Document Structure in Long-Document Transformers

Jan Buchmann[1*], **Max Eichler**[1*]
Jan-Micha Bodensohn[2], **Ilia Kuznetsov**[1], **Iryna Gurevych**[1]
[1] Ubiquitous Knowledge Processing Lab (UKP Lab)
Department of Computer Science and Hessian Center for AI (hessian.AI)
Technical University of Darmstadt
www.ukp.tu-darmstadt.de
[2] DFKI and Data and AI Systems Lab, Technical University of Darmstadt

Abstract

Long documents often exhibit structure with hierarchically organized elements of different functions, such as section headers and paragraphs. Despite the omnipresence of document structure, its role in natural language processing (NLP) remains opaque. Do long-document Transformer models acquire an internal representation of document structure during pre-training? How can structural information be communicated to a model after pre-training, and how does it influence downstream performance? To answer these questions, we develop a novel suite of probing tasks to assess structure-awareness of long-document Transformers, propose general-purpose structure infusion methods, and evaluate the effects of structure infusion on QASPER and Evidence Inference, two challenging long-document NLP tasks. Results on LED and LongT5 suggest that they acquire implicit understanding of document structure during pre-training, which can be further enhanced by structure infusion, leading to improved end-task performance. To foster research on the role of document structure in NLP modeling, we make our data and code publicly available[1].

1 Introduction

Long documents such as news articles, scientific papers, and clinical reports play a vital role in many human activities. These documents are usually organized into chapters, sections, subsections, and paragraphs, i.e. they are structured. This helps humans in navigating documents (Guthrie et al., 1991; Nguyen et al., 2021) and building a mental model of the content (Taylor and Beach, 1984; Meyer et al., 1980). The example in Fig. 1 shows how the hierarchy of sections and subsections helps when looking for the size of a dataset in an NLP paper: one would go via the "Experiments" section to the "Datasets" subsection.

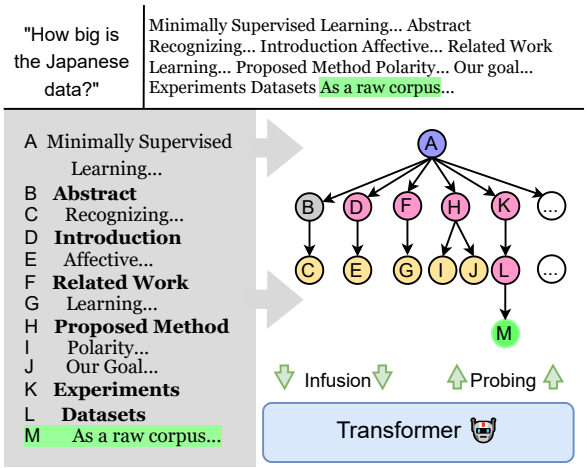

Figure 1: Transformer models receive unstructured text as input (top right) – yet long texts exhibit structure, which helps in finding information (bottom). We investigate whether Transformers learn representations of document structure during pre-training (§4), whether structure-awareness can be enhanced by infusion after pre-training (§5), and what effects infusion has on downstream task performance. Source: QASPER dataset, arxiv ID 1909.00694 (Dasigi et al., 2021).

Although structure is omnipresent and useful to humans, existing long-document Transformers (e.g. Ainslie et al. 2020; Beltagy et al. 2020; Ivgi et al. 2023) operate with linearized textual input: documents are converted to flat character strings, removing the distinction between different functional elements and their hierarchy (Fig. 1, top right).

Understanding the structural capabilities of long-document Transformers is important both theoretically and practically. From a theoretical standpoint, prior work in probing has demonstrated the ability of Transformers to learn syntactic representations on the sentence level (Hewitt and Liang, 2019) – yet little is known about the ability to induce higher-level discourse structures from linearized text. Probing methodology and datasets for this

[*] Equal contribution
[1] https://github.com/UKPLab/eacl2024-doc-structure, under Apache-2.0 license.

Proceedings of the 18th Conference of the European Chapter of the Association for Computational Linguistics
Volume 1: Long Papers, pages 1056–1073

investigation are missing. From a practical perspective, recent works demonstrate that structure-aware modeling can improve downstream task performance (Li et al., 2023; Cao and Wang, 2022; Ruan et al., 2022) – yet existing studies are limited to task-specific architectures and data formats, making it hard to generalize the findings to new tasks and document types. General-purpose methodology for communicating structural information to Transformer models is yet to be established.

Our work aims to close this gap. Instead of committing to a specific document format, we build the a task- and format-agnostic formalism of Intertextual graphs (ITG, Kuznetsov et al. 2022) to encode structure obtained from the original documents (§3).

Building on this formalism, we investigate the role of document structure in long document Transformers from two experimental angles: Probing and downstream tasks. We introduce a novel suite of probing tasks in §4 to investigate structure-awareness of pre-trained Transformer models. Probing experiments on two widely used long document Transformer models – LED (Beltagy et al., 2020) and LongT5 (Guo et al., 2022) – suggest that Transformers do acquire the ability to represent document structure during pre-training, but that there is room for improvement. Consequently, in §5, we test the effect of adding structural information to the Transformer input. We devise a general-purpose structure infusion kit and employ it in experiments on our probing suite and two challenging long-document NLP datasets: QASPER (Dasigi et al., 2021) and Evidence Inference (DeYoung et al., 2020). The results suggest that structure-awareness can be enhanced via infusion, leading to up to 6.8 F1 points increase on downstream tasks. Our work lays the foundation for the systematic analysis of the role of document structure in long document modeling.

2 Background

Document structure. The term "structure" is used ambiguously for textual documents. *Rhetorical structure* is the hierarchical organization of semantic units, usually latent and not available for explicit processing. (Kintsch and van Dijk, 1978; Mann and Thompson, 1987). *Abstract structure* refers to the hierarchical organization of a text into

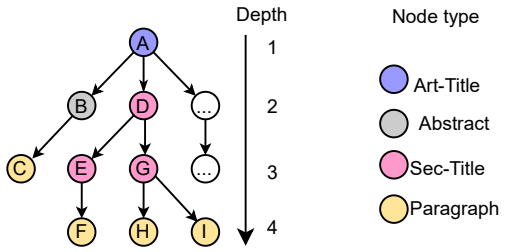

Figure 2: Document Graph. Black arrows show `parent` edges, `next` edges between alphabetically consecutive nodes are omitted for clarity. Node depth and node type information are infused in §5.

elements such as sections, paragraphs, and lists[2] (Nunberg, 1990; Power et al., 2003). *Concrete*, or *visual structure*, includes aspects of typesetting such as font size, spacing and the location of textual elements in a typeset text, classically ordered into pages (Power et al., 2003). In this work, we focus on the study of abstract document structure as the direct author expression of textual organization.

Long-document Transformers. The memory and computational requirements of the standard Transformer architecture (Vaswani et al., 2017) scale quadratically with the input length, making it hard to process long documents under computational constraints. Several innovations for increased efficiency have been proposed, surveyed by Tay et al. (2022). A popular and well-performing approach is the combination of local attention with a varied distribution of global attention (Ainslie et al., 2020; Beltagy et al., 2020; Guo et al., 2022), used by the top 5 models in the Scrolls benchmark for long-document processing (Shaham et al., 2022). We experiment with two representatives for this approach: LED (Beltagy et al., 2020), which is employed in many recent works on long documents (e.g. Dasigi et al. 2021; Cao and Wang 2022) and LongT5 (Guo et al., 2022), the best "base" model on the Scrolls leaderboard at the time of writing[3].

Probing. Probing tasks are diagnostic classification tasks which investigate whether a linguistic feature (e.g. sentence length, word content or syntax tree depth) is encoded in a representation (Conneau et al., 2018; Belinkov, 2022; Rogers et al., 2020). Early work on probing measured the en-

[2]Power et al. (2003) include phenomena such as emphasis and quotation into abstract document structure. They are not considered here, as they are rarely preserved or standardized.

