Argument Mining with Graph Representation Learning

Gechuan Zhang
gechuan.zhang@ucdconnect.ie
School of Computer Science,
University College Dublin
Dublin, Ireland

Paul Nulty
p.nulty@bbk.ac.uk
Department of Computer Science and
Information Systems,
Birkbeck, University of London
London, UK

David Lillis
david.lillis@ucd.ie
School of Computer Science,
University College Dublin
Dublin, Ireland

ABSTRACT

Argument Mining (AM) is a unique task in Natural Language Processing (NLP) that targets arguments: a meaningful logical structure in human language. Since the argument plays a significant role in the legal field, the interdisciplinary study of AM on legal texts has significant promise. For years, a pipeline architecture has been used as the standard paradigm in this area. Although this simplifies the development and management of AM systems, the connection between different parts of the pipeline causes inevitable shortcomings such as cascading error propagation.

This paper presents an alternative perspective of the AM task, whereby legal documents are represented as graph structures and the AM task is undertaken as a hybrid approach incorporating Graph Neural Networks (GNNs), graph augmentation and collective classification. GNNs have been demonstrated to be an effective method for representation learning on graphs, and they have been successfully applied to many other NLP tasks. In contrast to previous pipeline-based architecture, our approach results in a single end-to-end classifier for the identification and classification of argumentative text segments. Experiments based on corpora from both the European Court of Human Rights (ECHR) and the Court of Justice of the European Union (CJEU) show that our approach achieves strong results compared to state-of-the-art baselines. Both the graph augmentation and collective classification steps are shown to improve performance on both datasets when compared to using GNNs alone.

CCS CONCEPTS

• **Applied computing** → **Law**; • **Computing methodologies** → **Natural language processing**.

KEYWORDS

Argument Mining, Graph Neural Networks, Legal Text

ACM Reference Format:
Gechuan Zhang, Paul Nulty, and David Lillis. 2023. Argument Mining with Graph Representation Learning. In *Nineteenth International Conference on Artificial Intelligence and Law (ICAIL 2023), June 19-23, 2023, Braga, Portugal.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3594536.3595152

ICAIL 2023, June 19-23, 2023, Braga, Portugal

ACM ISBN 979-8-4007-0197-9/23/06…$15.00
https://doi.org/10.1145/3594536.3595152

1 INTRODUCTION

As a process to express human language in a logical manner [46, 53], argumentation is an important topic of study for Natural Language Processing (NLP). One specific NLP topic is Argument Mining (AM), which is a series of tasks aiming to automatically identify and analyse arguments as well as their reasoning process from human language corpora [26, 38, 51]. Considering the multiple application scenarios, argument mining has gradually become popular in interdisciplinary research. Previous AM studies have explored materials like student essays [1, 48], peer reviews [17], and comments on online forums [37]. Regarding the key aspect of argumentation in social and political science [54], AM has been particularly prominent at the intersection of Artificial Intelligence (AI) and Law [7, 14, 42, 56]. Specifically, different types of legal documents have been explored as AM materials including court decisions [14], clinical trials [30], and judicial decisions [56, 57]. Due to the complexity of argumentation in law, some works focus on mining argumentation from the logical or rhetorical background [58, 63], while others emphasise the argumentation from the perspective of legal professionals [14]. Further usages of AM in the legal field have also been tested such as tutoring systems [61] and legal text summarisation [7, 55].

When implementing an AM system for legal documents, most present works (e.g., [13, 31, 42]) still follow a *pipeline architecture* [50]. As illustrated in Figure 1, the pipeline architecture is connected by a sequence of sub-tasks: shrink the searching scope in the raw document at the beginning; identify the text segments as argument components (i.e., different functional units in argumentation); classify the argument components (e.g. premise or conclusion); and predict the argument relations between labelled components. Despite the advantages of this approach, [5, 39] suggest that the *error propagation* inherent in the AM pipeline architecture is an issue, which refers to the situation whereby a classification error in a prior task will have a cascading effect on later tasks. This motivates us to seek alternative perspectives to the task of modelling the argument mining process.

The ideal output of a complete AM system is an *argument graph* like the one displayed in Figure 2, which represents the extraction of argumentative structures [25] with the nodes symbolising the propositions and the edges between them illustrating different relations (e.g., support, attack) [18]. The reasoning process within the argument graph should match the requirements of an *argumentation model* [26, 27]. Developed from computational argumentation, the argumentation model represents arguments from the raw texts into computational structured data. Many argumentation models can be represented as graphs. This matches the general

background of graph computational tasks. Meanwhile, graph models have achieved impressive results in many research fields such as natural science (e.g., chemistry, physics [4, 19]), social science (social networks [8]), and knowledge graphs [47]. This inspires us to view AM as a problem of computational graphs. Considering that the graph is a powerful data structure that represents a set of objects (nodes) and their relations (edges), the AM problem can be abstracted into one end-to-end task, and be calculated by a derivative model which connects the features of different sentences and predicts the sentences as argument components [43].

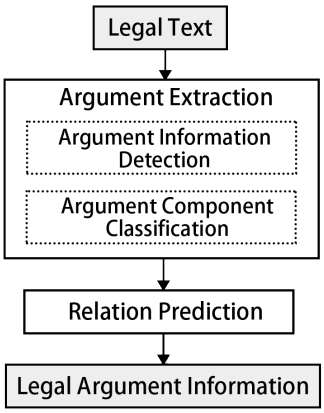

Figure 1: The Argument Mining Pipeline on Legal Text. The two light grey parts are the input and output of the pipeline. The two dot-lined sub-tasks within the argument extraction are connected sequentially.

In this work, we present a new architecture for mining argumentation in legal documents from end to end. Based on graph representation learning (i.e., graph networks), our algorithm consists of collective classification algorithms, text representation models (i.e., transformer-based models), and graph structure augmentation. In contrast to previous works, we redefine the modelling of legal AM into a graph-based node classification task. Therefore, our architecture encodes each entire legal document as a computational graph and predicts all argument components simultaneously. We test our approach with two practical legal document datasets. The primary contribution of this paper is the proposal of a novel implementation architecture for the AM task, which replaces the pipeline architecture with an end-to-end architecture. This is a hybrid method that combines graph augmentation, collective classification, and NLP-based models for legal document AM.

Section 2 discusses the context of this work by exploring related work in the areas of Argument Mining, Graph Structures, Graph Neural Networks and Collective Classification. Section 3 then outlines our proposed approach to AM based on graph representation learning. Following this, Section 4 presents the datasets used for our experiments, which are summarises in Section 5. We present our conclusions and suggestions for future work in Section 6.

