IEC
Institute of Electronics and Computer

Journal of Artificial Intelligence and Systems, 2025, 7, 61–75
https://iecscience.org/journals/AIS
ISSN Online: 2642–2859

Evaluating Three-Dimensional Topological Positioning in Generative Table Recognition

João Paulo Paiva Lima[1], **Denilson Alves Pereira**[1]

[1] Department of Computer Science - Federal University of Lavras
Câmpus Universitário UFLA, 3037, Minas Gerais, Brazil
Email: joao.lima10@estudante.ufla.br, denilsonpereira@ufla.br

How to cite this paper: João Paulo Paiva Lima, Denilson Alves Pereira (2025) Evaluating Three-Dimensional Topological Positioning in Generative Table Recognition. Journal of Artificial Intelligence and Systems, 7, 61–75.
https://doi.org/10.33969/AIS.2025070104

Received: June 11, 2025
Accepted: September 25, 2025
Published: October 16, 2025

Abstract

Transformer-based large language models still have unexplored possibilities, specifically, in the use of multi-dimensional positioning during autoregressive generation. Non-sequential positioning is already a common feature in various visual document understanding and table understanding encoder models. It is often used by associating every input token with its x and y coordinates in a visual document or table. However, the same technique is left mostly unexplored when it comes to generative decoder models. In this work, we investigate whether decoder models for table generation can also be improved by incorporating a more complex type of positioning. We adapted a pretrained image-to-sequence model to incorporate three positional dimensions to each generated token in a table, with each dimension representing the token's position inside a cell, the cell's position inside a row, and the row's position inside a table. The adapted model was then trained for the task of table recognition using the PubTabNet dataset. To assess its effectiveness, we compared the trained model's performance against an identical baseline using standard positional encoding. The resulting model showed a significantly improved overall score over baseline (+1.2%), with a more pronounced advantage in complex (+2.2%) and very large tables (+16.9%).

Keywords

Table recognition, generative model, transformer, natural language processing, image processing.

1. Introduction

From its first publication in [19], the Transformer architecture has revolutionized the field of machine learning and natural language processing (NLP). Its innovative self-attention mechanism is the backbone that allowed Large Language Models (LLMs) to achieve near-human capabilities in various NLP tasks [18]. However, even with its immense popularity, there are still under-explored aspects of this architecture, specifically, non-sequential positioning during the generative process.

To produce text, LLMs usually employ a method called autoregressive generation, a process in which one token is generated at a time and then re-inputted to the model. Although most language and code generation tasks are better suited as a single, linear sequence of

Instance	S							
	With local search				Without local search			
	A	R	B	K	A	R	B	K
1	-5,25	3	-2,25	3	-2,25	-2,25	3	3
2	-35	25	5	-15	-25	15	35	-5
3	-66,5	57	-38	19	-9,5	-38	57	19

Figure 1. Original example table.

tokens, this type of generation is not exclusive to this kind of sequence. In some instances, the generation process can be more effectively expressed as displaying information within a multidimensional grid. Table generation is one such case, as each generated piece of text relates to a specific point in a two-dimensional matrix of cells and not a single sequence.

So, exploring techniques like incorporating additional positioning information in Transformer models' generative process is crucial to better understand how LLMs interpret and generate data displayed in space. To evaluate this type of generation, we train a model capable of perceiving tabular topology during generation and compare it to a baseline in the task of table recognition.

1.1. The Table Recognition Task and Table Structure

Table recognition (TR) aims to convert unstructured tabular information, like images, PDF files or non-standardized digital documents, into standardized formats, such as JSON, HTML or CSV. The latter being more accessible in terms of indexing and computing [27, 26, 17]. As tables are predominantly designed for human interpretation and require a certain interpretive ability from the readers, effectively parsing this type of information is a complex and widely researched problem. It also commonly employs autoregressive generation of tabular data, making it a clear candidate for multi-dimensional positioning.

To fully comprehend the task and how different positionings might affect the results, we must first analyze how tables are usually presented.

While there is no universally accepted standard for a table structure, common patterns are presented in most structured representations. In office applications, like Microsoft Word, Google Docs, LibreOffice Writer, tables start as an empty grid of cells, which then can be molded into the desired shape through cell merging, splitting and editing. Markup languages, like HTML and LaTeX, tabular environments have a fixed amount of columns, either explicitly defined or inferred, with rows being added as needed; similar to the office applications, a table shape is defined by cells spanning multiple rows and/or columns. In the table understanding literature, authors commonly use one of the formats previously mentioned for datasets and models [4, 27, 26, 2, 28]; in the rare cases where authors diverge from the norm by creating their own structured formats [17], aspects such as cell spanning and merging are still present.

Taking Figure 1 as an example, we outline the general components of a table as follows:

- **Cells:** Cells are basic units of a table, delimiting a self-contained portion of data or empty space.
- **Rows and Columns:** Rows and columns are axes of a table. Their intersections form a matrix of topological coordinates, shown in Figure 2, fundamental for indexing. Since all cells in a table must be indexable, each coordinate of the matrix contains at most one cell.
- **Spanning Cells:** Spanning cells are the ones that occupy more than one coordinate of the tabular matrix, potentially covering multiple rows and columns. They can be understood as a result of merging neighboring cells into a single rectangle. Figure 3 highlights spanning cells in the original example.

