
AUTOMATIC LOGICAL FORMS IMPROVE FIDELITY IN TABLE-TO-TEXT GENERATION

Iñigo Alonso, Eneko Agirre

HiTZ Basque Center for Language Technology - Ixa NLP Group

University of the Basque Country UPV/EHU

{inigoborja.alonso,e.agirre}@ehu.eus

ABSTRACT

Table-to-text systems generate natural language statements from structured data like tables. While end-to-end techniques suffer from low factual correctness (fidelity), a previous study reported gains when using manual logical forms (LF) that represent the selected content and the semantics of the target text. Given the manual step, it was not clear whether automatic LFs would be effective, or whether the improvement came from content selection alone. We present *TIT* which, given a table and a selection of the content, first produces LFs and then the textual statement. We show for the first time that automatic LFs improve quality, with an increase in fidelity of 30 points over a comparable system not using LFs. Our experiments allow to quantify the remaining challenges for high factual correctness, with automatic selection of content coming first, followed by better Logic-to-Text generation and, to a lesser extent, better Table-to-Logic parsing.

1 INTRODUCTION

Data-to-text generation is the task of taking non-linguistic structured input such as tables, knowledge bases, tuples, or graphs, and automatically produce factually correct¹ textual descriptions of the contents of the input (Reiter & Dale, 1997; Covington, 2001; Gatt & Krahmer, 2018). Note that the task is somehow underspecified: for the same table many textual descriptions are correct, each one focusing on a selection of the contents. This makes the use of manual evaluation like fidelity key to measure quality.

Recent Data-to-Text techniques (Chen et al., 2020a;c; Aghajanyan et al., 2022; Kasner & Dusek, 2022) leverage the performance of large-scale pre-trained models (Devlin et al., 2019), with significant performance gains.

However, end-to-end systems struggle to produce high-fidelity statements. As a result, Chen et al. (2020c) propose to reformulate Data-to-Text as a Logic-to-Text problem focusing on tables, although the technique can be applied to other structured inputs. The input to the language realization module is a logical representation of the semantics of the target text along with the table information. The authors report an increase in factual correctness from 20% to 82%, compared to a system not using LFs. Note that the manually produced LFs include, implicitly, a selection of the contents to be used in the description. The authors left two open problems: Firstly, the improvement could come from the implicit content selection alone, casting doubts about the actual contribution of LFs. Secondly, it is not clear whether a system using automatic LFs would be as effective.

In this work, we present *TIT* (short from Table-to-Logic-to-Text), a two-step model that produces descriptions by automatically generating LFs and then producing the text from those LFs. Our model allows Table-to-Text generation systems to leverage the advantages of using LFs without requiring

¹We use factual correctness and fidelity indistinctly.

manually written LFs. We separate the content selection process from the logical form generation step, allowing to answer positively to the open questions mentioned above with experiments on the Logic2Text dataset (Chen et al., 2020c). Although content selection alone improves results, the best results are obtained using automatic LFs, with noteworthy gains in fidelity compared to a system not using LFs. Our results allow to estimate the impact in fidelity of the remaining challenges, with automatic content selection coming first, followed by better Logic-to-Text and to a lesser extent Table-to-Logic. We also provide qualitative analysis of each step.

All code, models and derived data are public ².

2 LOGICAL FORMS

The LFs used in this work are tree-structured logical representations of the semantics of a table-related statement, similar to AMR graphs (Banarescu et al., 2012), and follow the grammar rules defined by (Chen et al., 2020c). Each rule can be executed against a database, a table in this case, yielding a result based on the operation it represents. As these graphs represent factual statements, the root is a boolean operation that should return True. Figure 1 shows an example of a table with its caption and logical form.

2.1 DATASET

We use the Logic2Text dataset (Chen et al., 2020c). As mentioned in the introduction, Table-to-Text tasks are underspecified, as there are multiple descriptions about the table that could be factually correct and relevant. Logic2Text contains 4992 open-domain tables with an average of 2 manually constructed LFs and textual descriptions per table, making a total of 10753 samples (8566 train, 1092 dev. and 1095 test).

2.2 LOGICAL FORM GRAMMAR

The grammar contains several non-terminals (nodes in the graph, some of which are illustrated in Fig. 1), as follows:

Stat represents boolean comparative statements such as greater than, less than, equals (shown as *eq* in the figure), not equals, most equals or all equals, among others. This is the root of the LF graph.

C refers to an specific column in the input table (*attendance* and *result* in the figure).

V is used for specific values, which can be either values explicitly stated in the table (*w* in the figure) or arbitrary values used in comparisons or filters (*52500* in the figure).

View refers to a set of rows, which are selected according to a filter over all rows. The filters refer to specific conditions for the values in a specific column, e.g. *greater*. The figure shows *all_rows*, which returns all rows, and also *filter_str_eq* which returns the rows that contain the substring “*w*” in the *result* column.

N is used for operations that return a numeric value given a view and column as input, such as sums, averages (shown as *avg* in the figure), maximum or minimum values, and also counters.

Row is used to select a single row according to maximum or minimum values in a column.

Obj is used for operations that extract values in columns from rows (either views or specific rows). The most common operations are *hop* extractors that extract a unique value, for instance *str_hop_first* extracts a string from the first row of a given *View*.

I is used to select values from ordinal enumerations in *N* and *Row* rules, as for instance in order to select the “the 2nd highest” *I* would equal to 2.

Please refer to the Appendix C for full details. Note that *Stat*, *View*, *N*, *Row* and *Obj* are internal nodes that constitute the structure of the LF (shown in blue in the figure), while column *C*, value *V* and index *I* nodes are always leaf nodes.

²<https://github.com/AlonsoApp/tlt>

Caption:
1979 philadelphia eagles season

Table:

opponent	result	attendance
new york giants	w 23-17	67000
atlanta falcons	l 14-10	39700
new orleans saints	w 26-14	54000
new york giants	w 17-13	27500
pittsburgh steelers	w 17-14	61500

Statement: In the 1979 philadelphia eagles season there was an average attendance of 52500 in all wining games.

LF: eq { avg { filter_str_eq { all_rows ; result ; w } ; attendance } ; 52500 } = true

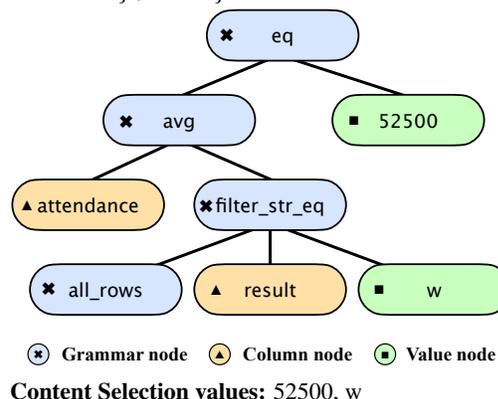


Figure 1: Example of a table with its caption, a logical form (in linearized and graph forms), its corresponding content selection values and the target statement. Note that w in the table stands for *win*. More details in the text.

We detected several ambiguities in the original grammar formulation that prevented training a semantic parser that outputs LFs.

