AoI-based Temporal Attention Graph Neural Network for Popularity Prediction and Content Caching

Introduction

Introduction

- **Under a constrained backhaul link with sharply growing data traffic**
  - **Bring a better quality of user experiences and quality of service to user is problem.**
  - **Though, one of the simple solution is larger scale antennas, setting more bandwidth(대역폭) → increasing the capacity of cellular networks.**
    - **But 대부분은 확장성(Scalability), 비용(Cost) 및 유연성(Flexibility) 측면에서 내구성 있는 솔루션을 제공하지 못함.**

- **Tremendous data load comes from the repeated requests for a few same popular targets**
  - **Storing part of fashionable content at the network edge is main goal. (=edge caching is the main problem)**

Introduction

❑ ICN(Information-Centric Network) mobile edge caching

- BS(edge node)는 다양한 content(copy본)을 가지고 있음.
- User's request content를 edge가 network storage에 보유하고 있다면 쉽게 제공 가능
- But limited cache space → recommendations are essential

❑ Traditional caching strategies

- Ignore the dynamic characteristics of requests

❑ Solution:

- GAT(Graph Attention Network)

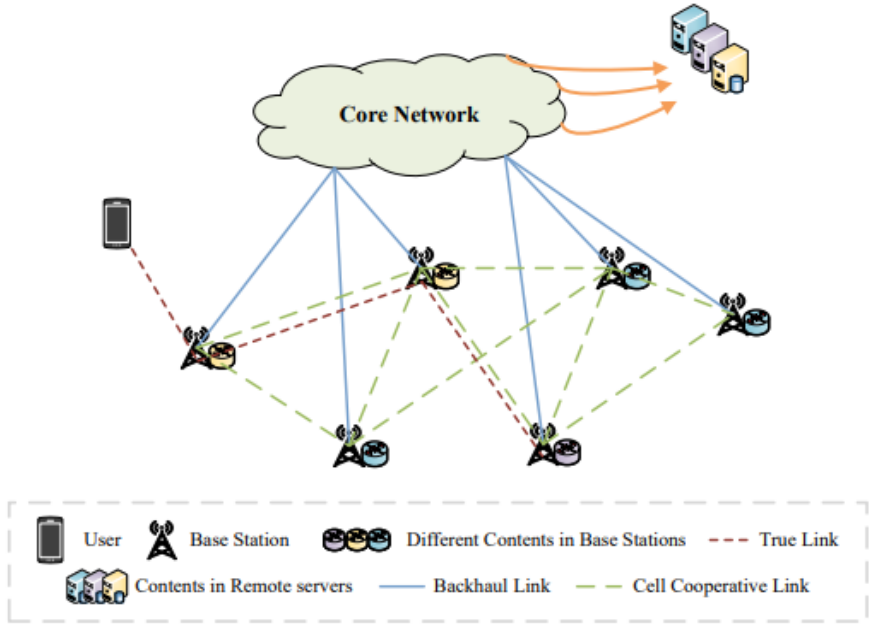


Fig. 1. Mobile edge caching in ICN.

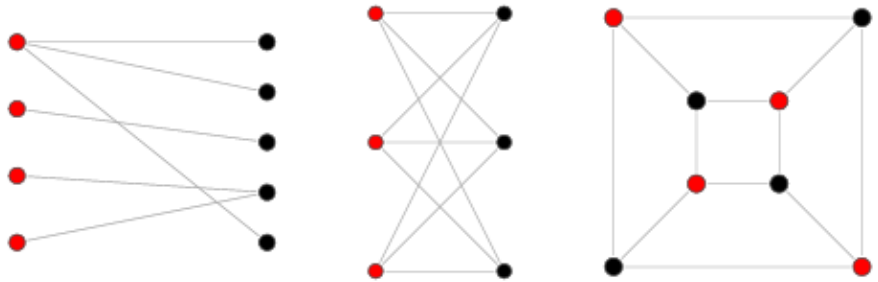
Introduction

❑ GAT

- **Interactions between the content and users could be also regarded as a dynamic bipartite graph(이분 그래프) when we attempt to predict their popularity in ICN**

❑ Bipartite Graph

이분 그래프(Bipartite Graph)란



인접한 정점끼리 서로 다른 색으로 칠해서 모든 정점을 두 가지 색으로만 칠할 수 있는 그래프.

- 즉, 그래프의 모든 정점이 두 그룹으로 나눠지고 서로 다른 그룹의 정점이 간선으로 연결되어져 있는(<=> 같은 그룹에 속한 정점끼리는 서로 인접하지 않도록 하는) 그래프를 이분 그래프라고 한다.

출처 : https://gmlwjd9405.github.io/2018/08/23/algorithm-bipartite-graph.html

Introduction

❑ Solution

- **DGNN(Dynamic Graph Neural Network) = CDGNN(Continuous Dynamic Graph Neural Network)**
  - **Learn the structural and temporal pattern in dynamic bipartite graph of users and requested content**
  - **Specifically, we focus on discovering "how to abstract temporal features" and "how many historical message" we should utilize while mining the dynamic features.**

- **AoI(Age of Information)**
  - **Guide the selection of fresh information**
  - **Use multi-head attention mechanism to refine temporal characteristics.**

❑ Contributions

- **사용자 선호도를 정확하게 예측 by CDGNN (bipartite graph의 structural, dynamical patterns를 동시에 파악)**
- **AoI-based temporal attention graph neural network(ATAGNN) 개발**
  - **effectively mine the temporal features in the dynamic graph.**

Related Work

Related Work

- 기존 GAT : ignore dynamic features

- Therefore, proposed DGNN :
  - DGNN models at the early stage achieve their ‘dynamics extraction’ by sampling series snapshots from the evolving graph with equal time intervals.

