TableMBR: Minimum Bayes Risk Table Generation Based on Structural Consistency

Daiki Yoshida[†], **Hiroyuki Deguchi**[‡], **Yusuke Sakai**[†],
Hidetaka Kamigaito[†], **Taro Watanabe**[†]
[†]Nara Institute of Science and Technology (NAIST) [‡]NTT, Inc.
yoshida.daiki.ye6@naist.ac.jp, hiroyuki.deguchi@ntt.com
{sakai.yusuke.sr9, kamigaito.h, taro}@is.naist.jp

Abstract

The text-to-table task aims to generate structured data in tabular formats from unstructured text. While the integration of large language models (LLMs) has significantly enhanced the comprehensiveness and flexibility of generation, challenges regarding inconsistent output quality persist, such as the inclusion of redundant information and numerical inaccuracies. We propose TableMBR, a robust table generation method that maintains structural consistency through minimum Bayes risk (MBR) decoding. Experimental results showed that TableMBR outperforms the baseline, achieving relative improvements of up to 15% in F1 score on Rotowire and 23% in accuracy on LiveSum.

1 Introduction

The text-to-table task aims to generate structured tables from input text written in natural language. With the emergence of large language models (LLMs), flexible table generation is becoming increasingly feasible, including the generation of the schema, defined as the structural framework consisting of row and column headers (Deng et al., 2024; Sundar et al., 2024; Ahuja et al., 2025). However, ensuring structural consistency remains a significant challenge. This involves generating the appropriate number of row and column headers and producing exact cell values that correspond to those headers. Map&Make (Ahuja et al., 2025) addressed this by fixing the schema before generating each cell value, and achieved better accuracy compared with other existing methods. Nevertheless, Map&Make lacks a mechanism to explicitly maintain the consistency of the table structures. This leads to risks such as the addition of redundant columns or the inclusion of numerical errors.

In natural language text generation tasks such as machine translation, minimum Bayes risk (MBR) decoding has been proposed to generate high-quality text with fewer errors by maximizing the expected utility (Kumar and Byrne, 2004; Eikema and Aziz, 2020). MBR decoding generates multiple output hypothesis texts and estimates the expected utility of each hypothesis using a utility function such as an automatic evaluation metric. The hypothesis that maximizes the estimated utility is then selected as the final output.

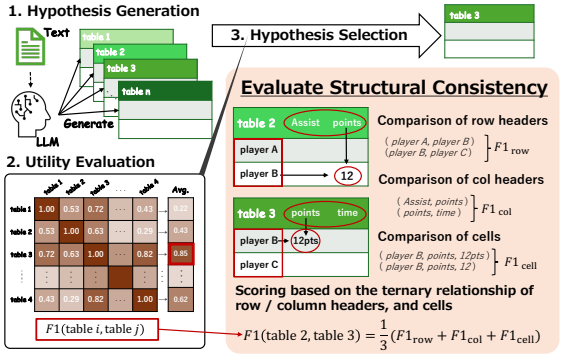

Figure 1: Table generation by our proposed TableMBR

To improve the structural consistency of table generation, we propose TableMBR, a minimum Bayes risk table generation method that maximizes the expected structural consistency. Figure 1 illustrates an overview of TableMBR. TableMBR first generates multiple hypothesis tables using Map&Make. We then sample pseudo-reference tables and evaluate the F1-based structural utility to estimate the expected utility of each hypothesis. Specifically, we define the utility as a linear combination of the F1 scores of three components: row headers, column headers, and cells. To enable more flexible matching in F1 score calculation, we use not only exact match but also lexical similarity (Popović, 2015) and semantic similarity (Zhang et al., 2019). Finally, TableMBR outputs the best hypothesis table that maximizes the expected utility. We evaluated the effectiveness of the proposed TableMBR using Gemini-2.5 Flash (Comanici et al., 2025) and GPT-4.1 mini (OpenAI et al., 2024). The experimental results demon-

Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 4: Student Research Workshop), pages 1087–1102

strated that TableMBR outperforms the baseline, Map&Make, achieving relative improvements of up to 15% in F1 score on Rotowire (Wu et al., 2022) and 23% in accuracy on LiveSum (Ahuja et al., 2025).

2 Background and Related Work

2.1 Text-to-Table Generation

Wu et al. (2022) proposed text-to-table as an information extraction task that involves generating tables from natural language text. They demonstrated that table-formatted information, which possesses complex two-dimensional structures that are difficult to handle within conventional named entity recognition or relation extraction frameworks, can be represented by extending sequence-to-sequence (seq2seq) models (Lewis et al., 2020). They also identified several challenges in table generation, including the generation of multiple tables, schema uncertainty, and sparse information extraction.

End-to-End Table Generation Wu et al. (2022) initially proposed a seq2seq model that treats a table as a sequence of tokens. However, this approach struggled to capture the two-dimensional dependencies inherent in tables, making it difficult to maintain structural consistency. In response to this, STable (Pietruszka et al., 2024) generates tables using permutation-based decoding, which does not rely on a fixed cell generation order and accounts for two-dimensional structure.

Multi-Step Table Generation Recent methods leverage the reasoning capabilities of LLMs to perform the generation process in stages. Text-Tuple-Table (Deng et al., 2024) extracts event tuples from texts and then integrates them into a tabular format. This approach improved performance in tasks that require not only extraction but also the aggregation of information. Similarly, gTBLS (Sundar et al., 2024) divides the generation process into two steps: schema extraction and question answering (QA). It improved structural consistency by first identifying row and column headers and then filling each cell in a QA format based on that structure.

Map&Make Map&Make (Ahuja et al., 2025) is one of the multi-step table generation methods. It first decomposes the input text into a sequence of atomic facts and extracts the elements necessary for tabulation. It then sequentially generates the schema, such as row and column headers, based on the extracted information. Finally, it processes each statement according to the generated schema and updates the corresponding cells to complete the table. Map&Make outperforms existing prompting methods by decoupling schema generation from cell value generation, which allows for the flexible generation of tables even with complex structures.

Nevertheless, Map&Make does not explicitly guarantee the structural consistency. In addition, since the generation process in Map&Make is sequential, if irrelevant information is mistakenly extracted as part of the schema in the early step, the table structures and cell values are susceptible to errors. Consequently, tables generated by Map&Make still suffer from errors such as the addition of redundant columns, missing values, and over-/under-estimated numerical values.

2.2 Minimum Bayes Risk (MBR) Decoding

MBR decoding (Kumar and Byrne, 2004; Eikema and Aziz, 2020) generates high-quality text by maximizing the expected utility of output hypotheses.

Let $\mathcal{X}$ and $\mathcal{Y}$ denote the input and output spaces, respectively. Conventional maximum a posteriori (MAP) decoding, which is the most widely used method, outputs a hypothesis $y_{\text{MAP}} \in \mathcal{Y}$ that maximizes the output probability given an input text $x \in \mathcal{X}$. Since searching the entire output space $\mathcal{Y}$ is infeasible, the output is selected from a finite set of output hypotheses $\mathcal{H} \subset \mathcal{Y}$ obtained by beam search or sampling:

$$y_{\text{MAP}} = \underset{y \in \mathcal{H}}{\operatorname{argmax}}\, p(y|x;\theta), \tag{1}$$

where θ denotes parameters of a text generation model. MBR decoding extends this hypothesis selection rule to maximize the expected utility rather than the output probability, as follows:

$$y_{\text{MBR}} = \underset{h \in \mathcal{H}}{\operatorname{argmax}}\, \mathbb{E}_{y \sim p(\cdot|x;\theta)} \left[u(h, y) \right], \tag{2}$$

where $u\colon \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}$ denotes the utility function. It evaluates the utility of a hypothesis h given a reference output y, and satisfies $h \succeq_y h' \iff u(h, y) \geq u(h', y)$ where $\succeq_y$ denotes the preference relation under the reference y. Since calculating the expectation over the entire output space $\mathcal{Y}$ is computationally intractable, MBR decoding typically estimates the expected utility using a Monte Carlo approximation with a finite multiset of pseudo-references $\mathcal{R} = \{r_1, \ldots, r_{|\mathcal{R}|}\}$ sampled

from $p(\cdot|x;\theta)$, as follows:

$$y_{\mathrm{MBR}} \approx \operatorname*{argmax}_{h\in\mathcal{H}} \frac{1}{|\mathcal{R}|}\sum_{r\in\mathcal{R}} u(h,r). \quad (3)$$

Typically, the same set of sampled texts is used for both hypothesis set $\mathcal{H}$ and pseudo-reference set $\mathcal{R}$.

