Enhancing factualness and controllability of Data-to-Text Generation via data Views and constraints

Craig Thomson[1*], **Clément Rebuffel**[2*],
Ehud Reiter[1], **Laure Soulier**[2], **Somayajulu Sripada**[1], **Patrick Gallinari**[2,3]
[1]University of Aberdeen, UK
[2]LIP6, Sorbonne Université, Paris, France
[3]Criteo AI Lab, Paris, France

Abstract

Neural data-to-text systems lack the control and factual accuracy required to generate useful and insightful summaries of multidimensional data. We propose a solution in the form of *data views*, where each view describes an entity and its attributes along specific dimensions. A sequence of views can then be used as a high-level schema for document planning, with the neural model handling the complexities of micro-planning and surface realization. We show that our view-based system retains factual accuracy while offering high-level control of output that can be tailored based on user preference or other norms within the domain.

1 Introduction

The original vision of data-to-text generation was to take complex data and describe it using natural language such that humans could better understand it (Reiter et al., 2005; Reiter, 2007). Neural data-to-text systems commonly transcribe an isolated, simple data structure to a natural language. However, in many domains, e.g., finance, cinema box-office, or weather, the structured-data to be described are not independent, but rather exist as points along multi-dimensional axes such as time or entities (like people, companies, or locations). Figure 1 shows an example of a human-authored basketball game summary that requires data from outwith an individual game record. Such summaries often mention the upcoming games for each team (e.g. last two sentences) and aggregate player statistics over prior games.

Commercial applications deal with this complex scenario using hand-crafted rules (Reiter and Dale, 1997; Teixeira et al., 2020; Dale, 2020), although recent research leveraging Deep Learning techniques has looked to alleviate this burden. However, such systems remain largely end-to-end, offering little user specified control[1]. On small sets of triples or attributes (Gardent et al., 2017; Dušek et al., 2018), neural systems can produce fluent, in domain generations, although they can struggle with factual accuracy mistakes such as hallucination. Additional challenges are introduced when using complex data such as tables (Parikh et al., 2020), compounded when considering interlinked structured data and longer texts (Wiseman et al., 2017).

While challenging, this complex setting is also an opportunity. In this paper, we propose the novel concept of data views, where all available data is divided into manageable subsets, describing entities and their attributes along one or more dimensions. Views are then aligned with corresponding spans of text. They can then be combined to form high level document plans (*schema*) for neural data-to-text systems, controlling the generation of text without sacrificing factual accuracy. We investigate the use of views in the domain of automated journalism (English language basketball game summaries), using the SportSett dataset (Thomson et al., 2020a) which extends the RotoWire dataset (Wiseman et al., 2017).

2 Related Work

Factualness and controllability are critical issues for data-to-text systems. Studies have shown that for end users of systems, accuracy is more of a concern than readability (Law et al., 2005) and users prefer texts tailored to their needs (van der Lee et al., 2017; Gatt and Krahmer, 2018).

Recently, the research community has focused on neural approaches, aiming to solve data-to-text tasks by leveraging advances of deep learning in language modeling fields (in particular neural machine translation) (Bahdanau et al., 2015; Vaswani

*These authors contributed equally to this work. Corresponding author: Craig Thomson (c.thomson@abdn.ac.uk, c.thomson.nlp@gmail.com)

[1]The INLG2021 panel 'What users want from real-world NLG' highlighted the need for system control.

<Whole-Game=T_{G1}+T_{G2}> The Oklahoma City Thunder defeated the host Miami Heat, 118-102, at American Airlines Arena on Friday. **<Within-Game=T_{E1}+T_{E2}>** While this wasn't a wire-to-wire win for Oklahoma City, they won this game in dominating fashion. **<Within-Game=T_{E1}+T_{E2}>** In fact, a 31-24 first quarter really set the tone, with a 41-29 second quarter sealing the victory. **<Within-Game=T_{E1}+T_{E2}>** The Thunder actually led by at least 15 points for the entirety of the second half. **<Whole-Game=T_{G1}+T_{G2}>** Three-point shooting was the key difference, with Oklahoma City hitting 16-of-30 and Miami connecting on 11-of-36. **<Whole-Game=T_{G1}+T_{G2}>** The Thunder also dominated in transition, winning the fastbreak differential, 23-8. **<Whole-Game=P_{G3}>** The Thunder (33-18) were led by Paul George, as he tallied 43 points, seven rebounds, five assists and two steals. **<Whole-Game=P_{G4}>** Russell Westbrook collected 14 points, 12 rebounds and 14 assists. **<Whole-Game=P_{G5}>** Steven Adams accrued 13 points, seven rebounds, two assists and three steals. **<Whole-Game=P_{G6}>** Dennis Schroder was huge off the bench, as he provided 28 points on 11-of-13 from the field. **<Whole-Game=P_{G16}>** The Heat (24-26) were led by Kelly Olynyk, as he provided 21 points, seven rebounds and two assists off the bench. **<Whole-Game=P_{G17}>** Josh Richardson led the starters with 18 points, four rebounds, three assists and two steals. **<Whole-Game=P_{G18}>** Hassan Whiteside amassed 12 points and 16 rebounds. **<Whole-Game=P_{G19}>** Justise Winslow finished with 10 points, two rebounds and five assists. **<Between-Game=T_{T2}>** Oklahoma City returns to action on Sunday, as they travel to face the Boston Celtics. **<Between-Game=T_{T1}>** As for Miami, they play host to the struggling Indiana Pacers on Saturday for their next outing.

Figure 1: Human authored summary for OKC@MIA on February 1st 2019. Tags and colours such as **<Whole-Game=T_{G1}+T_{G2}>** map to views (or unions of), some examples of which are shown in Figure 2

ID	Team Name	PTS	REB	Wins	Losses	…
T_{G1}	Miami Heat	102	47	24	26	…
T_{G2}	Oklahoma City Thunder	118	50	33	18	…

Team Whole-GameViews (partial)

ID	Name	PTS	REB	AST	STL	BLK	…
P_{G3}	Paul George	43	7	5	2	0	…
P_{G4}	Russel Westbrook	14	12	14	1	1	…

Player Whole-GameViews (partial)

ID	Team Name	H1_PTS	Q1_PTS	Q2_PTS	…
T_{E1}	Miami Heat	53	24	29	…
T_{E2}	Oklahoma City Thunder	72	31	41	…

Team Within-GameViews (partial)

ID	Name	Q1_PTS	Q1_REB	Q2_PTS	Q2_REB	…
P_{E3}	Paul George	16	1	10	1	…
P_{E4}	Russel Westbrook	5	5	0	4	…

Player Within-GameViews (partial)

ID	Team Name	Opp_Place	Opp_Name	Location	…
T_{T1}	Miami Heat	Indiana	Pacers	Miami	…
T_{T2}	Oklahoma City Thunder	Boston	Celtics	Boston	…

Team Between-GameViews (partial)

ID	Name	PTS_2	…	PTS_7	…	REB_2	…	REB_7
P_{T3}	Paul George	80	…	237	…	0	…	2
P_{T4}	Russel Westbrook	37	…	140	…	2	…	7

Player Between-GameViews (partial).

Figure 2: Example (partial) data views. PTS=point, REB=rebound, AST=assist, STL=steal, BLK=block. Q1, Q2, H1 etc., refer to **Q**uarters and **H**alves. PTS_X indicates the SUM of PTS over X games. Each row is considered to be an individual record within the given type of view

et al., 2017). Neural systems can blur the distinction between each sub-task of the pipeline approach, and are able to learn end-to-end to generate in-domain text from structured data (Lebret et al., 2016; Wiseman et al., 2017; Wang, 2019; Puduppully et al., 2019b).

