STable Permutation-based Framework for Table Generation in Sequence-to-Sequence Models

Anonymous Authors[1]

Abstract

We present a permutation-based text-to-table neural framework that unifies diverse NLP tasks into table outputs. The framework uses a probabilistic approach during training, maximizing the expected log-likelihood across all random permutations of table content factorization. At the inference stage, we optimize model uncertainties and minimize error propagation by leveraging the model's ability to generate cells in any order. Our method accelerates inference by up to 4× on some datasets and improves text-to-table performance by up to 15% over previous solutions, all while preserving output quality.

1. Introduction

In Natural Language Processing, encoder-decoder models unify various tasks by casting them as Question Answering (Kumar et al., 2016; Raffel et al., 2020; McCann et al., 2018; Khashabi et al., 2020; Powalski et al., 2021; Kim et al., 2022). However, this overlooks tasks that demand structured outputs like tables. The problem of accurately extracting and structuring complex information from text into tables is critically important in numerous fields, including data analysis, machine learning, business intelligence, and more. Given the exponential growth of unstructured data, solving this problem is essential because it can help people find and understand information more easily, positively impacting areas of business and science.

We introduce STable, a framework that shifts from text-to-text towards text-to-table inference, opening up new possibilities for tasks like line items extraction and joint entity, and relation extraction. Our model reduces error propagation and eliminates the constraints of a fixed generation order, offering an improvement in decoding speed and performance across multiple datasets.

[1]Anonymous Institution, Anonymous City, Anonymous Region, Anonymous Country. Correspondence to: Anonymous Author <anon.email@domain.com>.

Preliminary work. Under review by the SPIGM workshop at ICML 2023. Do not distribute.

1.1. Limitation of Current Approaches

Existing models based on the transformer encoder-decoder achieve remarkable results in generating complex outputs (Chen et al., 2021; Townsend et al., 2021; Wang et al., 2019; Dwojak et al., 2020). However, they often fail to guarantee the formal validity of the outputs produced, due to error propagation and suboptimal sequential order. Our model uses token-based probabilities to guide the generation process and learns the optimal order of generation without human guidance.

1.2. Contribution

We offer three main contributions: (1) we enhance transformer models with permutation-based decoder training that enables comprehending complex, position-dependent relationships within tables, (2) we develop a structured decoding mechanism that generates table content cell-by-cell, following a dynamic, data-dependent order, and (3) introduce 'tabular attention bias' to the decoder to improve performance and accuracy.

1.3. Related Works

Decoding of Data Structures: Few studies have approached table generation via an encoder-decoder framework. Previous work, such as by Zhong et al. (2020), separated table recognition and cell content generation. Unlike those, we use a single decoder that understands both table structure and content and operates effectively with or without a provided table.

Flexible Generation: The technique of permutation-based training allows output generation in any order. This method has shown promise in areas like machine translation and summarization. Unlike other models, such as by Stern et al. (2019), that generate additional numbers to indicate positions in the output sequence, we sample all cells simultaneously and select the best-scored ones for insertion into their locations.

Permutation-based Language Modeling: Permutation-based language modeling, as demonstrated by Yang et al. (2019), adapted the BERT-like model to work with an autoregressive objective. However, our table-decoding problem

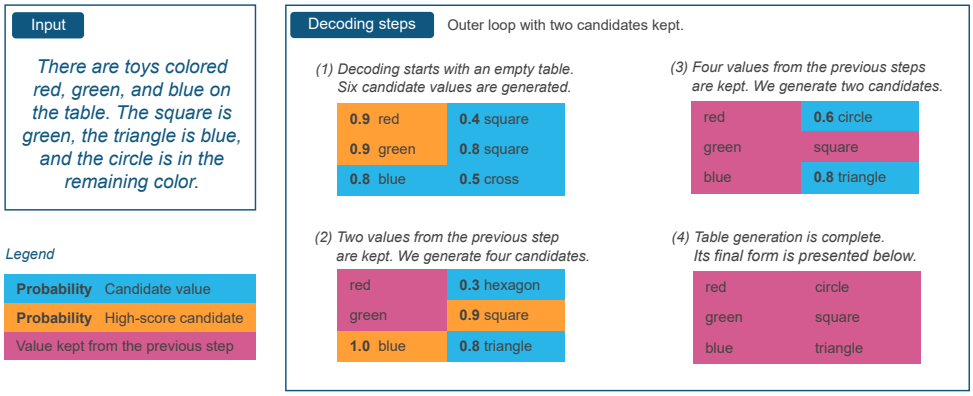

Figure 1. (Inference) A possible progression of decoding a table given the text on the input. The task is to fill in the pairs color-shape. Since the probabilities guide the decoding order, the circle's color that was not explicitly stated in the text is determined at the last step.

necessitates additional constraints, leading us to permute the factorization order of blocks of tokens (representing cells), while maintaining causal order within each cell.

2. STable: A Text-to-Table Framework

STable transforms text into serialized table data using a unique approach that goes beyond a standard autoregressive Transformer decoder. It recognizes table structure, navigating the challenges of sequential expansion and unidirectional context limitation.

The framework addresses the issues of interdependence among table cells and balances the need for generating cells in a non-specific order. It does this by generating complex answers at later stages based on already generated cells. Training the model to operate under varying order generation conditions ensures the model's flexibility during inference.

2.1. Decoding Invariant Under Cell Order

STable's novel decoder design maximizes the expected log-likelihood of cell sequences over all possible orders rather than using the typical top-down, left-to-right approach. More specifically, suppose that we are given a document containing a table with row labels $\mathbf{r} = (r_1, \ldots, r_N)$, and column labels $\mathbf{c} = (c_1, \ldots, c_M)$, which we will collectively denote $\mathbf{h} = (\mathbf{r}, \mathbf{c})$. Note that in practice, there are usually no row labels; however, in the decoder, the special tokens used for distinguishing rows take this role. A linear ordering of the table cells can be represented with a bijection

$$\sigma\colon \{1, 2, \ldots, C\} \to \{1, \ldots, N\} \times \{1, \ldots, M\},$$

where $C = NM$ is the number of cells, so that $\sigma(n) = (i, j)$ are the row and column coordinates of the n-th cell in the ordering. Given such a σ and cell values $\mathbf{v} = (v_{ij})_{i \le N, j \le M}$,

