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 TlT 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 correct textual descriptions of the contents of the input [Reiter & Dale, 1997; Covington, 2001; Gatt & Krahnke, 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 TlT (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

1 We 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 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 a 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 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.

https://github.com/AlonsoApp/tlt
**Caption:**
1979 Philadelphia Eagles season

**Table:**

<table>
<thead>
<tr>
<th>opponent</th>
<th>result</th>
<th>attendance</th>
</tr>
</thead>
<tbody>
<tr>
<td>New York Giants</td>
<td>w 23-17</td>
<td>67000</td>
</tr>
<tr>
<td>Atlanta Falcons</td>
<td>l 14-10</td>
<td>39700</td>
</tr>
<tr>
<td>New Orleans Saints</td>
<td>w 26-14</td>
<td>54000</td>
</tr>
<tr>
<td>New York Giants</td>
<td>w 26-14</td>
<td>54000</td>
</tr>
<tr>
<td>Pittsburgh Steelers</td>
<td>w 17-14</td>
<td>61500</td>
</tr>
</tbody>
</table>

**Statement:** In the 1979 Philadelphia Eagles season, there was an average attendance of 52500 in all winning games.

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
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 (TtT) 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 PNs 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.
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 \textit{TlT}_{noCS}, last row for \textit{TlT}, as introduced in Sect. 5.

<table>
<thead>
<tr>
<th>Subset</th>
<th>Sketch</th>
<th>Full</th>
</tr>
</thead>
<tbody>
<tr>
<td>No content selection (\textit{TlT}_{noCS})</td>
<td>15.0</td>
<td>4.9</td>
</tr>
<tr>
<td>TAB</td>
<td>42.6</td>
<td>27.3</td>
</tr>
<tr>
<td>INF</td>
<td>28.7</td>
<td>11.0</td>
</tr>
<tr>
<td>AUX</td>
<td>14.0</td>
<td>6.2</td>
</tr>
<tr>
<td>TAB, INF</td>
<td>56.5</td>
<td>39.3</td>
</tr>
<tr>
<td>TAB, AUX</td>
<td>44.3</td>
<td>28.6</td>
</tr>
<tr>
<td>TAB, INF, AUX</td>
<td>58.5</td>
<td>38.9</td>
</tr>
<tr>
<td>TAB, INF, AUX + FCR (\textit{TlT})</td>
<td>56.0</td>
<td>46.5</td>
</tr>
</tbody>
</table>

Figure 3: Model configurations used in main experiments.

Following (Guo et al., 2019), Table2Logic performs two decoding iterations. In a first iteration, a \textit{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 \textit{False}, as they would be factually incorrect. Thus, the candidate LF in the beam that executes to \textit{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 \textit{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\textsubscript{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\textsubscript{gold} and TIT\textsubscript{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\textsubscript{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\textsubscript{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\textsubscript{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\textsubscript{noCS} and TIT\textsubscript{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\textsubscript{gold} results. Note that our results for TIT\textsubscript{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...
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.

<table>
<thead>
<tr>
<th>Model</th>
<th>B-4</th>
<th>R-1</th>
<th>R-2</th>
<th>R-L</th>
<th>BERTs</th>
<th>BARTs</th>
</tr>
</thead>
<tbody>
<tr>
<td>T2T_{noCS}</td>
<td>16.8</td>
<td>37.7</td>
<td>19.3</td>
<td>31.6</td>
<td>88.8</td>
<td>-4.04</td>
</tr>
<tr>
<td>TIL_{noCS}</td>
<td>15.6</td>
<td>39.0</td>
<td>18.9</td>
<td>32.2</td>
<td>87.9</td>
<td>-4.03</td>
</tr>
<tr>
<td>T2T</td>
<td>26.8</td>
<td>55.2</td>
<td>31.5</td>
<td>45.7</td>
<td>91.9</td>
<td>-2.98</td>
</tr>
<tr>
<td>TIL (ours)</td>
<td>27.2</td>
<td>56.0</td>
<td>33.1</td>
<td>47.7</td>
<td>92.0</td>
<td>-2.99</td>
</tr>
<tr>
<td>TIL_{gold}</td>
<td>31.7</td>
<td>62.4</td>
<td>38.7</td>
<td>52.8</td>
<td>93.1</td>
<td>-2.65</td>
</tr>
<tr>
<td>TIL_{gold}*</td>
<td>31.4*</td>
<td>64.2*</td>
<td>39.5*</td>
<td>54.0*</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Model</th>
<th>Faithful</th>
<th>Unfaithful</th>
<th>Nonsense</th>
</tr>
</thead>
<tbody>
<tr>
<td>T2T_{noCS}*</td>
<td>20.2*</td>
<td>79.8*</td>
<td>-</td>
</tr>
<tr>
<td>T2T</td>
<td>44.9</td>
<td>49.3</td>
<td>5.8</td>
</tr>
<tr>
<td>TIL (ours)</td>
<td>75.0</td>
<td>20.3</td>
<td>4.7</td>
</tr>
<tr>
<td>TIL_{gold}</td>
<td>82.4</td>
<td>13.51</td>
<td>4.1</td>
</tr>
</tbody>
</table>

