A Hierarchical Model for Data-to-Text Generation

Clément Rebuffel[1,2(✉)], Laure Soulier[1], Geoffrey Scoutheeten[2], and Patrick Gallinari[1,3]

[1] LIP6, Sorbonne Université, Paris, France
{clement.rebuffel,laure.soulier,patrick.gallinari}@lip6.fr
[2] BNP Paribas, Paris, France
{clement.rebuffel,geoffrey.scoutheeten}@bnpparibas.com
[3] Criteo AI Lab, Paris, France

Abstract. Transcribing structured data into natural language descriptions has emerged as a challenging task, referred to as "data-to-text". These structures generally regroup multiple elements, as well as their attributes. Most attempts rely on translation encoder-decoder methods which linearize elements into a sequence. This however loses most of the structure contained in the data. In this work, we propose to overpass this limitation with a hierarchical model that encodes the data-structure at the element-level and the structure level. Evaluations on RotoWire show the effectiveness of our model w.r.t. qualitative and quantitative metrics.

Keywords: Data-to-text · Hierarchical encoding · Language generation

1 Introduction

Knowledge and/or data is often modeled in a structure, such as indexes, tables, key-value pairs, or triplets. These data, by their nature (e.g., raw data or long time-series data), are not easily usable by humans; outlining their crucial need to be synthesized. Recently, numerous works have focused on leveraging structured data in various applications, such as question answering [24,34] or table retrieval [7,32]. One emerging research field consists in transcribing data-structures into natural language in order to ease their understandablity and their usablity. This field is referred to as "data-to-text" [8] and has its place in several application domains (such as journalism [22] or medical diagnosis [25]) or wide-audience applications (such as financial [26] and weather reports [30], or sport broadcasting [4,39]). As an example, Fig. 1 shows a data-structure containing statistics on NBA basketball games, paired with its corresponding journalistic description.

Designing data-to-text models gives rise to two main challenges: (1) understanding structured data and (2) generating associated descriptions. Recent data-to-text models [18,28,29,39] mostly rely on an encoder-decoder architecture [2] in which the data-structure is first encoded sequentially into a fixed-size

J. M. Jose et al. (Eds.): ECIR 2020, LNCS 12035, pp. 65–80, 2020.
https://doi.org/10.1007/978-3-030-45439-5_5

TEAM	H/V	WINS	LOSSES	PTS	REB	AST	...
Hawks	H	46	12	95	42	27	...
Magic	V	19	41	88	40	22	...

PLAYER	PTS	REB	AST	STL	BLK	CITY	...
Al Horford	17	13	4	2	0	Atlanta	...
Kyle Korver	8	3	2	1	2	Atlanta	...
Jeff Teague	17	0	7	2	0	Atlanta	...
N. Vucevic	21	15	3	1	1	Orlando	...
Tobias Harris	15	4	1	2	1	Orlando	...
...	...	...	...	...	...	...	...

H/V: home or visiting; PTS: points; REB: rebounds; AST: assists; STL: steals; BLK: blocks

The **Atlanta Hawks** (**46-12**) beat the **Orlando Magic** (**19-41**) **95-88** on Friday. **Al Horford** had a good all-around game, putting up **17 points, 13 rebounds, four assists and two steals** in a tough matchup against **Nikola Vucevic**. **Kyle Korver** was the lone Atlanta starter not to reach double figures in points. **Jeff Teague** bounced back from an illness, he scored **17 points** to go along with **seven assists and two steals**. After a rough start to the month, the **Hawks** have won three straight and sit atop the Eastern Conference with a nine game lead on the second place Toronto Raptors. The **Magic** lost in devastating fashion to the Miami Heat in overtime Wednesday. They blew a seven point lead with 43 seconds remaining and they might have carried that with them into Friday's contest against the **Hawks**. **Vucevic** led the **Magic** with **21 points and 15 rebounds**. **Aaron Gordon** (ankle) and **Evan Fournier** (hip) were unable to play due to injury. The **Magic** have four teams between them and the eighth and final playoff spot in the Eastern Conference. The **Magic** will host the Charlotte Hornets on Sunday, and the **Hawks** with take on the Heat in Miami on Saturday.

Fig. 1. Example of structured data from the RotoWire dataset. Rows are entities (either a team or a player) and each cell a record, its key being the column label and its value the cell content. Factual mentions from the table are boldfaced in the description.

vectorial representation by an encoder. Then, a decoder generates words conditioned on this representation. With the introduction of the attention mechanism [19] on one hand, which computes a context focused on important elements from the input at each decoding step and, on the other hand, the copy mechanism [11,33] to deal with unknown or rare words, these systems produce fluent and domain comprehensive texts. For instance, Roberti et al. [31] train a character-wise encoder-decoder to generate descriptions of restaurants based on their attributes, while Puduppully et al. [28] design a more complex two-step decoder: they first generate a plan of elements to be mentioned, and then condition text generation on this plan. Although previous work yield overall good results, we identify two important caveats, that hinder precision (*i.e.* factual mentions) in the descriptions:

1. *Linearization of the data-structure.* In practice, most works focus on introducing innovating decoding modules, and still represent data as a unique sequence of elements to be encoded. For example, the table from Fig. 1 would be linearized to [(Hawks, H/V, H), ..., (Magic, H/V, V), ...], effectively leading to losing distinction between rows, and therefore entities. To the best of our knowledge, only Liu et al. [17,18] propose encoders constrained by the structure but these approaches are designed for single-entity structures.
2. *Arbitrary ordering of unordered collections in recurrent networks (RNN).* Most data-to-text systems use RNNs as encoders (such as GRUs or LSTMs), these architectures have however some limitations. Indeed, they require in practice their input to be fed sequentially. This way of encoding unordered sequences (*i.e.* collections of entities) implicitly assumes an arbitrary order within the collection which, as demonstrated by Vinyals et al. [37], significantly impacts the learning performance.

