Data-to-text Generation with Macro Planning

Ratish Puduppully and Mirella Lapata
Institute for Language, Cognition and Computation
School of Informatics, University of Edinburgh
10 Crichton Street, Edinburgh EH8 9AB
r.puduppully@sms.ed.ac.uk   mlap@inf.ed.ac.uk

Abstract

Recent approaches to data-to-text generation have adopted the very successful encoder-decoder architecture or variants thereof. These models generate text that is fluent (but often imprecise) and perform quite poorly at selecting appropriate content and ordering it coherently. To overcome some of these issues, we propose a neural model with a *macro planning* stage followed by a generation stage reminiscent of traditional methods which embrace separate modules for planning and surface realization. Macro plans represent high level organization of important content such as entities, events, and their interactions; they are learned from data and given as input to the generator. Extensive experiments on two data-to-text benchmarks (RotoWire and MLB) show that our approach outperforms competitive baselines in terms of automatic and human evaluation.

1 Introduction

Data-to-text generation refers to the task of generating textual output from non-linguistic input (Reiter and Dale, 1997, 2000; Gatt and Krahmer, 2018) such as databases of records, simulations of physical systems, accounting spreadsheets, or expert system knowledge bases. As an example, Figure 1 shows various statistics describing a major league baseball (MLB) game, including extracts from the *box score* (i.e., the performance of the two teams and individual team members who played as batters, pitchers or fielders; Table (A)), *play-by-play* (i.e., the detailed sequence of each play of the game as it occurred; Table (B)), and a human written game *summary* (Table (C)).

Traditional methods for data-to-text generation (Kukich, 1983; McKeown, 1992; Reiter and Dale,

1997) follow a pipeline architecture, adopting separate stages for *text planning* (determining which content to talk about and how it might be organized in discourse), *sentence planning* (aggregating content into sentences, deciding specific words to describe concepts and relations, and generating referring expressions), and *linguistic realization* (applying the rules of syntax, morphology, and orthographic processing to generate surface forms). Recent neural network–based approaches (Lebret et al., 2016; Mei et al., 2016; Wiseman et al., 2017) make use of the encoder-decoder architecture (Sutskever et al., 2014), are trained end-to-end, and have no special-purpose modules for how to best generate a text, aside from generic mechanisms such as attention and copy (Bahdanau et al., 2015; Gu et al., 2016). The popularity of end-to-end models has been further boosted by the release of new datasets with thousands of input-document training pairs. The example shown in Figure 1 is taken from the MLB dataset (Puduppully et al., 2019b), which contains baseball game statistics and human written summaries (∼25K instances). RotoWire (Wiseman et al., 2017) is another widely used benchmark, which contains NBA basketball game statistics and their descriptions (∼5K instances).

Wiseman et al. (2017) show that despite being able to generate fluent text, neural data-to-text generation models are often imprecise, prone to hallucination (i.e., generate text that is not supported by the input), and poor at content selection and document structuring. Attempts to remedy some of these issues focus on changing the way entities are represented (Puduppully et al., 2019b; Iso et al., 2019), allowing the decoder to skip low-confidence tokens to enhance faithful generation (Tian et al., 2019), and making the encoder-decoder architecture more modular by introducing *micro planning* (Puduppully et al., 2019a; Moryossef et al., 2019). Micro planning operates at the record level (see Table (A) in

510

(A)

TEAM	Inn1	Inn2	Inn3	Inn4	...	TR	TH	E	...
Orioles	1	0	0	0	...	2	4	0	...
Royals	1	0	0	3	...	9	14	1	...

BATTER	H/V	AB	BR	BH	RBI	TEAM	...
C.Mullins	H	4	2	2	1	Orioles	...
J.Villar	H	4	0	0	0	Orioles	...
W.Merrifield	V	2	3	2	1	Royals	...
R.O'Hearn	V	5	1	3	4	Royals	...
...		...	...	...	...	...	

PITCHER	H/V	W	L	IP	PH	PR	ER	BB	K	...
A.Cashner	H	4	13	5.1	9	4	4	3	1	...
B.Keller	V	7	5	8.0	4	2	2	2	4	...
...		...	...	...	...	...			

Inn1: runs in innings, TR: team runs, TH: team hits, E: errors, AB: at-bats, RBI: runs-batted-in, BR: batter runs, BH: batter hits, H/V: home or visiting, W: wins, L: losses, IP: innings pitched, PH: hits given, PR: runs given, ER: earned runs, BB: walks, K: strike outs, INN: inning with (T)op/(B)ottom, PL-ID: play id.