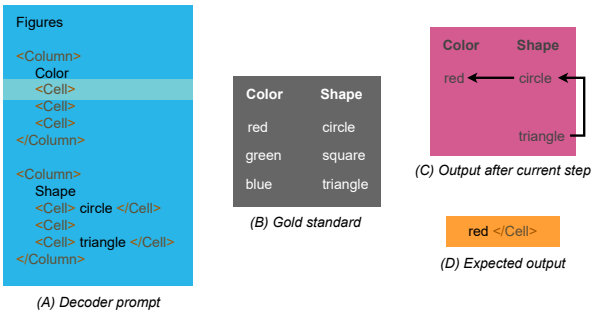

Figure 2. (Training) An example depicting how the answer `red` is produced based on the partially filled cells containing `circle` and `triangle`. (A) The highlighted cell denotes a position where the expected `red </Cell>` should be predicted autoregressively starting from a `<Cell>` token. A successfully decoded cell will lead to the state visible in (C), i.e., the partially decoded gold standard table (B). The generation order of a table is random for each example in the training.

we factorize the likelihood of $\mathbf{v}$ given $\mathbf{h}$ as

$$p_\theta(\mathbf{v}|\mathbf{h}) = \prod_{n=1}^{C} p_\theta\big(v_{\sigma(n)}\big|(v_{\sigma(k)})_{k<n}, \mathbf{h}\big), \tag{1}$$

and using this factorization, we maximize the expected log-likelihood

$$\frac{1}{C!}\sum_{\sigma}\sum_{n=1}^{C} \log p_\theta\big(v_{\sigma(n)}\big|(v_{\sigma(k)})_{k<n}, \mathbf{h}\big) \tag{2}$$

over θ. The likelihoods $p_\theta\big(v_{\sigma(n)}\big|(v_{\sigma(k)})_{k<n}, \mathbf{h}\big)$ themselves can be factorized according to the standard autoregressive approach as

$$p_\theta\big(v_{\sigma(n)}\big|(v_{\sigma(k)})_{k<n}, \mathbf{h}\big) = \\ = \prod_{t=1}^{\ell(v_{\sigma(n)})} p_\theta\big(v^t_{\sigma(n)}\big|(v^i_{\sigma(n)})_{i<t}, (v_{\sigma(k)})_{k<n}, \mathbf{h}\big) \tag{3}$$

Table 1. Results on public and private datasets (mean±std over three runs). The sequence-to-sequence baseline that learns and generates tables as text are provided in the *Linearized* column. The † symbol denotes our TILT training. Underline signifies our model is significantly better than baseline. We calculate speedup by timing the model run for various factors of cell decoding's parallelization, while maintaining performance within one standard deviation from the mean.

Dataset	State-of-the-Art Reference		Linearized	Our Model		Speedup
PWC★	T5 2D (Borchmann et al., 2021)	26.8	27.8 ± 1.0	**<u>30.8</u>** ± 0.5	T5 2D + STable	3.6×
CORD	TILT (Powalski et al., 2021)	**96.3**	92.4 ± 0.7	<u>95.6</u> ± 0.2	TILT† + STable	
Rotowire						
Player	Text-to-Table (Wu et al., 2022)	**86.8**	84.5 ± 0.7	84.5 ± 0.2	T5 + STable	1.1×
Team	*(BART backbone)*	**86.3**	83.8 ± 0.9	<u>84.7</u> ± 0.2		1.8×
DWIE	KB-both (Verlinden et al., 2021)	**62.9**	60.2 ± 1.5	<u>59.2</u> ± 1.5	T5 + STable	2.7×
Recipe…		71.9	60.1 ± 0.3	**<u>75.5</u>** ± 1.6		
Payment…	TILT†	77.0	72.0 ± 2.3	**<u>79.1</u>** ± 0.9	TILT† + STable	
Bank…		61.1	58.7 ± 4.9	**<u>69.9</u>** ± 4.8		
Average		71.1	67.4 ± 1.5	**<u>72.4</u>** ± 1.2		2.3×

where $\ell(v_{\sigma(n)})$ is the length of $v_{\sigma(n)}$ represented as a sequence of tokens $(v^i_{\sigma(n)})_{i \le L}$. In practice, the expected log-likelihood is estimated by sampling bijections σ at random. An example is provided in Figure 2.

2.2. Tabular Attention Bias

STable also introduces an element of tabular attention bias to account for the table structure, building on the concept of relative positional bias in the T5 model. This bias accommodates the two-dimensional nature of tables, including row and column positions, and adjusts for the relative sequential position of tokens in the same cell.

The *tabular bias* τ_{ij} encodes the relative position of table cells in which the tokens lie, while the *local sequential bias* λ_{ij} corresponds to the relative sequential position of tokens belonging to the same cell.

$$\tau_{ij} = \begin{cases} R(r_i - r_j) + C(c_i - c_j) & \text{if } r_j > 0 \\ R_0 + C(c_i - c_j) & \text{if } r_j = 0 \end{cases}$$
$$\lambda_{ij} = \begin{cases} L(i-j) & \text{if } (c_i, r_i) = (c_j, r_j) \\ 0 & \text{otherwise} \end{cases} \quad (4)$$

where (c_i, r_i) are cell coordinates as given by its 1-based column and row indices (with 0 reserved for the header row/column), and $R(k)$, $C(k)$, $L(k)$ and R_0 are trainable weights. After adjusting for these biases, the final attention score takes the form $\alpha'_{ij} = \alpha_{ij} + \tau_{ij} + \lambda_{ij}$.

2.3. Row Count Prediction

The model explicitly predicts the number of rows by passing the embedding of the first token through a linear layer to estimate it. This is contrary to the common practice of ceasing generation on encountering a special token. Training shows that predicting the row count improves results, and that value is used to create a template during inference.

2.4. Model-guided Structured Decoding

As the model was trained with all potential permutations of cell orderings, it inherently learned to handle any variant of a partially-filled table. This gives it the capability to forecast values for all remaining blank cells, as seen in Figure 1. The order in which these cells are filled in the table may only be influenced by fluctuations in the cell's probabilities.

STable uses a model-guided structured decoding algorithm for inference. It minimizes uncertainty by operating greedily, selecting the least uncertain cell at each step. The inference process includes an inner loop for generating cell content and an outer loop for selecting which cell to fill next based on a heuristic. By producing potential content for each cell and then selecting the most likely cell to fill next, the model ensures table validity. See Appendix 1 for details.

3. Experiments

We tested our STable decoder's performance against T5, T5 2D, and TILT models, following the methodology of Wu et al. (2022). Details on evaluation metrics and training procedures are in Appendix.

Our decoder was applied to public datasets (PWC★ and CORD), used for extracting information into table format. Additionally, we evaluated its performance on three private datasets involving tasks like payment details extraction from Payment Stubs and Recipe Composition, and account balances from Bank Statements. As shown in Table, our STable-equipped models outperformed the others by an average of +2.7%, maintaining high scores even on the solved CORD dataset, excluded from our ablation studies.

We also tested our STable decoder on the DWIE dataset for the task of joint entity and relation extraction. Our model achieved close-to-top scores, demonstrating the feasibility of an end-to-end encoder-decoder framework for this task.

Table 2. Results of studies (1), (2), (3), and (5) in relation to complete STable (see also Appendix D).