The application of MBR decoding has expanded to structured data generation tasks, such as code generation and Text-to-SQL (Shi et al., 2022; Yadegari et al., 2026). For code generation, MBR-EXEC (Shi et al., 2022) uses the agreement of execution results as a utility function rather than relying on surface-level string matching. Similarly, in the Text-to-SQL task, several methods have been proposed that incorporate consistency checks based on query execution results or explicitly integrate generation probabilities into scores (Yadegari et al., 2026). For generating high-quality structured data, the design of utility functions that account for domain-specific characteristics, e.g., executability and logical consistency, is important.

3 Proposed Method: TableMBR

We propose *TableMBR*, a generation method for maintaining structural consistency in table generation. As illustrated in Figure 1, TableMBR generates multiple hypothesis tables using an LLM and selects the final output table based on MBR decoding with the proposed utility function that evaluates the structural consistency. The generation process consists of the following steps:

1) **Hypothesis Generation**: Generate multiple hypothesis tables.

2) **Utility Evaluation**: Evaluate the structural consistency of each hypothesis table based on pseudo-reference tables.

3) **Hypothesis Selection**: Select the hypothesis table that maximizes the expected utility.

1) Hypothesis Generation We first generate multiple output hypotheses $\mathcal{H} = \{h_1, \ldots, h_n\}$ and pseudo-references $\mathcal{R} = \{r_1, \ldots, r_{|\mathcal{R}|}\}$ using Map&Make (Ahuja et al., 2025), where $n \in \mathbb{N}$ is the number of hypotheses. To diversify the hypotheses and pseudo-references, we use temperature sampling.Appendix A provides more details.

2) Utility Evaluation We then evaluate the F1-based utility of each hypothesis $h \in \mathcal{H}$ for each pseudo-reference $r \in \mathcal{R}$. We formulate the utility function as $u(h,r) = \mathrm{F1}^{\mathrm{sim}}(h,r)$ with a similarity function sim that measures the similarity between two elements in tables. To maintain the consistency between structures, defined by headers, and contents, $\mathrm{F1}^{\mathrm{sim}}$ is composed of F1 scores for each table components: row headers (row), column headers (col), and cell values (cell), i.e., $\mathrm{F1}^{\mathrm{sim}}(h,r) = \frac{1}{3}\left(\mathrm{F1}^{\mathrm{sim}}_{\mathrm{row}}(h,r) + \mathrm{F1}^{\mathrm{sim}}_{\mathrm{col}}(h,r) + \mathrm{F1}^{\mathrm{sim}}_{\mathrm{cell}}(h,r)\right)$. For each component $\ell \in \{\mathrm{row}, \mathrm{col}, \mathrm{cell}\}$, the F1 score $\mathrm{F1}^{\mathrm{sim}}_{\ell}(h,r)$ is calculated from the precision (Prec) and recall (Recall) as usual:

$$\mathrm{F1}^{\mathrm{sim}}_{\ell}(h,r) = \frac{2\mathrm{Prec}^{\mathrm{sim}}_{\ell}(h,r)\mathrm{Recall}^{\mathrm{sim}}_{\ell}(h,r)}{\mathrm{Prec}^{\mathrm{sim}}_{\ell}(h,r) + \mathrm{Recall}^{\mathrm{sim}}_{\ell}(h,r)}. \quad (4)$$

We evaluate $\mathrm{Prec}^{\mathrm{sim}}_{\ell}$ and $\mathrm{Recall}^{\mathrm{sim}}_{\ell}$ in the element unit for each table component by treating each component as a set of the element values. Let $\mathcal{E}$ be the element space, $\mathcal{E}_\ell \subseteq \mathcal{E}$ be the element subspace of the table component $\ell \in \{\mathrm{row}, \mathrm{col}, \mathrm{cell}\}$, and $\phi_\ell \colon \mathcal{Y} \to 2^{\mathcal{E}_\ell}$ be a function that extracts elements from a given table, e.g., $\phi_{\mathrm{col}}(y) = \{\text{“name”}, \text{“age”}, \ldots\}$. For cell, since a set consisting solely of cell values cannot identify which row and column the cells belong to, e.g., $\{20, 30, 5.0, \ldots\}$ does not contain the positional information, we include the two-dimensional positional information in a cell element representation using its corresponding row and column headers. For example, $\phi_{\mathrm{cell}}(y) = \{(\text{“Bob”}, \text{“age”}, 20), \ldots\}$. We calculate $\mathrm{Prec}^{\mathrm{sim}}_{\ell} \colon \mathcal{Y} \times \mathcal{Y} \to [0,1]$ and $\mathrm{Recall}^{\mathrm{sim}}_{\ell} \colon \mathcal{Y} \times \mathcal{Y} \to [0,1]$ by aligning the most similar element in the corresponding set using the similarity function sim:

$$\mathrm{Prec}^{\mathrm{sim}}_{\ell}(h,r) = \frac{1}{|\phi_\ell(h)|}\sum_{e_h\in\phi_\ell(h)} \max_{e_r\in\phi_\ell(r)} \mathrm{sim}(e_h, e_r), \quad (5)$$

$$\mathrm{Recall}^{\mathrm{sim}}_{\ell}(h,r) = \frac{1}{|\phi_\ell(r)|}\sum_{e_r\in\phi_\ell(r)} \max_{e_h\in\phi_\ell(h)} \mathrm{sim}(e_h, e_r). \quad (6)$$

where $\mathrm{sim} \colon \mathcal{E} \times \mathcal{E} \to [0,1]$ denotes the similarity function. For cell, we calculate the similarity between two cells as the product of the individual similarities of their sub-elements.

3) Hypothesis Selection Finally, we select the hypothesis that maximizes the expected utility estimated using pseudo-references:

$$\operatorname*{argmax}_{h\in\mathcal{H}} \frac{1}{|\mathcal{R}|}\sum_{r\in\mathcal{R}} u(h,r). \quad (7)$$

4 Experiments

We evaluate the effectiveness of TableMBR from two perspectives, structural consistency and accuracy of cell value prediction. The former, structural consistency in dynamic schema generation, evaluates how well over-/under-generation is mitigated, especially when the input text contains irrelevant information. The latter, content fidelity in a fixed-schema setting, evaluates the accuracy of cell value prediction in a setting where the schema is given.

4.1 Experimental Setup

We generate 32 hypotheses for each input, i.e., $n = 32$, using GPT-4.1 mini (OpenAI et al., 2024) and Gemini-2.5 Flash (Comanici et al., 2025). For both models, we do not perform any additional fine-tuning and only call the inference through their respective APIs. We use the same set of output candidates for the hypotheses and pseudo-references. We compare our method with the output of a single generation, i.e., $n = 1$, as the baseline for comparison, which is equivalent to the results of Map&Make (Ahuja et al., 2025). To control the balance between the diversity and quality of the generated outputs from the LLMs in our experiments, the following decoding parameters for hypothesis generation were used: temperature: 1.0, top-k: 40, and top-p: 0.95.