Our work draws on a wide body of prior research on controllability and factualness, namely data engineering, controllable text generation, as well as planning.

Data engineering It is increasingly clear that careful design of datasets used to train deep neural models matters significantly (Rogers, 2021). On simpler data-to-text tasks such as the E2E challenge (Dušek et al., 2018), a number of data-level techniques have been proposed to improve factual accuracy and controllability, including dataset curation (Nie et al., 2019), data-to-text alignment (Dušek et al., 2019; Perez-Beltrachini and Lapata, 2018), straightforward control via input manipulation (Filippova, 2020), and fine-grained annotation (Castro Ferreira et al., 2018; Rebuffel et al., 2021). However, these techniques are not suited for the complexity of the problem at hand, and do not enable *interactability*.

On complex datasets, adding extra records to increase the coverage of the source data over the description (Thomson et al., 2020b) has shown encouraging results, but not all cases of hallucination are clear-cut and easy to solve with a few records. Gong et al. (2019) suggested that it might be possible to model tables as three-dimensional, with rows, column, and time as the dimensions. This is not satisfactory because (1) current neural models cannot correctly perform the arithmetics required to generate the types of utterances found in the reference texts (Nie et al., 2018); (2) adding dimensions leads to intractable complexity.

Controllable Text Generation Controllable Text Generation (CTG) techniques traditionally

involve conditioning an NLG system on several control factors of style (e.g. tone, tense, length, etc.) (Dong et al., 2017; Hu et al., 2017; Ficler and Goldberg, 2017), or content (e.g. customized summaries based on aspect queries (Amplayo et al., 2021)). Control factors are often framed as a collection of key-value pairs, similarly to a typical data-to-text setting.

In the biography domain (Lebret et al., 2016), Filippova (2020) explicitly introduced CTG to data-to-text via an *hallucination score* simply attached as an additional attribute which reflects the amount of divergence in the target reference. Prompting (i.e. starting generation from textual instructions (Liu et al., 2021)) can also provide some control, with (Li and Liang, 2021) obtaining encouraging results at managing the length of generated descriptions. Contrasting with document-level approaches, Rebuffel et al. (2021) propose a finer-grained controllability, via word-level attributes to learn the relevant parts of each training instance.

Planning and Schema Macro-Planning, i.e. high-level planning of ideas, has long been used in traditional NLG pipelines, and has recently been introduced to neural systems as well. On small-scale datasets with short inputs/outputs (e.g. the WebNLG corpus), these approaches rely on detailed annotations of sentence structure and mention placement (Castro Ferreira et al., 2018), or the strong assumption that descriptions describe the associated data entirely (and nothing else) (Xu et al., 2021). However, these are unreasonable dependencies for large-scale datasets, with prohibitive size and complexity of inputs and outputs.

On more complex tasks, the two-step neural approach of (Puduppully et al., 2019a; Puduppully and Lapata, 2021) has proven effective at reducing factual mistakes and provides a small degree of controllability. Given all possible combinations of entities, a planner first selects which will be part of the narrative. In a second step, a generative module learns to output descriptions based on the selected entities. Designed this way, the planning step scales poorly in the input size, since it needs to consider all possible combinations. Furthermore, while the plans can be edited, the impact of individual edits on the final output is unclear (since the model is an end-to-end encoder-decoder). Additionally, no restrictions are placed during decoding to ensure that (1) the decoding process follows the order of the plan; (2) the decoder's copy mechanism doesn't copy attributes from entities in other part of the plans. Lastly, no restrictions are placed on the length of the texts corresponding to each part of the plan, the decoder has to decide the number of sentences to attribute to each item. Generated texts are of similar size, independent of the chosen plan.

3 Data views and their design

In multidimensional settings, descriptions of a data structure mostly focus on an initial point based upon the narrative intent, but also often compare subsets of data along different points of an axis. For example, the best player in the game being summarised might be described in an initial sentence:

> *"Steph Curry led the Warriors with 43 points and 12 assists."*

before an elaboration for the same player, but describing their performance over multiple games:

> *"It was his fourth consecutive double-double*[2]*."*

We make explicit this latent partitioning of the data, via views, as a solution to both the handling of dimensional data, and the alignment of data to text. We split the associated data following the same partitioning, and a view is defined as the records (i.e., key-value pairs) for one entity, from within one partition. In cases where several views are aligned with the same span of text, e.g. a sentence comparing two entities, views can be combined to form view sets.

Figures 1 and 2 illustrate this mirrored multi-dimensionality, along the time axis. The opening sentence describes the basketball game in focus, with details of the teams and their respective scores. This team data can be seen in the left (team) `<Whole-Game>` views. In contrast, the final two sentences report the upcoming opponents for both teams. This data, which lies elsewhere on the axis, is shown in the left `<Between-Game>` views. The third sentence mentions team statistics within quarters of the game, this data is shown in the left `<Within-Game>` views.

[2] https://en.wikipedia.org/wiki/Double-double

3.1 Designing views

We argue that despite its subtleties, data can be broadly categorized along distinct axes. These axes are data and task dependent and must be decided by experts on a case by case basis. We illustrate such an ad-hoc characterization using the SportSett database (Thomson et al., 2020a). This dataset expands on the game-level data from the RotoWire dataset, adding information on the time dimensional axis. SportSett provides a representation of the more general problem of data-to-text with multidimensional data. While specifics may vary between domains and applications, in this case we are able to define three view types corresponding to different time spans. Firstly, we have the `<Whole-Game>` view, which describes the entities (players and teams), with their statistics (such as points and rebounds) for the game overall (Figure 2: first row). This is similar to the data in the original RotoWire dataset, and is the focal point of the narrative (texts are descriptions of a game). We then define views describing the same entities, but for different time spans. The `<Within-Game>` view describes the entities within parts of each game (such as a half or quarter, Figure 2: middle row), and the `<Between-Game>` view describes entities in past or future games (Figure 2: last row). `<Between-Game>` views can either include information about upcoming games, or aggregate statistics for players over a span of games.

For a given dataset, we are therefore able to create views, by assigning each data record to a view type (e.g. in Figure 2 PTS (points) can be assigned to `<Whole-Game>`) and grouping all data of a single type and belonging to a single entity together, forming one view. This manual partitioning of the data is crucial to allow later control in a manner relevant to the task at hand and the goals of the system's user. In particular, we emphasize that while we apply our framework to a basketball-specific task, our approach is not specific to basketball – or sports – and can work in a number of settings. In the financial domain for instance, expert users of an NLG system could choose to describe a stock's performance against a benchmark (e.g. S&P500) on its own, or compared to stocks from the same domain (e.g. all Pharmaceutical stocks), or from the same country (e.g US stocks). Views could also be created on the time axis, to compare a stock's performance to its own in previous months or years.

We argue that in order to present domain experts with convincing NLG software for their business use cases, this light involvement on their part is actually beneficial, since they will all require subtle handling of the data that cannot be anticipated while creating the model, but can be passed during training on their in-house dataset via an ad-hoc partitioning they design and understand.

3.2 Aligning descriptions to views

To identify which views ground the data to each text, we align spans of tokens (in our case sentences) to one or more views (a view set), as shown in Figures 1 and 2. We consider as view sets any sensible combination of views. An example commonly seen in the reference texts is the two teams' `<Whole-Game>` view set which is often used at the beginning of human written descriptions[3]. This alignment between sentences and view sets could be performed by human annotators, or learned (Perez-Beltrachini and Lapata, 2018). We used relatively simple heuristics (see Appendix E for more details) based on our knowledge of the domain as the focus of this paper is on the generation system.