To address these shortcomings, we propose a new structured-data encoder assuming that structures should be hierarchically captured. Our contribution focuses on the encoding of the data-structure, thus the decoder is chosen to be a classical module as used in [28,39]. Our contribution is threefold:

- We model the general structure of the data using a two-level architecture, first encoding all entities on the basis of their elements, then encoding the data structure on the basis of its entities;
- We introduce the Transformer encoder [36] in data-to-text models to ensure robust encoding of each element/entities in comparison to all others, no matter their initial positioning;
- We integrate a hierarchical attention mechanism to compute the hierarchical context fed into the decoder.

We report experiments on the RotoWire benchmark [39] which contains around $5K$ statistical tables of NBA basketball games paired with human-written descriptions. Our model is compared to several state-of-the-art models. Results show that the proposed architecture outperforms previous models on BLEU score and is generally better on qualitative metrics.

In the following, we first present a state-of-the art of data-to-text literature (Sect. 2), and then describe our proposed hierarchical data encoder (Sect. 3). The evaluation protocol is presented in Sect. 4, followed by the results (Sect. 5). Section 6 concludes the paper and presents perspectives.

2 Related Work

Until recently, efforts to bring out semantics from structured-data relied heavily on expert knowledge [6,30]. For example, in order to better transcribe numerical time series of weather data to a textual forecast, Reiter et al. [30] devise complex template schemes in collaboration with weather experts to build a consistent set of data-to-word rules.

Modern approaches to the wide range of tasks based on structured-data (*e.g.* table retrieval [7,41], table classification [9], question answering [12]) now propose to leverage progress in deep learning to represent these data into a semantic vector space (also called embedding space). In parallel, an emerging task, called "data-to-text", aims at describing structured data into a natural language description. This task stems from the neural machine translation (NMT) domain, and early work [1,15,39] represent the data records as a single sequence of facts to be entirely translated into natural language. Wiseman et al. [39] show the limits of traditional NMT systems on larger structured-data, where NMT systems fail to accurately extract salient elements.

To improve these models, a number of work [16,28,40] proposed innovating decoding modules based on planning and templates, to ensure factual and coherent mentions of records in generated descriptions. For example, Puduppully et al. [28] propose a two-step decoder which first targets specific records and then use them as a plan for the actual text generation. Similarly, Li et al. [16] proposed a delayed copy mechanism where their decoder also acts in two steps: (1) using a classical LSTM decoder to generate delexicalized text and (2) using a pointer network [38] to replace placeholders by records from the input data.

Closer to our work, very recent work [17,18,29] have proposed to take into account the data structure. More particularly, Puduppully et al. [29] follow

entity-centric theories [10,20] and propose a model based on dynamic entity representation at decoding time. It consists in conditioning the decoder on entity representations that are updated during inference at each decoding step. On the other hand, Liu et al. [17,18] rather focus on introducing structure into the encoder. For instance, they propose a dual encoder [17] which encodes separately the sequence of element names and the sequence of element values. These approaches are however designed for single-entity data structures and do not account for delimitation between entities.

Our contribution differs from previous work in several aspects. First, instead of flatly concatenating elements from the data-structure and encoding them as a sequence [18,28,39], we constrain the encoding to the underlying structure of the input data, so that the delimitation between entities remains clear throughout the process. Second, unlike all works in the domain, we exploit the Transformer architecture [36] and leverage its particularity to directly compare elements with each others in order to avoid arbitrary assumptions on their ordering. Finally, in contrast to [5,29] that use a complex updating mechanism to obtain a dynamic representation of the input data and its entities, we argue that explicit hierarchical encoding naturally guides the decoding process via hierarchical attention.

3 Hierarchical Encoder Model for Data-to-Text

In this section we introduce our proposed hierarchical model taking into account the data structure. We outline that the decoding component aiming to generate descriptions is considered as a black-box module so that our contribution is focused on the encoding module. We first describe the model overview, before detailing the hierarchical encoder and the associated hierarchical attention.

3.1 Notation and General Overview

Let's consider the following notations:

- An *entity* e_i is a set of J_i unordered records $\{r_{i,1}, ..., r_{i,j}, ..., r_{i,J_i}\}$; where record $r_{i,j}$ is defined as a pair of *key* $k_{i,j}$ and *value* $v_{i,j}$. We outline that J_i might differ between entities.
- A *data-structure* s is an unordered set of I entities e_i. We thus denote $s \coloneqq \{e_1, ..., e_i, ..., e_I\}$.
- For each data-structure, a textual *description* y is associated. We refer to the first t words of a description y as $y_{1:t}$. Thus, the full sequence of words can be noted as $y = y_{1:T}$.
- The *dataset* $\mathcal{D}$ is a collection of N aligned (data-structure, description) pairs (s, y).

For instance, Fig. 1 illustrates a data-structure associated with a description. The data-structure includes a set of entities (*Hawks, Magic, Al Horford, Jeff Teague, ...*). The entity Jeff Teague is modeled as a set of records {(PTS, 17), (REB, 0), (AST, 7) ...} in which, e.g., the record (PTS, 17) is characterized by a *key* (PTS) and a *value* (17).

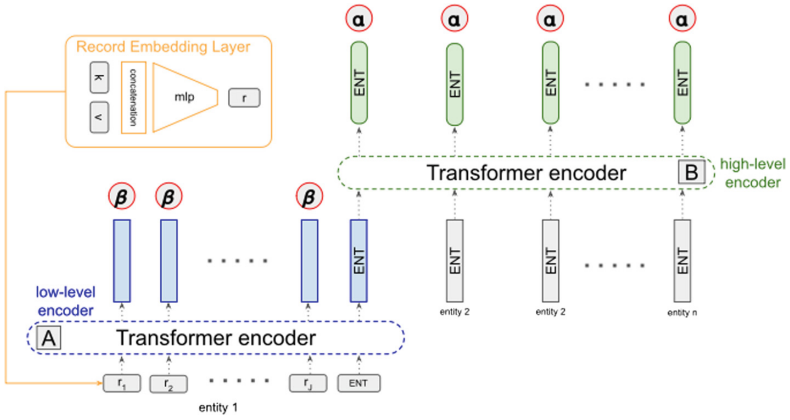

Fig. 2. Diagram of the proposed hierarchical encoder. Once the records are embedded, the low-level encoder works on each entity independently (A); then the high-level encoder encodes the collection of entities (B). In circles, we represent the hierarchical attention scores: the α scores at the entity level and the β scores at the record level.

