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

중요 Node에 가중치를 부여하는 어텐션 메커니즘을 사용하여

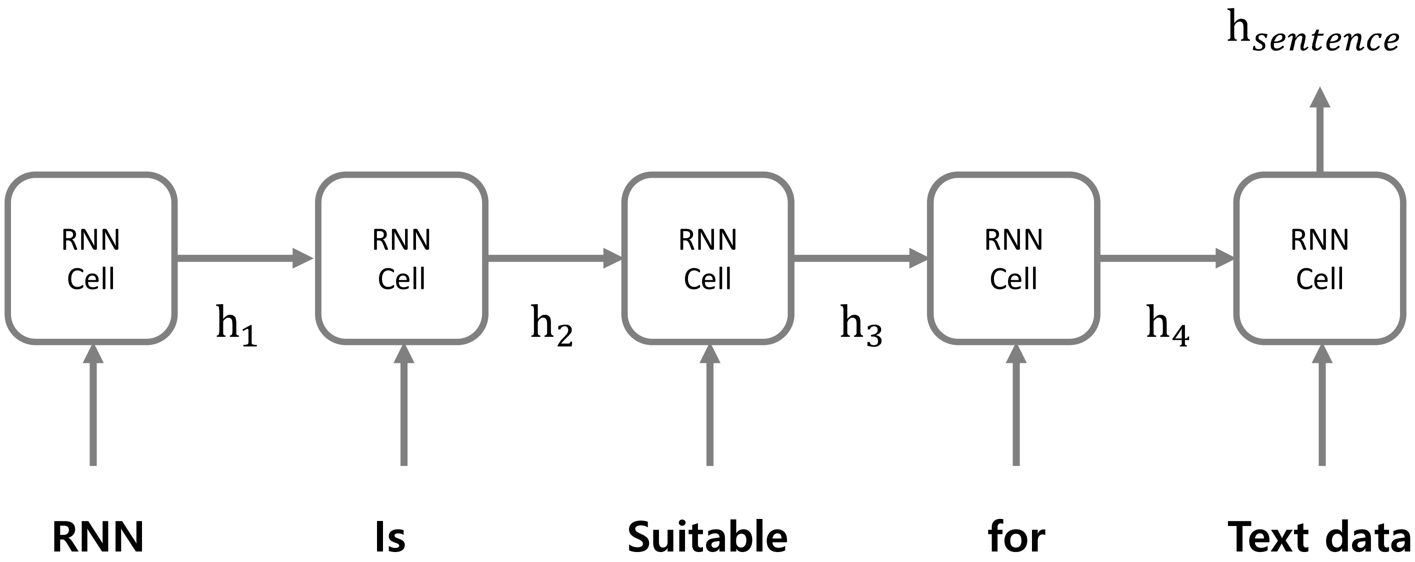
Graph Attention Networks

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

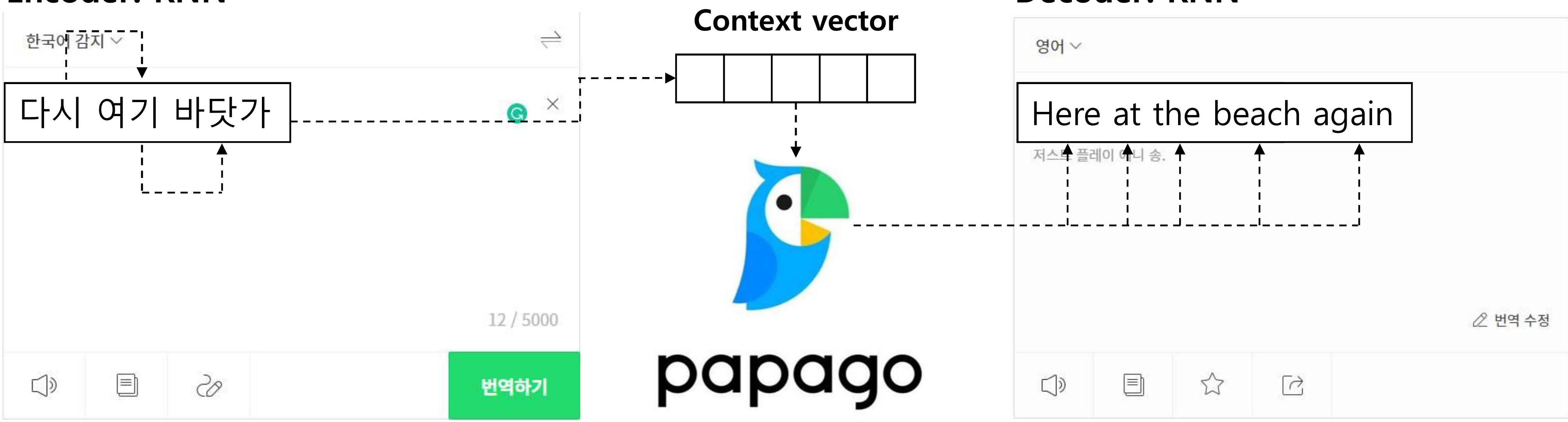


02 | Attention

Data Mining Quality Analytics

- Seq2seq
  - RNN encoder + RNN decoder
  - Machine translation

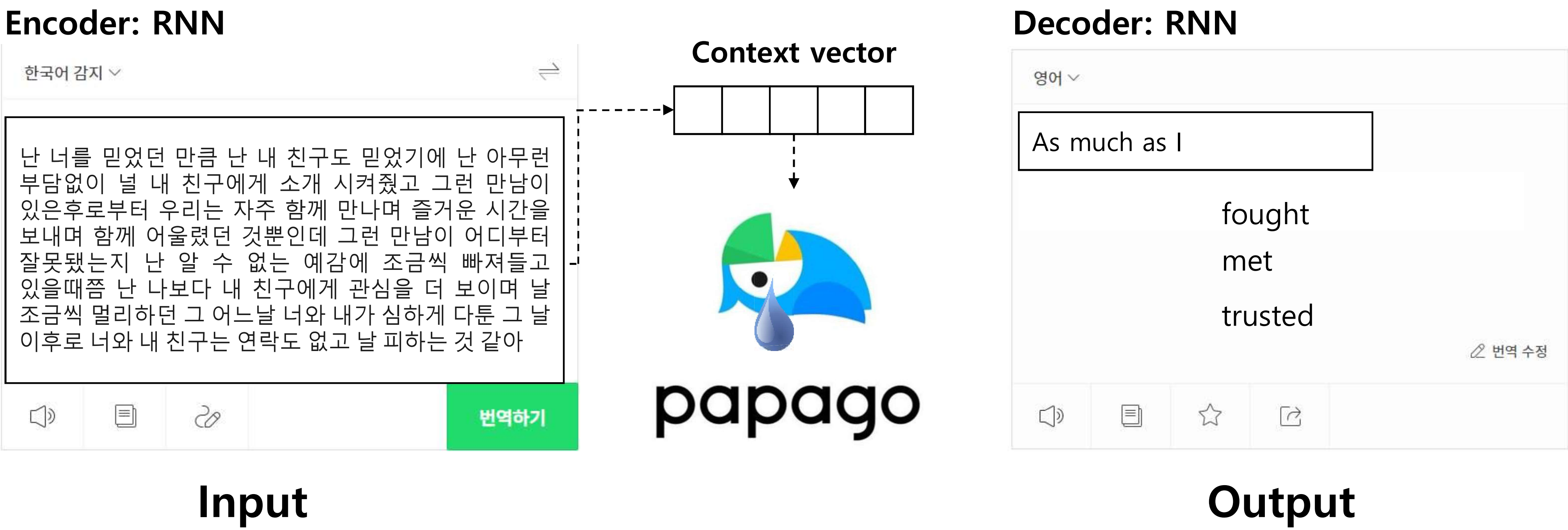
Encoder: RNN **Decoder: RNN**



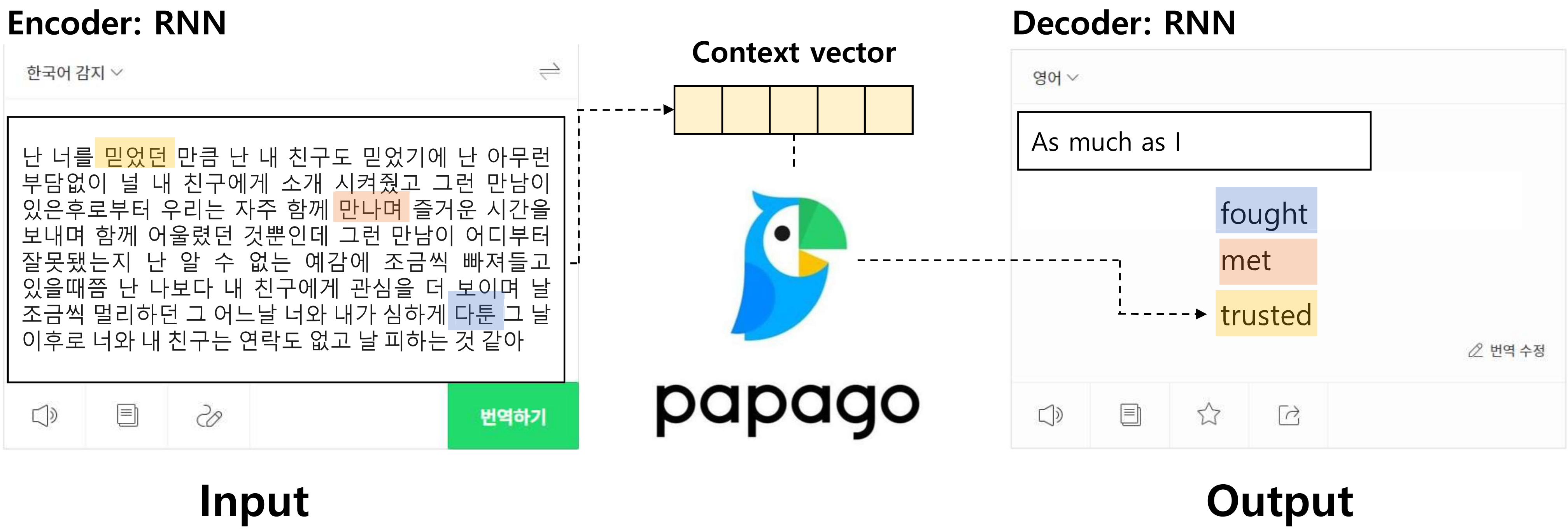
Input **Output**

Data Mining Quality Analytics

- Seq2seq
  - Long term dependency
  - Vanishing/Exploding gradient



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



02 | Attention

- 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

- 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

02 | Attention

- 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
이효리	0	천옥
엄정화	0	만옥
제시	0	은비
화사	0	실비
유재석	1	지미 유