Figure 3 shows the result of this process, with four example sentences grounded to their respective views. In the second column, for instance, the sentence "Russel Westbrook put up fourteen points" is aligned to Russel Westbrook's `<Whole-Game>` view because the noun phrases "Russel Westbrook" and "14 points" are both valid for that view. Most sentences in the corpus are longer than our brevity-focused examples.

3.3 Schemata

View sets can be combined as ordered lists to form document plan schemata in order to guide generation (see Section 4). To train the model, we use the schemata extracted from the reference texts. During inference, we experiment with either following static, extracted, or simple rule-driven schemata (Section 5). While this is another manual input from the expert users of the NLG system, we refer readers to the discussion in Section 6.1 where we discuss the impact of also predicting the plan. Briefly, models that predict the plan are often *dull* in the sense that they always predict the same plan,

[3]To reduce complexity, we allow view sets to be comprised of one or two entities (players/teams) for one view type.

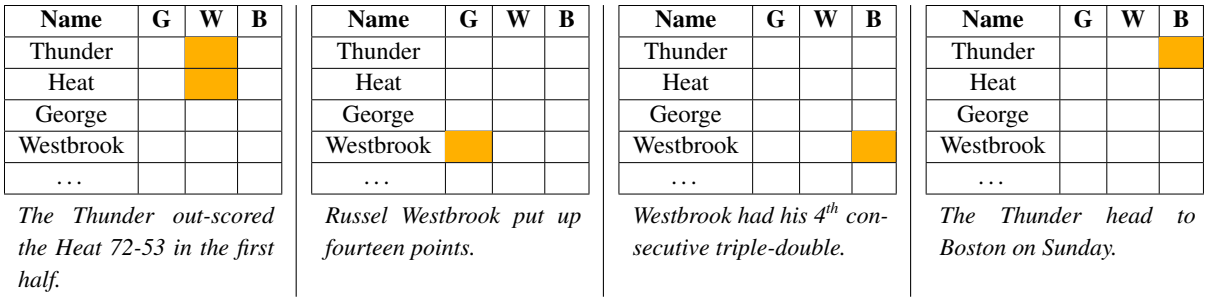

Figure 3: Example sentences aligned to views (Whole-**G**ame, **W**ithin-Game, and **B**etween-Game).

and tend to include a number of irrelevant facts that scored high in evaluation metrics.

4 Generating text with a Hierarchical Model leveraging views

As discussed in Section 2, it is unclear to what extent current models deal with extremely large inputs or can utilize views: copy actions are harder to train; memory and compute constraints make training and inference very slow; no model has been proposed to explicitly constrain the order and scope of sentences in generated descriptions.

The hierarchical system of Rebuffel et al. (2020) can be extended to fully leverage view annotations, scaling well with increased input size, generating descriptions following a precise ordering of content. Specifically, the original model was designed with an emphasis on structure, which we use to our advantage to constrain the copy mechanism, as well as input additional data and guide the generation process. We provide an overview of the model here, but refer readers to the original paper for an extensive description.

From a high-level perspective, the system was designed to handle data structured in a hierarchical fashion, i.e. data that can be split into distinct entities, each of them being described by a collection of records, in the form of (key, value) pairs. The system follows a standard encoder-decoder architecture. In particular, it:

1. Encodes each entity independently, as collections of records;
2. Encodes the input data-structure, as a collection of entities;
3. Generates text using hierarchical attention/copy mechanisms:
   (a) An attention distribution is computed over entities;
   (b) Inside each entity, an attention distribution is computed over its records.

Views are integrated to this framework by considering each as an independent entity, with an important distinction to adapt to the now extremely large input size. In their original work, (Rebuffel et al., 2020) encode all input data, which is not satisfactory when adding a large number of views. Akin to teacher forcing (Williams and Zipser, 1989), we encode additional views only when they are relevant to the description at hand. In other words, we always encode all `<Whole-Game>` views which provide a needed high-level understanding of the game, but only encode the `<Between-Game>` and `<Within-Game>` views that ground sentences of the current description, leaving out views which will never be solicited by the decoder.

This prior enables the model to have a broad overview of the game (using `<Whole-Game>` views), while being able to copy specific information from other dimensions for each entity mentioned in the target descriptions. During inference, we only encode the views which are part of the specified control schema, as other views need not be solicited during decoding.

4.1 Hierarchical Encoding

Formally, we consider the following setting: • Let $\mathcal{D}$ be a *dataset* which is a collection of aligned (data-structure, description) pairs (s, y).

• A *data-structure* s is originally seen as an unordered set of I views e_i. We thus denote $s := \{e_1, ..., e_i, ..., e_I\}$.

• Each view e_i is a labelled set of J_i unordered records $\{r_{i,1}, ..., r_{i,j}, ..., r_{i,J_i}\}$; where a record $r_{i,j}$ is defined as a (key, value) pair: $(k_{i,j}, v_{i,j})$. Note that J_i might differ between views. Importantly, the set of records is labelled by the *view-type*, e_{type}

Following (Rebuffel et al., 2020), we first encode each view independently, and then together. We denote by $\mathbf{e_i}$ the learned representation of view e_i, computed by the high-level encoder.

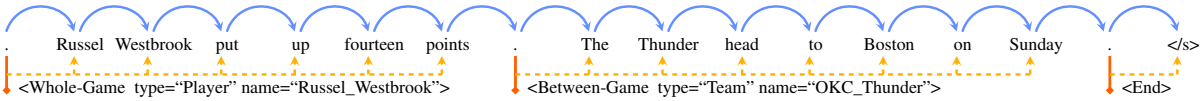

Figure 4: Example decoding of sentences. Decoder is trained to predict next words (blue arrows). During sentence decoding the attention and copy mechanisms are restricted to entities grounded to the current views (see Figure 3), selected at sentence delimiter. We regularize training with an added loss (see Section 4.3). The decoder must predict the next grounding views (orange bars) at each sentence delimiter.

Encoding a subset of views Each sentence of the target description is grounded to a small subset of views. To encode high-level information about the entities and the dimension of the considered views, our system relies on self-attention, which enables encoding sets of unordered objects.

Formally, for a given sentence s, we denote $\mathcal{G}_s$ the subset of views that grounds the sentence, as explained in Section 3. We compute a fixed-size representation $\boldsymbol{\mathcal{G}_s}$ of this subset using self-attention (SA):

$$\begin{aligned} \hat{\boldsymbol{\mathcal{G}}}_{\boldsymbol{s}} &= \text{SA}(\ [\mathbf{e_i};\ \forall e_i \in \mathcal{G}_s]\ ) \\ \boldsymbol{\mathcal{G}_s} &= [\hat{\boldsymbol{\mathcal{G}}}_{\boldsymbol{s}};\ \mathbf{e_{type}}] \end{aligned} \tag{1}$$

where $[\ \cdot\ ;\ \cdot\ ]$ represents concatenation, and $\mathbf{e_{type}}$ is a learned embedding of the view-type e_{type}, which is the same for all views grounding a same sentence.

4.2 View-aware decoding

In addition to relying on the encoded content of the views, we guide and constrain the decoding process sentence by sentence.

1. The system uses the encoded high-level information about the entities and the type of the considered subset of views;
2. The system decodes word by word, using the previously encoded information;
3. Attention and copy mechanisms are limited to records from the considered views.

