Retrieving Argument Graphs Using Vision Transformers

Kilian Bartz[1] and **Mirko Lenz**[1,2] and **Ralph Bergmann**[1,2]

[1]German Research Center for Artificial Intelligence (DFKI),
Behringstr. 21, 54296 Trier, Germany, ebls.dfki.de
[2]Artificial Intelligence and Intelligent Information Systems, Trier University,
Universitätsring 15, 54296 Trier, Germany, www.wi2.uni-trier.de
Correspondence: info@mirko-lenz.de

Abstract

Through manual annotation or automated argument mining processes, arguments can be represented not only as text, but also in structured formats like graphs. When searching for relevant arguments, this additional information about the relationship between their elementary units allows for the formulation of fine-grained structural constraints by using graphs as queries. Then, a retrieval can be performed by computing the similarity between the query and all available arguments. Previous works employed Graph Edit Distance (GED) algorithms such as A* search to compute mappings between nodes and edges for determining the similarity, which is rather expensive. In this paper, we propose an alternative based on Vision Transformers where arguments are rendered as images to obtain dense embeddings. We propose multiple space-filling visualizations and evaluate the retrieval performance of the vision-based approach against an existing A* search-based method. We find that our technique runs orders of magnitude faster than A* search and scales well on larger argument graphs while achieving competitive results.

1 Introduction

Argumentation plays an important role in daily life and is essential for cultural, social, and intellectual progress (Van Eemeren, 2018). Arguments are deeply woven into decision-making processes: People who have the most convincing arguments are more likely to influence others and shape public opinion. Traditional search engines allow users such as journalists to find relevant arguments based on their semantics, but have limited to no support for incorporating structural aspects into the retrieval. To overcome this limitation, structure-aware representations combined with Argument Mining (AM) (Lawrence and Reed, 2019) techniques may be used—for instance, argument graphs with nodes representing Argumentative Discourse Units (ADUs) (Peldszus and Stede, 2013) and edges representing relationships between them (see Section 2). Consider the following example shown in Figure 1: A journalist is looking for a counter-argument against a policy that is being discussed in the media. In addition, they would like to obtain another argument attacking the relation between the policy and its counter-argument. In a traditional search engine, they would have to formulate a text-based query describing these constraints in a rather verbose way. This might work for smaller arguments, but as the complexity increases, it becomes increasingly difficult to express them in natural language. In contrast, with argument graphs, the journalist can create a graph-based query where the constraints are expressed via edges and only the semantics of the arguments need to be described in natural language (by labeling the nodes). Now, the search engine can incorporate both aspects into the retrieval process.

This structured graph format introduces a new challenge: *How to efficiently retrieve arguments based on their structure?* Existing approaches employ *graph matching* (Livi and Rizzi, 2013) to tackle this problem—for instance, by computing the Graph Edit Distance (GED) using the A* algorithm (Bergmann et al., 2019; Lenz et al., 2019). While *effective*, these techniques do not scale well as the computing the GED is an NP-hard problem (Bunke, 1997), requiring the use of heuristics to reduce the search space. One could also use *graph embeddings* to determine similarity scores between graphs by mapping them to some vector space (Marro et al., 2022). Their main advantage is that the resulting vectors can be computed in parallel on powerful Graphics Processing Units (GPUs) and can even be cached for future use—making the retrieval process much faster and scalable. However, these models typically require feature engineering to obtain sensible vector representations and need to be trained on large annotated datasets—

Proceedings of the 12th Argument Mining Workshop, pages 32–45

which are often not available for specific domains. In this paper, we propose an alternative approach to obtain structure-aware embeddings based on Vision Transformers (ViTs) (Dosovitskiy et al., 2021) and visualizations of argument graphs. Building on the idea of Bergmann et al. (2019), we use a two-step retrieval process: First, a set of semantically similar argument graphs is retrieved from the corpus at hand (e.g., using a text embedding model). Then, the remaining arguments are rendered to images, fed to the aforementioned ViT to determine structure-aware embeddings, and finally assess the similarity to the query—leading to a ranking of semantically *and* structurally relevant argument graphs. Special consideration is given to the design of the visualizations, as they need to be optimized for characteristics of ViTs and not human perception. Compared to the previously discussed graph embeddings, the use of visualizations as an intermediate representation also offers increased interpretability. In addition, the "fuzzier" ViT embeddings may even be a better approximation to the way human experts assess structural similarity by focusing on the global structure of the graphs rather than local features.

Hence, the following research question is evaluated in this paper: "Are vision-based graph similarities *more efficient* than and *equally effective* as ones based on GED for the retrieval of argument graphs?" Our vision is to speed up the structural similarity computation in a way that enables real-time argument graph retrieval that is backed by AM to construct the required graph representations. Our main contributions for answering this question are: (i) Three space-filling visualizations for argument graphs optimized for the characteristics of ViT, (ii) a pre-training and fine-tuning pipeline for ViT models to learn structural similarities from these visualizations, (iii) an open-source implementation of the visualization for hierarchical graphs and the training pipeline, and (iv) an experimental evaluation comparing our vision-based to a baseline A* retrieval on a dataset with reference rankings from human experts.

In the remainder of this paper, we first introduce the foundations of argumentation and discuss related work concerning graph-based retrieval in Section 2. Then, we present our visualization techniques and training pipeline in Section 3, followed by an evaluation of the proposed approach in Section 4. Finally, we conclude the paper and discuss future work in Section 5.

2 Foundations and Related Work

In this section, we will briefly introduce the core concepts behind our work and discuss relevant works from the literature, starting with the concept of argumentation. In its simplest form, an argument consists of one *claim* that is supported or attacked by one or more *premises* (Peldszus and Stede, 2013). A claim may also serve as a premise for other claims, allowing for the creation of complex argument structures—in which case the argument often also contains a *major claim* that encodes the overall conclusion. Such larger constructs can be represented as argument graphs, for example via the Argument Interchange Format (AIF) (Chesñevar et al., 2006). This standard specifies two types of nodes: Information Nodes (I-nodes) representing the contents of the argument and Scheme Nodes (S-nodes) representing the applied argumentation schemes. Such argument graphs are acyclic and directed, an example is shown in Figure 1.

Vision Transformers and Image Retrieval The original transformer architecture (Vaswani et al., 2017) was developed for text processing tasks, such as machine translation. To support image data, Dosovitskiy et al. (2021) proposed dividing an image into fixed-size patches, which are then fed into a linear projection layer. After combining the patch embeddings from the projection with position embeddings, they can be fed into a Transformer model as a sequence of vectors where self-attention can be applied. Based on the original ViT architecture, Swin Transformer V1 (Liu et al., 2021) and V2 (Liu et al., 2022) improve on it by increasing its efficiency and suitability as a large-scale vision model. ViTs have been successfully applied for general image retrieval (El-Nouby et al., 2021) by training a ViT with a Siamese architecture and a metric learning objective to generate image embeddings. More broadly, generating a ranking of images w.r.t. to some query is tackled by Content-based Image Retrieval (CBIR) systems (Pedronette and Torres, 2013). Besides optimizations regarding the numeric representation of images, re-ranking based on similarity of ranked lists (Pedronette and Torres, 2013), query-specific semantic signatures (Wang et al., 2013), click data (Jain and Varma, 2011) and other means available to the respective CBIR system have been explored to improve the retrieval quality.