For each data-structure s in $\mathcal{D}$, the objective function aims to generate a description $\hat{y}$ as close as possible to the ground truth y. This objective function optimizes the following log-likelihood over the whole dataset $\mathcal{D}$:

$$\arg\max_{\theta} \mathcal{L}(\theta) = \arg\max_{\theta} \sum_{(s,y)\in\mathcal{D}} \log P(\hat{y} = y \mid s; \theta) \tag{1}$$

where θ stands for the model parameters and $P(\hat{y} = y \mid s; \theta)$ the probability of the model to generate the adequate description y for table s.

During inference, we generate the sequence $\hat{y}^*$ with the maximum a posteriori probability conditioned on table s. Using the chain rule, we get:

$$\hat{y}^*_{1:T} = \arg\max_{\hat{y}_{1:T}} \prod_{t=1}^{T} P(\hat{y}_t | \hat{y}_{1:t-1}; s; \theta) \tag{2}$$

This equation is intractable in practice, we approximate a solution using beam search, as in [17,18,28,29,39].

Our model follows the encoder-decoder architecture [2]. Because our contribution focuses on the encoding process, we chose the decoding module used in [28,39]: a two-layers LSTM network with a copy mechanism. In order to supervise this mechanism, we assume that each record value that also appears in the target is copied from the data-structure and we train the model to switch between freely generating words from the vocabulary and copying words from the input. We now describe the hierarchical encoder and the hierarchical attention.

3.2 Hierarchical Encoding Model

As outlined in Sect. 2, most previous work [16,28,29,39,40] make use of flat encoders that do not exploit the data structure. To keep the semantics of each element from the data-structure, we propose a hierarchical encoder which relies on two modules. The first one (module A in Fig. 2) is called *low-level encoder* and encodes entities on the basis of their records; the second one (module B), called *high-level encoder*, encodes the data-structure on the basis of its underlying entities. In the low-level encoder, the traditional embedding layer is replaced by a record embedding layer as in [18,28,39]. We present in what follows the record embedding layer and introduce our two hierarchical modules.

Record Embedding Layer. The first layer of the network consists in learning two embedding matrices to embed the record keys and values. Keys $k_{i,j}$ are embedded to $\mathbf{k}_{i,j} \in \mathbb{R}^d$ and values $v_{i,j}$ to $\mathbf{v}_{i,j} \in \mathbb{R}^d$, with d the size of the embedding. As in previous work [18,28,39], each record embedding $\mathbf{r}_{i,j}$ is computed by a linear projection on the concatenation $[\mathbf{k}_{i,j}; \mathbf{v}_{i,j}]$ followed by a non linearity:

$$\mathbf{r}_{i,j} = \text{ReLU}(\mathbf{W}_r[\mathbf{k}_{i,j};\ \mathbf{v}_{i,j}] + \mathbf{b}_r) \tag{3}$$

where $\mathbf{W}_r \in \mathbb{R}^{2d \times d}$ and $\mathbf{b}_r \in \mathbb{R}^d$ are learnt parameters.

The low-level encoder aims at encoding a collection of records belonging to the same entity while the high-level encoder encodes the whole set of entities. Both the low-level and high-level encoders consider their input elements as unordered. We use the Transformer architecture from [36]. For each encoder, we have the following peculiarities:

- the **Low-level encoder** encodes each entity e_i on the basis of its record embeddings $\mathbf{r}_{i,j}$. Each record embedding $\mathbf{r}_{i,j}$ is compared to other record embeddings to learn its final hidden representation $\mathbf{h}_{i,j}$. Furthermore, we add a special record `[ENT]` for each entity, illustrated in Fig. 2 as the last record. Since entities might have a variable number of records, this token allows to aggregate final hidden record representations $\{\mathbf{h}_{i,j}\}_{j=1}^{J_i}$ in a fixed-sized representation vector $\mathbf{h}_i$.
- the **High-level encoder** encodes the data-structure on the basis of its entity representation $\mathbf{h}_i$. Similarly to the **Low-level encoder**, the final hidden state $\mathbf{e_i}$ of an entity is computed by comparing entity representation $\mathbf{h}_i$ with each others. The data-structure representation $\mathbf{z}$ is computed as the mean of these entity representations, and is used for the decoder initialization.

3.3 Hierarchical Attention

To fully leverage the hierarchical structure of our encoder, we propose two variants of hierarchical attention mechanism to compute the context fed to the decoder module.