During the decoding process for sentence s, the decoder uses the learned representation $\boldsymbol{\mathcal{G}_s}$, computed at Equation 1, to update its hidden state at each decoding step. Recall that the standard decoder of (Rebuffel et al., 2020) is an LSTM which updates its hidden state $\mathbf{d_t}$ using the previously decoded token:

$$\mathbf{d_t} = \text{LSTM}([\ \mathbf{d_{t-1}};\ \mathbf{y_{t-1}}\ ]) \tag{2}$$

where $\mathbf{y_{t-1}}$ is the learned embedding of token y_{t-1}. In this work, we adapt this update so that the grounding's representation is taken into account:

$$\mathbf{d_t} = \text{LSTM}([\ \mathbf{d_{t-1}};\ \mathbf{y_{t-1}};\ \boldsymbol{\mathcal{G}_s}\ ]) \tag{3}$$

when the current token y_t is from sentence s.

Hierarchical attention constrained on views To fully leverage the hierarchical structure of their encoder, (Rebuffel et al., 2020) proposed a hierarchical attention mechanism to compute the context fed to the decoder module. The dynamic context is computed in two steps: first attending to views, then to records corresponding to these views. At each decoding step t, the model learns a first set of attention scores $\alpha_{i,t}$ over views e_i and a second set of attention scores $\beta_{i,j,t}$ over records $r_{i,j}$ belonging to view e_i. The $\alpha_{i,t}$ scores are normalized to form a distribution over all views e_i, and $\beta_{i,j,t}$ scores are normalized to form a distribution over records $r_{i,j}$ of view e_i. Each view is then represented as a weighted sum of its record embeddings, and the entire data structure is represented as a weighted sum of the view representations.

Formally, the dynamic context is computed as:

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

$$\text{where}\ \ \alpha_{i,t} \propto \exp(\mathbf{d}_t \mathbf{W}_\alpha \mathbf{e}_i) \tag{5}$$

$$\text{and}\ \ \beta_{i,j,t} \propto \exp(\mathbf{d}_t \mathbf{W}_a \mathbf{k}_{i,j}) \tag{6}$$

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, $\mathbf{e_i}$ and $\mathbf{k}_{i,j}$ are the computed representation of views and records' keys respectively, $\sum_i \alpha_{i,t} = 1$, and for all $i \in \{1, ..., I\}$ $\sum_j \beta_{i,j,t} = 1$.

In this work, we constrain the attention mechanism such that it is computed only on grounded views. This has the benefit of restraining the copy mechanism to a few specific records, minimizing the number of inaccurate copies. Figure 4 illustrates this mechanism: during the first sentence, only records from `<Whole-Game> Russel_Westbrook` can be attended to, while during the second sentence, only records from

<Between-Game> OKC_Thunder can be attended to.

4.3 Dual Loss Regularization

For each data-structure t 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 $\mathcal{D}$:

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

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

Early experiments showed that training using only the standard objective function can be somewhat unstable, and that adding regularization proves useful and increases the model's performances. In practice, in addition to predicting next words, the decoder is also trained to predict next grounding views and view-type at the end of each sentence (illustrated in Figure 4).

Formally, let $\mathcal{L}_w$ refers to the original token-level loss (Section 4.3), and $\mathcal{L}_\ell$ and $\mathcal{L}_e$ refer to two classification losses, on view-type and grounding entities respectively. Then, our model is trained to minimize the following loss:

$$\mathcal{L} = \lambda_1 \mathcal{L}_w + \lambda_2 \mathcal{L}_\ell + \lambda_3 \mathcal{L}_e \quad (7)$$

where $\sum_i \lambda_i = 1$ and are manually tuned.

5 Experimental Setup

Data We use the sports journalism (basketball) dataset SportSett Thomson et al. (2020a), based on the RotoWire datset introduced by (Wiseman et al., 2017). It consists of game statistics paired with human-authored descriptions. The original dataset contained train-test corruption so we partition by season[4] to provide a suitable proxy of a real-world task.

System comparisons[5] We trained one model, then used three different schemata to produce text:

- V-SIMPLE - A simple static schema that is the same for all games (see Figure 5).
- V-EXTENDED - Based on V-SIMPLE, schema varies for each game, adding <Between-Game> / <Within-Game> elaborations to some players. These elaborations are chosen with simple heuristics (e.g. when a player has had 3 double-doubles in his last games, add a <Between-Game> elaboration for this player).
- V-GUIDED - Schema automatically extracted from the human authored descriptions. These vary from game to game.

We compare our three variants to two variants of the hierarchical model of (Rebuffel et al., 2020) (that do not have planning modules and represent state-of-the art without such a feature) and two variants of the explicit-planning approach of (Puduppully and Lapata, 2021):

- H-FULL - **H**ierarchical encoder based system of (Rebuffel et al., 2020) but with all views made available to it.
- H-NEXT - **H**ierarchical encoder based system of (Rebuffel et al., 2020), configured as per (Thomson et al., 2020b) to include additional information for the game and next games.
- MP-SIMPLE - System of (Puduppully and Lapata, 2021) using our simple static schema (same as our variant V-SIMPLE).
- MP-GUIDED - System of (Puduppully and Lapata, 2021) using the schema extracted from human authored texts (same as our variant V-GUIDED).

Note that the hierarchical model of (Rebuffel et al., 2020) is not able to handle schema guidance during generation, and that the system of (Puduppully and Lapata, 2021) doesn't differentiate between <Whole-Game>, <Between-Game>, and <Within-Game> dimensions: descriptions generated using this system are identical when using the SIMPLE or EXTENDED schema.

6 Human evaluation of factual accuracy

With the gold standard protocol of Thomson and Reiter (2020), human annotators highlight non-overlapping spans of text that contain errors, also selecting an error category. Participants are provided with links to basketball-reference.com, which is at least as complete as the system input data, and much easier for humans to navigate. Four main error categories are used; *Name* (named

[4] 2014-16 train; 2017 valid; 2018 test

[5] To ensure comparable results, we have retrained all baselines with the same dataset partitions. Hyper-parameters and other training details will be included in the code repository.

Document plan (schema):
<Whole-Game> OKC Thunder, Miami Heat
<Whole-Game> OKC Thunder, Miami Heat
<Whole-Game> OKC Thunder, Miami Heat
<Whole-Game> OKC Thunder, Miami Heat
<Whole-Game> Paul George
<Whole-Game> Dennis Schroder
<Whole-Game> Russell Westbrook
<Whole-Game> Steven Adams
<Whole-Game> Jerami Grant
<Whole-Game> Patrick Patterson
<Whole-Game> Kelly Olynyk
<Whole-Game> Josh Richardson
<Whole-Game> Hassan Whiteside
<Whole-Game> Bam Adebayo
<Between-Game> OKC Thunder
<Between-Game> Miami Heat

The Oklahoma City Thunder (33-18) defeated the Miami Heat (24-26) 118-102 on Friday. The Thunder shot 48 percent from the field and 53 percent from three-point range. They also dominated the rebounding battle, 50-47. The **Heat**[N] also held the Heat to 45 percent shooting and outrebounded them 50-**38**[U]. Paul George led the way for the Thunder with 43 points, seven rebounds, five assists and two steals. Dennis Schroder added 28 points of his own off the bench. Russell Westbrook finished with 14 points, **12**[U] assists and 12 rebounds. Steven Adams scored 13 points and grabbed seven rebounds in 27 minutes. Jerami Grant chipped in eight points, **while Jerami Grant**[W] added eight. Patrick Patterson **led the bench**[W] with five points and two rebounds. Kelly Olynyk was the **only other**[W] **Heat player**[C] in double figures, finishing with 21 points. Josh Richardson scored 18 points on 7-of-18 shooting. Hassan Whiteside had 12 points and 16 rebounds in 19 minutes. Bam Adebayo, it **wasn't enough to overcome the Heat**[W]. Oklahoma City will travel to Boston to take on the Celtics on Sunday. Miami plays host to the Pacers, also on Saturday.