(C)

KANSAS CITY, Mo. – Brad Keller kept up his recent pitching surge with another strong outing. <P> Keller gave up a home run to the first batter of the game – Cedric Mullins – but quickly settled in to pitch eight strong innings in the Kansas City Royals' 9–2 win over the Baltimore Orioles in a matchup of the teams with the worst records in the majors. <P> Keller (7–5) gave up two runs and four hits with two walks and four strikeouts to improve to 3–0 with a 2.16 ERA in his last four starts. <P> Ryan O'Hearn homered among his three hits and drove in four runs, Whit Merrifield scored three runs, and Hunter Dozier and Cam Gallagher also went deep to help the Royals win for the fifth time in six games on their current homestand. <P> With the score tied 1–1 in the fourth, Andrew Cashner (4–13) gave up a sacrifice fly to Merrifield after loading the bases on two walks and a single. Dozier led off the fifth inning with a 423-foot home run to left field to make it 3-1. <P> The Orioles pulled within a run in the sixth when Mullins led off with a double just beyond the reach of Dozier at third, advanced to third on a fly ball and scored on Trey Mancini's sacrifice fly to the wall in right. <P> ...

(B)

BATTER	PITCHER	SCORER	ACTION	TEAM	INN	PL-ID	SCORE	...
C.Mullins	B.Keller	-	Home run	Orioles	1-T	1	1	...
H.Dozier	A.Cashner	W.Merrifield	Grounded	Royals	1-B	3	1	...
W.Merrifield	A.Cashner	B.Goodwin	Sac fly	Royals	4-B	5	2	...
H.Dozier	A.Cashner	-	Home run	Royals	5-B	1	3	...
...	...	...	...	...	...	...	...	...

(D)

```
V(Orioles), V(Royals),              V(Royals) V(Orioles),
V(C.Mullins), V(J.Villar),          V(Orioles) V(C.Mullins), V(Orioles) V(J.Villar),
V(W.Merrifield), V(R.O'Hearn),      V(Royals) V(W.Merrifield), V(Royals)
V(A.Cashner), V(B.Keller),          V(R.O'Hearn), V(Orioles) V(A.Cashner), V(Royals)
V(H.Dozier), ...,                   V(B.Keller), ...,
V(1-T), V(1-B), V(2-T), V(2-B),     V(C.Mullins) V(Royals) V(Orioles),
V(3-T), V(3-B), ...                 V(J.Villar) V(Royals) V(Orioles), ...
```

(E)

```
V(B.Keller)<P>V(B.Keller) V(C.Mullins) V(Royals) V(Orioles)<P>V(B.Keller)<P>
V(R.O'Hearn) V(W.Merrifield) V(H.Dozier) V(C.Gallagher) <P>V(4-B, 5-B) <P> V(6-T)<P>
```

Figure 1: MLB statistics tables and game summary. Tables summarize the performance of teams and individual team members who played as batters and pitchers as well as the most important actions (and their actors) in each play (Tables (A) and (B)). Macro plan for the game summary is shown at the bottom (Table (E)). <P> indicates paragraph delimiters. There is a plan for every paragraph in the game summary (correspondence shown in same color); <V(entity)> verbalizes entities, while <V(inning-T/B)> verbalizes events related to the top/bottom side of an inning (see Section 3.1). Set of candidate paragraph plans are shown above macro plan (Table (D)) and grouped into two types: plans describing a single entity/event or their combinations. Best viewed in color.

Figure 1; e.g., C.Mullins BH 2, J.Villar TEAM Orioles), it determines which facts should be mentioned within a textual unit (e.g., a sentence) and how these should be structured (e.g., the sequence of records). An explicit content planner essentially makes the job of the neural network less onerous allowing to concentrate on producing fluent natural language output, without expending too much effort on content organization.

In this work, we focus on *macro planning*, the high-level organization of information and how it should be presented which we argue is important for the generation of long, multi-paragraph documents (see text (C) in Figure 1). Problematically, modern datasets like MLB (Puduppully et al., 2019b; and also Figure 1) and RotoWire (Wiseman et al., 2017) do not naturally lend themselves to document planning as there is no explicit link between the summary and the content of the game (which is encoded in tabular form). In other words, the underlying plans are *latent*, and it is not clear how they might be best represented, namely, as sequences of records from a table, or simply words. Nevertheless, game summaries through their segmentation into paragraphs (and lexical overlap with the input) give clues as to how content might be organized. Paragraphs are a central element of discourse (Chafe, 1979; Longacre, 1979; Halliday and Hasan, 1976), the smallest domain where coherence and topic are defined and anaphora resolution is possible

511

(Zadrozny and Jensen, 1991). We therefore operationalize the macro plan for a game summary as a *sequence* of *paragraph* plans.