[3]`https://www.scrolls-benchmark.com/leaderboard`, October 2023.

coded knowledge through the delta to a majority baseline or randomly initialized embeddings. Control tasks were introduced as a better approximation of what a probing classifier is able to learn in its own neural representation compared to what linguistic features it can extract from the underlying representations (Hewitt and Liang, 2019). We follow this line of work by designing a novel atomic control setting where we remove contextual information. To measure contextual information beyond a given span, we employ edge probing introduced by Tenney et al., (2019).

Syntax trees have been shown to be encoded in BERT (Hewitt and Manning, 2019), but the representation of higher-order document structure has not been investigated. For the first time, we show that long-document Transformers internally represent several aspects of document structure, and that this internal representation can be enhanced.

Document structure in Transformers. Existing approaches that make use of abstract document structure in Transformers broadly fall into two categories. In *hierarchical processing* (Zhang et al., 2022; Qi et al., 2022; Liu and Lapata, 2019; Ruan et al., 2022), complex, task specific architectures are built, from which results and analyses are hard to generalize. In *structure infusion*, additional structural information is added to pre-trained Transformer models. We employ the latter setting, because methods and models can be reused and analyzed more easily. Structure infusion through special tokens (Aghajanyan et al. 2022; Fisch et al. 2019), attention masks (Liu et al., 2021; Hong et al., 2022), absolute (Bai et al. 2021) or relative position embeddings (Cao and Wang, 2022) has been shown to improve downstream task performance. Here, we combine special tokens and position embeddings which only require changes at the input layer, making them easily transferrable to other transformer models.

3 Representing Structure

Formalism. We model the abstract structure of a document (Power et al. 2003, see §2) as an ordered graph G (Fig. 2) as in Kuznetsov et al. (2022), using their notation. Structural elements such as section headings or paragraphs are represented as a set of typed nodes N^G. The node types correspond to the *function* of the element in the document. We consider the types `article-title`, `section-title`, `abstract`, and `paragraph`[4]. The set of typed, directed edges E^G encodes the *hierarchical organization* of the textual elements with `parent` edges and the linear order with `next` edges. Node function and hierarchical organization can be seen as orthogonal pieces of information that together fully describe the abstract document structure.

Data conversion. All datasets used in the present work were converted to the intertextual graph (ITG) format[5] introduced in Kuznetsov et al. (2022), which is a generic JSON representation of the graph data structure introduced above. Many different types of documents can be easily converted to the ITG format without loss of information on the document structure, including XML or LaTeXfiles. All our methods and experiments are based on ITG, and are therefore dataset agnostic, easily adaptable, and extensible.

4 Probing for Structure

4.1 Probing Suite Design

As the first step towards the systematic study of document structure in long document processing, we propose a suite of seven probing tasks that measure the ability of pre-trained Transformers to capture structural information from their input, described in Tab. 1. For example, the `parent predecessor` probe measures the representation of document hierarchy in a Transformer by learning to distinguish between pairs of document elements (e.g. headings or paragraphs) that are in a parent-child relationship and pairs that are not. As shown in our introduction example, a good representation of the hierarchy can help in locating relevant information in a document (Fig. 1).

All probing tasks are cast as classification and evaluated via accuracy. Assuming a model that computes vector representations of textual nodes, classification is implemented as a linear layer projecting from the representation of a node or a node pair to the label space. If a model has multiple layers, node representations are computed as a weighted sum (Tenney et al., 2019) of the representations from each layer. For tasks on node pairs, the representations of two nodes are concatenated. Only the linear layer and the scalar mix weights are updated during training on the probing task.

[4] We do not consider sentences, as their borders often cannot be extracted unambiguously from English texts.

[5] https://github.com/UKPLab/intertext-graph

Name	Classification task	Labels
`Node type`	Type of n_j with all nodes of type `section` and a tree depth > 1 grouped as `subsection`[1].	`Section`, `subsection`, `paragraph`
`Sibling`	Do n_j and n_k share the same parent n_p?	`Boolean`
`Ancestor`	Is n_j on the `parent` path of n_k and the root n_0?	`Boolean`
`Position`	Position within an ordered set S for all nodes $n_j \in S$ with the same parent n_p.	`Begin`, `inside`, `outside`
`Parent predecessor`	Is n_p the parent of n_j?	`Boolean`
`Tree depth`	Depth of n_j from the root n_0.	`Integer`
`Structural`	Shortest `parent` path between n_j and n_k.	`Integer`

Table 1: Definitions of probing tasks and their labels. $n_{j,k,p,0}$ denote nodes in the document graph G. [1] `Subsection` is a mixture of functional and hierarchical description, so it is not part of the node types defined in §3. It is added to the `node type` probing task to increase the difficulty.

4.2 Experiments and Results

Probing dataset. We instantiate our probing tasks with research papers from the open science platform F1000Research[6]. Based on the pre-processing used for the F1000RD corpus (Kuznetsov et al., 2022) we convert each paper into the ITG format (Fig. 2), removing all non-textual nodes[7]. Removing all papers exceeding the maximum input length of LED (16384 tokens) results in a corpus of 2,499 documents. All probing tasks are balanced through downsampling on document basis, meaning that the label distribution is uniform in most cases (Tab. 5). For some probes, e.g. `tree depth`, not all labels occur in all documents, resulting in a non-uniform label distribution.

Probing architecture. We compare probing of the "vanilla" LED and LongT5 encoders with two control configurations each: *atomic* and *random*. In the atomic control (Fig. 3), nodes are input to the model individually, i.e. without their document context. Comparing the vanilla and atomic configurations shows the effect of contextualization on the representation of structure. For the random control, all model weights except for the embedding layer are re-initialized randomly (Jawahar et al., 2019). It shows the effect of pre-training on the representation of structure. Details on implementation and hyperparameters can be found in Appx. B.2.

Results. In all probes, the accuracy of the vanilla model is higher than the random control (Tab. 2). The difference varies between 34% for LongT5 on `position` and 2.7% for LED on `node type` – a magnitude comparable to reported results from prior work on probing (e.g. Conia and Navigli 2022). This result suggests that LED and LongT5 learn to represent document structure during pre-training, but the effect varies between different aspects of document structure. The cases with small difference between vanilla and random control imply that the input token and position embeddings, not being re-initialized, contain much of the information needed to solve the task. The scores of the atomic control are lower than those of the vanilla configuration on all probes, showing that context helps to represent document structure.

Vanilla LED and LongT5 achieve accuracies of 0.9 on some probes, e.g. `node type`, suggesting that they are able to encode some aspects of structural information well even without its explicit input. It is surprising that the accuracy on the `sibling probe` is far below that of `parent predecessor`, because the information on the parents of two nodes is enough to determine their siblinghood. It seems that the combination of parent information from two nodes in a queried pair is difficult. The `structural` probe can be considered the most complex, as it has the most classes. Thus, the large room for improvement is expected.

We could show for the first time that long-document Transformers can learn to represent document structure, even though the models were not explicitly trained for this. However, the representation of some aspects of structure is far from optimal. In the following, we investigate whether structure infusion, i.e. the input of additional, explicit infor-

[6] `https://F1000research.com`, downloaded on April 9th, 2021. We use the paper first versions.

[7] For the node type probe we remove the document title and abstract as well, as these occur once per document.

mation on document structure, improves the internal representation of structure and if this translates to improvements on downstream tasks.

5 Infusing Structure

As exemplified in Fig. 1, abstract document structure can help humans in working with documents. While previous work shows that the addition of structural information can improve the downstream performance of Transformer models (Li et al., 2023; Cao and Wang, 2022; Ruan et al., 2022), the use of task-specific architectures and document formats prevents comparison of structure infusion methods across the studies, and makes it challenging to relate performance to probing results. To remedy this, we introduce a task- and format-agnostic structure infusion kit, and demonstrate its wide applicability by studying the effects of structure infusion on LED and LongT5 and two challenging long-document tasks.

5.1 Methodology [8]

Structure infusion. We infuse information on abstract document structure through position embeddings added to the token embeddings (indicated as `emb`, see Fig. 4) and special tokens that are prepended to the tokens of the corresponding node (`tok`). Both methods only modify the input layer and are therefore easily applicable to any Transformer model.