Figure 5: Instructions and generated (V-SIMPLE system) game summary for OKC@MIA on February 1st 2019. **Name**[N], **Number**[U], **Word**[W], and **Context**[C] mistakes are highlighted in the summary.

entities), *Number* (ordinal, cardinal, etc), *Word* (a word or phrase that is not a name or number), *Context* (such as implicature errors). There is also the last resort category *Other*, for text that is nonsensical. Finally, there is a *Not Checkable* category, which covers facts that are impractical or impossible to check using the provided reference data.

We performed our experiment with one annotator per text[6]. The original protocol limited annotators to only 4 prior games before defaulting to *Not Checkable* errors. We felt this was overly restrictive so we asked annotators to check all assertions within the current season. We compared generations for 35 random basketball games, generating a text for each of the 7 systems. A Latin square design was used whereby each participant annotated 5 texts for each system, never seeing the same game twice (245 total annotated texts).

6.1 Human Evaluation Results

The best performing systems were our View-based system, and that of Puduppully and Lapata (2021) (when both are provided with a simple schema). They were both significantly different, in terms of the number of errors reported, to all other systems. Figure 6 shows a box plot of error count for each system Figure 5 shows the generation for the V-SIMPLE system, marked up for errors by an annotator, on the same game shown in Figure 1.

[6]MTurk workers were recruited by the same process as Thomson et al. (2023) who reported high precision and recall of single annotators by this method. It is expensive to run with three annotators per text, therefore we prioritised having more texts over more annotators per-text.

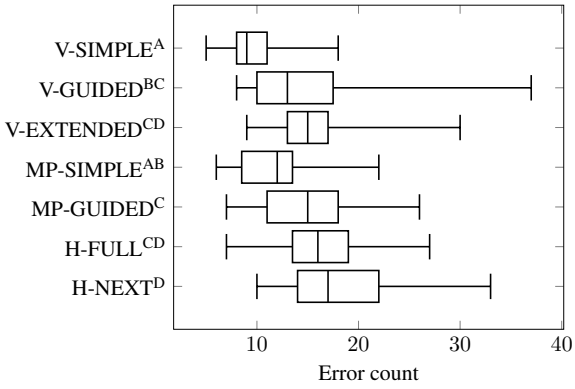

Figure 6: Error count, systems sharing superscript letters are not significantly different.

7 Discussion and Limitations

7.1 Different Domains and Datasets

Our work focuses on a single domain/dataset. Expanding this to include additional domains and datasets would be useful future work now that a process has been outlined in this paper for working with multi-dimensional data. The MLB dataset (Puduppully et al., 2019a) and the Chart-to-Text dataset of Obeid and Hoque (2020) both exhibit problems related to modelling of time that is seen in the basketball domain.

7.2 Large Language Models

The system evaluated in this paper used a transformer encoder and an LSTM decoder. Whilst the LSTM, in particular, may be considered an out-

dated NLP model[7], the view-based design could also be implemented using pre-trained large language models such as GPT-3 (Brown et al., 2020) or even using ChatGPT[8]. For systems such as ChatGPT, a series of prompts could be generated that reference views, e.g.,

1. Given *View A*; write a sentence that describes Steph Curry's performance in this game.

2. Then, given *View B*; add a sentence that describes Steph Currys performance over the past 7 games.

Question-answer-based document planning has been explored for tasks such as summarisation (Narayan et al., 2023) and by referencing views it could be applied to data-to-text generation. Note that even with the larger token windows of GPT-4, it remains impossible to feed the model all input data that could be used to extract useful insights, meaning that some form of insight selection is required upstream of the language model. Our view-based approach handles this at a high level, whilst leaving micro-planning and realisation to the neural model.

7.3 Generality and View Grounding

View grounding was performed using simple heuristics in order to use the grounded spans of text (sentences) for the downstream generation task. This should be explored and evaluated as a standalone task in future work. It is also unclear for an individual view, what the limits of complexity are. This introduction of control through the use of views is not a loss of generality, but rather a requirement for generating text that is useful and interesting to human readers, rather than "general" but vague or dull.

8 Conclusion

Increasing concern has been raised regarding the quality of both task setup (Raji et al., 2021), as well as evaluation in NLP/NLG, with caveats of systems and experimental results often going unreported (Gehrmann et al., 2022). We contribute to the meaningful progress of both. In exploring an alternative task where generation of an exact human reference text is not the goal (it is just part of the available information), we bring the generation process more inline with a real-world problem where control is a major requirement.

We expanded the data-to-text task by considering the mirrored multi-dimensional aspects of both data and text. We have shown that by splitting this extended data into manageable views based on its dimensionality, meaningful control can be introduced over system output without sacrificing factual accuracy. Control comes in the form of ordering views using schema, in the way messages might be ordered in rule-based systems, but allowing the neural model to handle the complexities of micro-planning and surface realization.

The method of splitting data into views could be applied to complex data from sources such as relational databases or multi-dimensional arrays. Our implementation of views is only one possible way to structure the data. We intuit that views that describe a named entity and a set of direct attributes, in any dimension, work well.

Considering the multidimensionality of data and text brings the problem closer to that encountered and addressed by humans in the real world. Enhancing the structure of datasets, as well as designing models that leverage this multidimensionality, will move systems closer to the goal of human-like descriptions of complex data.

Ethics Statement

Ethical approval was obtained from our ethics review board. We paid our Mechanical Turk participants $8 US per text annotated during qualification, practice, and live work. This equates to approximately $20 per hour and multiple workers indicated to us that this was "about right". Only when qualifying participants uploaded blank documents (no reported errors when there should be around 20) did we reject HITs. In the rare cases that qualified workers made mistakes such as accidentally uploading a blank annotation document rather than the completed one, we still paid them for the HIT and contacted them to get the correct file.

Online Resources

All code, data, and human evaluation resources will be made available on GitHub[9].

[7] Our reasoning for using an LSTM decoder was that it was used by previous models (Puduppully and Lapata, 2021) and we were exploring changes to the encoder component only.

[8] https://openai.com/blog/chatgpt

[9] https://github.com/nlgcat/inlg2023views

Acknowledgements

We would like to thank the Mechanical Turk workers who participated in the human evaluation. The work presented here is partially funded by the Engineering and Physical Sciences Research Council (EPSRC), which funded Craig Thomson under a National Productivity Investment Fund Doctoral Studentship (EP/R512412/1).

References

Reinald Kim Amplayo, Stefanos Angelidis, and Mirella Lapata. 2021. Aspect-controllable opinion summarization. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 6578–6593, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel Ziegler, Jeffrey Wu, Clemens Winter, Chris Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language models are few-shot learners. In *Advances in Neural Information Processing Systems*, volume 33, pages 1877–1901. Curran Associates, Inc.

Thiago Castro Ferreira, Diego Moussallem, Emiel Krahmer, and Sander Wubben. 2018. Enriching the WebNLG corpus. In *Proceedings of the 11th International Conference on Natural Language Generation*, pages 171–176, Tilburg University, The Netherlands. Association for Computational Linguistics.

Robert Dale. 2020. Natural language generation: The commercial state of the art in 2020. *Natural Language Engineering*, 26(4):481–487.

Li Dong, Shaohan Huang, Furu Wei, Mirella Lapata, Ming Zhou, and Ke Xu. 2017. Learning to generate product reviews from attributes. In *EACL*.

Ondřej Dušek, David M. Howcroft, and Verena Rieser. 2019. Semantic noise matters for neural natural language generation. In *Proceedings of the 12th International Conference on Natural Language Generation*, pages 421–426, Tokyo, Japan. Association for Computational Linguistics.

Ondřej Dušek, Jekaterina Novikova, and Verena Rieser. 2018. Findings of the E2E NLG challenge. In *Proceedings of the 11th International Conference on Natural Language Generation*, pages 322–328, Tilburg University, The Netherlands. Association for Computational Linguistics.

