive and inductive graph benchmarks: the *Cora*, *Citeseer* and *Pubmed* citation network datasets, as well as a *protein-protein interaction* dataset (wherein test graphs remain unseen during training).

Veličković, Petar, et al. "Graph attention networks." *arXiv preprint arXiv:1710.10903* (2017).

Data Mining Quality Analytics

- Model Architecture
  - Aggregate

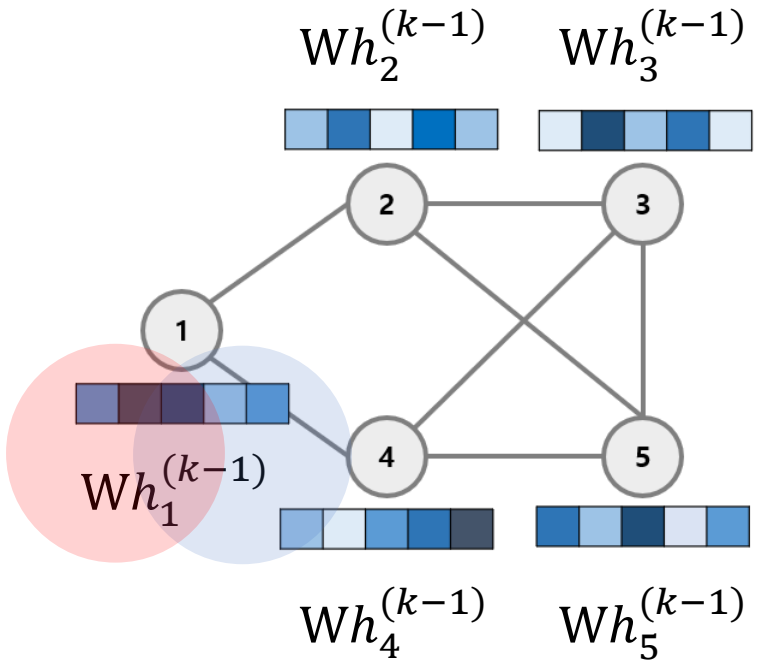
$$A(q, K, V) = \sum_{i} softmax(f(K, q))\, V$$

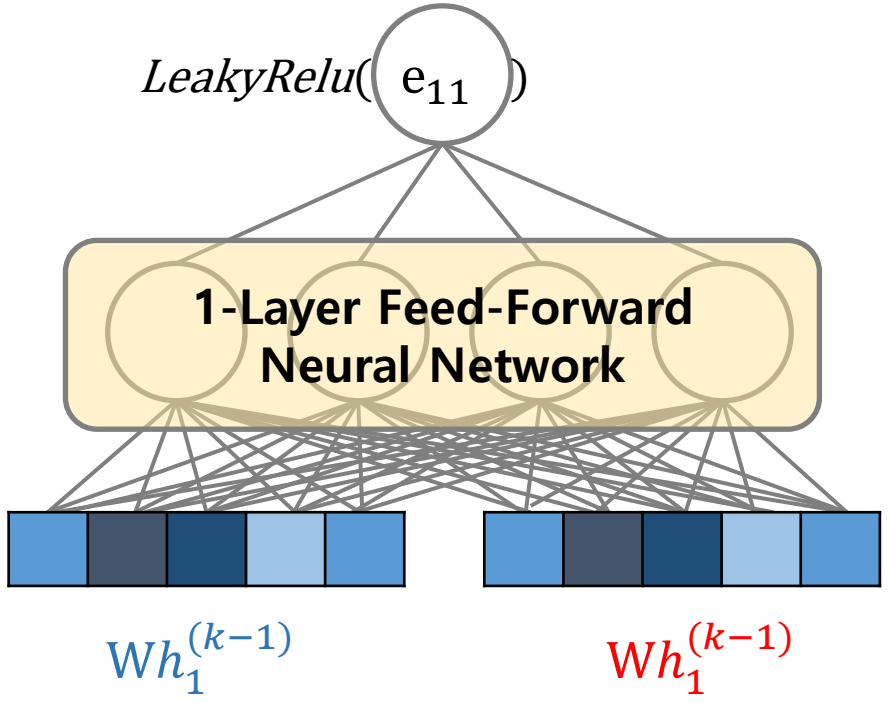


✓ $a_v^{(k-1)} = Attention(\{ h_u^{(k-1)},\ u \in N(v) \cup \{v\} \})$

✓ Key = Query = Value = $h_u^{(k-1)}$

✓ $Similarity\ Funtion(f) = 1 - layer\ Feed - Forward\ NN$



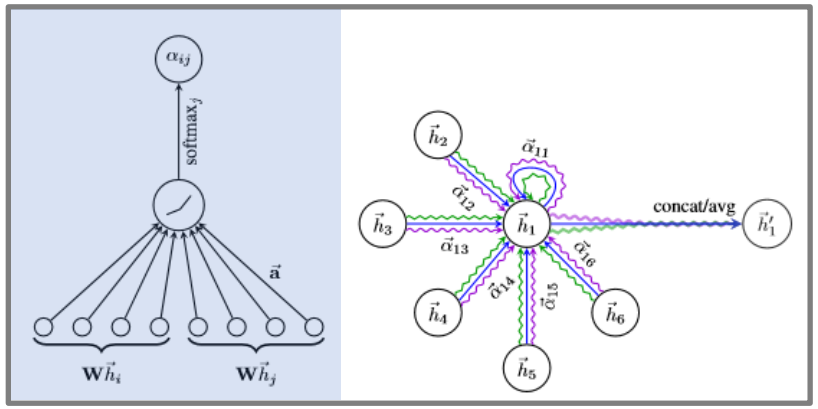


Veličković, Petar, et al. "Graph attention networks." *arXiv preprint arXiv:1710.10903* (2017).

Data Mining Quality Analytics

- Model Architecture
  - Aggregate

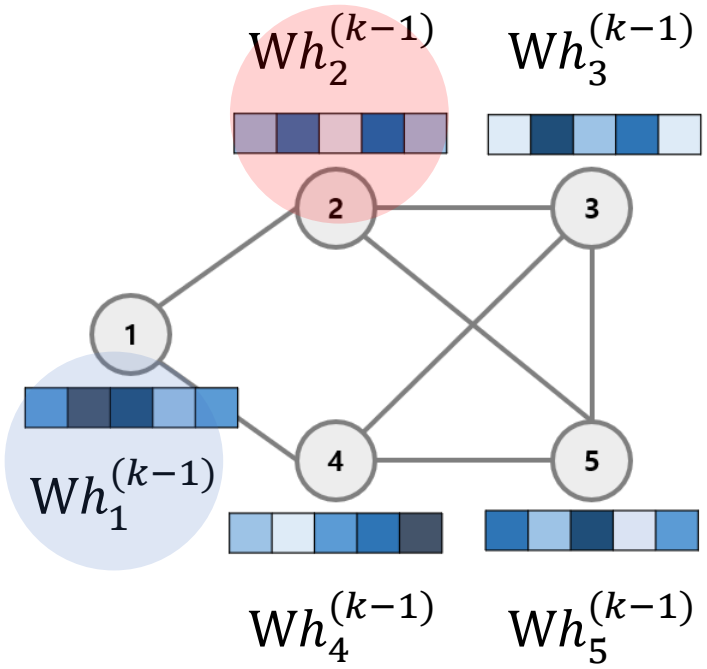
$$A(q, K, V) = \sum_{i} softmax(f(K, q))\, V$$