부캐 Dictionary

SimXValue
0
0
0
0
지미 유

- 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)
```

- 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

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

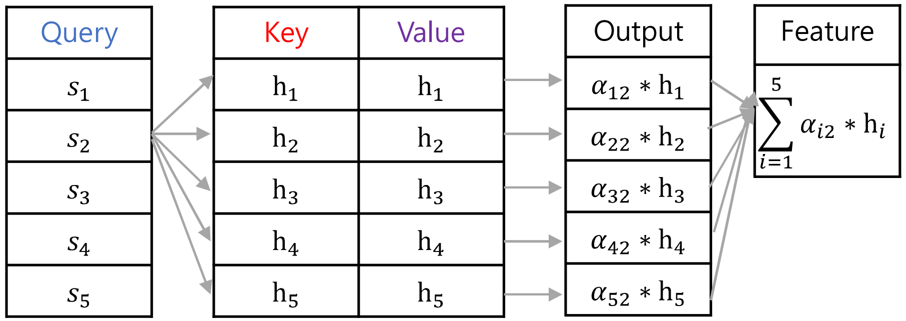
Dictionary 자료 구조

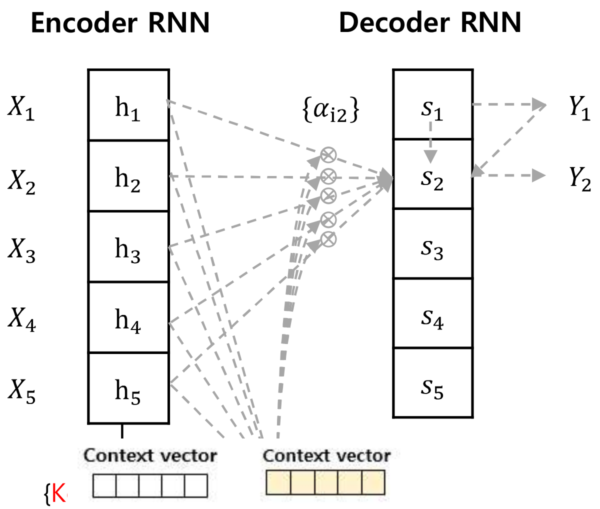
$$A(q, K, V) = \sum_{i} softmax(f(K, q))\,V$$

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





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

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

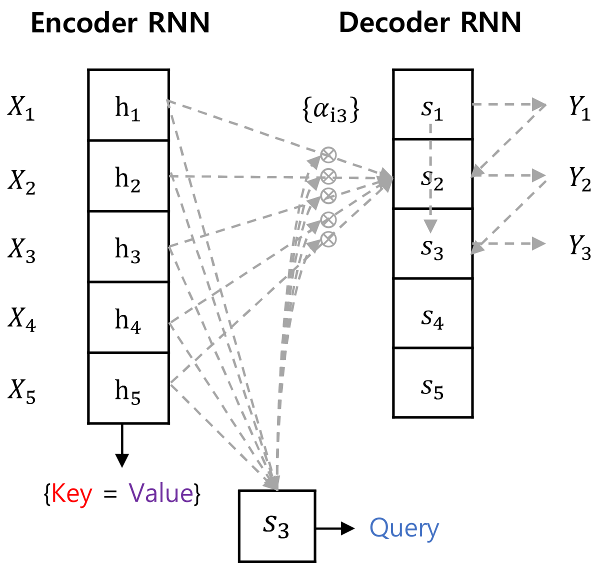
$$A(q, K, V) = \sum_{i} softmax(f(K, q))\, V$$

papago papago **Machine Translation**

02 | Attention

- 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

Query	Key	Value	Output	Feature
s_1	h_1	h_1	$\alpha_{13} * \mathrm{h}_1$	$\sum_{i=1}^{5} \alpha_{i3} * \mathrm{h}_i$
s_2	h_2	h_2	$\alpha_{23} * \mathrm{h}_2$	
s_3	h_3	h_3	$\alpha_{33} * \mathrm{h}_3$	
s_4	h_4	h_4	$\alpha_{43} * \mathrm{h}_4$	
s_5	h_5	h_5	$\alpha_{53} * \mathrm{h}_5$	



$$\{\alpha_{i3}\} = softmax(f(\mathrm{h}_i, s_3))$$

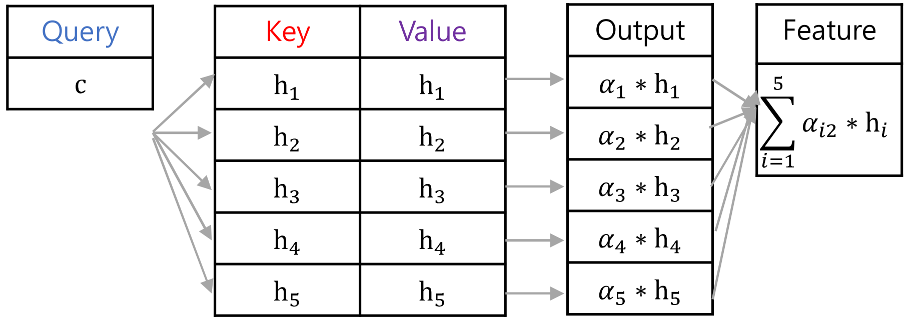
$$feature = \sum_{i=1}^{5} \alpha_{i3} * \mathrm{h}_i$$

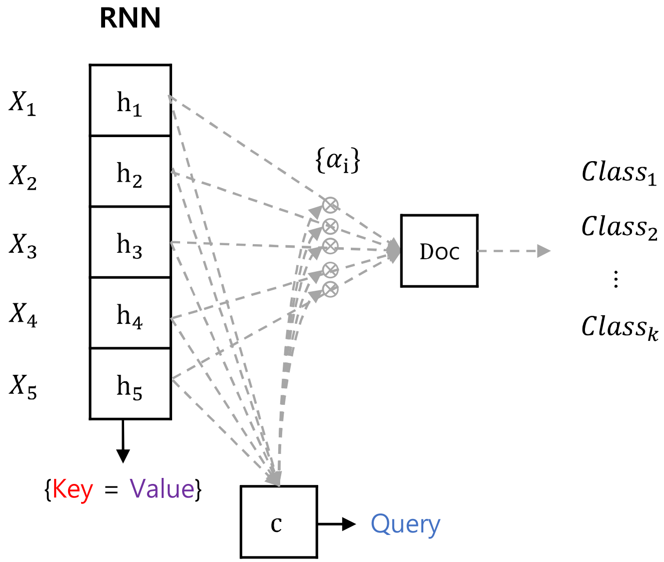
$$A(q, K, V) = \sum_{\mathrm{i}} softmax(f(K, q))\, V$$

Machine Translation

02 | Attention

- 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)





$$\{\alpha_i\} = softmax(f(\mathrm{h}_i, c))$$

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

⬇

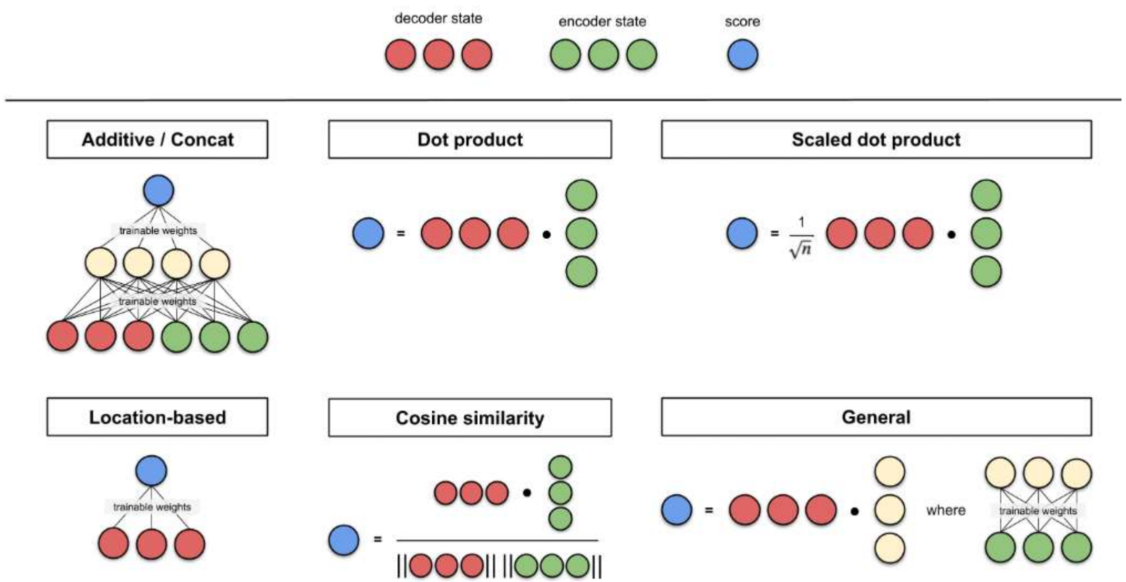
$$A(q, K, V) = \sum_{\mathrm{i}} softmax(f(K, q))\, V$$