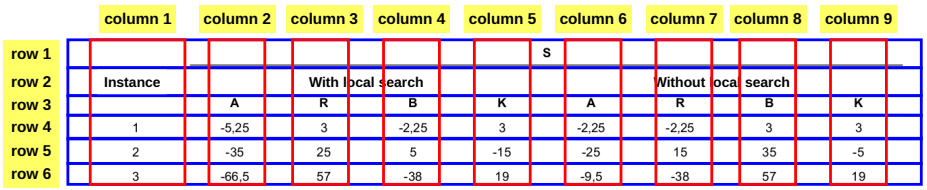


Figure 2. Rows and columns of the example table, creating a grid.

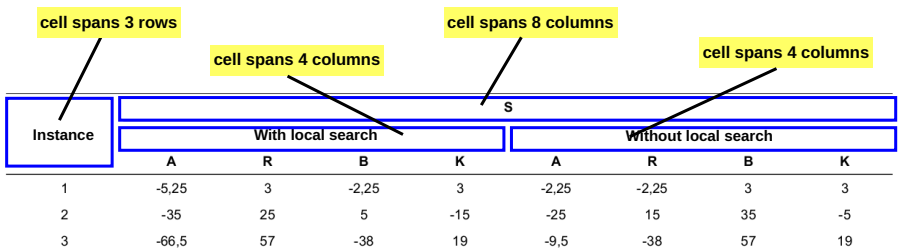


Figure 3. Spanning cells in a table.

- **Headers:** Headers are labels for the columns of a table. They are typically located at the top and can be organized into super-divisions, occupying more than one row. Figure 4 shows the headers of the example table.

1.2. Positioning in the Transformer Architecture

In transformer models, textual data is divided into tokens, inputted as vectors, which are processed independently in feed-forward networks and intertwined through self-attention mechanisms.

A particularity of the self-attention mechanism is that it is indifferent to the arrangement of its input vectors, and it cannot parse each token's position in a sequence. Since the other components of the model treat tokens singularly, these kinds of models can not understand a sequence ordering without it being explicitly supplied. This is usually achieved by adding a positional vector to each input token. Each token is ultimately supplied to the model as the sum of its vector representation and a positional vector. These position vectors are either learned positional embeddings or predefined positional encodings.

Of greater relevance to this work, this specific form of positioning also provides new pathways for experimentation. Since the positioning of each token is not dependent on the order in which data is processed, it is possible to encode a sequence order non-sequentially, like with topological coordinates of rows and columns of a table.

1.3. Goals and Contributions

Our main goal is to investigate the incorporation of higher-dimensional positioning data in autoregressive transformer models. Specifically, the use of three-dimensional positional

Figure 4. Header example.

embeddings (rows, columns, and in-cell sequence) in a generative transformer model for table recognition.

To this end, we trained a topology-sensitive generative model for TR. Then, we compared the model to a baseline of identical architecture with topology-indifferent sinusoidal positional encoding. To better adapt the task for the new kind of positioning, we also developed a topology-centric representation for tables, which can be converted to and from HTML without the loss of information.

Both approaches were compared using the TEDS (Tree Edit Distance Score) metric, as provided by [27]. We evaluated the models' optimization progress during training, in both structure and content recognition; how well they perform for complex tables, those with one or more spanning cells; how robust models are when dealing with exceptionally large tables; and overall performance in the test set.

As the main contributions of this work, we

- Evaluated the use of multidimensional positional embeddings for autoregressive table generation. This is the first of its kind, to the best of our knowledge.
- Showed that a topology-centric approach to TR can significantly improve the generations output (+1.17% TEDS), with greater improvements in complex tables (+2.21% TEDS).
- Showed that TR models employing three-dimensional embeddings can be more robust for larger tables when compared to baseline, with a 29.70% improvement for the top 2% of tables.
- Developed a new topology-centric format for tabular input/output in generative models, with a clearer spanning cell definition.

2. Related Work

This study explores the intersection of multidimensional embeddings within transformer models and their specific application to TR, framed as a generative problem. To contextualize our research, it is essential to evaluate the existing literature in both domains.

2.1. Multidimensional Positional Embeddings

Multidimensional positional embeddings have been utilized effectively in various previous works on visual document understanding (VDU) and table processing. In VDU, language models are often coupled with OCR engines, which provide the document's transcripts along with each word's placement. The LayoutLM family of models [22, 21, 23, 6] effectively combines the textual information and positions provided by OCR engines as input for an encoder model, setting a new standard for VDU classification and QA benchmarks. The LiLT model enhanced VDU multi-language support by employing a dual encoder architecture, with one pretrained model dedicated to text and another to the layout, which consists of the x and y coordinates for each word in the document [20]. When considering tables, [5] showed that table encoding can also be significantly improved by providing models with tabular placement of each token via positional embeddings. Nonetheless, the application of these multidimensional positional embeddings within the context of autoregressive table generation remains mostly unexplored.

2.2. Table Recognition

The field of Table Recognition has seen impressive developments in recent years [3, 9]. As illustrated in works such as [26, 27], as well as contributions from the ICDAR 2021 Competition on Scientific Literature Parsing [7], LLMs are a fundamental factor in such advancements. A recurring theme among the leading algorithms is the integration of a visual encoder with a multi-decoder architecture [26]. These methods often feature a

specialized decoder structure recognition, which is tasked with predicting the tags that define a table's layout [25]; a specialized decoder for content recognition, which is tasked with recognizing the content within each cell detected by the structure model; and, commonly, a decoder model for predicting cell's bounding boxes, either to better reward the encoder for cell detection or allow the use of OCR engines [15, 12, 13]. The last type of encoder clearly underscores the critical role of spatial information in achieving effective recognition, as it shows that a model can enhance its performance simply by better attending to the content's disposition. However, bounding boxes are not always accessible and OCR engines are not always reliable, with inconsistent results when switching from ground-truth text and bounding boxes to actual engines output [8].