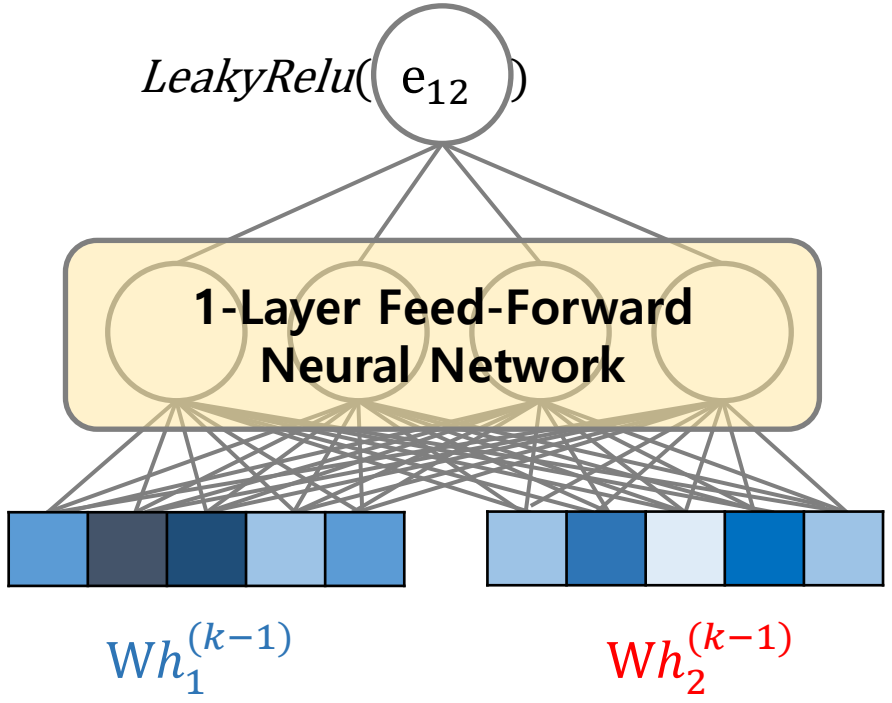


✓ $a_v^{(k-1)} = Attention(\{ h_u^{(k-1)},\ u \in N(v) \cup \{v\} \})$

✓ Key = Query = Value = $h_u^{(k-1)}$

✓ $Similarity\ Funtion(f) = 1 - layer\ Feed - Forward\ NN$



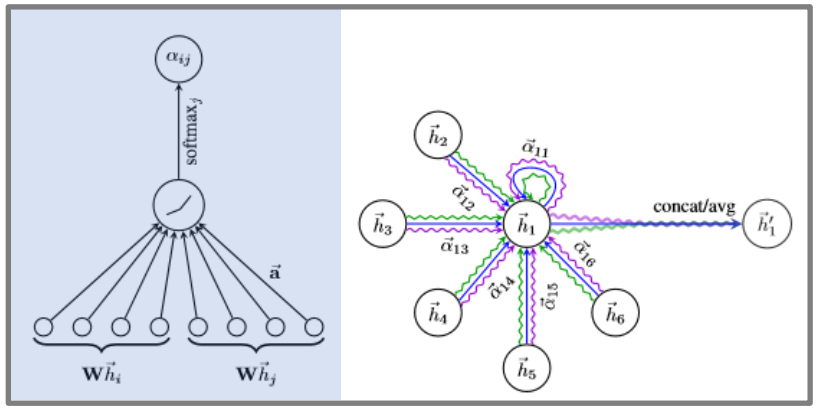


Veličković, Petar, et al. "Graph attention networks." *arXiv preprint arXiv:1710.10903* (2017).

Data Mining Quality Analytics

- Model Architecture
  - Aggregate

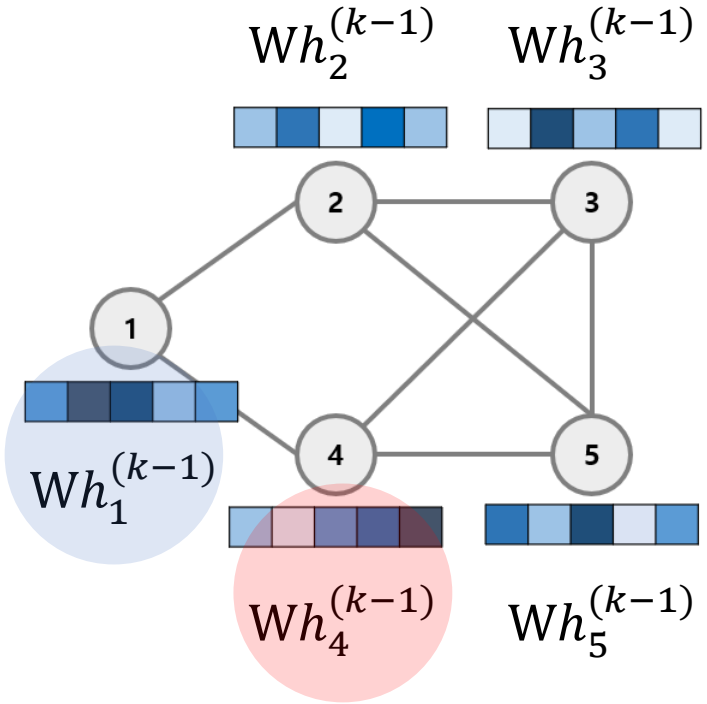
$$A(q, K, V) = \sum_{i} softmax(f(K, q))V$$