Jessica Ficler and Yoav Goldberg. 2017. Controlling linguistic style aspects in neural language generation. In *Workshop on Stylistic Variation @ ACL*.

Katja Filippova. 2020. Controlled hallucinations: Learning to generate faithfully from noisy data. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 864–870, Online. Association for Computational Linguistics.

J. Fleiss. 1971. Measuring nominal scale agreement among many raters. *Psychological Bulletin*, 76:378–382.

Claire Gardent, Anastasia Shimorina, Shashi Narayan, and Laura Perez-Beltrachini. 2017. Creating training corpora for NLG micro-planners. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 179–188, Vancouver, Canada. Association for Computational Linguistics.

Albert Gatt and Emiel Krahmer. 2018. Survey of the state of the art in natural language generation: Core tasks, applications and evaluation. *J. Artif. Int. Res.*, 61(1):65–170.

Sebastian Gehrmann, Elizabeth Clark, and Thibault Sellam. 2022. Repairing the cracked foundation: A survey of obstacles in evaluation practices for generated text.

Heng Gong, Xiaocheng Feng, Bing Qin, and Ting Liu. 2019. Table-to-text generation with effective hierarchical encoder on three dimensions (row, column and time). In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3143–3152, Hong Kong, China. Association for Computational Linguistics.

Zhiting Hu, Zichao Yang, Xiaodan Liang, Ruslan Salakhutdinov, and Eric P. Xing. 2017. Toward controlled generation of text. In *ICML*.

Anna S. Law, Y. Freer, J. Hunter, R. Logie, N. McIntosh, and John Quinn. 2005. A comparison of graphical and textual presentations of time series data to support medical decision making in the neonatal intensive care unit. *Journal of Clinical Monitoring and Computing*, 19:183–194.

Rémi Lebret, David Grangier, and Michael Auli. 2016. 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*, pages 1203–1213, Austin, Texas. Association for Computational Linguistics.

Xiang Lisa Li and Percy Liang. 2021. Prefix-tuning: Optimizing continuous prompts for generation. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 4582–4597, Online. Association for Computational Linguistics.

Pengfei Liu, Weizhe Yuan, Jinlan Fu, Zhengbao Jiang, Hiroaki Hayashi, and Graham Neubig. 2021. Pre-train, prompt, and predict: A systematic survey of prompting methods in natural language processing. *arXiv preprint arXiv:2107.13586*.

Nitika Mathur, Timothy Baldwin, and Trevor Cohn. 2020. Tangled up in BLEU: Reevaluating the evaluation of automatic machine translation evaluation metrics. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 4984–4997, Online. Association for Computational Linguistics.

Shashi Narayan, Joshua Maynez, Reinald Kim Amplayo, Kuzman Ganchev, Annie Louis, Fantine Huot, Anders Sandholm, Dipanjan Das, and Mirella Lapata. 2023. Conditional generation with a question-answering blueprint.

Feng Nie, Jinpeng Wang, Jin-Ge Yao, Rong Pan, and Chin-Yew Lin. 2018. Operation-guided neural networks for high fidelity data-to-text generation. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 3879–3889, Brussels, Belgium. Association for Computational Linguistics.

Feng Nie, Jin-Ge Yao, Jinpeng Wang, Rong Pan, and Chin-Yew Lin. 2019. A simple recipe towards reducing hallucination in neural surface realisation. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 2673–2679, Florence, Italy. Association for Computational Linguistics.

Jason Obeid and Enamul Hoque. 2020. Chart-to-text: Generating natural language descriptions for charts by adapting the transformer model. In *Proceedings of the 13th International Conference on Natural Language Generation*, pages 138–147, Dublin, Ireland. Association for Computational Linguistics.

Ankur P Parikh, Xuezhi Wang, Sebastian Gehrmann, Manaal Faruqui, Bhuwan Dhingra, Diyi Yang, and Dipanjan Das. 2020. ToTTo: A controlled table-to-text generation dataset. In *Proceedings of EMNLP*.

Laura Perez-Beltrachini and Mirella Lapata. 2018. Bootstrapping generators from noisy data. In *NAACL-HLT*.

Ratish Puduppully, Li Dong, and Mirella Lapata. 2019a. Data-to-text generation with content selection and planning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 6908–6915.

Ratish Puduppully, Li Dong, and Mirella Lapata. 2019b. Data-to-text generation with entity modeling. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 2023–2035, Florence, Italy. Association for Computational Linguistics.

Ratish Puduppully and Mirella Lapata. 2021. Data-to-text generation with macro planning. *Transactions of the Association for Computational Linguistics*, 9:510–527.

Inioluwa Deborah Raji, Emily M. Bender, Amandalynne Paullada, Emily Denton, and Alex Hanna. 2021. Ai and the everything in the whole wide world benchmark.

Clément Rebuffel, Marco Roberti, Laure Soulier, Geoffrey Scoutheeten, R. Cancelliere, and P. Gallinari. 2021. Controlling hallucinations at word level in data-to-text generation. *ArXiv*, abs/2102.02810.

Clément Rebuffel, Laure Soulier, Geoffrey Scoutheeten, and Patrick Gallinari. 2020. A hierarchical model for data-to-text generation. In *Advances in Information Retrieval - 42nd European Conference on IR Research, ECIR 2020*, pages 65–80.

Ehud Reiter. 2007. An architecture for data-to-text systems. In *proceedings of the eleventh European workshop on natural language generation (ENLG 07)*, pages 97–104.

Ehud Reiter and Robert Dale. 1997. Building applied natural language generation systems. *Natural Language Engineering*, 3(1):57–87.

Ehud Reiter, S. Sripada, J. Hunter, Jin Yu, and I. Davy. 2005. Choosing words in computer-generated weather forecasts. *Artif. Intell.*, 167:137–169.

Anna Rogers. 2021. Changing the world by changing the data. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 2182–2194, Online. Association for Computational Linguistics.

M. S. Silberman, B. Tomlinson, R. LaPlante, J. Ross, L. Irani, and A. Zaldivar. 2018. Responsible research with crowds: Pay crowdworkers at least minimum wage. *Commun. ACM*, 61(3):39–41.

André Luiz Rosa Teixeira, João Campos, Rossana Cunha, Thiago Castro Ferreira, Adriana Pagano, and Fabio Cozman. 2020. DaMata: A robot-journalist covering the Brazilian Amazon deforestation. In *Proceedings of the 13th International Conference on Natural Language Generation*, pages 103–106, Dublin, Ireland. Association for Computational Linguistics.

Craig Thomson and Ehud Reiter. 2020. A gold standard methodology for evaluating accuracy in data-to-text systems. In *Proceedings of the 13th International Conference on Natural Language Generation*, pages 158–168, Dublin, Ireland. Association for Computational Linguistics.

Craig Thomson and Ehud Reiter. 2021. Generation challenges: Results of the accuracy evaluation shared task. In *Proceedings of the 14th International Conference on Natural Language Generation*, pages 240–248, Aberdeen, Scotland, UK. Association for Computational Linguistics.

Craig Thomson, Ehud Reiter, and Somayajulu Sripada. 2020a. SportSett:basketball - a robust and maintainable data-set for natural language generation. In *Proceedings of the Workshop on Intelligent Information Processing and Natural Language Generation*, pages 32–40, Santiago de Compostela, Spain. Association for Computational Lingustics.

Craig Thomson, Ehud Reiter, and Barkavi Sundararajan. 2023. Evaluating factual accuracy in complex data-to-text. *Computer Speech & Language*, 80.

