PAPER Special Section on Information and Communication Technology to Support Hyperconnectivity

Pattern-Based Meta Graph Neural Networks for Argument Classifications

Shiyao DING[†a)] ***and*** **Takayuki ITO**[†b)]**,** ***Members***

SUMMARY Despite recent advancements in utilizing meta-learning for addressing the generalization challenges of graph neural networks (GNN), their performance in argumentation mining tasks, such as argument classifications, remains relatively limited. This is primarily due to the underutilization of potential pattern knowledge intrinsic to argumentation structures. To address this issue, our study proposes a two-stage, pattern-based meta-GNN method in contrast to conventional pattern-free meta-GNN approaches. Initially, our method focuses on learning a high-level pattern representation to effectively capture the pattern knowledge within an argumentation structure and then predicts edge types. It then utilizes a meta-learning framework in the second stage, designed to train a meta-learner based on the predicted edge types. This feature allows for rapid generalization to novel argumentation graphs. Through experiments on real English discussion datasets spanning diverse topics, our results demonstrate that our proposed method substantially outperforms conventional pattern-free GNN approaches, signifying a significant stride forward in this domain.

key words: *argument mining, argument classification, graph neural networks, meta learning*

1. Introduction

Argument classification is a fundamental task in the field of argumentation mining (AM) that involves the automatic classification of argument labels [1]–[4]. Given that argumentation elements exhibit a strong relational inductive bias [5], graph neural networks (GNN) are often used in argument classification tasks as they are adept at learning the latent graphical information inherent in argumentation structures.

However, a significant challenge arises when employing GNN in argumentation tasks—adapting trained GNN models to new argumentation graphs, a situation referred to as the 'generalization problem'. This arises because each argumentation data set corresponds to a unique argumentation graph, and even minor alterations on the graph can cause trained GNN models to fail when applied to the new argumentation graph. Meta-learning, a method that strives to establish a model that can adapt the learned model to various tasks, is an efficient solution to this generalization problem. Meta-learning based GNN methods like Meta-GNN [6], G-META [7], and Sub-Meta [8] have been proposed to address the GNN models' generalization problem. However, their effectiveness is usually limited when directly applied to argumentation mining tasks. This limitation is primarily due to the inherent predefined patterns within argumentation structures (such as the relationships of argumentation elements with specific labels), a factor that these methods neglect. For instance, the argumentation structures in platforms like D-Agree [9] often follow a topic-comment-reply pattern. Thus, acknowledging and utilizing such pattern knowledge could enhance GNN's performance in AM tasks.

In contrast to the pattern-free GNN methods, we introduce a two-stage, pattern-based meta-GNN method for argument classification. In the first stage, we propose a pattern representation algorithm to capture the pattern knowledge which is subsequently used to predict edge types in argumentation graphs. In the second stage, based on the predicted labels for all edges, a relational-GCN (RGCN) [10] is employed to efficiently identify different edge relationships for the prediction of argument labels. Additionally, we implement a meta-learning framework, model-agnostic meta-learning (MAML), that comprises an inner and an outer loop. Within the inner loop, each topic maintains a topic-specific GNN that is trained using the data from that particular topic. In the outer loop, a meta-learner that updates its parameters by considering the update directions of all topic-specific GNN models from the inner loop. Finally, we introduce a pattern-based Weisfeiler-Lehman (PWL) test to ascertain whether two graphs share a common pattern and analyze our proposed method's computational complexity. Finally, we evaluate our method on two cross-topic discussion data sets. The experimental results clearly indicate that our proposed method significantly outperforms traditional pattern-free GNN methods.

2. Related Work

2.1 Meta-Learning for GNN

Graph convolutional networks (GCN), as a prominent GNN model, have demonstrated remarkable capabilities in classic GNN tasks such as node classification and link prediction. The Relational-GCN (RGCN) [10] further extends the capabilities of GCN for accommodating multiple link relationships. It incorporates the information of link type into the convolution layer using a weight, effectively dealing with graphical data that possesses heterogeneous relationships.

Many research has been carried out to address the generalization problem inherent in GNN models by applying meta-learning techniques. For instance, Meta-GNN [6] ef-

Manuscript received June 2, 2023.
Manuscript revised October 14, 2023.
Manuscript publicized December 11, 2023.

[†]The authors are with the Graduate School of Informatics, Kyoto University, Kyoto-shi, 606–8501 Japan.
a) E-mail: ding@i.kyoto-u.ac.jp
b) E-mail: ito@i.kyoto-u.ac.jp

DOI: 10.1587/transinf.2023IHP0013

fectively handles node classification problems across a range of graphs. It views the learning process on each graph as a separate task, with each graph corresponding to a distinct GNN model. Following a model-agnostic meta-learning (MAML) approach for graph meta-learning, the Meta-GNN incorporates the update directions of all graphs within its meta-learner. Similarly, G-META [7] tackles the issue of scalability in GNN models, particularly when the graph is too extensive to be inputted into the GNN entirely. Instead, it uses sub-graphs for meta-learning. However, these meta-learning-based GNN models often fail to consider the specific characteristics of argumentation structures, resulting in diminished efficiency when implemented in AM tasks.

2.2 Argumentation Mining (AM)

The field of argumentation mining (AM) focuses on the automatic extraction of structured arguments from unstructured textual documents [4], with learning-based methods gaining significant traction in recent years. For instance, Stab et al. employed the support vector machine method to conduct argument classification tasks using a persuasive essay data set. They classified arguments into one of three types: Major Claim, Claim, and Premise [11]. Similarly, Suzuki et al. [12] investigated an argument classification task on a persuasive essay data set using the Issue-Based Information System (IBIS) structure, a traditional argumentation-based approach for addressing complex problems [13]. In IBIS, there are four types of arguments: Topic, Issue, Idea, Pros, and Cons. They used a graph attention network (GAT) to achieve high accuracy. However, these studies primarily focus on persuasive essay data, overlooking discussion data.