Document Classification

- 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

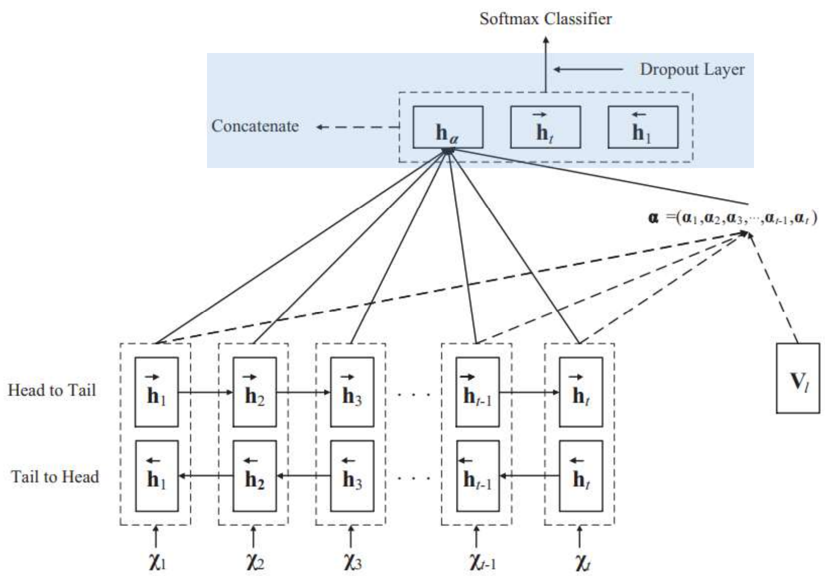
$$A(q, K, V) = \sum_i softmax(f(K, q))\, V$$

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

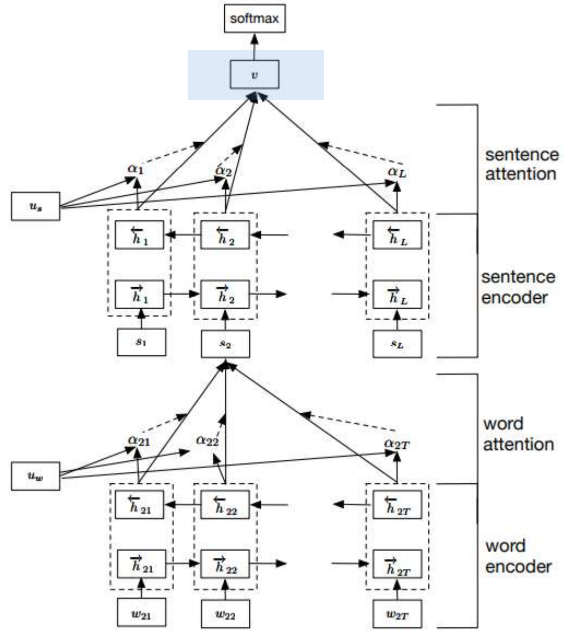


02 | Attention

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



Bidirectional RNN with Attention



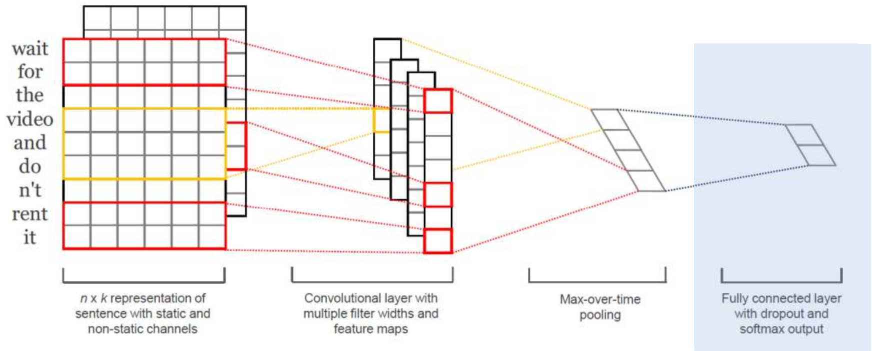
Hierarchical Attention Network

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

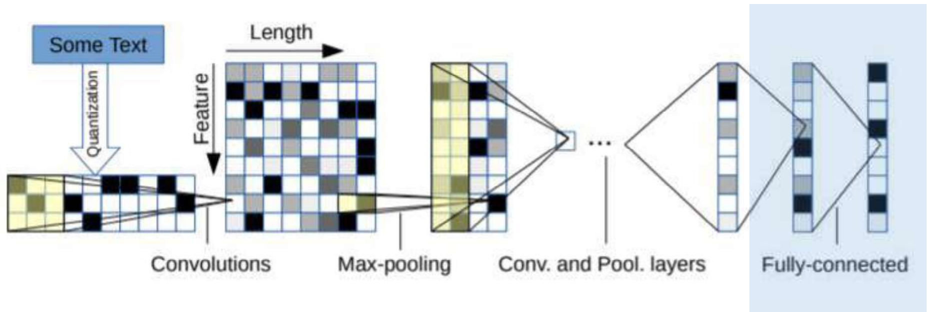
Yang, Zichao, et al. "Hierarchical attention networks for document classification." *Proceedings of the 2016 conference of the North American chapter of the association for computational linguistics* 2016.

Data Mining Quality Analytics

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



TextCNN



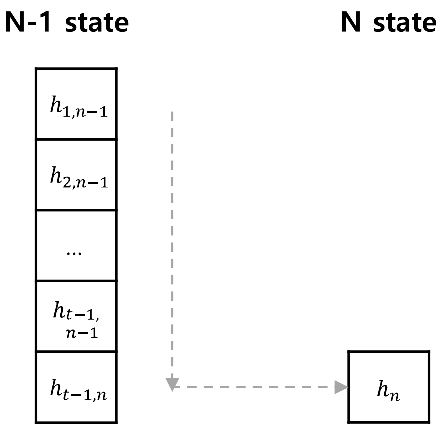
Character-level CNN

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

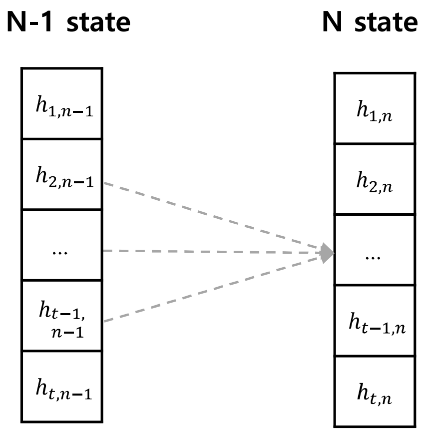
Zhang, Xiang, Junbo Zhao, and Yann LeCun. "Character-level convolutional networks for text classification." *Advances in neural information processing systems*. 2015.

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

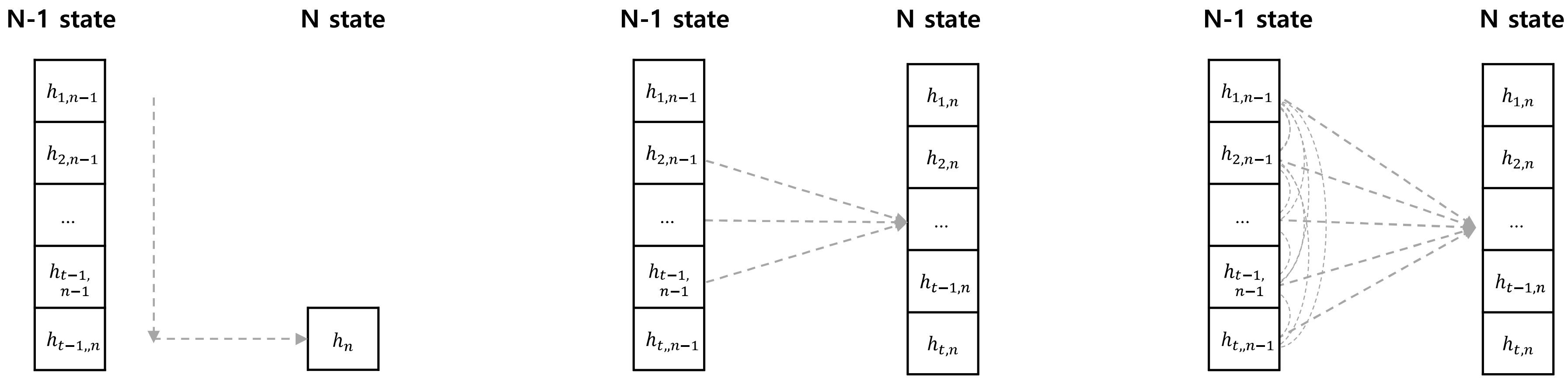


CNN-Based

- 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

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

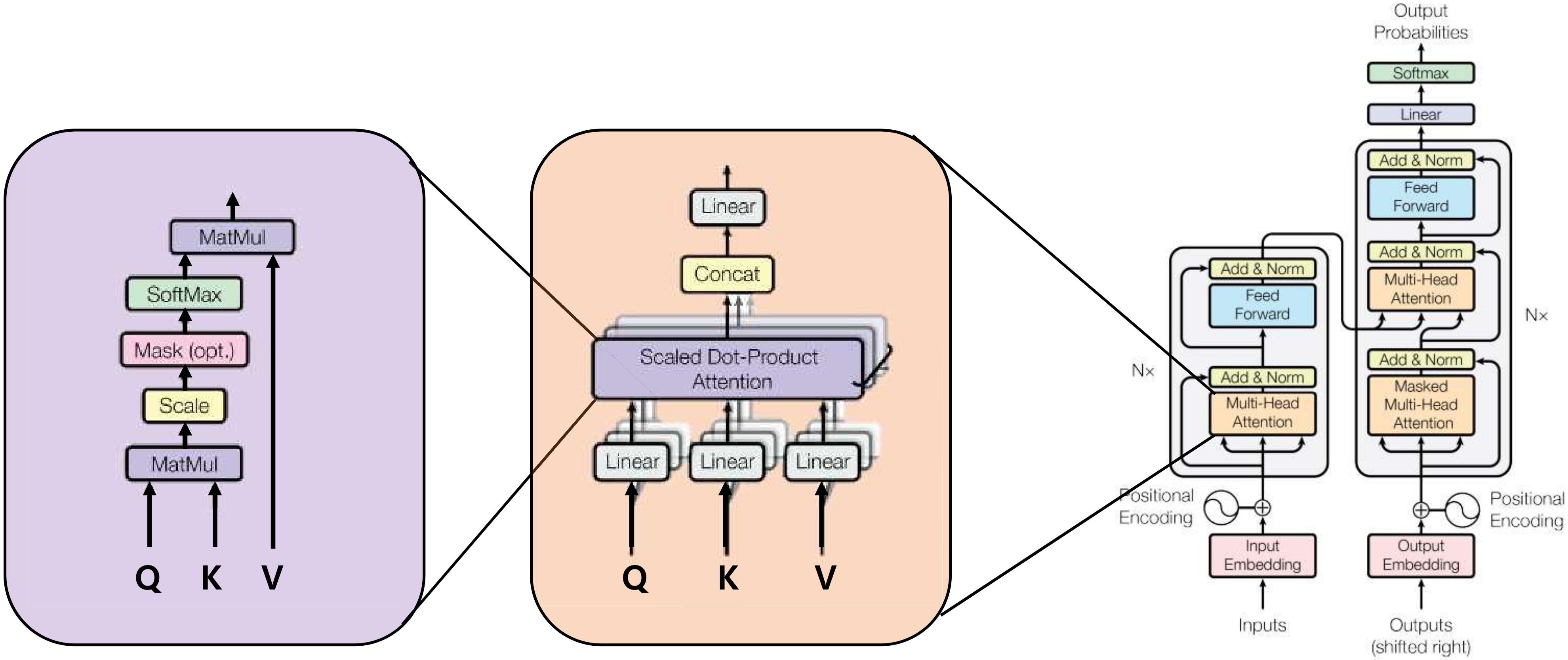
$$Key = Query = Value$$



RNN-Based **CNN-Based** **Self-Attention**