Model	Score	Change	
Complete STable	62.9 ± 1.0	—	
Semi-templated expansion	61.4 ± 0.1	-1.5	(1)
Fixed causal order	60.0 ± 0.4	-2.9	(2)
Decoding constraint			(3)
Column-by-column	62.4 ± 0.6	-0.5	
Row-by-row	62.1 ± 0.6	-0.8	
L→R and T→B	62.0 ± 0.5	-0.9	
No distant rows	62.2 ± 0.5	-0.7	
Sequential decoder bias only	3.9 ± 0.1	-59.0	(5)
Sequential and header bias	33.2 ± 0.3	-29.7	

Despite some issues with returning entities in their most extensive form, the performance on this dataset suggests a less error-prone approach.

Finally, using the Rotowire table-to-text dataset, we evaluated our approach on generating tables from text. Results demonstrated that our T5+STable model surpassed the Linearized T5 model on Rotowire Team. Our performance reflects the benefits of avoiding linearization, particularly in datasets like Rotowire where finding an optimal column order decoding is computationally expensive for linearized baselines.

4. Unpacking the STable Method: A Study of Variations and Constraints

We evaluated variations of the STable method on Rotowire, DWIE, and PWC★ datasets, repeating trials with different random seeds to mitigate impact of randomness. Results are in Table 3 and Appendices D and B.

(1) From Semi-Templated Expansion to Direct Row Number Prediction: Using a standard approach, we added a NULL-only row at each table's end during training. The resulting performance drop indicates explicit row number prediction's superiority over this method.

(2) Permutation-Based Training vs. Fixed Causal Order: Without permutation-based training, we used a fixed causal order for table reading, resulting in a 2.9 points performance decrease. This reveals the value of bidirectional contexts.

(3) Investigating Constraints in Cell Decoding Order: We examined various constraint methods for cell decoding. Except for the column-by-column approach, all other constraints resulted in performance decreases. Our findings suggest the model-guided inference doesn't require specific decoding order constraints, and column-by-column constraint is intrinsically included in our method.

(4) Effects of Parallelizing Cell Decoding on Performance and Time: We experimented with parallel cell decoding aiming to reduce inference time. Figure 3 shows results; parallelization reduced processing time significantly, without always affecting performance. Specific benefits varied across datasets, therefore, we recommend its experimental determination per dataset.

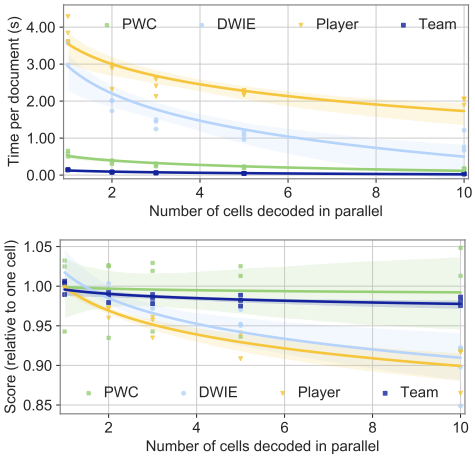

Figure 3. Results of decoding ablation (4). Three runs for 1, 2, 3, 5, and 10 cells decoded in parallel.

(5) Influence of Tabular Attention Biases on Model Performance: Replacing inter-cell and intra-cell relations with a singular 1D global bias affected performance negatively. Adding attention to header names improved performance, although it didn't reach the full model's levels. Analysis showed tabular attention biases positively impact table-modeling.

5. Summary

The cornerstone of our research is the innovation of 1) a training method that randomizes the order of cell factorization, 2) a parallel decoding technique that allows flexible filling of table cells, guided by token probabilities for determining the generation sequence, and 3) an explicit representation of the table in the output. These advancements, supported by our ablation studies, demonstrate that permitting models to generate in any order yields significantly better results than generating in a predetermined order.

Our STable model outperforms the state-of-the-art on the PWC★ dataset. It surpasses linearized models on the CORD and Rotowire-Team datasets and reference models on confidential datasets. Notably, we saw a 14.9% relative improvement on the PWC★ dataset and a 14.4% increase on the Bank Statements dataset.

Interestingly, our model remained robust during parallel cell decoding. This result implies the model's resilience and opens up potential avenues for exploring the advantages and limitations of parallel processing in language models.

References

Borchmann, Ł., Pietruszka, M., Stanislawek, T., Jurkiewicz, D., Turski, M., Szyndler, K., and Graliński, F. DUE: End-to-end document understanding benchmark. In Vanschoren, J. and Yeung, S. (eds.), *Proceedings of the Neural Information Processing Systems Track on Datasets and Benchmarks*, volume 1, 2021. URL https://datasets-benchmarks-proceedings.neurips.cc/paper/2021/file/069059b7ef840f0c74a814ec9237b6ec\protect\discretionary{\char\hyphenchar\font}{}{}-Paper-round2.pdf.

Chen, M., Tworek, J., Jun, H., Yuan, Q., de Oliveira Pinto, H. P., Kaplan, J., Edwards, H., Burda, Y., Joseph, N., Brockman, G., Ray, A., Puri, R., Krueger, G., Petrov, M., Khlaaf, H., Sastry, G., Mishkin, P., Chan, B., Gray, S., Ryder, N., Pavlov, M., Power, A., Kaiser, L., Bavarian, M., Winter, C., Tillet, P., Such, F. P., Cummings, D., Plappert, M., Chantzis, F., Barnes, E., Herbert-Voss, A., Guss, W. H., Nichol, A., Paino, A., Tezak, N., Tang, J., Babuschkin, I., Balaji, S., Jain, S., Saunders, W., Hesse, C., Carr, A. N., Leike, J., Achiam, J., Misra, V., Morikawa, E., Radford, A., Knight, M., Brundage, M., Murati, M., Mayer, K., Welinder, P., McGrew, B., Amodei, D., McCandlish, S., Sutskever, I., and Zaremba, W. Evaluating large language models trained on code. *CoRR*, abs/2107.03374, 2021. URL https://arxiv.org/abs/2107.03374.

Dwojak, T., Pietruszka, M., Borchmann, Ł., Chłędowski, J., and Graliński, F. From dataset recycling to multi-property extraction and beyond. In *Proceedings of the 24th Conference on Computational Natural Language Learning*, pp. 641–651, Online, November 2020. Association for Computational Linguistics. URL https://www.aclweb.org/anthology/2020.conll-1.52.

Khashabi, D., Min, S., Khot, T., Sabharwal, A., Tafjord, O., Clark, P., and Hajishirzi, H. UNIFIEDQA: Crossing format boundaries with a single QA system. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pp. 1896–1907, Online, November 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.findings-emnlp.171. URL https://aclanthology.org/2020.findings-emnlp.171.