In the Government's submission, the fact that they had maintained that this was the foundation of the suspicion should be given considerable weight by the Court. ... ... They also pointed to a number of other facts capable of supporting, albeit indirectly, the reasonableness of the suspicion, including notably the findings made by the domestic courts in the proceedings ... ... They submitted that all these matters taken together provided sufficient facts and information to satisfy an objective observer that there was a reasonable suspicion in the circumstances of the case. ... ... Any other conclusion by the Court would, they feared, prohibit arresting authorities from effecting an arrest of a person suspected of being a terrorist based primarily on reliable but secret information and would inhibit the arresting authorities ... ... The first applicant, on the other hand, considered that the Government had failed to discharge the onus of disclosing sufficient facts to enable the Convention institutions to conclude that the suspicion grounding her arrest was reasonable.

Premise 1

Conclusion 2

Premise 3

Conclusion 4

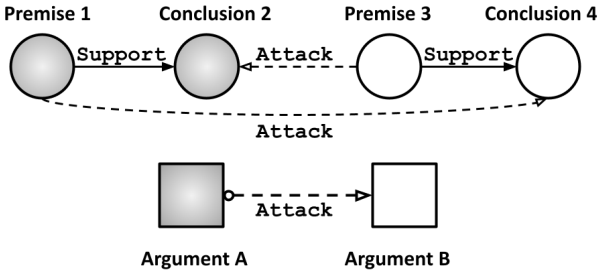

Figure 2: Example of Argument Graph Extracted from Plain Text. The input text is part of a legal document sampled from a practical corpus. Different parts segmented from the text are identified as argument components: premises are covered with light grey; conclusions are underlined. The argument graph contains both element and relation information in each argument.

2 RELATED WORK

2.1 Argument Mining

Due to the fact that the "argument" has more complex and varied structures in the professional law field compared to daily communication, in this work, we follow a high-level structure concluded by the Walton argumentation model [59], one of the typical guidance schemes when annotating AM corpora. Using this structural argumentation model, one can construct an argument with three constituents: *premises* containing evidence or reasons supporting the argument, *conclusions* that serve as the stance and centre of the argument, and the *inferences* that follow from premises and conclusions. Walton's model is highly generalisable, making it suitable for various contexts, such as case law.

The pipeline architecture in Figure 1 treats the AM task as having two stages: the *argument extraction* stage decomposes a long, complete argument into segments, each of which has a rhetoric or logical role; the *relation prediction* task then focuses on mining the relations between the segments identified from the argument

extraction in order to retrieve full argumentative structures [31]. The first stage is designed as a classification task in [25, 31]. In the practical development of AM systems [15, 31, 42, 65], the argument extraction stage usually comprises a series of sub-tasks: segmenting original documents into text spans; identifying those text segments that contain argumentation information (i.e., *argument information detection* in Figure 1); and dividing the argumentative segments into groups based on the argumentation model (i.e., *argument component classification* in Figure 1). Current designs for legal text AM commonly follow this pipeline architecture, which is attractive from the point of view of project management and separation of concerns, but also has obvious drawbacks: the sub-tasks are solved independently without utilising interrelationships between variables [6]; the connection between each part leads to the error propagation problem in the evaluation process [39, 65].

The inter-dependency between sub-tasks [24] within the pipeline has led to the study of *multi-objective learning*, where all sub-tasks are learned and performed simultaneously using end-to-end approaches. [16] created a joint inference method for AM on debating texts. [10] viewed AM as a multi-objective learning task and applied residual networks on user comments. [32] designed a parallel constrained pointer architecture for AM tasks on online civic discussion and persuasive essays.

An alternative point of view is framing AM as a dependency parsing task. When approached in this way, the AM tasks usually involve token-level sequence tagging and elaborate post-processing for argumentative structure extraction. [6] first considered AM as a token-based multi-task problem of dependency parsing and sequence tagging. Based on this, [65] developed biaffine neural models for AM on persuasive essays. [33] experimented with a joint learning framework on sentence-level AM with multiple tasks of component classification, relation detection and relation classification. In [45], they developed an end-to-end argument parser, leveraging the intrinsic relationship between discourse units (argument components) and operating at two levels of granularity: tokens and discourse units. Due to the specific characteristics of legal documents (i.e., strict formatting, long, and complex), when applying AM, current studies still heavily rely on the pipeline. To ultimately implement the end-to-end legal AM solution, we explore AM in terms of the graph, which is introduced in Section 3.

2.2 Graph Structures

In leveraging graph data structures to represent legal text documents, this work follows the graph definition and notation from [29, 62]. For convenience, all the notation used in this paper are outlined in Table 1. A graph is denoted as $G = (V, E)$, where V represents a set of nodes (also described as "vertices") and E represents a set of edges (i.e. connections between nodes). The neighbourhood of a node v is defined as $N(v) = \{u \in V | (v, u) \in E\}$. To represent existing edges, each graph has an adjacency matrix $A \in \mathbb{R}^{n \times n}$ where $n = |V|$ is the number of nodes. A graph may have node attributes and edge attributes. $X \in \mathbb{R}^{n \times d}$ is the node feature matrix with $x_v \in \mathbb{R}^d$ representing the node feature, and $X^e \in \mathbb{R}^{m \times c}$ is the edge feature matrix with $x^e_{(v,u)} \in \mathbb{R}^c$ representing the feature vector of an edge (v, u) pointing from node v to node u. A *directed graph* is a graph with all edges directed from one node to another. An *undirected graph* is a special case where all edges are bidirectional [62].

Table 1: Notations used in this paper.

Notation	Descriptions
G	A graph.
V	The set of nodes in a graph.
v	A node $v \in V$.
E	The set of edges in a graph.
e_{ij}	An edge $e_{ij} \in E$.
$N(v)$	The neighbourhood nodes of a node v.
A	The graph adjacency matrix
n	The number of nodes, $n = \|V\|$.
m	The number of edges, $m = \|E\|$.
d	The dimension of a node feature vector.
b	The dimension of a hidden node feature vector.
c	The dimension of an edge feature vector.
$X \in \mathbb{R}^{n \times d}$	The node feature matrix of a graph.
$x_v \in \mathbb{R}^d$	The feature vector of the node v.
$X^e \in \mathbb{R}^{m \times c}$	The edge feature matrix of a graph.
$x^e_{(v,u)} \in \mathbb{R}^c$	The edge feature vector of an edge (v, u).
$H \in \mathbb{R}^{n \times b}$	The hidden node feature matrix of a graph.
$h_v \in \mathbb{R}^b$	The hidden feature vector of the node v.
Y	The set of node labels.
$y_v \in Y$	The label of the node v.
z_v	The neighbourhood labels' summary of node v.
k	The layer index.
t	The time step/iteration index.
$\square(\cdot)$	A differentiable, permutation invariant function.
$\gamma(\cdot)$, $\phi(\cdot)$	Differentiable functions.
$S(\cdot)$	The label summary function.
$f(\cdot)$	The classifier model.