- *Traditional Hierarchical Attention.* As in [29], we hypothesize that a dynamic context should be computed in two steps: first attending to entities,

then to records corresponding to these entities. To implement this hierarchical attention, at each decoding step t, the model learns a first set of attention scores $\alpha_{i,t}$ over entities e_i and a second set of attention scores $\beta_{i,j,t}$ over records $r_{i,j}$ belonging to entity e_i. The $\alpha_{i,t}$ scores are normalized to form a distribution over all entities e_i, and $\beta_{i,j,t}$ scores are normalized to form a distribution over records $r_{i,j}$ of entity e_i. Each entity is then represented as a weighted sum of its record embeddings, and the entire data structure is represented as a weighted sum of the entity representations. The dynamic context is computed as:

$$\mathbf{c_t} = \sum_{i=1}^{I} (\alpha_{i,t} (\sum_{j} \beta_{i,j,t} \mathbf{r}_{i,j})) \tag{4}$$

$$where \;\; \alpha_{i,t} \propto exp(\mathbf{d}_t \mathbf{W}_\alpha \mathbf{e}_i) \;\; and \;\; \beta_{i,j,t} \propto exp(\mathbf{d}_t \mathbf{W}_\beta \mathbf{h}_{i,j}) \tag{5}$$

where $\mathbf{d_t}$ is the decoder hidden state at time step t, $\mathbf{W}_\alpha \in \mathbb{R}^{d \times d}$ and $\mathbf{W}_\beta \in \mathbb{R}^{d \times d}$ are learnt parameters, $\sum_i \alpha_{i,t} = 1$, and for all $i \in \{1, ..., I\}$ $\sum_j \beta_{i,j,t} = 1$.

- *Key-guided Hierarchical Attention.* This variant follows the intuition that once an entity is chosen for mention (thanks to $\alpha_{i,t}$), only the type of records is important to determine the content of the description. For example, when deciding to mention a player, all experts automatically report his score without consideration of its specific value. To test this intuition, we model the attention scores by computing the $\beta_{i,j,t}$ scores from Eq. (5) solely on the embedding of the *key* rather than on the full record representation $\mathbf{h}_{i,j}$:

$$\hat{\beta}_{i,j,t} \propto exp(\mathbf{d}_t \mathbf{W}_{a_2} \mathbf{k}_{i,j}) \tag{6}$$

Please note that the different embeddings and the model parameters presented in the model components are learnt using Eq. 1.

4 Experimental Setup

4.1 The Rotowire Dataset

To evaluate the effectiveness of our model, and demonstrate its flexibility at handling heavy data-structure made of several types of entities, we used the RotoWire dataset [39]. It includes basketball games statistical tables paired with journalistic descriptions of the games, as can be seen in the example of Fig. 1. The descriptions are professionally written and average 337 words with a vocabulary size of 11.3K. There are 39 different record keys, and the average number of records (resp. entities) in a single data-structure is 628 (resp. 28). Entities are of two types, either team or player, and player descriptions depend on their involvement in the game. We followed the data partitions introduced with the dataset and used a train/validation/test sets of respectively 3,398/727/728 (data-structure, description) pairs.

4.2 Evaluation Metrics

We evaluate our model through two types of metrics. The BLEU score [23] aims at measuring to what extent the generated descriptions are literally closed to the ground truth. The second category designed by [39] is more qualitative.

BLEU Score. The **BLEU score** [23] is commonly used as an evaluation metric in text generation tasks. It estimates the correspondence between a machine output and that of a human by computing the number of co-occurrences for ngrams ($n \in 1, 2, 3, 4$) between the generated candidate and the ground truth. We use the implementation code released by [27].

Information Extraction-Oriented Metrics. These metrics estimate the ability of our model to integrate elements from the table in its descriptions. Particularly, they compare the gold and generated descriptions and measure to what extent the extracted relations are aligned or differ. To do so, we follow the protocol presented in [39]. First, we apply an information extraction (IE) system trained on labeled relations from the gold descriptions of the RotoWire train dataset. Entity-value pairs are extracted from the descriptions. For example, in the sentence *Isaiah Thomas led the team in scoring, totaling 23 points [...].*, an IE tool will extract the pair (Isaiah Thomas, 23, PTS). Second, we compute three metrics on the extracted information:

- **Relation Generation (RG)** estimates how well the system is able to generate text containing factual (i.e., correct) records. We measure the precision and absolute number (denoted respectively RG-P% and RG-#) of unique relations r extracted from $\hat{y}_{1:T}$ that also appear in s.

- **Content Selection (CS)** measures how well the generated document matches the gold document in terms of mentioned records. We measure the precision and recall (denoted respectively CS-P% and CS-R%) of unique relations r extracted from $\hat{y}_{1:T}$ that are also extracted from $y_{1:T}$.

- **Content Ordering (CO)** analyzes how well the system orders the records discussed in the description. We measure the normalized Damerau-Levenshtein distance [3] between the sequences of records extracted from $\hat{y}_{1:T}$ that are also extracted from $y_{1:T}$.

CS primarily targets the "what to say" aspect of evaluation, CO targets the "how to say it" aspect, and RG targets both. Note that for CS, CO, RG-% and BLEU metrics, higher is better; which is not true for RG-#. The IE system used in the experiments is able to extract an average of 17 factual records from gold descriptions. In order to mimic a human expert, a generative system should approach this number and not overload generation with brute facts.

4.3 Baselines

We compare our hierarchical model against three systems. For each of them, we report the results of the best performing models presented in each paper.

- *Wiseman* [39] is a standard encoder-decoder system with copy mechanism.
- *Li* [16] is a standard encoder-decoder with a delayed copy mechanism: text is first generated with placeholders, which are replaced by salient records extracted from the table by a pointer network.
- *Puduppully-plan* [28] acts in two steps: a first standard encoder-decoder generates a plan, *i.e.* a list of salient records from the table; a second standard encoder-decoder generates text from this plan.
- *Puduppully-updt* [29]. It consists in a standard encoder-decoder, with an added module aimed at updating record representations during the generation process. At each decoding step, a gated recurrent network computes which records should be updated and what should be their new representation.

Model Scenarios. We test the importance of the input structure by training different variants of the proposed architecture:

- *Flat*, where we feed the input sequentially to the encoder, losing all notion of hierarchy. As a consequence, the model uses standard attention. This variant is closest to *Wiseman*, with the exception that we use a Transformer to encode the input sequence instead of an RNN.
- *Hierarchical-kv* is our full hierarchical model, with traditional hierarchical attention, *i.e.* where attention over records is computed on the full record encoding, as in Eq. (5).
- *Hierarchical-k* is our full hierarchical model, with key-guided hierarchical attention, *i.e.* where attention over records is computed only on the record key representations, as in Eq. (6).