Kim, G., Hong, T., Yim, M., Nam, J., Park, J., Yim, J., Hwang, W., Yun, S., Han, D., and Park, S. Ocr-free document understanding transformer. In Avidan, S., Brostow, G., Cissé, M., Farinella, G. M., and Hassner, T. (eds.), *Computer Vision – ECCV 2022*, pp. 498–517, Cham, 2022. Springer Nature Switzerland. ISBN 978-3-031-19815-1.

Kumar, A., Irsoy, O., Ondruska, P., Iyyer, M., Bradbury, J., Gulrajani, I., Zhong, V., Paulus, R., and Socher, R. Ask me anything: Dynamic memory networks for natural language processing. In Balcan, M. F. and Weinberger, K. Q. (eds.), *Proceedings of The 33rd International Conference on Machine Learning*, volume 48 of *Proceedings of Machine Learning Research*, pp. 1378–1387, New York, New York, USA, 20–22 Jun 2016. PMLR. URL https://proceedings.mlr.press/v48/kumar16.html.

Kwiatkowski, T., Palomaki, J., Redfield, O., Collins, M., Parikh, A., Alberti, C., Epstein, D., Polosukhin, I., Kelcey, M., Devlin, J., Lee, K., Toutanova, K. N., Jones, L., Chang, M.-W., Dai, A., Uszkoreit, J., Le, Q., and Petrov, S. Natural questions: a benchmark for question answering research. *Transactions of the Association of Computational Linguistics*, 2019.

Langley, P. Crafting papers on machine learning. In Langley, P. (ed.), *Proceedings of the 17th International Conference on Machine Learning (ICML 2000)*, pp. 1207–1216, Stanford, CA, 2000. Morgan Kaufmann.

McCann, B., Keskar, N. S., Xiong, C., and Socher, R. The natural language decathlon: Multitask learning as question answering. *CoRR*, abs/1806.08730, 2018.

Park, S., Shin, S., Lee, B., Lee, J., Surh, J., Seo, M., and Lee, H. CORD: A consolidated receipt dataset for post-ocr parsing. In *Document Intelligence Workshop at NeurIPS*, 2019.

Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Kopf, A., Yang, E., DeVito, Z., Raison, M., Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L., Bai, J., and Chintala, S. Pytorch: An imperative style, high-performance deep learning library. In Wallach, H., Larochelle, H., Beygelzimer, A., d'Alché-Buc, F., Fox, E., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems 32*, pp. 8024–8035. Curran Associates, Inc., 2019. URL http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style\protect\discretionary{\char\hyphenchar\font}{}{}-high-\protect\discretionary{\char\hyphenchar\font}{}{}performance-deep-learning-\protect\discretionary{\char\hyphenchar\font}{}{}library.pdf.

Powalski, R., Borchmann, Ł., Jurkiewicz, D., Dwojak, T., Pietruszka, M., and Pałka, G. Going full-TILT

boogie on document understanding with text-image-layout transformer. In Lladós, J., Lopresti, D., and Uchida, S. (eds.), *Document Analysis and Recognition – ICDAR 2021*, pp. 732–747, Cham, 2021. Springer International Publishing. ISBN 978-3-030-86331-9. doi: 10.1007/978-3-030-86331-9_47. URL https://link.springer.com/content/pdf/10.1007%2F978-3-030-86331-9_47.pdf.

Raffel, C., Shazeer, N., Roberts, A., Lee, K., Narang, S., Matena, M., Zhou, Y., Li, W., and Liu, P. J. Exploring the limits of transfer learning with a unified text-to-text transformer. *JMLR*, 21(1), jan 2020. ISSN 1532-4435.

Stern, M., Chan, W., Kiros, J., and Uszkoreit, J. Insertion transformer: Flexible sequence generation via insertion operations. In Chaudhuri, K. and Salakhutdinov, R. (eds.), *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pp. 5976–5985. PMLR, 09–15 Jun 2019. URL https://proceedings.mlr.press/v97/stern19a.html.

Townsend, B., Ito-Fisher, E., Zhang, L., and May, M. Doc2dict: Information extraction as text generation. *CoRR*, abs/2105.07510, 2021. URL https://arxiv.org/abs/2105.07510.

Verlinden, S., Zaporojets, K., Deleu, J., Demeester, T., and Develder, C. Injecting knowledge base information into end-to-end joint entity and relation extraction and coreference resolution. In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, pp. 1952–1957, Online, August 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.findings-acl.171. URL https://aclanthology.org/2021.findings-acl.171.

Wang, X., Tu, Z., Wang, L., and Shi, S. Self-attention with structural position representations. 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)*, pp. 1403–1409, Hong Kong, China, November 2019. Association for Computational Linguistics. doi: 10.18653/v1/D19-1145. URL https://aclanthology.org/D19-1145.

Wu, X., Zhang, J., and Li, H. Text-to-Table: A new way of information extraction. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 2518–2533, Dublin, Ireland, May 2022. Association for Computational Linguistics. doi: 10.18653/v1/2022.acl-long.180. URL https://aclanthology.org/2022.acl-long.180.

Yang, Z., Dai, Z., Yang, Y., Carbonell, J., Salakhutdinov, R. R., and Le, Q. V. XLNet: Generalized autoregressive pretraining for language understanding. In Wallach, H., Larochelle, H., Beygelzimer, A., d'Alché-Buc, F., Fox, E., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019. URL https://proceedings.neurips.cc/paper/2019/file/dc6a7e655d7e5840e66733e9ee67cc69-Paper.pdf.

Zaporojets, K., Deleu, J., Develder, C., and Demeester, T. DWIE: An entity-centric dataset for multi-task document-level information extraction. *Information Processing & Management*, 58(4):102563, 2021. ISSN 0306-4573. doi: https://doi.org/10.1016/j.ipm.2021.102563. URL https://www.sciencedirect.com/science/article/pii/S0306457321000662.

Zhong, X., ShafieiBavani, E., and Jimeno Yepes, A. Image-based table recognition: Data, model, and evaluation. In Vedaldi, A., Bischof, H., Brox, T., and Frahm, J.-M. (eds.), *Computer Vision – ECCV 2020*, pp. 564–580, Cham, 2020. Springer International Publishing. ISBN 978-3-030-58589-1.

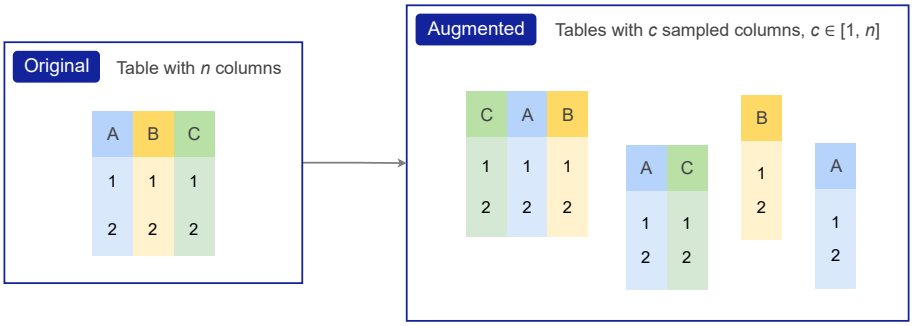

Figure 4. Change in training illustrated as augmentation of permuted sub-tables from the original table.