2.3 Graph Neural Networks (GNNs)

Inspired by the success of neural network models like Convolutional Neural Networks (CNNs) in deep learning methods, Graph Neural Networks (GNNs) are a type of neural model that can be applied directly to graph data structures to make inferences on data provided in a graph while simplifying the performance of prediction tasks [29]. GNNs capture the dependencies within graphs via *message passing* between the graph nodes [69]. Message passing is the generalised convolution operator in GNNs, which is a neighbourhood aggregation scheme described as in this equation:

$$x_v^{(k)} = \gamma^k(x_v^{(k-1)}, \square_{u \in N(v)} \phi^{(k)}(x_v^{(k-1)}, x_u^{(k-1)}, x^e_{(v,u)}))$$

where $x_v^{(k-1)}$ denotes node features of node v in layer $k-1$, $N(v)$ is the neighbourhood nodes of node v. $\square$ denotes a differentiable, permutation invariant function (e.g., sum, mean, or max), γ and ϕ denote differentiable functions (e.g., Multi-Layer Perceptrons).

A basic implementation of message passing is the Graph Convolutional Network (GCN) [20], which is one of the popular basic modules when designing GNNs. During its neighbourhood aggregation, the GCN operator first transforms node features using a

weight matrix, then normalised them by their degree, before applying the bias vector to the aggregated output. Compared to GCN, the Residual Gated Graph ConvNet (ResGCN) from [2] extends the message passing by adapting both the residual network and the gated recurrent units (GRU), which achieves positive results [3].

Due to its strong performance, GNNs have been used to solve various NLP tasks including text classification: [64] designed a two-layer GCN on a heterogeneous text graph with word nodes and document nodes; [36] applied GNN for multi-class label text classification; [52] used a GCN architecture for short text classification. As for tasks related to AM, before GNNs were adopted, some works applied graph-based methods for argument structure analysis and argument evaluation. The argumentation structure in scientific publications was studied through annotations in [21]. Different annotation graphs were compared using a graph-based agreement measure. Similarly, to automatically review issues within a debate, [22] proposed a graph-based analytic method that uses operational and measurable properties representing networks of arguments. Yet, few studies to date have attempted to test the potential of GNNs on tasks related to AM. In [43], they explored the classification performance of GNNs with Tree Kernels on a group of text sampled from AM corpora. Inspired by AM, [44] applied a graph-based network for automatic debate evaluation. The idea behind our work is mostly close to [40], which tried to extract a graph from a document using textual fragments as vertices. [40] applied an unsupervised graph-based ranking model, which attempted to discover whether there is an overlap between approaches used in summarisation and AM tasks. However, none of these included legal documents as their AM material.

2.4 Collective Classification

As mentioned in Section 2.1, a major focus in AM research is to solve the problem as a series of individual classification tasks. Due to the argumentative structure in the textual material, the prediction variants in these classification tasks, are interrelated in many aspects [6]. For example, [11] suggests that the distance between two argument components is a relevant feature for argument mining. This property of AM is different from the assumptions of independence and identical distribution that are associated with general machine learning, but matches the *collective classification* case where it can be beneficial to improve the prediction process if the correct class labels are known for the items being related. Unlike classification methods that treat each item in isolation and predict the class label individually, collective classification methods aim to solve the classification problem in a joint or collective manner. Given a set of nodes V, and the neighbourhood function N, in collective classification V is further divided into two sets: V_{known}, the nodes with labels (observed variables) and, $V_{unknown}$, the nodes whose values remain unknown. The task is to label the nodes $v \in V_{unknown}$ with one of the predefined labels in Y.

According to [34], collective classification approaches can be considered to fall into one of two groups: a) approaches focusing on local representations and propagation methods such as Iterative Classification Algorithm, and Relation Classification; b) approaches representing the problem with a high-level global graphical representation and then applying learning and inference techniques (i.e., Graph-based Models). In contrast to traditional classification, *relational classification* classifies the instance using not only the instance's own attributes but also the attribute of that instance's neighbour [49]. Another method, *iterative classification algorithm* (ICA) [35] uses the class labels assigned to the neighbour instead of the neighbour's attributes for classifying the instance. In addition, *collective classification with graph-based models* [12, 23] represents the entire problem using a global graph-based model and is generally able to apply both neighbourhood labels and the observed attributes of neighbours.

Inspired by the ICA in collective classification, we considered how collective classification can be applied in legal AM and designed our novel graph-based collective classification method according to it. This is discussed in the following section.

3 GRAPH-BASED ARGUMENT MINING

Here we introduce our graph-based architecture for mining arguments from the complex context of legal documents. This section includes three parts: 1) the graph-based solution for text AM which represents the problem with a high-level global model; 2) the graph augmentation used in our AM approach; 3) the collective classification algorithm we have designed for graph-based AM.

3.1 Modelling AM as a Graph-based Task

In contrast to the text sequence in the previous AM pipeline, we first express the text material as a graph structure, in order to apply the GNN model for classification. For each input text, we construct a sentence-level graph G. Each node in the graph $v \in V$ is a text segment with feature x_v encoded by the transformer model (i.e., the node feature is an embedding of the text segment it represents) as well as a node label $y_v \in Y$, where Y denotes the argument component annotations (i.e., "premise", "conclusion", and "neither"). We then add the edges E, based on the order of the text segments within the document (i.e., from the first segment to the second segment, etc.), and connect the final segment's node back to the first segment's node. According to the graph structure, the first stage in AM (see Section 3) is now able to be computed as a node-level classification task.

3.2 Graph Structure Augmentation

The nature of the graph representation above is such that each document is operationally represented by a cycle graph. In traversing the graph, each text segment can be reached only from the edges connecting it to the segments that immediately precede or follow it. In legal texts, however, related argumentative clauses can frequently be spread throughout quite a long document, and as such the cycle graph structure limits the degree to which any graph traversal algorithm can consider related clauses that are not adjacent. Additionally, this approach results in a sparse adjacency matrix. To address these issues, we augment the graphs by adding *virtual nodes*. This is an augmentation method for attributed graphs from [41] which can increase the number of edges and allow relationships between non-adjacent text segments to be explored. For each graph, the virtual node is bidirectionally connected to all existing nodes as illustrated in Figure 3, and represents the latent aspects of the graph. We initialise the virtual node's feature with

the average of existing text nodes' attributes during the message passing in graph representation learning. The virtual node's label is fixed and does not participate in the evaluation.

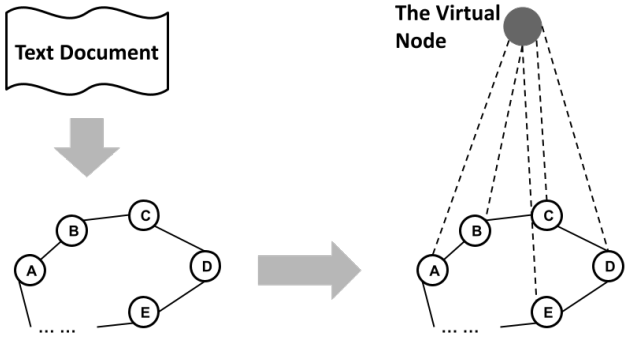

Figure 3: The Virtual Node Graph Augmentation Method. The input text document is first represented as a computational graph, in which the nodes are labelled based on their sequence within the document (A, B, C, etc.). The virtual node added in the graph structure is in dark grey.