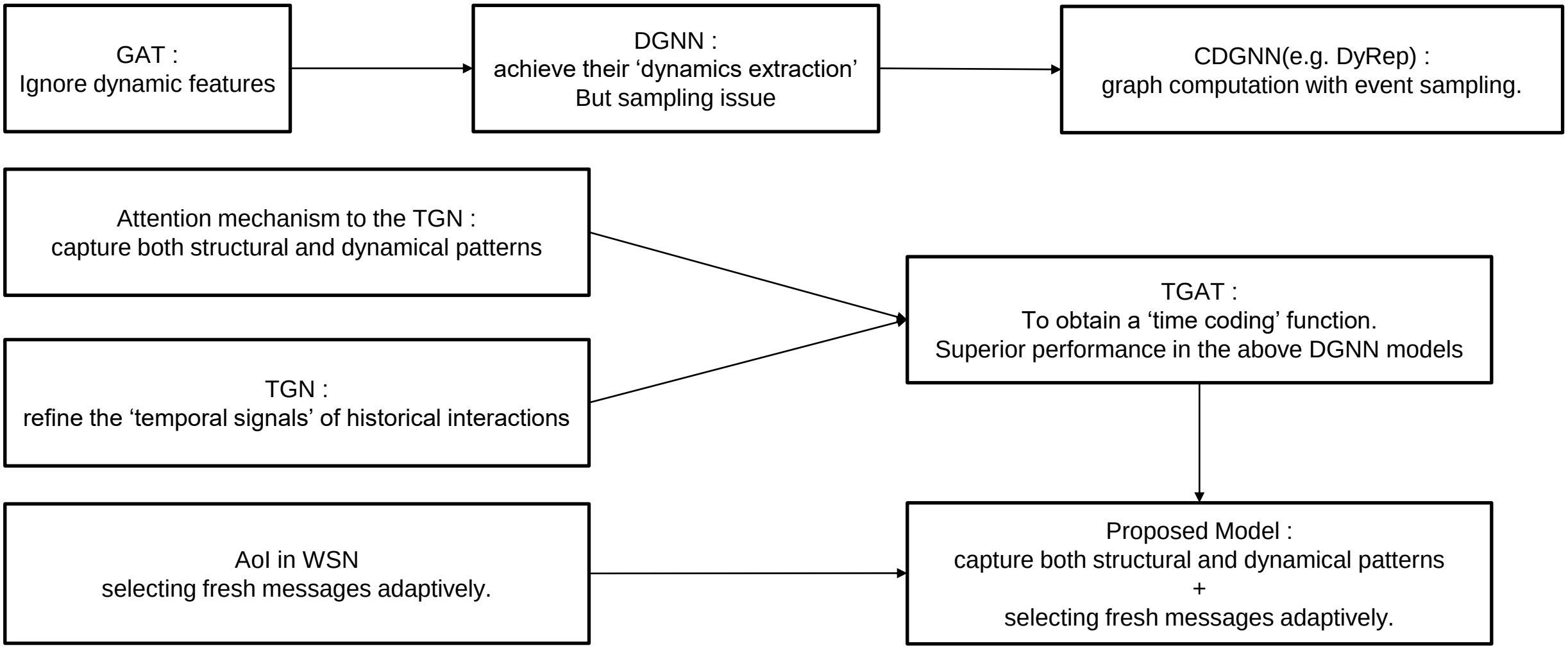
- Choice of sample is a problem.
  - May result in the failure of a snapshot with a new effective graph structure.
  - How to avoid? : CDGNN(Continuous Dynamic Graph Neural Network) → e.g.(=for example) DyRep
    - propose to complete the graph computation with event sampling.

- Try to inject information of interaction timestamps into the node = TGAT
- TGN(Temporal Graph Network) :
  - intends to refine the temporal signals of historical interactions by adding a memory module to the TGAT.
  - Achieves superior performance in the above DGNN models.

Related Work

- **TGN : important tool to realize the proactive caching.**
  - **But still leave questioned on how to design a much more effective information ‘aggregation method’ in CDGNN**
- **TGN + attention mechanism = can capture both structural and dynamical patterns**
- **AoI in WSN(Wireless Sensor Network)**
  - **To balance the huge data and the limited transmission capacity, introduce the concept into our NN for selecting fresh message adaptively.**

Related Work



System Model and Problem Formulation

System model overview

- $\mathcal{U} = \{u_0, u_1, \dots, u_J\} \rightarrow$ *set of users*
- $I(Item) = \{i_0, i_1, \dots, i_K\} \rightarrow$ *set of available contents in the network(edge server)*
- V → Node information(e.g., the age of a user or the category of content)
  - $V_u = \{v_{u_0}, v_{u_1}, \dots, v_{u_j}\}$
    - v_{u_j} : original features of user j
  - $V_I = \{v_{i_0}, v_{i_1}, \dots, v_{i_K}\}$
    - v_{i_K} : original features of content k
- $A^k(\delta_p, p_j^k)$: request actions(item will be requested or not)
- $Inf_{jk}\left(= f\left(v_{u_j}, v_{i_k}, e_{jk}\right)\right)$: *message between user j and content k* , (message = relationship)
- $\Phi_{d_T}(\Delta_t)$: time encoding
- E_j^u, E_k^i : Final representation of user j and content k

1. Time encoding module

→ $\Phi_{d_T}(\Delta_t)$

2. Time-concatenating module

- Combines (Initial message embedding Inf_{jk} , Encoded time feature($\Phi_{d_T}(\Delta_t)$))