Besides argument classification, GNN-based models can be effectively applied to other AM tasks. For instance, Kuhlmann et al. [14] worked on abstract argumentation semantics tasks, which involve finding a mapping that accepts an abstract argumentation framework as input and outputs a binary label representing the acceptability of all arguments under a specific semantic. They employed a GCN-based classifier to approximate acceptance under preferred semantics with an average class accuracy of approximately 0.61. Subsequently, Craandijk et al. [15] proposed a recurrent-GCN based learning algorithm for prediction tasks that achieved higher accuracy. However, these algorithms primarily focus on graph-level tasks, which aim to predict whether a graph is true or false and computation complexity corresponds to $O(1)$. In contrast, our work focuses on predicting the label of each argument, corresponding to a computational complexity of $O(|V|)$, which is considerably more complicated.

There are also some pattern-based GNN methods. [16] delved into the development of a tree kernel that emphasizes common substructures, or fragments. While they introduced a novel GNN architecture, their focus wasn't on the overarching generalization issues inherent to GNNs. In the case of [17], their research proposed the meta-path concept aiming to encapsulate broader abstract ideas. They introduced a heterogeneous graph neural network called MeGnn. However, their primary objective was the optimization of heterogeneous graph neural networks, which is a departure from our central focus on homogeneous graphs. Lastly, [18] made strides in generating GNN models that cater to subgroups rather than individual nodes. Yet, their work did not delve into GNN generalization challenges, specifically, they didn't explore areas such as zero-shot and one-shot learning, which are pivotal in our research.

3. Problem

In this section, we state the generalization problem (meta-graph learning tasks) of argument classification. First, we recall the definition of Dung's abstract argumentation frameworks as follows [19].

Definition 1. An abstract argumentation framework (AF) is a pair $\mathcal{G}^i =< V^i, E^i >$ where i is the index of AF, $V^i = \{v_1^i, v_2^i, \ldots, v_{|V^i|}^i\}$ is a (finite) set of arguments and $E^i \subseteq V^i \times V^i$ is the attack relation. The pair $(v_j^i, v_k^i) \in E^i$ means that v_j^i attacks v_k^i. A set $V_s^i \subseteq V^i$ attacks v_k^i if there is an $v_j^i \in V_s^i$, such that $(v_j^i, v_k^i) \in V^i$. An argument $v_j^i \in V^i$ is defended by $V_s^i \subseteq V^i$ iff, for each $v_j^i \in V^i$ such that $(v_k^i, v_j^i) \in E^i, V_s^i$ attacks v_k^i.

Although only attack relations are considered in Dung's AF, the types of relations can be extended further. Also, each argument is assigned a label and we denote L^i as a set of all labels in $\mathcal{G}^i$, i.e., $lab_j^i \in L^i$. Moreover, we define a function $\mathrm{Lab}^i : V^i \to L^i$ to identify a label for each node, i.e., $\mathrm{Lab}^i(v_j^i) = lab_j^i$.

Graph Set We regard each AF $\mathcal{G}^i =< V^i, E^i >$ as a directed graph. As for edge (v_j^i, v_k^i), v_j^i is called the source node and v_k^i is called the target node. We consider there is a set $\mathbf{G} = \{\mathcal{G}^1, \ldots, \mathcal{G}^{|\mathbf{G}|}\}$ including multiple graphs. In meta-graph learning tasks of argument classification, $\mathbf{G}$ is divided into train set $\mathbf{G}^{train}$ and test set $\mathbf{G}^{test}$, i.e., $\mathbf{G} = \mathbf{G}^{train} \cup \mathbf{G}^{test}$, and also each task $\mathcal{G}^i$ can be divided into a support set and a query set as follows.

$$\begin{aligned} \mathcal{G}_{sup}^i &=< V_{sup}^i, E_{sup}^i > \\ \mathcal{G}_{que}^i &=< V_{que}^i, E_{que}^i > \end{aligned} \tag{1}$$

Take argument classification as an instance shown in Fig. 1, the nodes with labels are support set used for training and the other nodes without labels are query set used for testing.

Based on this, the goal is to find a Meta-GNN model trained on $\mathbf{G}^{train}$ which is expected to perform well on query set $\{\mathcal{G}_{que}^i | \mathcal{G}_{que}^i \in \mathbf{G}^{test}\}$ after training few steps on support set $\{\mathcal{G}_{sup}^i | \mathcal{G}_{sup}^i \in \mathbf{G}^{test}\}$ of test set $\mathbf{G}^{test}$.

Specifically, in the argument classification task, given the graph $\mathcal{G}^i =< V^i, E^i >$ without label information, the goal is to predict each node's label. It means to find an approximate function $\hat{Lab}$ of Lab that outputs each node's

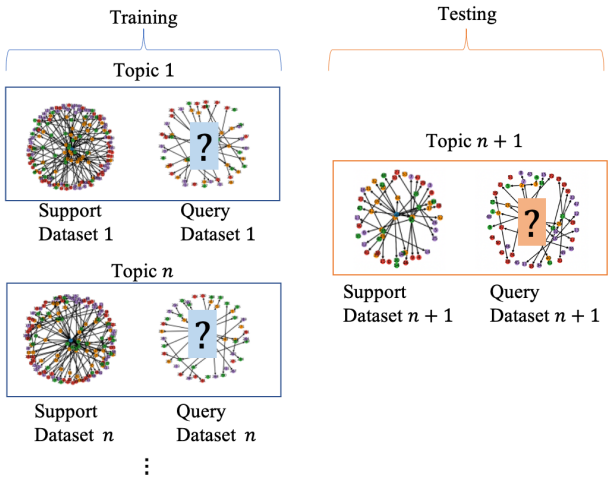


Fig. 1 An example of meta argument classification task.