3.3 Collective Classification with AM

Regarding the characteristics of argumentation, it is important to take context into account when identifying argumentative relations. The argument components in argumentative texts are related, which we assume matches the collective classification hypothesis (see Section 2.4). During our research of collective classification methods, the classic ICA inspired us to use the label information from the node's neighbourhood and to optimise the prediction result through iterative updates. Then we moved further with the collective graph-based models, as we modelled the AM problem in a new graph-based structure. Combining the ideas from both ICA and graph representation learning, we designed the graph-based collective classification algorithm (GCA).

Algorithm 1 depicts our algorithm as pseudo-code, which includes both the training and testing processes. A set of documents are represented as a set of graphs $\mathcal{G} = \{G : G = (V, E)\}$ and its neighbourhood function N, which is as defined in Section 2.2. The text graphs $\mathcal{G}$ are further divided into two sets: $\mathcal{G}_{train} = \{G : G = (V_{train}, E_{train})\}$ and $\mathcal{G}_{test} = \{G : G = (V_{test}, E_{test})\}$, where $\mathcal{G}_{train} \subset \mathcal{G}$, $\mathcal{G}_{test} \subset \mathcal{G}$, and $\mathcal{G}_{train} \cap \mathcal{G}_{test} = \emptyset$. V_{train} represents nodes with correct labels, and V_{test} are nodes whose labels need to be determined. Our task, as mentioned in Section 3.1, is to label all nodes $v \in V_{test}$ with one of the labels $y_v \in Y$.

To begin with, two classifiers f_1 and f_2 are initialised, which are graph neural networks. The classifier f_1 first focuses on predicting the labels of nodes in the training set $v \in V_{train}$, using only the node feature x_v. The classifier f_1 is updated for t_1 time steps. Then, the classifier f_2 predicts the label y_v of the node $v \in V_{train}$ based on both the node's representation vector h_v generated by f_1 and the neighbourhood labels' summary z_v computed through aggregation function S. Using all the neighbour nodes $u \in N_v$, the neighbourhood labels' summary z_v is aggregated based on the label

Algorithm 1: Iterative Collective Classification on Graphs

```
initialise neural network classifiers f1, f2;
repeat
    repeat
        for node v ∈ V_train do
            compute node representation vector and label
            h_v, ŷ_v ← f1(x_v);
        end
        update classifier f1;
    until t1 iterations;
    repeat
        for node v ∈ V_train do
            compute z_v using neighbourhood aggregation
            z_v ← S_{u∈N(v)} ŷ_u;
            compute label using vector h_v and summary z_v
            y_v ← f2(h_v, z_v);
        end
        update classifier f2;
    until t2 iterations;
until a threshold number of training iterations have elapsed;
fix the neural network classifier f1 and f2;
for node v ∈ V_test do
    compute node representation vector and label
    h_v, ŷ_v ← f1(x_v);
    compute summary z_v using the aggregation
    z_v ← S_{u∈N(v)} ŷ_u;
    compute label using vector h_v and summary z_v
    y_v ← f2(h_v, z_v);
    repeat
        update summary z_v using the aggregation
        z_v ← S_{u∈N(v)} y_u;
        update final label using vector h_v and summary z_v
        y_v ← f2(h_v, z_v);
    until t3 iterations;
end
```

$\hat{y_u}$ predicted by the first classifier f_1. The classifier f_2 is updated for t_2 time steps.

After a threshold number of training iterations, we fix both classifiers f_1 and f_2 before applying them to the test set $\mathcal{G}_{test}$. Using only the node feature x_v, each node $v \in V_{test}$ is assigned a pseudo-label $\hat{y}_v$ predicted by classifier f_1. Next, the classifier f_2 takes both the hidden representation h_v given by f_1 as well as the pseudo-label summary z_v from the neighbourhood N_v and returns a label y_v for each node v. Similar to ICA, our algorithm includes the iterative updates of the neighbourhood label summary z_v, and the node label y_v for each node v until this process reaches a threshold number t_3 of times.

Note that in the algorithm, y_v denotes the label of node v, which refers to a likelihood value vector of the corresponding label in practice. The algorithm is presented generically, and is intended to be generalisable for different methods of embedding nodes, different

Table 2: Datasets for Graph-based Argument Mining. ECHR = the European Court of Human Rights, CJEU = the Court of Justice of the Europan Union, doc = document number, word = word number, graph = graph number, node = node number, pre = premise component number, con = conclusion component number.

Dataset	doc	word	graph	node	pre	con
ECHR	42	290K	40	10,354	1,935	734
CJEU	40	346K	40	9,320	2,375	160

iteration thresholds, different aggregation functions, etc. Specific examples are presented in our experiment description in Section 5.

4 LEGAL DOCUMENTS AND ARGUMENTATION MINING DATASETS

During the development of AM research, several datasets have been curated, focusing on different application domains for AM [67]. For the crucial role played by arguments in the legal field, our study focuses on measuring how our graph-based AM structure works on legal documents. Two English text datasets are used for our experiment, both of which are generated from real-life legal professional corpora.

The first dataset, **ECHR**[1], developed from a well-known resource used in the research area of AI and Law, the online database HUDOC[2] managed by the European Court of Human Rights (ECHR) has been used in AM studies for years [31]. The original file from HUDOC follows a standard format of sections: each document starts with the introduction of the court and the parties involved in the case; then it provides the procedure and facts of the case; next, it includes the discussions and arguments presented by both parties, and finally the judgment of the court is presented. The ECHR corpus [42] contains 42 case-law documents: 20 decisions (3.5K words on average), and 22 judgments (10K words on average). The ECHR dataset follows a sentence-level annotation scheme where each sentence was labelled as *premise*, *conclusion*, or *non-argumentative* based on Walton's argumentation model [59] (as described in Section 2.1).

The second dataset, *Demosthenes*[3], consists of 40 documents (346K words) from the EUR-Lex database[4]. Since the original files are decisions on Fiscal State Aid made by the Court of Justice of the European Union (CJEU), in this work we use **CJEU** to refer to this dataset. This dataset has been recently annotated for the purpose of AM [13] because the CJEU decisions include sufficient and diverse legal arguments. In a similar way to the ECHR cases, the CJEU decision files are structured in a consistent format. Each source file starts with the information of both parties and the Court; it includes case background of facts and procedural case history before the assessment of the General Court. Next, it reports the grounds of appeal containing the arguments of the parties and the findings of the court. Finally, it presents the ruling decision and orders to the parties. When developing the CJEU dataset, the annotators focused on the sections of *Findings of the Court*, which includes all argumentative steps in the Court reasoning process leading to the final ruling. This section consists of a set of interacting inferences, each of which links a set of *premises* to a *conclusion* through *support* or *attack* relations. Based on this characteristic of inference in the documents, the annotation in the CJEU corpus include further information on premise types and argumentation schemes.