In this context, it is clear that the use of spatial information, whether during inference or training, consistently improves metrics in visual document understanding, table understanding recognition. However, this kind of data is typically limited to encoder models or used to train predictive components, rather than directly supplied to the generative models themselves. Consequently, this still unexplored type of positional encoding could provide significant benefits and warrants further assessment.

3. Topology-based Positioning in Autoregressive Sequence Generation

In this section, we describe our approach for incorporating topological positioning during autoregressive generation. We begin by presenting the data and the pretrained model used in the evaluation. Next, we describe the proposed generation process. Finally, we detail the topology-centric table format used for in model.

3.1. The Data

As previously stated, we chose the task of table recognition for our evaluation. Of the most common TR datasets, PubTabNet [26] is the most cited and provides three major advantages: its size, with just over 500,000 example tables, it is one of the largest datasets ever published for this task; complete annotations specific to image-to-sequence models; and the adequate image resolution of the examples, being sufficient for content recognition without excessive storage and processing requirements.

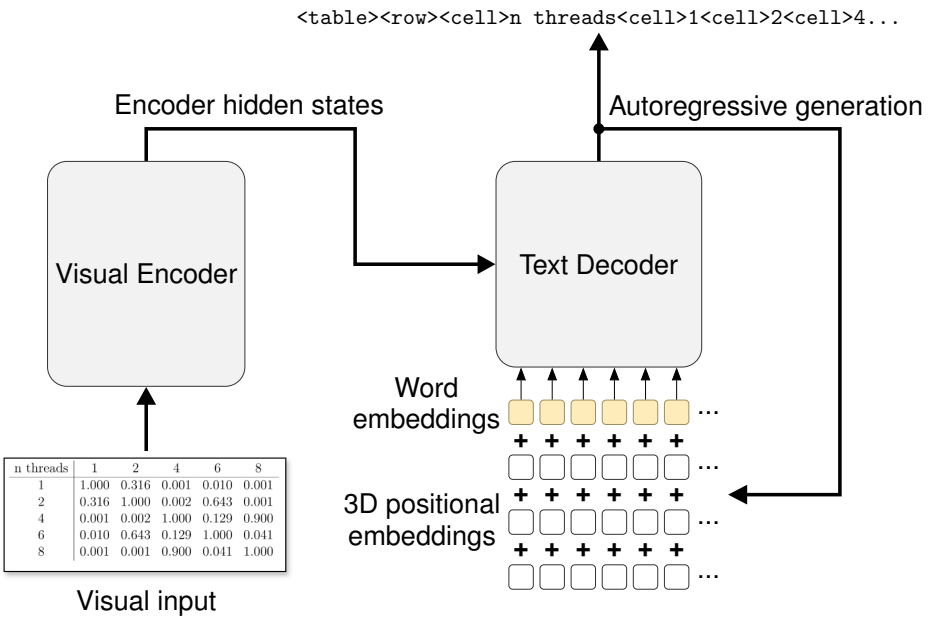
The current version of the dataset consists of a collection of tables from the PubMed website. With predefined splits consisting of roughly 500,000 examples for training, 10,000 for validation, and 10,000 for testing. The set is fully in English, except for some technical/scientific terms in Latin, and on the same topic, Life Sciences. The data is arranged in pairs of images (in color and PNG format) and JSON annotations containing the target, in HTML format, and the metadata.

Most tables in the set contain headers and exactly half of them are complex (i.e., contain at least one spanning cell). The tables' contents feature texts in italics, bold, superscript and subscript. In complex tables, the set has cells with a maximum span of 10 in both rows and columns.

3.2. The Model

To perform the evaluation, we chose to adapt the pretrained model Donut, proposed by [8]. This model employs a Swin Transformer [11] visual encoder for its input, and an mBart decoder [10] to autoregressively generate outputs. Its main application is image-to-sequence parsing of visual documents, which it does without the need for bounding boxes or OCR engines.

Although it has not been thoroughly tested for TR, we chose this model for three main reasons:



n threads	1	2	4	6	8
1	1.000	0.316	0.001	0.010	0.001
2	0.316	1.000	0.002	0.643	0.001
4	0.001	0.002	1.000	0.129	0.900
6	0.010	0.643	0.129	1.000	0.041
8	0.001	0.001	0.900	0.041	1.000

Figure 5. General description of the adapted encoder-decoder architecture. A table is inputted into the encoder, then, the decoder parses the information in an autoregressive generation.

1. Since a pretrained version was openly provided by the authors, the fine-tuning process will not be as computationally intensive as training a new model from scratch.
2. It has not been trained on data present on PubTabNet, avoiding data leakage.
3. Its code is openly available on Hugging Face's Transformers Library[1], which facilitates our modifications.

During our first assessments of the model, we detected that the model's tokenizer was poorly adapted for the task. Its vocabulary lacked a large portion of the scientific symbols common in the data (such as ‡ and ¶). Furthermore, because of its unfamiliarity with the scientific/medical jargon present in PubTabNet, the tokenization was highly unbalanced and inefficient. To address this, we trained a new tokenizer on PubTabNet's training set. Given the data's fairly well-defined context, we set the tokenizer's vocabulary size to a modest 8,129 word pieces. Despite the tighter context, the average size of the tokenized sequences was only slightly reduced, from 559 tokens per example to 517.