label accurately, i.e., $\hat{Lab}(v_j^i|\theta) = lab_j^i$, and the object is to minimize the following loss defined by

$$loss(\mathbf{G}^{test}) = \sum_{\mathcal{G}_{que}^i \in \mathbf{G}^{test}} \sum_{v_j^i \in \mathcal{G}_{que}^i} entropy(\hat{Lab}(v_j^i|\theta) - lab_j^i) \quad (2)$$

As for meta-learning tasks we consider its three classic tasks:

- **Few-shot learning** It divides each graph in the testing set into support set and query set, it makes the meta-model learn a few epochs on support set and test on query set.
- **One-shot learning** It is similar to few-shot learning besides that only one epoch is trained on the support set rather than a few shots.
- **Zero-shot learning** It takes the whole set of graph as a query set and tests the trained model without training.

4. Algorithm

In this section, we introduce the two-stage, pattern-based meta-GNN methods. In stage 1, we focus on learning a high-level pattern representation to effectively capture the pattern knowledge within an argumentation structure, followed by predicting the edge type. It then utilizes a meta-learning framework in the second stage, designed to train a meta-learner based on the predicted edge types.

4.1 Stage 1: Pattern Representation

In argumentation mining, the pattern of an argumentation structure is usually predefined. For instance, in IBIS the node with the *Issue* label can only be linked with the node with label of the *Idea*. Then, utilizing such kind of pattern knowledge would benefit AM tasks. Before defining the pattern of an argumentation structure, we first define meta-link label as follows, where one node with a specific type label should link with some nodes with other specific type labels.

Definition 2 (**Meta-Link Label**). we define the label of a link $< v_j, v_k >$ is determined by its end nodes' labels $< Lab(v_j), Lab(v_k) >$

Based on the definition of the meta-link label, the definition pattern is given by

Definition 3 (**Pattern**). A pattern can be defined as follows. $\mathcal{AS} =< L, \mathcal{E}^L >$ where L is the label set and $\mathcal{E}^L$ denotes the set of meta-link labels $< lab^j, lab^k >$ with $lab^j, lab^k \in L$

For instance of IBIS, $L = \{Topic, Issue, Idea, Pros, Cons\}$ and $\mathcal{E}^L = \{(Topic, Issue), (Issue, Idea), (Idea, Pros), (Idea, Cons)\}$

Pattern Representation Algorithm Since we have defined how to represent a pattern of an argumentation structure, the corresponding algorithm is stated as follows. Our idea is to update each node's label by its one-hop neighborhood node labels iteratively. Then, through k-iterations updating, each node would include its k-hop neighborhood node information, which is a unique representation of the graph.

Specifically, a pattern representation algorithm is defined on the meta-link label where each label is assigned a corresponding node and its source nodes and target nodes are given. Then, the node labels from source nodes and target nodes are updated separately, according to the following algorithm.

- **Iterative update of label** Given an argumentation structure $\mathcal{AS} =< L, \mathcal{E}^L >$, assign a node v^l for each label $l \in L$ and the set of nodes is denoted as V^L. Then, the label updating starts from iteration 0, i.e., $Lab(v^l)^{ite}|_{ite=0} = \{l^0\}$, where each node v^l receives its source node labels and target node labels separately, with the iteration number ite at each iteration. For all nodes $v^l \in V^L$, each label is updated as follows.

$$Lab^{Nei}(v^l)^{ite+1} = Lab^{Nei}(v^l)^{ite} \bigcup_{v^j \in Nei(v^l)} \left(Lab^{Nei}(v^l)^{ite} \cap Lab^{Nei}(v^j)^{ite} \right) \quad (3)$$

  where $Nei(v^l)$ can be $Sou(v^l)$, the set of all source nodes of v^l, or $Tar(v^l)$, the set of all target nodes of v^l. Since $Lab^{Nei}(v^l)^{ite}$ is a set, it only adds the labels that currently do not exist during each iteration.
- **Termination of update** The above update of node labels will terminate until the set of all node labels does not change after iteration, i.e.,

$$Lab^{Nei}(v^l)^{ite} = Lab^{Nei}(v^l)^{ite+1}, \qquad \forall v^l \in V^L \quad (4)$$

We denote the label of node v^l at final iteration as compressed labels $Lab^{Nei}(v^l)^{fin}$, i.e., $Lab^{Nei}(v^l)^{ite}|_{ite=fin} = Lab^{Nei}(v^l)^{fin}$ and $L^{\mathcal{AS}} = \{L(v^l) = \{Lab^{Sou}(v^l)^{fin}, Lab^{Tar}(v^l)^{fin}\} | v^l \in V^L\}$ where the first element is the set of

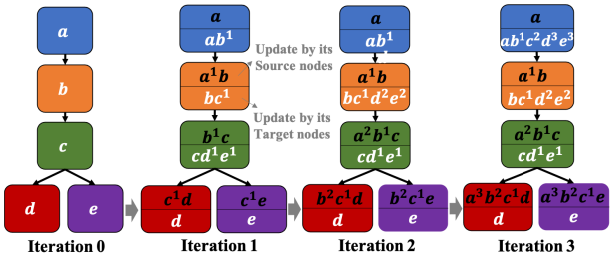


Fig. 2 An example of pattern representation.