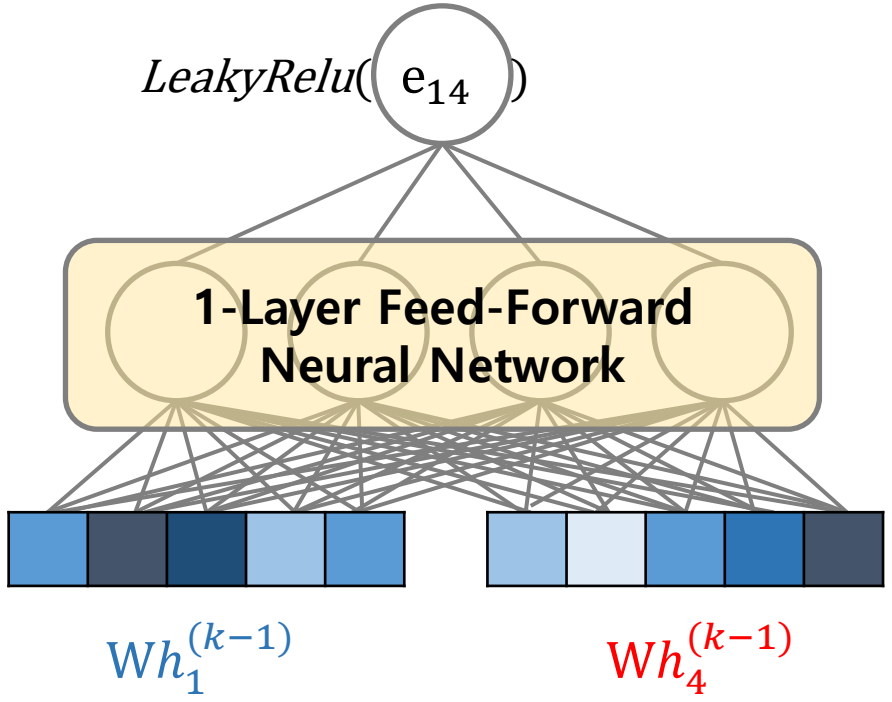


✓ $a_v^{(k-1)} = Attention(\{h_u^{(k-1)}, u \in N(v) \cup \{v\}\})$

✓ Key = Query = Value = $h_u^{(k-1)}$

✓ $Similarity\ Funtion(f) = 1 - layer\ Feed - Forward\ NN$



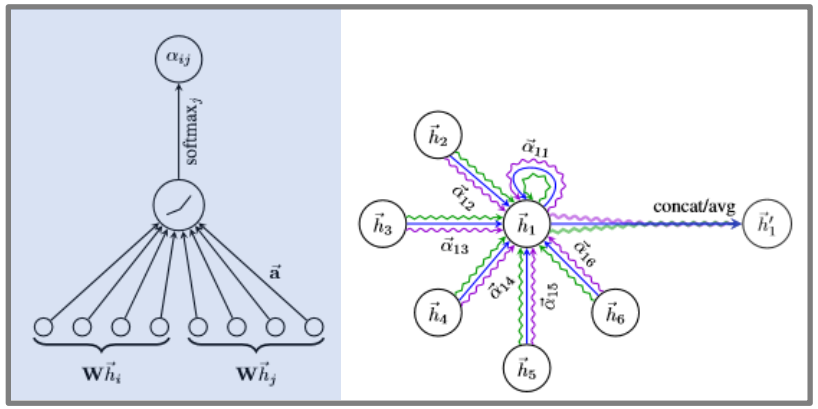


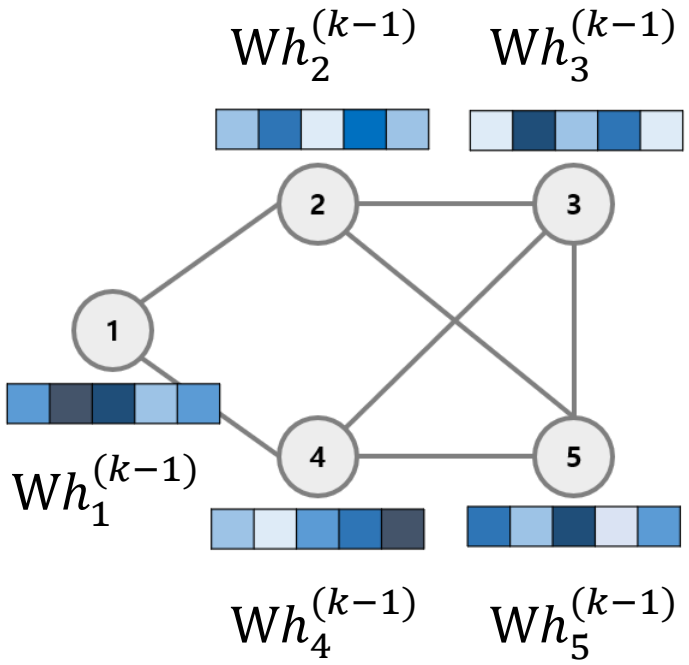
Veličković, Petar, et al. "Graph attention networks." *arXiv preprint arXiv:1710.10903* (2017).

Data Mining Quality Analytics

- Model Architecture
  - Aggregate
    - ✓ $a_v^{(k-1)} = Attention(\{ h_u^{(k-1)}, u \in N(v) \cup \{v\}\})$

$$A(q, K, V) = \sum_i softmax(f(K, q))V$$





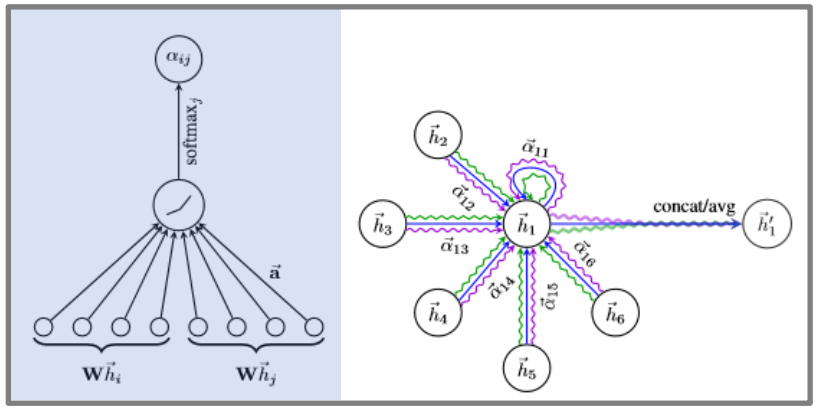
e_{11}	e_{12}		e_{14}	
e_{21}	e_{22}	e_{23}		e_{25}
	e_{32}	e_{33}	e_{34}	e_{35}
e_{41}		e_{43}	e_{44}	e_{45}
	e_{25}	e_{53}	e_{54}	e_{55}

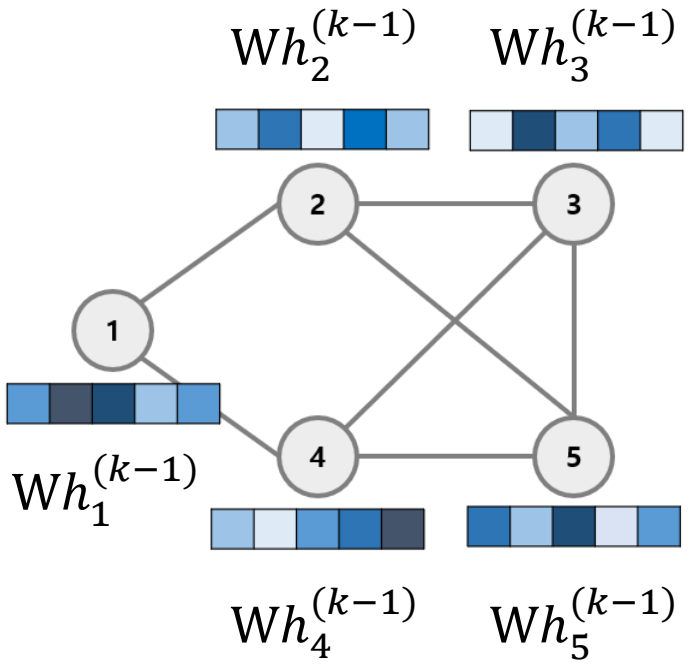
Veličković, Petar, et al. "Graph attention networks." *arXiv preprint arXiv:1710.10903* (2017).

Data Mining Quality Analytics

- Model Architecture
  - Aggregate
    - ✓ $a_v^{(k-1)} = Attention(\{ h_u^{(k-1)}, u \in N(v) \cup \{v\}\})$

$$A(q, K, V) = \sum_i softmax(f(K, q))V$$





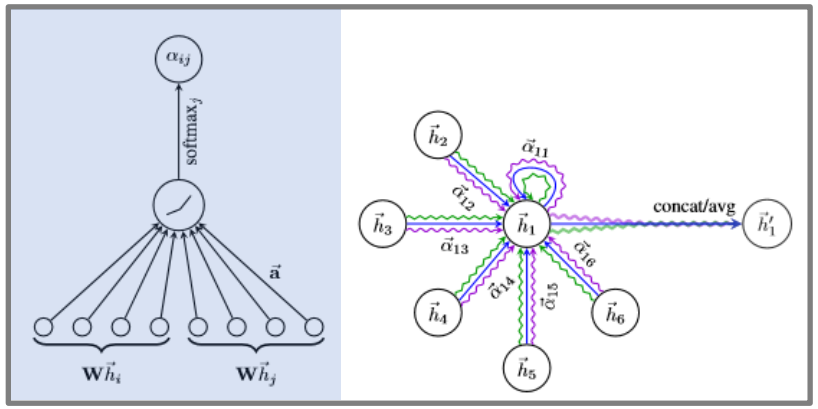
e_{11}	e_{12}	Softmax	e_{14}	
e_{21}	e_{22}	Softmax		e_{25}
	e_{32}	Softmax	e_{34}	e_{35}
e_{41}		Softmax	e_{44}	e_{45}
	e_{25}	Softmax	e_{54}	e_{55}