Table 1. Evaluation on the RotoWire testset using relation generation (RG) count (#) and precision (P%), content selection (CS) precision (P%) and recall (R%), content ordering (CO), and BLEU. –: number of parameters unavailable.

	BLEU	RG		CS			CO	Nb
		P%	#	P%	R%	F1		Params
Gold descriptions	100	96.11	17.31	100	100	100	100	
Wiseman	14.5	75.62	**16.83**	32.80	39.93	36.2	15.62	45M
Li	16.19	84.86	19.31	30.81	38.79	34.34	16.34	–
Pudupully-plan	16.5	87.47	34.28	34.18	51.22	41	18.58	35M
Puduppully-updt	16.2	**92.69**	30.11	38.64	48.51	43.01	**20.17**	23M
Flat	$16.7_{.2}$	76.62_{1}	$18.54_{.6}$	$31.67_{.7}$	42.9_{1}	$36.42_{.4}$	$14.64_{.3}$	14M
Hierarchical-kv	$17_{.3}$	89.04_{1}	$21.46_{.9}$	$38.57_{1.2}$	$51.50_{.9}$	$44.19_{.7}$	$18.70_{.7}$	14M
Hierarchical-k	$\mathbf{17.5}_{.3}$	$89.46_{1.4}$	$21.17_{1.4}$	$\mathbf{39.47}_{1.4}$	$\mathbf{51.64}_{1}$	$\mathbf{44.7}_{.6}$	$18.90_{.7}$	14M

4.4 Implementation Details

The decoder is the one used in [28,29,39] with the same hyper-parameters. For the encoder module, both the low-level and high-level encoders use a two-layers multi-head self-attention with two heads. To fit with the small number of record keys in our dataset (39), their embedding size is fixed to 20. The size of the record value embeddings and hidden layers of the Transformer encoders are both set to 300. We use dropout at rate 0.5. The models are trained with a batch size of 64. We follow the training procedure in [36] and train the model for a fixed number of 25K updates, and average the weights of the last 5 checkpoints (at every 1K updates) to ensure more stability across runs. All models were trained with the Adam optimizer [13]; the initial learning rate is 0.001, and is reduced by half every 10K steps. We used beam search with beam size of 5 during inference. All the models are implemented in OpenNMT-py [14]. All code is available at https://github.com/KaijuML/data-to-text-hierarchical.

5 Results

Our results on the RotoWire testset are summarized in Table 1. For each proposed variant of our architecture, we report the mean score over ten runs, as well as the standard deviation in subscript. Results are compared to baselines [28,29,39] and variants of our models. We also report the result of the oracle (metrics on the gold descriptions). Please note that gold descriptions trivially obtain 100% on all metrics expect RG, as they are all based on comparison with themselves. RG scores are different, as the IE system is imperfect and fails to extract accurate entities 4% of the time. RG-# is an absolute count.

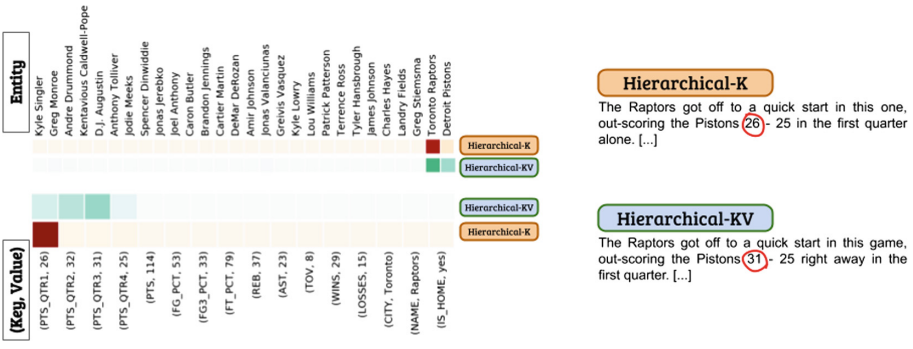

Fig. 3. Right: Comparison of a generated sentence from *Hierarchical-k* and *Hierarchical-kv*. Left: Attention scores over entities (top) and over records inside the selected entity (bottom) for both variants, during the decoding of respectively 26 or 31 (circled in red). (Color figure online)

Ablation Studies. To evaluate the impact of our model components, we first compare scenarios *Flat*, *Hierarchical-k*, and *Hierarchical-kv*. As shown in Table 1, we can see the lower results obtained by the *Flat* scenario compared to the other scenarios (*e.g.* BLEU 16.7 vs. 17.5 for resp. *Flat* and *Hierarchical-k*), suggesting the effectiveness of encoding the data-structure using a hierarchy. This is expected, as losing explicit delimitation between entities makes it harder a) for the encoder to encode semantics of the objects contained in the table and b) for the attention mechanism to extract salient entities/records.

Second, the comparison between scenario *Hierarchical-kv* and *Hierarchical-k* shows that omitting entirely the influence of the record values in the attention mechanism is more effective: this last variant performs slightly better in all metrics excepted CS-R%, reinforcing our intuition that focusing on the structure modeling is an important part of data encoding as well as confirming the intuition explained in Sect. 3.3: once an entity is selected, facts about this entity are relevant based on their key, not value which might add noise. To illustrate this intuition, we depict in Fig. 3 attention scores (recall $\alpha_{i,t}$ and $\beta_{i,j,t}$ from Eqs. (5) and (6)) for both variants *Hierarchical-kv* and *Hierarchical-k*. We particularly focus on the timestamp where the models should mention the number of points scored during the first quarter of the game. Scores of *Hierarchical-k* are sharp, with all of the weight on the correct record (PTS_QTR1, 26) whereas scores of *Hierarchical-kv* are more distributed over all PTS_QTR records, ultimately failing to retrieve the correct one.