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 TIL_{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. TIL);
- **Manual LFs** give 7 points (TIL vs. TIL_{gold});
- **Perfect Logic2Text** generation would yield 18 points (TIL_{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.
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).

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

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

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

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 where “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 where 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.
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., 2020d), 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 incorporates 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 TIT 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 MCI/N AEI 10.13039/501100011033 and by the European Union NextGenerationEU/ PRTR, as well as the Basque Government IT1570-22.
REFERENCES


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 TLT, 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 TLT:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Random seed</td>
<td>90</td>
</tr>
<tr>
<td>Maximum sequence lengthy</td>
<td>512</td>
</tr>
<tr>
<td>Batch size</td>
<td>8</td>
</tr>
<tr>
<td>Epochs</td>
<td>50</td>
</tr>
<tr>
<td>Base learning rate</td>
<td>$5 \times 10^{-5}$</td>
</tr>
<tr>
<td>Connection learning rate</td>
<td>$1 \times 10^{-4}$</td>
</tr>
<tr>
<td>Transformer learning rate</td>
<td>$2 \times 10^{-5}$</td>
</tr>
<tr>
<td>Scheduler gamma</td>
<td>0.5</td>
</tr>
<tr>
<td>ADAM maximum gradient norm</td>
<td>1.0</td>
</tr>
<tr>
<td>Gradient clipping</td>
<td>0.1</td>
</tr>
<tr>
<td>Loss epoch threshold</td>
<td>50</td>
</tr>
<tr>
<td>Sketch loss weight</td>
<td>1.0</td>
</tr>
<tr>
<td>Word embedding size</td>
<td>300</td>
</tr>
<tr>
<td>Size of LSTM hidden states</td>
<td>300</td>
</tr>
<tr>
<td>Attention vector size</td>
<td>300</td>
</tr>
<tr>
<td>Grammar type embedding size</td>
<td>128</td>
</tr>
<tr>
<td>Grammar node embedding size</td>
<td>128</td>
</tr>
<tr>
<td>Column node embedding size</td>
<td>300</td>
</tr>
<tr>
<td>Index node embedding size</td>
<td>300</td>
</tr>
<tr>
<td>Readout</td>
<td>'identity'</td>
</tr>
<tr>
<td>Column attention</td>
<td>'affine'</td>
</tr>
<tr>
<td>Dropout rate</td>
<td>0.3</td>
</tr>
<tr>
<td>Largest index for I nodes</td>
<td>20</td>
</tr>
<tr>
<td>Include OOV token</td>
<td>True</td>
</tr>
<tr>
<td>Beam size</td>
<td>2048</td>
</tr>
<tr>
<td>Max decoding steps</td>
<td>50</td>
</tr>
<tr>
<td>False Candidate Rejection</td>
<td>True</td>
</tr>
</tbody>
</table>
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 | filter_str_eq View C Obj |
       filter_not_eq View C Obj | filter_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 | nth_min View C |
Row ::= argmax View C | argmin View C | nth_argmax View C | nth_argmin View C |
Obj ::= str_hop_first View C | num_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:

<table>
<thead>
<tr>
<th>rank</th>
<th>nation</th>
<th>gold</th>
<th>silver</th>
<th>bronze</th>
<th>total</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>austria</td>
<td>0</td>
<td>0</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>2</td>
<td>italy</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>2</td>
</tr>
<tr>
<td>3</td>
<td>west germany</td>
<td>0</td>
<td>2</td>
<td>0</td>
<td>2</td>
</tr>
<tr>
<td>4</td>
<td>poland</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>5</td>
<td>switzerland</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