→ $Msg_j(t)$

3. Memory embedding module(TGN)

- aggregates the historical interactions stored in the message buffer

→ structural characteristics for all edges in the graph

4. Embedding module

- Extract the structural pattern within the graph and merge the time feature

→ $\tilde{p}_j^k(\delta_P) = F(E_j^u, E_k^i)$

System Model

- **To realize the goal of caching**
  - compute content popularity for the users within the edge server and download the most popular content

- **J users, K contents**
  - $\mathcal{U} = \{u_0, u_1, \dots, u_J\} \rightarrow$ *set of users*
  - $I(Item) = \{i_0, i_1, \dots, i_K\} \rightarrow$ *set of available contents in the network(edge server)*

- **V → Node information(e.g., the age of a user or the category of content)**
  - $V_u = \{v_{u_0}, v_{u_1}, \dots, v_{u_J}\}$
    - v_{u_J} : original features of user j
  - $V_I = \{v_{i_0}, v_{i_1}, \dots, v_{i_K}\}$
    - v_{i_K} : original features of content k

System Model

- $p_j^k(\delta_P)$: Possibility(p) of content k requested by user j during the content updating period δ_P
  - DGNN model의 목적. Predict possibility.
- Caching 을 하기 위해서는 item will be requested or not 을 결정해야한다.
- Total request behaviors about content k for all users($\mathcal{U}$) within the edge server during the updating period(δ_P)
  - $A^k(\delta_p, p_j^k) = \sum_{j \in \mathcal{U}} 1(p_j^k(\delta_p) > P_I), \forall k \in I$
    - P_I : threshold. 즉 possibility가 P_I 보다 높아야 request의 가능성이 있다고 생각되며 function 1의 input으로 제공된다.
  - $A^k(\delta_p, p_j^k)$: predicted request actions.
- C : edge server's capability(=storage space)
  - $C(\delta_p)$: cached set under limited storage space C(popularity ranking list top-C items)
- $Hit\ rate:\ h(\delta_p) = \frac{\sum_{k \in C(\delta_p)} \tilde{A}^k(\delta_p, \bar{p}_j^k)}{\sum_{k' \in I} A^{k'}(\delta_p, p_j^k)}$ → 전체 request 중에 cache 되어 있는 개수.

Dynamic Graph Model

- Users and their interests are often subject to change over time
- $\mathcal{E} = \{e_{01}, e_{21}, \dots, e_{jk}\}$: interaction(=edge) between user j and content k
- Assume all the different files have an equal size
- New request(graph에 없는 요청) = generation of edge.
  - (R_{jk}, T_N) : Relation(j , k) and Timestamp(occurrence of edge)
    - $R_{jk} = \{v_{u_j}, v_{i_k}, e_{jk}\}$
  - G = $\{(R_{01}, T_1), (R_{21}, T_2), \dots, (R_{jk}, T_N)\}$

Dynamic Graph Model

- Inf: $interaction\ event$, f : $concatenation\ function$
  - $Inf_{jk}\left(= f\left(v_{u_j}, v_{i_k}, e_{jk}\right)\right)$: $message\ between\ user\ j\ and\ content\ k$
  - $Inf_{kj}\left(= f\left(v_{i_k}, v_{u_j}, e_{jk}\right)\right)$: $embedding\ of\ message$
  - Node information(user information, content information) : v_{u_j} , v_{i_k}
  - User : u_j , Item : i_k
    - What is message … 정확한 설명이 나와있진 않음

- Node u_j at time t
  - Request targets N(v_{u_j} ; t) = $\{v_{i_0}, v_{i_1}, \dots, v_{i_M}\}$
  - 사용자 u_j 가 request 할 목록 :

Temporal Graph Network Model for Popularity Prediction

❑ Consider temporal relationship.

- **Interval between the latest request and the target one is much more meaningful than the absolute time points.**
  - Δ_t **: Interval between the latest request and the target one.**
- **Time-coding module**
  - $\Phi_{d_T}(\Delta_t) = \sqrt{\frac{1}{d_T}}[(cos(w_1\Delta_t), cos(w_2\Delta_t), \dots, cos(w_d\Delta_t)]^T$
    - **w : parameters to be trained**
    - d_T **: dimension. (the number of the time embedding we want)**
    - **TGAT와 function이 동일하다는데… 설명x**
- **Time-concatenating module**
  - **Combines (Initial message embedding** Inf_{jk} **, Encoded time feature(**$\Phi_{d_T}(\Delta_t)$**))**
  - **Set of user j's all interactions**
    - $Msg_j(t) = [Inf_{j0}||\Phi_{d_T}(0), \dots, Inf_{jk}||\Phi_{d_T}(\Delta t_N)]^T \rightarrow$ *the final input feature to the DGNN model*
    - **Operator || : concatenation operation**

Temporal Graph Network Model for Popularity Prediction

❑ Consider temporal relationship.

- **Memory embedding module (inspired by TGN)**
  - The extraction of short-term and long-term interests of users.
  - It aggregates the historical interactions stored in the message buffer (=information about short-term preference)
  - But TGN aggregation methods are for extracting short-term temporal characteristic.
    - Therefore, we further propose to adopt AoI-based attention mechanism for utilizing raw request information(talk about later chapter)

- **Embedding module**
  - Extract the structural pattern within the graph and merge the time feature
  - Output : final representations(=contain both temporal and structural characteristics for all edges in the graph)
    - Final representation of user j and content k : E_j^u, E_k^i
    - Predicted preference of user j for content k : $\tilde{p}_j^k(\delta_P) = F(E_j^u, E_k^i) \rightarrow derive\ A^k(\delta_p, p_j^k) = \sum_{j \in u} 1(p_j^k(\delta_p) > P_I), \forall k \in I\ by\ Algorithm\ 1$
    - Function F : 다양한 알고리즘 가능. 해당 논문에서는 MLP.