Comparison w.r.t. Baselines. From a general point of view, we can see from Table 1 that our scenarios obtain significantly higher results in terms of BLEU over all models; our best model *Hierarchical-k* reaching 17.5 vs. 16.5 against the best baseline. This means that our models learns to generate fluent sequences of words, close to the gold descriptions, adequately picking up on domain lingo. Qualitative metrics are either better or on par with baselines. We show in Fig. 4 a text generated by our best model, which can be directly compared to the gold

The **Atlanta Hawks** (46 - 12) defeated the **Orlando Magic** (19 - 41) 95 - 88 on Monday at Philips Arena in Atlanta. The **Hawks** got out to a quick start in this one, out - scoring the **Magic** 28 - 16 in the first quarter alone. Along with the quick start, the **Hawks** were able to hold off the **Magic** late in the fourth quarter, out - scoring the **Magic** 19 - 21. The **Hawks** were led by **Nikola Vucevic**, who went 10 - for - 16 from the field and 0 - for - 0 from the three-point line to score a team - high of 21 points, while also adding 15 rebounds in 37 minutes. It was his second double - double in a row, a stretch where he's averaging 22 points and 17 rebounds. Notching a double - double of his own, **Al Horford** recorded 17 points (7 - 9 FG , 0 - 0 3Pt , 3 - 4 FT), 13 rebounds and four steals. He's now averaging 15 points and 6 rebounds on the year. **Paul Millsap** had a strong showing , posting 20 points (8 - 17 FG , 4 - 7 3Pt , 0 - 2 FT), four rebounds and three blocked shots. He's been a pleasant surprise for the **Magic** in the second half, as he's averaged 14 points and 5 rebounds over his last three games. **DeMarre Carroll** was the other starter in double figures, finishing with 15 points (6 - 12 FG , 3 - 6 3Pt), eight rebounds and three steals. He's had a nice stretch of three games , averaging 24 points, 3 rebounds and 2 assists over that span. **Tobias Harris** was the only other **Magic** player to reach double figures, scoring 15 points (5 - 9 FG , 2 - 4 3Pt , 3 - 4 FT). The **Magic** 's next game will be at home against the Miami Heat on Wednesday, while the **Magic** will travel to Charlotte to play the Hornets on Wednesday.

Fig. 4. Text generated by our best model. Entites are boldfaced, factual mentions are in green, erroneous mentions in red and hallucinations are in blue. (Color figure online)

description in Fig. 1. Generation is fluent and contains domain-specific expressions. As reflected in Table 1, the number of correct mentions (in green) outweighs the number of incorrect mentions (in red). Please note that, as in previous work [16,28,29,39], generated texts still contain a number of incorrect facts, as well hallucinations (in blue): sentences that have no basis in the input data (e.g. *"[...] he's now averaging 22 points [...]."*). While not the direct focus of our work, this highlights that any operation meant to enrich the semantics of structured data can also enrich the data with incorrect facts.

Specifically, regarding all baselines, we can outline the following statements.

• Our hierarchical models achieve significantly better scores on all metrics when compared to the flat architecture *Wiseman*, reinforcing the crucial role of structure in data semantics and saliency. The analysis of RG metrics shows that *Wiseman* seems to be the more naturalistic in terms of number of factual mentions (RG#) since it is the closest scenario to the gold value (16.83 vs. 17.31 for resp. *Wiseman* and *Hierarchical-k*). However, *Wiseman* achieves only 75.62% of precision, effectively mentioning on average a total of 22.25 records (wrong or accurate), where our model *Hierarchical-k* scores a precision of 89.46%, leading to 23.66 total mentions, just slightly above *Wiseman*.

• The comparison between the *Flat* scenario and *Wiseman* is particularly interesting. Indeed, these two models share the same intuition to flatten the data-structure. The only difference stands on the encoder mechanism: bi-LSTM vs. Transformer, for *Wiseman* and *Flat* respectively. Results shows that our *Flat* scenario obtains a significant higher BLEU score (16.7 vs. 14.5) and generates fluent descriptions with accurate mentions (RG-P%) that are also included in the gold descriptions (CS-R%). This suggests that introducing the Transformer architecture is promising way to implicitly account for data structure.

• Our hierarchical models outperform the two-step decoders of *Li* and *Puduppully-plan* on both BLEU and all qualitative metrics, showing that capturing structure in the encoding process is more effective that predicting a structure in the decoder (i.e., planning or templating). While our models sensibly outperform in precision at factual mentions, the baseline *Puduppully-plan* reaches 34.28 mentions on average, showing that incorporating modules dedicated to entity extraction leads to over-focusing on entities; contrasting with our models that learn to generate more balanced descriptions.

• The comparison with *Puduppully-updt* shows that dynamically updating the encoding across the generation process can lead to better Content Ordering (CO) and RG-P%. However, this does not help with Content Selection (CS) since our best model *Hierarchical-k* obtains slightly better scores. Indeed, *Puduppully-updt* updates representations after each mention allowing to keep track of the mention history. This guides the ordering of mentions (CO metric), each step limiting more the number of candidate mentions (increasing RG-P%). In contrast, our model encodes saliency among records/entities more effectively (CS metric). We note that while our model encodes the data-structure once and for all, *Puduppully-updt* recomputes, via the updates, the encoding at each step and therefore significantly increases computation complexity. Combined with their

RG-# score of 30.11, we argue that our model is simpler, and obtains fluent description with accurate mentions in a more human-like fashion.

We would also like to draw attention to the number of parameters used by those architectures. We note that our scenarios relies on a lower number of parameters (14 millions) compared to all baselines (ranging from 23 to 45 millions). This outlines the effectiveness in the design of our model relying on a structure encoding, in contrast to other approach that try to learn the structure of data/descriptions from a linearized encoding.