The first one affects all functions that deal with strings. In the LF execution engine of Chen et al. (2020c) the implementation of those functions are divided in two: one that deals with normalization of numeric and date-like strings, and a strict version for other string values. We thus have two different functions in the grammar: a set for numerical and date-like values and another set for other string values, represented with the suffix “.str”. The second one deals with an issue of inconsistency with the *hop* function, which, given a row, returns the value associated to one of its columns. While the grammar states that these functions are only performed over *Row* objects, in 25% of the examples in the dataset the function is used over a *View* object, which can contain multiple rows. We defined a new function *hop.first* for these latter cases.

The grammar in Appendix C contains the new rules that fix the ambiguity issues. We also converted automatically each LF in the dataset to conform to the unambiguous grammar. The conversion script is publicly available.

2.3 CONTENT SELECTION

In order to separate the effect of content selection and full LFs, we extracted the values in the LF, so we can test the performance of all models with and without content selection. The extracted values include values that are explicitly mentioned in table cells, but also other values present in the LF that are not explicitly found in the table. The set of these values constitute the additional input to the systems when using content selection (CS for short), classified as follows:

TAB: Values present in a table cell, verbatim or as a substring of the cell values.

Figure 1 shows an example, where “w” is a substring in several cells. 72.2% of the values are of this type.

INF: Values not in the table that are inferred, e.g. as a result of an arithmetic operation over values in the table. For instance 52500 in Figure 1 corresponds to the average over attendance values. 20.8% of *Value* nodes are INF.

AUX: Auxiliary values not in the table nor INF that are used in operations, e.g. to be compared to actual values in cells, as in “All scores are bigger than 5.”. Only 7.1% are of type AUX.

In principle, one could train a separate model to select and produce all necessary content selection values to be fed into any Table-to-Text model, as follows: 1) Choose some values from table cells, either full or substring (TAB); 2) Infer some values via operations like average, count or max (INF); 3) Induce values to be used in comparisons (AUX). In order to separate the contribution of content

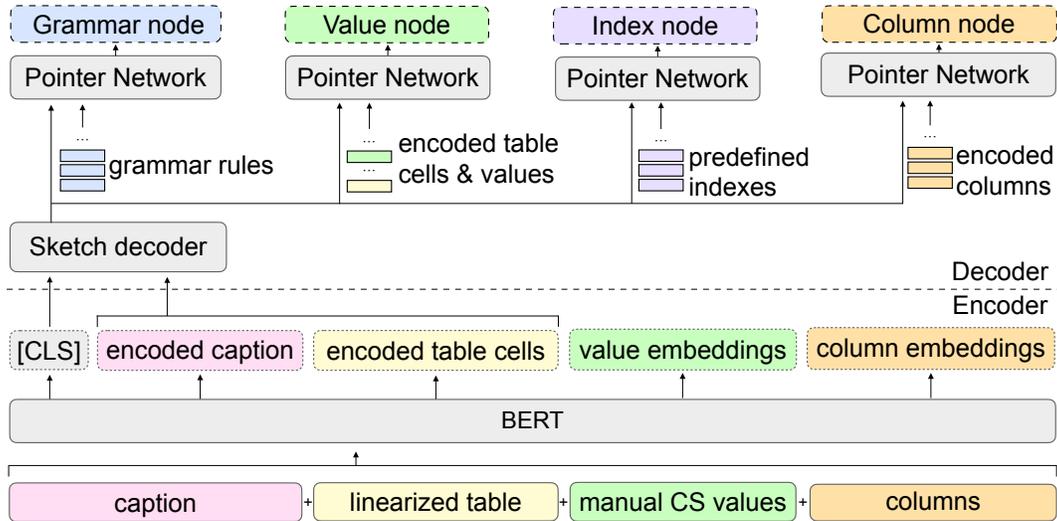


Figure 2: Table2Logic architecture, with input in the top and output in the bottom. See text for details.

selection and the generation of LFs, we decided to focus on the use of content selection, and not yet in producing the actual values. We thus derive these values from the manual gold LFs, and feed them to the models. The experiments will show that this content selection step is very important, and that current models fail without it. We leave automatic content selection for further research.

3 GENERATING TEXT VIA LOGICAL FORMS

Our Text-to-Logic-to-Text (*T/T*) system has two main modules in a pipeline.

Given a table, its caption and, optionally, selected content, **Table2Logic** generates an LF. With the same table information, plus the generated LF, **Logic2Text** produces the statement text.

3.1 TABLE2LOGIC

We frame this model as semantic parsing, adapting the IRNet grammar-based decoder by (Guo et al., 2019) to LFs. Given a table and corresponding LF in the dataset, the parser needs to produce the sequence of grammar derivations that leads to the given. More specifically, we follow the implementation of Valuenet by Brunner & Stockinger (2021), which is a more up to date revision of IRNet. Both models are NL-to-SQL semantic parsers that generate grammatically correct SQL sentences based on their descriptions. We adapted the system to produce logical forms instead of SQL.

The architecture of Table2Logic is presented in Figure 2.

We first feed a pre-trained BERT encoder (Devlin et al., 2019) with the concatenation of the following table data: the caption text, the table content in linearized form, the column names, and, in some of our model configurations, a set of content selection values manually extracted from the associated gold reference LF.

The output embeddings of the *CLS* token, the caption tokens and the linearized values in the table are fed into an LSTM decoder (Hochreiter & Schmidhuber, 1997).

At each decoding step, the attention vector of the LSTM is used by four pointer networks (PN) (Vinyals et al., 2015) that select the next grammar-related actions to be taken. Each of the PN accesses the attention vector of the LSTM plus additional information: the grammar PN has access to grammar information; the value PN uses output embeddings of table cells and other values; the index PN uses a separate set of embeddings for possible ordinal index values; the column PN uses column output embeddings.

Model	Sketch	Full
No content selection (TIT_{noCS})	15.0	4.9
TAB	42.6	27.3
INF	28.7	11.0
AUX	14.0	6.2
TAB, INF	56.5	39.3
TAB, AUX	44.3	28.6
TAB, INF, AUX	58.5	38.9
TAB, INF, AUX + FCR (TIT)	56.0	46.5

Table 1: Table2Logic: Accuracy (% on dev.) over sketch and full LFs using different subsets of content selection (CS) and FCR in development. First row for TIT_{noCS} , last row for TIT , as introduced in Sect. 5.

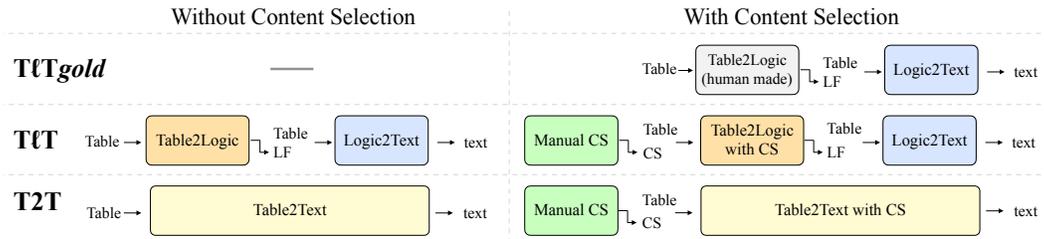


Figure 3: Model configurations used in main experiments.