3.3. The Adapted Architecture

The base Donut model's mBart decoder uses a single embedding list to define each generated token's position. To encompass three dimensions of the data, we increased the number of embedding lists by two. Each token, then, is inputted as the sum of one word embedding and three positional embedding vectors.

Figure 5 illustrates the overall architecture of the adapted model. Firstly, an image is fed into the visual encoder. The visual encoder processes the image and generates a list of hidden state vectors. The hidden states are then provided to a textual decoder model, together with a task token, the first element of the word embeddings. Finally, the decoder starts the autoregressive generation process, in which every generated token is re-inputted into the model's word embeddings until the generated sequence is complete.

Although incorporating new dimensions into the input of encoders is mostly a direct

[1] https://github.com/huggingface/transformers

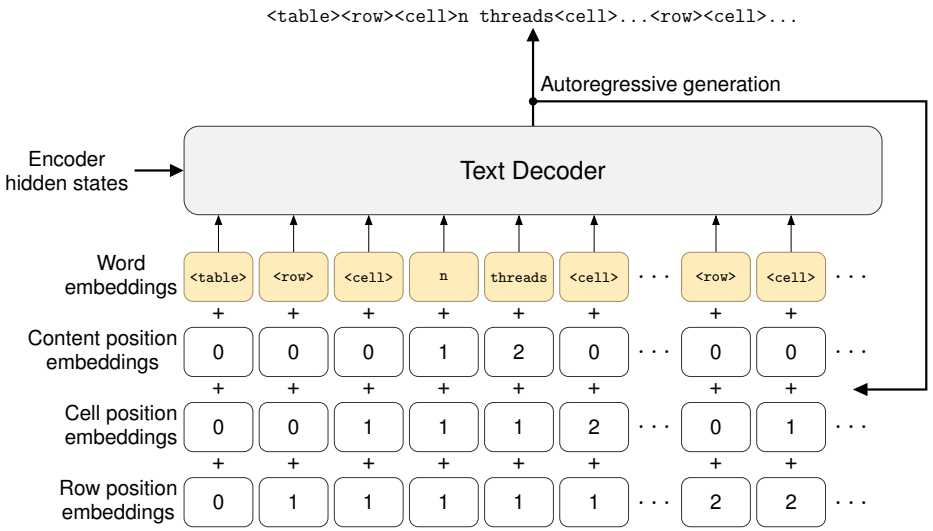


Figure 6. Topological positioning during generation. The tree white lists represent the three dimensions of the tokens in a table. The numbers inside the whites squares represent the token's coordinates in the sequence.

process of retrieval and sum, since the positions are defined outside the model, it is not as simple for generative decoders. Since every generated token is re-introduced to the model, the positions must be calculated inside the model during inference. The next section elaborates on how the model defines each token's position.

3.4. Generation-time Positioning

Figure 6 shows how the position of each token is calculated. The first line, in orange, represents the word embeddings; the other three, in white, represent the positional embeddings. Each line represents a different dimension of the generated table. The overall position of each token in the sequence is given by the sum of the three embeddings.

Each list of embeddings represents a specific progression in the table grid or a cell's content. All three have a specific place in a progression hierarchy:

- The first line of embeddings in white, content position, is the lowest in the hierarchy and encodes positions of the contents inside cells. Its index increases for every token generated.
- The second white list, cell position, is in the middle of the hierarchy and encodes positions of the cells inside a specific row. Its index increases every time a cell-separating token is generated, such as `<cell>` in the example.
- The final list, row position, is the highest in the hierarchy and encodes positions of the rows in the table. Its index increases every time a row-separating token is generated, such as `<row>` in the example.

Just as in the process of exploring the values in a matrix, the tokens are decoded from left to right and then top to bottom. And, just like in matrices, every time the generation advances in a dimension, the dimensions hierarchically under it are reset to zero. In the figure, this process is shown during the generation of the `<cell>` and `<row>` tokens.

This configuration allows the model to explicitly define the topological position of each generated token in constant time during inference. Although not explored in this work, it is reasonable to assume that the same process can be extrapolated to even higher dimensions, such as multiple tables within a single sequence.

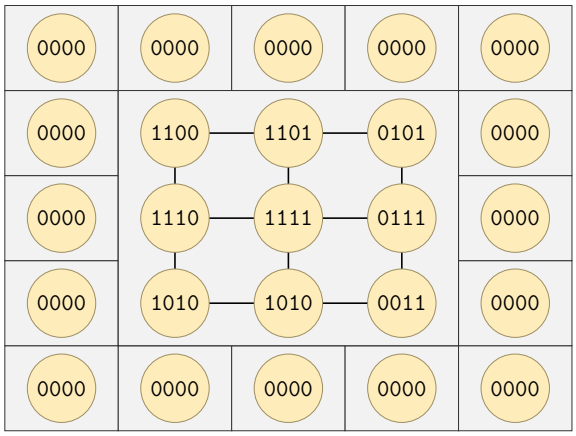


Figure 7. Example of the representation graph of a 5 x 5 table with a 3 x 3 spanned cell at the center. Every node '0000' is a unit subgraph, the nodes in the center form a spanned cell subgraph.