Graph Embeddings for Retrieval The goal of graph embeddings is to encode the graph's structure and content into a fixed-size vector representation suitable for downstream tasks (Xu, 2021). Popular approaches are random walk-based methods (Perozzi et al., 2014; Grover and Leskovec, 2016) and neural network-based methods, using Graph Convolutional Networks (Kipf and Welling, 2016) or Graph Transformers (Tang et al., 2020). These embed the elements of a graph individually and then aggregate them. To represent an entire graph as a vector instead, graph kernels have been used (Cai et al., 2018). Here, the resulting vector contains the counts of the elementary substructures from which the graph is constructed. Different methods include decomposing a graph into so-called graphlets (fixed-sized sub-graphs) or subtree patterns (Cai et al., 2018).

Graph Edit Distance for Retrieval As mentioned in Section 1, incorporating structural aspects into the retrieval of arguments has been tackled by multiple works in the past (Bergmann et al., 2019; Lenz et al., 2019)—their approach will serve as a baseline for our evaluation. The authors employ Case-Based Reasoning (CBR) (Aamodt and Plaza, 1994)—a methodology that uses past experience to solve new problems and often works with highly structured data. A core idea for such representations is the use of global and local similarities: Instead of a sophisticated measure for complex data, one can break it down into simpler (local) similarity metrics for its components and combine them into a global similarity measure (Burkhard and Richter, 2001). The subfield Process-Oriented Case-Based Reasoning (POCBR) (Minor et al., 2014) applies this methodology to graph-based representations of business workflows—here, similarities are defined for the nodes and edges of the graphs and combined into a global score by finding an optimal mapping between two graphs (Bergmann and Gil, 2014). This mapping is defined via a type-preserving, partial, injective function that maps the nodes and edges of the query graph to the case graph. For argument graphs, Bergmann et al. (2019) propose the use of embeddings for the similarity between I-nodes a binary or taxonomy-based measure for S-nodes. Finding the optimal mapping usually requires an exhaustive search, which is infeasible for large graphs. The authors use two optimizations to reduce the search space: (i) An A* search algorithm with admissible heuristics to prune the search space and (ii) a pre-filter based on embeddings to reduce the number of cases that have to be considered in the search phase—also known as Many Are Called / Few Are Chosen (MAC/FAC) (Forbus et al., 1995). Recent works also investigated the use of GPUs for this task (Hoffmann et al., 2022), but there exists no universally applicable solution for GPU-based graph matching that could be applied to the problem at hand.

3 Vision-Based Graph Retrieval

In the following, we describe the vision-based pipeline for structural argument graph retrieval. It uses argument graphs that can be obtained from AM systems—for instance, from plain texts or other prestructured data like debates or discussions. Given some query graph q, the goal is to generate a ranking with the k most relevant/similar argument graphs $(c_1, \ldots, c_k), c_i \in C$ from some corpus/case base C. The structured query may be constructed either by hand from expert users or automatically built using AM techniques—even enabling novices to benefit from structure-aware retrieval. Both the query q and the cases c_i are represented as AIF graphs (see Section 2), meaning that the arguments contain structural and semantic information that should be incorporated into the ranking. We propose a three-step pipeline for this task: (i) Filter the argument graphs in C to remove all cases which are topically (semantically) irrelevant to the query q, (ii) convert the remaining argument graphs to some visual representation, and (iii) use a ViT model to generate embeddings from these visualizations. This allows us to calculate the similarity between arguments using standard methods like cosine similarity and re-rank the arguments based on this.

A critical aspect of this pipeline is the visualization choice, as this image is the only input the model receives. Traditional node-link diagrams are well studied and probably used most frequently for graph-based structure. However, layout algorithms for node-link drawings may produce hardly readable visualizations when data gets too large and complex. Such a graph drawing generally inherits the shape of the underlying structure when using uniform node sizes, possibly leading to sparse graphs that may be overly wide or deep and thus not ideal for ViT models with a square input window. Therefore, we propose three space-filling visualizations that are more suitable for this task, as they can be scaled up or down to fully utilize its

context size. They are specialized for displaying hierarchical data and as such, need some starting point—which in our case is the major claim of the graph. If the graph has no explicit major claim, one can be set arbitrarily (e.g., the topmost node).

3.1 Visualization

We explored the curated tree visualization library treevis.net (Schulz, 2011) to obtain an initial set of candidates. As of April 2025, it contains a collection of 341 techniques grouped by dimensionality, representation, and alignment. After implementing and adapting some of the listed options for our use case, we settled on three variants: (i) Treemap, (ii) Logical, and (iii) Space Reclaiming Icicle Plots (SRIP). All of them visualize the structure of the argument graphs (which are often trees) hierarchically in a space-filling manner, bringing the following advantages: (i) Vision models tend to ignore filigree lines (i.e., edges) of traditional node-link drawings, which might lead to vision models completely ignoring certain relations between ADUs. Because of this, we also avoid using explicit lines to mark borders between areas and instead rely on different colors and hues to separate ADUs. (ii) In node-link drawings, related nodes might be separated by a large space if this suits the layout algorithm better. This makes it harder for the vision model to capture these relations. (iii) All node-link graph visualizations, even if they are intended to visualize very large graphs, use white-space, on which a graph's nodes and edges are then laid out. When an image constructed using one of these visualizations must be scaled down to fit into the square input window of a vision model, the first issue is further amplified. (iv) The layout of our space–filling visualizations is unambiguous and simple in contrast to some node-link visualizations (e.g., force-directed layouts), allowing us to generate deterministic embeddings. An example of an argument graph in all three visualizations can be found in Figure 1.

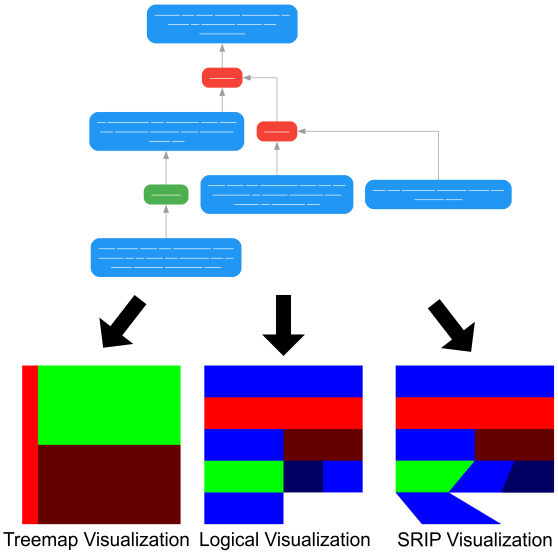

Figure 1: Example of an argument graph (top) in all three visualization (bottom). (Peldszus and Stede, 2015) The treemap only visualizes S-nodes, while the Logical and SRIP visualization also include I-nodes. Blue represents I-nodes, red attacking S-nodes, and green supporting S-nodes.