Following (Guo et al., 2019), Table2Logic performs two decoding iterations. In a first iteration, a **sketch** LF is generated using the grammar pointer network. The sketch LF consisting only of grammar related nodes (e.g. those in blue in Fig. 1), where *Value*, *Column* and *Index* nodes are represented by placeholders that are filled in a second decoding iteration by the corresponding PN.

We follow teacher-based training to calculate the loss for each decoding iteration. In the first iteration the loss is calculated by accumulating the cross entropy loss for each generated grammar node given the previous gold reference nodes. The sketch is then used to calculate the cross entropy loss of generating *Value*, *Column* and *Index* nodes. The weights of the network are updated using the sum of both loss values.

During inference, we use beam search to produce a set of candidates. In addition, we explore a False Candidate Rejection (FCR) policy to filter out all LFs in the beam that execute to *False*, as they would be factually incorrect. Thus, the candidate LF in the beam that executes to *True* with maximum probability would be selected. Section 4 reports experiments with FCR.

3.2 LOGIC2TEXT

For the language realization model we use the top performer in (Chen et al., 2020c), which fine-tunes GPT-2 Radford et al. (2019) to produce text from tables. Their implementation allows to produce text from table information alone (caption, linearized table, list of column names) or both table information and a linearized logical form. See original publication for details.

4 DEVELOPMENT OF TABLE2LOGIC

In order to develop Table2Logic, we checked the effect of content selection, as well as the impact of rejecting LFs that evaluate to *False* (FCR) in development data. Accuracy was computed using strict equality with respect to any of the manual Gold LFs. Both sketch accuracy (using placeholders for non-grammar nodes) and full accuracy are reported. As mentioned in the introduction, this task is underspecified, in that multiple LFs which are very different from the gold LFs could be also correct. Still, the accuracy is a good proxy of quality to discriminate between better and worse models. The results correspond to the checkpoints, out of 50 epochs, with the best full accuracy on development.

We tuned some hyperparameters on development and used default values for the rest (see Appendix B for details).

Table 1 shows the results for different subsets of content selection values, with the last row reporting results when FCR is used. Without FCR, the most important set of values are those explicit in the table (TAB), and the best results correspond to the use of all values, although AUX values do not seem to help much (in fact, the best non-FCR full results are obtained without using AUX, by a very small margin). The last row reports a sizeable improvement in accuracy for full LFs when using FCR, showing that FCR is useful to reject faulty LFs that do not evaluate to True.

Overall, the full accuracy of TIT might seem low, but given that the gold LFs only cover a fraction of possible LFs they are actually of good quality, as we will see in the next sections.

We also performed an additional ablation experiment where we removed the table information from the system in the last row (TIT). The sketch and full accuracies dropped (50.3 and 42.7 respectively), showing that access to table information is useful even when content selection is available.

5 EXPERIMENTS

In this section we report the results on text generation using the test split of the Logic2Text dataset. We first introduce the different models, the automatic evaluation and the manual evaluation.

5.1 MODEL CONFIGURATIONS

The configuration of the different models are shown in Figure 3. All models take as input the table information, including table caption, linearized table and column headers. In the top row, we include the upperbound system TIT_{gold} , which takes the table plus the manually produced gold LF as input. In the middle row we include our system TIT , which is composed by the Table2Logic module and the Logic2Text module. Both TIT and TIT_{gold} use the same Logic2Text module, but while the first uses automatically produced LFs, the second uses manual LFs. TIT is evaluated in two variants, with and without content selection (TIT and TIT_{noCS} , respectively). Logic2Text uses default hyperparameters (Chen et al., 2020c).

The bottom row shows our baselines (T2T, short for Table2Text), which generate the text directly from table information, with and without content selection data. As Logic2Text is based on state-of-the-art generation (Chen et al., 2020c), and for the sake of comparability, both T2T and $T2T_{noCS}$ have the same codebase. That is, T2T uses the same GPT-2 model architecture as in Chen et al. (2020c) but trained without LFs. Receiving only the linearized table (in case of $T2T_{noCS}$) and, in the case of T2T, the same list of manual CS values as TIT .

5.2 AUTOMATIC EVALUATION

The automatic metrics compare the produced description with the reference descriptions in the test split. As shown in Table 2, we report the same automatic metrics as in (Chen et al., 2020c), BLEU-4 (B-4), ROUGE-1, 2, and L (R-1, R-2, and R-L for short), along with two additional metrics BERTscore (BERTs) (Zhang et al., 2019) and BARTscore (BARTs) (Yuan et al., 2021) which can capture the semantic similarity between the ground truth and generation results. The results show that generation without content selection is poor for both the baseline system and our system ($T2T_{noCS}$ and TIT_{noCS} , respectively). Content selection is key for good results in both kinds of systems, which improve around 10 points in all metrics when incorporating content selection (T2T and TIT). Automatic generation of LFs (TIT) allows to improve over the system not using them (T2T) in at least one point. If TIT had access to correct LFs it would improve 4 points further, as shown by the TIT_{gold} results. Note that our results for TIT_{gold} are very similar to those reported in (Chen et al., 2020c), as shown in the last row. We attribute the difference to minor variations in the model released by the authors.

5.3 HUMAN FIDELITY EVALUATION

Given the cost of human evaluation, we selected three models to manually judge the fidelity of the produced descriptions: the baseline T2T model, our TIT model and the upperbound with manual

Model	B-4	R-1	R-2	R-L	BERTs	BARTs
T2T _{noCS}	16.8	37.7	19.3	31.6	88.8	-4.04
TlT _{noCS}	15.6	39.0	18.9	32.2	87.9	-4.03
T2T	26.8	55.2	31.5	45.7	91.9	-2.98
TlT (ours)	27.2	56.0	33.1	47.7	92.0	-2.99
TlT _{gold}	31.7	62.4	38.7	52.8	93.1	-2.65
TlT _{gold} *	31.4*	64.2*	39.5*	54.0*	-	-

Table 2: Automated metrics for textual descriptions (test). Bottom two rows are upperbounds, as they use manual LFs. See text for system description. * for results reported in Chen et al. (2020c). Both BERTs and BARTs correspond to the f1 score. In case of the BARTscore higher is better.

Model	Faithful	Unfaithful	Nonsense
T2T _{noCS} *	20.2*	79.8*	-
T2T	44.9	49.3	5.8
TlT (ours)	75.0	20.3	4.7
TlT _{gold}	82.4	13.51	4.1

Table 3: Manual fidelity results. * for results reported in (Chen et al., 2020c).

LFs, TlT_{gold}. For this, we randomly selected 90 tables from the test set and generated a statement with each of the three models. In order to have two human judgements per example, we provided each evaluator with 30 sentences, along with the corresponding table and caption. The evaluators were asked to select whether the description is true, false or nonsense according to the caption and the table. This group of evaluators was comprised of eighteen volunteer researchers unrelated to this project. The evaluation concluded with a strong inter-evaluator agreement of 0.84 Fleiss’ kappa coefficient (Fleiss, 1971). We discarded examples where there was disagreement.