We infuse the two types of abstract structural information that are missing in the input of Transformer models (§3): node function and hierarchy. Node function is infused through embeddings and special tokens representing the node type (`type`). To infuse the hierarchical organization, tokens and embeddings represent the depth of a node in the graph, i.e. its distance to the document root (`depth`). As a baseline for structural tokens, we prepend each node with the same separator token (`sep`). We refer to the infusion configurations using short descriptors, e.g. the combination of node depth position embeddings and node type tokens is shortened to `emb-depth-tok-type`.

Probing. The probing experiments were conducted as described in §4 using the same probing dataset, with the addition of structural information in the input. We omit the atomic and random control here, as we are interested in the capabilities of the configuration that is used for downstream tasks.

Downstream task datasets. We selected QASPER (Dasigi et al., 2021) and Evidence Inference (DeYoung et al., 2020) by the following criteria: they are based on long documents, abstract document structure is available, and several types of downstream tasks are covered, to see possible differences in the effect of structure infusion.

QASPER is a collection of scientific papers from computational linguistics / NLP and corresponding questions with one or multiple answers with evidence. We model question answering as a generative problem and evidence selection as paragraph classification. Answer generation and evidence selection are evaluated with F1 scores using the evaluation script provided by the authors[9].

Evidence Inference consists of reports from clinical studies, "prompts" in the form of *intervention*, *comparator*, and *outcome*, one or multiple labels for the prompt ("significantly increased", "significantly decreased", or "no significant difference") and corresponding evidence spans. We model prompt answering as 3-way classification, and convert evidence span selection to node classification by mapping evidence spans to nodes. As there is no adaptable evaluation script, and for consistency with QASPER, we re-implemented evaluation, choosing the annotation resulting in the highest score as gold standard. This means that we can only compare the models in our work.

Training Downstream tasks were fine-tuned for 10,200 steps with an effective batch size of 8 in a multi task fashion. We report mean test set results of 3 random seeds.

In all experiments in this section, the models were pre-trained for 15,000 steps, with an effective batch size of 16, with the respective structure infusion configuration on the relevant probing (F1000RD) or downstream task dataset (QASPER or Evidence Inference), as we noted this to be beneficial in early experiments (Gururangan et al., 2020). "T5-style" denoising (Raffel et al., 2020) was used as the pre-training task as suggested in Xiong et al, (2022).

5.2 Probing of Structure-Infused Models

We see an improvement in all probes through structure infusion (Fig. 5, Tab. 4). The `node type` and `tree depth` probes show an accuracy of around 1 with tree depth infusion, as this information suffices to solve the tasks. Node type infusion

[8] We provide implementation details in Appx. B.3-B.6.

[9] https://github.com/allenai/qasper-led-baseline

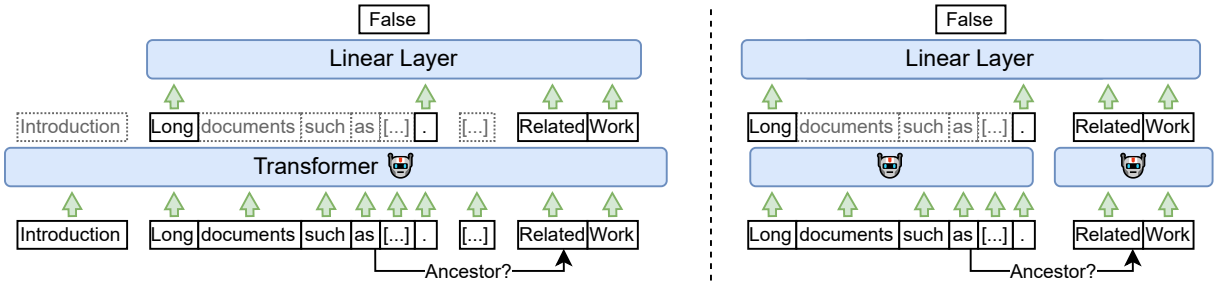

Figure 3: Probing classifier with the vanilla probing architecture encoding a full document (left) and the atomic encoding two nodes individually without any context (right). Tokens w/ arrow are used as input to the next layer.

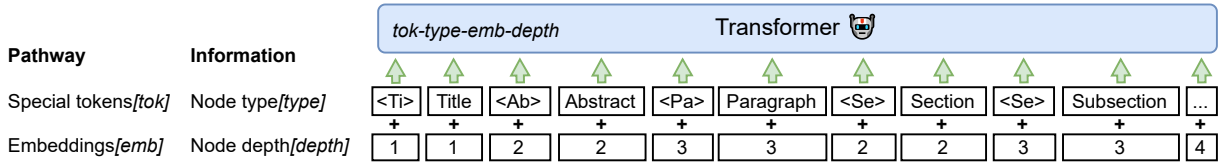

Figure 4: Structure infusion via special tokens and embeddings. Special tokens ("<Ti>", "<Ab>") are prepended to the text of the corresponding node, embeddings are summed with the token embeddings. The figure shows the combination of hierarchical embeddings and node type special tokens, short description `tok-type-emb-depth`.

	Nod	Sib	Anc	Pos	Par	Tre	Str
LED	**93.98**	**64.93**	**89.53**	**86.05**	**85.68**	**84.12**	**41.49**
LED Atom	92.75	60.26	87.30	65.53	84.82	82.41	40.64
LED Rand	88.21	58.36	86.73	56.44	82.90	73.76	35.33
LongT5	**95.28**	**65.85**	**89.38**	**91.95**	**86.13**	**87.88**	**42.97**
LongT5 Atom	91.84	50.79	86.60	61.05	83.77	78.90	34.68
LongT5 Rand	88.21	57.41	84.81	57.97	81.54	73.40	33.49

Table 2: Probing accuracy of LED and LongT5 with atomic and random controls. Best result per model and probe in bold, second best underlined.

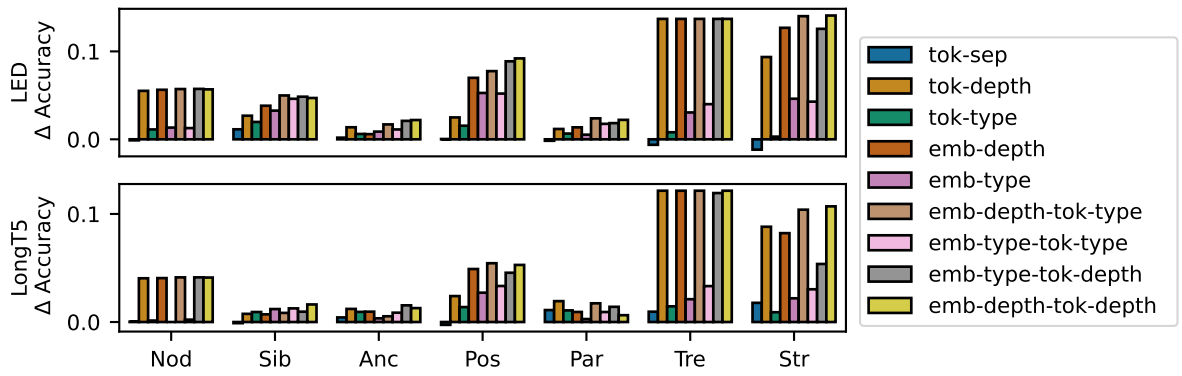

Figure 5: Probing of structure-infused models. Bars show the difference in accuracy to the vanilla baseline (Tab. 2) For absolute values see Tab. 4.