A. Table Decoding Algorithm

The algorithm presented above operates on the output of the encoder model and reuses the cached encoded representations that are considered to be a part of the DECODERMODEL for brevity. Another important characteristic of the DECODERMODEL introduced for conciseness of the pseudocode is that it produces all cell tokens and handles the sequential text decoding on its own.

The decoding employs an OUTERLOOP, parametrized by the k parameter (denoting the parallelization of cell decoding) that progresses cell-by-cell, the INNERLOOP function that generates each cell that is yet to render, and OUTERCRITERION — a selection heuristics that determine which cell, from all the finalized in the inner loop, should be added to the outer loop. The INNERCRITERION is a heuristic we utilize that selects the cell with the maximum probability for its tokens' predictions.

In the INNERLOOP, each cell is decoded until the special token determining the end of cell generation is placed. As the INNERLOOP generates each cell autoregressively and independently from other cells, the process can be treated as generating multiple concurrent threads of an answer and is well parallelizable. In the worst case, it takes as many steps as the number of tokens in the most extended cell.

After the selection by the OUTERCRITERION heuristic, the cell from the inner loop is inserted into the outer loop, and made visible to all other cells, while the cells that were not selected are to be reset and continuously generated in the future steps until they are chosen by the OUTERCRITERION heuristics.

B. Negative Result: Prevention of Column Order Leakage

In STable, the sequence of column labels $\mathbf{c}$, on which the likelihoods are conditioned, may leak additional unwanted information to the decoder. If the data in the document are indeed formatted as a table, and the order of labels in $\mathbf{c}$ matches the column order, the model might learn to extract cells by location, instead of using the actual semantics of the cell label. However, during inference, while we know which entities we want to extract from the document, we are not given the order in which they appear, which can be perceived as a serious train-inference discrepancy.

To remedy this problem, we tried to further modify the training objective (See Figure 4). Denote by $\mathcal{C}$ the set of all non-empty sequences of distinct column labels. Instead of all the cells $\mathbf{v}$, we can predict only the cells $\mathbf{v_c}$ corresponding to a sequence $\mathbf{c} \in \mathcal{C}$ of columns, in the order defined by the order of columns in $\mathbf{c}$. The expected log-likelihood over all $\mathbf{c} \in \mathcal{C}$ can be then expressed as

$$\log p_\theta(\mathbf{v}|\mathbf{h}) = \frac{1}{|\mathcal{C}|} \sum_{\mathbf{c} \in \mathcal{C}} \log p_\theta(\mathbf{v_c}|\mathbf{r}, \mathbf{c}), \tag{5}$$

where $p_\theta(\mathbf{v_c}|\mathbf{r}, \mathbf{c})$ decomposes according to original STable.

In practice, we found it to have no relevant impact on the training process. It did not lead to significant changes in evaluation scores when used in the supervised pretraining stage or on a downstream task. Consequently, we abandoned the idea and did not use it for any of the models reported in the paper. This study helps us state that the model learns the semantics of the cell labels without a need for regularization.

Algorithm 1 Table Decoding Algorithm of our proposal.

procedure OUTERLOOP(k)
  $T \leftarrow 0_{n,m,l}$ {$n \times m$ table with l padding tokens per cell}
  $C \leftarrow 0_{n,m}$ {current cell status (decoded or not)}
  while SUM(C) $< nm$ **do** {while there is a cell to decode}
    $T', L \leftarrow$ INNERLOOP(T, C) {create complete table candidate T' and cell scores}
    $\mathcal{B} \leftarrow$ OUTERCRITERION(L) {sequence of coordinates sorted according to scores}
    for $c \leftarrow 1, k$ **do** {for k best cells from T'}
      $i, j \leftarrow \mathcal{B}_c$ {get coordinates}
      $T_{i,j} \leftarrow T'_{i,j}$ {...copy values to table T accordingly}
      $C_{i,j} \leftarrow 1$ {...and mark the appropriate cell as already decoded}
    end for
  end while
  return T
end procedure

procedure INNERLOOP(T, C)
  $L \leftarrow 0_{n,m}$ {scores for each cell in $n \times m$ table}
  $T' \leftarrow T$ {inner loop's table copy}
  parfor $i \leftarrow 1, n$ **do** {for each table row}
    parfor $j \leftarrow 1, m$ **do** {...and each table cell processed in parallel}
      if $C_{i,j} = 0$ **then** {...if it was not decoded yet}
        $s, t \leftarrow$ DECODERMODEL(T, i, j) {produce cell tokens t and their scores s}
        $L_{i,j} \leftarrow$ INNERCRITERION(s) {aggregate per-token scores into cell score}
        $T'_{i,j} \leftarrow t$ {update table copy}
      end if
    end parfor
  end parfor
  return (T', L)
end procedure

procedure INNERCRITERION(s)
  /* Any $\mathbb{R}^n \rightarrow \mathbb{R}$ function. STable assumes max, but we test other in the ablation studies. */
end procedure

procedure OUTERCRITERION(L)
  /* Some $\mathbb{R}^{m \times n} \rightarrow (\mathbb{N} \times \mathbb{N})^{mn}$ function returning a permutation of indices of the input matrix L. STable assumes sort of matrix coordinates according to descending values of its elements, but we test other functions in the ablation studies. */
end procedure

Table 3. Results of studies on decision criteria. Modified models in relation to complete STable. See Appendix D for per-dataset results.

Model		Score	Change
Complete STable		62.9 ± 1.0	—
Criteria (inner, outer)			
min	max	61.7 ± 0.7	-1.2
mean	max	62.7 ± 0.7	-0.2
mean	min	60.8 ± 0.7	-2.1
min	min	62.1 ± 0.4	-0.8
max	min	61.2 ± 0.2	-1.7

C. Inner/Outer Loop Decision Criteria

The heuristic we test selects the cell in the outer loop based on the minimal or maximal inner score. Such inner score is calculated in three different ways: by taking the minimal, maximal, and mean of the token's logits score. The results, presented in Table 3, point to the lesser importance of choosing the inner scoring method, while the choice of the outer loop heuristics impacts results more significantly. The former is the desired behavior since the STable's algorithm is based on the assumption that it is beneficial to decode cells starting from those with the model's highest confidence. On the other hand, as there is a significant variance depending on the dataset chosen, these and other inference parameters can be subject to cost-efficient, task-specific hyperparameter optimization.