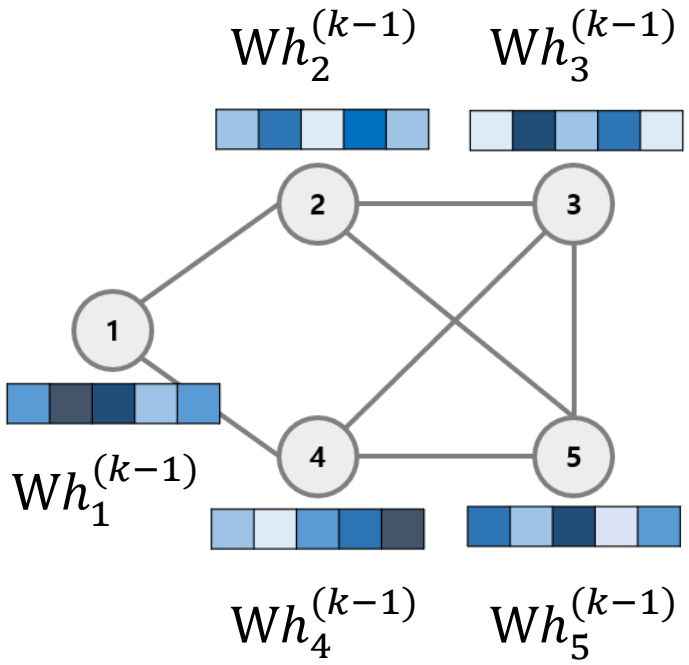
Veličković, Petar, et al. "Graph attention networks." *arXiv preprint arXiv:1710.10903* (2017).

Data Mining Quality Analytics

- Model Architecture
  - Aggregate
    - ✓ $a_v^{(k-1)} = Attention(\{ h_u^{(k-1)}, u \in N(v) \cup \{v\}\})$

$$A(q, K, V) = \sum_i softmax(f(K, q))V$$





a_{11}	a_{12}		a_{14}	
a_{21}	a_{22}	a_{23}		a_{25}
	a_{32}	a_{33}	a_{34}	a_{35}
a_{41}		a_{43}	a_{44}	a_{45}
	a_{25}	a_{53}	a_{54}	a_{55}

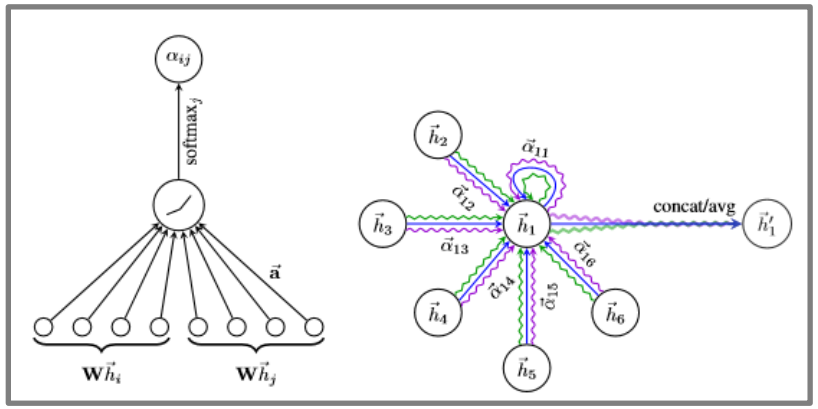
Veličković, Petar, et al. "Graph attention networks." *arXiv preprint arXiv:1710.10903* (2017).

Data Mining Quality Analytics

- Model Architecture

$$A(q, K, V) = \sum_{i} softmax(f(K, q))V$$

  - Aggregate
    - ✓ $a_v^{(k-1)} = Attention(\{ h_u^{(k-1)}, u \in N(v) \cup \{v\}\})$
  - Masked Attention
    - ✓ Using adjacency matrix, masked value with negative values before softmax.



e_{11}	e_{12}	e_{13}	e_{14}	e_{15}
e_{21}	e_{22}	e_{23}	e_{24}	e_{25}
e_{31}	e_{32}	e_{33}	e_{34}	e_{35}
e_{41}	e_{42}	e_{43}	e_{44}	e_{45}
e_{51}	e_{52}	e_{53}	e_{54}	e_{55}

Similarity

e_{11}	e_{12}	$-9e^{10}$	e_{14}	$-9e^{10}$
e_{21}	e_{22}	e_{23}	$-9e^{10}$	e_{25}
$-9e^{10}$	e_{32}	e_{33}	e_{34}	e_{35}
e_{41}	$-9e^{10}$	e_{43}	e_{44}	e_{45}
$-9e^{10}$	e_{52}	e_{53}	e_{54}	e_{55}

Masking

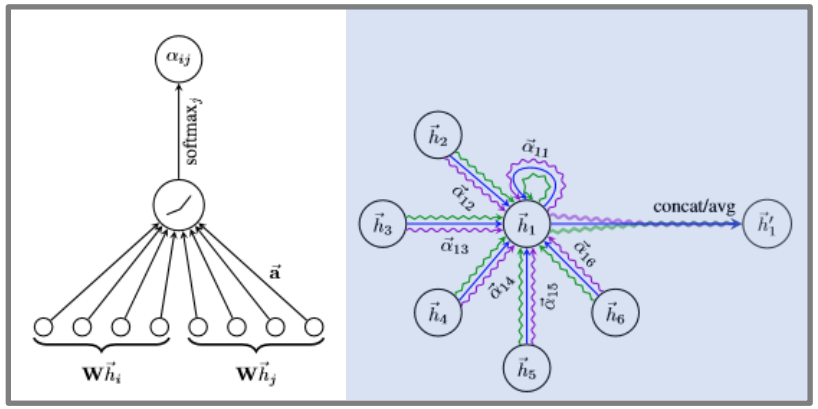
a_{11}	a_{12}	0	a_{14}	0
a_{21}	a_{22}	a_{23}	0	a_{25}
0	a_{32}	a_{33}	a_{34}	a_{35}
a_{41}	0	a_{43}	a_{44}	a_{45}
0	a_{52}	a_{53}	a_{54}	a_{55}