Although resorting to paragraphs describes the summary plan at a coarse level, we still need to specify individual paragraph plans. In the sports domain, paragraphs typically mention entities (e.g., players important in the game), key events (e.g., scoring a run), and their interaction. And most of this information is encapsulated in the statistics accompanying game summaries (see Tables (A) and (B) in Figure 1). We thus define paragraph plans such that they contain verbalizations of entity and event records (see plan (E) in Figure 1). Given a *set* of paragraph plans and their corresponding game summary (see Table (D) and summary (C) in Figure 1), our task is twofold. At training time, we must *learn* how content was selected in order to give rise to specific game summaries (e.g., how input (D) led to plan (E) for summary (C) in Figure 1), while at test time, given input for a new game, we first predict a macro plan for the summary and then generate the corresponding document.

We present a two-stage approach where macro plans are induced from training data (by taking the table and corresponding summaries into account) and then fed to the text generation stage. Aside from making data-to-text generation more interpretable, the task of generating a document from a macro plan (rather than a table) affords greater control over the output text and plays to the advantage of encoder-decoder architectures which excel at modeling sequences. We evaluate model performance on the ROTOWIRE (Wiseman et al., 2017) and MLB (Puduppully et al., 2019b) benchmarks. Experimental results show that our plan-and-generate approach produces output that is more factual, coherent, and fluent compared with existing state-of-the-art models. Our code, trained models, and dataset with macro plans can be found at `https://github.com/ratishsp/data2text-macro-plan-py`.

2   Related Work

Content planning has been traditionally considered a fundamental component in natural language generation. Not only does it determine which information-bearing units to talk about, but also arranges them into a structure that creates coherent output. Many content planners have been based on theories of discourse coherence (Hovy, 1993), schemas (McKeown et al., 1997), or have relied on generic planners (Dale, 1989). Plans are mostly based on hand-crafted rules after analyzing the target text, although a few approaches have recognized the need for learning-based methods. For example, Duboue and McKeown (2001) learn ordering constraints in a content plan, Konstas and Lapata (2013) represent plans as grammar rules whose probabilities are estimated empirically, while others make use of semantically annotated corpora to bootstrap content planners (Duboue and McKeown, 2002; Kan and McKeown, 2002).

More recently, various attempts have been made to improve neural generation models (Wiseman et al., 2017) based on the encoder-decoder architecture (Bahdanau et al., 2015) by adding various planning modules. Puduppully et al. (2019a) propose a model for data-to-text that first learns a plan from the records in the input table and then generates a summary conditioned on this plan. Shao et al. (2019) introduce a Planning-based Hierarchical Variational Model where a plan is a sequence of groups, each of which contains a subset of input items to be covered in a sentence. The content of each sentence is verbalized, conditioned on the plan and previously generated context. In their case, input items are a relatively small list of attributes ($\sim$28) and the output document is also short ($\sim$110 words).

There have also been attempts to incorporate neural modules in a pipeline architecture for data-to-text generation. Moryossef et al. (2019) develop a model with a symbolic text planning stage followed by a neural realization stage. They experiment with the WebNLG dataset (Gardent et al., 2017) which consists of RDF $\langle$ Subject, Object, Predicate $\rangle$ triples paired with corresponding text. Their document plan is a sequence of sentence plans that in turn determine the division of facts into sentences and their order. Along similar lines, Castro Ferreira et al. (2019) propose an architecture composed of multiple steps including discourse ordering, text structuring, lexicalization, referring expression generation, and surface realization. Both approaches show the effectiveness of pipeline architectures, however, their task does not require content selection and the output texts are relatively short (24 tokens on average).

Although it is generally assumed that task-specific parallel data is available for model training, Laha et al. (2020) do away with this assumption and present a three-stage pipeline model which learns from monolingual corpora. They first convert the input to a form of tuples, which in turn are expressed in simple sentences, followed by the third stage of merging simple sentences to form more complex ones by aggregation and referring expression generation. They also evaluate on data-to-text tasks which have relatively short outputs. There have also been efforts to improve the coherence of the output, especially when dealing with longer documents. Puduppully et al. (2019b) make use of hierarchical attention over entity representations which are updated dynamically, while Iso et al. (2019) explicitly keep track of salient entities and memorize which ones have been mentioned.

Our work also attempts to alleviate deficiencies in neural data-to-text generation models. In contrast to previous approaches, (Puduppully et al., 2019a; Moryossef et al., 2019; Laha et al., 2020), we place emphasis on *macro planning* and create plans representing high-level organization of a document including both its content *and* structure. We share with previous work (e.g., Moryossef et al. 2019) the use of a two-stage architecture. We show that macro planning can be successfully applied to *long* document data-to-text generation resulting in improved factuality, coherence, and fluency without any postprocessing (e.g., to smooth referring expressions) or recourse to additional tools (e.g., parsing or information extraction).