3.5. The Intermediate Sequence

To better adapt inputs to the new generation setup, we also developed a novel table format for the objective sequence. The format can be converted from and to HTML without loss of information, and these differences facilitate the calculation of the topological positionings. These adaptations are:

- Removal of closing tags, such as `</td>` and `</tr>`, found in the HTML language. Only the opening tags are used to indicate the cell's text content, similar to separators of elements in lists. This reduces elements outside the table topology.
- Replacement of the header tags used for each group of lines in the HTML format with header tags in each header cell, replacing the aforementioned start tag. Like the previous change, this reduces unnecessary information outside the table topology.
- A novel spanning cell representation that maintains a uniform structure for the table by representing all coordinates with at least one token. This allows the model to always attend to every position in the table's grid. Otherwise, like in HTML, spanning cells would reduce the amount of cells represented in the next rows and columns, distancing the encoded indexes from the placements in the input image.
  In the new representation, each table is a graph G, with N disconnected subgraphs, where N is the number of cells. Every node in the graph is a position in the table coordinate matrix, illustrated in Figure 2. Each unit cell (that does not span) is a subgraph G_i with 1 vertex and 0 edges. Each spanning cell is a subgraph G_j with $n \times m$ vertices; each vertex connects to its immediate neighbors that are also part of the subgraph, totaling $(n-1) \times m + (m-1) \times n$ edges, where n and m are the size of the spanning cell in the rows and columns, respectively.

  To define the edges of a vertex, the format uses a binary encoding (connected or not connected) with 4 positions, one for each neighbor. The first position represents a connection to the right; the second, a connection below; the third, a connection above; and the fourth, a connection to the left. To not further increase sequence size, the content of an expanded cell is represented only within the initial vertex of each subgraph. Figure 7 is an example of the graph representation for a table.

4. Experimental Setup

To effectively evaluate the proposed model, we compared it against a baseline. The baseline is the exact same Donut model, but adapted for sinusoidal positional encodings, as described by [19], in place of the learned embeddings. This adaptation was made for two reasons: firstly, maintaining the already trained positional embeddings would give the baseline an unfair advantage; second, the sinusoidal position encoding is the preferred type of encoding in most of the table recognition literature, as it allows the model to extrapolate for sequence lengths longer than the ones seen during training [13, 26, 27]. The proposed model was trained to generate the novel table format, which was then converted to HTML for evaluation. The baseline model was trained to generate HTML directly, as it did not require any of the adaptations made.

4.1. Training Setup

Both models were trained on the entire test set for 3 epochs with a batch size of 20, totaling 75,165 optimization steps. The training setup included four 32GB NVIDIA Tesla V100 GPUs, 48 CPU cores and 384GB of RAM. We used an initial learning rate of $8 \cdot 10^{-5}$ with an exponential decay by a factor of $0.125^{\frac{1}{40}}$ every $\frac{75,165}{20}$ steps, until it reaches a value of $1 \cdot 10^{-5}$ at approximately half the training duration, and this value was then maintained until the last step. This learning rate setup was chosen based on estimates by Donut's original authors [8], as resource limitations hindered a thorough hyperparameter search.

To reduce memory consumption during training, we truncated all sequences to a maximum of 1,500 tokens, which still accounts for more than 98% of the total training tokens. We also reduced the input resolution from the base model's 1920×2560 pixels to 640×1280 pixels for the same reason.

Following [1], we also employed data augmentation for the training images as a way to reduce repeating patterns, namely random rotations, random perspective, gaussian blur, color jitter and JPEG compression artifacts.

4.2. Evaluation Setup and Metric

The evaluation setup included a single 32GB NVIDIA Tesla V100 GPU, 12 CPU cores and 96GB of RAM. All evaluations employed beam search with 3 beams and early stopping heuristics, as provided by Hugging Face's Transformers Library. The generation length was capped at 2,048 tokens.

The outputs were evaluated using the Tree Edit Distance Score (TEDS), as made available by [27]. This score quantifies the minimum number of edits required to transform one structured sequence into another, in this case, the structure of HTML tables. The metric calculates two types of distances: the distance between structural tags, which is determined using an edit tree search; and distance between contents, which is given by the Levenshtein distance for string comparison. The distance values are then combined and converted into similarity coefficients ranging from 0 to 100%.

All evaluations were conducted on PubTabNet's validation set. The validation set has the same data composition as the test set, with around 9000 examples equally divided between simple and complex tables.

5. Results

As shown in Figure 8, the models improved quite quickly in the first half of training. Both achieved at least 85% TEDS in the first 12,000 steps and only improved around 1% after the halfway point. Overall, the 3D-Embeddings model maintained a significant

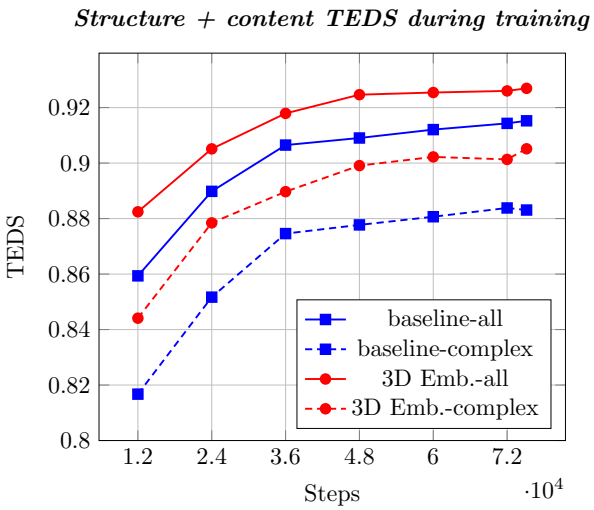


Figure 8. Comparison between models TEDS in the validation set during training. Evaluation considered both content and structure recognition. Complex tables are ones containing at least one spanned cell.

advantage throughout the entire training duration, performing, on average, 1.2% better than the baseline model. Focusing on complex tables (those containing spanned cells), this difference increased to approximately 2.0%, indicating a major improvement in the model's understanding of cell span. Notably, during the final steps of training, the *3D Emb.* model exhibited a significantly larger improvement, particularly for complex tables, which suggests it could benefit from additional training.