Table 3 shows the fidelity figures for the three models. After the evaluation, we noticed that the faithfulness results for TlT_{gold} in our experiment matched the figure reported by Chen et al. (2020c), so we decided, for completeness, to include in the table their figures for T2T_{noCS}, which should be roughly comparable to the other results in the table.

In general, the differences in human fidelity evaluation are much higher than for automatic metrics, which we attribute to widely recognised issues of automatic metrics when evaluating text generation. From low to high, the results allow us to estimate the **separate contributions** of each component:

- **Manual content selection** improves fidelity in 24 points (T2T_{noCS} vs. T2T) ;
- **Automatic LFs** improve an additional 30 points (T2T vs. TlT);
- **Manual LFs** give 7 points (TlT vs. TlT_{gold});
- **Perfect Logic2Text** generation would yield 18 points (TlT_{gold} vs. 100%).

The figures confirm our contribution: it is possible to produce logical forms automatically, and they allow to greatly improve fidelity, with the largest fidelity improvement in the table, 30 points. Note that the other improvements are actually gaps which allow us to prioritize the areas for further research: automatic content selection (24 pt.), better Logic2Text (18 pt.) and better Table2Logic (7 pt.). In the following section we analyse the errors in the two later modules.

6 QUALITATIVE ANALYSIS

We performed a qualitative analysis of failure cases in both Table2Logic and Logic2Text, as well as examples of factually correct descriptions generated from LFs different from gold LFs.

	Fr.	Total	Confusions
Stat	0.38	0.13	greater → less all equals → most equals equals → and
C	0.25	0.19	column 3 → column 0 column 1 → column 0
Row	0.16	0.02	row 0 → row 2 row 2 → row 0 row 2 → row 1
View	0.11	0.20	filter_greater → filter_less filter_greater → filter_eq filter_eq → all_rows
N	0.05	0.03	sum → avg avg → sum
Obj	0.03	0.26	str_hop → num_hop num_hop → str_hop
V	0.01	0.16	value 72 → value 73 value 70 → value 71
I	0.01	0.01	1 → 0

Table 4: Table2Logic: Distribution of differing node types (*TIT* vs. gold LFs). Fr. for frequency of node type in differing LFs, Total for overall frequency in gold. Rightmost column for most frequent confusions (*TIT* → gold).

6.1 TABLE2LOGIC

We automatically compared the LFs generated by *TIT* in the development set that did not match their corresponding gold LFs. Note that the produced LFs can be correct even if they do not match the gold LF. We traverse the LF from left to right and record the first node that is different. Table 4 shows, in decreasing order of frequency, each grammar node type (cf. Section 2.2) with the most frequent confusions.

The most frequent differences focus on *Stat* nodes, where a different comparison is often generated. The next two frequent nodes are column and row selections, where *TIT* selects different columns and rows, even if *TIT* has access to the values in the content selection. The frequency of differences of these three node types is well above the distribution in gold LFs. The rest of differences are less frequent, and also focus on generating different comparison or arithmetic operations.

6.2 LOGIC2TEXT

The faithfulness score of descriptions generated from gold LFs (*TIT_{gold}*) is 82%, so we analysed a sample of the examples in this 18%. For the sake of space, we include full examples in Appendix D, which include table, caption, gold LF and generated description. We summarize the errors in three types:

Comparative arithmetic: Logic2Text miss-represented comparative arithmetic action rules in the LF in 40% of the cases. This resulted in cases where the output sentence declared that a given value was *smaller* than another when the LF stated it was *larger*. Logic2Text also seem to ignore *round* and *most* modifiers of comparison operations, producing sentences with strict equality and omitting qualifiers like “roughly” or “most”. The absence of these qualifiers made the produced sentences factually incorrect.

LF omission: Logic2Text disregarded part of the LF (33% of errors), resulting in omissions that led to false sentences. Many of these errors involved omitting an entire branch of the LF, leading, for instance, to sentences wrongly referring to all the instances in the data instead of the subset described in the LF.

LF difference	Sentences
Similar structure, semantically equivalent	<p><i>TIT</i>: In the list of Appalachian regional commission counties, Schoharie has the highest unemployment rate.</p> <p>Human: The appalachian county that has the highest unemployment rate is Schoharie.</p>
Similar structure, semantically different	<p><i>TIT</i>: Dick Rathmann had a lower rank in 1956 than he did in 1959.</p> <p>Human: Dick Rathmann completed more laps in the Indianapolis 500 in 1956 than in 1959.</p>
Different structure, semantically different	<p><i>TIT</i>: Most of the games of the 2005 Houston Astros’ season were played in the location of arlington.</p> <p>Human: Arlington was the first location used in the 2005 Houston Astros season.</p>
Simpler structure, more informative	<p><i>TIT</i>: Aus won 7 events in the 2006 asp world tour.</p> <p>Human: Seven of the individuals that were the runner up were from aus.</p>

Table 5: Examples of faithful sentences produced by *TIT* from intermediate LFs that do not match the gold LF.

Verbalization: Logic2Text incurred in wrong verbalization and misspellings (27% of cases). For instance Logic2Text producing a similar but not identical name like in *foulisco* instead of *francisco*.

We attribute the errors to the fact that the generator is based on a general Language Model such as GPT-2. While these language models are excellent in producing fluent text, it seems that, even after fine-tuning, they have a tendency to produce sentences that do not fully reflect the data in the input logical form. It also seems that the errors might be explained by the lower frequency of some operations. The 18% gap, even if it is much lower than the gap for systems that do not use LFs, together with this analysis, show that there is still room for improvement.

6.3 CAN AN INCORRECT LF PRODUCE A FAITHFUL DESCRIPTION?

The results in Table 1 show that our Table2Logic system has low accuracy when evaluated against gold logical forms (46%). On the contrary, the results in fidelity for the text generated using those automatically generated logical forms is very high, 75%, only 7 points lower to the results when using gold logical forms. This high performance in fidelity for automatic LFs might seem counter-intuitive, but we need to note that it is possible to generate a correct and faithful LF that is completely different from the gold logical form, i.e. the system decides to produce a correct LF that focuses on a different aspect of the information in the table with respect to the gold LF.

In order to check whether this is actually the case, we manually examined the automatic LFs from *TIT* that resulted in faithful sentences in the manual evaluation while being “erroneous”, that is, different from their gold LF references. In all cases, such *TIT* LFs are correctly formed and faithful, i.e. even if these LFs were “wrong” according to the strict definition of accuracy, the semantics they represent are informative and faithful to the source data. Table 5 shows a sample of the output sentence, with full details including table and LFs in Appendix E.

We categorized the samples as follows. 69% of them share a similar LF structure as their corresponding gold references, but with changes in key *Value* or *Column* nodes, making them semantically different. In 15% of the cases the LF had similar structure, but although there were differences, the LF was semantically equivalent to the gold LF. The rest of *TIT* LFs (16%) had a different structure, and were semantically different from reference counterparts, while still being correct and faithful to the table.