6 Conclusion and Future Work

In this work we have proposed a hierarchical encoder for structured data, which (1) leverages the structure to form efficient representation of its input; (2) has strong synergy with the hierarchical attention of its associated decoder. This results in an effective and more light-weight model. Experimental evaluation on the RotoWire benchmark shows that our model outperforms competitive baselines in terms of BLEU score and is generally better on qualitative metrics. This way of representing structured databases may lead to automatic inference and enrichment, e.g., by comparing entities. This direction could be driven by very recent operation-guided networks [21,35]. In addition, we note that our approach can still lead to erroneous facts or even hallucinations. An interesting perspective might be to further constrain the model on the data structure in order to prevent inaccurate of even contradictory descriptions.

Acknowledgements. We would like to thank the H2020 project AI4EU (825619) which partially supports Laure Soulier and Patrick Gallinari.

References

1. Agarwal, S., Dymetman, M.: A surprisingly effective out-of-the-box char2char model on the E2E NLG challenge dataset. In: Proceedings of the 18th Annual SIGdial Meeting on Discourse and Dialogue, Saarbrücken, Germany, 15–17 August 2017, pp. 158–163 (2017). https://www.aclweb.org/anthology/W17-5519/
2. Bahdanau, D., Cho, K., Bengio, Y.: Neural machine translation by jointly learning to align and translate (2014). http://arxiv.org/abs/1409.0473, cite arxiv:1409.0473Comment. Accepted at ICLR 2015 as oral presentation
3. Brill, E., Moore, R.C.: An improved error model for noisy channel spelling correction. In: Proceedings of the 38th Annual Meeting on Association for Computational Linguistics (ACL 2000), pp. 286–293. Association for Computational Linguistics, Stroudsburg, PA, USA (2000). https://doi.org/10.3115/1075218.1075255
4. Chen, D.L., Mooney, R.J.: Learning to sportscast: a test of grounded language acquisition. In: Proceedings of the 25th International Conference on Machine Learning (ICML 2008), pp. 128–135. ACM, New York (2008). https://doi.org/10.1145/1390156.1390173

5. Clark, E., Ji, Y., Smith, N.A.: Neural text generation in stories using entity representations as context. In: Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers), pp. 2250–2260. Association for Computational Linguistics, New Orleans, Louisiana, June 2018. https://doi.org/10.18653/v1/N18-1204. https://www.aclweb.org/anthology/N18-1204
6. Deng, D., Jiang, Y., Li, G., Li, J., Yu, C.: Scalable column concept determination for web tables using large knowledge bases. In: Proceedings of the VLDB Endowment, vol. 6, no. 13, pp. 1606–1617, August 2013. https://doi.org/10.14778/2536258.2536271. http://dl.acm.org/citation.cfm?doid=2536258.2536271
7. Deng, L., Zhang, S., Balog, K.: Table2Vec: neural word and entity embeddings for table population and retrieval. In: Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR 2019), pp. 1029–1032. ACM Press, Paris (2019). https://doi.org/10.1145/3331184.3331333. http://dl.acm.org/citation.cfm?doid=3331184.3331333
8. Gatt, A., Krahmer, E.: Survey of the state of the art in natural language generation: core tasks, applications and evaluation. J. Artif. Int. Res. **61**(1), 65–170 (2018). http://dl.acm.org/citation.cfm?id=3241691.3241693
9. Ghasemi-Gol, M., Szekely, P.A.: TabVec: table vectors for classification of web tables. CoRR abs/1802.06290 (2018). http://arxiv.org/abs/1802.06290
10. Grosz, B., Joshi, A., Weinstein, S.: Centering: a framework for modelling the coherence of discourse. Technical Reports (CIS), January 1995
11. Gulcehre, C., Ahn, S., Nallapati, R., Zhou, B., Bengio, Y.: Pointing the unknown words. In: Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pp. 140–149. Association for Computational Linguistics, Berlin, Germany, August 2016. https://doi.org/10.18653/v1/P16-1014
12. Haug, T., Ganea, O.-E., Grnarova, P.: Neural multi-step reasoning for question answering on semi-structured tables. In: Pasi, G., Piwowarski, B., Azzopardi, L., Hanbury, A. (eds.) ECIR 2018. LNCS, vol. 10772, pp. 611–617. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-76941-7_52
13. Kingma, D.P., Ba, J.: Adam: a method for stochastic optimization (2014). http://arxiv.org/abs/1412.6980, cite arxiv:1412.6980Comment. Published as a conference paper at the 3rd International Conference for Learning Representations, San Diego (2015)
14. Klein, G., Kim, Y., Deng, Y., Senellart, J., Rush, A.M.: OpenNMT: open-source toolkit for neural machine translation. In: Proceedings of the ACL (2017). https://doi.org/10.18653/v1/P17-4012.
15. Lebret, R., Grangier, D., Auli, M.: Neural text generation from structured data with application to the biography domain. In: Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, pp. 1203–1213. Association for Computational Linguistics, Austin, Texas, November 2016. https://doi.org/10.18653/v1/D16-1128. https://www.aclweb.org/anthology/D16-1128
16. Li, L., Wan, X.: 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. Association for Computational Linguistics, Santa Fe, New Mexico, USA, August 2018
17. Liu, T., Luo, F., Xia, Q., Ma, S., Chang, B., Sui, Z.: Hierarchical encoder with auxiliary supervision for neural table-to-text generation: learning better representation for tables. In: Proceedings of the AAAI Conference on Artificial Intelligence, vol. 33, pp. 6786–6793, July 2019. https://doi.org/10.1609/aaai.v33i01.33016786