We also evaluate the influence of the number of hypotheses n. Specifically, we examine the extent to which applying TableMBR is effective compared with single generation in the text-to-table task, as well as how an increase in n contributes to performance. We vary the number of hypotheses $n \in \{1, 4, 8, 16, 32\}$. Furthermore, to ensure the reliability of any observed performance differences across these settings, we conduct a bootstrap significance test (Koehn, 2004) with $p < 0.05$.

4.2 Structural Consistency in Dynamic Schema Generation

Dataset We evaluate the structural consistency in the Rotowire dataset (Wiseman et al., 2017), which pairs NBA game summaries with two types of structured data: player tables and team tables. A characteristic of this dataset is that the input text contains numerous descriptions that are considered noise for the table generation task, e.g., qualitative commentary on game developments and player emotions. Hence, we can evaluate two critical aspects of structural consistency: factual consistency, the ability to select only the facts necessary for tabulation, and schema consistency, the ability to suppress the generation of redundant or incorrect columns. We use the test data manually corrected by Ahuja et al. (2025).

Evaluation Metrics For the evaluation metrics and utility functions of the structural consistency, we use F1 scores with three similarity functions: exact match (EM), CHRF (Popović, 2015), and BERTScore (BERT) (Zhang et al., 2019). For calculating BERTScore, we use `google-bert/bert-base-uncased` (Devlin et al., 2019). Since row and column headers in Rotowire consist of descriptive text such as player names or statistical categories, we also use ChrF and BERTScore to allow for soft matching of semantically similar expressions, in addition to EM.

Results Table 1 shows the experimental results on the Rotowire dataset. TableMBR achieved statistically significant gains in the player table, as indicated by †. For instance, when we used $\mathrm{F1}^{\mathrm{EM}}$ for the utility function, Gemini-2.5 Flash and GPT-4.1 mini achieved 3.44 and 4.90 points of improvement in $\mathrm{F1}^{\mathrm{BERT}}_{\mathrm{cell}}$, respectively. Conversely, in the team table, where the baseline scores were extremely low, the effects of MBR decoding were inconsistent. While some configurations showed slight improvements, others fell below the baseline performance, suggesting that the benefits of TableMBR are limited when the hypothesis quality is extremely low.

We also confirmed a clear trend where the performance of TableMBR generally improves as the number of hypotheses n increases, as shown in Figure 2. By comparing the results for $n = 32$ with those for smaller hypothesis sets, i.e., $n < 32$, we observed a consistent upward trend, demonstrating that increasing the number of hypotheses directly contributes to the final scores across all table types and components, including row headers, column headers, and structured cells.

4.3 Content Fidelity in Fixed-Schema Setting

Dataset In this setting, we use the LiveSum dataset (Deng et al., 2024). LiveSum focuses on accurately generating cell values given a fixed and predefined schema. We evaluate the accuracy of cell value prediction for each difficulty level classified in the original dataset. The dataset details are provided in Appendix B.

Method	Utility	Row Header			Column Header			Structured Cells		
		$F1^{EM}_{row}$	$F1^{CHRF}_{row}$	$F1^{BERT}_{row}$	$F1^{EM}_{col}$	$F1^{CHRF}_{col}$	$F1^{BERT}_{col}$	$F1^{EM}_{cell}$	$F1^{CHRF}_{cell}$	$F1^{BERT}_{cell}$
Gemini-2.5 Flash - Player Table										
Map&Make	–	85.27	89.44	89.38	24.35	50.46	66.56	28.31	46.28	57.56
TableMBR	$F1^{EM}$	85.55† (+0.28)	89.58† (+0.14)	89.61† (+0.23)	**27.31† (+2.96)**	**53.30† (+2.84)**	**70.51† (+3.95)**	**30.26† (+1.95)**	**48.98† (+2.70)**	**61.00† (+3.44)**
	$F1^{CHRF}$	**85.58† (+0.31)**	**89.60† (+0.16)**	**89.64† (+0.26)**	24.41 (+0.06)	51.63† (+1.17)	67.13† (+0.57)	27.85 (-0.46)	47.68† (+1.40)	58.13 (+0.57)
	$F1^{BERT}$	85.56† (+0.29)	89.59† (+0.15)	89.62† (+0.24)	25.91† (+1.56)	52.34† (+1.88)	69.11† (+2.55)	29.41† (+1.10)	48.48† (+2.20)	59.97† (+2.41)
Gemini-2.5 Flash - Team Table										
Map&Make	–	9.56	60.95	**30.71**	8.96	33.72	40.32	**0.94**	16.05	9.88
TableMBR	$F1^{EM}$	**9.68 (+0.12)**	62.81† (+1.86)	28.70 (-2.01)	**10.56† (+1.60)**	**34.65† (+0.93)**	**42.48† (+2.16)**	0.80 (-0.14)	**16.95† (+0.90)**	**10.36 (+0.48)**
	$F1^{CHRF}$	8.89 (-0.67)	**62.83† (+1.88)**	27.63 (-3.08)	6.97 (-1.99)	33.30 (-0.42)	37.38 (-2.94)	0.69 (-0.25)	15.78 (-0.27)	9.03 (-0.85)
	$F1^{BERT}$	9.61 (+0.05)	62.71† (+1.76)	28.60 (-2.11)	9.80† (+0.84)	34.42† (+0.70)	41.02† (+0.70)	0.75 (-0.19)	16.69† (+0.64)	10.00 (+0.12)
GPT-4.1 mini - Player Table										
Map&Make	–	85.19	89.81	89.51	31.52	57.28	70.50	29.32	49.06	58.24
TableMBR	$F1^{EM}$	87.08† (+1.89)	90.84† (+1.03)	90.90† (+1.39)	**34.43† (+2.91)**	**59.98† (+2.70)**	**74.37† (+3.87)**	**33.68† (+4.36)**	**53.04† (+3.98)**	**63.14† (+4.90)**
	$F1^{CHRF}$	87.02† (+1.83)	**90.89† (+1.08)**	**90.91† (+1.40)**	32.70† (+1.18)	59.35† (+2.07)	71.93† (+1.43)	31.92† (+2.60)	52.77† (+3.71)	61.06† (+2.82)
	$F1^{BERT}$	**87.11† (+1.92)**	90.87† (+1.06)	**90.91† (+1.40)**	34.12† (+2.60)	59.60† (+2.32)	73.95† (+3.45)	33.62† (+4.30)	53.02† (+3.96)	62.91† (+4.67)
GPT-4.1 mini - Team Table										
Map&Make	–	6.56	59.82	**27.17**	**5.68**	32.29	**38.15**	**1.49**	15.90	**9.33**
TableMBR	$F1^{EM}$	**7.79† (+1.23)**	62.08† (+2.26)	26.54 (-0.63)	4.50 (-1.18)	33.74† (+1.45)	37.66 (-0.49)	0.68 (-0.81)	**17.49† (+1.59)**	8.69 (-0.64)
	$F1^{CHRF}$	6.40 (-0.16)	61.94† (+2.12)	24.46 (-2.71)	2.95 (-2.73)	33.68† (+1.39)	34.91 (-3.24)	0.65 (-0.84)	17.43† (+1.53)	7.63 (-1.70)
	$F1^{BERT}$	7.64† (+1.08)	61.96† (+2.14)	26.43 (-0.74)	4.58 (-1.10)	**33.95† (+1.66)**	37.38 (-0.77)	0.48 (-1.01)	17.34† (+1.44)	8.57 (-0.76)

Table 1: F1 scores of each table component in Rotowire dataset. Values in parentheses "(·)" denote improvements over Map&Make. **Bold** scores indicate best score within each evaluation metric. † indicates statistical significance ($p < 0.05$).