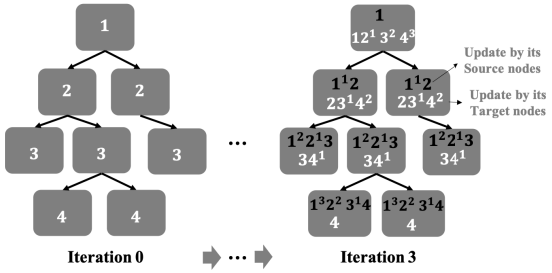


Fig. 3 An example of pattern representation without label information.

compressed labels updated by source nodes, and the second element is the set of compressed labels from target nodes. As shown in Fig. 2, the structure has nodes a, b, c, d, e and edges $(a,b),(b,c),(c,d),(c,e)$. After following the update, the compressed labels are given by $L(a) = \{\{a\},\{ab^1c^2d^3e^3\}\}$, $L(b) = \{\{a^1b\},\{bc^1d^2e^2\}\}$, $L(c) = \{\{a^2b^1c\},\{cd^1e^1\}\}$, $L(d) = \{\{a^3b^2c^1d\},\{d\}\}$, $L(e) = \{\{a^3b^2c^1e\},\{e\}\}$. According to the final compressed labels, there are four link types,i.e., $L^{\mathcal{AS}}=$

- $(\{\{a\},\{ab^1c^2d^3e^3\}\},\{\{a^1b\},\{bc^1d^2e^2\}\})$,
- $(\{\{a^1b\},\{bc^1d^2e^2\}\},\{\{a^2b^1c\},\{cd^1e^1\}\})$,
- $(\{\{a^2b^1c\},\{cd^1e^1\}\},\{\{a^3b^2c^1d\},\{d\}\})$
- $(\{\{a^2b^1c\},\{cd^1e^1\}\},\{\{a^3b^2c^1e\},\{e\}\})$.

Pattern Representation without Label Algorithm Then, we state how to predict edge types in a graph without node label information.

- **Identify root node** Given a graph without $\mathcal{G} = < V, E >$, identify a node without any target node as the root node.
- **Arrange to tree structure** Arrange the graph to a tree-structure according to the distance (n-hop) to the root node. The root node's k-hop neighbor nodes are set at k-th layer. Then, the nodes are assigned the same label if they are at the same layer.
- **Updating** Following pattern representation algorithm to update all node labels to obtain final labels $L^{\mathcal{G}} = \{Lab(v)^{fin} | v \in V\}$.

Based on the above compressed node labels, we have three types of links in Fig. 3, i.e., $L^{\mathcal{G}}=$

- $(\{\{1\},\{12^13^24^3\}\},\{\{1^12\},\{23^14^2\}\})$
- $(\{\{1^12\},\{23^14^2\}\},\{\{1^22^13\},\{34^1\}\})$
- $(\{\{1^22^13\},\{34^1\}\},\{\{1^32^23^14\},\{4\}\})$.

Since it is known that this argumentation graph follows the pattern in Fig. 2, the goal is then to identify a function $f_{inj} : L^{\mathcal{G}} \rightarrow L^{\mathcal{AS}}$ that to map the labels in the pattern for all the links as follows.

$$min \quad \sum_{l \in L^{\mathcal{G}}} \mathcal{I}\left(f_{inj}(l) - L^{\mathcal{AS}}(l)\right) \tag{5}$$

where $\mathcal{I} = n$ and n is the number of elements with different representations of the edges. The error summation is calculated based on "$l \in L^{\mathcal{G}}$" rather than "$l \in L^{\mathcal{AS}}$", that is because some graphs are generated based on some pattern but may leak some label nodes. As for the uncertain relationship such as $(1^22^13, 1^32^23^14)$ can map to $(a^2b^1c, a^3b^2c^1d)$ or $(a^2b^1c, a^3b^2c^1e)$, there two methods to tackle it. The first is to choose one of the types deterministically. The second is to give a probability of each type of link. Specifically, predicting certain edge types becomes challenging when there's a symmetric topology within the pattern. For instance, with a symmetric topology, the probability of predicting each edge type in that topology drops to 50%. The accuracy further diminishes as the number of symmetric topologies increases. In this paper, we employ the first type method.

4.2 Stage 2: Meta Learning Based RGCN

Relational-GCN (RGCN) Since the types of links have been predicted, we consider to use RGCN, a GNN model that can well cope with heterologous edge relationships, to capture node features to process an argument classification. Specifically, given a graph $\mathcal{G} =< V, E >$, each node v_i can be embedded to a numeric vector such as Word2Vec, Universal sequence embedding (USE) or Bidirectional encoder representations from transformers (BERT). We use $v_{i,e} \in \mathbb{R}^d$ to denote the embedding of each node where d is the dimension of the embedding. Correspondingly, we can construct a feature matrix $X \in \mathbb{R}^{|V| \times d}$ where each row represents one node's embedding. we denote h_i^l as the embedding vector of node $v_i \in V$ at layer l of RGCN. At the first layer, we denote it as an embedding result of USE, i.e., $h_i^0 = v_{i,e} = USE(v_i)$. We consider all the edge types (relationships) $r \in L^{AS}$ based on pattern-based prediction. Then each node would use features $\{h_j^l | j \in Sou^r(i)$ of its source nodes to update its feature at each layer.

$$h_i^{(k+1)} = \sigma\left(\sum_{r \in L^{AS}} \sum_{j \in Sou^r(i)} \frac{1}{c_{i,r}} W_r^{(k)} h_j^{(k)} + W_0^{(k)} h_i^{(k)}\right) \tag{6}$$

where the $\frac{1}{c_{i,r}}$ denotes the weight of relationship r for node v_i; W_r^k and W_0^l are the weight matrix for relationship r and itself at layer k. That means each layer has $|L^{AS}| + 1$ parameter matrices and totally $K * (|L^{AS}| + 1)$ matrices are required to be trained in a RGCN with K layers. We let the final layer h^K pass a liner layer to output the probability of each label for all nodes, i.e., $h^K \in \mathbb{R}^{|V| \times |Lab|}$.