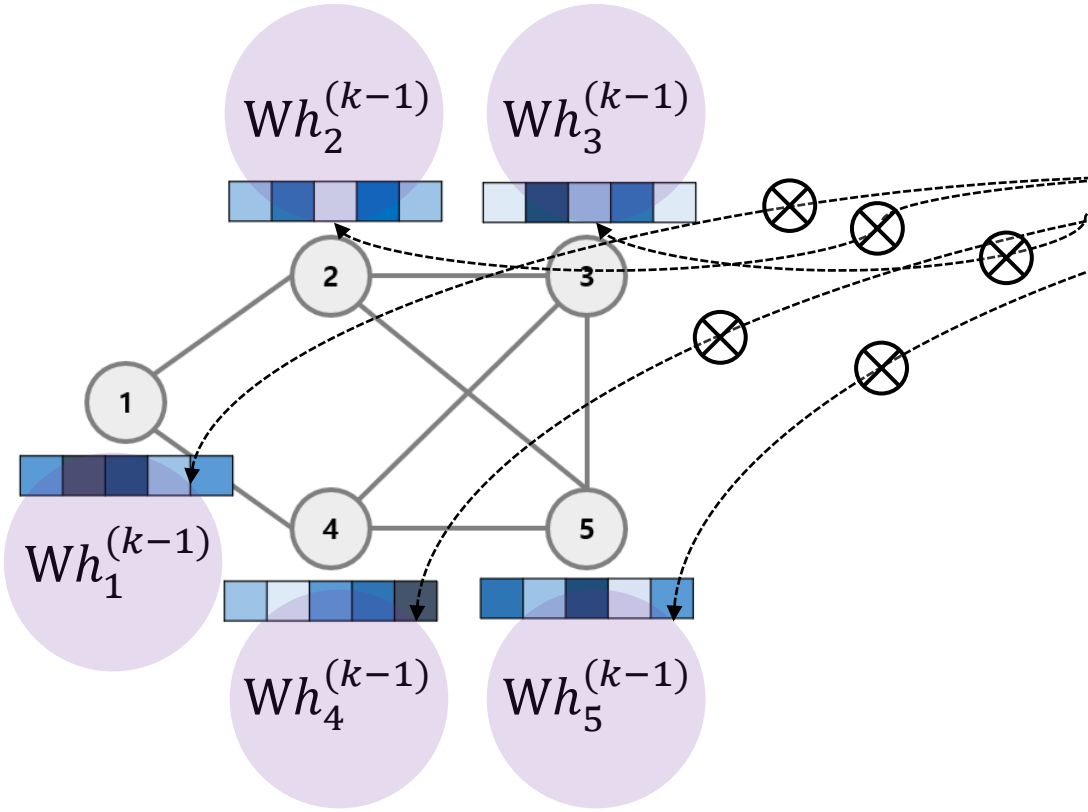
Softmax

Data Mining Quality Analytics

- Model Architecture
  - Combine
    - ✓ $h_v^{(k)} = \sum_{u \in N(v) \cup \{v\}} a_u^{(k-1)} h_u^{(k-1)}$
    - ✓ Weight-sum of aggregated information & k-1 hidden state (Value)

$$A(q, K, V) = \sum_{i} softmax(f(K, q))V$$





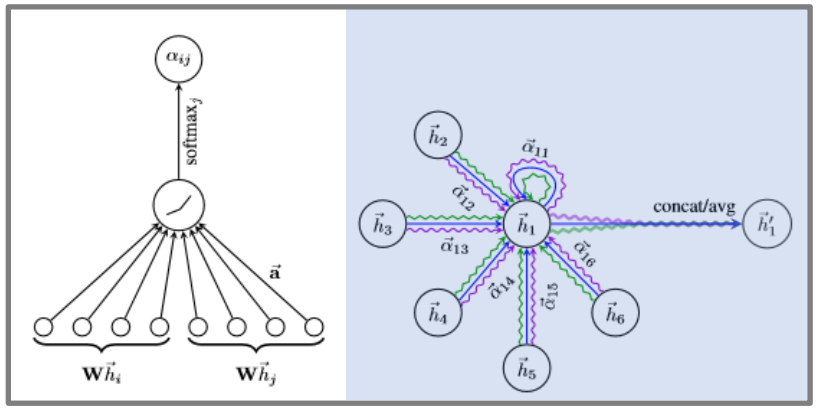
a_{11}	a_{12}	0	a_{14}	0
a_{21}	a_{22}	a_{23}	0	a_{25}
0	a_{32}	a_{33}	a_{34}	a_{35}
a_{41}	0	a_{43}	a_{44}	a_{45}
0	a_{25}	a_{53}	a_{54}	a_{55}

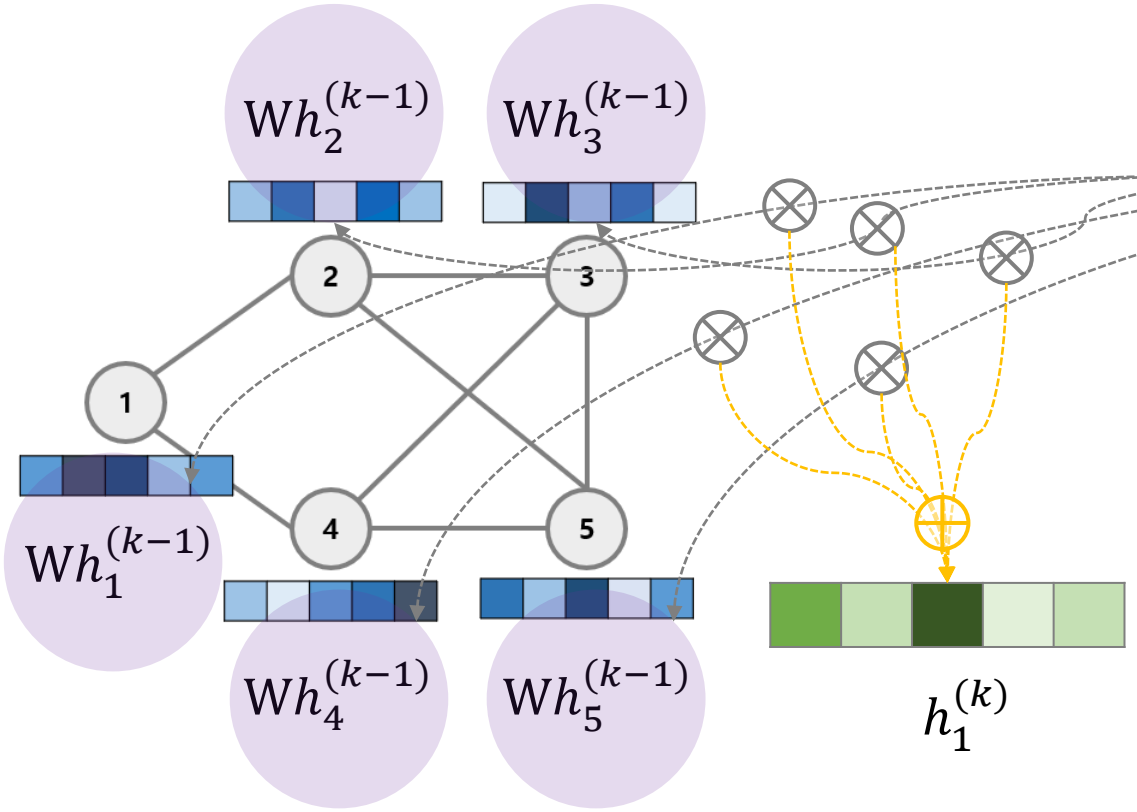
Veličković, Petar, et al. "Graph attention networks." *arXiv preprint arXiv:1710.10903* (2017).

Data Mining Quality Analytics

- Model Architecture
  - Combine
    - ✓ $h_v^{(k)} = \sum_{u \in N(v) \cup \{v\}} a_u^{(k-1)} h_u^{(k-1)}$
    - ✓ Weight-sum of aggregated information & k-1 hidden state (Value)

$$A(q, K, V) = \sum_i softmax(f(K, q))V$$





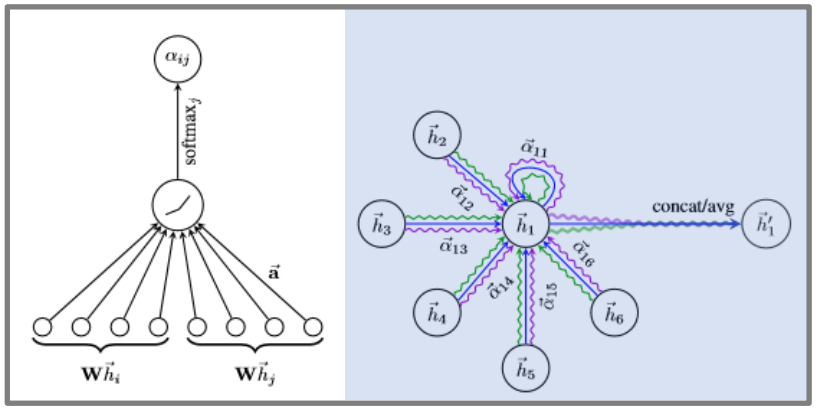
a_{11}	a_{12}	0	a_{14}	0
a_{21}	a_{22}	a_{23}	0	a_{25}
0	a_{32}	a_{33}	a_{34}	a_{35}
a_{41}	0	a_{43}	a_{44}	a_{45}
0	a_{25}	a_{53}	a_{54}	a_{55}