Model	n	Easy	Medium	Hard	Avg.
Gemini-2.5 Flash	1	98.67	79.82	60.48	79.66
	4	98.87	85.38†	68.20†	84.15†
	8	98.97	86.75†	71.88†	85.87†
	16	**99.07**†	87.33†	73.97†	86.79†
	32	99.04†	**87.50**†	**74.47**†	**87.00**†
GPT-4.1 mini	1	97.61	48.74	13.83	53.39
	4	97.94	48.49	13.79	53.41
	8	98.31†	49.54	15.25†	54.37†
	16	**98.41**†	49.75†	**15.45**†	**54.54**†
	32	98.18†	**50.03**†	14.59	54.27†

Table 2: Accuracy of cell values when using $F1^{EM}$ for utility function on LiveSum dataset. **Bold** scores indicate the best scores within the model in each difficulty. † indicates statistical significance ($p < 0.05$).

Evaluation Metrics We use exact match as the evaluation metric for cell value prediction because all the cell values in LiveSum are numerical counts, and other metrics based on semantic similarity are unsuitable for this task.

Results The experimental results are shown in Table 2. We observed that TableMBR outperformed the baseline in accuracy across all models. Most of these improvements were found to be statistically significant ($p < 0.05$). Notably, for Gemini-2.5 Flash in the "Hard" difficulty setting, there was a substantial improvement of nearly 14 points, with the score increasing from 60.48 for $n = 1$ to 74.47 for $n = 32$. Regarding the trends by difficulty level, the accuracy for "Easy" tasks had already reached a high level of over 97% at the baseline; nevertheless, the proposed method yielded slight further gains. Conversely, for tasks requiring complex aggregation such as "Medium" and "Hard" levels, the improvement margin provided by the proposed method becomes more pronounced. As for the impact of increasing n, Gemini-2.5 Flash, which has a relatively high baseline score, shows monotonic improvement as n increases. In contrast, for GPT-4.1 mini, which has a relatively lower baseline, the performance peaked at $n = 16$ and subsequently declined at $n = 32$.

5 Discussion

5.1 Baseline Performance and Convergence to False Consensus

In Rotowire, while MBR decoding yielded significant improvements for the player table, the gains for the team table were marginal or, in some configurations, even showed a downward trend. This suggests a prerequisite for the effectiveness of the approach based on "selecting consistency through consensus among multiple hypotheses." Theoretically, MBR decoding ensures reliability by suppressing outliers, based on the assumption that the set of hypotheses covers the proximity of the ground truth to some extent (Ohashi et al., 2024). When the baseline performance is above a certain threshold, as seen in the player table, hypotheses close to the correct answer form a majority.

However, in scenarios where the baseline is extremely low (around a few percent), such as in the

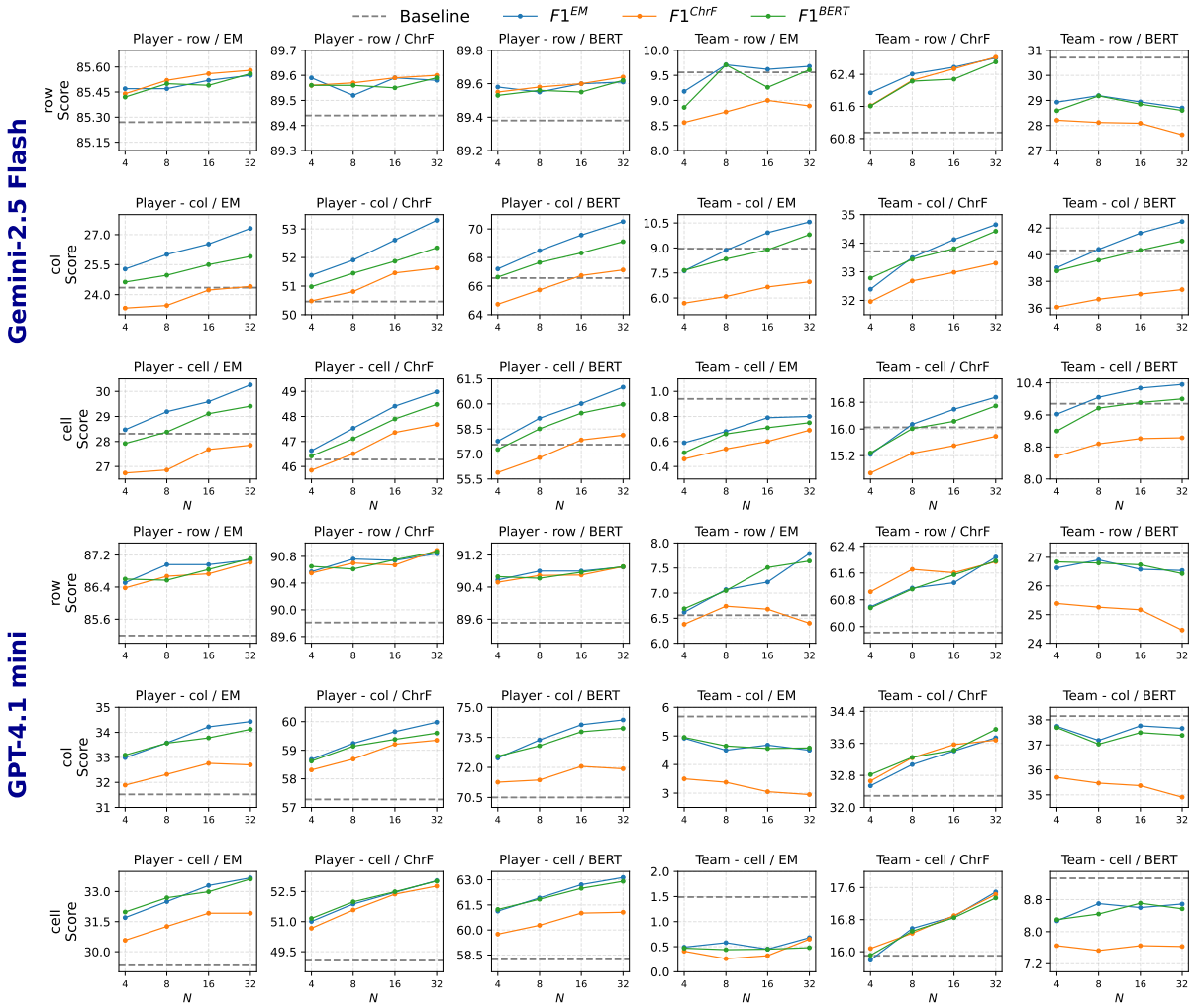

Figure 2: F1 scores when varying number of hypotheses n in Rotowire

team table, the set of hypotheses contains almost no correct answers. In such cases, specific hallucinations generated confidently by the model or frequently occurring erroneous patterns become dominant. Consequently, MBR decoding may select this "false consensus" among incorrect hypotheses. This suggests that the application of MBR decoding in text-to-table tasks requires the single-generation model to guarantee a minimum level of quality (the probability of including the correct answer). In tasks that fall below this threshold, there is an inherent risk that the method may reinforce systematic errors.

5.2 Improved Robustness in both Schema Generation and Numerical Aggregation

The experimental results using LiveSum clarify that the selection mechanism based on structural consensus through MBR decoding significantly contributes to improved numerical accuracy, representing "content-level consistency." Live commentary text contains substantial noise that can easily be confused with actual events, such as "a near miss" or "just over the goal." In single generation (Baseline), the model tends to react oversensitively to these descriptions, leading to "overcounting" where predicted values deviate significantly from the actual values. In contrast, the proposed TableMBR effectively filters out sporadic hallucinations (outliers) that appear only in specific hypotheses through consensus building, converging toward numerical values with higher certainty.