Meta-learning As for training RGCN models, we empoly MAML to train it, which aims at learning a meta model with a set of initial parameters that can efficiently adapt to the new task. It consists of two loops where each task has a specific model and it will update in the inner loop based on its own support set. In the **inner loop**, we sample a batch $\mathcal{B}$ from training set $\mathbf{G}^{train}$, i.e., $\mathcal{B} = \{\mathcal{G}^b | \mathcal{G}^b \in \mathbf{G}^{train}\}$ and update the parameter for each model using each graph's support set $\mathcal{G}^b_{sup}$ in the batch.

$$\theta_b \leftarrow \theta_b - \alpha \nabla_\theta loss_{G_b}(f_\theta) \tag{7}$$

where the $loss$ function is defined in Eq. (2). Then testing the updated parameter on query set and the meta-learner would consider all query set parameter updating directions to update. It has a good generalization ability to multiple gradient descent tasks. In the **outer loop** of meta-GNN, we use the GNN with updated parameters to train it further on query sets of $\mathcal{G}^i_{que}$. Then the meta-learner updates by considering the summation of all updating directions.

$$\theta^{meta} \leftarrow \theta^{meta} - \beta \nabla_\theta \sum_{G_i \sim \mathbf{G}^{train}} loss_{G_i}\left(f_{\theta'_i}\right) \tag{8}$$

4.3 Analysis

In this section, we analyze some properties of pattern-based RGCN. Weisfeiler-Lehman (WL) test is a classic method to judge whether two graphs are isomorphic. It follows the following steps. $Aggregation: \quad L_i = \{lab(v_j) | v_j \in nei(v_i)\}_{mul}$, each node has a label and then each node would aggregate the labels of its neighborhood as a multiset denoted as $\{\}_{mul}$. Then it requires finding a hash function to convert L_i to a new "compressed" label, $Combination: L_i[ite+1] \leftarrow hash(L_i[ite])$ where $hash$ is an injective function mapping $L_i[ite]$ to $L_i[ite+1]$. The above aggregation and combination processes would be repeated until the compressed labels of two graphs do not change, i.e., $L_i[ite] = L_i[ite+1]$. Once the two figures all have the same labels, the possibility of the two graphs being isomorphic would be high. Based on the WL-test, we define a pattern-based WL-test (PWL-test) to judge whether two graphs share the same pattern defined in Definition 2.

Pattern-based WL-test (PWL-test) Aggregation: We consider each node to have a label and then each node would aggregate the labels of its neighborhood as a set.

$$Aggregation: \quad L_i = \{lab(v_j) | v_j \in nei(v_i)\} \tag{9}$$

where there do not exist repeated labels in L_i. This can be regarded as what kind of labels exist in the neighbor nodes.

Then we find a hash function to convert L_i to a new "compressed" label.

$$Combination: \quad L_i[ite+1] \leftarrow hash(L_i[ite]) \tag{10}$$

Thus, we can see the key difference between WL-test is aggregating the label types rather than node labels. Compared with WL-test which is based on edge level (each edge is determined by its end nodes), PWL-test is based on meta-edge level (each meta-edge is determined by its end nodes' labels). Thus we can have the following theorem.

Theorem 1. Once two argumentation graphs pass the PWL-test, it means they share the same pattern and the meta model trained by pattern-based RGCN can be applied to the argumentation graphs.

Also, we analyze the computational complexity of our proposed method.

Theorem 2. The complexity of pattern representation algorithm is $O\big(|V|(D+1)\big)$.

Proof. Given a graph with $|V|$ nodes, all the node labels will be updated once at an iteration. The total iteration number depends on the longest path in the graph, which would be the node with the longest distance (hop number) from the root node. Thus, if we denote the longest distance D hop, the pattern representation algorithm would stop at $(D+1)$-th iteration, which corresponds to computational complexity of $|V|(D+1)$. □

5. Evaluation

5.1 Evaluation Setting

In this paper, we use IBIS as an instance to conduct experiments. Unlike Dung's argumentation structure, multiple types of relations exist such as relations of $Idea$ 'solve' $Issue$. We perform the meta graph learning tasks stated in section of Problem on the following two data sets.

- **Strict-IBIS English-Discussion Data Set** The data set is collected by ten conversations from 5 negative English speakers in 2022 [20]. Each conversation has a topic and each argument belongs to one of IBIS labels. Each conversation nearly includes 250 arguments.
- **Strict-IBIS English-Discussion Data Set with Isolate Argument** This data set is collected by ten conversations from 11 negative English speakers in 2021. Each conversation nearly includes 200 arguments. Although each argument graph follows an IBIS structure, some arguments lack its corresponding link which is called isolate arguments.

In the above two data sets, each topic-based conversation is regarded as an argument graph and then is divided into support data set and query data set according to a ratio of 4 : 1. We apply cross-validation where each graph i can be as a test graph and the other $-i$ graphs are as a train data set. Correspondingly, we list the test result on each graph and compare their average accuracy over ten graphs. We then compared our proposed method with two classic pattern-free GNN methods: GCN and Meta-GCN, which are illustrated as follows.