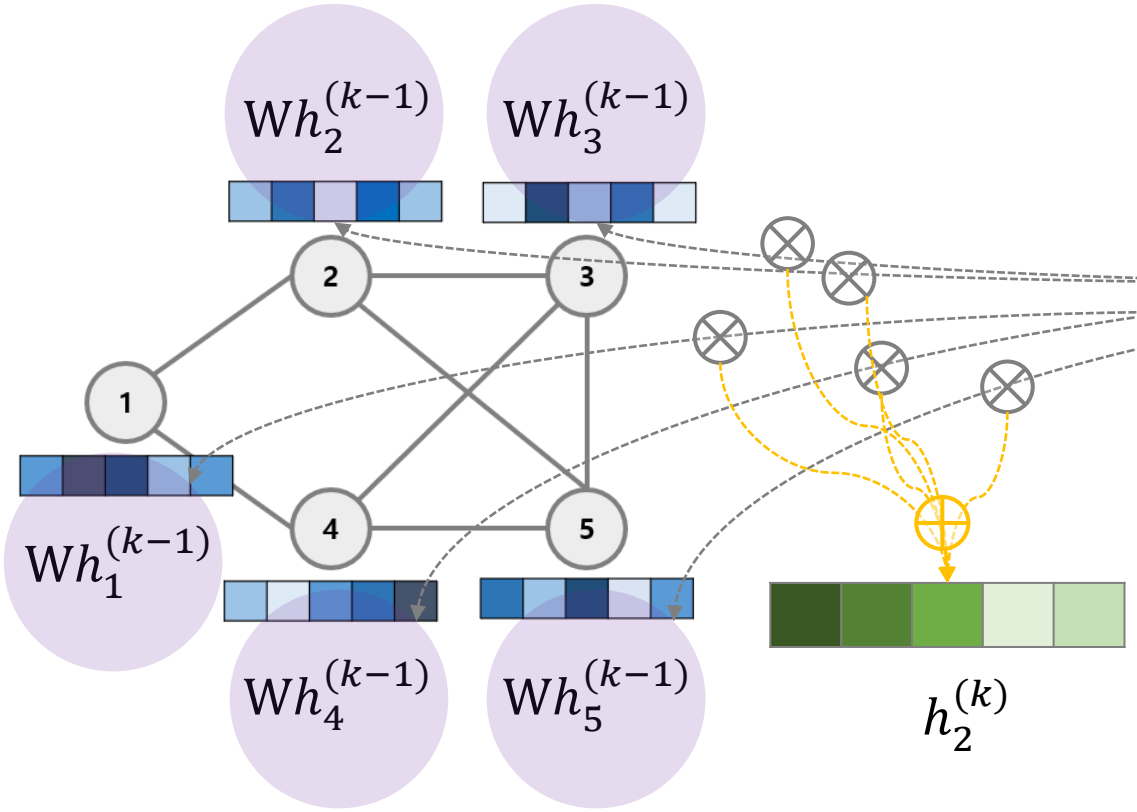
Veličković, Petar, et al. "Graph attention networks." *arXiv preprint arXiv:1710.10903* (2017).

Data Mining Quality Analytics

- Model Architecture
  - Combine
    - ✓ $h_v^{(k)} = \sum_{u \in N(v) \cup \{v\}} a_u^{(k-1)} h_u^{(k-1)}$
    - ✓ Weight-sum of aggregated information & k-1 hidden state (Value)

$$A(q, K, V) = \sum_i softmax(f(K, q))V$$





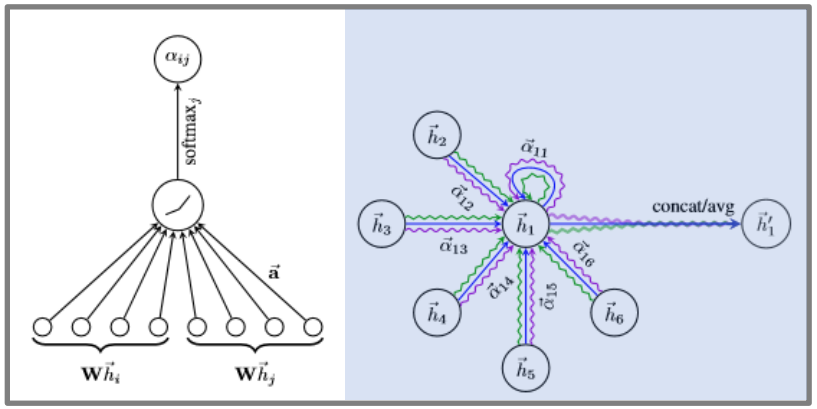
a_{11}	a_{12}	0	a_{14}	0
a_{21}	a_{22}	a_{23}	0	a_{25}
0	a_{32}	a_{33}	a_{34}	a_{35}
a_{41}	0	a_{43}	a_{44}	a_{45}
0	a_{25}	a_{53}	a_{54}	a_{55}

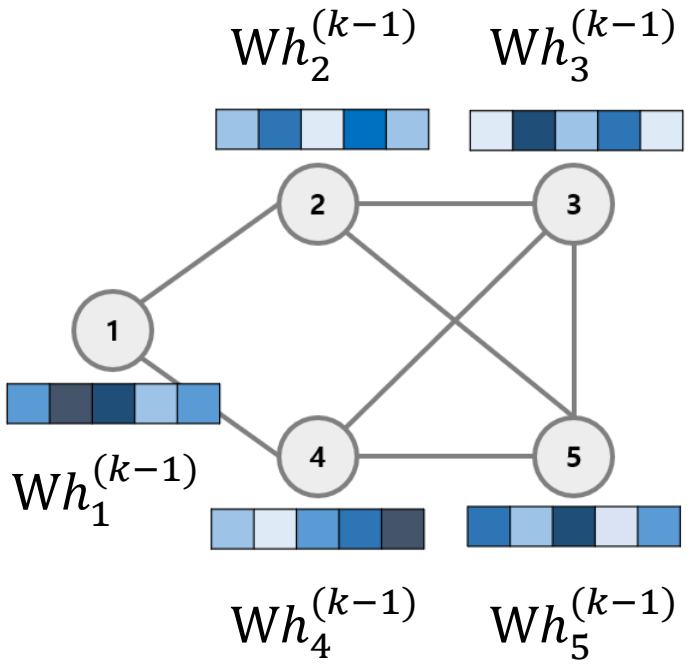
Veličković, Petar, et al. "Graph attention networks." *arXiv preprint arXiv:1710.10903* (2017).