All in all the quality of LFs and corresponding text produced by *TIT* for this sample is comparable to those of the gold LF, and in some cases more concise and informative. This analysis confirms that the quality of Table2Logic is well over the 46% accuracy estimate, and that it can be improved, as the produced text lags 7 points behind gold LFs.

7 RELATED WORK

Natural Language Generation from structured data is a long-established research line. Over time, multiple techniques have been developed to solve this task in different ways, such as leveraging the structural information of the input data (Wiseman et al., 2017; Liu et al., 2018; Puduppully et al., 2019a; Rebuffel et al., 2020; Chen et al., 2020b), using neural templates (Wiseman et al., 2018; Li & Wan, 2018) or focusing on content ordering (Sha et al., 2018; Puduppully et al., 2019b; Su et al., 2021). However, recent techniques (Chen et al., 2020a; Aghajanyan et al., 2022; Kasner & Dusek, 2022) leverage pre-trained language models (Devlin et al., 2019; Radford et al., 2019).

The use of pre-trained language models has allowed for highly fluent outputs, but fidelity is still a big challenge and focus of recent work. Matsumaru et al. (2020) remove factually incorrect instances from the training data. Others take control of the decoder by making it attend to the source (Tian et al., 2019), using re-ranking techniques on it (Harkous et al., 2020) or applying constraints that incorporate heuristic estimates of future cost (Lu et al., 2021). Other work relies on heuristics such as surface matching of source and target to control generation (Wang et al., 2020; Shen et al., 2020; Li & Rush, 2020).

In a complementary approach, Chen et al. (2020c) focus on improving the fidelity of the generated texts by reformulating Table-to-Text as a Logic-to-Text problem. They incorporate a tree-structured logical representation of the semantics of the target text, logical forms (LF), along with the table information. This logical form highly conditions the language realization module to produce the statement represented in it, greatly improving fidelity. However, the logical forms in this work are manually produced by humans, highly reducing the applicability of this solution in a real world scenario. Solving this challenge would allow data-to-text models to leverage the benefits of this approach, which motivated our research.

Automatically generating LFs requires of techniques capable of producing outputs following a set of pre-defined grammar rules. This challenge is commonly addressed in many semantic parsing tasks (Yin & Neubig, 2017; Radhakrishnan et al., 2020). Guo et al. (2019) present IRNet, a NL-to-SQL semantic parser that generates grammatically correct SQL sentences based on their natural language descriptions. Valuenet Brunner & Stockinger (2021) introduced a BERT-based encoder (Devlin et al., 2019). In this work we adapted the grammar-based decoder of Valuenet to produce LFs, which allowed us to show that we can produce high quality LFs. More recent advances in semantic parsing, e.g. the use of larger language models (Raffel et al., 2020; BigScience Workshop, 2022; Zhang et al., 2022), could be easily folded in our system and would further increase the contribution of LFs.

8 CONCLUSIONS AND FUTURE WORK

We have presented *T_LT* which, given a table and a selection of the content, first produces logical forms and then the textual statement. We show for the first time that automatic LFs improve results according to automatic metrics and, especially, manually estimated factual correctness. In addition, we separately study the contribution of content selection and the formalization of the output as an LF, showing a higher impact in fidelity of the later. In this paper we focus on tables, but our findings and software can be easily ported to other structured inputs.

Our analysis allowed us to quantify that content selection would provide the largest boost in performance, followed, to a lesser extent in improved logic-to-text generation, and, finally, improved table-to-logic generation. We thus plan to focus on automatic content selection, which we think can be largely learned from user preference patterns found in the training data. We also plan to leverage our qualitative analysis to study complementary approaches to improve factual correctness in logic-to-text.

ACKNOWLEDGEMENTS

This work is partially funded by MCIN/AEI 10.13039/501100011033 and by the European Union NextGenerationEU/ PRTR, as well as the Basque Government IT1570-22.

REFERENCES

- Armen Aghajanyan, Dmytro Okhonko, Mike Lewis, Mandar Joshi, Hu Xu, Gargi Ghosh, and Luke Zettlemoyer. HTLM: Hyper-text pre-training and prompting of language models. In *International Conference on Learning Representations*, 2022. URL <https://openreview.net/forum?id=P-pPW1nxflr>.
- Laura Banarescu, Claire Bonial, Shu Cai, Madalina Georgescu, Kira Griffitt, Ulf Hermjakob, Kevin Knight, Philipp Koehn, Martha Palmer, and Nathan Schneider. Abstract meaning representation (amr) 1.0 specification. In *Abstract meaning representation (amr) 1.0 specification*, volume Parsing on Freebase from Question-Answer Pairs. In Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing. Seattle: ACL, pp. 1533–1544, 2012.
- BigScience Workshop. Bloom (revision 4ab0472), 2022. URL <https://huggingface.co/bigscience/bloom>.
- Ursin Brunner and Kurt Stockinger. Valuenet: A natural language-to-sql system that learns from database information. In *2021 IEEE 37th International Conference on Data Engineering (ICDE)*, pp. 2177–2182, 2021. doi: 10.1109/ICDE51399.2021.00220.
- Wenhu Chen, Jianshu Chen, Yu Su, Zhiyu Chen, and William Yang Wang. Logical natural language generation from open-domain tables. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pp. 7929–7942, Online, July 2020a. Association for Computational Linguistics. doi: 10.18653/v1/2020.acl-main.708. URL <https://aclanthology.org/2020.acl-main.708>.
- Wenhu Chen, Yu Su, Xifeng Yan, and William Yang Wang. KGPT: Knowledge-grounded pre-training for data-to-text generation. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 8635–8648, Online, November 2020b. Association for Computational Linguistics. doi: 10.18653/v1/2020.emnlp-main.697. URL <https://aclanthology.org/2020.emnlp-main.697>.
- Zhiyu Chen, Wenhu Chen, Hanwen Zha, Xiyu Zhou, Yunkai Zhang, Sairam Sundaresan, and William Yang Wang. Logic2Text: High-fidelity natural language generation from logical forms. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pp. 2096–2111, Online, November 2020c. Association for Computational Linguistics. doi: 10.18653/v1/2020.findings-emnlp.190. URL <https://aclanthology.org/2020.findings-emnlp.190>.
- Michael A Covington. Building natural language generation systems. *Language*, 77(3):611–612, 2001. doi: 10.1353/lan.2001.0146.
- Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 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)*, pp. 4171–4186, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics. doi: 10.18653/v1/N19-1423. URL <https://aclanthology.org/N19-1423>.
- Joseph L Fleiss. Measuring nominal scale agreement among many raters. *Psychological bulletin*, 76(5):378, 1971.
- Albert Gatt and Emiel Kraemer. Survey of the state of the art in natural language generation: Core tasks, applications and evaluation. *Journal of Artificial Intelligence Research*, 61:65–170, 2018. doi: 10.1613/jair.5477.
- Jiaqi Guo, Zecheng Zhan, Yan Gao, Yan Xiao, Jian-Guang Lou, Ting Liu, and Dongmei Zhang. Towards complex text-to-SQL in cross-domain database with intermediate representation. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pp. 4524–4535, Florence, Italy, July 2019. Association for Computational Linguistics. doi: 10.18653/v1/P19-1444. URL <https://aclanthology.org/P19-1444>.