	LED				LongT5			
	QAS		EvI		QAS		EvI	
	Ans	Evi	Cla	Evi	Ans	Evi	Cla	Evi
vanilla	36.80	42.05	74.30	61.55	45.89	52.09	**81.54**	70.39
tok-sep	37.35	42.54	75.17	66.81	45.54	54.12	81.08	75.92
tok-depth	36.24	41.90	74.60	64.19	46.60	**56.14**	80.90	76.88
tok-type	37.43	42.32	75.85	66.93	**46.76**	56.08	80.75	76.28
emb-depth	36.17	42.53	73.78	60.67	44.91	51.53	81.36	71.18
emb-type	36.03	42.92	74.71	61.05	46.37	53.89	80.86	68.91
emb-depth-tok-type	37.83	43.16	**76.49**	66.07	45.63	56.04	79.94	75.57
emb-type-tok-type	38.02	43.83	76.38	65.31	46.43	55.70	81.42	**77.23**
emb-type-tok-depth	**39.08**	44.41	75.30	64.58	44.72	55.60	80.71	75.86
emb-depth-tok-depth	37.74	**44.64**	76.34	**67.07**	45.33	54.27	80.98	75.96

Table 3: Downstream task results on test sets. All scores are F1 scores averaged over 3 runs with different random seeds. Best result in column in bold, second best underlined. QAS: QASPER. EvI: Evidence Inference. Ans: Answer F1. Evi: Evidence F1. Cla: Classification F1.

does not lead to perfect scores on the `node type` probe, as the subsection node type is part of the probing task, but not of the infusion (Tab. 1).

Except for LongT5 on `sibling`, infusion of node depth results in higher accuracy than node type or node boundary information infused on the same pathway. For the majority of LED probes (`sibling`, `position`, `tree depth`, and `structural`), models with position embedding infusion show higher metrics than their counterparts with the same information in special tokens, while for LongT5, the results are mixed. LED, based on BART (Lewis et al., 2020), is pre-trained with absolute position embeddings like our structural embeddings, while LongT5, based on T5 (Raffel et al., 2020), uses relative position embeddings. LED might therefore have a better capability to use the information from absolute embeddings.

5.3 Structure infusion in Downstream Tasks

QASPER For LED in answer generation, the `emb-type-tok-depth` configuration results in the best performance, with an improvement of 2.28 F1 points over vanilla (Tab. 3). In evidence selection, `emb-depth-tok-depth` outperforms the vanilla configuration by 2.59 F1 points. This is an improvement of 5.58 F1 points for answer generation and 14.04 F1 points for evidence selection over the LED state-of-the-art (SOTA) (Caciularu et al., 2022) on QASPER. The vanilla configuration already outperforms the SOTA by 3.30 and 11.45 F1 points, respectively. Infusing the node depth through two pathways improves over a single pathway. While unintuitive, this was also observed for the `sibling`, `parent predecessor`, and `tree depth` probes (Fig. 5).

For LongT5, special tokens structure infusion results in the highest scores. The best answer F1 of 46.76 with node type tokens improves the vanilla model by 0.87 points and is slightly higher than the current LongT5-base SOTA of 46.6 (Guo et al., 2022). In evidence selection, infusion of depth tokens increases the vanilla configuration by 4.05 F1 points. To our knowledge, there are no reported scores for LongT5 on QASPER evidence selection.

Evidence Inference For LED, the best performance in classification is obtained by the `emb-depth-tok-type` configuration, improving 2.19 F1 points over the vanilla configuration. In evidence selection, `emb-depth-tok-depth` outperforms the vanilla baseline by 5.52 F1 points, but adding node separator tokens already leads to an increase of 5.26 F1 points.

For LongT5, no structure infused variant outperforms vanilla in classification, while in evidence selection, `emb-type-tok-type` outperforms vanilla by 6.84 F1 points.

Comparison of infusion configurations. In most cases, adding node separator tokens improves performance. This was expected, as it is common practice to signify segment boundaries to models (e.g. Beltagy et al. 2020) and could also be seen

in probing. For LED, the combination of position embeddings and structural tokens exhibits the best scores, which again resembles the probing results. For LongT5, combining both infusion pathways only results in the best scores on Evidence Inference evidence selection. Infusion via structural tokens outperforms infusion via position embeddings for LongT5 on most subtasks.

The increases for LED of about 2 F1 points are similar to the reported performance increases through document structure infusion on other long-document datasets, showing that our employed methods are effective. These works use relative position embeddings (Cao and Wang, 2022) or special attention patterns (Liu et al., 2021; Hong et al., 2022), while we use structural tokens and absolute position embeddings. Our methods are easier to apply and adapt, as only the input to the model needs to be modified. For LongT5, the performance gains through structure infusion of up to 6.84 F1 points suggest that this is a promising research direction.

5.4 Correlation between Probing and Downstream Tasks

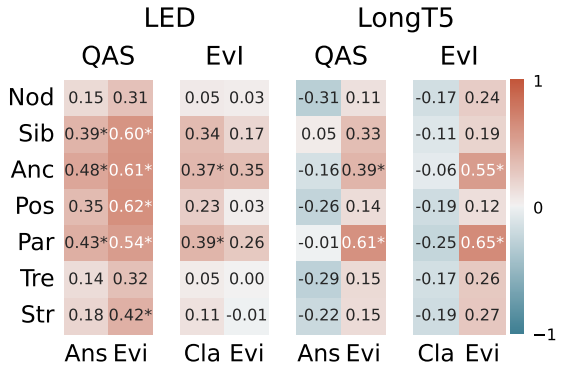

Figure 6: Pearson correlation between probing and downstream tasks. * denotes significance ($p < 0.05$).

To find associations between the representation of document structure and downstream task performance, we computed the Pearson correlation between probing and downstream task metrics [10] (Fig. 6). All combinations of probing and downstream tasks for LED, and evidence selection and all probing tasks for LongT5 have a correlation greater or around 0. In contrast, the performance of LongT5 on QASPER answer generation and Evidence Inference classification is mostly negatively correlated with the probing task metrics. These were also the tasks with the least improvements through structure infusion. As they are decoder-based tasks, while evidence selection is encoder-based (§B.5), it seems that LongT5 has less need for structure infusion on decoder-based tasks.

For LED in both QASPER subtasks and Evidence Inference classification and for LongT5 in evidence selection on both Evidence Inference and QASPER, we see significant ($p < 0.05$) correlation with the `ancestor` and `parent predecessor` probes, which measure the representation of relations between nodes on one directed path of `parent` edges. These usually have more defined semantic relationships among each other compared to nodes from different paths, e.g. a section heading has more relevant information about the paragraphs belonging to that section than about those in other sections. Our results suggest that better representation of these relations is associated with better downstream performance.

6 Conclusion

In this work, we provided an in-depth analysis of the representation of abstract document structure in long-document Transformers. Experiments with our novel probing suite show that LED and LongT5 have learned to represent node function and hierarchical organization through pre-training without explicit supervision, with room for improvement.

To investigate the effect of infusing the aspects of document structure that are missing in Transformer inputs due to linearization, we developed a modular structure infusion framework. Probing shows that structure infusion enhances the internal representation of document structure, and we see performance improvements from structure infusion on QASPER and Evidence Inference, two downstream tasks where this has not been shown before. The significant correlation between several probing and downstream tasks suggests that it is indeed the improved representation of document structure that leads to downstream task performance gains.

Our probing, structure infusion and downstream task suite is easily extensible with new probing and downstream tasks and new types of infused information. While this work provides proof of the utility of our graph-based framework for documents from the scientific domain, the framework can be applied to other document types (e.g. web pages or conversation threads). Given that the ad-

[10] The absolute values from each set of bars in Fig. 5 were paired with the unaggregated values from each column in Tab. 3 for the same model.

dition of separator tokens between document elements can already increase performance, we deem applying our methods to documents with less well-defined structure promising. Our probing methods are fully compatible with the current generation of Transformer-based LLMs (BigScience Workshop, 2023; Touvron et al., 2023), as long as the internal states of the model can be accessed. We hope that our contributions pave the path towards systematic study of the role of document structure in NLP.