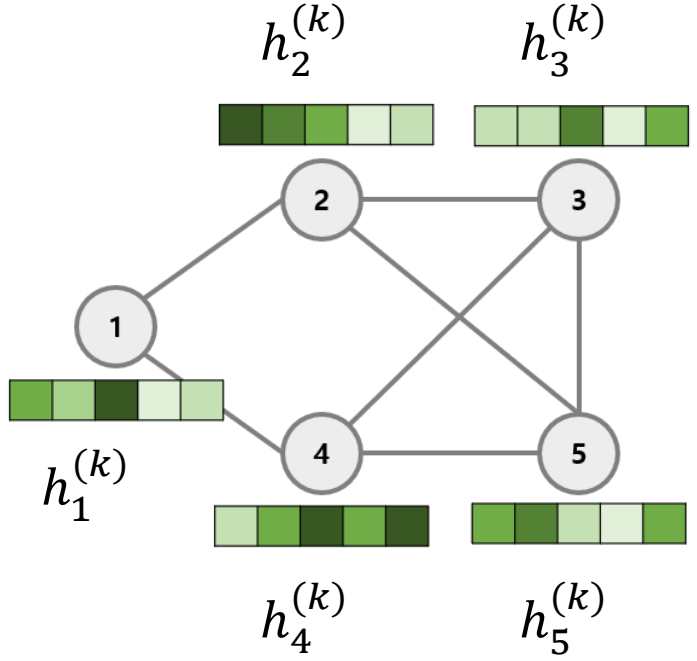
Data Mining Quality Analytics

- Model Architecture
  - Combine
    - ✓ $h_v^{(k)} = \sum_{u \in N(v) \cup \{v\}} a_u^{(k-1)} h_u^{(k-1)}$
    - ✓ Weight-sum of aggregated information & k-1 hidden state (Value)

$$A(q, K, V) = \sum_{i} softmax(f(K, q))V$$





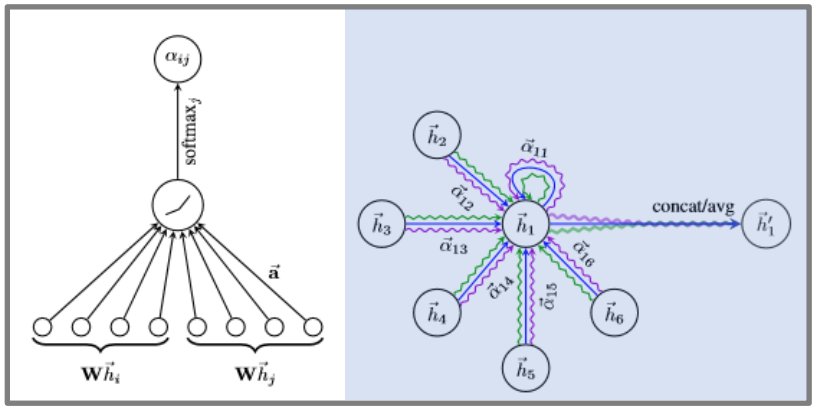


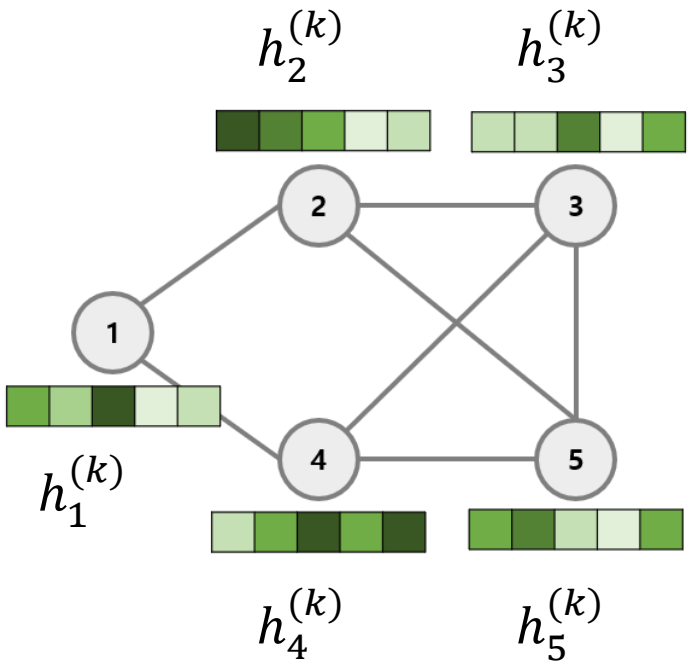
Veličković, Petar, et al. "Graph attention networks." *arXiv preprint arXiv:1710.10903* (2017).

Data Mining Quality Analytics

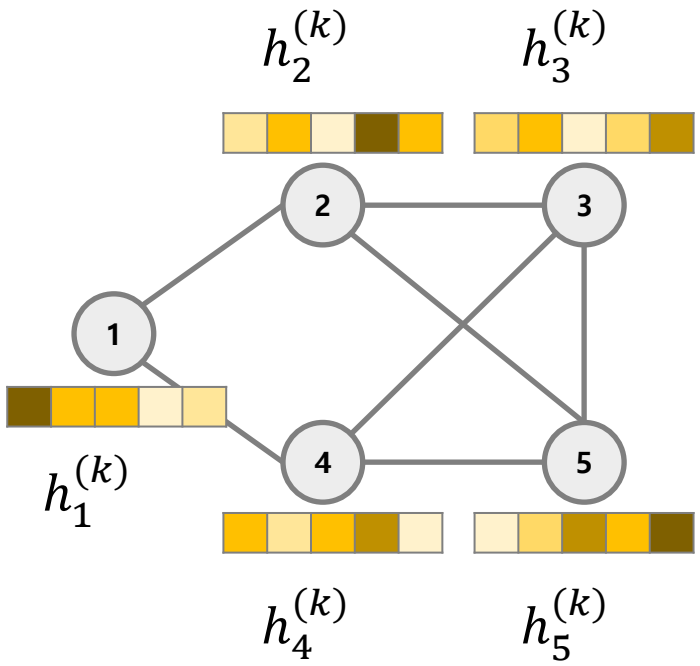
- Model Architecture
  - Multi-head Attention
    - ✓ Concatenate or average