- Hamza Harkous, Isabel Groves, and Amir Saffari. Have your text and use it too! end-to-end neural data-to-text generation with semantic fidelity. In *Proceedings of the 28th International Conference on Computational Linguistics*, pp. 2410–2424, Barcelona, Spain (Online), December 2020. International Committee on Computational Linguistics. doi: 10.18653/v1/2020.coling-main.218. URL <https://aclanthology.org/2020.coling-main.218>.
- Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Computation*, 9(8): 1735–1780, Nov 1997. ISSN 0899-7667. doi: 10.1162/neco.1997.9.8.1735.
- Zdeněk Kasner and Ondrej Dusek. Neural pipeline for zero-shot data-to-text generation. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 3914–3932, Dublin, Ireland, May 2022. Association for Computational Linguistics. doi: 10.18653/v1/2022.acl-long.271. URL <https://aclanthology.org/2022.acl-long.271>.
- Liunian Li and Xiaojun Wan. Point precisely: Towards ensuring the precision of data in generated texts using delayed copy mechanism. In *Proceedings of the 27th International Conference on Computational Linguistics*, pp. 1044–1055, Santa Fe, New Mexico, USA, August 2018. Association for Computational Linguistics. URL <https://aclanthology.org/C18-1089>.
- Xiang Lisa Li and Alexander Rush. Posterior control of blackbox generation. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pp. 2731–2743, Online, July 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.acl-main.243. URL <https://aclanthology.org/2020.acl-main.243>.
- Tianyu Liu, Kexiang Wang, Lei Sha, Baobao Chang, and Zhifang Sui. Table-to-text generation by structure-aware seq2seq learning. In *Table-to-text generation by structure-aware seq2seq learning*, volume Proceedings of the AAAI Conference on Artificial Intelligence 32(1), 2018. doi: 10.1609/aaai.v32i1.11925.
- Ximing Lu, Sean Welleck, Peter West, Liwei Jiang, Jungo Kasai, Daniel Khashabi, Ronan Le Bras, Lianhui Qin, Youngjae Yu, and Rowan Zellers. Neurologic a* esque decoding: Constrained text generation with lookahead heuristics. *arXiv preprint arXiv:2112.08726*, 2021.
- Kazuki Matsumaru, Sho Takase, and Naoaki Okazaki. Improving truthfulness of headline generation. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pp. 1335–1346, Online, July 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.acl-main.123. URL <https://aclanthology.org/2020.acl-main.123>.
- Ratish Puduppully, Li Dong, and Mirella Lapata. Data-to-text generation with entity modeling. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pp. 2023–2035, Florence, Italy, July 2019a. Association for Computational Linguistics. doi: 10.18653/v1/P19-1195. URL <https://aclanthology.org/P19-1195>.
- Ratish Puduppully, Li Dong, and Mirella Lapata. Data-to-text generation with content selection and planning. *Proceedings of the AAAI Conference on Artificial Intelligence*, 33(01):6908–6915, Jul. 2019b. doi: 10.1609/aaai.v33i01.33016908. URL <https://ojs.aaai.org/index.php/AAAI/article/view/4668>.
- A Radford, J Wu, R Child, D Luan, and D Amodei. . . . Language models are unsupervised multitask learners. *OpenAI Blog*, 2019. URL <https://d4mucfpksywv.cloudfront.net/better-language-models/language-models.pdf>.
- Karthik Radhakrishnan, Arvind Srikantan, and Xi Victoria Lin. ColloQL: Robust text-to-SQL over search queries. In *Proceedings of the First Workshop on Interactive and Executable Semantic Parsing*, pp. 34–45, Online, November 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.intexsempar-1.5. URL <https://aclanthology.org/2020.intexsempar-1.5>.
- Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research*, 21(140):1–67, 2020. URL <http://jmlr.org/papers/v21/20-074.html>.

- Clément Rebuffel, Laure Soulier, Geoffrey Scoutheeten, and Patrick Gallinari. A hierarchical model for data-to-text generation. In *A Hierarchical Model for Data-to-Text Generation*, volume European Conference on Information Retrieval, pp. 65–80. Springer, 2020. doi: 10.1007/978-3-030-45439-5_5.
- Ehud Reiter and Robert Dale. Building applied natural language generation systems. *Natural Language Engineering*, 3(1):57–87, 1997. doi: 10.1017/S1351324997001502.
- Lei Sha, Lili Mou, Tianyu Liu, Pascal Poupart, Sujian Li, Baobao Chang, and Zhifang Sui. Order-planning neural text generation from structured data. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence and Thirtieth Innovative Applications of Artificial Intelligence Conference and Eighth AAAI Symposium on Educational Advances in Artificial Intelligence*, AAAI’18/IAAI’18/EAAI’18. AAAI Press, 2018. ISBN 978-1-57735-800-8.
- Xiaoyu Shen, Ernie Chang, Hui Su, Cheng Niu, and Dietrich Klakow. Neural data-to-text generation via jointly learning the segmentation and correspondence. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pp. 7155–7165, Online, July 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.acl-main.641. URL <https://aclanthology.org/2020.acl-main.641>.
- Yixuan Su, David Vandyke, Sihui Wang, Yimai Fang, and Nigel Collier. Plan-then-generate: Controlled data-to-text generation via planning. In *Findings of the Association for Computational Linguistics: EMNLP 2021*, pp. 895–909, Punta Cana, Dominican Republic, November 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.findings-emnlp.76. URL <https://aclanthology.org/2021.findings-emnlp.76>.
- Ran Tian, Shashi Narayan, Thibault Sellam, and Ankur P Parikh. Sticking to the facts: Confident decoding for faithful data-to-text generation. *arXiv preprint arXiv:1910.08684*, 2019.
- Oriol Vinyals, Meire Fortunato, and Navdeep Jaitly. Pointer networks. In C. Cortes, N. Lawrence, D. Lee, M. Sugiyama, and R. Garnett (eds.), *Advances in Neural Information Processing Systems*, volume 28. Curran Associates, Inc., 2015. URL <https://proceedings.neurips.cc/paper/2015/file/29921001f2f04bd3baee84a12e98098f-Paper.pdf>.
- Zhenyi Wang, Xiaoyang Wang, Bang An, Dong Yu, and Changyou Chen. Towards faithful neural table-to-text generation with content-matching constraints. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pp. 1072–1086, Online, July 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.acl-main.101. URL <https://aclanthology.org/2020.acl-main.101>.
- Sam Wiseman, Stuart Shieber, and Alexander Rush. Challenges in data-to-document generation. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pp. 2253–2263, Copenhagen, Denmark, September 2017. Association for Computational Linguistics. doi: 10.18653/v1/D17-1239. URL <https://aclanthology.org/D17-1239>.
- Sam Wiseman, Stuart Shieber, and Alexander Rush. Learning neural templates for text generation. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pp. 3174–3187, Brussels, Belgium, October–November 2018. Association for Computational Linguistics. doi: 10.18653/v1/D18-1356. URL <https://aclanthology.org/D18-1356>.
- Pengcheng Yin and Graham Neubig. A syntactic neural model for general-purpose code generation. In *The 55th Annual Meeting of the Association for Computational Linguistics (ACL)*, Vancouver, Canada, July 2017. URL <https://arxiv.org/abs/1704.01696>.
- Weizhe Yuan, Graham Neubig, and Pengfei Liu. Bartscore: Evaluating generated text as text generation. *Advances in Neural Information Processing Systems*, 34:27263–27277, 2021.
- Susan Zhang, Stephen Roller, Naman Goyal, Mikel Artetxe, Moya Chen, Shuohui Chen, Christopher Dewan, Mona Diab, Xian Li, Xi Victoria Lin, Todor Mihaylov, Myle Ott, Sam Shleifer, Kurt Shuster, Daniel Simig, Punit Singh Koura, Anjali Sridhar, Tianlu Wang, and Luke Zettlemoyer. Opt: Open pre-trained transformer language models, 2022. URL <https://arxiv.org/abs/2205.01068>.