Ethical Considerations

Long documents lie at the core of text work, and structure is omnipresent in long documents. We believe that developing a better understanding of the role of document structure in NLP would allow us to build more efficient, robust, and interpretable systems for the analysis of long texts. We envision a trade-off between structural modeling capabilities of NLP systems (which, as we show, can be enhanced by providing explicit document structure) and the computational and storage overhead associated with processing additional structural information in the documents. Future work would investigate this trade-off and determine in which cases this overhead is justified. As document structure is openly present in documents and easily accessible by humans, we do not envision additional ethical risks or misuse scenarios due to the use of document structure in NLP modeling. Our work only uses data published under permissive licenses; our adaptations of this data are made available under permissive conditions as well.

Limitations

We see our work as an important step towards the general study of the role of document structure in NLP modeling. Below we outline the limitations of our work, which present excellent opportunities for follow-up research.

Dataset diversity. Our work unifies structured document data from multiple sources. Yet all of this data originates form the scientific domain. There are several benefits to this: scientific documents are long, clearly licensed, and exhibit structure – and the scientific domain offers multiple long-document processing tasks. In addition, focusing on one general domain allows us to control for domain shift during our measurements. We note that no part of our methodology is tailored to the particularities of the scientific domain – and as long as source documents can be converted into the domain-agnostic ITG formalism, our methods should be easily adaptable to other domains like Wikipedia or conversation threads. Similarly, we limit our studies to the English language, as other languages face scarcity both in terms of available long-document Transformer models and academic texts. As more data and models become available, it will become possible to evaluate our findings in new contexts.

Models and Tasks. Our setup involves multiple probing tasks coupled with a range of structure infusion methods, resulting in a wide experimental grid. To make in-depth analysis feasible, we had to limit our focus on a few models and tasks. We chose two datasets which combine generative question answering, segment classification and document classification. Our experiments show that structure infusion can be useful for all tasks and models considered. This suggests that experiments on other tasks are a promising direction for future research, which is facilitated by our open implementation.

Large language models. While it would be technically possible to apply our kit to the recent decoder-only models such as LLaMA (Touvron et al., 2023) or BLOOM (Fan et al., 2022), this would require substantial computational resources – which illustrates the challenges of long-document processing by modern NLP models and does not constitute a limitation of our proposed approach. Similarly, commercially hosted models with increased input length such as GPT-4[11] (32k tokens) and Claude [12] (100k tokens) could be evaluated and infused with document structure – yet their closed-source nature and lack of access to model weights prevents such investigation. We hope that the progress in efficient NLP and the ongoing open-source LLM development make such studies possible in the near future. This would also pave the way for investigating the effect of abstract document structure in zero-shot experiments.

Other types of document structure As noted in the "Document structure" paragraph in §2, we focus on investigating abstract document structure. We also mention visual and discourse structure as important structural properties of documents. While we don't study them here, this is done in current works, e.g. Huang et al. (2022) or Du et al. (2023). We believe that joint investigations of

[11] https://openai.com/gpt-4
[12] https://www.anthropic.com/product

the different aspects of document structure are a promising direction for future work.

Correlated model states. The structure-infused models in this work were first pre-trained using a language modeling loss on probing or downstream task data, and then further fine-tuned using a task-specific loss. The probing and downstream task datasets in our work are *not identical*; thus, strictly speaking, the scores used to compute the correlation in Fig. 6 come from models with the same structure infusion configuration, but not the same *state*. We believe this to be unproblematic and expect the states to be comparable, since each model is pre-trained under the same regime. To confirm this, future work could create probing datasets from downstream task datasets to use the same model state in probing and downstream tasks – at the cost of a drastic increase in the number of probing experiments. This technical limitation only pertains to §5.4 and Fig. 6 and leaves all other results unaffected.

Acknowledgements

Funded by the European Union (ERC, InterText, 101054961). Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union or the European Research Council. Neither the European Union nor the granting authority can be held responsible for them.

We thank the anonymous reviewers for their helpful comments and suggestions.

References

Armen Aghajanyan, Dmytro Okhonko, Mike Lewis, Mandar Joshi, Hu Xu, Gargi Ghosh, and Luke Zettlemoyer. 2022. HTLM: Hyper-text pre-training and prompting of language models. In *International Conference on Learning Representations*.

Joshua Ainslie, Santiago Ontanon, Chris Alberti, Vaclav Cvicek, Zachary Fisher, Philip Pham, Anirudh Ravula, Sumit Sanghai, Qifan Wang, and Li Yang. 2020. ETC: Encoding long and structured inputs in transformers. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 268–284, Online. Association for Computational Linguistics.

He Bai, Peng Shi, Jimmy Lin, Yuqing Xie, Luchen Tan, Kun Xiong, Wen Gao, and Ming Li. 2021. Segatron: Segment-aware transformer for language modeling and understanding. *Proceedings of the AAAI Conference on Artificial Intelligence*, 35(14):12526–12534.

Yonatan Belinkov. 2022. Probing classifiers: Promises, shortcomings, and advances. *Computational Linguistics*, 48(1):207–219.

Iz Beltagy, Matthew E. Peters, and Arman Cohan. 2020. Longformer: The long-document transformer. *arXiv:2004:05150*.

BigScience Workshop. 2023. Bloom: A 176b-parameter open-access multilingual language model. *arXiv:2211.05100*.

Avi Caciularu, Ido Dagan, Jacob Goldberger, and Arman Cohan. 2022. Long context question answering via supervised contrastive learning. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 2872–2879, Seattle, United States. Association for Computational Linguistics.

Shuyang Cao and Lu Wang. 2022. HIBRIDS: Attention with hierarchical biases for structure-aware long document summarization. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 786–807, Dublin, Ireland. Association for Computational Linguistics.

Simone Conia and Roberto Navigli. 2022. Probing for predicate argument structures in pretrained language models. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 4622–4632, Dublin, Ireland. Association for Computational Linguistics.

Alexis Conneau, German Kruszewski, Guillaume Lample, Loïc Barrault, and Marco Baroni. 2018. What you can cram into a single $&!#* vector: Probing sentence embeddings for linguistic properties. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2126–2136, Melbourne, Australia. Association for Computational Linguistics.

Pradeep Dasigi, Kyle Lo, Iz Beltagy, Arman Cohan, Noah A. Smith, and Matt Gardner. 2021. A dataset of information-seeking questions and answers anchored in research papers. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 4599–4610, Online. Association for Computational Linguistics.

Jay DeYoung, Eric Lehman, Benjamin Nye, Iain Marshall, and Byron C. Wallace. 2020. Evidence inference 2.0: More data, better models. In *Proceedings of the 19th SIGBioMed Workshop on Biomedical Language Processing*, pages 123–132, Online. Association for Computational Linguistics.

Haowei Du, Yansong Feng, Chen Li, Yang Li, Yunshi Lan, and Dongyan Zhao. 2023. Structure-discourse hierarchical graph for conditional question answering on long documents. In *Findings of the Association*

for Computational Linguistics: ACL 2023, pages 6282–6293, Toronto, Canada. Association for Computational Linguistics.

Angela Fan, Suzana Ilic, Thomas Wolf, and Matthias Gallé, editors. 2022. *Proceedings of BigScience Episode #5 – Workshop on Challenges & Perspectives in Creating Large Language Models*. Association for Computational Linguistics, virtual+Dublin.

Adam Fisch, Alon Talmor, Robin Jia, Minjoon Seo, Eunsol Choi, and Danqi Chen. 2019. MRQA 2019 shared task: Evaluating generalization in reading comprehension. In *Proceedings of the 2nd Workshop on Machine Reading for Question Answering*, pages 1–13, Hong Kong, China. Association for Computational Linguistics.

Matt Gardner, Joel Grus, Mark Neumann, Oyvind Tafjord, Pradeep Dasigi, Nelson F. Liu, Matthew Peters, Michael Schmitz, and Luke Zettlemoyer. 2018. AllenNLP: A deep semantic natural language processing platform. In *Proceedings of Workshop for NLP Open Source Software (NLP-OSS)*, pages 1–6, Melbourne, Australia. Association for Computational Linguistics.

Mandy Guo, Joshua Ainslie, David Uthus, Santiago Ontanon, Jianmo Ni, Yun-Hsuan Sung, and Yinfei Yang. 2022. LongT5: Efficient text-to-text transformer for long sequences. In *Findings of the Association for Computational Linguistics: NAACL 2022*, pages 724–736, Seattle, United States. Association for Computational Linguistics.