18. Liu, T., Wang, K., Sha, L., Chang, B., Sui, Z.: Table-to-text generation by structure-aware Seq2seq learning. In: AAAI (2018)
19. Luong, T., Pham, H., Manning, C.D.: Effective approaches to attention-based neural machine translation. In: Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, pp. 1412–1421. Association for Computational Linguistics, Lisbon, Portugal, September 2015. https://doi.org/10.18653/v1/D15-1166. https://www.aclweb.org/anthology/D15-1166
20. Mann, W.C., Thompson, S.A.: Rhetorical structure theory: toward a functional theory of text organization. Text - Interdisc. J. Study Discourse **8**, 243–281 (1988)
21. Nie, F., Wang, J., Yao, J., Pan, R., Lin, C.: Operation-guided neural networks for high fidelity data-to-text generation. In: Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, Brussels, Belgium, 31 October–4 November 2018, pp. 3879–3889 (2018). https://www.aclweb.org/anthology/D18-1422/
22. Oremus, W.: The First News Report on the L.A. Earthquake Was Written by a Robot (2014). https://slate.com/technology/2014/03/quakebot-los-angeles-times-robot-journalist-writes-article-on-la-earthquake.html
23. Papineni, K., Roukos, S., Ward, T., Zhu, W.J.: BLEU: a method for automatic evaluation of machine translation. In: Proceedings of the 40th Annual Meeting on Association for Computational Linguistics (ACL 2002), pp. 311–318. Association for Computational Linguistics, Stroudsburg, PA, USA (2002). https://doi.org/10.3115/1073083.1073135
24. Pasupat, P., Liang, P.: Compositional semantic parsing on semi-structured tables. In: Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers), pp. 1470–1480. Association for Computational Linguistics, Beijing, China, July 2015. https://doi.org/10.3115/v1/P15-1142. https://www.aclweb.org/anthology/P15-1142
25. Pauws, S., Gatt, A., Krahmer, E., Reiter, E.: Making effective use of healthcare data using data-to-text technology: methodologies and applications. Data Science for Healthcare, pp. 119–145. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-05249-2_4
26. Plachouras, V., et al.: Interacting with financial data using natural language. In: Proceedings of the 39th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR 2016), pp. 1121–1124. ACM, New York (2016). https://doi.org/10.1145/2911451.2911457
27. Post, M.: A call for clarity in reporting BLEU scores. In: Proceedings of the Third Conference on Machine Translation: Research Papers, pp. 186–191. Association for Computational Linguistics, Belgium, Brussels, October 2018. https://doi.org/10.18653/v1/W18-6319. https://www.aclweb.org/anthology/W18-6319
28. Puduppully, R., Dong, L., Lapata, M.: Data-to-text generation with content selection and planning. In: AAAI (2018)
29. Puduppully, R., Dong, L., Lapata, M.: Data-to-text generation with entity modeling. In: Proceedings of the 57th Conference of the Association for Computational Linguistics (ACL 2019), Florence, Italy, 28 July–2 August 2019, Volume 1: Long Papers, pp. 2023–2035 (2019). https://www.aclweb.org/anthology/P19-1195/
30. Reiter, E., Sripada, S., Hunter, J., Yu, J., Davy, I.: Choosing words in computer-generated weather forecasts. Artif. Intell. **167**(1–2), 137–169 (2005). https://doi.org/10.1016/j.artint.2005.06.006

31. Roberti, M., Bonetta, G., Cancelliere, R., Gallinari, P.: Copy mechanism and tailored training for character-based data-to-text generation. CoRR abs/1904.11838 (2019). http://arxiv.org/abs/1904.11838
32. Sarma, A.D., et al.: Finding related tables. In: SIGMOD (2012). http://i.stanford.edu/~anishds/publications/sigmod12/modi255i-dassarma.pdf
33. See, A., Liu, P.J., Manning, C.D.: Get to the point: summarization with pointer-generator networks. In: Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pp. 1073–1083. Association for Computational Linguistics, Vancouver, Canada, July 2017. https://doi.org/10.18653/v1/P17-1099
34. Sun, H., Ma, H., He, X., Yih, W.T., Su, Y., Yan, X.: Table cell search for question answering. In: Proceedings of the 25th International Conference on World Wide Web (WWW 2016), pp. 771–782. ACM Press (2016)
35. Trask, A., Hill, F., Reed, S.E., Rae, J.W., Dyer, C., Blunsom, P.: Neural arithmetic logic units. CoRR abs/1808.00508 (2018). http://dblp.uni-trier.de/db/journals/corr/corr1808.html#abs-1808-00508
36. Vaswani, A., et al.: Attention is all you need. In: Proceedings of the 31st International Conference on Neural Information Processing Systems (NIPS 2017), pp. 6000–6010. Curran Associates Inc., USA (2017). http://dl.acm.org/citation.cfm?id=3295222.3295349
37. Vinyals, O., Bengio, S., Kudlur, M.: Order matters: Sequence to sequence for sets. In: International Conference on Learning Representations (ICLR) (2016). http://arxiv.org/abs/1511.06391
38. Vinyals, O., Fortunato, M., Jaitly, N.: Pointer networks. In: Cortes, C., Lawrence, N.D., Lee, D.D., Sugiyama, M., Garnett, R. (eds.) Advances in Neural Information Processing Systems 28, pp. 2692–2700. Curran Associates, Inc. (2015). http://papers.nips.cc/paper/5866-pointer-networks.pdf
39. Wiseman, S., Shieber, S., Rush, A.: Challenges in data-to-document generation. In: Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, pp. 2253–2263. Association for Computational Linguistics, Copenhagen, Denmark, September 2017. https://doi.org/10.18653/v1/D17-1239. https://www.aclweb.org/anthology/D17-1239
40. Wiseman, S., Shieber, S., Rush, A.: Learning neural templates for text generation. In: Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pp. 3174–3187. Association for Computational Linguistics, Brussels, Belgium, October–November 2018. https://doi.org/10.18653/v1/D18-1356. https://www.aclweb.org/anthology/D18-1356
41. Zhang, S., Balog, K.: Web table extraction, retrieval and augmentation. In: Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR 2019), pp. 1409–1410. ACM Press (2019)