- **GCN** [21] method consists of one input layer with 1024 neurals, 6 hidden convolutional layers with 512 neurals

Table 1 Compare the performances of GCN and meta-GCN on IBIS english-discussion data set

Topic:	1	2	3	4	5	6	7	8	9	10
GCN	.19	.02	.09	.15	**.38**	.18	.40	.32	.35	.13
Meta-GCN	.19	.14	.09	.15	.35	.08	.23	.24	.23	.13
Pattern-Based Meta-RGCN	**.58**	**.70**	**.60**	**.74**	.35	**.41**	**.67**	**.68**	**.65**	**.67**

(a) Zero-Shot Learning Task

Topic:	1	2	3	4	5	6	7	8	9	10
GCN	.31	.25	.46	.28	.10	.08	.23	.32	.23	.33
Meta-GCN	.31	.14	.09	.15	.25	.16	.03	.24	.21	.13
Pattern-Based Meta-RGCN	**.58**	**.73**	**.60**	**.74**	**.35**	**.41**	**.67**	**.68**	**.65**	**.63**

(b) One-Shot Learning Task

Topic:	1	2	3	4	5	6	7	8	9	10
GCN	.25	.25	.09	.17	.10	.08	.17	.32	.16	.13
Meta-GCN	.31	.36	.09	.38	.25	.14	.40	.32	.51	.50
Pattern-Based Meta-RGCN	**.79**	**.86**	**.91**	**.74**	**.46**	**.61**	**.70**	**.65**	**.74**	**.93**

(c) Few-Shot Learning Task (10 epochs)

Topic:	1	2	3	4	5	6	7	8	9	10
GCN	.25	.25	.09	.17	.10	.08	.17	.32	.16	.13
Meta-GCN	.31	.36	.09	.38	.25	.14	.40	.32	.51	.50
Pattern-Based Meta-RGCN	**.83**	**.89**	**.89**	**.87**	**.87**	**.69**	**.90**	**.65**	**.79**	**.90**

(d) Few-Shot Learning Task (100 epochs)

Table 2 Compare the performances of GCN and meta-GCN on IBIS english-discussion data set with isolate argument.

Topic:	1	2	3	4	5	6	7	8	9	10
GCN	.33	.24	.29	.23	.29	.26	.14	.17	.02	.02
Meta-GCN	.12	.17	.15	.31	.29	.14	.28	.17	.15	.27
Pattern-Based Meta-RGCN	**.79**	**.68**	**.79**	**.54**	**.48**	**.71**	**.56**	**.76**	**.71**	**.67**

(a) Zero-Shot Learning Task

Topic:	1	2	3	4	5	6	7	8	9	10
GCN	.33	.24	.29	.23	.29	.26	.14	.17	.02	.02
Meta-GCN	.12	.17	.15	.31	.29	.14	.28	.17	.15	.27
Pattern-Based Meta-RGCN	**.88**	**.73**	**.82**	**.58**	**.55**	**.69**	**.58**	**.76**	**.76**	**.67**

(b) One-Shot Learning Task

Topic:	1	2	3	4	5	6	7	8	9	10
GCN	.33	.24	.29	.23	.29	.26	.14	.17	.02	.02
Meta-GCN	.12	.17	.15	.31	.29	.14	.28	.17	.15	.27
Pattern-Based Meta-RGCN	**.82**	**.85**	**.82**	**.73**	**.81**	**.77**	**.79**	**.83**	**.78**	**.69**

(c) Few-Shot Learning Task (10 epochs)

Topic:	1	2	3	4	5	6	7	8	9	10
GCN	.58	.54	.35	.54	.29	.57	.40	.55	.66	.42
Meta-GCN	.52	.66	.56	.50	.55	.60	.44	.57	.68	.48
Pattern-Based Meta-RGCN	**.85**	**.88**	**.68**	**.85**	**.84**	**.83**	**.77**	**.76**	**.85**	**.79**

(d) Few-Shot Learning Task (100 epochs)

and output layer with 5 neurals. Under cross-validation, each test graph corresponds to one GCN model which is independently trained and evaluated. Specifically, each graph trained on the support set of each graph and tested on the query set as the final result.

- **Meta-GCN** [6] We applied GCN as the GNN model in Meta-GNN and MAML is employed as a meta-learning framework. For each test graph $\mathcal{G}^i$, we keep all the graphs other than graph $\mathcal{G}^i$ as training data set, i.e., $\mathcal{G}^{-i} = \mathcal{G} - \mathcal{G}^i$. After training, we applied Meta-GCN to test graph $\mathcal{G}^i$.

5.2 Result Analysis

In the training phase, we ran all methods for 1000 epochs. The results of the meta-learning tasks on the two data sets are detailed in Tables 1 and 2. In regards to the comparison result of zero-shot learning tasks shown in Tables 1 (a) and 2 (a), both meta-GCN and GCN have underperformed. The reason is that meta-GCN, which inherently has the potential to quickly adapt to new tasks, requires training on the support data set. However, in zero-shot learning, the meta-model is directly applied to the query set without any prior training on the support set.

As seen in Tables 1 (b) and 2 (b), meta-GCN performs better than GCN even with training for only 1 epoch. This advantage grows with the number of learning epochs. As shown in Tables 1 (c)(d) and 2 (c)(d), we see that meta-GCN shows an improvement of around 30% over GCN, demonstrating that meta-learning can maintain a good generalization capability for new argumentation graphs.

However, our proposed structure-based learning performs well across all types of meta-learning graph tasks. It even achieves an average accuracy of over 60% on zero-shot learning tasks. Compared to meta-GCN, this suggests that pattern-based knowledge can provide the GNN model with a strong initial performance without any training, indicating good generalization ability across different graphs. This generalization capability tends to improve along with the training epochs. As demonstrated in Tables 1 (c)(d) and 2 (c)(d), it achieves high accuracy after training for only 10 and 100 epochs.