Data Mining Quality Analytics

- Transformer

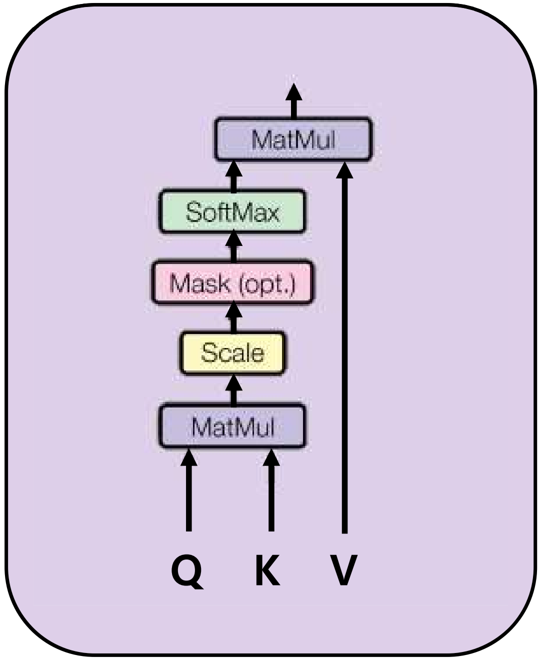


Scaled Dot-Product Attention **Multi-Head Attention** **Transformer**

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

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

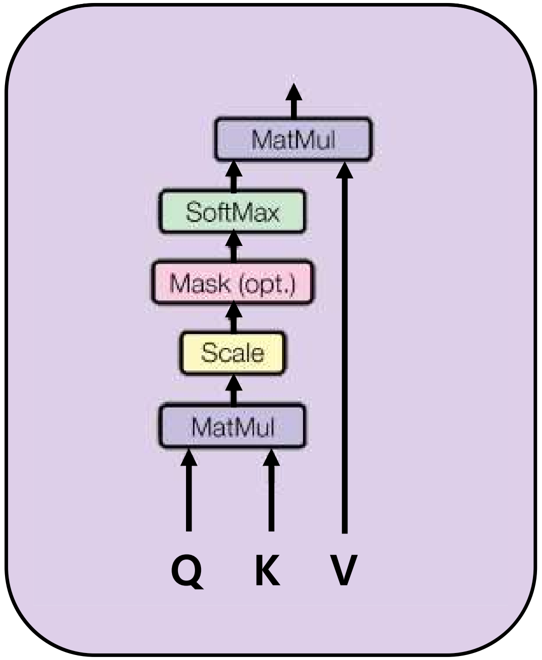
① **MatMul**

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

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..

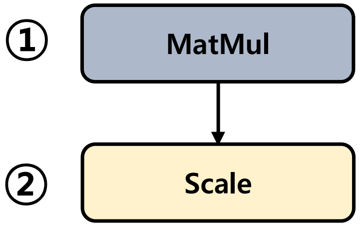
- Transformer
  - Scale-dot product attention (Self-Attention)
    - ✓ Similarity function: Scaled-dot product



Scaled Dot-Product Attention

Generalized Attention Form

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



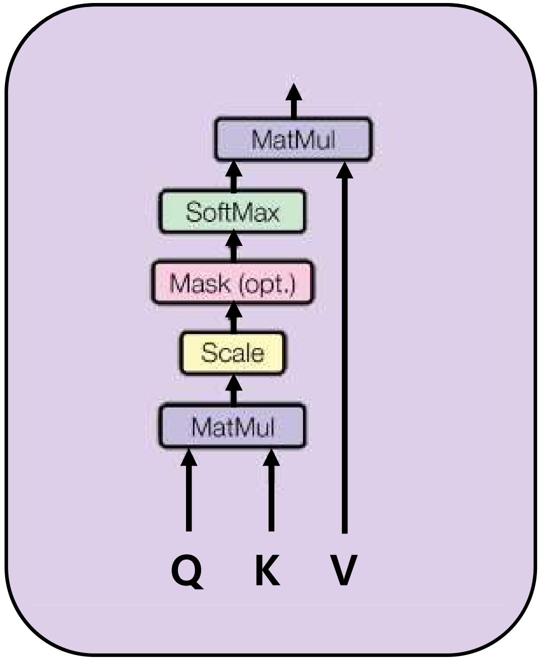
$$f(K, Q) = QK^{\mathrm{T}} \quad (K = XW^{K}, Q = XW^{Q},\ V = XW^{V})$$

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

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 (...)

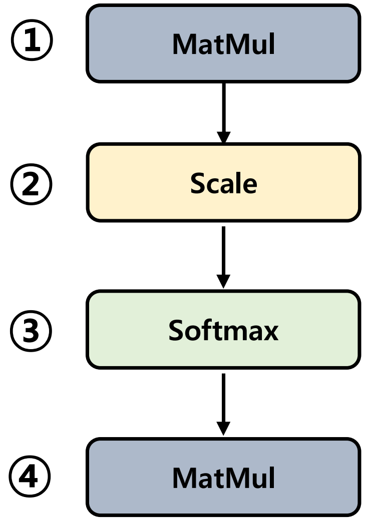
Vaswani, Ashish, et al. "Attention is all you need." *Advances in neural information processing systems.* 2017..

- 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 $f(K,Q) = QK^{T}$ $(K = KW^{K}, Q = QW^{Q},\ V = QW^{V})$

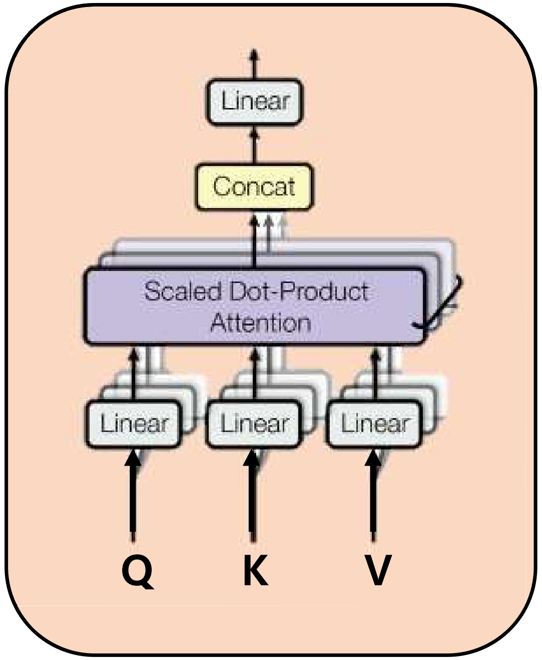
② $\frac{QK^T}{\sqrt{d_k}}$

③ $softmax(\frac{QK^T}{\sqrt{d_k}})$

④ $softmax(\frac{QK^T}{\sqrt{d_k}})V$

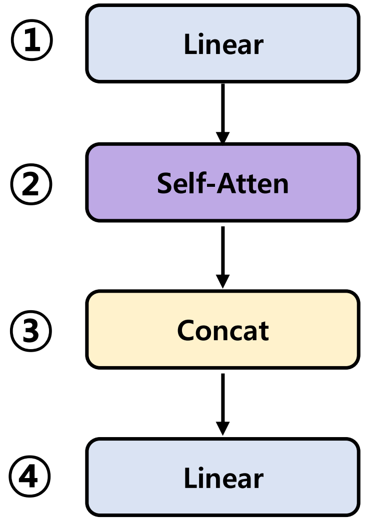
Vaswani, Ashish, et al. "Attention is all you need." *Advances in neural information processing systems*. 2017..