Both datasets are annotated by legal experts and use sentence-level segmentation for argument components. The argument component type (i.e., premise, and conclusion) in both annotation schemes generally follow the Walton model as mentioned in Section 2.1. Detailed information on the properties of the experiment datasets is concluded in Table 2.

5 EXPERIMENTS

In this work, we tested the complete argument extraction stage (see Section 2.1) in AM, where the related text segments are identified from the documents and classified into argumentative components. The input text is first encoded into computational graphs as described in Section 3.1, then applied with the graph-based collective classification algorithm (Section 3.3). As part of the experiment, we test model combinations of different GNNs on both legal document datasets.

5.1 Experimental Setting

When encoding the English text into the graph structure, we use text embeddings as the feature x_v for each node v. While several options are available to choose from [60], to align with previous AM studies [42, 66, 68], the text embeddings chosen were those generated by the RoBERTa transformer [28]. Through the final pooling layer, each text segment was expressed as a vector that was further calculated as the node feature. The edges E in the computational graph were generated following the process mentioned in Section 3.1. One virtual node was added per graph to augment the graph adjacency matrix, which linked to all the existing nodes with undirected edges (see Section 3.2). The virtual node's feature is the average of all existing nodes' embeddings. Its label is fixed as "not argumentative" during both the training and testing process but is not counted during evaluation.

5.2 Model Implementation

We built two GNN classification models individually using GCN layers and ResGCN layers (both are introduced in Section 2.3). Based on its message passing operator, we named them *GCN* and *ResGCN*. When implementing the classifiers f_1, and f_2 in Algorithm 1 from Section 3.3, we follow the general GNN structure. Our GNN-based classifier therefore contains two graph neural layers, which compute the hidden node representations, and a final linear layer, which pools the node representations into the label prediction vector. The GNN models are implemented based on PyTorch Geometric (PyG) [9]. The f_1 model consists of two 400-neuron layers, and the f_2 model consists of two 64-neuron layers. The f_1 model calculates the node hidden representative h_v (dim=400) and the node label $\hat{y}_v$ (dim=3) based on the embedding stored as the node feature x_v (dim=768). For our experiment, we use *sum* as the aggregation

[1] https://www.di.uevora.pt/~pq/echr/
[2] https://hudoc.echr.coe.int/
[3] https://github.com/adele-project/demosthenes
[4] https://data.europa.eu/data/datasets/eur-lex-statistics?locale=en

function S, and constrain hop=1 in the neighbourhood function N. The virtual node is excluded from the neighbouring nodes' label summary z_v. The f_2 model then takes the concatenation of both vectors, node representation h_v, and the summary vector z_v, as input (dim=403) to make the final node label prediction y_v.

In each training epoch, heuristically, the iteration step $t1$ is 10, and the iteration step $t2$ is 20. During testing, f_1 predicts the label once and the iteration threshold $t3$ for f_2 remains 20. To compare with the iterative collective classifiers (i.e., f_1, and f_2), we set another two *GCN* and *ResGCN* models, with the same structure of two graph neural layers (196 neurons) and a final liner layer. We trained GCN-based models for 1000 epochs and ResGCN-based models for 500 epochs. For each classifier, we used Adam optimiser (lr=1e-5), and with graph batch size=4 on both ECHR and CJEU datasets.

5.3 Evaluation

For both legal document datasets, CJEU and ECHR, we follow the evaluation process in the original works [13, 42] and ran a 5-fold cross-validation (training set=32 documents, test set=8 documents). In each fold, 6 documents were randomly selected from the training set for validation. For all experiments, the model with the best macro F1-score on the validation set was selected and applied to the test set.

Although previous studies of AM on legal text have tended to follow the pipeline design [42], the prediction results of the argument extraction stage presented in their works are not end-to-end. In particular, before the argument component classification sub-task, it is assumed that all of the argument text segments have been successfully identified from the previous argument information detection sub-task, which avoids error propagation and allows the individual pipeline components to be evaluated separately. However, in the absence of an end-to-end evaluation metric, it is not possible to perform a direct comparison with our non-pipeline approach. As such, we reproduced the pipeline architecture following the guidance in [42]. The two RoBERTa models were individually fine-tuned (epoch=10) on each binary classification sub-task to be the same as [15, 42]. After the fine-tuning, both models' parameters were fixed when being connected as the pipeline: the first transformer divided the text segments into argument/non-argument groups; the second transformer then labelled the argumentative segments that were identified by the first model into argument component groups (i.e., premise/conclusion). We use this pipeline as the baseline for our experiments on the ECHR dataset. For the CJEU dataset, [13] tested different combinations of embeddings and ML classifiers, where they achieved the best result by using TF-IDF and Linear Support Vector Classification (SVM). We use their state-of-the-art result as the baseline for the experiments on CJEU. For each dataset, we report the results obtained by the GNN models, which contain the F1-score for each argument component label and the average classification result.

5.4 Results and Discussion

As displayed in Table 3, when doing argument extraction on the ECHR dataset, the graph-based collective classification algorithm improved the performance of *ResGCN* models. The average F1-score of *ResGCN* in the general classification process is surpassed both on general graphs (0.540 vs. 0.594) and augmented graphs (0.554 vs. 0.618). The *GCN* classifier also achieved better prediction results on augmented graphs with the iteration algorithm, whose average F1-score raised from 0.516 to 0.562. It has less effect on *GCN* models when applied to graphs without augmentation, which we suggest is due to the sparse graph structure not being suitable for message aggregation through neighbouring nodes. Considering the graph structure, the *GCN* model's performance was improved substantially with the virtual node graph augmentation, whose average F1-score increased from 0.478 to 0.516 with the general classification process, and from 0.463 to 0.562 when applied with the iterative collective algorithm. Similarly, when using the *ResGCN* model as the classifier, the enhancement given by virtual nodes raises its average F1-score both in the general process (0.540 vs. 0.554) and in the collective process (0.594 vs. 0.618). The virtual node augmentation enhanced the *ResGCN* model's prediction result from 0.594 to 0.618 (average F1-score) during iterative classification, which is quite close compared to the best average F1-score remaining by the baseline (0.635). Taking each argument component group into account, the graph-based collective algorithm improves both groups of GNNs' prediction result on premise overall: *ResGCN* model achieved better F1-scores than the RoBERTa baseline on the general graph structure (0.571 vs. 0.551) and the augmented graph structure (0.588 vs. 0.551); The precision F1-score presented by *GCN* model increased 10% on average. The *ResGCN* models also reached the best result on the non-argumentative texts (0.891), when applied to virtual node graphs.