Suchin Gururangan, Ana Marasović, Swabha Swayamdipta, Kyle Lo, Iz Beltagy, Doug Downey, and Noah A. Smith. 2020. Don't stop pretraining: Adapt language models to domains and tasks. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 8342–8360, Online. Association for Computational Linguistics.

John T. Guthrie, Tracy Britten, and K. Georgene Barker. 1991. Roles of Document Structure, Cognitive Strategy, and Awareness in Searching for Information. *Reading Research Quarterly*, 26(3):300.

John Hewitt and Percy Liang. 2019. Designing and interpreting probes with control tasks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 2733–2743, Hong Kong, China. Association for Computational Linguistics.

John Hewitt and Christopher D. Manning. 2019. A structural probe for finding syntax in word representations. 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)*, pages 4129–4138, Minneapolis, Minnesota. Association for Computational Linguistics.

Giwon Hong, Jeonghwan Kim, Junmo Kang, and Sung-Hyon Myaeng. 2022. Graph-induced transformers for efficient multi-hop question answering. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 10288–10294, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Yupan Huang, Tengchao Lv, Lei Cui, Yutong Lu, and Furu Wei. 2022. Layoutlmv3: Pre-training for document ai with unified text and image masking. In *Proceedings of the 30th ACM International Conference on Multimedia*, MM '22, page 4083–4091, New York, NY, USA. Association for Computing Machinery.

Maor Ivgi, Uri Shaham, and Jonathan Berant. 2023. Efficient Long-Text Understanding with Short-Text Models. *Transactions of the Association for Computational Linguistics*, 11:284–299.

Ganesh Jawahar, Benoît Sagot, and Djamé Seddah. 2019. What does BERT learn about the structure of language? In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 3651–3657, Florence, Italy. Association for Computational Linguistics.

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *3rd International Conference on Learning Representations, ICLR, Conference Track Proceedings*.

W. Kintsch and T.A. van Dijk. 1978. Toward a model of text comprehension and production. *Psychological Review*, 5(85):363–394.

Ilia Kuznetsov, Jan Buchmann, Max Eichler, and Iryna Gurevych. 2022. Revise and Resubmit: An Intertextual Model of Text-based Collaboration in Peer Review. *Computational Linguistics*, 48(4):949–986.

Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer. 2020. BART: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7871–7880, Online. Association for Computational Linguistics.

Miao Li, Eduard Hovy, and Jey Lau. 2023. Summarizing multiple documents with conversational structure for meta-review generation. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 7089–7112, Singapore. Association for Computational Linguistics.

Yang Liu and Mirella Lapata. 2019. Hierarchical transformers for multi-document summarization. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 5070–5081, Florence, Italy. Association for Computational Linguistics.

Ye Liu, Jianguo Zhang, Yao Wan, Congying Xia, Lifang He, and Philip Yu. 2021. HETFORMER: Heterogeneous transformer with sparse attention for long-text extractive summarization. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 146–154, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Ilya Loshchilov and Frank Hutter. 2019. Decoupled weight decay regularization. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*.

William C Mann and Sandra A Thompson. 1987. *Rhetorical structure theory: A theory of text organization*. University of Southern California, Information Sciences Institute Los Angeles.

Bonnie J. F. Meyer, David M. Brandt, and George J. Bluth. 1980. Use of Top-Level Structure in Text: Key for Reading Comprehension of Ninth-Grade Students. *Reading Research Quarterly*, 16(1):72–103.

Laura Nguyen, Thomas Scialom, Jacopo Staiano, and Benjamin Piwowarski. 2021. Skim-attention: Learning to focus via document layout. In *Findings of the Association for Computational Linguistics: EMNLP 2021*, pages 2413–2427, Punta Cana, Dominican Republic. Association for Computational Linguistics.

Geoffrey Nunberg. 1990. *The linguistics of punctuation*. Number 18 in Lecture Notes. Center for the Study of Language (CSLI).

Richard Power, Donia Scott, and Nadjet Bouayad-Agha. 2003. Document Structure. *Computational Linguistics*, 29(2):211–260.

Siya Qi, Lei Li, Yiyang Li, Jin Jiang, Dingxin Hu, Yuze Li, Yingqi Zhu, Yanquan Zhou, Marina Litvak, and Natalia Vanetik. 2022. SAPGraph: Structure-aware extractive summarization for scientific papers with heterogeneous graph. In *Proceedings of the 2nd Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics and the 12th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 575–586, Online only. Association for Computational Linguistics.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *J. Mach. Learn. Res.*, 21(1).

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. SQuAD: 100,000+ questions for machine comprehension of text. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2383–2392, Austin, Texas. Association for Computational Linguistics.

Anna Rogers, Olga Kovaleva, and Anna Rumshisky. 2020. A primer in BERTology: What we know about how BERT works. *Transactions of the Association for Computational Linguistics*, 8:842–866.

Qian Ruan, Malte Ostendorff, and Georg Rehm. 2022. HiStruct+: Improving extractive text summarization with hierarchical structure information. In *Findings of the Association for Computational Linguistics: ACL 2022*, pages 1292–1308, Dublin, Ireland. Association for Computational Linguistics.

Uri Shaham, Elad Segal, Maor Ivgi, Avia Efrat, Ori Yoran, Adi Haviv, Ankit Gupta, Wenhan Xiong, Mor Geva, Jonathan Berant, and Omer Levy. 2022. SCROLLS: Standardized CompaRison over long language sequences. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 12007–12021, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Yi Tay, Mostafa Dehghani, Dara Bahri, and Donald Metzler. 2022. Efficient transformers: A survey. *ACM Computing Surveys*, 55(6):1–28.

Barbara M. Taylor and Richard W. Beach. 1984. The Effects of Text Structure Instruction on Middle-Grade Students' Comprehension and Production of Expository Text. *Reading Research Quarterly*, 19(2):134–146. Publisher: Wiley, International Reading Association.

Ian Tenney, Dipanjan Das, and Ellie Pavlick. 2019. BERT rediscovers the classical NLP pipeline. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4593–4601, Florence, Italy. Association for Computational Linguistics.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, Aurelien Rodriguez, Armand Joulin, Edouard Grave, and Guillaume Lample. 2023. Llama: Open and efficient foundation language models. *arXiv:2302.13971*.

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

Elena Voita and Ivan Titov. 2020. Information-theoretic probing with minimum description length. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 183–196, Online. Association for Computational Linguistics.

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.

Wenhan Xiong, Anchit Gupta, Shubham Toshniwal, Yashar Mehdad, and Wen-tau Yih. 2022. Adapting pretrained text-to-text models for long text sequences. *arXiv:2209:10052*.

Haopeng Zhang, Xiao Liu, and Jiawei Zhang. 2022. HEGEL: Hypergraph transformer for long document summarization. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 10167–10176, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

A Table of Probing Results

See Tab. 4

B Implementation Details

B.1 Models

In all experiments, we used the huggingface Transformers[13] (Wolf et al., 2020) implementations and weights of LED base (162M parameters, Beltagy et al. 2020) and LongT5 base with transient global attention (220M parameters, Guo et al. 2022).

B.2 Probing

Dataset. Our probing dataset is split 0.6/0.2/0.2 across train, dev, and test using in-document balancing. For boolean and the `position` probe we see a uniform distribution of instances per label, compared to the `node type` probe where subsections occur not in all documents, resulting in a non-uniform distribution. The `structural` and `tree depth` probes naturally feature a diverse set of labels and instances. A full overview of the label distribution can be found in Tab. 5.

Implementation and hyperparamenters. Our probing kit is implemented using the AllenNLP library (Gardner et al., 2018). We stack a frozen pre-trained Transformer model with an endpoint span extractor from AllenNLP, extracting and concatenating the first and last token of a given span. Our hyperparameters are described in Tab. 6.