- 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 $Q` = QW_i^Q \quad K` = KW_i^K \quad V` = VW_i^V \quad (i = 1 \ldots h)$

② $\text{head}_i = SA(Q`, K`, V`)$

③ $[\text{head}_1, \text{head}_2, \ldots, \text{head}_h]$

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

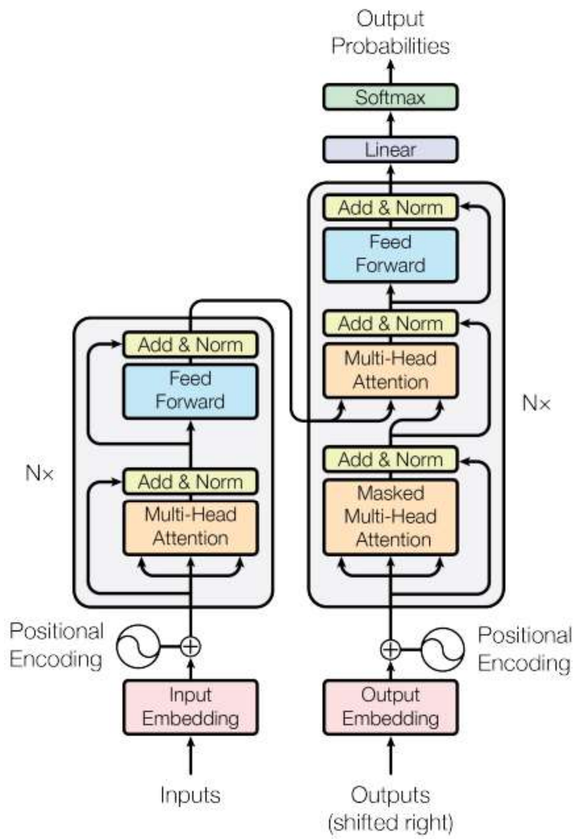
Vaswani, Ashish, et al. "Attention is all you need." *Advances in neural information processing systems.* 2017.

02 | Attention

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.

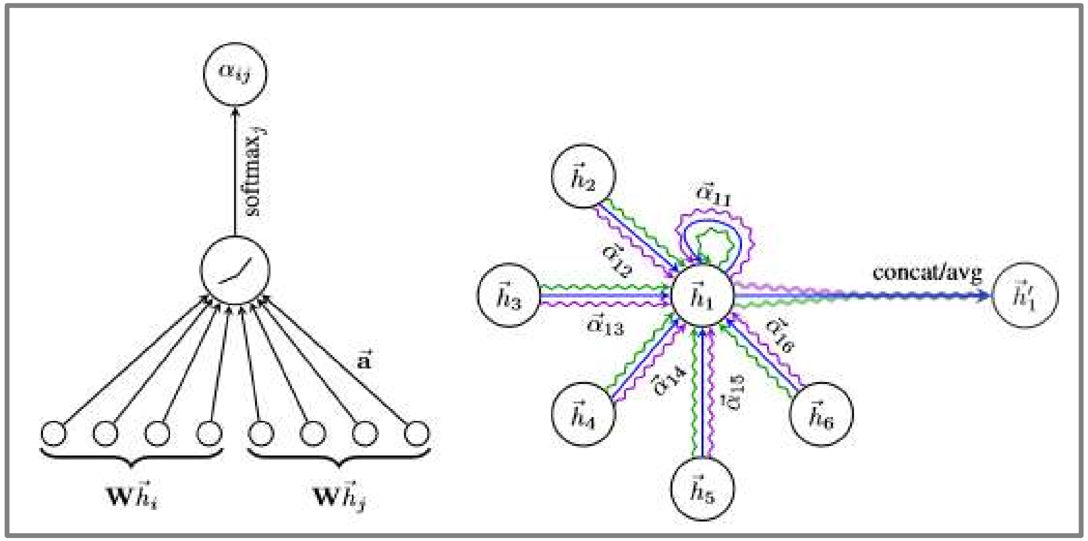


Transformer

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

04 | Graph Attention Networks

- 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).

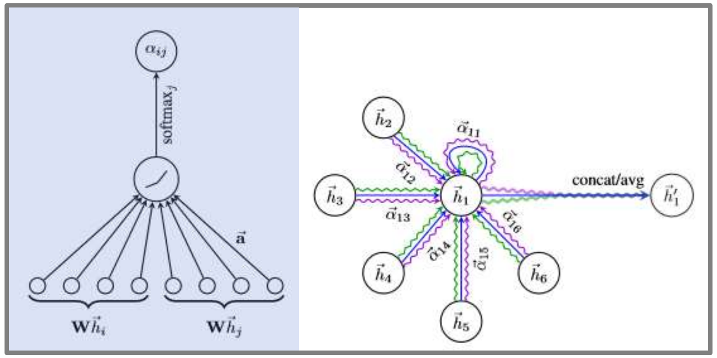
- Model Architecture
  - Aggregate

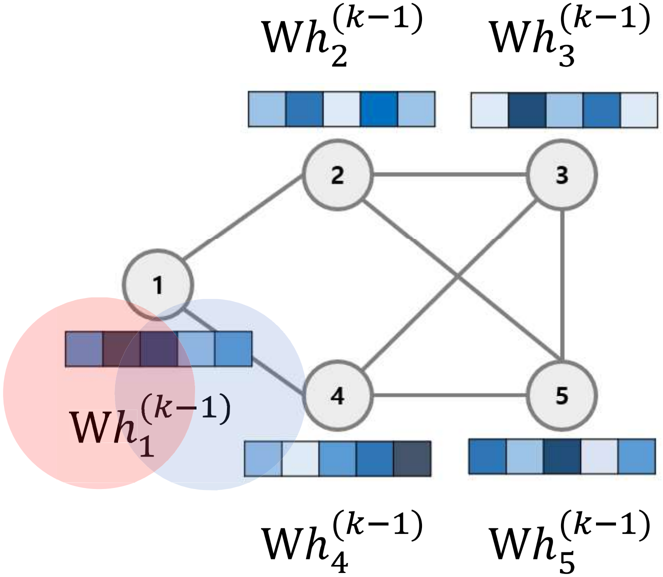
$$A(q,K,V)=\sum_{i} softmax(f(K,q))\,V$$

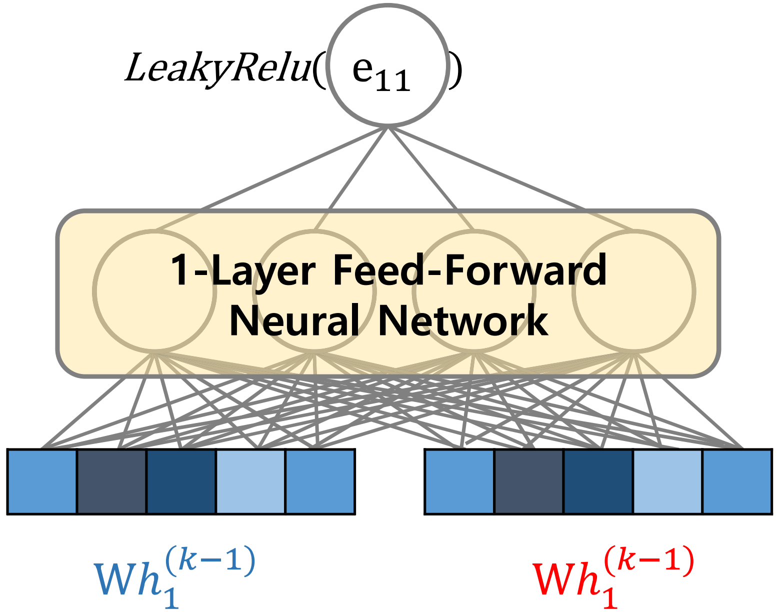
✓ $a_v^{(k-1)} = Attention(\{ h_u^{(k-1)},\ u \in N(v) \cup \{v\}\})$

✓ 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).

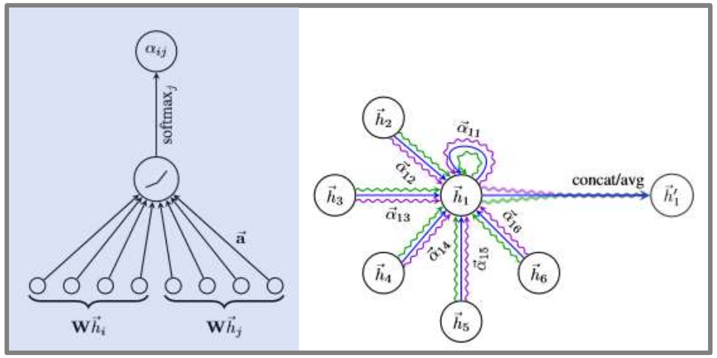
- Model Architecture
  - Aggregate

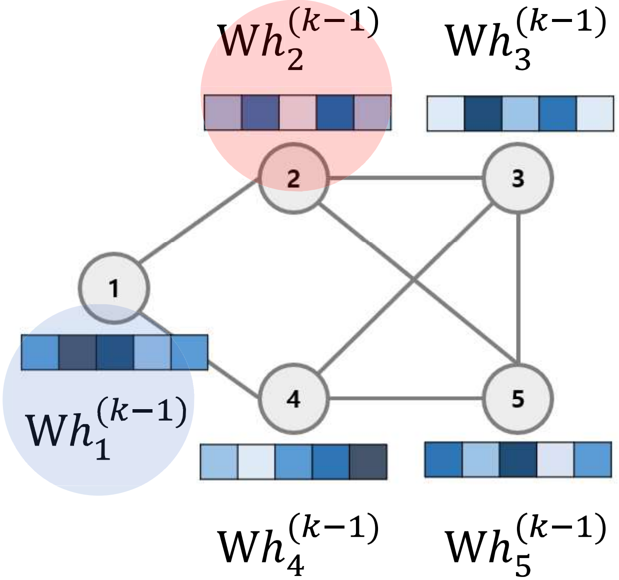
$$A(q,K,V) = \sum_{i} softmax(f(K,q))\,V$$

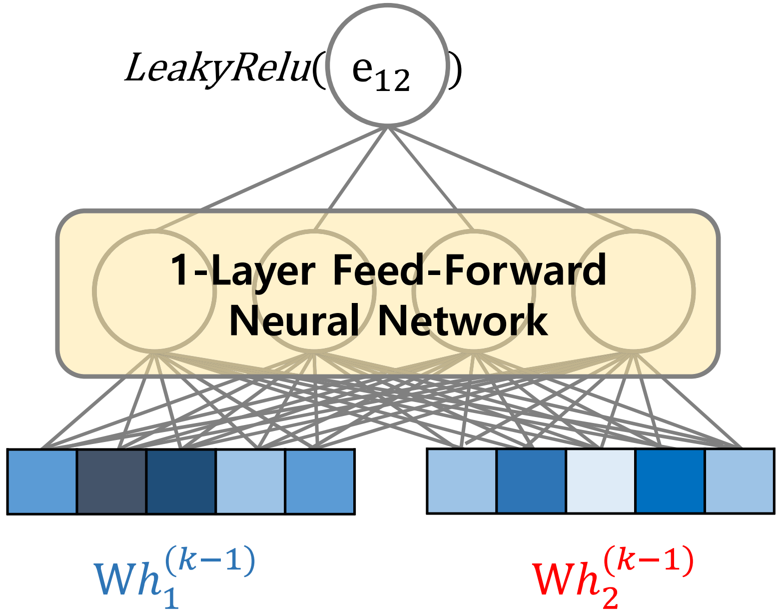
✓ $a_v^{(k-1)} = Attention(\{ h_u^{(k-1)},\ u \in N(v) \cup \{v\} \})$

✓ Key = Query = Value = $h_u^{(k-1)}$

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





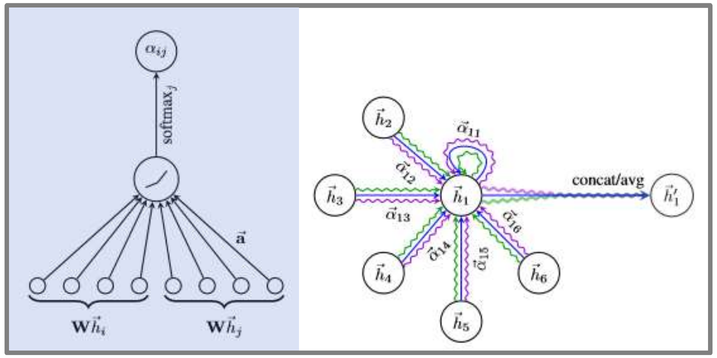


Veličković, Petar, et al. "Graph attention networks." *arXiv preprint arXiv:1710.10903* (2017).

Data Mining Quality Analytics