Treemap Visualization Argument graphs often have a hierarchical, tree-like structure—for which treemaps (Johnson and Shneiderman, 1991) are a commonly used visualization. This visualization works by recursively subdividing the space of a parent node into rectangles for its children and as such allows to completely fill the available space. While in principle it would be possible to add I-nodes together with S-nodes to the visualization, this would lead to a very cluttered image. Instead, we chose to only visualize S-nodes, as we argue the branching degree of I-nodes is secondary to the overall graph structure in the context of argument retrieval. Relying solely on S-nodes allows us to focus on the relations between them to visually represent serial, linked, or convergent premises. As this greatly reduces the number of nodes that need to be visualized, even images of large graphs remain readable. The colors red and green are used to represent attacking and supporting S-nodes respectively. While the choice of red and green as a differentiator may not be ideal for human consumption w.r.t. color deficiencies, it maximizes the contrast in the RGB color space and is therefore well-suited for ViT models.

Traditionally, treemaps work by only displaying a single layer: The entire space for one parent node is equally divided into rectangles of its children. However, this means that nested parent nodes are lost, meaning that the chain/hierarchy of S-nodes from the root of the tree to its leaf nodes is not visible. To overcome this limitation, we propose a modification to the traditional treemap algorithm: We reserve a fixed percentage of the parent's area to visualize the parent itself. This way, the parent node is always visible even if it has many children. Based on our experiments, we found that a 10%

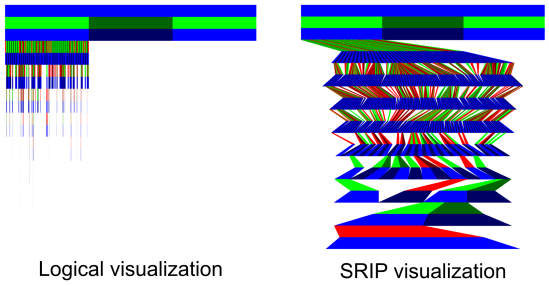

Figure 2: Visualization of a large argument graph. (Agarwal et al., 2022) The nodes are too small to discern an S-node's type. The space-reclaiming visualization remains more readable and wastes less space, especially at the bottom.

area for the parent node is a good compromise between visibility and space utilization.

Logical Visualization Our second visualization is based on a "Formal Logical Representation of Set Inclusions" (Baron, 1969). Here, we visualize the entire argument graph including I-nodes from the bottom up—similar to the way node-link diagrams for argument graphs are commonly constructed. The reason for including I-nodes is that the focus of this visualization is not on showing nested structures, but rather on the argumentation threads themselves in a row-by-row manner. All of the major claim's incoming nodes are processed recursively with the current node being treated as the root node of the respective subgraph. As a result, the visualization is a series of rectangles, each representing a node in the argument graph.

Space-Reclaiming Icicle Plots (SRIP) Our logical representation has the weakness that a child can only ever use the full width of its parent, even if there are no other nodes in the current row. This leads to a subpar space utilization for argument graphs with a single, very long argumentation thread (see Figure 2). SRIP (van de Wetering et al., 2020) can remedy this by allowing a node (i.e., an area) to begin with the width of a parent, but, if no other nodes are in the same row, the area can grow at the bottom (reclaim space), to form trapezoids instead of rectangles. This still preserves the hierarchical structure, but enlarges small-sized hierarchy elements in deeper levels to increase the readability. To reduce meandering, SRIP can prevent nodes from growing by placing invisible sticky nodes beneath nodes without children which last for a configurable depth.

3.2 Model Training

We trained three different Vision Transformer models for our three visualizations using self-supervised training methods to reduce the need for labeled training data. As a base model, we used a Swinv2 Transformer model which was already trained on the ImageNet dataset (Deng et al., 2009). However, because that dataset is comprised almost entirely of photos of natural objects, we implemented an additional pre-training step on a large corpus of synthetic, random graph visualizations. Similarly to the way Vision Transformers are able to recognize relations between objects in a photo, we expect the pre-training step to enable our models to pick up on relations between graph segments. To improve performance on real argument graphs, we then performed a fine-tuning step on visualizations of argument graphs. For this, we used contrastive fine-tuning. We expect it to be especially well suited for our task, as the training objective of learning to recognize similar objects and differentiating them from unrelated ones aligns well with the goal of graph retrieval. More information is provided in Section A

In order to compare the performance of our relatively specialized Vision Transformers to much larger, universal models, we also fine-tuned OpenAI's model "gpt-4o-2024-08-06", capable of advanced text and image comprehension, on a dataset of argument graph visualizations, generated using our SRIP visualization. Because of its generic nature, we were able to adapt our contrastive fine-tuning strategy for this as well. Additional details are given in Section B.

4 Evaluation

Having introduced the core concepts and related work in the previous section, we now present our evaluation of the vision-based structural argument graph retrieval. We examine the argument graph retrieval task outlined in Section 3. The semantic pre-filter has already been evaluated in other works (Bergmann et al., 2019; Lenz et al., 2019), so we focus on the structural re-ranking part of our pipeline. To this end, we use an ideal filter that chooses all relevant argument graphs as determined by the human experts, resulting in a perfectly filtered set of semantically similar arguments. Then, we compare the re-ranking performance of our vision-based pipeline (separately for each visualization design and ViT model) to the baseline

approach of an A* search as described in Section 2 against a benchmark ranking of human experts. Additionally, we perform an ablation study to examine how our pipeline's retrieval time scales with graph complexity. To assess the research question formulated in Section 1—Are vision-based graph similarities *more efficient* than and *equally effective* as ones based on GED for the retrieval of argument graphs?—we evaluate the following hypotheses:

H1 (Effectiveness)**.** The retrieval quality of vision-based structural similarity computation closely approximates those of an A* search.

H2 (Efficiency)**.** Vision-based structural similarity computation greatly reduces retrieval times compared to A* search by utilizing GPUs.

H3 (Specialization)**.** Contrastive fine-tuning increases the effectiveness of ViT models compared to pre-training only.