[13] `https://huggingface.co/`

Layer utilization. The layer utilization shown in Fig. 7 reveals differences between the probed models and their controls. For LED, the vanilla configuration shows a more uniform layer utilization compared to the control configurations. The atomic control puts more weight on the last layer for all probes except `node type` and `tree depth`. For LongT5, both vanilla and atomic put all weight on the last layer. For LED and LongT5, the random control mostly uses the first layer, which has also been observed in other works (Voita and Titov, 2020). The random control relies solely on the input embeddings, as there is no additional information in the Transformer layers. Input words such as "Introduction" and the number of tokens in a text node can be used to infer the node type. Node type and word overlaps between two nodes can give hints to the relation between two nodes. With LongT5, the intermediate layers are not used at all.

As the atomic control cannot compare the position embeddings of different nodes, it makes full use of the contextualization through the entire forward pass. To solve the `node type` task, the length of a node provides useful information. It is retained in the atomic position embeddings, explaining the more uniform layer utilization on this probe. The random control puts most weight on the the first layer, which has also been observed in other works (Voita and Titov, 2020). It relies on the input embeddings, as there is no additional information in the Transformer layers.

B.3 Structure Infusion

Embeddings. Structural embeddings are added to the token embeddings of each token in a node (including special tokens) before the first encoder self-attention layer (Fig. 4). They were initialized according to a Gaussian distribution with mean 0 and standard deviation 0.0305 (LED) and 4.875 (LongT5). Standard deviation for LED was chosen to be the same as the standard deviation of the absolute linear position embeddings matrix. As LongT5 does not have absolute position embeddings, the standard deviation for structural embedding initialization was chosen to result in the same ratio of token embedding standard deviation to structural embedding standard deviation as for LED.

Special tokens. Special tokens are prepended to the tokens of the respective node, leading to an increase in total sequence length (Fig. 4). They were initialized using the

	Nod	Sib	Anc	Pos	Par	Tre	Str
LED	93.98	64.93	89.53	86.05	85.68	84.12	41.49
LED Atom	92.75	60.26	87.30	65.53	84.82	82.41	40.64
LED Rand	88.21	58.36	86.73	56.44	82.90	73.76	35.33
tok-boundaries	94.15	65.87	89.80	87.13	86.30	85.64	40.68
tok-depth	99.78	67.41	90.99	89.59	87.64	**99.96**	51.22
tok-type	95.39	66.70	90.23	88.64	87.12	87.06	42.16
emb-depth	99.90	68.55	90.21	94.09	87.83	**99.96**	54.54
emb-type	95.60	67.99	90.49	92.37	86.99	89.32	46.48
emb-depth-tok-type	99.98	**69.71**	91.31	94.85	**88.85**	**99.96**	55.87
emb-type-tok-type	95.54	69.34	90.74	92.30	88.23	90.26	46.14
emb-type-tok-depth	**100.00**	69.57	91.72	95.97	88.31	**99.96**	54.43
emb-depth-tok-depth	99.95	69.43	**91.81**	**96.30**	88.68	**99.96**	**55.94**
LongT5	95.28	65.85	89.38	91.95	86.13	87.88	42.97
LongT5 Atom	91.84	50.79	86.60	61.05	83.77	78.90	34.68
LongT5 Rand	88.21	57.41	84.81	57.97	81.54	73.40	33.49
tok-sep	95.88	66.93	90.41	93.16	87.62	88.76	45.47
tok-depth	99.90	67.79	91.20	95.82	**88.45**	**99.96**	52.51
tok-type	95.99	67.96	90.92	94.80	87.59	89.26	44.60
emb-depth	99.92	67.75	90.94	98.32	87.45	**99.96**	51.92
emb-type	95.85	68.23	90.33	96.13	86.79	89.92	45.89
emb-depth-tok-type	**99.98**	67.88	90.52	**98.86**	88.25	**99.96**	54.09
emb-type-tok-type	96.07	68.30	90.85	96.75	87.44	91.13	46.73
emb-type-tok-depth	**99.98**	67.99	**91.53**	97.98	87.92	99.74	49.07
emb-depth-tok-depth	99.97	**68.66**	91.27	98.70	87.15	**99.96**	**54.40**

Table 4: Probing result numbers for Fig. 5 and from Tab. 2 for comparison. The best result per model is printed in bold, the second best is underlined.

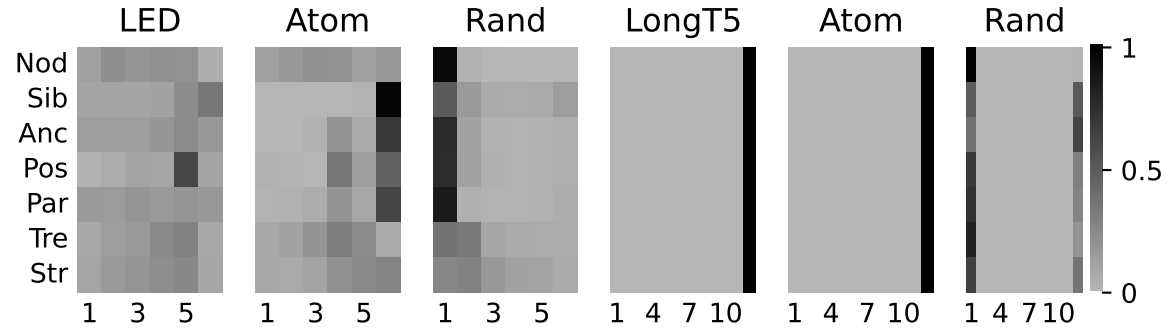

Figure 7: Layer utilization in probing of the vanilla LED and LongT5 models.

	Label	Dev	Test	Train
Anc	False	7665	7999	23488
	True	7665	7999	23488
	Total	15330	15998	46976
Nod	Paragraph	2353	2369	7046
	Section	2278	2298	6708
	Subsection	1250	1262	3611
	Total	5881	5929	17365
Par	False	7665	7999	23488
	True	7665	7999	23488
	Total	15330	15998	46976
Pos	Begin	3049	3180	9406
	End	3049	3180	9406
	Inside	3049	3180	9406
	Total	9147	9540	28218
Sib	False	7665	7999	23488
	True	7665	7999	23488
	Total	15330	15998	46976
Str	1	2939	3044	8946
	2	2939	3044	8946
	3	2939	3044	8946
	4	2912	3018	8823
	5	1840	1926	5560
	6	985	1124	3161
	7	-	10	5
	8	-	-	5
	Total	14554	15210	44392
Tre	1	2892	2895	8642
	2	2892	2895	8642
	3	1634	1639	4872
	4	-	3	1
	5	-	-	1
	Total	7418	7432	22158

Table 5: Label distribution across probing tasks. Anc: Ancestor; Nod: Node type; Par: Parent predecessor; Pos: Position; Sib: Sibling; Str: Structural; Tre: Tree depth.

Training	
Batch size	4 (VR), 64 (AT)
Epochs	20
Patience	10
Optimization	
Algorithm	Adam (Kingma and Ba, 2015)
β_1, β_2	0.9, 0.999
ϵ	10^{-8}
Weight decay	0.01
Learning rate	10^{-3}(LED), 10^{-1}(LongT5)

Table 6: Vanilla and random (VR), and atomic (AT) configuration hyperparameters.

Config	$n_{parameters}$
tok-type	3K
emb-type	3K
tok-depth	15K
emb-depth	15K

Table 7: Number of added parameters in structure infusion

resize_token_embeddings() function in the model implementation.

Number of added parameters. For the number of added parameters for each infusion configuration see Tab. 8. Each special token and each embedding adds d_{model} parameters to a model ($d_{LED} = d_{LongT5} = 768$). There were 4 structural tokens / embeddings and 20 node depth tokens / embeddings.

B.4 Pre-Training

All structure infused models and baselines were pre-trained on the respective probing or evaluation dataset using a "T5-style" denoising task. Noise was added to the model input using code provided by the authors of the T5 (Raffel et al., 2020) paper[14], which replaces spans of tokens in the input with numbered mask tokens. The mask tokens were initialized using the resize_token_embeddings() function in the model implementation. Masking is controlled by two hyperparameters: *noise density*, the proportion of masked tokens in the input, and *mean noise span length*. We chose the noise density as 3%, the mean noise span length was uniformly chosen for each input sequence from 4, 8 or 12 tokens.