Figure 9 shows how models improved during training when only considering structure recognition. Across all tables, the 3D-embeddings model outperformed the baseline significantly during the initial training phase; however, this difference rapidly diminished in the latter half, with the baseline eventually showing a marginal 0.1% advantage. When considering complex tables specifically, the *3D Emb.* maintained an average 0.8% advantage over baseline. This indicates that, even though a sizable benefit from the new positional embeddings comes from content detection, it still significantly enhances spanning cell recognition. The same jump in performance observed previously is mirrored in structure recognition.

An important factor to consider in TR is the generated sequence size. As Figure 10 shows, table lengths follow a lognormal distribution. The vast majority of tables are under 1,000 tokens in length, with around 10% of examples in the 1,000 to 2,000 range, 1% of examples longer than 2,000 tokens and a maximum length of 6083 tokens. Although larger models can easily reach 8K generation length, the skewness of the data still poses a complex issue. As the vast majority of examples are concentrated in the 150 to 1,000 tokens range, models can often get biased towards the mean and lose accuracy for data points outside the most common range.

Table 1 compares the robustness of our model against the baseline when considering different subsets of table size. In this comparison, we consider six subsets selected using the mean and standard deviation from the underlying normal distribution. The first three subsets consider examples that are smaller or equal, to the mean by zero, one, or two standard deviations; the second three, examples that are greater or equal than the mean by the same metric. It is clear that both models perform very well in smaller tables, but performance degrades significantly as they grow in length. When comparing the models, one can see that the three-dimensional model's advantage over the baseline increases with the distance from the mean in both directions, indicating higher robustness in the peripheries of the

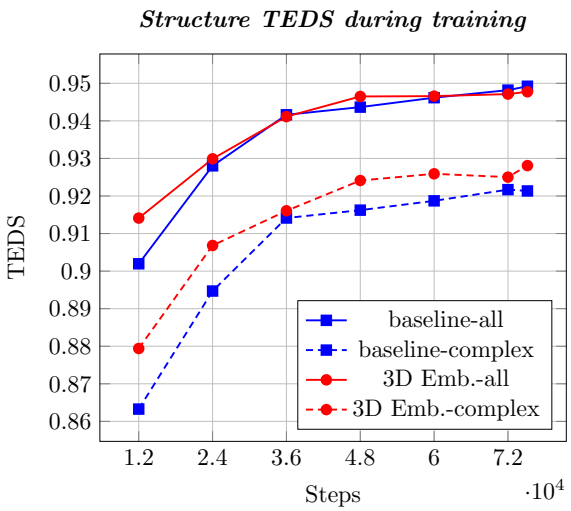


Figure 9. Comparison between models structure TEDS in the validation set during training. Complex tables are ones containing at least one spanned cell.

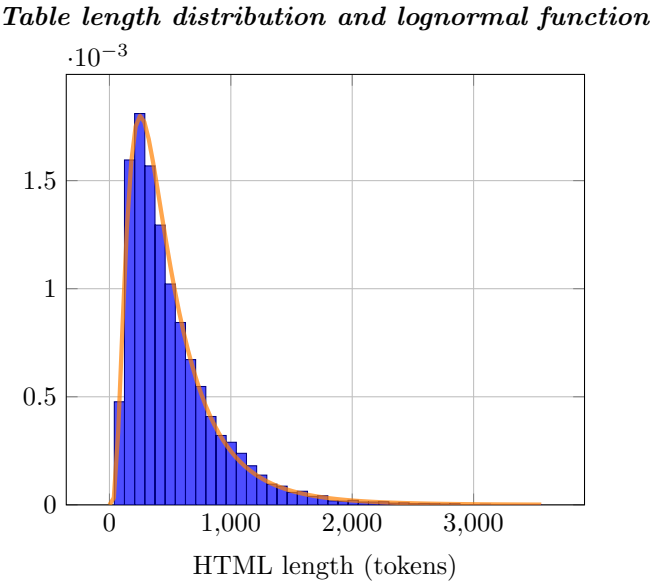


Figure 10. Density histogram of the length, in number of tokens, distribution of the tables in the validation set. The line in orange shows that the lengths closely follow a lognormal distribution. For better visualization, the only two examples longer than 4,000 tokens were removed.

		Subset					
		$\leq \mu - 2\sigma$	$\leq \mu - \sigma$	$\leq \mu$	$\geq \mu$	$\geq \mu + \sigma$	$\geq \mu + 2\sigma$
Structure only	baseline	98.68	97.97	**96.73**	**93.10**	88.84	70.86
	3D Emb.	**98.82**	**97.86**	96.53	93.01	**89.10**	**81.97**
Structure+content	baseline	96.79	95.59	94.12	88.92	82.96	57.05
	3D Emb.	**98.24**	**96.89**	**95.27**	**90.10**	**84.81**	**74.00**

Table 1. Models' TEDS on tables that are smaller and larger than the average by zero, one or two standard deviations. The μ and σ signs represent the mean and the standard deviation of the underlying normal distribution, respectively.