This correction effect was more pronounced in the "Hard" sections, where model output is typically unstable, compared to the "Easy" sections, where the baseline already shows high performance. This fact suggests that the proposed method functions as a robust framework for ensuring information reliability in situations where the model's inference uncertainty is high. The series of experimen-

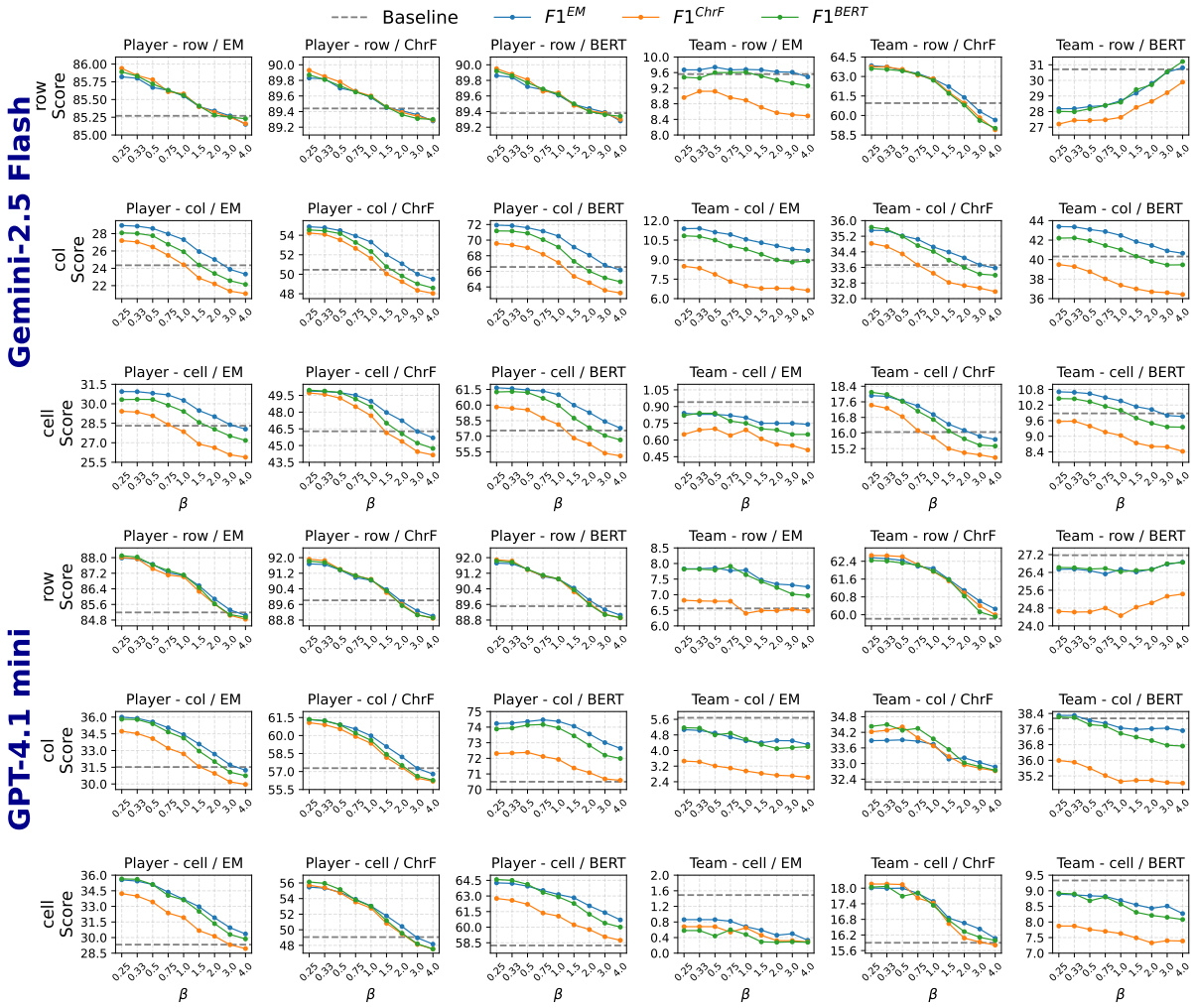

Figure 3: F1 scores in Rotowire when we use Fβ for utility function instead of F1.

tal results in this study indicate that the proposed framework of consistency optimization via MBR decoding is effective for both "structural errors" (redundant column generation) in Rotowire and "content errors" (numerical discrepancies) in LiveSum. This substantiates that the proposed method possesses high generalizability for improving reliability across the text-to-table task in general, regardless of task definitions such as dynamic schema generation versus fixed settings or differences in domains.

5.3 Impact of Hypothesis Number and Computational Efficiency

As shown in Figure 2 and Table 2, we observed a general trend in which performance improves as the number of hypotheses n increases. However, it is difficult to define a universally optimal value of n for all tasks. Our results on LiveSum indicate that performance gains begin to saturate or even slightly decline for some models around $n = 16$. A similar, though less pronounced, saturation trend can be observed in Rotowire. The computational cost for hyphothesis generation scales linearly with the number of hypotheses n. Therefore, simply maximizing n is not always desirable in practical applications. Based on our empirical observations, we consider $n = 16$ to be a practical value that effectively balances substantial performance gains with computational efficiency across different datasets and models.

5.4 Balancing Weight Between Precision and Recall in Utility Function

We analyze the effect of the balance between precision and recall by substituting F1 with Fβ in the utility function. Specifically, we vary $\beta \in \{0.25, 0.33, 0.5, 0.75, 1.0, 1.5, 2.0, 3.0, 4.0\}$ in Rotowire. Note that $\beta < 1$ corresponds to a precision-oriented setting and $\beta > 1$ corresponds to a recall-

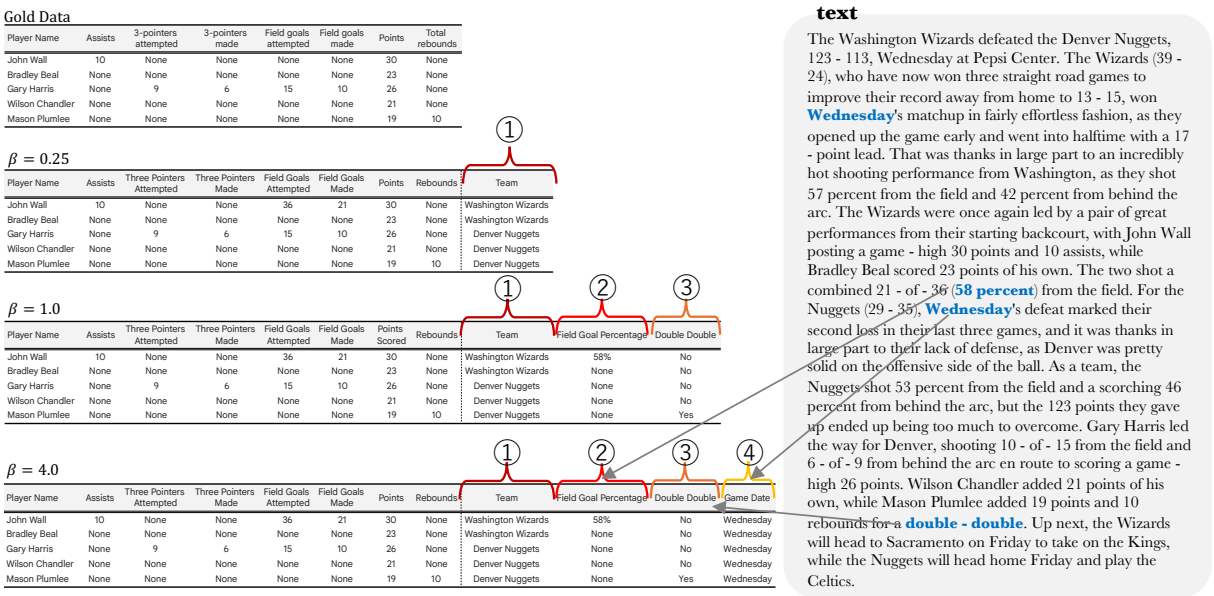

Figure 4: Comparisons of generation examples on Rotowire dataset. Upper row presents gold data, while lower row displays selection results obtained by varying parameter β. As shown, prioritizing recall tends to increase number of extraneous columns that reflect non-statistical information from input text, such as game dates and qualitative descriptions. In contrast, precision-oriented setting suppresses such excessive column generation, resulting in selection of compact schema that closely resembles gold data.

oriented setting.