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

A TRAINING PROCEDURE

All experiments were carried out in a machine with a GPU NVIDIA TITAN Xp 12GB. The average training runtime for all Table2Logic based models is 19 hours. For the Logic2Text presented models, it averaged 10 hours. Both Table2Logic and Logic2Text models have a very similar amount of parameters (117M).

B MODEL HYPER-PARAMETERS

We keep Logic2Text’s hyper-parameters the same as Chen et al. (2020c). We refer the reader to the paper. Regarding the Table2Logic model in *TIT*, which is based on Brunner & Stockinger (2021)’s Valuenet, we changed the grammar and added additional input data, as well as changing the code accordingly to our use case. We use the same hyper-parameters as stated in the paper, with the exception of the base learning rate, beam size, number epochs, and gradient clipping. This is the list of hyper-parameters used by Table2Logic for the model *TIT*:

Random seed: 90	Attention vector size: 300
Maximum sequence length: 512	Grammar type embedding size: 128
Batch size: 8	Grammar node embedding size: 128
Epochs: 50	Column node embedding size: 300
Base learning rate: $5 * 10^{-5}$	Index node embedding size: 300
Connection learning rate: $1 * 10^{-4}$	Readout: 'identity'
Transformer learning rate: $2 * 10^{-5}$	Column attention: 'affine'
Scheduler gamma: 0.5	Dropout rate: 0.3
ADAM maximum gradient norm: 1.0	Largest index for I nodes: 20
Gradient clipping: 0.1	Include OOV token: True
Loss epoch threshold: 50	Beam size: 2048
Sketch loss weight: 1.0	Max decoding steps: 50
Word embedding size: 300	False Candidate Rejection: True
Size of LSTM hidden states: 300	

C LOGICAL FORM GRAMMAR

Stat ::= *only* View | *and* Stat Stat | *greater* Obj Obj | *less* Obj Obj | *eq* Obj Obj |
 str_eq Obj Obj | *not_eq* Obj Obj | *not_str_eq* Obj Obj | *round_eq* Obj Obj |
 all_eq View C Obj | *all_str_eq* View C Obj | *all_not_eq* View C Obj |
 all_str_not_eq View C Obj | *all_less* View C Obj | *all_less_eq* View C Obj |
 all_greater View C Obj | *all_greater_eq* View C Obj | *most_eq* View C Obj |
 most_str_eq View C Obj | *most_not_eq* View C Obj |
 most_str_not_eq View C Obj | *most_less* View C Obj | *most_less_eq* View C Obj |
 most_greater View C Obj | *most_greater_eq* View C Obj
 View ::= *all_rows* | *filter_eq* View C Obj | *fiter_str_eq* View C Obj |
 filter_not_eq View C Obj | *fiter_str_not_eq* View C Obj |
 filter_less View C Obj | *filter_greater* View C Obj | *filter_greater_eq* View C Obj |
 filter_less_eq View C Obj | *filter_all* View C
 N ::= *count* View | *avg* View C | *sum* View C | *max* View C | *min* View C |
 nth_max View C I | *nth_min* View C I
 Row ::= *argmax* View C | *argmin* View C | *nth_argmax* View C I | *nth_argmin* View C I
 Obj ::= *str_hop* Row C | *num_hop* Row C | *str_hop_first* View C |
 num_hop_first View C | *diff* Obj Obj | N | V
 C ::= column
 I ::= index
 V ::= value

Figure 4: The Logical Form Grammar after fixing the ambiguity issues in the original version (Chen et al., 2020c). We follow the same notation as in IRNet and Valuenet. The tokens to the left of the ::= represent non-terminals (node types in the graph). Tokens in italics represent the possible rules for each node, with pipes (|) separating the rules. The rules added to the original grammar in order to fix ambiguity issues are highlighted in green.

D LOGIC2TEXT ERRORS

This section shows examples of error cases where the logic-to-text stage of the pipeline failed to produce faithful sentences given a gold LF. We include one example for each error type, including table, caption, gold logical form and generated description. See Section 6.2 for more details.

D.1 COMPARATIVE ARITHMETIC

Caption: fil world luge championships 1961

Table:

rank	nation	gold	silver	bronze	total
1	austria	0	0	3	3
2	italy	1	1	0	2
3	west germany	0	2	0	2
4	poland	1	0	0	1
5	switzerland	1	0	0	1

Logical Form:

```
and
├── only
│   ├── filter_greater
│   │   ├── 0
│   │   ├── all_rows
│   │   └── bronze
│   └── str_eq
│       ├── austria
│       └── str_hop_first
│           ├── filter_greater
│           │   ├── 0
│           │   ├── all_rows
│           │   └── bronze
│           └── nation
```

TIT sentence: austria was the only country to win 0 bronze medals at the fil world luge championships .

Gold sentence: austria was the only country to have bronze medals in the luge championship in 1961 .

D.2 LF OMISSION

Caption: geography of moldova

Table:

land formation	area , km square	of which currently forests , km square	% forests	habitat type
northern moldavian hills	4630	476	10.3 %	forest steppe
dniester - rãut ridge	2480	363	14.6 %	forest steppe
middle prut valley	2930	312	10.6 %	forest steppe
bãlți steppe	1920	51	2.7 %	steppe
ciuluc - soloneț hills	1690	169	10.0 %	forest steppe
cornești hills (codru)	4740	1300	27.5 %	forest
lower dniester hills	3040	371	12.2 %	forest steppe
lower prut valley	1810	144	8.0 %	forest steppe
tigheci hills	3550	533	15.0 %	forest steppe
bugeac plain	3210	195	6.1 %	steppe
part of podolian plateau	1920	175	9.1 %	forest steppe
part of eurasian steppe	1920	140	7.3 %	steppe

Logical Form:

```
eq
├── 8
└── count
    ├── filter_str_eq
    │   ├── all_rows
    │   ├── forest steppe
    │   └── habitat type
```

TIT sentence: there are 8 habitats that can be found in moldova .