D. Details of Experiments and Ablation Studies

All models were trained three times with different random seeds. We relied on *large* variants of the models for experiments in Table 1, and on *base* variants for the ablation studies. These are analyzed in Table 3 given the average results over Rotowire, PWC*, and DWIE datasets (see Table 4 for detailed scores).

Metrics. We rely on the original metrics for all but the DWIE dataset, i.e., GROUP-ANLS for PWC*, F1 for CORD, and non-header exact match cell F1 for Rotowire (other variants proposed by the authors are reported in Table 7). Use of the original DWIE metric was not possible, as it assumes a step-by-step process. In contrast, we tackle the problem end-to-end, i.e., return *(object, relation, subject)* tuples without detecting all entity mentions within the document and their locations. To ensure a fair comparison, we use the F1 score calculated on triples; that is, we require the model to return the exact match of the triple. Such a setup is very demanding for encoder-decoder models as the convention in DWIE is to require *object* and *subject* to be returned in the longest form of appearance in the document.

Pretraining/adaptation. Due to the switch to permutative training and the addition of the regression head, there is a significant change in the model objective. Consequently, we anticipated the necessity of the model adaptation phase. It consists of the pretraining stage equivalent to the one conducted by authors of the TILT model (Powalski et al., 2021) extended by Natural Questions (Kwiatkowski et al., 2019) and WebTables[1] datasets. To utilize WebTables we rendered webpages, from which the tables were scraped and taught models to extract table contents from webpages. The said stage is applied to all T5+STable, T5 2D+STable, and TILT+STable models.

Hyperparameters. We use task-independent hyperparameters that roughly follow these proposed by the authors of the T5 model for its finetuning, as during our initial experiments, they turned out to be a robust default (see Table 5).

Maximal input sequence lengths were chosen in such a way a fair comparison with reference models was ensured. In particular, we use T5+2D's limit despite the fact one can achieve better results when consuming a more significant part of the input document. Similarly, the max number of updates follows the limit in reference models except for the DWIE dataset, where the state-of-the-art solution is based on the incomparable multi-step pipeline. See Table 6 for these task-specific details.

Software and hardware. All experiments and benchmarks were performed on DGX-A100 servers equipped with eight A100-SXM4-80GB GPUs that feature automatic mixed precision. Our models and references were implemented in PyTorch 1.8.0a0 (Paszke et al., 2019) with CUDA 11.4 and NVIDIA drivers 470.82.01.

[1] https://webdatacommons.org/webtables/

Table 4. Per-dataset results of studies (1), (2), (3), and (4). Modified models in relation to Complete STable.

Model	RW Player	RW Team	PWC★	DWIE	
Complete STable (reference)	82.7 ± 0.3	84.1 ± 0.7	27.5 ± 2.2	56.0 ± 1.4	
Semi-templated expansion	80.4 ± 0.5	84.1 ± 0.5	25.0 ± 0.8	56.1 ± 1.0	(1)
Fixed causal order	83.2 ± 0.4	84.3 ± 0.3	26.3 ± 1.6	46.5 ± 0.5	(2)
Decoding constraint					(3)
Column-by-column	82.5 ± 0.4	84.0 ± 0.5	28.4 ± 1.5	54.8 ± 0.8	
Row-by-row	80.2 ± 0.4	83.8 ± 0.4	27.6 ± 1.6	56.8 ± 0.8	
L→R and T→B	83.1 ± 0.5	84.1 ± 0.7	27.7 ± 1.8	53.2 ± 0.5	
No distant rows	82.7 ± 0.5	83.8 ± 0.6	28.1 ± 1.0	54.2 ± 1.2	
Decision criteria (inner × outer)					(4)
min max	81.9 ± 0.4	83.7 ± 0.5	26.5 ± 2.0	54.2 ± 0.8	
mean max	83.0 ± 0.3	83.8 ± 0.8	27.8 ± 1.4	56.1 ± 1.1	
mean min	81.2 ± 1.1	83.7 ± 0.6	26.4 ± 1.9	51.9 ± 0.5	
min min	82.8 ± 0.6	83.8 ± 0.5	27.6 ± 1.3	54.0 ± 0.5	
max min	82.3 ± 0.3	84.5 ± 1.0	20.7 ± 1.6	52.7 ± 0.4	
Sequential decoder bias only	0.3 ± 0.1	0.6 ± 0.3	14.1 ± 0.3	0.6 ± 0.1	(5)
Sequential and header bias	16.0 ± 0.4	45.1 ± 0.4	27.7 ± 2.0	44.2 ± 1.2	

Table 5. Task-independent hyperparameters used across all experiments.

Hparam	Dropout	Batch	Learning rate	Weight decay	Label smoothing	Optimizer
Value	.1	64	1e-3	1e-5	.1	AdamW

Table 6. Task-dependent hyperparameters and training details. (*) Length equal to the one consumed by the baseline model.

Dataset	Max steps		Max input
	Ablation	Final	length
PWC★	500	1,000	6,144*
Rotowire	3,000	8,000	1,024
CORD	—	36,000	1,024
DWIE	4,000	8,000	2,048
Recipe Composition	—	400	2600
Payment Stubs	—		
Bank Statements	—	200	7000

Table 7. Detailed results of experiments on reversed Rotowire dataset. See Wu et al. (2022) for metrics' specification.

.

	Row header F1			Column header F1			Non-header F1		
	Exact	Chrf	BERT	Exact	Chrf	BERT	Exact	Chrf	BERT
Team	94.9	95.2	97.8	88.9	85.8	88.7	84.7	85.6	90.3
Player	93.5	95.3	95.1	88.1	91.2	94.5	84.5	86.8	90.4

Table 8. Summary of the confidential datasets.

	Recipe Composition	Payment Stubs	Bank Statements
train documents	119	80	111
val documents	16	10	10
test documents	30	20	10
avg doc len (words)	0.6k	0.3k	1.3k
max doc len (words)	1.6k	2k	4.9k
avg doc len (characters)	3.3k	2k	8.3k
max doc len (characters)	10k	14.2k	37.9k
properties total	64	11	10
properties in tables (tables columns)	64	4	4
properties outside of tables	0	7	6
mean number of table rows	12	5	2
max number of rows	60	15	5
mean length of cell (characters)	12	8	9
max length of cell (characters)	308	44	36

E. Business Datasets

Due to the sparsity of public benchmarks for complex information extraction, we decided to provide results on three confidential datasets. They assume, respectively, (1) the extraction of payments' details from *Payment Stubs*, (2) *Recipe Composition* from documents provided by multinational snack and beverage corporation, as well as (3) account balances from *Bank Statements*. Their details are covered in the present section and Table 8.

Recipe Composition. The problem faced is extracting proprieties of food ingredients from confidential food manufacturer's documentation. This dataset contains 165 annotated fragments from 55 documents, three pieces for each document, with annotations sourced from the corporation's CRM system.