- Model Architecture
  - Aggregate

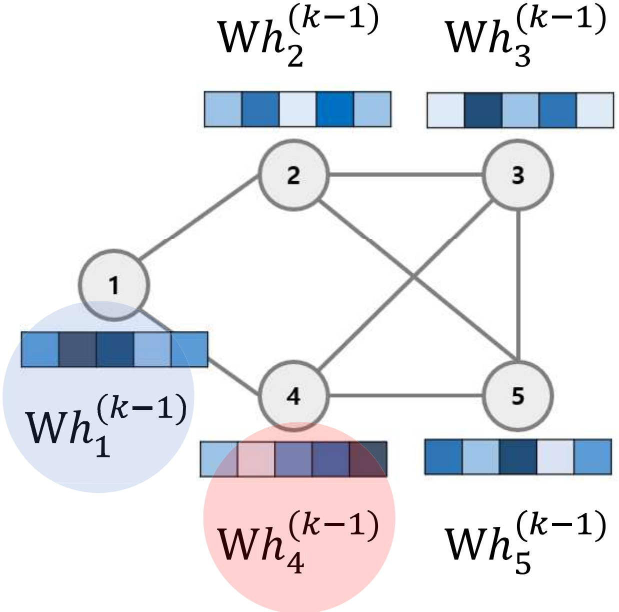
$$A(q,K,V) = \sum_i softmax(f(K,q))V$$

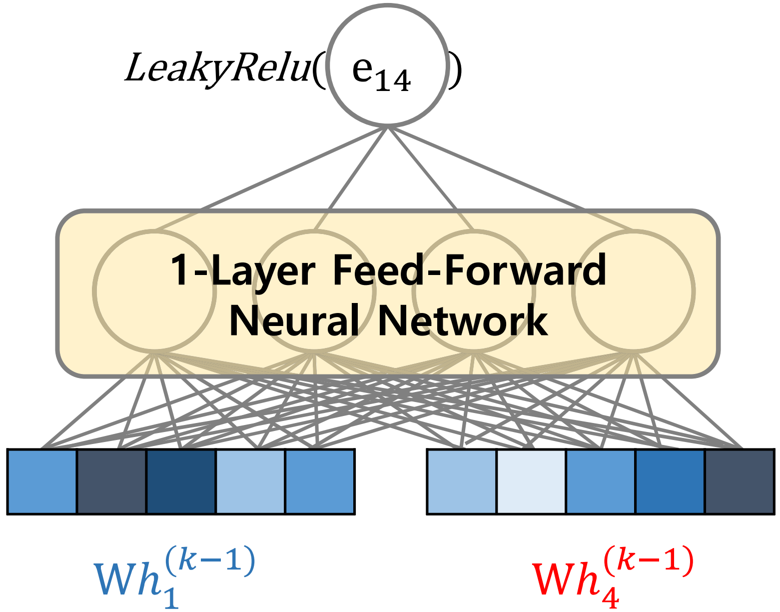


✓ $a_v^{(k-1)} = Attention(\{ h_u^{(k-1)},\ u \in N(v) \cup \{v\}\})$

✓ Key = Query = Value = $h_u^{(k-1)}$

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





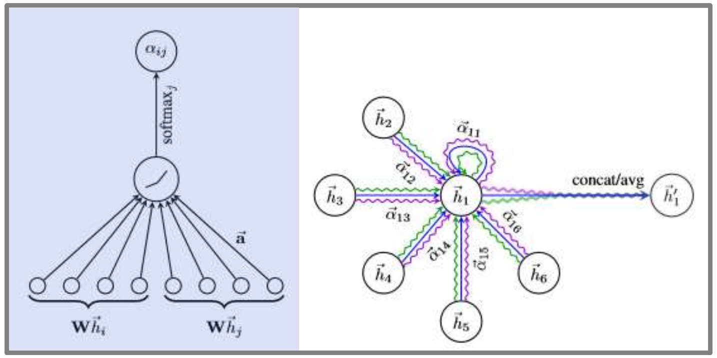
Veličković, Petar, et al. "Graph attention networks." *arXiv preprint arXiv:1710.10903* (2017).

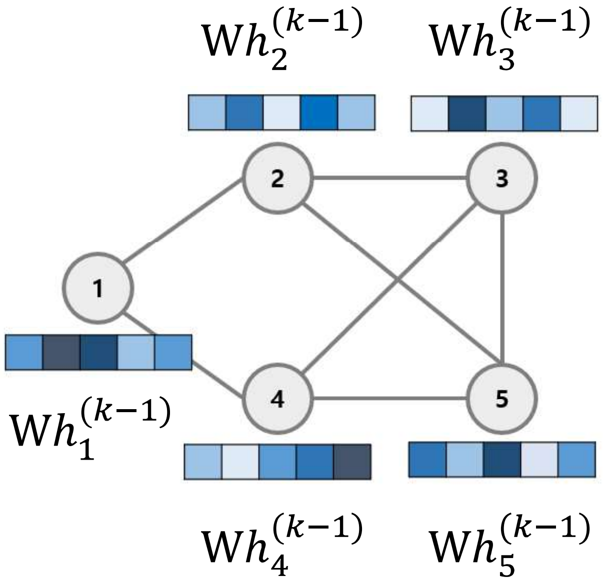
04 | Graph Attention Networks

Data Mining Quality Analytics

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

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





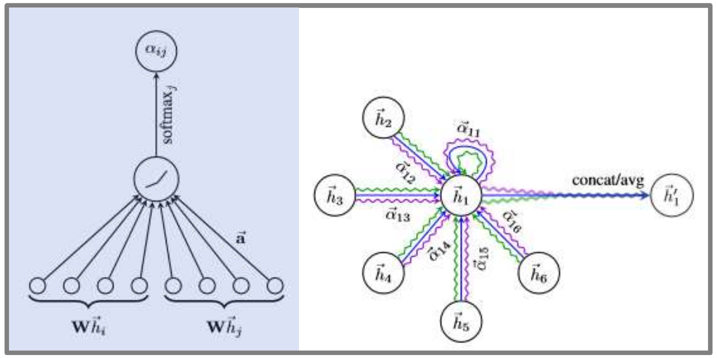
e_{11}	e_{12}		e_{14}	
e_{21}	e_{22}	e_{23}		e_{25}
	e_{32}	e_{33}	e_{34}	e_{35}
e_{41}		e_{43}	e_{44}	e_{45}
	e_{25}	e_{53}	e_{54}	e_{55}

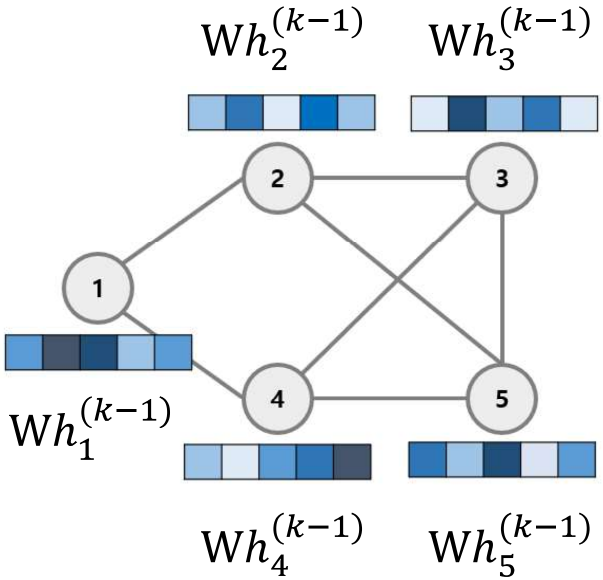
Veličković, Petar, et al. "Graph attention networks." *arXiv preprint arXiv:1710.10903* (2017).

Data Mining Quality Analytics

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

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





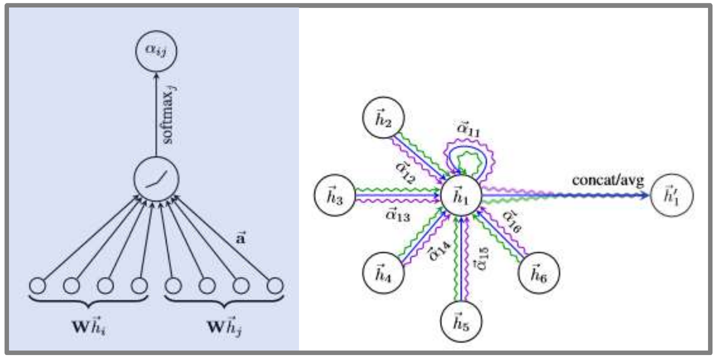
e_{11}	e_{12}	Softmax	e_{14}	
e_{21}	e_{22}	e_{23} Softmax		e_{25}
	e_{32}	e_{33} Softmax	e_{34}	e_{35}
e_{41}		e_{43} Softmax	e_{44}	e_{45}
	e_{25}	e_{53} Softmax	e_{54}	e_{55}

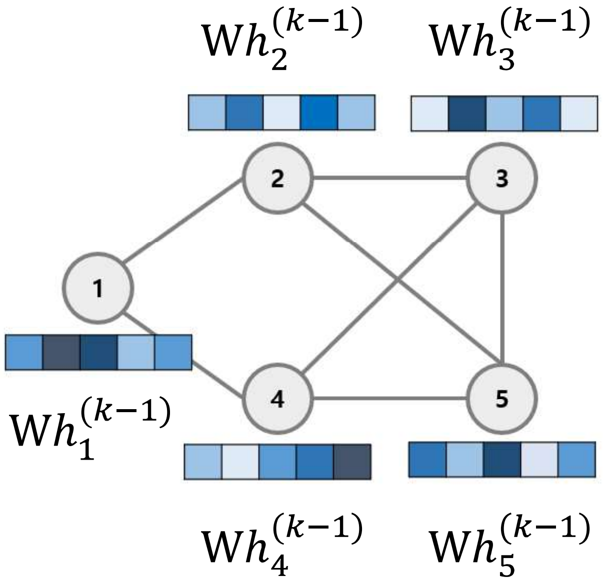
Veličković, Petar, et al. "Graph attention networks." *arXiv preprint arXiv:1710.10903* (2017).

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}