❑ Loss function : BCE(Binary Cross Entropy)

System model overview

- $\mathcal{U} = \{u_0, u_1, \dots, u_J\} \rightarrow$ *set of users*
- $I(Item) = \{i_0, i_1, \dots, i_K\} \rightarrow$ *set of available contents in the network(edge server)*
- V → Node information(e.g., the age of a user or the category of content)
  - $V_u = \{v_{u_0}, v_{u_1}, \dots, v_{u_J}\}$
    - v_{u_j} : original features of user j
  - $V_I = \{v_{i_0}, v_{i_1}, \dots, v_{i_K}\}$
    - v_{i_K} : original features of content k
- $A^k(\delta_p, p_j^k)$: request actions(item will be requested or not)
- $Inf_{jk}\left(= f\left(v_{u_j}, v_{i_k}, e_{jk}\right)\right)$: *message between user j and content k* , (message = relationship)
- $\Phi_{d_T}(\Delta_t)$: time encoding
- E_j^u, E_k^i : Final representation of user j and content k

1. Time encoding module

→ $\Phi_{d_T}(\Delta_t)$

2. Time-concatenating module

- Combines (Initial message embedding Inf_{jk} , Encoded time feature($\Phi_{d_T}(\Delta_t)$))

→ $Msg_j(t)$

3. Memory embedding module(TGN)

- aggregates the historical interactions stored in the message buffer

→ structural characteristics for all edges in the graph

4. Embedding module

- Extract the structural pattern within the graph and merge the time feature

→ $\tilde{p}_j^k(\delta_P) = F(E_j^u, E_k^i)$

AoI-Based Temporal Graph Neural Network

AoI-Based Temporal Graph Neural Network

- **Aggregating history information can extract the short-term interests of users**

- **Aggregation in TGN are still far from perfection**
  - **Keeping the latest request may lead to some errors due to the lack of historical references.**
  - **Averaging method**
    - **ignores the fact that requests in different time have distinct influence on future behavior.**
    - **Out of date methods. Even that may bring adverse effects.**

- **Thus, we propose AoI-based attention mechanism**

Attention Mechanism for Temporal Pattern Extraction

- **TGN-A : TGN + Attention mechanism**
  - **we can calculate the degrees of correlation between previous interactions(=edge = message) and the latest one with a self-attention mechanism**

Msg : set of user j's all interactions.
- **Message embedding(user j, content k) + time encoding**

- **Aggregated message of user j**
  - $h_j(t) = ATT(Msg_j(t), [Msg_j(t)]_0, V) = \sigma(\sum_{m \in N} \alpha_{jm} V_{jm})$
    - $V_{jn} = [Msg_j(t)]_n * W_V$ **(W : weight matrix)**
    - α_{jm} **: attention coefficient(importance between the m-th history message and the most recent one)**
      - $softmax([Msg_j(t)]_m, [Msg_j(t)]_0)$ **:0 : latest one , m : m-th history message**

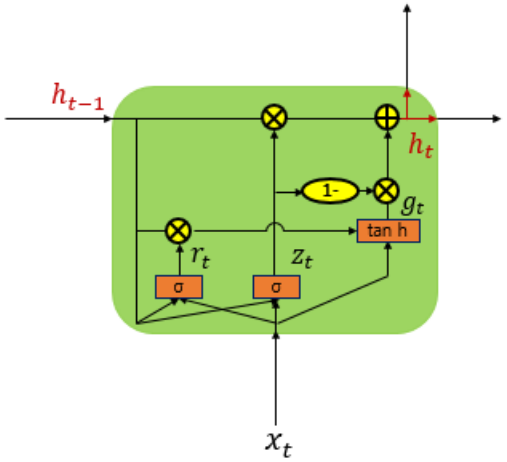
- **Inspired by GraphSAGE and TGN**
  - **Only sample the most recent N historical messages for aggregating.**
- **Inspired by ResNet**
  - **Introduce the 'skip-connection', x : base message($[Msg_j(t)]_0$)**

Attention Mechanism for Temporal Pattern Extraction

- **Inspired by ResNet**
  - $\overline{Msg_j(t)}$: output from FN(Feed forward Network with skip-connection)
    - Time-aware embedding at time t

- **Multi-head attention : avoid the instability of the training process**
  - L : the number of heads(In this paper, L=3)
  - $\overline{Msg_j(t)} = FN(h_j^{(1)}(t)||h_j^{(2)}(t)||\ldots||h_j^{(L)}(t)||R_{j0}) \rightarrow FN : derived\ from\ ResNet.\ Feed\ forward\ Network$

- **여기까지 extracting short-term interests with multi-head attention**



- **How to update ? GRU 사용 (다른 RNN 아키텍처도 비슷한 gain 확인)**
  - $Mem'_j = GRU\left(Mem_j, \overline{Msg_j(t)}\right) \rightarrow GRU(h, x)$
    - Contains all the history that we have chosen in aggregating module.
    - It can capture the long-term interest.

AoI-based Attention Mechanism

- **Excluding the information with an age that is too stale to be detrimental to the final prediction.**

- **TGN with AoI-based attention for temporal learning : ATAGNN**
  - **AoI :** $\Delta_{jk}(t,n) = t - B_{jk}(n)$
    - $B_{jk}(n)$ **: birth time of the nth request(between user j and content k)**
    - **t : current time**

- **Concatenate all the age data as an N-length vector(**a_j**)**
  - **nth request 이므로 길이가 N.**
  - **Pass** a_j **to a two-layer MLP module to finish the adaptive selecting task of the worthy information**

AoI-based Attention Mechanism

- **Two-layer MLP module : 둘 다 ReLU 사용.**
  - $h_{th} = MLP(a_j) = ReLU(a_j W^1 + b^1)$
  - $thre_t = MLP(h_{th})$
    - **Threshold. (모든 information age를 concat 후 MLP 통과 하였기 때문에 threshold)**