[14] https://github.com/google-research/text-to-text-transfer-transformer

Masking	
Noise density	3%
Mean noise span length	[4,8,12]*
Training	
Batch size	16 (PT), 8 (FT)
Steps	15000 (PT)
	10200 (FT)
Optimization	
Algorithm	AdamW [1]
β_1, β_2	0.9, 0.999
ϵ	10^{-8}
Weight decay	0.01
Learning rate	10^{-5} (LED)
	10^{-4} (LongT5)
Warmup	Linear (PT), - (FT)
Warmup steps	500 (PT), - (FT)

Table 8: Pre-training (PT) and fine-tuning (FT) hyperparameters. *: Mean noise span length is chosen uniformly from the given values for each input sequence. [1] Loshchilov and Hutter 2019

The model is trained with a cross entropy loss to generate each mask token followed by the tokens replaced by that mask, respecting the order of masked spans. To save computation, only one checkpoint was pre-trained for each combination of model, infusion configuration and dataset. This checkpoint was used in all replicates of a downstream experiment.

Training hyperparameters For training hyperparameters, see Tab. 8.

The only optimized hyperparameter is the learning rate, which was done by grid search with the respective non-pretrained vanilla configuration on the QASPER dataset.

B.5 Downstream Tasks

B.5.1 QASPER

Dataset conversion. Each entry in the QASPER dataset (Dasigi et al., 2021) consists of a paper title, abstract, full text in the form of a list of sections with section name and corresponding paragraphs, a list of figures and tables, as well as a list of questions, answers and evidence. We converted the QASPER dataset into the Intertext Graph (ITG) format (Kuznetsov et al., 2022) creating a node for the title, abstract, each section title and each paragraph, as well as figures and tables. We added an additional `abstract` node with the content "Abstract" to serve as the parent for the abstract text.

All answer types (extractive, abstractive, yes/no, unanswerable) were mapped to a single reference answer string for each question as done by the dataset authors. The provided evidence strings were mapped to the ITG nodes through string matching, which which was successful for 99.35% of evidence pieces from the original dataset. For 0.41%, there was no match, and for 0.24% there were multiple matches, which were discarded. Questions, answers and evidence are stored in the ITG metadata. We follow the original data splits, resulting in 888 train, 281 validation and 416 test documents.

Model input. For LED, model input was formed as "`<s> [question] </s> [document]`". For LongT5, the initial `<s>` token was not used, as it is not pre-trained with this token. Figures and tables were discarded for model input.

Evaluation. QASPER evaluation was implemented by adapting the evaluation script provided by the creators of the dataset[15]. If there are multiple reference answers to a question, the answer that results in the highest score is chosen as the gold standard. Answer generation is evaluated with a token-level F1 score as in SQuAD (Rajpurkar et al., 2016). Evidence selection is evaluated with a node-level F1 score.

Answer generation. Answers were generated with beam search, using 4 beams, length penalty 1.0 and a maximum generated length of 100 tokens.

Evidence selection. Evidence selection was implemented as paragraph classification. There can be multiple evidence paragraphs for a question. The final encoder hidden state h of the first token of each `paragraph` node in a document is used as the representation for the paragraph. This vector is passed through a fully connected linear layer W_1 followed by a tanh nonlinearity and a linear layer W_2 projecting to the score vector $s \in \mathbb{R}^2$ for evidence and no-evidence.

$$s = W_2 \tanh(W_1 h),\ W_1 \in \mathbb{R}^{d \times d},\ W_2 \in \mathbb{R}^{d \times 2} \tag{1}$$

Fine-tuning. Models pre-trained as described above on the QASPER train documents were fine-tuned on with the hyperparameters given in Tab. 8.

[15] https://github.com/allenai/qasper-led-baseline

Answer generation and evidence selection were trained with cross entropy loss:

$$\mathcal{L} = w_A \mathcal{L}_{Answer} + w_E \mathcal{L}_{Evidence} \quad (2)$$

For LED and LongT5 the loss weights were set to $w_A = w_E = 0.5$. The checkpoint with the best score on the dev set was used for evaluation.

B.5.2 Evidence Inference

Dataset conversion. Evidence Inference 2.0 (DeYoung et al., 2020) is provided as sets of articles, prompts and labels with evidence. The article full texts are provided as plain text files and NXML files following the PubMed DTD schema[16]. We used the parser from the dataset creators[17] to parse the NXML files, and converted the output to the ITG format. We added an additional `abstract` node with the content "Abstract" to serve as the parent for the abstract text.

Evidence annotations are given as character offsets pertaining to the articles in plain text format. We transform this span selection problem to a node classification problem by mapping evidence strings to ITG nodes. Evidence text at a given offset is extracted from a text file and then matched against ITG nodes using fuzzysearch[18]. Full string matching resulted in low recall, because of small differences between the plain text files and NXML files. For 92.03% of evidence spans, we find exactly one ITG node, for 5.10% we find no node, and for 2.07% we find more than one node, which are discarded. The prompts, labels and evidence for a document are stored in the ITG metadata. We follow the original data splits, resulting in 3562 train, 443 validation and 449 test documents.

Model input. For LED, model input was formed as "`<s> With respect to [outcome], characterize the reported difference between patients receiving [intervention] and those receiving [comparator]. </s> [document]`". For LongT5, the initial `<s>` token was not used, as it is not pre-trained with this token.

[16] https://pubmed.ncbi.nlm.nih.gov/download/

[17] https://github.com/jayded/evidence-inference

[18] https://github.com/taleinat/fuzzysearch

Evaluation. Evidence Inference classification is evaluated with macro F1 score. Evidence selection is evaluated with a node-level F1 score. If there are multiple annotations to a prompt, the annotation that results in the highest score is chosen. We chose to implement the evaluation similar to QASPER evaluation for consistency, and thus different from the implementation by the creators of the dataset. The main differences are (1) the conversion of evidence selection to a node classification task and (2) choosing the classification annotation that results in the highest score, where in the original implementation the class with the highest number of annotations is chosen as the gold standard.

Classification. To get the class of a prompt-document pair, a vector representation v of the document is passed through a fully connected layer M_1, followed by a tanh nonlinearity and a linear layer M_2 projecting to the score vector $l \in \mathbb{R}$.

$$l = M_2(\tanh(M_1(v))),\ M_1 \in \mathbb{R}^{d \times d},\ M_2 \in \mathbb{R}^{d \times 3} \quad (3)$$

For LED, v was chosen as the final encoder hidden state of the initial `<s>` token, because it has global attention. As LongT5 does not have configurable global attention, a dummy `</s>` token was input to the decoder, which has full cross attention over the input document. The final decoder hidden state of this token served as v for LongT5.

Evidence selection. Evidence selection was implemented as for QASPER (§B.5.1).

Fine-tuning. Models pre-trained as described above on the Evidence Inference train documents were fine-tuned with the hyperparameters given in Tab 8. Classification and evidence selection were trained with cross entropy loss:

$$\mathcal{L} = w_C \mathcal{L}_{Classification} + w_E \mathcal{L}_{Evidence} \quad (4)$$

For LED, the loss weights were set to $w_C = w_E = 0.5$. For LongT5, they were set to $w_C = 0.25,\ w_E = 0.75$. The checkpoint with the best score on the dev set was used for evaluation.

B.6 Computation

Experiments were performed on NVIDIA A100, A180 and A6000 GPUs. Depending on the GPU size and speed, pre-training, probing (all 7 tasks) and downstream task experiments took 1-2 days. Estimating an average of 1.5 days per experiment, the total number of GPU days is 264 (26 probing

runs, 30 pre-training runs, 120 downstream fine-tuning runs).

B.7 Use of AI Assistants in Development

Some of the code for the structure infusion framework was developed with assistance from GitHub Copilot[19].

[19] https://github.com/features/copilot