Logical Form:

\[
\text{and} \quad \text{only} \quad \text{filter_greater} \quad 0 \quad \text{all_rows} \quad \text{bronze} \\
\text{str_eq} \quad \text{austria} \\
\text{str_hop_first} \quad \text{filter_greater} \quad 0 \quad \text{all_rows} \quad \text{bronze} \quad \text{nation}
\]

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:

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

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:

<table>
<thead>
<tr>
<th>player</th>
<th>nationality</th>
<th>jersey number(s)</th>
<th>position</th>
<th>years</th>
<th>from</th>
</tr>
</thead>
<tbody>
<tr>
<td>craig ehlo</td>
<td>united states</td>
<td>3</td>
<td>sg</td>
<td>1996 - 1997</td>
<td>washington state</td>
</tr>
<tr>
<td>pervis ellison</td>
<td>united states</td>
<td>29</td>
<td>c</td>
<td>2000</td>
<td>louisville</td>
</tr>
<tr>
<td>francisco elson</td>
<td>netherlands</td>
<td>16</td>
<td>c</td>
<td>2008</td>
<td>california</td>
</tr>
<tr>
<td>reggie evans</td>
<td>united states</td>
<td>34, 30</td>
<td>pf</td>
<td>2002 - 2006</td>
<td>iowa</td>
</tr>
<tr>
<td>patrick ewing</td>
<td>united states</td>
<td>33</td>
<td>center</td>
<td>2000 - 2001</td>
<td>georgetown</td>
</tr>
</tbody>
</table>

Logical Form:

```plaintext
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 than pervis 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:

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

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:

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

**TIT Logical Form:**

```
less
├── num_hop_first
│   ├── filter_str_eq
│   │   ├── 1956
│   │   │   └── all_rows
│   │   └── year
│   └── rank
│       └── num_hop_first
│           ├── filter_str_eq
│           │   └── 1959
│           │       └── all_rows
│           └── year
└── laps
```

**Gold Logical Form:**

```
greater
├── num_hop_first
│   ├── filter_str_eq
│   │   └── 1956
│   │       └── all_rows
│   │           └── year
│   └── laps
│       └── num_hop_first
│           ├── filter_str_eq
│           │   └── 1959
│           │       └── all_rows
│           └── year
└── laps
```

**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.
### Table:

<table>
<thead>
<tr>
<th>date</th>
<th>winning team</th>
<th>score</th>
<th>winning pitcher</th>
<th>losing pitcher</th>
<th>attendance</th>
<th>location</th>
</tr>
</thead>
<tbody>
<tr>
<td>May 20</td>
<td>Texas</td>
<td>7 - 3</td>
<td>Kenny Rogers</td>
<td>Brandon Backe</td>
<td>38109</td>
<td>Arlington</td>
</tr>
<tr>
<td>May 21</td>
<td>Texas</td>
<td>18 - 3</td>
<td>Chris Young</td>
<td>Ezequiel Astacio</td>
<td>35781</td>
<td>Arlington</td>
</tr>
<tr>
<td>May 22</td>
<td>Texas</td>
<td>2 - 0</td>
<td>Chan Ho Park</td>
<td>Roy Oswalt</td>
<td>40583</td>
<td>Arlington</td>
</tr>
<tr>
<td>June 24</td>
<td>Houston</td>
<td>5 - 2</td>
<td>Roy Oswalt</td>
<td>Ricardo Rodriguez</td>
<td>36199</td>
<td>Houston</td>
</tr>
<tr>
<td>June 25</td>
<td>Texas</td>
<td>6 - 5</td>
<td>Chris Young</td>
<td>Brandon Backe</td>
<td>41868</td>
<td>Houston</td>
</tr>
</tbody>
</table>

**TIT Logical Form:**

```
most_str_eq
├── all_rows
│   └── Arlington
└── location
```

**Gold Logical Form:**

```
str_eq
├── Arlington
│   └── str_hop
│       └── argmin
│           ├── all_rows
│           └── location
│       └── 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:

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


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.