4.1 Experimental Setup

For our evaluation, we implemented the visualization strategies in Python using Matplotlib (Hunter, 2007) and set up a training and inference pipeline that is publicly available on GitHub.[1] To allow comparisons with the existing approach, we used the corpus of annotated microtexts (Peldszus and Stede, 2016) containing 110 argument graphs with the same 24 queries as Bergmann et al. (2019). Half of these queries do not contain any S-node (only one I-node), while the other half contains up to two S-nodes. The queries come with a reference ranking from human experts, which we use to evaluate the retrieval quality of our approach. As part of an ongoing project, we have developed an additional set of 15 more complex queries with corresponding expert ranking having at least two S-nodes that we also include in our evaluation to better assess the scalability of our approach. The A* search was conducted using the original implementation of the authors[2] with the Universal Sentence Encoder (USE) (Cer et al., 2018) embedding model (their best performing variant). To ensure a fair comparison, we use the same ideal semantic pre-filter based on expert rankings for the A* search. In total, we perform six experiments: one for each of our visualizations (Treemap, Logical, and SRIP) using only pre-trained models and one for each visualization with the fine-tuned models.

[1] github.com/recap-utr/vision-retrieval (MIT license)
[2] github.com/recap-utr/argument-graph-retrieval

We use the following metrics to assess our hypotheses: DURATION, Average Precision (AP) (Turpin and Scholer, 2006), Normalized Discounted Cumulative Gain (NDCG), and CORRECTNESS/COMPLETENESS (Cheng et al., 2010). All metrics except for CORRECTNESS are in the range $[0, 1]$, with higher values indicating better retrieval quality. CORRECTNESS is in the range $[-1, 1]$ with -1 meaning an inversely correct ranking, 0 meaning random ordering, and 1 meaning a correct ranking. For our vision-based models, DURATION only includes the time to embed the visualized argument graphs and compute the cosine similarities for re-ranking. These durations are measured on a single Nvidia Tesla V100 GPU and are averaged over 10 runs. The time to visualize the argument graph is not included as it heavily depends on the implementation of the visualization algorithm. In a practical application, the visualizations of a large case base would most likely be cached, contributing only to the one-time cost of creating the case base. The A* computations are performed on 2019 MacBook Pro with an 8-core Intel Core i9 CPU.

4.2 Results and Discussion

Having outlined our setup, we now present the results of our evaluation as shown in Table 1, starting with the set of simple queries used in previous work and then moving on to the more complex queries.

Simple Queries Regarding NDCG, the deviations between different visualizations and models are quite small, although the fine-tuned model for Treemaps and the pre-trained model for the SRIP visualization marginally outperform the other models. Contrary to our expectations, the pre-trained SRIP model, not A*, delivers the best retrieval quality across all metrics. The CORRECTNESS for all models (including the baseline) is very low, indicating that the queries are too limited for any of the approaches to closely match the ranking of the human experts. However, our vision models seem to be more capable in placing the most important queries at the beginning of the ranking which is over proportionally valued by NDCG.

Regarding DURATION, the initial embedding process of our vision models for the argument graphs within the case base takes between 95% and 108% of the entire retrieval time using A* search with Treemaps taking the longest time. This only has to be done once upfront, meaning that the embeddings can be cached in main memory and

Table 1: Evaluation results for all queries. The column FT refers to the use of contrastive fine-tuning in addition to pre-training. EMB is time in seconds to embed all 110 argument graphs (upfront cost), while DUR measures the time for re-ranking the queries. For OpenAI, the duration is defined by the API request.

Model	FT	Queries	NDCG	AP	COR	COM	DUR	EMB
Treemap	✓	Simple	**0.92**	1.00	0.10	1.00	0.02	29.45
Treemap	✗	Simple	0.91	1.00	0.09	1.00	0.02	26.78
Logical	✓	Simple	0.90	1.00	-0.05	1.00	0.02	**25.87**
Logical	✗	Simple	0.91	1.00	0.07	1.00	0.02	26.57
SRIP	✓	Simple	0.90	1.00	-0.05	1.00	0.02	28.63
SRIP	✗	Simple	**0.92**	1.00	**0.11**	1.00	0.02	26.03
GPT-4o	–	Simple	0.91	1.00	-0.021	1.00	195.74	–
A*	–	Simple	0.85	1.00	0.05	1.00	27.16	–
Treemap	✓	Complex	0.94	1.00	0.38	1.00	0.01	26.42
Treemap	✗	Complex	0.91	1.00	0.21	1.00	0.01	25.67
Logical	✓	Complex	**0.98**	1.00	**0.68**	1.00	0.01	**25.51**
Logical	✗	Complex	0.96	1.00	0.66	1.00	0.01	25.46
SRIP	✓	Complex	0.97	1.00	0.62	1.00	0.01	30.84
SRIP	✗	Complex	0.95	1.00	0.59	1.00	0.01	26.42
GPT-4o	–	Complex	0.91	1.00	0.20	1.00	96.53	–
A*	–	Complex	0.95	1.00	0.632	1.00	199	–

reused for each query. The time needed for retrieval using the GPT-4o model is the longest (at 7 times the processing time of A*) and also has the highest fluctuations. This likely stems from the rather complex model (although OpenAI does not disclose the number of parameters) and the heterogeneous workload of the API.

Complex Queries When using more complex queries, all models perform better. There are minor gains in regard to NDCG (from 0 to 0.08) and especially CORRECTNESS (from 0.12 to 0.73) for the vision models. This is expected as the complex queries carry more information which can be visualized and embedded. Lack of information in simple queries is a problem specially for the trivial queries, with 0 S-nodes, where our visualizations only produce an unicolored image that does not enable the derivation of any meaningful graph structure. This is likely also the reason why our Treemap performs worst, as it only displays S-nodes and therefore contain less information. Our best model is the fine-tuned Logical model, outperforming the other visualization in all retrieval quality metrics. This suggests that the evaluated graphs were not complex enough to demonstrate the advantages of SRIP.

Comparing the DURATION to those of the simple queries, we see that the value for GPT-4o and our vision models scales linearly with the number of requests, while the small increase in query complexity does not have any noticeable effects. On the other hand, the added complexity of the query graphs over proportionally influences A* processing times. These noticeably lower request processing times together with the improved retrieval quality leads to a much better user experience and suitability for a real argument retrieval machine.

Discussion Overall, H1 can be accepted as the vision-based structural similarity pipeline with non-fine-tuned SRIP for simple queries and fine-tuned Logical for complex queries provides the best retrieval quality based on our metrics. When looking at the gains in retrieval quality for complex queries, it is even plausible that the retrieval quality slightly increases for even more complex queries. H2 can be accepted, as only the new query embeddings and cosine similarities have to be computed with each query, while the bulk of the work, the computation of embeddings for the (large) static case base only has to occur once. Also, the scaling is far superior, based on the durations reported in Table 1 and our scaling study in Section 4.3. H3 has to be partly discarded as the pre-trained model for SRIP outperformed the fine-tuned model for simple queries. However, H3 holds for complex argument graphs.