Gold sentence: 8 land formations are classified with a habitat type of forest steppe .

D.3 VERBALIZATION

Caption: seattle supersonics all - time roster

Table:

player	nationality	jersey number (s)	position	years	from
craig ehlo	united states	3	sg	1996 - 1997	washington state
dale ellis	united states	3	sg / sf	1986 - 1991 1997 - 1999	tennessee
pervis ellison	united states	29	c	2000	louisville
francisco elson	netherlands	16	c	2008	california
reggie evans	united states	34 , 30	pf	2002 - 2006	iowa
patrick ewing	united states	33	center	2000 - 2001	georgetown

Logical Form:

```

greater
├── num_hop_first
│   ├── filter_str_eq
│   │   ├── all_rows
│   │   ├── francisco elson
│   │   └── player
│   └── years
├── num_hop_first
│   ├── filter_str_eq
│   │   ├── all_rows
│   │   ├── pervis ellison
│   │   └── player
│   └── years

```

TIT sentence: foulisco elson played for the supersonics after pervis ellison .

Gold sentence: francisco elson played 8 years later thanpervis ellison .

E EXAMPLES OF FAITHFUL *TIT* SENTENCES WHERE LF IS DIFFERENT TO GOLD

This section shows examples of automatic LFs from *TIT* that resulted in faithful sentences in the manual evaluation while being different from their gold LF references. Each example extends the information shown in Table 5.

E.1 SIMILAR STRUCTURE, SEMANTICALLY EQUIVALENT

Caption: list of appalachian regional commission counties

Table:

county	population	unemployment rate	market income per capita	poverty rate	status
allegany	49927	5.8 %	16850	15.5 %	- risk
broome	200536	5.0 %	24199	12.8 %	transitional
cattaraugus	83955	5.5 %	21285	13.7 %	transitional
chautauqua	136409	4.9 %	19622	13.8 %	transitional
chemung	91070	5.1 %	22513	13.0 %	transitional
chenango	51401	5.5 %	20896	14.4 %	transitional
cortland	48599	5.7 %	21134	15.5 %	transitional
delaware	48055	4.9 %	21160	12.9 %	transitional
otsego	61676	4.9 %	21819	14.9 %	transitional
schoharie	31582	6.0 %	23145	11.4 %	transitional
schuyler	19224	5.4 %	21042	11.8 %	transitional
steuben	98726	5.6 %	28065	13.2 %	transitional
tioga	51784	4.8 %	24885	8.4 %	transitional

TIT Logical Form:

```

str_eq
├─ schoharie
└─ str_hop
   └─ county
      └─ nth_argmax
         └─ 1
            └─ all_rows
               └─ unemployment rate

```

Gold Logical Form:

```

str_eq
├─ schoharie
└─ str_hop
   └─ argmax
      └─ all_rows
         └─ unemployment rate
      └─ county

```

***TIT* sentence:** in the list of appalachian regional commission counties , schoharie has the highest unemployment rate .

Human sentence: the appalachian county that has the highest unemployment rate is schoharie .

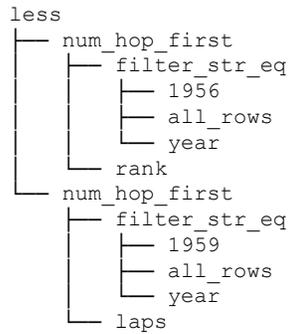
E.2 SIMILAR STRUCTURE, SEMANTICALLY DIFFERENT

Caption: dick rathmann

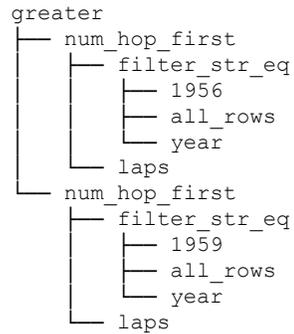
Table:

year	qual	rank	finish	laps
1950	130.928	17	32	25
1956	144.471	6	5	200
1957	140.780	withdrew	withdrew	withdrew
1958	145.974	1	27	0
1959	144.248	5	20	150
1960	145.543	6	31	42
1961	146.033	8	13	164
1962	147.161	13	24	51
1963	149.130	14	10	200
1964	151.860	17	7	197

TIT Logical Form:



Gold Logical Form:



TIT sentence: dick rathmann had a lower rank in 1956 than he did in 1959 .

Human sentence: dick rathmann completed more laps in the indianapolis 500 in 1956 than in 1959 .

E.3 DIFFERENT STRUCTURE, SEMANTICALLY DIFFERENT

Caption: 2005 houston astros season

Table:

date	winning team	score	winning pitcher	losing pitcher	attendance	location
may 20	texas	7 - 3	kenny rogers	brandon backe	38109	arlington
may 21	texas	18 - 3	chris young	ezequiel astacio	35781	arlington
may 22	texas	2 - 0	chan ho park	roy oswalt	40583	arlington
june 24	houston	5 - 2	roy oswalt	ricardo rodriguez	36199	houston
june 25	texas	6 - 5	chris young	brandon backe	41868	houston

TIT Logical Form:

```

most_str_eq
├── all_rows
├── arlington
└── location

```

Gold Logical Form:

```

str_eq
├── arlington
└── str_hop
    ├── argmin
    │   ├── all_rows
    │   └── date
    └── location

```

TIT sentence: most of the games of the 2005 houston astros ' season were played in the location of arlington .

Human sentence: arlington was the first location used in the 2005 houston astros season .

E.4 SIMPLER, MORE INFORMATIVE SEMANTIC

Caption: 2006 asp world tour

Table:

location	country	event	winner	runner - up
gold coast	australia	roxy pro gold coast	melanie redman - carr (aus)	layne beachley (aus)
tavarua	fiji	roxy pro fiji	melanie redman - carr (aus)	layne beachley (aus)
teahupoo , tahiti	french polynesia	billabong pro tahiti women	melanie redman - carr (aus)	chelsea georgeson (aus)
itacarā	brazil	billabong girls pro	layne beachley (aus)	jessi miley - dyer (aus)
hossegor	france	rip curl pro mademoiselle	chelsea georgeson (aus)	melanie redman - carr (aus)
manly beach	australia	havaianas beachley classic	stephanie gilmore (aus)	layne beachley (aus)
sunset beach , hawaii	united states	roxy pro	melanie bartels (haw)	stephanie gilmore (aus)
honolua bay , hawaii	united states	billabong pro	jessi miley - dyer (aus)	keala kennelly (haw)

TIT Logical Form:

```

eq
├── 7
└── count
    ├── filter_str_eq
    │   ├── all_rows
    │   ├── aus
    │   └── winner
    
```

Gold Logical Form:

```

eq
├── 7
└── count
    ├── filter_str_eq
    │   ├── all_rows
    │   ├── aus
    │   └── runner - up
    
```

TIT sentence: aus won 7 events in the 2006 asp world tour .

Human sentence: seven of the individuals that were the runner up were from aus .