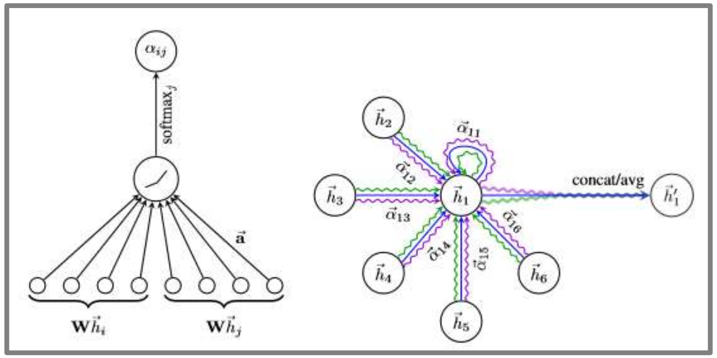
Veličković, Petar, et al. "Graph attention networks." *arXiv preprint arXiv:1710.10903* (2017).

04 | Graph Attention Networks

- 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.



Similarity

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}

→

Masking

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}

→

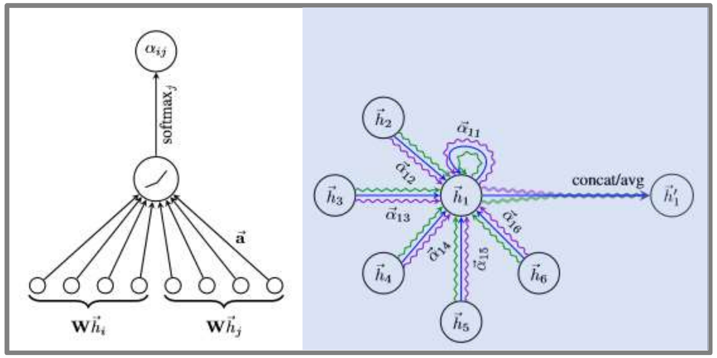
Softmax

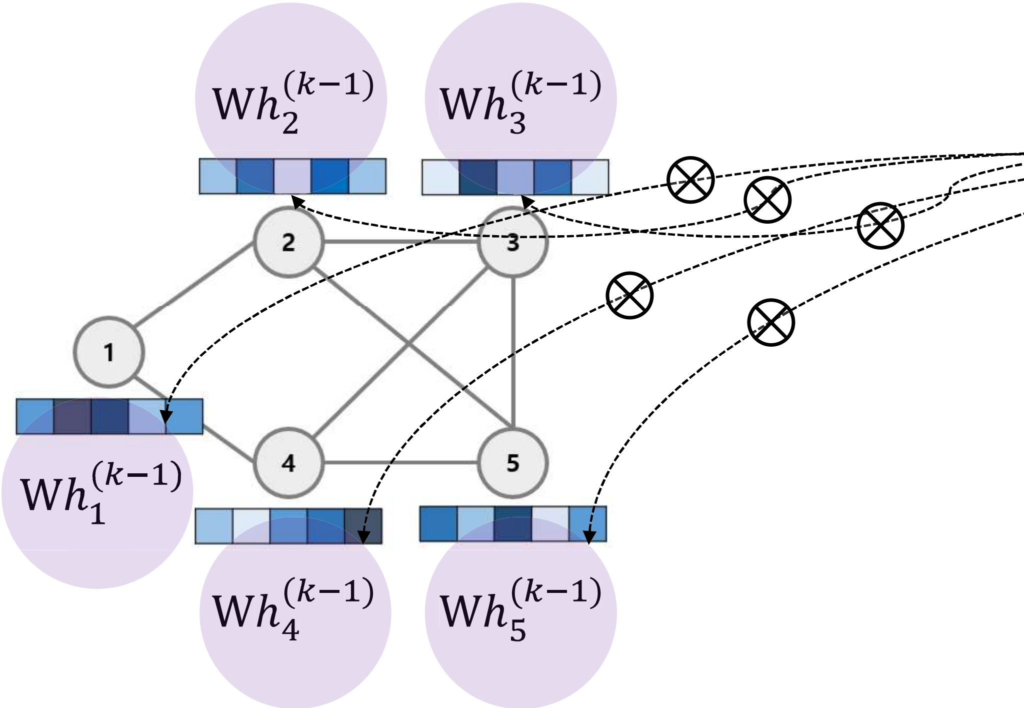
a_{11}	a_{12}	0	a_{14}	0
a_{21}	a_{22}	a_{23}	0	a_{25}
0	a_{32}	a_{33}	a_{34}	a_{35}
a_{41}	0	a_{43}	a_{44}	a_{45}
0	a_{52}	a_{53}	a_{54}	a_{55}

04 | Graph Attention Networks

- 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}

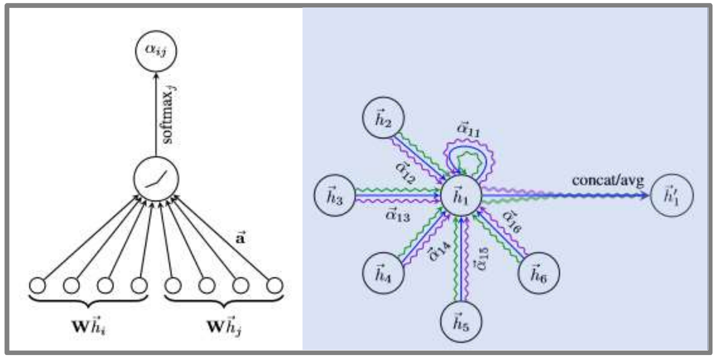
Veličković, Petar, et al. "Graph attention networks." *arXiv preprint arXiv:1710.10903* (2017).

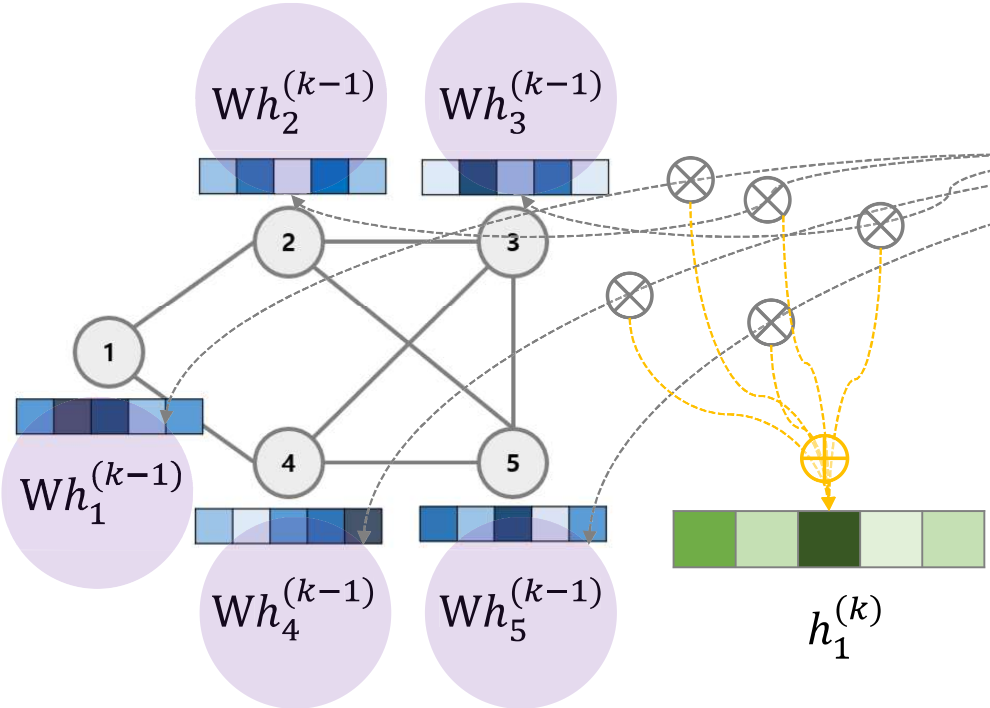
Data Mining Quality Analytics

- Model Architecture

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

  - 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_{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}

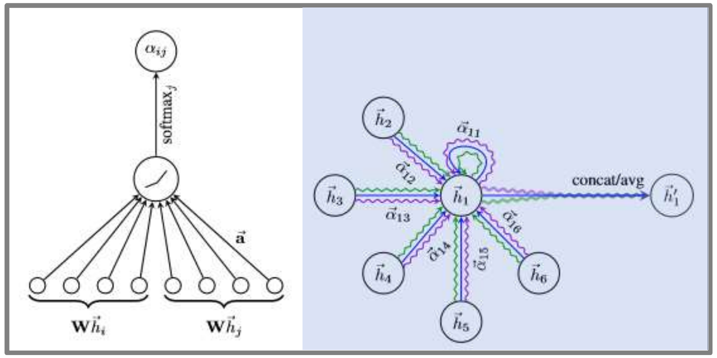
Veličković, Petar, et al. "Graph attention networks." *arXiv preprint arXiv:1710.10903* (2017).

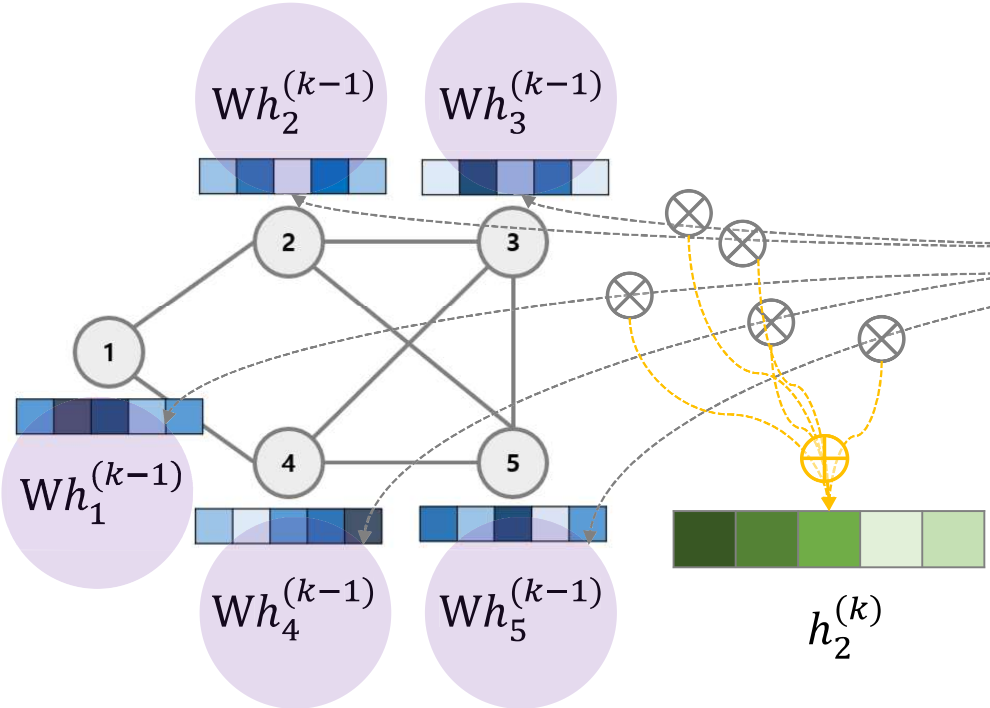
04 | Graph Attention Networks

- Model Architecture
  - Combine

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

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





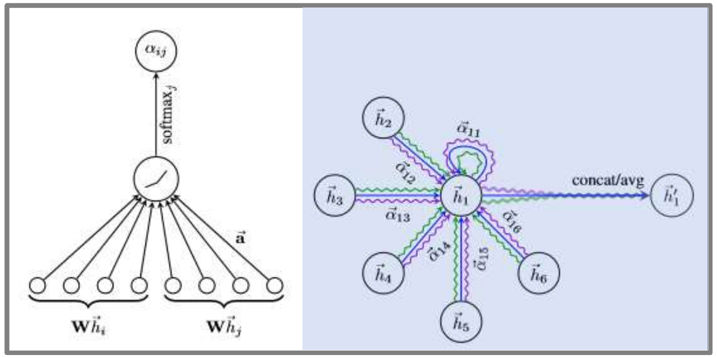
a_{11}	a_{12}	0	a_{14}	0
a_{21}	a_{22}	a_{23}	0	a_{25}