4.3 Ablation Study on Scaling

In this study, we evaluate how the graph complexity (measured by the number of the graph's S-nodes) affects the computation time of structural similarity. This is sufficient to estimate a graph's complexity, as the number of I-nodes equals the number of S-nodes + 1 for every argument graph we evaluate. To study graph complexity scaling, we chose 117 argument graphs from the Kialo GraphNLI dataset (Agarwal et al., 2022) making up the set of case base argument graphs C with 4-120 S-nodes. As the query, we randomly selected a single argument graph from the same dataset with 2540 S-nodes. As the query's complexity is constant, this setup allows studying the impact of increasing graph complexity on retrieval time in isolation. In this study, we use our SRIP visualization together with our fine-tuned model. This is because, even though our Logical visualization outperformed the SRIP visualization in our evaluation, the SRIP visualization should in theory work better for really deep argument graphs (see Figure 2).

Vision-based similarity computation requires the 3 steps outlined in Section 3: visualization, embedding, and cosine similarity calculation. The scaling behaviors of each of these steps can be seen in Figure 3. The embedding step, as well as the cosine similarity calculation, require constant time and are not influenced by the complexity of the input graphs. The visualization time increases linearly with graph complexity, even though there are several outliers. These could be caused by deviations in the size of the argument graph files, of which the entire content (i.e., also the argumentative text) is read, although only the information about the node types is considered to visualize the argument graph.

For a practical implementation of an argumentation machine, the linear scaling of visualization time in respect to graph complexity is likely not a problem, as only the query has to be visualized at runtime, whereas the case base graphs visualizations and embeddings can be pre-computed.

Comparing the total processing time of our vision-based approach to A*-search, it can be seen clearly that while the processing time using our vision-based approach increases linearly with respect to the number of S-nodes, they over proportionally hurt the performance of the A* search. Regarding the absolute times for both approaches, it is apparent that A* is not viable for retrieval of complex arguments in a production argumentation machine, as a *single* comparison between a complex argument graph with 2540 S-nodes, and an argument graph with more than 8 S-nodes takes at least 1,000s.

4.4 Limitations

While our results are promising, there are some limitations to our approach. In order to layout graphs in a compressed format, we made simplifications such as ignoring I-nodes in treemaps. Also, graphs containing cycles currently cannot be rendered due to our focus on hierarchical visualizations. For large graphs with skewed distributions of nodes (e.g., long chains of ADUs), the ranking quality of our approach may suffer due to large amounts of whitespace. Similarly, for graphs with nearly identical structure but different content, the visualizations may be indistinguishable, potentially leading to poor retrieval quality—which we solved by introducing a semantic pre-filter.

Regarding the vision models, we used relatively small models (197M parameters) with limited training datasets. Graphs having more elements than the model's maximum number of pixels (e.g., 256×256) need to be clipped or downsampled, meaning that some information is lost. Given the scalability of transformers, we anticipate that larger models with more extensive training data could yield improved performance in future evaluations. Lastly, our scaling study disregards the quality of the retrieval for larger argument graphs due to missing ground truth data.

5 Conclusion and Future Work

We proposed a vision-based pipeline for argument graph retrieval based on their structure that builds on the output of AM systems. It works by filtering for semantically similar arguments, visualizing their graph representations, embedding these rendered images with a vision model, and finally ranking the arguments based on the cosine similarity to the query's embedding. The research question whether vision-based argument retrieval can provide a faster and more scalable alternative to A* search for structural argument graph retrieval can be affirmed; however, not every dataset of arguments allows for the effective use of the vision-based approach. On the one hand, our evaluation suggests that there is a minimum complexity argument graphs should have for our vision-based

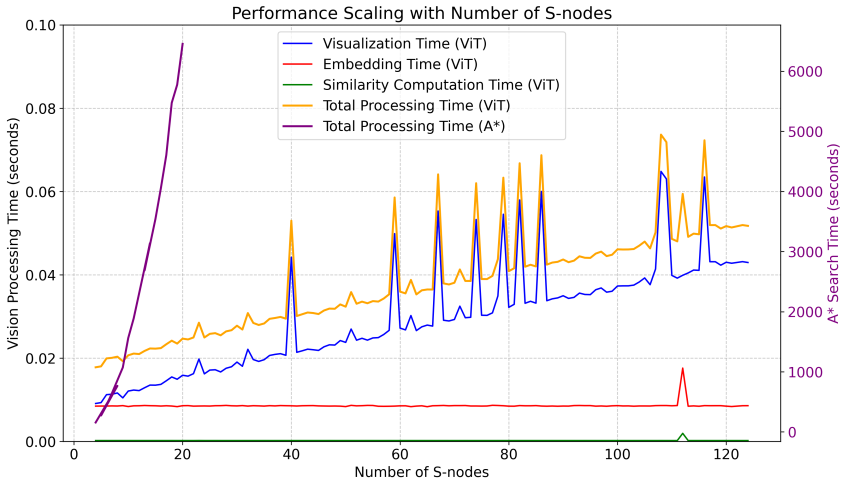

Figure 3: Processing times of vision-based and A* retrieval for graphs between 4 and 124 S-nodes.

approach to be able to perform meaningful similarity computation. On the other hand, the information which can be displayed in the limited input window of a vision model imposes an upper limit on argument graph complexity which can be sensibly processed using our approach. Regarding efficiency and scaling, the use of embeddings allows storing a uniform, query-independent representation of the original argument graphs, which can be pre-computed to allow for fast comparisons even across large case bases. While we investigated a re-ranking task for our evaluation, vision-based argument retrieval could also be used to enhance the pipeline proposed by Bergmann et al. (2019): Our vision-based retrieval could serve as a second pre-filter to further decrease the search space of the expensive A* search to ensure that only graphs that are semantically and structurally similar are considered at all. This pipeline enables to construct mappings between queries and case base graphs which are absent in purely vision-based retrieval.

One possible avenue for future work is to investigate the use of more detailed argumentation schemes (Walton, 2013) to differentiate between additional types of S-nodes in the argument graph. As Lenz et al. (2019) showed, using schemes can have a positive impact on the retrieval quality. A key challenge in this regard is the inclusion of the additional information into the generated visualizations. Furthermore, we focused on a single model training pipeline. As has been shown before (Qu et al., 2020; Asai et al., 2022; Khan et al., 2022; Wang et al., 2022; El-Nouby et al., 2021; Grill et al., 2020; Tian et al., 2021), training pipeline refinements can notably improve the predictions. An open question here is how to apply existing training techniques for texts or pictures for our graph visualizations.

Additionally, our evaluation was limited to a single dataset (Peldszus and Stede, 2015). Future work should verify whether the findings can be generalized to other datasets, especially with more complex argument graphs and extended ADU relations. One candidate for this could be the AbstRCT dataset (Mayer et al., 2020).

References

Agnar Aamodt and Enric Plaza. 1994. Case-Based Reasoning - Foundational Issues, Methodological Variations, and System Approaches. *AI Commun.*

Vibhor Agarwal, Sagar Joglekar, Anthony P. Young, and Nishanth Sastry. 2022. Graphnli: A graph-based natural language inference model for polarity prediction in online debates. In *The ACM Web Conference (TheWebConf)*.