For each file, there are five tables to be extracted. The first one describes the ingredient's physical and chemical parameters (i.e., parameter name, testing method, range of allowed values, unit of measurement, and testing method details). The second one describes sub-components of the ingredient (i.e., its quantity, name, allergens, ingredient function, and country of origin). The third table informs about the presence of allergens (e.g., their names and binary information about their presence). The last two tables contain a quantity of the allergens (e.g., names and their qualities) as sub-components and caused by contamination retrospectively.

The first table needs to be extracted from the first document fragment, the second table – from the second fragment, and the three last tables – from the third document fragment. Input documents feature tables and fulfilled forms, where properties are presented in the form of text or check-boxes.

The analysis of expected outputs shows a high level of variability concerning the factors of table length (1 to 60 rows) and answer type (either a binary value, number, complex chemical name, or a more extended description).

Payment Stubs. The second of our private datasets consists of 110 American payment stubs, i.e., documents obtained by an employee regarding the salary received.

We aim to extract employee and employer names, dates, and payment tables, where each row consists of payment type, hours worked, and payment amount. Since documents come from different companies, their layouts differ significantly.

Due to the straightforward form of information to be extracted, a single annotator annotated each document. We state these were annotated ethically by our paid co-workers.

Bank Statements. The last dataset consists of 131 annotated bank statements. The goal here is to extract bank and customer name, date of issue, and table of account balances (e.g., account number, balance at the beginning of the period, and balance at the end).

Data to be comprehended is partially presented in the document's header and partially in multiple forms (each for one

account).

Similar to the Payment Stubs dataset, documents here were issued by different banks and represent a broad spectrum of layouts. The annotation process was the same as for the Payment Stubs dataset.

F. Adaptation to Table Structure Recognition Task

Our method by design does not generate the table header since we assume that the names of the datapoints to infer are given in advance. To tackle problems such as table structure recognition where the set of possible header values is not limited, one needs to slightly modify the proposed solution. However, we do not consider it a serious limitation as the required modification is relatively straightforward, and for the sake of completeness, we describe it below.

To adjust the proposed method to be applicable to the task of Table Structure Recognition, one must understand the differences in framing the problem between the tasks here.

Table Structure Recognition or Table Extraction aims to generate headers and the table content based on the document with the table provided explicitly. STable described in the main part of this paper can generate the table given any text and its position on pages. This capacity generalizes well to any input, including when the table is provided on the input. The difference is that the output form in STable assumes the headers are known upfront, while for Table Structure Recognition, inferring them is a part of the task. STable can achieve such capabilities to solve the Table Structure Recognition task by (1) adding a linear layer to predict the number of columns, (2) treating headers as the values to be inferred in the first row, (3) using dummy names of the columns, e.g., "first column," "second column," and (4) increasing the predicted number of rows by 1.

In this setup, the model will predict the number of columns and the number of rows, while the first row will represent the values of header names. The dummy headers will have to be removed during postprocessing, and the values in the first row should be treated as valid headers.

G. Sample Input-Output Pairs

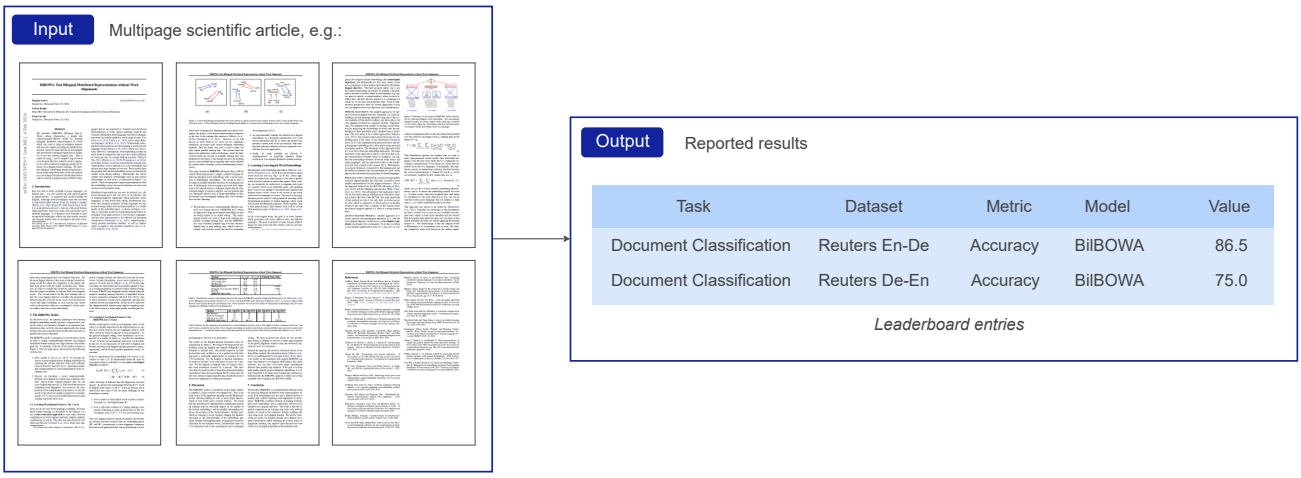

Figure 5. An example from PWC★ dataset considered in the document-to-table paradigm.

PWC★ (Borchmann et al., 2021). Input in the PWC★ consists of born-digital, multipage PDF files containing an article from the machine learning field. The expected output is a list of tuples describing achieved results on arbitrary datasets (see Figure 5).

CORD (Park et al., 2019). Input in the dataset is a single scanned or photographed receipt. From our point of view, the

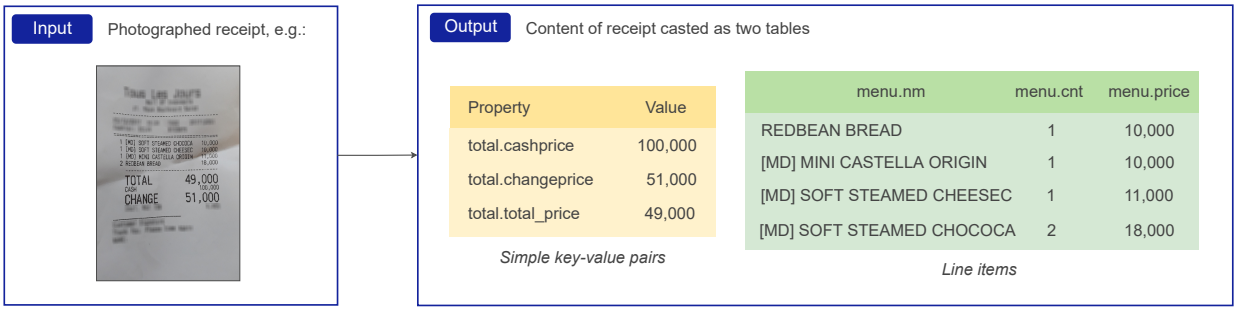

Figure 6. Sample document from CORD dataset and its expected output as interpreted in our approach.