Specifically, we present some sample cases to elucidate the experiments. The Fig. 4 demonstrates the one-shot learning outcome for topic 1 in Table 1. Both GCN and Meta-GCN predict the labels of all nodes as "Issue," yielding an accuracy of 0.31. In contrast, our proposed method attains an accuracy of 0.58 after a single training epoch. Notably, the centrally located nodes are predicted with high accuracy, aligning with the Topic Issue Idea labels. However, the prediction accuracy for Cons and Pros is lower. This is attributed to the pattern-based stage where Cons and Pros exhibit a symmetric topology, complicating the distinction of edge types compared to labels without a symmetric topology. Additionally, we showcase the few-shot learning results (based on 10 shots) in the Fig. 5. While the accuracy of our proposed method elevates to 0.79, the outcomes from GCN and Meta-GCN exhibit minimal variation.

Considering the results from Tables 1 and 2, we com-

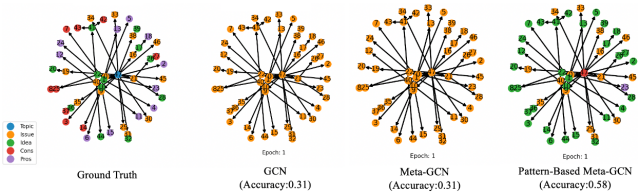


Fig. 4 The case study of topic 1 in Table 1 about one-shot learning task.

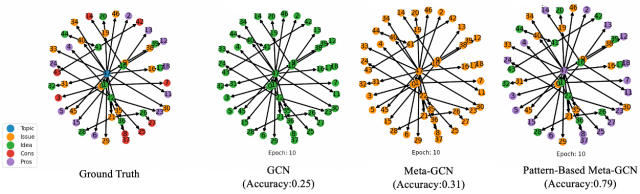


Fig. 5 The case study of topic 1 in Table 1 about few-shot (10epochs)learning task.

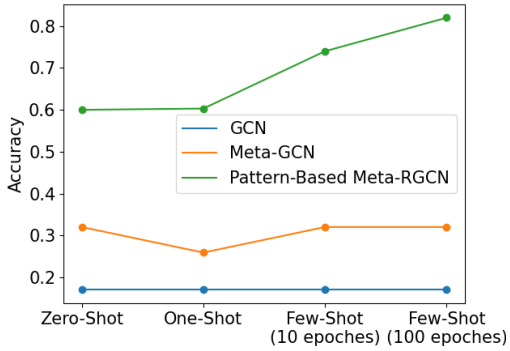


Fig. 6 Comparison of average accuracies: Results on the IBIS English-Discussion Data Set.

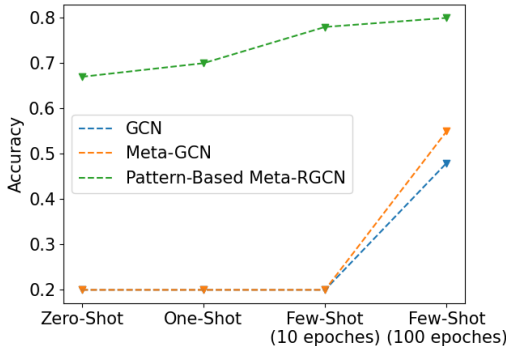


Fig. 7 Comparison of average accuracies: Results on the IBIS English-Discussion Data Set with Isolated Argument.

puted the average accuracy over ten graphs, as depicted in Figs. 6 and 7. The performance of the three algorithms improves with the number of learning epochs. From the results of the zero-shot learning tasks, we observe that our proposed method maintains high accuracy for new argumentation, exhibiting good generalization ability, and achieves an accuracy of 80% for few-shot learning tasks on both data sets.

6. Conclusion

In this paper, we tackled the generalization problem in the argument classification problem. Unlike most of the existing pattern-free GNN methods, we proposed a pattern-based meta GNN model that can make trained GNN model fastly adapt to new argumentation graphs. We also proposed a new pattern-based WL-test to judge whether two graphs share the same pattern and processed the theoretic analysis of our proposed method. The experiment results show great advantages over pattern-free based GNN methods. Regarding the experimental setting, we employ cross-validation to circumvent the overfitting problem. Although the results slightly fluctuate depending on the different graphs, the difference is not significantly large. From this, we can infer that the overfitting problem does not persist in our proposed method. Additionally, we tested two IBIS datasets to affirm this conclusion.

Although we use argumentation data with IBIS structure as an instance, our proposed pattern-based RGCN is a general method that can be extended to other argument data with a predefined argumentation structure. In the future work, it is imperative to scrutinize the applicability and robustness of our proposed method across various other structures and datasets.

Acknowledgements

This work was supported by JST CREST Grant Number JPMJCR20D1, Japan and a Grant-in-Aid for Scientific Research (C) (23K11230, 2023–2026) from the Japan Society for the Promotion of Science (JSPS).