Jason Ansel, Edward Yang, Horace He, Natalia Gimelshein, Animesh Jain, Michael Voznesensky, Bin Bao, Peter Bell, David Berard, Evgeni Burovski, Geeta Chauhan, Anjali Chourdia, Will Constable, Alban Desmaison, Zachary DeVito, Elias Ellison, Will Feng, Jiong Gong, Michael Gschwind, Brian Hirsh, Sherlock Huang, Kshiteej Kalambarkar, Laurent Kirsch, Michael Lazos, Mario Lezcano, Yanbo Liang, Jason Liang, Yinghai Lu, CK Luk, Bert Maher, Yunjie Pan, Christian Puhrsch, Matthias Reso, Mark Saroufim, Marcos Yukio Siraichi, Helen Suk, Michael Suo, Phil Tillet, Eikan Wang, Xiaodong Wang, William Wen, Shunting Zhang, Xu Zhao, Keren Zhou, Richard Zou, Ajit Mathews, Gregory Chanan, Peng Wu, and Soumith Chintala. 2024. PyTorch 2: Faster Machine Learning Through Dynamic Python Bytecode Transformation and Graph Compilation. In *29th ACM International Conference on*

Architectural Support for Programming Languages and Operating Systems, Volume 2 (ASPLOS '24). ACM.

Akari Asai, Timo Schick, Patrick Lewis, Xilun Chen, Gautier Izacard, Sebastian Riedel, Hannaneh Hajishirzi, and Wen tau Yih. 2022. Task-aware retrieval with instructions. *Preprint*, arXiv:2211.09260.

Margaret E. Baron. 1969. A Note on the Historical Development of Logic Diagrams: Leibniz, Euler and Venn. *The Mathematical Gazette*, 53(384):113–125.

Elias Bassani. 2022. Ranx: A Blazing-Fast Python Library for Ranking Evaluation and Comparison. In *Advances in Information Retrieval*, pages 259–264, Cham. Springer International Publishing.

Ralph Bergmann and Yolanda Gil. 2014. Similarity assessment and efficient retrieval of semantic workflows. *Information Systems*, 40:115–127.

Ralph Bergmann, Mirko Lenz, Stefan Ollinger, and Maximilian Pfister. 2019. Similarity Measures for Case-Based Retrieval of Natural Language Argument Graphs in Argumentation Machines. In *Proceedings of the Thirty-Second International Florida Artificial Intelligence Research Society Conference*, pages 329–334, Sarasota, Florida, USA. AAAI Press.

H. Bunke. 1997. On a relation between graph edit distance and maximum common subgraph. *Pattern Recognition Letters*, 18(8):689–694.

Hans-Dieter Burkhard and Michael M. Richter. 2001. On the Notion of Similarity in Case Based Reasoning and Fuzzy Theory. In Sankar K. Pal, Tharam S. Dillon, and Daniel S. Yeung, editors, *Soft Computing in Case Based Reasoning*, pages 29–45. Springer London, London.

Hongyun Cai, Vincent W Zheng, and Kevin Chen-Chuan Chang. 2018. A comprehensive survey of graph embedding: Problems, techniques, and applications. *IEEE transactions on knowledge and data engineering*, 30(9):1616–1637.

Daniel Cer, Yinfei Yang, Sheng-yi Kong, Nan Hua, Nicole Limtiaco, Rhomni St John, Noah Constant, Mario Guajardo-Cespedes, Steve Yuan, Chris Tar, Yun-Hsuan Sung, Brian Strope, and Ray Kurzweil. 2018. Universal Sentence Encoder. *arXiv:1803.11175 [cs]*.

Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey E. Hinton. 2020a. A simple framework for contrastive learning of visual representations. *CoRR*, abs/2002.05709.

Ting Chen, Simon Kornblith, Kevin Swersky, Mohammad Norouzi, and Geoffrey E. Hinton. 2020b. Big self-supervised models are strong semi-supervised learners. *CoRR*, abs/2006.10029.

Weiwei Cheng, Michaël Rademaker, Bernard De Baets, and Eyke Hüllermeier. 2010. Predicting Partial Orders: Ranking with Abstention. In *Machine Learning and Knowledge Discovery in Databases*, Lecture Notes in Computer Science, pages 215–230, Barcelona, Spain. Springer.

Carlos Iván Chesñevar, Jarred McGinnis, Sanjay Modgil, Iyad Rahwan, Chris Reed, Guillermo Ricardo Simari, Matthew South, Gerard Vreeswijk, and Steven Willmott. 2006. Towards an argument interchange format. *The Knowledge Engineering Review*, 21(04):293.

Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. 2009. Imagenet: A large-scale hierarchical image database. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pages 248–255.

Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. 2021. An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale. In *International Conference on Learning Representations*.

Alaaeldin El-Nouby, Natalia Neverova, Ivan Laptev, and Hervé Jégou. 2021. Training vision transformers for image retrieval. *arXiv preprint arXiv:2102.05644*.

William Falcon and The PyTorch Lightning team. 2019. PyTorch Lightning.

Kenneth D Forbus, Dedre Gentner, and Keith Law. 1995. MAC/FAC - A Model of Similarity-Based Retrieval. *Cognitive Science*, 19(2):141–205.

Tianyu Gao, Xingcheng Yao, and Danqi Chen. 2021. Simcse: Simple contrastive learning of sentence embeddings. *CoRR*, abs/2104.08821.

Jean-Bastien Grill, Florian Strub, Florent Altché, Corentin Tallec, Pierre H. Richemond, Elena Buchatskaya, Carl Doersch, Bernardo Ávila Pires, Zhaohan Daniel Guo, Mohammad Gheshlaghi Azar, Bilal Piot, Koray Kavukcuoglu, Rémi Munos, and Michal Valko. 2020. Bootstrap your own latent: A new approach to self-supervised learning. *CoRR*, abs/2006.07733.

Aditya Grover and Jure Leskovec. 2016. node2vec: Scalable feature learning for networks. In *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 855–864.

Annette Hautli-Janisz, Zlata Kikteva, Wassiliki Siskou, Kamila Gorska, Ray Becker, and Chris Reed. 2022. QT30: A corpus of argument and conflict in broadcast debate. In *Proceedings of the Thirteenth Language Resources and Evaluation Conference*, pages 3291–3300, Marseille, France. European Language Resources Association.

Maximilian Hoffmann, Lukas Malburg, Nico Bach, and Ralph Bergmann. 2022. GPU-Based Graph Matching for Accelerating Similarity Assessment in Process-Oriented Case-Based Reasoning. In *Case-Based Reasoning Research and Development*, pages 240–255, Cham. Springer International Publishing.

J. D. Hunter. 2007. Matplotlib: A 2d graphics environment. *Computing in Science & Engineering*, 9(3):90–95.

Vidit Jain and Manik Varma. 2011. Learning to re-rank: query-dependent image re-ranking using click data. In *Proceedings of the 20th international conference on World wide web*, pages 277–286.