Models	TEDS	
	all	structure
TableMaster [24]	**96.32%**	-
MultitaskLearning [14]	96.17%	**97.88%**
TabStruct-Net [16]	90.01%	90.01%
EDD [27]	88.30%	-
3D Emb. (ours)	91.88%	94.77%

Table 2. Test split complete and structure-only TEDS comparison with models present in the literature.

distribution. This is especially noticeable in tables at the highest length, in which the *3D Emb.* model displays almost 30% higher TEDS.

6. Discussion

It is clear that a topology-focused approach can improve the overall performance for table generation. Especially for content and complex structure recognition. We theorized that this comes as a result of the model being more "positionally aware" of how elements are displayed within the table, better understanding the grid-like disposition of cells and switching more easily between coordinates. This is especially important in complex tables, in which the model can quickly identify how a cell spans by how many spaces it occupies in the aforementioned grid.

However, it must be acknowledged that the more "positionally aware" model still performed similarly to the baseline in structure recognition. Moreover, its advantage in recognizing complex structures implies a lower-than-baseline performance in simple tables. A possible explanation is that the non-visual code elements, like cell and row delimiters, being assigned coordinates could confuse the model as to their placement in the input image. If this is the case, designing a decoder that generates the coordinates in conjunction with the content, instead of using delimiters, could possibly overcome this limitation.

Lastly, when it comes to the robustness to larger tables, the advantage over baseline is most likely due to the three dimensions being more efficient at representing the content's placement and more common during training. For instance, if a model is aware of a table's coordinates, it could infer the positioning of a table of a previously unseen size $N \times M$ as long as it has been trained on tables of size $n \times M$ and $N \times m$, with $n < N$ and $m < M$. In this case, the model would already be familiar with the N and M coordinates and only need to extrapolate their union. On the other hand, if the model treats the whole table as a single sequence, it will be completely unfamiliar with positions outside the maximum training sequence length.

7. Conclusion and Future Works

In this paper, we introduced and evaluated a novel positional encoding for table generation, which accounts for both coordinates of a table's topology in addition to its cell's contents. In our evaluations, we found that this positioning improved performance over the baseline on table recognition by an average of 1.2% TEDS for all tables and 2.0% TEDS for complex tables. Furthermore, the data indicate that the resulting model is significantly more robust than the baseline for very large tables, resulting in a 27.7% improvement in the top 2% of the tables in the evaluated set.

Acknowledging the limitations in the presented work, we hope for more thorough evaluations in the future. Firstly, we believe the architecture used is not fully optimized for the task, therefore, it would be of interest to explore different setups in combination with

topological positioning. Such as the coordinate-generating decoder theorized in the previous section or a multi-decoder setup, which is more common in the TR literature. We also hope that similar techniques are evaluated in fields outside table understanding, like generating components in three-dimensional space or simulating movements in boardgames, like chess or Go.

We believe that the provided evaluation makes it clear that incorporating more complex positional information in generative models is not only possible, but can also enhance a model's output for table generation. Therefore, we hope the information presented can further our understanding of how language models deal with spatial information and incentivize additional explorations on positional dimensionality's role in generative models.

Acknowledgements

The authors acknowledge the National Laboratory for Scientific Computing (LNCC/MCTI, Brazil) for providing HPC resources of the SDumont supercomputer, which have contributed to the research results reported in this paper. URL: http://sdumont.lncc.br. We also thank the Coordination for the Improvement of Higher Education Personnel (CAPES) and the National Council for Scientific and Technological Development (CNPq) for providing financial support during the research period.

Conflicts of Interest

No competing interest is declared.

Data Availability

The source code for model training, evaluation and experiments is available at: https://github.com/joaopaulo7/3D-Positioning-in-Generative-Table-Recognition.

All datasets and pretrained models used are publicly accessible, with references provided in Section 3.

References

[1] Lukas Blecher, Guillem Cucurull, Thomas Scialom, and Robert Stojnic. Nougat: Neural optical understanding for academic documents, 2023.

[2] Michael Cafarella, Alon Halevy, Hongrae Lee, Jayant Madhavan, Cong Yu, Daisy Zhe Wang, and Eugene Wu. Ten years of webtables. *Proceedings of the VLDB Endowment.*, 11(12):2140–2149, aug 2018.

[3] Zewen Chi, Heyan Huang, Heng-Da Xu, Houjin Yu, Wanxuan Yin, and Xian-Ling Mao. Complicated table structure recognition. *arXiv preprint arXiv:1908.04729*, 2019.

[4] Max Göbel, Tamir Hassan, Ermelinda Oro, and Giorgio Orsi. Icdar 2013 table competition. In *12th International Conference on Document Analysis and Recognition*, pages 1449–1453. IEEE, 2013.

[5] Jonathan Herzig, Pawel Krzysztof Nowak, Thomas Müller, Francesco Piccinno, and Julian Eisenschlos. TaPas: Weakly supervised table parsing via pre-training. In Dan Jurafsky, Joyce Chai, Natalie Schluter, and Joel Tetreault, editors, *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 4320–4333, Online, July 2020. Association for Computational Linguistics.

[6] Yupan Huang, Tengchao Lv, Lei Cui, Yutong Lu, and Furu Wei. 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, 2022. Association for Computing Machinery.

[7] Antonio Jimeno Yepes, Peter Zhong, and Douglas Burdick. Icdar 2021 competition on scientific literature parsing. In *Document Analysis and Recognition – ICDAR 2021: 16th International Conference, Lausanne, Switzerland, September 5–10, 2021, Proceedings, Part IV*, page 605–617, Berlin, Heidelberg, 2021. Springer-Verlag.