Table 3: Detailed F1-score results on ECHR. CA = Collective Classification Algorithm (GCA = Graph-based Collective Classification Algorithm), GS = Graph Structure (N = Normal Graph, V = Virtual Node Graph), avg = macro F1-score, pre = premise, con = conclusion, not = not argumentative

Model	CA	GS	avg	pre	con	not
RoBERTa Pipe	-	-	**0.635**	0.551	**0.467**	0.887
GCN	-	N	0.478	0.459	0.102	0.873
ResGCN	-	N	0.540	0.442	0.309	0.868
GCN	-	V	0.516	0.410	0.274	0.864
ResGCN	-	V	0.554	0.458	0.357	0.847
GCN	GCA	N	0.463	0.512	0.005	0.871
ResGCN	GCA	N	0.594	0.571	0.326	0.884
GCN	GCA	V	0.562	0.565	0.234	0.886
ResGCN	GCA	V	0.618	**0.588**	0.376	**0.891**

Table 4 shows the results of the experiment on the CJEU dataset. Here, both groups of GNN models display higher results on this task with the collective algorithm. In particular, *GCN* model's average F1-score raised over 8% (from 0.664 to 0.718) after using the collective classification algorithm, when applied on the augmented graph structure; *ResGCN* model's average F1-score increased 3% (from 0.655 to 0.683) when applied on basic graph structure. Echoing results on the ECHR dataset, the graph augmentation effect given by virtual nodes is clear on GNN models for the CJEU dataset also. In particular, when using the *GCN* classifier, its average F1-score raised from 0.531 to 0.664 during the general classification process,

Table 4: Detailed F1-score results on CJEU. CA = Collective Classification Algorithm (GCA = Graph-based Collective Classification Algorithm), GS = Graph Structure (N = Normal Graph, V = Virtual Node Graph), avg = macro F1-score, pre = premise, con = conclusion, not = not argumentative

Model	CA	GS	avg	pre	con	not
tf-idf+SVM [13]	-	-	0.700	0.650	0.580	0.880
GCN	-	N	0.531	0.649	0.063	0.881
ResGCN	-	N	0.655	0.546	0.559	0.858
GCN	-	V	0.664	0.555	0.575	0.862
ResGCN	-	V	0.722	0.661	0.621	0.882
GCN	GCA	N	0.542	0.693	0.051	0.883
ResGCN	GCA	N	0.683	**0.747**	0.386	**0.914**
GCN	GCA	V	0.718	0.690	0.593	0.870
ResGCN	GCA	V	**0.758**	0.709	**0.677**	0.888

and from 0.542 to 0.718 during the iterative collective process. The average score of *ResGCN* was also boosted from 0.655 to 0.722 during its general classification process. Compared to the baseline given by TF-IDF and SVM, the *ResGCN* model reached its highest average F1-score (0.758) and maintained a strong performance even without the graph augmentation, which reached the highest average F1-score on two argument component groups. Adding virtual nodes in sparse graphs helps the GNN models to predict conclusions in the CJEU dataset. It largely improves the *GCN* model's performance, and supports the *ResGCN* model reaching higher results (0.621 and 0.677, with and without collective algorithm) than the baseline (0.580). This graph augmentation also slightly improves the prediction result of the *ResGCN* model on not argumentative text segments. Similar to the previous experiments on ECHR, the collective classification process retains its improvement on GNNs when identifying premises. By using the iterative algorithm, both *GCN* and *ResGCN* models surpass the baseline (0.650) around 4% and 10% in each case at the task of identifying premises.

In general, it is clear that the virtual node graph augmentation increases the performance of GNN models in most cases. Especially, the *GCN* model achieved sufficient improvement when predicting "conclusion" texts on graphs with virtual nodes in both legal corpora. The collective classification algorithm enhances *ResGCN* and *GCN* have advanced results when predicting the "premise" and "not argumentative" nodes among all graph sets. When combined with virtual node graphs, the collective classification generally amplifies the prediction ability of GNN models. Also, the *ResGCN* model exceeds the *GCN* model on larger graphs generated by long text materials (i.e., legal documents).

6 CONCLUSION AND FUTURE WORK

In this paper, we explore the potential of viewing AM from the angle of a graph-based data structure. Accordingly, we present a new architecture for mining arguments from text files in the legal field, where an entire document is encoded as a computational graph. We proposed a method that combines representation learning, collective classification and graph augmentation. Several conclusions can be drawn from the evaluation results: it has been possible to state that our graph-based solution performed better than the previous pipeline architecture with connected sub-tasks that approach argument component identification sequentially. Given the observation from our experiments, the GNN model with gate operator (i.e., ResGCN) provides great performance on graph-structured data generated from long legal texts. Besides, when encoding document graphs for AM, the graph augmentation method (i.e., adding virtual nodes) adds benefit in most cases. Therefore, when implementing an AM system, graph models could be used as a powerful tool in complex long contexts like legal documents. In our view, this work not only provides methods that yield better performance but also offers a different option for researchers when studying the AM problem.

In spite of this, much study remains to be done, and a number of aspects are still unknown. For future work, our graph-based structure has anticipated merging the relation prediction stage in order to transform the previous pipeline into a multi-objective end-to-end AM system. Since graph augmentation provides effective enhancement, it may be worth studying the relationship between graph augmentation and the document graph. The current approach adds a single virtual node that connects to all document nodes. Alternative combinations of multiple virtual nodes that are more strategically connected to subsets of the document nodes may amplify the enhancement effect. Another major part of our future work will focus on creating more complex graph structures, including incorporating semantic information in our graph representation. Furthermore, we also plan to study recurrent/recursive architectures on linear graphs. Besides, reducing the time complexity of the current classification algorithm is able to improve the efficiency of the AM training process. Furthermore, we plan to explore the edge attributions in the computational graph as well as the variants of GNNs that have performed well on NLP tasks.

REFERENCES

[1] Tariq Alhindi and Debanjan Ghosh. 2021. "Sharks are not the threat humans are": Argument Component Segmentation in School Student Essays. In *Proceedings of the 16th Workshop on Innovative Use of NLP for Building Educational Applications*. Association for Computational Linguistics, 210–222. https://aclanthology.org/2021.bea-1.22

[2] Xavier Bresson and Thomas Laurent. 2017. Residual Gated Graph ConvNets. *arXiv e-prints* (2017), arXiv–1711.

[3] Yihao Chen, Xin Tang, Xianbiao Qi, Chun-Guang Li, and Rong Xiao. 2022. Learning graph normalization for graph neural networks. *Neurocomputing* 493 (2022), 613–625.

[4] Kien Do, Truyen Tran, and Svetha Venkatesh. 2019. Graph transformation policy network for chemical reaction prediction. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 750–760.

[5] Yang Du, Minglan Li, and Mengxue Li. 2017. Joint extraction of argument components and relations. In *2017 International Conference on Asian Language Processing (IALP)*. IEEE, 1–4.

[6] Steffen Eger, Johannes Daxenberger, and Iryna Gurevych. 2017. Neural End-to-End Learning for Computational Argumentation Mining. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. 11–22.

[7] Mohamed Elaraby and Diane Litman. 2022. ArgLegalSumm: Improving Abstractive Summarization of Legal Documents with Argument Mining. In *Proceedings of the 29th International Conference on Computational Linguistics*. 6187–6194.