As illustrated in Figure 3, across both models and table types, we confirmed that a smaller β, i.e., a precision-oriented setting, yields higher F1 scores. For instance, while the $\mathrm{F1}_{\mathrm{cell}}^{\mathrm{BERT}}$ score was 63.14 in the results for GPT-4.1 mini using $\mathrm{F}\beta^{\mathrm{EM}}$, the score reached 64.26 at $\beta = 0.25$. Conversely, the score dropped to 60.70 at $\beta = 4.0$. This highlights that, in table generation, the exclusion of errors through a precision-oriented utility function contributes more directly. While recall is often emphasized in general text generation tasks such as summarization, the inclusion of even a single incorrect row or column in structured data significantly undermines the reliability of the entire dataset.

This tendency is also qualitatively evident in the relationship between the descriptive characteristics of the Rotowire dataset and the generation process of Map&Make, as shown in Figure 4. The primary strength of Map&Make lies in its ability to dynamically induce a schema from input text without requiring a predefined structure. However, a side effect of this high versatility is its tendency to pick up various pieces of information within the text as unnecessary columns, resulting in overgeneration. In fact, the input text in Rotowire contains many descriptions that should not be tabulated, such as "Wednesday's matchup" or "double-double." Because Map&Make extracts these elements equally, it frequently generates unnecessary columns or hallucinated headers, such as "Game Date" or "Double Double," as shown in the bottom row of Figure 4.

In contrast, the results under the precision-oriented setting with $\beta = 0.25$, shown in the second row of Figure 4, successfully retained only the appropriate columns while suppressing irrelevant ones. This outcome occurs because precision-oriented TableMBR applies a strong penalty to hypotheses containing redundant cells that do not reach a consensus across multiple hypotheses. Thus, TableMBR successfully maintains the versatility of Map&Make while filtering out excessive noise from the perspective of structural consistency. We conclude that a precision-oriented utility function prioritizing the exclusion of misinformation is essential for structured data generation and serves as a key to enhancing the practicality of versatile prompting methods.

6 Conclusion

We proposed TableMBR, a text-to-table generation method that explicitly maintains structural consistency. Specifically, TableMBR selects the table that maximizes the expected utility based on matches of table rows, columns, and structured cells among multiple hypotheses. The proposed TableMBR

achieved performance improvements in two tasks with different settings, dynamic schema generation and fixed-schema settings. Specifically, we observed relative improvements of up to 15% in F1 on Rotowire and 23% in accuracy on LiveSum. In addition, we clarified that a precision-oriented configuration in the design of the utility function is particularly effective for structured data generation. By prioritizing precision, TableMBR effectively suppresses structural errors and redundant information that lacks solid consensus across multiple hypotheses, ensuring highly robust generation even for noisy input text.

For future work, we will investigate the generalizability of TableMBR across other domains, particularly medical and legal domains, where high reliability is crucial, aiming to reduce critical errors in table generation.

Limitations

Computational Cost MBR decoding increases the computational cost and inference time compared to MAP decoding. The associated API costs and latency are n times higher than those of the baseline, where $n \in \mathbb{N}$ is the number of hypotheses. As discussed in Section 5.3, this trade-off can be managed by selecting a moderate value of n to maintain practical efficiency.

Domain and Task Specificity Our current evaluation is focused on the sports domain through the Rotowire and LiveSum datasets. Further investigation is needed to verify the generalizability of the proposed structural utility function in other specialized domains such as medical or legal document processing.

Ethical Considerations

Dependence on Base Model Quality As discussed in Section 5.1, the effectiveness of TableMBR is dependent on the quality of the underlying LLM. When the baseline performance is extremely low, the hypothesis set may lack any correct answers, potentially leading to a "false consensus" where the model selects a consistently generated but incorrect hallucination.

Use of Benchmark Datasets In this study, we utilize the existing benchmark datasets Rotowire and LiveSum. These datasets are used in accordance with the terms of use provided by the original authors. We have confirmed that Rotowire and LiveSum are based on publicly available sports game results and live commentaries and do not contain highly sensitive personal information.

Use of AI Assistants. We used GitHub Copilot for coding, and all generated code was verified by the authors. For writing support, including grammatical checking and translation, we used DeepL, Grammarly, and ChatGPT. All original content was written and verified by the authors.

Licenses. The Rotowire and LiveSum datasets can be used for research purposes as described in the original papers (Wiseman et al., 2017; Ahuja et al., 2025; Deng et al., 2024). For LLM, Access to the Gemini-2.5 Flash (Comanici et al., 2025) and GPT-4.1 mini (OpenAI et al., 2024) models was conducted through the official Google and OpenAI APIs, respectively, in compliance with their terms of service. We also used mbrs (Deguchi et al., 2024), an MIT-licensed tool. Therefore, we confirmed that all datasets, models, and tools used in this research comply with their respective licenses and terms of use, and we identified no licensing issues. Therefore, we confirmed that all datasets, models, and tools used in this research comply with their respective licenses and terms of use, and identified no licensing issues.

References

Naman Ahuja, Fenil Bardoliya, Chitta Baral, and Vivek Gupta. 2025. Map&make: Schema guided text to table generation. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 30249–30262, Vienna, Austria. Association for Computational Linguistics.

Gheorghe Comanici, Eric Bieber, Mike Schaekermann, Ice Pasupat, Noveen Sachdeva, Inderjit Dhillon, Marcel Blistein, Ori Ram, Dan Zhang, Evan Rosen, Luke Marris, Sam Petulla, Colin Gaffney, Asaf Aharoni, Nathan Lintz, Tiago Cardal Pais, Henrik Jacobsson, Idan Szpektor, Nan-Jiang Jiang, and 3416 others. 2025. Gemini 2.5: Pushing the frontier with advanced reasoning, multimodality, long context, and next generation agentic capabilities. *Preprint*, arXiv:2507.06261.

Hiroyuki Deguchi, Yusuke Sakai, Hidetaka Kamigaito, and Taro Watanabe. 2024. mbrs: A library for minimum Bayes risk decoding. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 351–362, Miami, Florida, USA. Association for Computational Linguistics.

Zheye Deng, Chunkit Chan, Weiqi Wang, Yuxi Sun, Wei Fan, Tianshi Zheng, Yauwai Yim, and Yangqiu Song. 2024. Text-tuple-table: Towards information integration in text-to-table generation via global tuple extraction. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 9300–9322, Miami, Florida, USA. Association for Computational Linguistics.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. 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 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Bryan Eikema and Wilker Aziz. 2020. Is MAP decoding all you need? the inadequacy of the mode in neural machine translation. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 4506–4520, Barcelona, Spain (Online). International Committee on Computational Linguistics.

Philipp Koehn. 2004. Statistical significance tests for machine translation evaluation. In *Proceedings of the 2004 Conference on Empirical Methods in Natural Language Processing*, pages 388–395, Barcelona, Spain. Association for Computational Linguistics.

Shankar Kumar and William Byrne. 2004. Minimum Bayes-risk decoding for statistical machine translation. In *Proceedings of the Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics: HLT-NAACL 2004*, pages 169–176, Boston, Massachusetts, USA. Association for Computational Linguistics.

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.

Atsumoto Ohashi, Ukyo Honda, Tetsuro Morimura, and Yuu Jinnai. 2024. On the true distribution approximation of minimum Bayes-risk decoding. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 2: Short Papers)*, pages 459–468, Mexico City, Mexico. Association for Computational Linguistics.

OpenAI, Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, Red Avila, Igor Babuschkin, Suchir Balaji, Valerie Balcom, Paul Baltescu, Haiming Bao, Mohammad Bavarian, Jeff Belgum, and 262 others. 2024. Gpt-4 technical report. *Preprint*, arXiv:2303.08774.