0	a_{32}	a_{33}	a_{34}	a_{35}
a_{41}	0	a_{43}	a_{44}	a_{45}
0	a_{25}	a_{53}	a_{54}	a_{55}

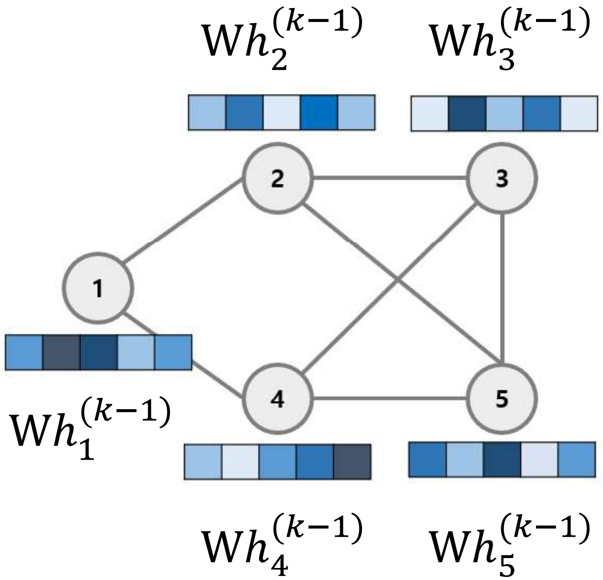
Veličković, Petar, et al. "Graph attention networks." *arXiv preprint arXiv:1710.10903* (2017).

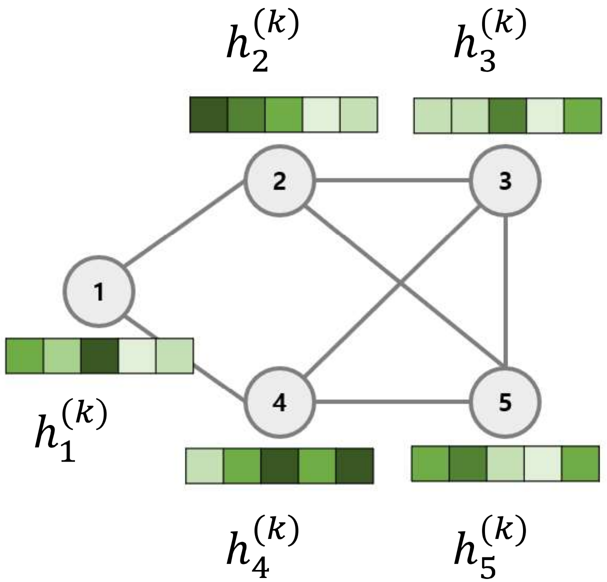
Data Mining Quality Analytics

- 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)

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



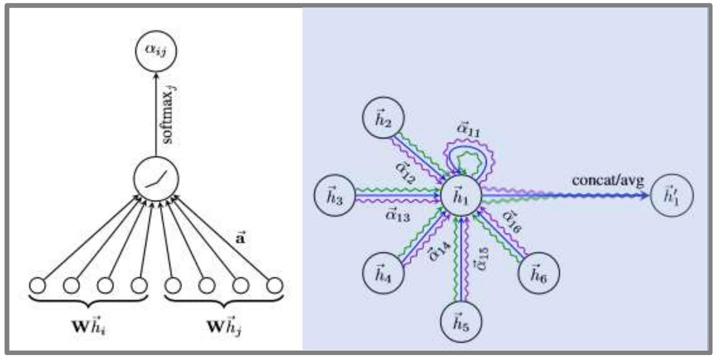


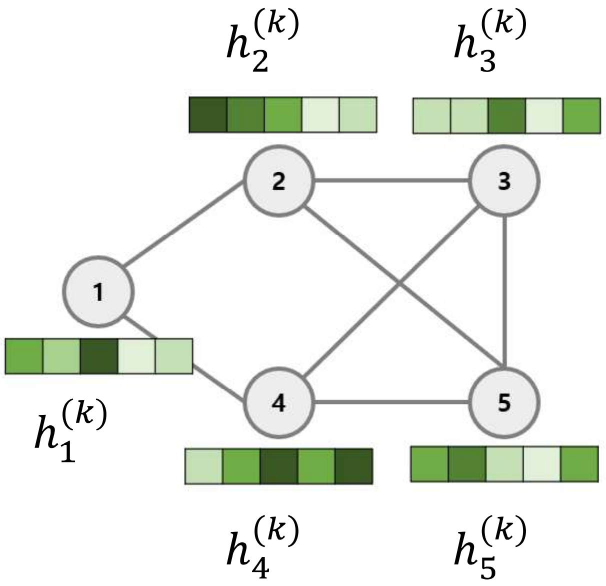


Veličković, Petar, et al. "Graph attention networks." *arXiv preprint arXiv:1710.10903* (2017).

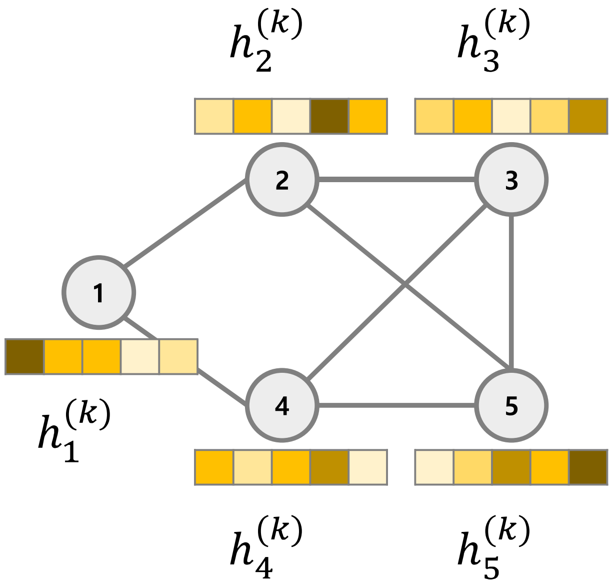
- Model Architecture
  - Multi-head Attention
    - ✓ Concatenate or average

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

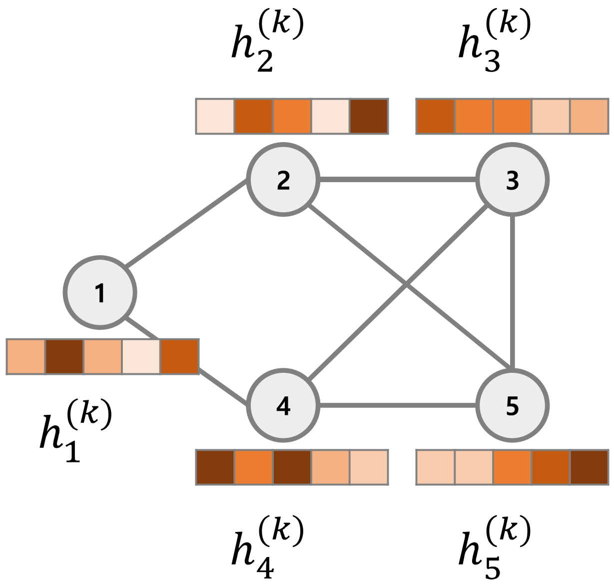




Head_1



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)	$\mathbf{0.973} \pm 0.002$

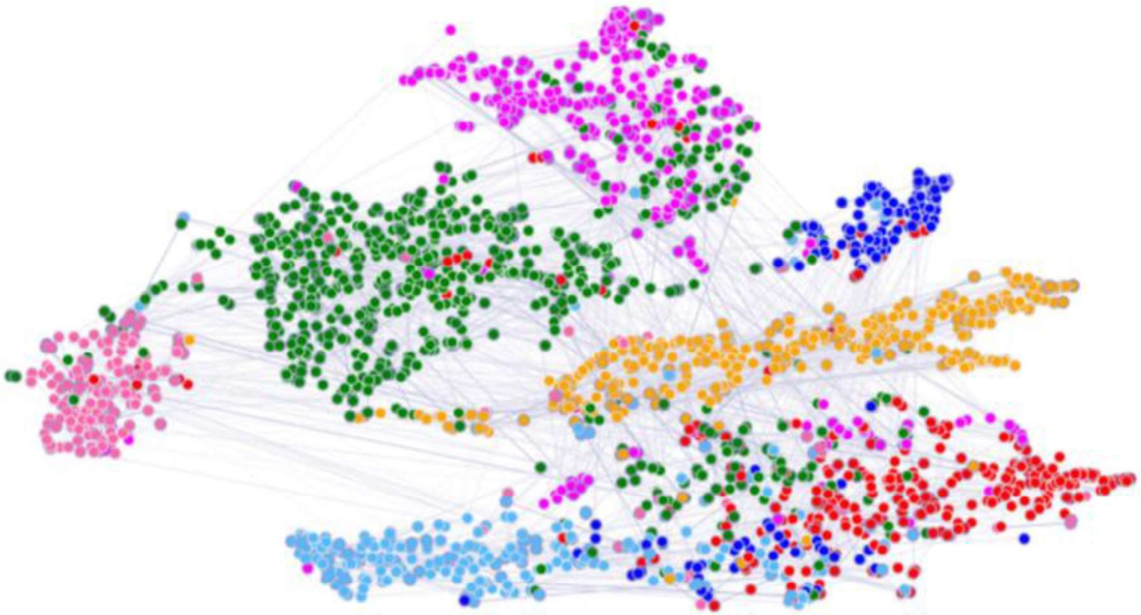
Transductive

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)	$81.7 \pm 0.5\%$	—	$78.8 \pm 0.3\%$
GCN-64*	$81.4 \pm 0.5\%$	$70.9 \pm 0.5\%$	$\mathbf{79.0} \pm 0.3\%$
GAT (ours)	$\mathbf{83.0} \pm 0.7\%$	$\mathbf{72.5} \pm 0.7\%$	$\mathbf{79.0} \pm 0.3\%$

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).

Data Mining Quality Analytics

- 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).

04 | Graph Attention Networks

Data Mining Quality Analytics

- 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

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