- **Offer a threshold for determining whether the information deserves to be taken into consideration or not.**

- **However, deviations inevitably(편차는 불가피)**
  - **'soft method' :** $t'^n_{jk} = t^n_{jk} * \sigma(100 * (\Delta_{jk}(t,n) - thre_t))$
    - t^n_{jk} **: raw timestamps (nth interaction between node j and node k)**
    - $thre_t$ **: threshold**
- **Mask all the requests with an age that is lower than the threshold time**

Future and Structural Patterns Embedding

- GAT 아키텍처 사용
  - But 기존 LeakyReLU 사용 x
  - Custom Dot-product 사용
    - $\alpha_{jk}^{g} = \frac{exp((\tilde{h}_k W_Q^g)^T(\tilde{h}_j W_K^g))}{\sum_{m \in N(v_j;t)} exp((\tilde{h}_m W_Q^g)^T(\tilde{h}_j W_K^g)}$
      - W_Q^g, W_K^g : relationship between time encoding and the output of Mem_j'(GRU update message)
      - $\tilde{h}_k$: $[Mem_j' \| \Phi_{d_T}(\Delta_t)] \rightarrow GRU(updated\ message)||time\ encoding$
  - More attention heads, the better structural representations will be extracted
- Node u_j at time t
  - Request targets N(v_{u_j} ; t) = $\{v_{i_0}, v_{i_1}, \dots, v_{i_M}\}$
  - 사용자 u_j 가 request 할 목록 :

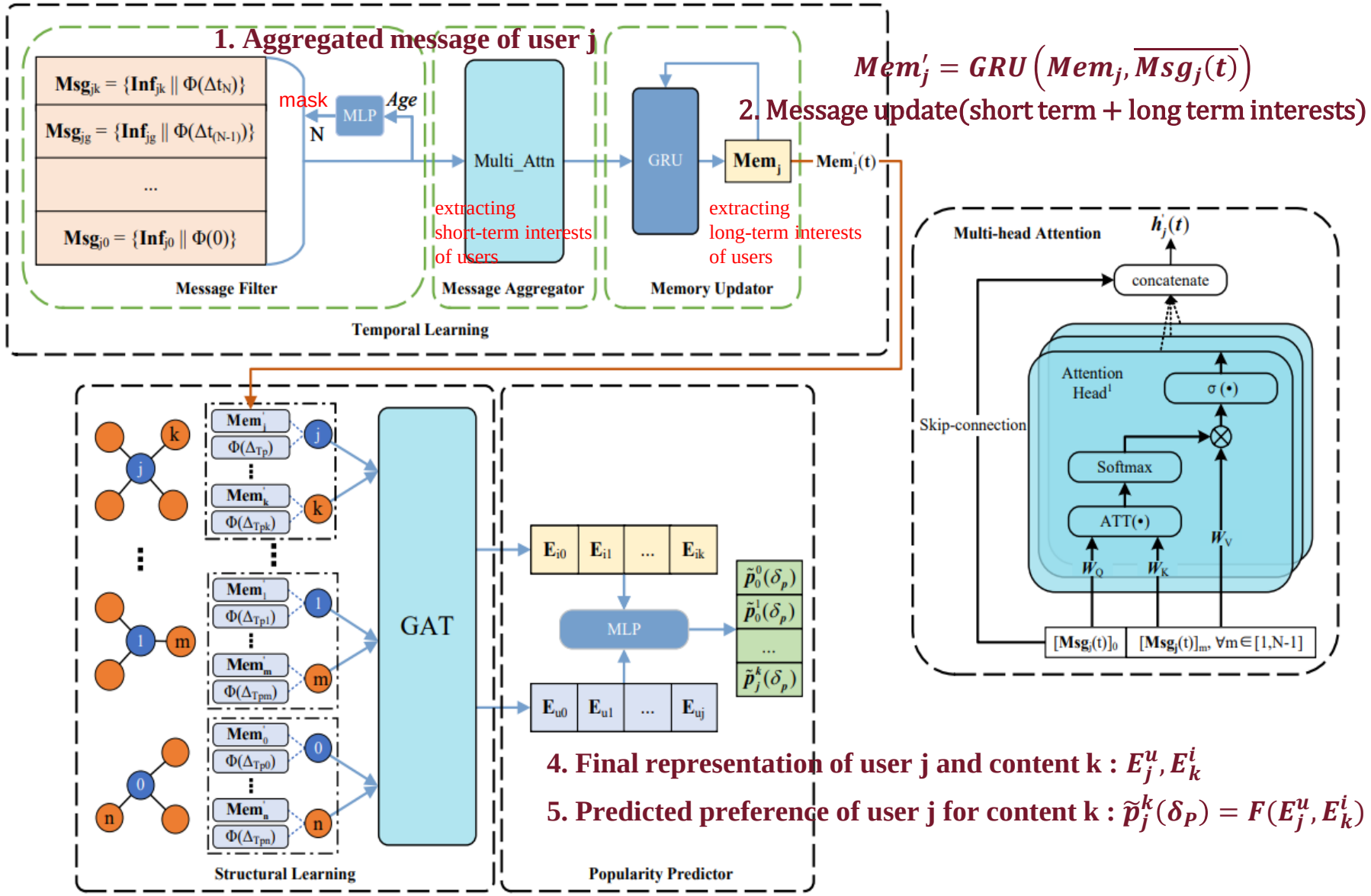


Fig. 3. The illustration of AoI-based temporal attention graph neural network and the multi-head attention mechanism.

Algorithm 1.