Michał Pietruszka, Michał Turski, Łukasz Borchmann, Tomasz Dwojak, Gabriela Nowakowska, Karolina Szyndler, Dawid Jurkiewicz, and Łukasz Garncarek. 2024. STable: Table generation framework for encoder-decoder models. In *Proceedings of the 18th Conference of the European Chapter of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2454–2472, St. Julian's, Malta. Association for Computational Linguistics.

Maja Popović. 2015. chrF: character n-gram F-score for automatic MT evaluation. In *Proceedings of the Tenth Workshop on Statistical Machine Translation*, pages 392–395, Lisbon, Portugal. Association for Computational Linguistics.

Freda Shi, Daniel Fried, Marjan Ghazvininejad, Luke Zettlemoyer, and Sida I. Wang. 2022. Natural language to code translation with execution. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 3533–3546, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Anirudh S. Sundar, Christopher Richardson, and Larry Heck. 2024. gtbls: Generating tables from text by conditional question answering. *ArXiv*, abs/2403.14457.

Sam Wiseman, Stuart Shieber, and Alexander Rush. 2017. Challenges in data-to-document generation. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2253–2263, Copenhagen, Denmark. Association for Computational Linguistics.

Xueqing Wu, Jiacheng Zhang, and Hang Li. 2022. 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)*, pages 2518–2533, Dublin, Ireland. Association for Computational Linguistics.

Mostafa Yadegari, Yuqiao Wen, Davood Rafiei, and Lili Mou. 2026. Exploring minimum bayes risk decoding for text-to-SQL ensemble.

Tianyi Zhang, Varsha Kishore, Felix Wu, Kilian Q. Weinberger, and Yoav Artzi. 2019. Bertscore: Evaluating text generation with bert. *ArXiv*, abs/1904.09675.

A Prompts and Generation Process

In these experiments, we generate hypothesis tables by applying the three-stage generation framework proposed in the Map&Make framework (Ahuja et al., 2025). Although Figure 5 through Figure 10 illustrate the specific prompts for each individual task (propositional atomization, schema extraction, and table generation), we adopted the **M&M-U**

(Unified) variant for the actual execution. Under this approach, the prompts for each stage are integrated into a single query, enabling the model to complete the entire transformation from unstructured text to a structured table within a single LLM call. The specific components of these integrated prompts for the Rotowire and LiveSum datasets correspond to the contents shown in Figure 5–Figure 7 and Figure 8–Figure 10, respectively.

B Overview of LiveSum

LiveSum is a dataset designed to evaluate information aggregation capabilities, consisting of live commentary text for 3,771 English Premier League (EPL) matches from 2014 to 2023. Each live commentary segment spans approximately 1,256 words on average. Models are required to extract and aggregate eight specific types of event information from these long, unstructured texts to generate a summary table.

Task Characteristics In contrast to Rotowire, which emphasizes "structure generation" where schemas are dynamically induced from input text, LiveSum focuses on "value generation". This primarily aims at the accurate aggregation of event counts from time-series text based on a predefined, fixed schema. In soccer live commentary, a single event may be mentioned across multiple commentaries (e.g., "Goal!" and "Player scores" referring to the same event), while conversely, similar expressions may refer to different events. Therefore, this task requires advanced information aggregation and reasoning capabilities rather than simple extraction.

Difficulty Classification Following the classification criteria established by Deng et al. (2024), the event categories in the LiveSum dataset are divided into three difficulty levels based on the diversity of their linguistic expressions and the resulting complexity of information aggregation. We used these predefined levels to analyze the performance of TableMBR across varied degrees of task difficulty.

- **Easy (Low Difficulty)**: Includes "Goals" and "Red Cards". Goals are often explicitly stated in the text, and red cards occur infrequently, making their identification relatively straightforward.
- **Medium (Moderate Difficulty)**: Includes four types: "Yellow Cards", "Corner Kicks", "Free Kicks", and "Offsides".
- **Hard (High Difficulty)**: Includes "Shots" and "Fouls". These have extremely diverse variations in expression and description within the commentary, making accurate identification and aggregation the most challenging.

Prompt with Map&Make for Rotowire (Task1: Propositional Atomization)

You are an expert at converting unstructured, detailed textual inputs into highly structured tables.
You are given 3 tasks to perform sequentially.
Perform TASK 1, then, use the output of TASK 1 to perform TASK 2. Finally, use the outputs of TASK 1 and TASK 2 to perform TASK 3. After completing all three tasks, present the outputs in the order of TASK 1, TASK 2, and TASK 3, separated clearly.

*****TASK 1***:**
Decompose the given paragraphs or sentences into clear, self-contained, and highly detailed short atomic statements without losing any information. Each atomic statement should capture a single fact or action with maximum granularity.

*****INSTRUCTIONS for TASK 1***:**
Capture only information explicitly stated in the input text.
No detail should be assumed, inferred, or added that is not present in the text.
Each atomic statement should contain only one key entity and one action or attribute.
If a sentence contains multiple pieces of information, decompose it further into more granular statements.
Eliminate ambiguity by resolving pronouns and ensuring each statement stands alone.
Preserve necessary context so each statement is meaningful on its own.
Represent numerical data and units exactly as given in the input text.
Ensure each statement conveys unique information without overlapping others.
Ensure statements are clear, direct, and free from unnecessary complexity.
DO NOT infer or calculate additional data points, such as missed shots, unless explicitly stated in the text.
Resolve pronouns to their corresponding nouns for clarity.
Maintain relationships between entities without combining multiple facts.

*****OUTPUT FORMAT for TASK 1***:**
<ATOMIC STATEMENT 1>
<ATOMIC STATEMENT 2>
<ATOMIC STATEMENT 3>
...

*****FINAL CHECKLIST for TASK 1***:**
All information from the input text is included.
No information or calculation is added that is not present in the text.
Every fact and detail is accurately represented.
Statements are clear and can be understood independently.
Numerical data and units are formatted exactly as provided in the text.
Each statement directly reflects the input text without inferred details.
Pronouns are resolved; statements are unambiguous.
Each statement contains only one key entity and one action or attribute.
Do not number the statements or add extra formatting.

Figure 5: Prompt for Propositional Atomization (for Rotowire)

Prompt with Map&Make for Rotowire (Task2: Schema Extraction)

*****TASK 2***:**
Given a set of atomic text statements, extract row and column headers to create a table schema.

*****INSTRUCTIONS for TASK 2***:**
Do not proceed without completing TASK 1.
Read the statements carefully to identify all attributes, entities, and data points mentioned, whether explicitly stated or implicitly implied.
Determine the row headers (primary keys) and column headers required to represent the data comprehensively and concisely:
Row headers are the unique identifiers for individual rows (key entities).
Column headers are the attributes of the primary keys that represent different aspects or data points.
Include all explicit and implicit data points, ensuring no relevant information is overlooked.
Pay close attention to numerical data, even if it is presented within comparative statements or descriptions of events or related to specific categories or time periods mentioned in the text.
Explicit numerical data must always be captured as attributes where appropriate.
Implicit data points or recurring attributes must also be included.
Avoid adding actions as column headers but extract any data points associated with them.
Ensure that all numerical values are captured as attributes, even if they are related to specific time periods or events within the context.
When encountering comparative statements or ratios like “X of Y”, ensure you capture both ’X’ and ’Y’ as potentially distinct and relevant data points if they represent different aspects of an attribute.
Be attentive to granular details and avoid focusing solely on general or aggregate values if more specific data points are available in the text.