B. Johnson and B. Shneiderman. 1991. Tree-maps: A space-filling approach to the visualization of hierarchical information structures. In *Proceeding Visualization '91*, pages 284–291.

Salman Khan, Muzammal Naseer, Munawar Hayat, Syed Waqas Zamir, Fahad Shahbaz Khan, and Mubarak Shah. 2022. Transformers in vision: A survey. *ACM computing surveys (CSUR)*, 54(10s):1–41.

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

John Lawrence and Chris Reed. 2019. Argument Mining: A Survey. *Computational Linguistics*, 45(4):765–818.

Mirko Lenz, Stefan Ollinger, Premtim Sahitaj, and Ralph Bergmann. 2019. Semantic Textual Similarity Measures for Case-Based Retrieval of Argument Graphs. In *Case-Based Reasoning Research and Development*, volume 11680 of *Lecture Notes in Computer Science*, pages 219–234, Otzenhausen, Germany. Springer International Publishing.

Ze Liu, Han Hu, Yutong Lin, Zhuliang Yao, Zhenda Xie, Yixuan Wei, Jia Ning, Yue Cao, Zheng Zhang, Li Dong, Furu Wei, and Baining Guo. 2022. Swin Transformer V2: Scaling Up Capacity and Resolution. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12009–12019.

Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo. 2021. Swin Transformer: Hierarchical Vision Transformer Using Shifted Windows. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 10012–10022.

Lorenzo Livi and Antonello Rizzi. 2013. The graph matching problem. *Pattern Analysis and Applications*, 16(3):253–283.

TorchVision maintainers and contributors. 2016. Torchvision: Pytorch's computer vision library. `https://github.com/pytorch/vision`.

Santiago Marro, Elena Cabrio, and Serena Villata. 2022. Graph Embeddings for Argumentation Quality Assessment. In *Findings of the Association for Computational Linguistics: EMNLP 2022*, pages 4154–4164, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Tobias Mayer, Elena Cabrio, and Serena Villata. 2020. Transformer-based argument mining for healthcare applications. In *ECAI 2020 - 24th European Conference on Artificial Intelligence*, volume 325 of *Frontiers in Artificial Intelligence and Applications*, pages 2108–2115. IOS Press.

Mirjam Minor, Stefania Montani, and Juan A. Recio-García. 2014. Process-oriented case-based reasoning. *Information Systems*, 40:103–105.

Daniel Carlos Guimaraes Pedronette and Ricardo da S Torres. 2013. Image re-ranking and rank aggregation based on similarity of ranked lists. *Pattern Recognition*, 46(8):2350–2360.

Andreas Peldszus and Manfred Stede. 2013. From Argument Diagrams to Argumentation Mining in Texts - A Survey. *IJCINI*, 7(1):1–31.

Andreas Peldszus and Manfred Stede. 2015. An annotated corpus of argumentative microtexts. In *Argumentation and Reasoned Action: Proceedings of the 1st European Conference on Argumentation, Lisbon*, volume 2, pages 801–815.

Andreas Peldszus and Manfred Stede. 2016. An Annotated Corpus of Argumentative Microtexts. In *Argumentation and Reasoned Action: Proceedings of the 1st European Conference on Argumentation*, volume 2, pages 801–816, Lisbon, Portugal. College Publications.

Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. 2014. Deepwalk: Online learning of social representations. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 701–710.

Yingqi Qu, Yuchen Ding, Jing Liu, Kai Liu, Ruiyang Ren, Wayne Xin Zhao, Daxiang Dong, Hua Wu, and Haifeng Wang. 2020. Rocketqa: An optimized training approach to dense passage retrieval for open-domain question answering. *arXiv preprint arXiv:2010.08191*.

Chris Reed. 2006. Preliminary results from an argument corpus. *Linguistics in the twenty-first century*, pages 185–196.

Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. 2015. Imagenet large scale visual recognition challenge. *Preprint*, arXiv:1409.0575.

Hans-Jorg Schulz. 2011. Treevis.net: A Tree Visualization Reference. *IEEE Computer Graphics and Applications*, 31(6):11–15.

Maria Skeppstedt, Andreas Peldszus, and Manfred Stede. 2018. More or less controlled elicitation of argumentative text: Enlarging a microtext corpus via crowdsourcing. In *Proceedings of the 5th Workshop on Argument Mining*, pages 155–163, Brussels, Belgium. Association for Computational Linguistics.

Christian Stab and Iryna Gurevych. 2017. Argument annotated essays (version 2).

Hao Tang, Donghong Ji, Chenliang Li, and Qiji Zhou. 2020. Dependency graph enhanced dual-transformer structure for aspect-based sentiment classification. In *Proceedings of the 58th annual meeting of the association for computational linguistics*, pages 6578–6588.

Yuandong Tian, Xinlei Chen, and Surya Ganguli. 2021. Understanding self-supervised learning dynamics without contrastive pairs. *CoRR*, abs/2102.06810.

Andrew Turpin and Falk Scholer. 2006. User performance versus precision measures for simple search tasks. In *SIGIR 2006: Proceedings of the 29th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, Seattle, Washington, USA, August 6-11, 2006*, pages 11–18.

Huub van de Wetering, Nico Klaassen, and Michael Burch. 2020. Space-Reclaiming Icicle Plots. In *2020 IEEE Pacific Visualization Symposium (PacificVis)*, pages 121–130.

Frans H. Van Eemeren. 2018. *Argumentation Theory: A Pragma-Dialectical Perspective*, volume 33 of *Argumentation Library*. Springer International Publishing, Cham.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is All you Need. In *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc.

Jacky Visser, Barbara Konat, Rory Duthie, Marcin Koszowy, Katarzyna Budzynska, and Chris Reed. 2020. Argumentation in the 2016 us presidential elections: annotated corpora of television debates and social media reaction. *Language Resources and Evaluation*, 54(1):123–154.

Marilyn A Walker, Jean E Fox Tree, Pranav Anand, Rob Abbott, and Joseph King. 2012. A corpus for research on deliberation and debate. In *LREC*, volume 12, pages 812–817. Istanbul, Turkey.

Douglas Walton. 2013. *Argumentation Schemes for Presumptive Reasoning*. Routledge.

Liang Wang, Nan Yang, Xiaolong Huang, Binxing Jiao, Linjun Yang, Daxin Jiang, Rangan Majumder, and Furu Wei. 2022. Simlm: Pre-training with representation bottleneck for dense passage retrieval. *arXiv preprint arXiv:2207.02578*.

Xiaogang Wang, Shi Qiu, Ke Liu, and Xiaoou Tang. 2013. Web image re-ranking using query-specific semantic signatures. *IEEE transactions on pattern analysis and machine intelligence*, 36(4):810–823.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Remi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander Rush. 2020. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45, Online. Association for Computational Linguistics.

Mengjia Xu. 2021. Understanding graph embedding methods and their applications. *SIAM Review*, 63(4):825–853.