Craig Thomson, Zhijie Zhao, and Somayajulu Sripada. 2020b. Studying the impact of filling information gaps on the output quality of neural data-to-text. In *Proceedings of the 13th International Conference on Natural Language Generation*, pages 35–40, Dublin, Ireland. Association for Computational Linguistics.

Ashish Upadhyay and Stewart Massie. 2022. Content type profiling of data-to-text generation datasets. In *Proceedings of the 29th International Conference on Computational Linguistics*, pages 5770–5782, Gyeongju, Republic of Korea. International Committee on Computational Linguistics.

Chris van der Lee, Emiel Krahmer, and Sander Wubben. 2017. PASS: A Dutch data-to-text system for soccer, targeted towards specific audiences. In *Proceedings of the 10th International Conference on Natural Language Generation*, pages 95–104, Santiago de Compostela, Spain. Association for Computational Linguistics.

Emiel van Miltenburg, Miruna Clinciu, Ondřej Dušek, Dimitra Gkatzia, Stephanie Inglis, Leo Leppänen, Saad Mahamood, Emma Manning, Stephanie Schoch, Craig Thomson, and Luou Wen. 2021. Underreporting of errors in NLG output, and what to do about it. In *Proceedings of the 14th International Conference on Natural Language Generation*, pages 140–153, Aberdeen, Scotland, UK. Association for Computational Linguistics.

Emiel van Miltenburg, Miruna Clinciu, Ondřej Dušek, Dimitra Gkatzia, Stephanie Inglis, Leo Leppänen, Saad Mahamood, Stephanie Schoch, Craig Thomson, and Luou Wen. 2023. Barriers and enabling factors for error analysis in nlg research. *Northern European Journal of Language Technology*, 9.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc.

Hongmin Wang. 2019. Revisiting challenges in data-to-text generation with fact grounding. In *Proceedings of the 12th International Conference on Natural Language Generation*, pages 311–322, Tokyo, Japan. Association for Computational Linguistics.

Ronald J. Williams and D. Zipser. 1989. A learning algorithm for continually running fully recurrent neural networks. *Neural Computation*, 1:270–280.

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

Xinnuo Xu, Ondřej Dušek, Verena Rieser, and Ioannis Konstas. 2021. AggGen: Ordering and aggregating while generating. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1419–1434, Online. Association for Computational Linguistics.

A Automated Metric Results

We include results on common metrics of BLEU, Relation Generation (RG), Content Selection (CS), and Content Ordering (CO) for this task in Table 1. Automatic metrics are often expected in NLP papers, although their usefulness in this domain is limited at best. We include them in the appendix for this reason.

The V-SIMPLE and MP-SIMPLE systems, based on simple schema, had the highest RG scores, and hierarchical systems the lowest. Interestingly, CO scores are highest when models follow extracted schema from gold texts.

BLEU scores are within a narrow range, with Mathur et al. (2020) having shown that larger differences are required in order to make judgments. The information extraction based metrics prove more useful, with Wiseman et al. (2017) stating that their results were generally inline with their human evaluation. However, Thomson and Reiter (2021) observed that state-of-the-art metrics can detect simple errors, but struggle with more complex semantic and contextual errors. It is also worth noting that running BLEU on a deranged copy of the test set (comparing each game with a random game other than itself) can yield BLEU scores in the region of 8.0 to 10.0, simply due to common terminology and syntax.

System	RG		CS			CO	BLEU
	P%	#	P%	R%	F1		
REF	0.84	26.84	-	-	-	-	-
V-SIMPLE	0.87	26.21	0.60	0.57	0.58	0.21	19.68
V-GUIDED	0.81	17.56	**0.71**	0.48	0.57	**0.30**	17.29
V-EXTENDED	0.84	27.06	0.57	0.58	0.57	0.21	21.90
MP-SIMPLE	**0.88**	43.27	0.48	**0.73**	0.58	0.22	21.52
MP-GUIDED	0.82	30.02	0.60	0.67	**0.63**	**0.30**	**22.27**
H-FULL	0.76	27.76	0.42	0.47	0.44	0.16	17.73
H-NEXT	0.77	23.09	0.51	0.47	0.49	0.18	21.22

Table 1: Automatic metric results for all systems.

B Content Ordering Experiment

This experiment aims to determine whether sentences in generated summaries are in the correct order. In designing this experiment we had two main concerns. Firstly, inter-annotator agreement should at least be moderate, ideally high. This precludes designs where participants are free to rearrange all sentences; the large number of permutations increases the likelihood of disagreement. Secondly, it should be possible to perform meaningful error analysis in order to better understand both the systems, and the protocol itself. This rules out Likert-based approaches because, with paragraph-sized generations, it is impossible to tell which part of the summary caused a participant to score the text in the way they did. Likert ratings have been shown to have poor agreement in this domain (Puduppully and Lapata, 2021).

B.1 Design

We presented generated summaries to participants with the first two sentences highlighted as 'the beginning', the final two sentences highlighted as 'the end', and everything in between highlighted as 'the middle'. We then asked participants, for each of the four sentences in the beginning and end, whether it should:

- **Remain** where it is.
- be **Transposed** with its partner, i.e., the other sentence from the beginning or end.
- be moved to the middle, a **Short** distance.
- be moved to the opposing end of the summary, a **Long** distance.

When asked if sentences should be moved to another section participants did not specify exactly where, simply which other section. We also asked the middle was in an acceptable order (Yes/No).

Participants were placed into 35 non-exclusive groups (the number of combinations of size three for 7 participants). Each group evaluated a summary from each of the 7 systems, such that 245 unique summaries were evaluated by 3 annotators.

B.2 Results

For content ordering, we first consider whether participants believed a sentence should be moved to a different section. Inter-annotator agreement by Fleiss Kappa (Fleiss, 1971) was 0.591, indicating a moderate agreement. However, this falls to 0.469 when we consider the *Short*/*Long* move distances, and to 0.350 if we also consider transposition of beginning/end sentence pairs (p-value was less than 0.001 in all cases). This confirms our design assumption that allowing participants to freely rearrange texts of this length would result in low or no agreement. We did run an experiment where different participants (MTurk masters with US high-school diplomas) were asked to rate how readable and understandable generations were. Agreement for this was even lower, below 0.2, and results are not included for that reason.

System	Long	Short	Transpose	Remain
V-SIMPLE	1	3	55	361
V-GUIDED	1	33	10	372
V-EXTENDED	0	10	59	351
MP-SIMPLE	3	15	4	398
MP-GUIDED	3	53	7	353
H-FULL	2	65	1	352
H-NEXT	1	88	4	327

Table 2: Number of sentences that annotators would move, by destination.

B.3 Conclusion

The results in Table 2 show that all models do a good job at avoiding *Long* errors, that is they do not confuse the beginning of the narrative with the end. The simple schema of both V-SIMPLE and MP-SIMPLE have fewer *Short* errors, especially compared with the hierarchical encoder systems. Our models in V-SIMPLE and V-EXTENDED mode *Transpose* sentences in the *beginning* or *end* with higher frequency. Looking into this further, our schema (for both models) was set to realize the upcoming game for the winning team in the *Penultimate* sentence, then the losing team in the *Final* sentence. This was deemed incorrect by some annotators (the losing teams players are usually discussed immediately before the end, therefore the context at that stage is the losing team). Our system is capable of adjusting for this, with a simple schema change reversing the order of these sentences. The MP-SIMPLE system does not have the fine-grained control to constrain generation to two separate sentences, therefore it frequently discusses both teams upcoming games in a single *Final* sentence and does not encounter this *Transpose* problem as often as our models. It is also unclear how the *Short* errors of such a system could be corrected.