References

[1] X. Hua and L. Wang, "Efficient argument structure extraction with transfer learning and active learning," Findings of the Association for Computational Linguistics: ACL 2022, Dublin, Ireland, pp.423–437, Association for Computational Linguistics, May 2022.
[2] A. Peldszus and M. Stede, "From argument diagrams to argumentation mining in texts: A survey," International Journal of Cognitive Informatics and Natural Intelligence (IJCINI), vol.7, no.1, pp.1–31, 2013.
[3] E. Cabrio and S. Villata, "Five years of argument mining: a data-driven analysis.," IJCAI, pp.5427–5433, 2018.
[4] M. Lippi and P. Torroni, "Argumentation mining: State of the art and emerging trends," ACM Transactions on Internet Technology (TOIT), vol.16, no.2, pp.1–25, 2016.
[5] P.W. Battaglia, J.B. Hamrick, V. Bapst, A. Sanchez-Gonzalez, V. Zambaldi, M. Malinowski, A. Tacchetti, D. Raposo, A. Santoro, R. Faulkner, et al., "Relational inductive biases, deep learning, and graph networks," arXiv preprint arXiv:1806.01261, 2018.
[6] F. Zhou, C. Cao, K. Zhang, G. Trajcevski, T. Zhong, and J. Geng, "Meta-gnn: On few-shot node classification in graph meta-learning," Proc. 28th ACM International Conference on Information and Knowledge Management, pp.2357–2360, 2019.
[7] K. Huang and M. Zitnik, "Graph meta learning via local subgraphs," Advances in Neural Information Processing Systems, vol.33, pp.5862–5874, 2020.
[8] A. Sankar, X. Zhang, and K.C.-C. Chang, "Meta-gnn: Metagraph

neural network for semi-supervised learning in attributed heterogeneous information networks," Proc. 2019 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining, pp.137–144, 2019.

[9] T. Ito, S. Suzuki, N. Yamaguchi, T. Nishida, K. Hiraishi, and K. Yoshino, "D-agree: crowd discussion support system based on automated facilitation agent," Proc. AAAI conference on artificial intelligence, vol.34, no.09, pp.13614–13615, 2020.

[10] M. Schlichtkrull, T.N. Kipf, P. Bloem, R.v.d. Berg, I. Titov, and M. Welling, "Modeling relational data with graph convolutional networks," European semantic web conference, pp.593–607, Springer, 2018.

[11] C. Stab and I. Gurevych, "Parsing argumentation structures in persuasive essays," Computational Linguistics, vol.43, no.3, pp.619–659, 2017.

[12] S. Suzuki, T. Ito, A. Moustafa, and R. Hadfi, "A node classification approach for dynamically extracting the structures of online discussions," The 34th Annual Conference of the Japanese Society for Artificial Intelligence, vol.2G5-ES-3, no.02, pp.1–4, 2020.

[13] W. Kunz and H.W. Rittel, Issues as elements of information systems, Citeseer, 1970.

[14] I. Kuhlmann and M. Thimm, "Using graph convolutional networks for approximate reasoning with abstract argumentation frameworks: A feasibility study," International Conference on Scalable Uncertainty Management, pp.24–37, Springer, 2019.

[15] D. Craandijk and F. Bex, "Deep learning for abstract argumentation semantics," arXiv preprint arXiv:2007.07629, 2020.

[16] F. Ruggeri, M. Lippi, and P. Torroni, "Tree-constrained graph neural networks for argument mining," arXiv preprint arXiv:2110.00124, 2021.

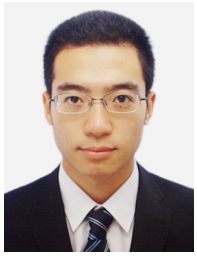
[17] Y. Chang, C. Chen, W. Hu, Z. Zheng, X. Zhou, and S. Chen, "Megnn: Meta-path extracted graph neural network for heterogeneous graph representation learning," Knowledge-Based Systems, vol.235, p.107611, 2022.

[18] Z. Luo, J. Lian, H. Huang, H. Jin, and X. Xie, "Ada-gnn: Adapting to local patterns for improving graph neural networks," Proc. Fifteenth ACM International Conference on Web Search and Data Mining, pp.638–647, 2022.

[19] P.M. Dung, "On the acceptability of arguments and its fundamental role in nonmonotonic reasoning, logic programming and n-person games," Artificial intelligence, vol.77, no.2, pp.321–357, 1995.

[20] Y. Dong, S. Ding, J. Haqbeen, and T. Ito, "The significant factors that affect the accuracy on classifying english ibis datasets," The 85th National Convention of IPSJ, March 2023.

[21] T.N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," arXiv preprint arXiv:1609.02907, 2016.



Shiyao Ding is an assistant professor in Graduate School of Informatics from Kyoto University, Japan. He received the Master degree in engineering from Osaka University, Japan in September 2019 and the Ph.D. degree from Kyoto University, Japan in September 2022. His current research interests include reinforcement learning, graph neural networks, multiagent systems, argumentation mining and services computing.

Takayuki Ito is a professor and head of Department of Social Informatics of Kyoto University. He received the Doctor of Engineering from the Nagoya Institute of Technology in 2000. He was a JSPS research fellow, an associate professor of JAIST, and a visiting scholar at USC/ISI, Harvard University, and MIT twice. He was a board member of IFAAMAS, the PCchair of AAMAS2013, PRIMA2009, GeneralChair of PRIMA2014, IEEE ICA2016, is the Local Arrangements Chair of IJCAI2020, and was a SPC/PC member in many top-level conferences (IJCAI, AAMAS, ECAI, AAAI, etc). He received the JSAI Contribution Award, the JSAI Achievement Award, the JSPS Prize, the Fundamental Research Award of JSSST, the Prize for Science and Technology of the Commendation for Science and Technology by the Minister of Education, Culture, Sports, Science, and Technology (MEXT), the Young Scientists' Prize of the Commendation for Science and Technology by the MEXT, the Nagao Special Research Award of IPSJ, the Best Paper Award of AAMAS2006, the 2005 Best Paper Award of JSSST, and the Super Creator Award of 2004 IPA Exploratory Software Creation Project. He was a JST PREST Researcher, and a principal investigator of the Japan Cabinet Funding Program for Next Generation World-Leading Researchers. He is currently principal investigator of his 2nd JST CREST project.