Algorithm 1 The preference prediction algorithm with AoI-based temporal attention GNN

Input: Dynamic request dataset;

Output: The presentations of users $\boldsymbol{E}_j^u$ and content $\boldsymbol{E}_k^i$. And predicate the preference between u_j and i_k p

Initialize the parameters for the whole network;
Initialize the memory buffer with zeros and message buffer.
Restore the graph information (**Inf** $\leftarrow$ all messages) and divide it into several mini batches;
for each batch$(\boldsymbol{v}_{u_j}, \boldsymbol{v}_{i_k}, \boldsymbol{e}_{ui}, t) \in$ training dataset **do**
  $\dot{n} \leftarrow$ Sample negatives;
  Calculate the age $\Delta_{jk}(t, n)$ and the threshold thre_t;
  Filter and concatenate the valuable messages $\mathbf{Msg}(t)$;
  Aggregate with multi-head attention mechanism in Eq. (7), (10) and obtain $\overline{\mathbf{Msg}_j(t)}$;
  Update features $\mathbf{Mem}_j$ in memory buffer with GRU in Eq. (11);
  Encode the time difference Δ_{T_p} with Eq. (3) for all nodes;
  Concatenate the encrypted time feature with $\mathbf{Mem}_j$, and obtained the new feature $\tilde{\boldsymbol{h}}_j$ as the input;
  Obtain $\boldsymbol{E}_j^u(T_p)$ and $\boldsymbol{E}_k^i(T_p)$ through the modified GAT;
  Predict the correlation degree between users and content with Eq. (5);
  Optimize this network with BCELoss$(\cdot)$;
end for

1. Calculate Age of information and threshold
    1. Mask all the request (graph update)
2. Aggregate with multi-head attention Mechanism
    1. Obtain $\overline{\boldsymbol{Msg}_j(t)}$
    2. ResNet의 skip-connection 포함
3. Msg update by using GRU
    1. Obtain $\boldsymbol{Mem}_j'$
    2. Do time encoding
4. (Concatenate $\boldsymbol{Mem}_j'$ and time encoding) as input to GAT
5. Through the modified GAT
    1. Obtain $\boldsymbol{E}_j^u, \boldsymbol{E}_k^i$
6. Optimize with BCE Loss

Simulation Results & Numerical Analysis

Simulation Results & Numerical Analysis

❑ Dataset

▪ Wikipedia Dataset

- The number of entries(items) and users is 9227(=nodes)
- 15,000 interactions(=edges)
- 70%, 15%, 15%(train, valid, test)

▪ MOOC Dataset

- 5763 users, 56 contents
- 175,856 interactions(=edges)
- 60%, 20%, 20%(train, valid, test)

Simulation Results & Numerical Analysis

❑ Experimental Setup

- **Transductive task(변환 작업):**
  - **evaluate model's ability of predicting the temporal links for those nodes that have been observed in the training phase**
- **Inductive task(유도 작업):**
  - **Evaluate model's ability of representing the nodes that have never been trained**
- **Max number of neighbors that we want to aggregate is 5**
- **Comparison between our model and TGN-A with a fixed age threshold(influence of AoI)**

Results Analysis

Results Analysis

Dataset		Wikipedia				MOOC			
Metric		Old AUC	Old AP	New AUC	New AP	Old AUC	Old AP	New AUC	New AP
Baseline	RNN	76.476	77.100	-	-	70.394	70.461	-	-
	DyRep	93.823	94.313	91.795	92.668	87.498	83.591	87.201	83.493
	TGAT	95.391	95.710	93.241	93.810	74.413	69.932	73.211	69.282
	TGN-L	98.406	98.470	97.700	97.805	92.026	89.855	92.447	90.494
	TGN-M	98.342	98.426	97.741	97.864	90.951	88.423	92.271	90.319
TGN-A	5n	98.481	98.559	97.806	97.911	93.091	90.944	92.994	91.014
	10n	98.509	98.600	97.923	98.044	93.088	91.078	92.737	90.783
	15n	98.508	98.589	97.909	98.029	93.384	91.280	93.103	91.090
ATAGNN without Eq. (15)	5n	98.514	98.591	97.925	98.035	93.183	91.150	93.109	91.111
	10n	98.530	98.605	97.883	98.013	93.554	91.603	93.310	91.421
	15n	98.510	98.594	97.955	98.074	93.541	91.516	**93.424**	**91.478**
ATAGNN with Eq. (15)	5n	**98.544**	98.625	97.915	98.026	93.362	91.396	93.302	91.394
	10n	98.526	98.610	97.941	**98.066**	**93.577**	**91.635**	93.330	91.376
	15n	98.539	**98.632**	**97.957**	98.061	93.568	91.572	93.269	91.300

TGN-L vs TGN-M
The number of aggregated information is an important factor

AUC(Accuracy) , AP(Average Precision)

Old? New?

$$Eq.(15)\ t'^{n}_{jk} = t^{n}_{jk} * \sigma\left(100 * \left(\Delta_{jk}(t,n) - thre_t\right)\right)$$

Age of Information

Results Analysis

TABLE III

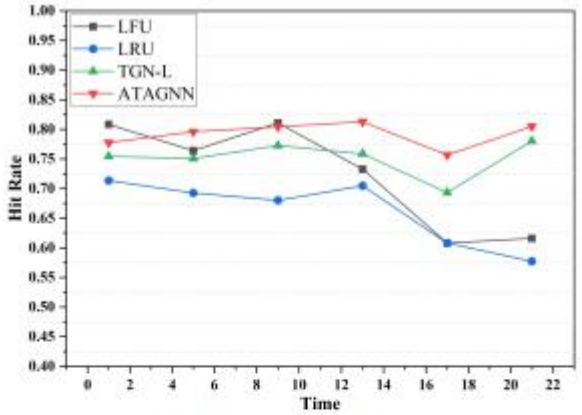
THE RESULT OF TGN-A WITH FIXED AGE THRESHOLD (I.E., 60S, 600S, 3600S), THE ORIGIN MODEL WITHOUT CONSIDERING THE AGE AND THE ATAGNN MODEL WITH MLP FOR ADAPTIVELY CHOOSING THRESHOLD. THE BEST RESULTS IN EACH COLUMN ARE HIGHLIGHTED IN **BOLD** FONT