This experiment is included in the appendix because whilst it was unsuccessful at demonstrating a difference between systems (agreement was low), it does provide some insight and with some refinement of experimental design could be a useful approach (agreement was not so low that there are no possible pathways to higher agreement).

C Post-hoc error analysis

In addition to the quantitative data, our accuracy evaluation yielded qualitative data in the form of free-text comments that annotators could leave when reporting each error. We therefore performed an error analysis, something that is crucial to to gain insight into where our systems are failing (van Miltenburg et al., 2021, 2023). With the MP-SIMPLE and MP-SIMPLE systems some annotators queried the protocol because some names were spelled incorrectly. This had not been a problem for word-based systems, but since the system of Puduppully and Lapata (2021) operates at the subword level, it would sometimes generate texts that contained out of vocabulary words once subwords were reconstructed. An example can be seen in the sentence: *"**Well ell ell ell ell ell ell Carter**^{N} , as he scored 25 points to go along with eight rebounds and five assists ."*, where *"ell"* is an out-of-vocabulary word. The annotator for this sentence marked it as an error, leaving the mildly derisive comment of *"more commonly referred to as just **Wendell Carter**^{N}"*. Upon further investigation, this problem is not uncommon in the generations of this system, yet it would be missed by the RG metric and at times our human evaluation as well[10]. In one of the worst cases (from the full test set, not an item from our human evaluation), the complete generation was: *"The Miami Heat (27 - 33) defeated the Golden State Warriors (43 - 18) 126 - 125 on Friday . Justise Winslow and Bam AAAAAAAAAAA"*, followed by the letter *'b'* repeated 808 times. Our view based systems also struggled at times to generate full sentences about players such as Bam Adebayo, who had not been seen during training. For example, one output was *"Bam Adebayo, it **wasn't enough to overcome the Heat**^{W}."*, where the model knew it should generate a sentence about Bam Adebayo, but did not include any statistics. It is possible the models are relying on the values of the player name field rather than generalizing.

To gain further insight, we performed some automated error analysis on outputs from the full test set (2018 season). Table 4 shows the average token counts and out of vocabulary[11] tokens for the generations of each system. Our view based systems each generated a small number of out-of-vocab tokens by erroneously copying boolean values from the input data (we would fix this by

[10]The factual accuracy annotation instructions of Thomson and Reiter (2020) ask annotators to ignore spelling, syntax and grammar, so some annotators did not mark these as errors (if they could make out which player was being referred to).

[11]A vocabulary was created using all test data values, training data texts and a range of numbers in word and digit form.

System	NAME	NUMBER	WORD	CONTEXT	OTHER	NOT CHECKABLE	TOTAL
V-SIMPLE	44	115	134	16	19	11	339
V-GUIDED	76	233	153	18	16	14	510
V-EXTENDED	60	218	206	18	30	17	549
MP-SIMPLE	195	79	91	22	6	5	398
MP-GUIDED	186	129	134	33	29	2	513
H-FULL	109	232	186	14	32	2	575
H-NEXT	113	232	243	24	38	2	652

Table 3: Errors for each system by type. Systems that were guided by simple schema (V-SIMPLE, MP-SIMPLE) produced the fewest factual mistakes whilst offering the most control.

only including lexical values as input data values). The references texts had out-of-vocab tokens because human authors are not constrained to the set of training words. The MP-SIMPLE and MP-SIMPLE systems both had more out of vocabulary words. Also shown is a count of singleton trigrams (where all three tokens in the trigram are identical), a measure of repetition, where gain the MP-SIMPLE and MP-SIMPLE systems had higher mean counts. In both cases, this is likely due to the incorrect recombination of subwords. It may be possible to adjust the training of models to aleviate this, but it is important to note that automatic metrics all miss this kind of error and it was only found because of our error analysis of human annotated errors.

Shot breakdowns, which are a type of domain specific syntax breaking down the shooting of a player using between 2 and 6 numbers, e.g. *"(4-8 FG, 1-4 3Pt, 2-2 FT)"*, were also counted in Table 4. The number of shot breakdowns (extracted by regular expression) included by the MP-SIMPLE and MP-GUIDED systems could explain part of the increased RG# seen in Table 1. They densely transcribe either 2, 4, or 6 numeric facts yet are simple (once the decision has been made to include one, the structure is deterministic). Systems learn to generate so many shot breakdowns because that they are present in the training data, although they are seldom found in the test set reference texts from the 2018 season. This could be explained by drift due to a change in the specific authors writing the reference texts during that year (Upadhyay and Massie, 2022).

D Crowd-sourced worker recruitment

Participants were recruited on the Amazon Mechanical Turk platform. We used the recruitment policy of Thomson and Reiter (2020) participants were required to hold a US Bachelors degree, be US residents, and be Mechanical Turk Masters workers (a qualification issues by Amazon for high worker reliability). In addition, candidates had to complete a (paid) custom qualification exercise. Fair treatment of crowd-sourced workers is important (Silberman et al., 2018) both from an ethical standpoint and to ensure high quality work. We aimed to pay workers approximately US$20 per-hour for their time, which meant paying $8 for each of the 35 factual accuracy annotation tasks they completed, these take 20-25 minutes to complete. We paid $2 for each of the ordering tasks which take 5-6 minutes to complete. We also paid the same for the any practice work. The same 7 participants completed all work for both our factual accuracy and ordering experiments.

E View Grounding

Given a sentence, we consider all possible view sets as candidates for grounding. We propose to judge the alignment between one view set and the sentence as inversely proportional to the number of *alignment errors* it would entail. An alignment error simply refers to any token that could belong to one of the generated noun phrases but cannot be justified by the data contained in the view set.

To identify individual alignment errors, we first use a simple rule-based system to generate noun phrases based on the data within the view set. This includes phrases based on statistics like '14 points', or alternative forms such as '14-point'. We also include those derived from multiple statistics, e.g., 'double-double'. Named entities are also included, for example, 'Russel Westbrook'. This does introduce a requirement of manual definition, but generating noun phrases for data is a much simpler task than constructing grammar and narrative to connect them. We take the best of both rules and neural, defining that what which is simple and learning that which is complex or time-

	Token Count		Out-of-Vocab Count		Singleton Trigram Count		Shot Breakdown Count	
System	mean	stdev	mean	stdev	mean	stdev	mean	stdev
V-SIMPLE	276	30	0.027	0.19	0.04	0.237	0.151	0.663
V-GUIDED	241	48	0.022	0.147	0.013	0.145	0.178	0.636
V-EXTENDED	340	33	0.026	0.179	0.05	0.253	0.229	0.817
MP-SIMPLE	292	62	2.673	3.625	0.386	3.008	1.551	2.185
MP-GUIDED	309	95	2.108	5.449	0.63	6.157	0.83	1.948
H-FULL	366	71	0	0	0	0	0.191	0.774
H-NEXT	386	94	0	0	0	0	1.142	2.008
GOLD	339	39	0.618	0.958	0	0	0.008	0.134

Table 4: Mean count and standard deviation of tokens, out-of-vocabulary tokens, singleton trigrams (where the set of tokens within the trigram is a singleton), and shot breakdowns per-text.

consuming. Each sentence is parsed token-wise, and once a known noun-phrase (from a global list) is started, it must be able to continue within that view ('14' can continue as '14 points' or '14 - point'), or conclude ('14 - point' must conclude as there is no possible continuation), otherwise it is an error. There will be a small number of cases where the grounding cannot be narrowed down to 1 or 2 compatible views. However, all we require is enough correctly grounded views to introduce a training signal. When there is ambiguity, a model can be instructed to not update weights.

We conclude the view set selection procedure by selecting the smallest one, i.e. in this case the singleton of Westbrook's `<Whole-Game>` view (which had zero errors).