[8] Geewook Kim, Teakgyu Hong, Moonbin Yim, JeongYeon Nam, Jinyoung Park, Jinyeong Yim, Wonseok Hwang, Sangdoo Yun, Dongyoon Han, and Seunghyun Park. Ocr-free document understanding transformer. In *17th European Conference on Computer Vision – ECCV: Tel Aviv, Israel, October 23–27, 2022, Proceedings, Part XXVIII*, page 498–517, Berlin, Heidelberg, 2022. Springer-Verlag.

[9] Minghao Li, Lei Cui, Shaohan Huang, Furu Wei, Ming Zhou, and Zhoujun Li. TableBank: Table benchmark for image-based table detection and recognition. In *Proceedings of the Twelfth Language Resources and Evaluation Conference*, pages 1918–1925, Marseille, France, May 2020. European Language Resources Association.

[10] Yinhan Liu, Jiatao Gu, Naman Goyal, Xian Li, Sergey Edunov, Marjan Ghazvininejad, Mike Lewis, and Luke Zettlemoyer. Multilingual denoising pre-training for neural machine translation. *Transactions of the Association for Computational Linguistics*, 8:726–742, 2020.

[11] Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo. Swin transformer: Hierarchical vision transformer using shifted windows. In *IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 9992–10002, 2021.

[12] Ning Lu, Wenwen Yu, Xianbiao Qi, Yihao Chen, Ping Gong, Rong Xiao, and Xiang Bai. MASTER: Multi-aspect non-local network for scene text recognition. *Pattern Recognition*, 2021.

[13] Nam Tuan Ly and Atsuhiro Takasu. An end-to-end multi-task learning model for image-based table recognition. In *Proceedings of the 18th International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications - Volume 5: VISAPP*, pages 626–634. SciTePress, 2023.

[14] Nam Tuan Ly and Atsuhiro Takasu. An end-to-end multi-task learning model for image-based table recognition. pages 626–634, 2023.

[15] Ahmed Samy Nassar, Nikolaos Livathinos, Maksym Lysak, and Peter W. J. Staar. Tableformer: Table structure understanding with transformers. *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4604–4613, 2022.

[16] Sachin Raja, Ajoy Mondal, and C. V. Jawahar. Table structure recognition using top-down and bottom-up cues. In Andrea Vedaldi, Horst Bischof, Thomas Brox, and Jan-Michael Frahm, editors, *Computer Vision – ECCV 2020*, pages 70–86, Cham, 2020. Springer International Publishing.

[17] Brandon Smock, Rohith Pesala, and Robin Abraham. Pubtables-1m: Towards comprehensive table extraction from unstructured documents. *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4624–4632, 2022.

[18] OpenAI Team. Gpt-4 technical report, 2024.

[19] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, page 6000–6010, Red Hook, NY, USA, 2017. Curran Associates Inc.

[20] Jiapeng Wang, Lianwen Jin, and Kai Ding. LiLT: A simple yet effective language-independent layout transformer for structured document understanding. In Smaranda Muresan, Preslav Nakov, and Aline Villavicencio, editors, *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 7747–7757, Dublin, Ireland, May 2022. Association for Computational Linguistics.

[21] Yang Xu, Yiheng Xu, Tengchao Lv, Lei Cui, Furu Wei, Guoxin Wang, Yijuan Lu, Dinei Florencio, Cha Zhang, Wanxiang Che, Min Zhang, and Lidong Zhou. Layoutlmv2: Multi-modal pre-training for visually-rich document understanding. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics (ACL) 2021*, 2021.

[22] Yiheng Xu, Minghao Li, Lei Cui, Shaohan Huang, Furu Wei, and Ming Zhou. Layoutlm: Pre-training of text and layout for document image understanding. 2020.

[23] Yiheng Xu, Tengchao Lv, Lei Cui, Guoxin Wang, Yijuan Lu, Dinei Florencio, Cha Zhang, and Furu Wei. Layoutxlm: Multimodal pre-training for multilingual visually-rich document understanding. 2021.

[24] Jiaquan Ye, Xianbiao Qi, Yelin He, Yihao Chen, Dengyi Gu, Peng Gao, and Rong Xiao. Pingan-vcgroup's solution for icdar 2021 competition on scientific literature parsing task b: Table recognition to html. *arXiv preprint arXiv:2105.01848*, 2021.

[25] Zhenrong Zhang, Pengfei Hu, Jiefeng Ma, Jun Du, Jianshu Zhang, Baocai Yin, Bing Yin, and Cong Liu. Semv2: Table separation line detection based on instance segmentation. *Pattern Recognition*, 149:110279, 2024.

[26] Xinyi Zheng, Douglas Burdick, Lucian Popa, and Nancy Xin Ru Wang. Global table extractor (gte): A framework for joint table identification and cell structure recognition using visual context. *IEEE Winter Conference on Applications of Computer Vision (WACV)*, pages 697–706, 2020.

[27] Xu Zhong, Elaheh ShafieiBavani, and Antonio Jimeno Yepes. Image-based table recognition: Data, model, and evaluation. In *16th European Conference on Computer Vision – ECCV 2020: Glasgow, UK, August 23–28, 2020, Proceedings, Part XXI*, page 564–580, Berlin, Heidelberg, 2020. Springer-Verlag.

[28] Xu Zhong, Jianbin Tang, and Antonio Jimeno Yepes. Publaynet: largest dataset ever for document layout analysis. In *2019 International Conference on Document Analysis and Recognition (ICDAR)*, pages 1015–1022. IEEE, Sep. 2019.