Datasets		MOOC			
Metric		Old AUC	Old AP	New AUC	New AP
TGN-A with Age	60	92.363	90.191	92.234	90.004
	600	92.695	90.612	92.686	90.696
	3600	93.088	91.026	92.792	90.641
TGN-A without Age		93.091	90.944	92.994	91.014
ATAGNN		**93.362**	**91.396**	**93.302**	**91.394**

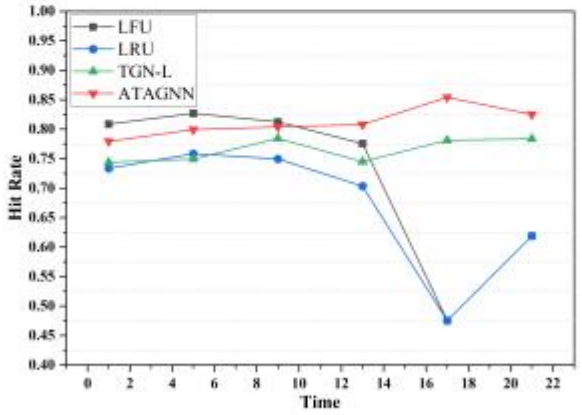
TGN에 absolute age threshold 도입 = 도입 안한게 더 낫다.
But Adaptive age threshold(by 2 layer MLP) 도입 = good

$$Eq.(15)\ t'^{n}_{jk} = t^{n}_{jk} * \sigma\left(100 * \left(\Delta_{jk}(t,n) - thre_t\right)\right)$$ **Age of Information**

Results Analysis



(a) Transductive Task



(b) Inductive Task

Fig. 4. 24-hour hit rate performance of MOOC datasets with different algorithms in Transductive and inductive tasks.

Hit rate within 23 hours with a cache space of 15.

Result:

- **ATAGNN based caching is able to keep superior for a long time**

Results Analysis

Hit rate of our caching policy with different maximum cache sizes (i.e., 5, 10, 15, 20)

Result:

- **ATAGNN always good.**

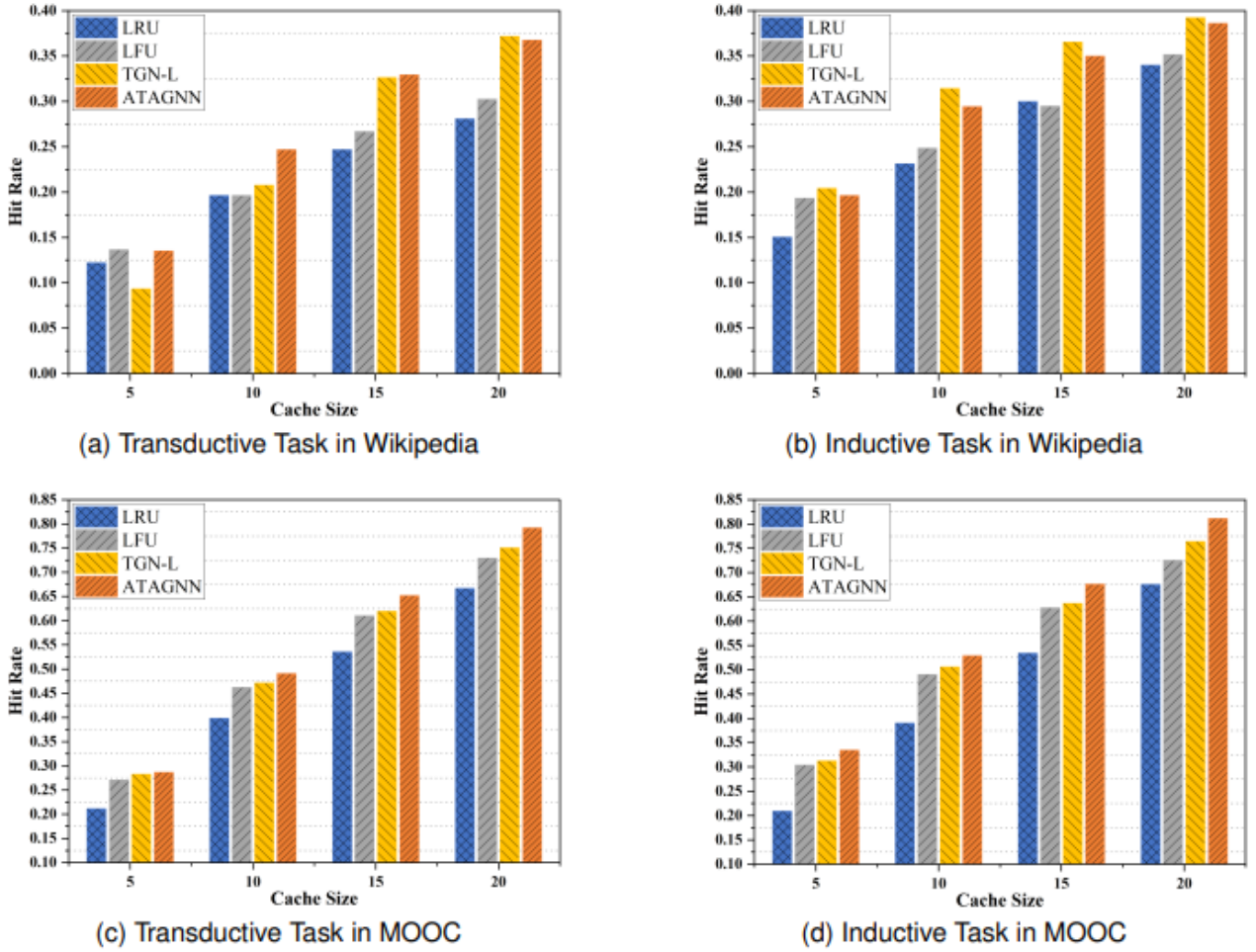


Fig. 5. Hit rate comparison with different cache sizes in different datasets.