[8] Wenqi Fan, Yao Ma, Qing Li, Yuan He, Eric Zhao, Jiliang Tang, and Dawei Yin. 2019. Graph neural networks for social recommendation. In *The world wide web conference*. 417–426.

[9] Matthias Fey and Jan Eric Lenssen. 2019. Fast Graph Representation Learning with PyTorch Geometric. *arXiv e-prints* (2019), arXiv–1903.

[10] Andrea Galassi, Marco Lippi, and Paolo Torroni. 2018. Argumentative link prediction using residual networks and multi-objective learning. In *Proceedings of the 5th Workshop on Argument Mining*. 1–10.

[11] Andrea Galassi, Marco Lippi, and Paolo Torroni. 2021. Multi-Task Attentive Residual Networks for Argument Mining. (2021).

[12] Lise Getoor, Eran Segal, Ben Taskar, and Daphne Koller. 2001. Probabilistic models of text and link structure for hypertext classification. In *IJCAI workshop on text learning: beyond supervision*. Seattle, WA„ 321–374.

[13] Giulia Grundler, Piera Santin, Andrea Galassi, Federico Galli, Francesco Godano, Francesca Lagioia, Elena Palmieri, Federico Ruggeri, Giovanni Sartor, and Paolo Torroni. 2022. Detecting Arguments in CJEU Decisions on Fiscal State Aid. In *Proceedings of the 9th Workshop on Argument Mining*. 143–157.

[14] Ivan Habernal, Daniel Faber, Nicola Recchia, Sebastian Bretthauer, Iryna Gurevych, Christoph Burchard, et al. 2022. Mining legal arguments in court decisions. *arXiv preprint arXiv:2208.06178* (2022).

[15] Shohreh Haddadan, Elena Cabrio, and Serena Villata. 2019. Yes, we can! mining arguments in 50 years of US presidential campaign debates. In *ACL 2019-57th Annual Meeting of the Association for Computational Linguistics*. 4684–4690.

[16] Yufang Hou and Charles Jochim. 2017. Argument relation classification using a joint inference model. In *Proceedings of the 4th Workshop on Argument Mining*. 60–66.

[17] Xinyu Hua, Mitko Nikolov, Nikhil Badugu, and Lu Wang. 2019. Argument Mining for Understanding Peer Reviews. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*. 2131–2137.

[18] B James. 1991. Freeman. Dialectics and the macrostructure of arguments: A theory of argument structure.

[19] Thomas Kipf, Ethan Fetaya, Kuan-Chieh Wang, Max Welling, and Richard Zemel. 2018. Neural relational inference for interacting systems. In *International conference on machine learning*. PMLR, 2688–2697.

[20] Thomas N Kipf and Max Welling. 2019. Semi-Supervised Classification with Graph Convolutional Networks. (2019).

[21] Christian Kirschner, Judith Eckle-Kohler, and Iryna Gurevych. 2015. Linking the thoughts: Analysis of argumentation structures in scientific publications. In *Proceedings of the 2nd Workshop on Argumentation Mining*. 1–11.

[22] Barbara Konat, John Lawrence, Joonsuk Park, Katarzyna Budzynska, and Chris Reed. 2016. A corpus of argument networks: Using graph properties to analyse divisive issues. In *Tenth International Conference on Language Resources and Evaluation*. European Language Resources Association, 3899–3906.

[23] John D Lafferty, Andrew McCallum, and Fernando CN Pereira. 2001. Conditional Random Fields: Probabilistic Models for Segmenting and Labeling Sequence Data. In *Proceedings of the Eighteenth International Conference on Machine Learning*. 282–289.

[24] John Lawrence and Chris Reed. 2015. Combining argument mining techniques. In *Proceedings of the 2nd Workshop on Argumentation Mining*. 127–136.

[25] John Lawrence and Chris Reed. 2020. Argument mining: A survey. *Computational Linguistics* 45, 4 (2020), 765–818.

[26] Marco Lippi and Paolo Torroni. 2015. Argument mining: A machine learning perspective. In *International Workshop on Theory and Applications of Formal Argumentation*. Springer, 163–176.

[27] Marco Lippi and Paolo Torroni. 2016. Argument mining from speech: Detecting claims in political debates. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 30.

[28] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. RoBERTa: A Robustly Optimized BERT Pretraining Approach. *arXiv e-prints* (2019), arXiv–1907.

[29] Masoud Malekzadeh, Parisa Hajibabaee, Maryam Heidari, Samira Zad, Ozlem Uzuner, and James H Jones. 2021. Review of graph neural network in text classification. In *2021 IEEE 12th Annual Ubiquitous Computing, Electronics & Mobile Communication Conference (UEMCON)*. IEEE, 0084–0091.

[30] Tobias Mayer, Elena Cabrio, Marco Lippi, Paolo Torroni, Serena Villata, et al. 2018. Argument Mining on Clinical Trials.. In *COMMA*. 137–148.

[31] Marie-Francine Moens. 2013. Argumentation mining: Where are we now, where do we want to be and how do we get there?. In *Post-Proceedings of the 4th and 5th Workshops of the Forum for Information Retrieval Evaluation*. 1–6.

[32] Gaku Morio and Katsuhide Fujita. 2018. End-to-End Argument Mining for Discussion Threads Based on Parallel Constrained Pointer Architecture. *EMNLP 2018* (2018), 11.

[33] Gaku Morio, Hiroaki Ozaki, Terufumi Morishita, Yuta Koreeda, and Kohsuke Yanai. 2020. Towards better non-tree argument mining: Proposition-level biaffine parsing with task-specific parameterization. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*. 3259–3266.

[34] Galileo Namata, Prithviraj Sen, Mustafa Bilgic, Lise Getoor, M Sahami, and A Srivastava. 2009. Collective classification for text classification. *Text Mining* (2009), 51–69.

[35] Jennifer Neville and David Jensen. 2000. Iterative classification in relational data. In *Proc. AAAI-2000 workshop on learning statistical models from relational data*. 13–20.

[36] Ankit Pal, Muru Selvakumar, and Malaikannan Sankarasubbu. 2020. Multi-label text classification using attention-based graph neural network. *arXiv preprint arXiv:2003.11644* (2020).

[37] Joonsuk Park, Cheryl Blake, and Claire Cardie. 2015. Toward machine-assisted participation in erulemaking: An argumentation model of evaluability. In *Proceedings of the 15th International Conference on Artificial Intelligence and Law*. 206–210.

[38] Andreas Peldszus and Manfred Stede. 2013. From argument diagrams to argumentation mining in texts: A survey. *International Journal of Cognitive Informatics and Natural Intelligence (IJCINI)* 7, 1 (2013), 1–31.

[39] Isaac Persing and Vincent Ng. 2016. End-to-end argumentation mining in student essays. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. 1384–1394.

[40] Georgios Petasis and Vangelis Karkaletsis. 2016. Identifying argument components through textrank. In *Proceedings of the Third Workshop on Argument Mining (ArgMining2016)*. 94–102.