A Reproducibility

In the following section, we outline how we trained our vision models and which dataset was used for the sake of reproducibility. For both training steps we used PyTorch (Ansel et al., 2024) version 2.5.0 together with PyTorch Lightning (Falcon and The PyTorch Lightning team, 2019). The vision transformer models were integrated via the transformers package (Wolf et al., 2020) (version 4.45.2). The evaluation is based on the ranx package (Bassani, 2022).

A.1 Pre-Training

For each of the visualizations, a separate large Swinv2 Transformer model (Liu et al., 2022) (released under Apache 2.0 License) with 195M parameters was pre-trained. We chose this model for its improved efficiency in relation to the original Vision Transformer model (Dosovitskiy et al., 2021) and its architecture which makes use of hierarchical feature maps and should align well with the nature of hierarchical graph drawings. A checkpoint which has been trained on the ImageNet-1k dataset (Russakovsky et al., 2015) is used as a starting point, which should speed up training compared to completely random initial weights. For the training, we used an Auto-Encoder setup, where the Swinv2 model was used as a encoder, transforming an input image into corresponding embeddings. During the training, a very simple decoder (a single linear layer) is used to reconstruct a lower resolution form of the original image using the embeddings provided by the encoder. The MSE loss is computed between the raw pixel values of the original

images (resized to 32x32px) and the reconstructed image. An AdamW optimizer with a learn rate of 0.001 is used. Additionally, we used early stopping after 3 epochs without a reduction in validation loss. The models are trained with a batch-size of 32 for a maximum of 50 epochs on 6 Nvidia Tesla V100 GPUs. The actual training time was 20-23 epochs (174-198 GPU hours).

As our pre-training dataset, we used a dataset of synthetic argument graphs. For every of our three visualizations, we generated 1.2 million random graphs with a maximum depth of 9 and a maximum branching number of 7, which decreases with increased depth. The motivation behind this is to generate graphs which deviate from each other; however the minimum area allocated to a single node in the corresponding visualization is fixed by the limited depth and number of siblings. The resulting images are then de-duplicated using *fclones* [3], which left us with 1,062,679 samples for the Logical model, 1,062,513 for the Treemaps model and 917,558 for the SRIP model. Of those samples, we always chose 90% as training samples and the remaining 10% as test samples.

A.2 Fine-Tuning

Each of the models from the pre-training stage are fine-tuned on a corpus of 6474 argument graphs (see Table 2) after filtering out too complex graphs which took longer than 3s to visualize. After deduplication with *fclones*, this left us with 4317 SRIP images, 4309 Logical images and 4173 Treemap images. The setup used for contrastive fine-tuning is derived from SimCLR (Chen et al., 2020a):

1. Each image x from the training batch is randomly augmented twice which generates two contrastive views of every input which represent each others positive pairs: q, k.

2. q and k are encoded using the encoder network (the pre-trained Swin Transformer v2 model), resulting in the embeddings e_q and e_k.

3. The embedding dimensionalities are reduced by passing them through an MLP projection head to prevent the curse of dimensionality (Chen et al., 2020b).

4. A contrastive loss is calculated between every element's corresponding image view and every other element in the batch (in-batch negatives) on the reduced embeddings.

The contrastive views are derived from the original images by using the following transformations: (i) random horizontal flips, (ii) random vertical flips, (iii) Gaussian Blur (iv) random crop (an area of 40% - 90% of the original image is resized to the original dimensions) and (v) dropout to simulate random noise. For these transformations, we used the implementations from torchvision (maintainers and contributors, 2016) (version 0.20.0). The first four transforms are derived from the original SimCLR transforms (Chen et al., 2020a); dropout is inspired by (Gao et al., 2021). It should be noted that color jitter, as one of the most important transforms (Chen et al., 2020a) could not be used. This is because a change of color for a node might completely change its meaning in all of our visualizations and therefore represent a different graph structure. The following contrastive loss is used (Chen et al., 2020a):

$$f(q,k) = \exp\left(\frac{\mathrm{sim}(q,k)}{\tau}\right) \tag{1}$$

$$\ell_i^{\text{NT-Xent}} = -\log \frac{f(q_i,k_i)}{f(q_i,k_i) + \sum_{j\neq i} f(q_i,k_j)} \tag{2}$$

for i, j in $\{0, \ldots, \text{batch\_size}\}$ where $\mathrm{sim}(\cdot,\cdot)$ denotes cosine similarity, and τ represents *temperature* as a hyperparameter. Our models were trained with a hidden dimension of 64, $\tau = 0.07$ and a weight decay of 0.0001.

As an optimizer, AdamW with a learn rate of 0.0005 is used. Additionally a *Cosine Annealing Scheduler* was used for the learn rate with a maximum of 500 iterations and a minimum learn rate of 0.00001. The same early stopping criterion was applied as for pre-training, however no fine-tuned model training was interrupted early. The fine-tuning is performed with a batch-size of 16 (i.e., 16 contrastive pairs) for a maximum of 500 epochs on 6 Nvidia Tesla V100 GPUs.

B GPT-4o Fine-Tuning

To emulate contrastive training with the limited interface OpenAI provides (i.e., training samples have to represent a conversation with a prompt and an expected answer from the model), we generated 900 samples containing two SRIP visualizations

[3] github.com/pkolaczk/fclones

Table 2: Argument graph corpora used to construct our fine-tuning dataset.

Dataset	Source	Description
Kialo Graph-NLI	Agarwal et al. (2022)	Graphs model discussion trees on Kialo, an online debates platform
Araucaria	Reed (2006)	Corpus of analyzed argumentation, constructed using the Araucaria tool
IAC	Walker et al. (2012)	A corpus for research on deliberation and debate
QT30	Hautli-Janisz et al. (2022)	Argument and conflict in broadcast debate
US2016	Visser et al. (2020)	Television debates and social media reactions to the 2016 US presidential elections
Persuasive Essays	Stab and Gurevych (2017)	Annotated persuasive essays
Microtexts Part 2	Skeppstedt et al. (2018)	Short argumentative texts

each. The model's task during the training process was to predict whether the images represent the same argument graph or a different graph. 450 samples contained two contrastive views of the same graph (see above) while the remaining 450 samples contained two different graphs. The model was trained for a single epoch with a batch size of 1 and a LR multiplier of 2. The training took about one hour.

During evaluation, we provide the model aSRIP representation of the query and the SRIP visualization of the retrieval candidates acquired from the MAC phase. The model's task is ordering the case graphs based on their relevance to the query. To eliminate any run-to-run variance, the temperature during evaluation is set to 0.

Note: We only trained the model for a single epoch as prior experiments indicated that the model's performance degraded for models with more epochs. This is most likely because our training dataset consisted only of singular, short answers ("Are the images visualizations of the same or different graphs?" $\rightarrow$ "same" or "different"") which caused the further trained checkpoints to adapt to this and only provide too short and therefore largely incomplete answers during the evaluation.