Results Analysis

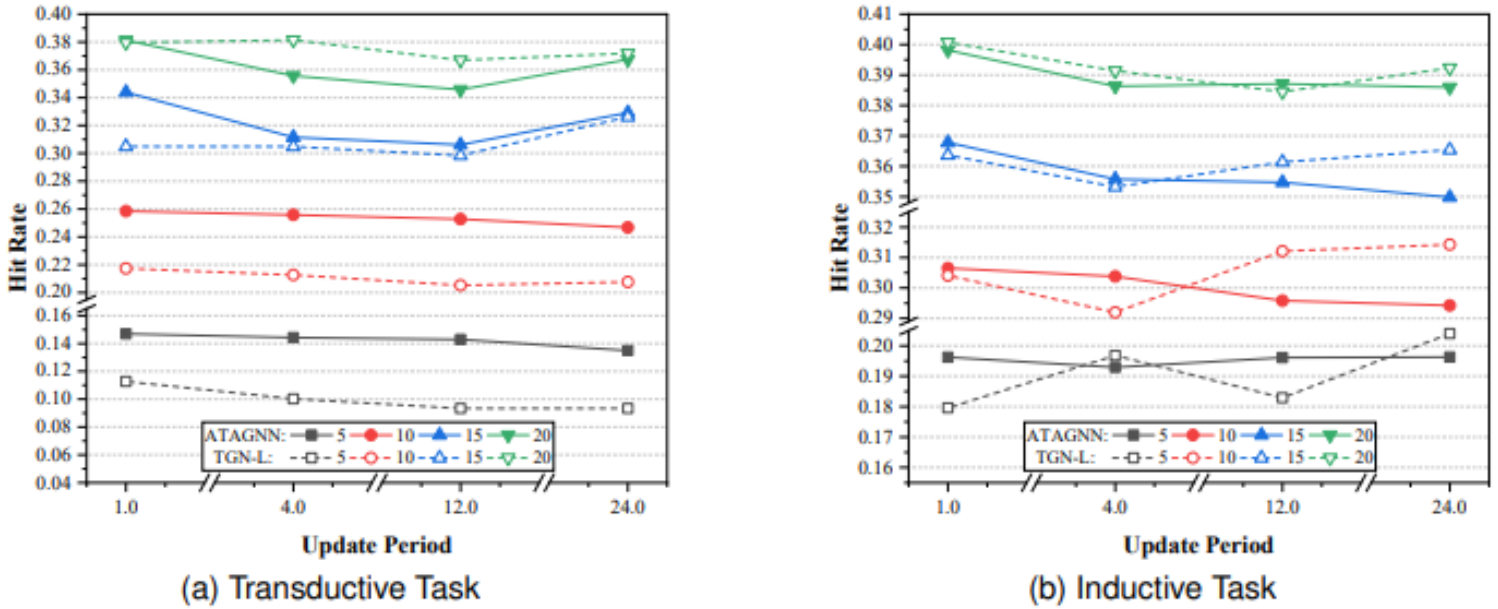


Fig. 6. Hit rate comparison with different update periods in Wikipedia.

ATAGNN vs TGN-L (various update periods)

Result:

- **The performance of both models gradually decreases with the increase of update period**
- **ATAGNN은 update period 변화에 따라 크게 꺾인다**
  relies more on the history messages and an appropriate memory update

Conclusion

Conclusion

- **Develop AoI-based Temporal Attention Graph Neural Network(ATAGNN)**
  - **Maximize the precision of user's interest prediction**

- **The concept of AoI is specifically introduced to exclude stale information for better refining history**

Simulation Results & Numerical Analysis

Simulation Results & Numerical Analysis

❑ Caching Policy Setting

- **Have little prior knowledge about the users, content and the possible timestamps of events**
  - **Inspired by LFU, if the number of candidates is too large, only choose those entities that have been observed in last one or two hours.**

- **Algorithm 2 참고**
  - **It caches the contents from all candidate items by counting the possible accessible actions with Eq. (1)**
    - **Eq. (1)** $A^k(\delta_p, p_j^k) = \sum_{j \in u} 1(p_j^k(\delta_p) > P_I), \forall k \in I$ $A^k(\delta_p, p_j^k)$ **: predicted request actions.**
  - **Also, generate 'fake requests' to download the popular content in advance**
    - **May mislead our subsequent prediction.**
    - **Thus, 일정 주기마다 memory update(called memory update period** T_u**)**

- **Compare LRU vs LFU**
  - **LRU : updates the caching by replacing the content that has not been requested for a longest time**
  - **LFU : try to keep those that have been most requested(not related to time)**

- **Compare ATAGNN(our model) vs TGN-L**
  - **to show the superiority of our ATAGNN over other models in caching**

Simulation Results & Numerical Analysis

❑ Cache-Hit Efficiency

- With in 23 hours and the cache size is set as 15.
- Default model's updating period for memory buffer is 24 hours.
  - Generate all results at once. Never update the model's memory in our simulation
  - Thus, test different memory update periods(24, 12, 4, 1 hours)
- User's preference to the content every 6s (with flexible thresholds) and list the ranking of the popularity for each hour.