$$A(q, K, V) = \sum_{i} softmax(f(K, q))V$$

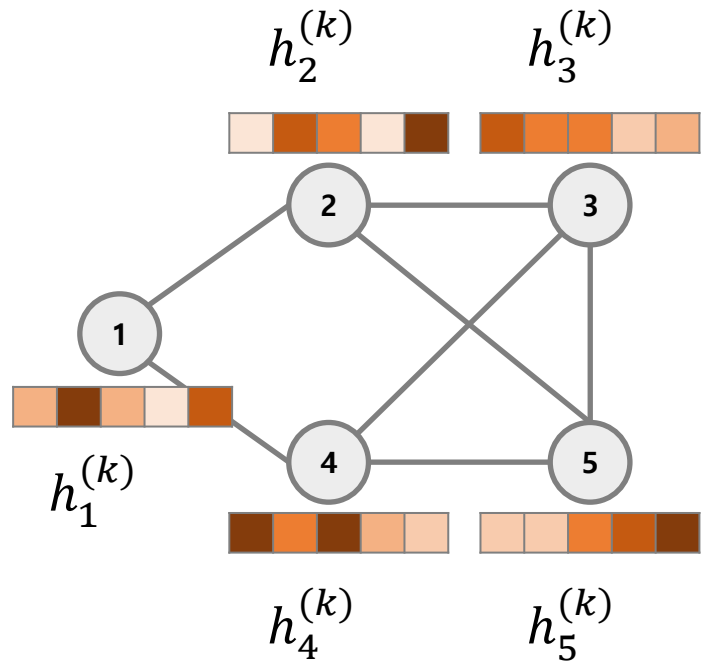




Head_1



Head_2



Head_3

- Result

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)	**0.973** ± 0.002

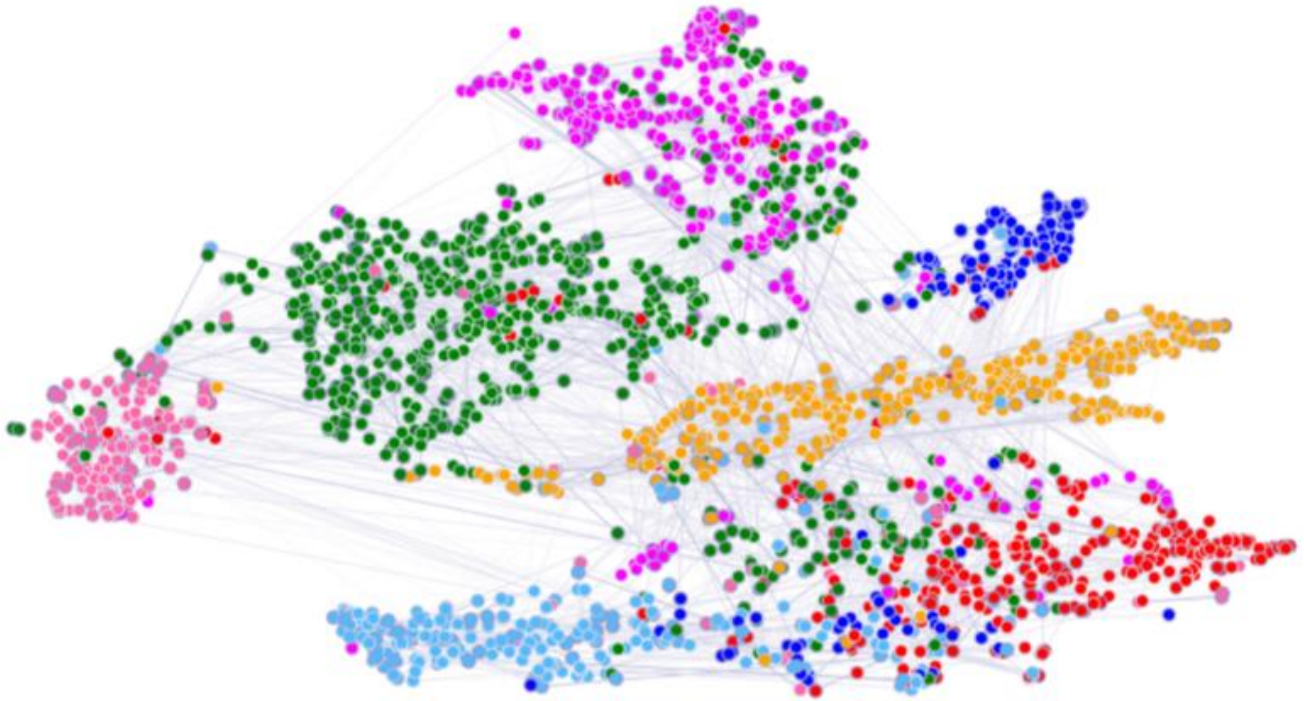
Transductive

Method	Cora	Citeseer	Pubmed
MLP	55.1%	46.5%	71.4%
ManiReg (Belkin et al., 2006)	59.5%	60.1%	70.7%
SemiEmb (Weston et al., 2012)	59.0%	59.6%	71.7%
LP (Zhu et al., 2003)	68.0%	45.3%	63.0%
DeepWalk (Perozzi et al., 2014)	67.2%	43.2%	65.3%
ICA (Lu & Getoor, 2003)	75.1%	69.1%	73.9%
Planetoid (Yang et al., 2016)	75.7%	64.7%	77.2%
Chebyshev (Defferrard et al., 2016)	81.2%	69.8%	74.4%
GCN (Kipf & Welling, 2017)	81.5%	70.3%	**79.0%**
MoNet (Monti et al., 2016)	81.7 ± 0.5%	—	78.8 ± 0.3%
GCN-64*	81.4 ± 0.5%	70.9 ± 0.5%	**79.0** ± 0.3%
GAT (ours)	**83.0** ± 0.7%	**72.5** ± 0.7%	**79.0** ± 0.3%

Veličković, Petar, et al. "Graph attention networks." *arXiv preprint arXiv:1710.10903* (2017).

Data Mining Quality Analytics

- Attention Score Visualization
  - Color = Class of node
  - Edge thickness = average of multi-head attention score



Veličković, Petar, et al. "Graph attention networks." *arXiv preprint arXiv:1710.10903* (2017).

- Conclusion
  - Attention
    - ✓ Attention: Query와 key의 유사도를 계산한 후 value의 가중합을 계산하는 과정
  - Graph Neural Network
    - ✓ Graph 구조를 학습하는 딥러닝 모델
    - ✓ Text, Image 데이터를 그래프로 표현해서 학습하는 모델들도 연구가 많이 되고 있음
  - Graph Attention Network
    - ✓ Attention 개념을 GNN에 적용하여 explainability + model performance

Veličković, Petar, et al. "Graph attention networks." *arXiv preprint arXiv:1710.10903* (2017).

- Pytorch-Geometric
  - Message Passing(Aggregate + Combine)

CLASS **MessagePassing** (**aggr**: Optional[str] = `'add'`, **flow**: str = `'source_to_target'`, **node_dim**: int = `-2`) [source]

Base class for creating message passing layers of the form

$$\mathbf{x}_i' = \gamma_{\mathbf{\Theta}} \left(\mathbf{x}_i, \square_{j \in \mathcal{N}(i)} \, \phi_{\mathbf{\Theta}} \left(\mathbf{x}_i, \mathbf{x}_j, \mathbf{e}_{j,i} \right) \right),$$

where $\square$ denotes a differentiable, permutation invariant function, *e.g.*, sum, mean or max, and $\gamma_{\mathbf{\Theta}}$ and $\phi_{\mathbf{\Theta}}$ denote differentiable functions such as MLPs. See here for the accompanying tutorial.

PARAMETERS

- **aggr** (*string, optional*) – The aggregation scheme to use (`"add"`, `"mean"`, `"max"` or `None`). (default: `"add"`)
- **flow** (*string, optional*) – The flow direction of message passing (`"source_to_target"` or `"target_to_source"`). (default: `"source_to_target"`)
- **node_dim** (*int, optional*) – The axis along which to propagate. (default: `-2`)

CLASS **GCNConv** (**in_channels**: int, **out_channels**: int, **improved**: bool = `False`, **cached**: bool = `False`, **add_self_loops**: bool = `True`, **normalize**: bool = `True`, **bias**: bool = `True`, ****kwargs**) [source]

The graph convolutional operator from the "Semi-supervised Classification with Graph Convolutional Networks" paper

$$\mathbf{X}' = \hat{\mathbf{D}}^{-1/2} \hat{\mathbf{A}} \hat{\mathbf{D}}^{-1/2} \mathbf{X} \mathbf{\Theta},$$

where $\hat{\mathbf{A}} = \mathbf{A} + \mathbf{I}$ denotes the adjacency matrix with inserted self-loops and $\hat{D}_{ii} = \sum_{j=0} \hat{A}_{ij}$ its diagonal degree matrix.

PARAMETERS

- **in_channels** (*int*) – Size of each input sample.
- **out_channels** (*int*) – Size of each output sample.

CLASS **GATConv** (**in_channels**: Union[int, Tuple[int, int]], **out_channels**: int, **heads**: int = `1`, **concat**: bool = `True`, **negative_slope**: float = `0.2`, **dropout**: float = `0.0`, **add_self_loops**: bool = `True`, **bias**: bool = `True`, ****kwargs**) [source]

The graph attentional operator from the "Graph Attention Networks" paper

$$\mathbf{x}_i' = \alpha_{i,i} \mathbf{\Theta} \mathbf{x}_i + \sum_{j \in \mathcal{N}(i)} \alpha_{i,j} \mathbf{\Theta} \mathbf{x}_j,$$

where the attention coefficients $\alpha_{i,j}$ are computed as

$$\alpha_{i,j} = \frac{\exp\left(\mathrm{LeakyReLU}\left(\mathbf{a}^\top [\mathbf{\Theta}\mathbf{x}_i \| \mathbf{\Theta}\mathbf{x}_j]\right)\right)}{\sum_{k \in \mathcal{N}(i) \cup \{i\}} \exp\left(\mathrm{LeakyReLU}\left(\mathbf{a}^\top [\mathbf{\Theta}\mathbf{x}_i \| \mathbf{\Theta}\mathbf{x}_k]\right)\right)}.$$

감사합니다