*****OUTPUT FORMAT for TASK 2***:**
{
“<Table name>”: {
“row_headers”: [“Row Header 1”, ...],
“column_headers”: [“Column Header 1”, ...]
}
...
}

*****REASONING STEPS for TASK 2***:**
Step 1 - Identify the context from all the statements to generate a short description
Step 2 - Create a empty list of row and column headers for the tables. This list would be updated as we keep on processing the statements and will keep adding relevant column and row headers to the list.
Step 3 - Process statements one by one and add relevant headers if not already present in the list.

*****Output Instructions for TASK 2***:**
1. For *every given statement* return the updates done to the schema and generate the “Team” Table and “Player” Table schema.
2. Do not return schema directly in any case.

Figure 6: Prompt for Schema Extraction (for Rotowire)

Prompt with Map&Make for Rotowire (Task 3: Table Generation)

*****TASK 3***:**
Do not proceed without completing TASK 2.
Given:
Statements: A sequence of atomic statements from TASK 1.
Schema: A json object with table names and their row headers and column headers of the respective tables from TASK 2.
Your goal is to:
Process each statement one by one.
Identify the correct set of table, row and column headers and the cell at that index to update based on the statement.
Update or add values to the tables accordingly.

*****OUTPUT FORMAT for TASK 3***:**
Final Output Tables:
<Table name>
| | <Column Header 1> | ... |
| <Row Header 1> | <Cell Value for (Row Header 1, Column Header 1)> | ... |

*****REASONING STEPS for TASK 3***:**
Follow the given algorithm thoroughly,
ALGORITHM
For each statement in the input:
Identify Table: Determine the correct table to be updated based on the table.
Identify Row and Column: Determine which set of row and columns headers have to be updated based on this table.
Update the Table: If no value exists, update the value of the cell as per the statement.

*****FINAL CHECKLIST for TASK 3***:**
Follow these guidelines to generate tables and return the final state of the table after processing all the statements.
Ensure all sentences are processed and for every statement return the update and revised state of the updated cells as shown in the example.
Return the final table in the exact specified format starting with ### Final Table.
Do not generate the final table directly in any case.
No need to generate the intermediate table states, just return the final table at the end.
Ensure the table is concise, well-structured, and contains all information from the input.
The output should contain “Team” table and “Player” table.

*****Output Instructions for TASK 3***:**
1. Handle Missing Data: If a column value is not present in the statements, keep it as None.
2. Structural Integrity: Do not add or remove any rows or columns unless explicitly instructed by the data. Ensure uniformity in the format of data across the table.
3. Table formatting: Use “|” to separate cells

*****FINAL OUTPUT FORMAT***:**
<ATOMIC STATEMENT 1>
...
{ “<Table name>”: { “row_headers”: [...], “column_headers”: [...] } }
Final Output Tables:
<Table name>
| | <Column Header 1> | ... |
| <Row Header 1> | <Cell Value for (Row Header 1, Column Header 1)> | ... |

*****FINAL CHECKLIST***:**
Ensure that all the tasks are completed.
Ensure that the final output contains only the output of all the 3 tasks.
Provide all *OUTPUT* in the specified format only.

Figure 7: Prompt for Table Generation (for Rotowire)

Prompt with Map&Make for LiveSum (Task1: Propositional Atomization)

You are an expert at converting unstructured, detailed textual inputs into highly structured tables.
You are given 3 tasks to perform sequentially.
Perform TASK 1, then, use the output of TASK 1 to perform TASK 2. Finally, use the outputs of TASK 1 and TASK 2 to perform TASK 3. After completing all three tasks, present the outputs in the order of TASK 1, TASK 2, and TASK 3, separated clearly.

*****TASK 1***:**
Decompose the given football live commentary sentences into clear, self-contained, and highly detailed short atomic statements without losing any information. Each atomic statement should capture a single fact or action with maximum granularity.

*****INSTRUCTIONS for TASK 1***:**
Capture only information explicitly stated in the commentary.
No detail should be assumed, inferred, or added that is not present in the text.
Each atomic statement should contain only one key entity (team or player) and one action or attribute.
If a sentence contains multiple pieces of information, decompose it further into more granular statements.
Eliminate ambiguity by resolving pronouns and ensuring each statement stands alone.
Preserve necessary context so each statement is meaningful on its own.
Represent numerical data and units exactly as given in the text.
Ensure each statement conveys unique information without overlapping others.
Resolve pronouns to their corresponding nouns for clarity.
Each statement must correspond to one football event (goal, shot, foul, card, corner, free kick, offside, etc.).

*****OUTPUT FORMAT for TASK 1***:**
<ATOMIC STATEMENT 1>
<ATOMIC STATEMENT 2>
<ATOMIC STATEMENT 3>
...

*****FINAL CHECKLIST for TASK 1***:**
All information from the commentary is included.
No information or calculation is added that is not present in the text.
Every fact and detail is accurately represented.
Statements are clear and can be understood independently.
Numerical data and units are formatted exactly as provided in the commentary.
Each statement directly reflects the commentary without inferred details.
Pronouns are resolved; statements are unambiguous.
Each statement contains only one key entity and one action or attribute.

Figure 8: Prompt for Propositional Atomization (for LiveSum)

Prompt with Map&Make for LiveSum (Task2: Schema Extraction)

*****TASK 2***:**
Given a set of atomic text statements, extract row and column headers to create a table schema.

*****INSTRUCTIONS for TASK 2***:**
Do not proceed without completing TASK 1.
Read the statements carefully to identify all attributes, entities, and data points mentioned, whether explicitly stated or implicitly implied.
Determine the row headers (primary keys) and column headers required to represent the data comprehensively and concisely.
Row headers should be the teams participating in the match: [“Home Team”, “Away Team”].
Column headers should be the football event categories: [“Goal”, “Shots”, “Fouls”, “Yellow Cards”, “Red Cards”, “Corner Kicks”, “Free Kicks”, “Offsides”].
Ensure that all events from the atomic statements map into these categories.
Avoid adding actions as column headers but extract any data points associated with them.
Explicit numerical data must always be captured as attributes.

*****OUTPUT FORMAT for TASK 2***:**
{
“Match Summary Table”: {
“row_headers”: [“Home Team”, “Away Team”],
“column_headers”: [“Goals”, “Shots”, “Fouls”, “Yellow Cards”, “Red Cards”, “Corner Kicks”, “Free Kicks”, “Offsides”]
}
}

*****REASONING STEPS for TASK 2***:**
Step 1 - Identify the context from all the statements to generate a short description
Step 2 - Create an empty list of row and column headers for the tables.
This list would be updated as we keep on processing the statements and will keep adding relevant headers to the list.
Step 3 - Process statements one by one and add relevant headers if not already present in the list.

Figure 9: Prompt for Schema Extraction (for LiveSum)

Prompt with Map&Make for LiveSum (Task 3: Table Generation)

*****TASK 3***:**
Do not proceed without completing TASK 2.
Given:
Statements: A sequence of atomic statements from TASK 1.
Schema: A json object with table name, row headers, and column headers from TASK 2.
Your goal is to:
Process each statement one by one.
Identify the correct row (team) and column (event) and update the cell at that index to increment the count.
Update or add values to the table accordingly.

*****OUTPUT FORMAT for TASK 3***:**
Final Output Tables:
Match Summary Table
	Goals	Shots	Fouls	Yellow Cards	Red Cards	Corner Kicks	Free Kicks	Offsides
Home Team	x	x	x	x	x	x	x	x
Away Team	x	x	x	x	x	x	x	x

*****FINAL CHECKLIST for TASK 3***:**
Follow these guidelines to generate the final table and return the final state of the table after processing all the statements.
Ensure all sentences are processed and for every statement return the update and revised state of the updated cells.
Handle Missing Data: If a column value is not present in the statements, keep it as 0.
Structural Integrity: Do not add or remove any rows or columns unless explicitly instructed by the data.
Ensure uniformity in the format of data across the table.
The output should contain exactly one “Match Summary Table”.

Figure 10: Prompt for Table Generation (for LiveSum)