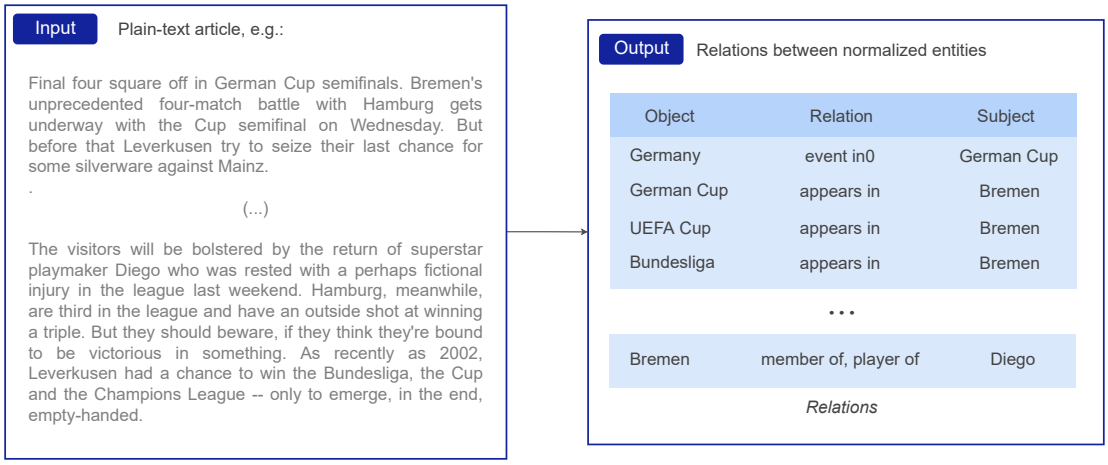

Figure 7. Sample input-output pair from the DWIE dataset. The table was shortened and consisted of 29 rows in our approach. Suppose multiple relations appear in the same direction between the pair of object-subject. In that case, we predict a list of them in a single cell, reducing the number of rows generated (see the example of the Bremen-Diego pair).

output here is twofold — there are simple data points that can be considered key-value pairs and data points that take the structured form of line items. We approach the problem as the generation of two tables from the document — one for each data kind (see Figure 6).

DWIE (Zaporojets et al., 2021). Input in the dataset is a plain-text article. The final goal is to extract the normed object, relation, and subject triples (though the original formulation assumes several intermediate stages). Triples are always complete (i.e., there are no NULL values, as we understand them (see Figure 7 for an example).

Reversed Rotowire (Wu et al., 2022). Input in the reversed Rotowire dataset, as reformulated by (Wu et al., 2022), is a plain-text sport news article. The task is to generate tables with team and player statistics. The number of rows in the *Team* table is from zero (if no team is mentioned in the text) to two, whereas the number of rows in the *Player* is highly variable and content-dependent. Figure 8 present sample pair of document and tables to generate.

The horse face emoji we feature is a part of Noto Emoji distributed under the Apache License 2.0. Copyright by Google Inc. No animals were harmed in the making of this article.

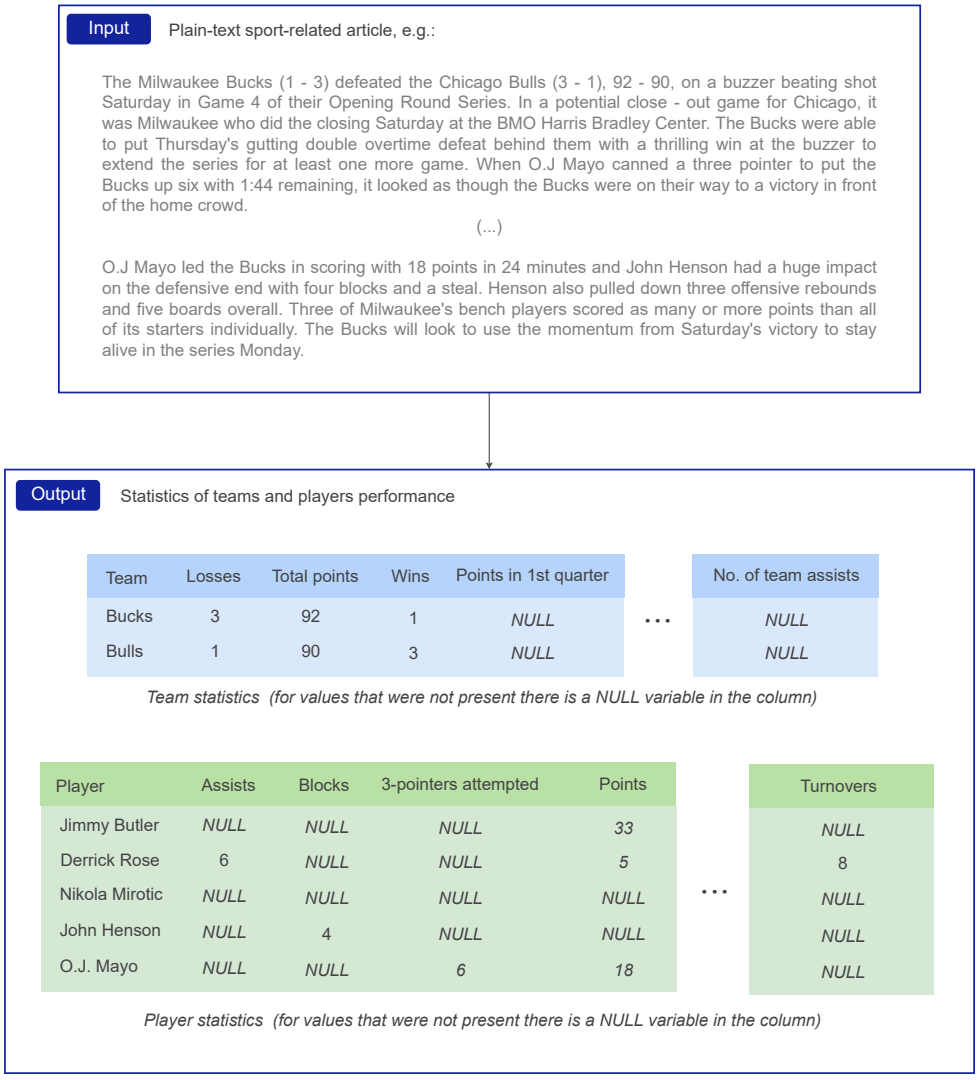

Figure 8. Input-output example from the reversed Rotowire dataset. We present shortened forms of tables than in real have 13 columns for Team and 20 columns for Player tables. Note that there is a *NULL* value in the column for values not present in the input text.