[41] Trang Pham, Truyen Tran, Hoa Dam, and Svetha Venkatesh. 2017. Graph Classification via Deep Learning with Virtual Nodes. *arXiv e-prints* (2017), arXiv–1708.

[42] Prakash Poudyal, Jaromír Šavelka, Aagje Ieven, Marie Francine Moens, Teresa Gonçalves, and Paulo Quaresma. 2020. Echr: legal corpus for argument mining. In *Proceedings of the 7th Workshop on Argument Mining*. 67–75.

[43] Federico Ruggeri, Marco Lippi, and Paolo Torroni. 2021. Tree-constrained graph neural networks for argument mining. *arXiv preprint arXiv:2110.00124* (2021).

[44] Ramon Ruiz-Dolz, Stella Heras, and Ana García-Fornes. 2022. Automatic Debate Evaluation with Argumentation Semantics and Natural Language Argument Graph Networks. *arXiv preprint arXiv:2203.14647* (2022).

[45] Sougata Saha, Souvik Das, and Rohini K Srihari. 2022. EDU-AP: Elementary Discourse Unit based Argument Parser. In *Proceedings of the 23rd Annual Meeting of the Special Interest Group on Discourse and Dialogue*. 183–192.

[46] Oliver Scheuer, Frank Loll, Niels Pinkwart, and Bruce M McLaren. 2010. Computer-supported argumentation: A review of the state of the art. *International Journal of Computer-supported collaborative learning* 5 (2010), 43–102.

[47] Michael Schlichtkrull, Thomas N Kipf, Peter Bloem, Rianne Van Den Berg, Ivan Titov, and Max Welling. 2018. Modeling relational data with graph convolutional networks. In *The Semantic Web: 15th International Conference, ESWC 2018, Heraklion, Crete, Greece, June 3–7, 2018, Proceedings 15*. Springer, 593–607.

[48] Claudia Schulz, Christian M Meyer, and Iryna Gurevych. 2019. Challenges in the automatic analysis of students' diagnostic reasoning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 33. 6974–6981.

[49] S SLATTERY. 1998. Combining Statistical and Relational Methods for Learning in Hypertext Domains. In *Proceedings of the 8th International Conference on Inductive Logic Programming, 1998*. Springer-Verlag.

[50] Noam Slonim, Yonatan Bilu, Carlos Alzate, Roy Bar-Haim, Ben Bogin, Francesca Bonin, Leshem Choshen, Edo Cohen-Karlik, Lena Dankin, Lilach Edelstein, et al. 2021. An autonomous debating system. *Nature* 591, 7850 (2021), 379–384.

[51] Christian Stab, C. Kirschner, Judith Eckle-Kohler, and I. Gurevych. 2014. Argumentation mining in persuasive essays and scientific articles from the discourse structure perspective. *CEUR Workshop Proceedings* 1341 (01 2014).

[52] Kshitij Tayal, Rao Nikhil, Saurabh Agarwal, and Karthik Subbian. 2019. Short text classification using graph convolutional network. In *NIPS workshop on Graph Representation Learning*.

[53] Frans H Van Eemeren and Rob Grootendorst. 2004. *A systematic theory of argumentation: The pragma-dialectical approach*. Cambridge University Press.

[54] Eva Maria Vecchi, Neele Falk, Iman Jundi, and Gabriella Lapesa. 2021. Towards Argument Mining for Social Good: A Survey. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*. Association for Computational Linguistics, Online, 1338–1352. https://doi.org/10.18653/v1/2021.acl-long.107

[55] S Villata et al. 2020. Using argument mining for legal text summarization. In *IOS Press*, Vol. 334. 184.

[56] Vern Walker, Dina Foerster, Julia Monica Ponce, and Matthew Rosen. 2018. Evidence types, credibility factors, and patterns or soft rules for weighing conflicting evidence: Argument mining in the context of legal rules governing evidence assessment. In *Proceedings of the 5th Workshop on Argument Mining*. 68–78.

[57] Vern R Walker, Parisa Bagheri, and Andrew J Lauria. 2015. Argumentation mining from judicial decisions: the attribution problem and the need for legal discourse models. In *Workshop on automated detection, extraction and analysis of semantic information in legal texts (ASAIL-2015)*.

[58] Douglas Walton. 2012. Argument mining by applying argumentation schemes. *Studies in Logic* 4, 1 (2012), 2011.

[59] Douglas Walton, Christopher Reed, and Fabrizio Macagno. 2008. *Argumentation schemes*. Cambridge University Press.

[60] Congcong Wang and David Lillis. 2020. A Comparative Study on Word Embeddings in Deep Learning for Text Classification. In *Proceedings of the 4th International Conference on Natural Language Processing and Information Retrieval (NLPIR 2020)*. Seoul, South Korea. https://doi.org/10.1145/3443279.3443304

[61] Hannes Westermann, Jaromir Savelka, Vern Walker, Kevin Ashley, and Karim Benyekhlef. 2022. *Toward an Intelligent Tutoring System for Argument Mining in Legal Texts.* https://doi.org/10.3233/FAIA220456

[62] Zonghan Wu, Shirui Pan, Fengwen Chen, Guodong Long, Chengqi Zhang, and S Yu Philip. 2020. A comprehensive survey on graph neural networks. *IEEE transactions on neural networks and learning systems* 32, 1 (2020), 4–24.

[63] Huihui Xu and Kevin Ashley. 2022. *Multi-Granularity Argument Mining in Legal Texts.* https://doi.org/10.3233/FAIA220477

[64] Liang Yao, Chengsheng Mao, and Yuan Luo. 2019. Graph convolutional networks for text classification. In *Proceedings of the AAAI conference on artificial intelligence*, Vol. 33. 7370–7377.

[65] Yuxiao Ye and Simone Teufel. 2021. End-to-end argument mining as biaffine dependency parsing. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*. 669–678.

[66] Gechuan Zhang, David Lillis, and Paul Nulty. 2021. Can domain pre-training help interdisciplinary researchers from data annotation poverty? A case study of legal argument mining with BERT-based transformers. In *Proceedings of the Workshop on Natural Language Processing for Digital Humanities*. 121–130.

[67] Gechuan Zhang, Paul Nulty, and David Lillis. 2022. A Decade of Legal Argumentation Mining: Datasets and Approaches. Springer International Publishing, 240–252. https://doi.org/10.1007/978-3-031-08473-7_22

[68] Gechuan Zhang, Paul Nulty, and David Lillis. 2022. Enhancing legal argument mining with domain pre-training and neural networks. *CoRR, abs/2202.13457* (2022).

[69] Jie Zhou, Ganqu Cui, Shengding Hu, Zhengyan Zhang, Cheng Yang, Zhiyuan Liu, Lifeng Wang, Changcheng Li, and Maosong Sun. 2020. Graph neural networks: A review of methods and applications. *AI Open* 1 (2020), 57–81.