3 Problem Formulation

We hypothesize that generation based on plans should fare better compared to generating from a set of records, since macro plans offer a bird's-eye view, a high-level organization of the document content and structure. We also believe that macro planning will work well for long-form text generation, that is, for datasets that have multi-paragraph target texts, a large vocabulary space, and require content selection.

We assume the input to our model is a set of paragraph plans $\mathcal{E} = \{e_i\}_{i=1}^{|\mathcal{E}|}$ where e_i is a paragraph plan. We model the process of generating output summary y given $\mathcal{E}$ as a two step process, namely, the construction of a macro plan x based on the set of paragraph plans, followed by the generation of a summary given a macro plan as input. We now explain how $\mathcal{E}$ is obtained and each step is realized. We discuss our model considering mainly an example from the MLB dataset (Puduppully et al., 2019b) but also touch on how the approach can be straightforwardly adapted to RotoWire (Wiseman et al., 2017).

3.1 Macro Plan Definition

A macro plan consists of a sequence of paragraph plans separated by a paragraph discourse marker <P>, that is, $x = e_i$ <P> $e_j \ldots$ <P> e_k where $e_i, e_j, e_k \in \mathcal{E}$. A paragraph plan in turn is a sequence of entities and events describing the game. By *entities* we mean individual players or teams and the information provided about them in box score statistics (see rows and column headings in Figure 1 Table (A)), while *events* refer to information described in play-by-play (see Table (B)). In baseball, plays are grouped in half-innings. During each half of an inning, a team takes its turn to bat (the visiting team bats in the top half and the home team in the bottom half). An example macro plan is shown at the bottom of Figure 1. Within a paragraph plan, entities and events are verbalized into a *text sequence* along the lines of Saleh et al. (2019). We make use of special tokens for the <TYPE> of record followed by the value of record from the table. We retain the same position for each record type and value. For example, batter *C.Mullins* from Figure 1 would be verbalized as <PLAYER>C.Mullins <H/V>H <AB>4
2 <BH>2 <RBI> 1 <TEAM>Orioles .... For the sake of brevity we use shorthand <V(C.Mullins)> for the full entity.

Paragraph Plan for Entities For a paragraph containing entities, the corresponding plan will be a verbalization of the entities in sequence. For paragraphs with multiple mentions of the same entity, the plan will verbalize an entity only once and at its first position of mention. Paragraph ''*Keller gave up a home run . . . the teams with the worst records in the majors*'' from the summary in Figure 1 describes four entities including *B. Keller*, *C. Mullins*, *Royals* and *Orioles*. The respective plan is the verbalization of the four entities in sequence: <V(B.Keller)> <V(C.Mullins)> <V(Royals)> <V(Orioles)>, where V stands for verbalization and <V(B. Keller)> is a

shorthand for <PLAYER>B.Keller <H/V>V <W>7 <L>5 <IP>8 ..., <V(Royals)> is a shorthand for the team <TEAM>Royals <TR>9 <TH>14 <E>1, and so on.

Paragraph Plan for Events A paragraph may also describe one or more events. For example, the paragraph ''*With the score tied 1–1 in the fourth . . . 423-foot home run to left field to make it 3-1*'' discusses what happened in the bottom halves of the *fourth* and *fifth* innings. We verbalize an event by first describing the participating entities followed by the plays in the event. Entities are described in the order in which they appear in a play, and within the same play we list the batter followed by the pitcher, fielder, scorer, and basemen. The paragraph plan corresponding to the bottom halves of the *fourth* and *fifth* inning is <V(4-B, 5-B)>. Here, <V(4-B, 5-B)> is a shorthand for <V(W.Merrifield)> <V(A.Cashner)> <V(B.Goodwin)> <V(H. Dozier)> ... <V(4-B,1)> <V(4-B,2)> ... <V(5-B,1)> <V(5-B,2)>, and so on. The entities <V(W.Merrifield)>, <V(A.Cashner)>, <V(B.Goodwin)>, and <V(H.Dozier)> correspond in turn to *W. Merrifield, A. Cashner, B. Goodwin*, and *H. Dozier* while <V(5-B,1)> refers to the first play in the bottom half of the fifth inning (see the play-by-play table in Figure 1) and abbreviates the following detailed plan: <INN>5 <HALF>B <BATTING>Royals <PITCHING>Orioles <PL-ID>1 <BATTER> H.Dozier <PITCHER>A. Cashner> <ACTION> Home-run <SCORES> Royals-3-Orioles-1, and so forth.

The procedure described above is not specific to MLB and can be ported to other datasets with similar characteristics such as ROTOWIRE. However, ROTOWIRE does not provide play-by-play information, and as a result there is no event verbalization for this dataset.

3.2 Macro Plan Construction

We provided our definition for macro plans in the previous sections, however, it is important to note that such macro plans are not readily available in data-to-text benchmarks like MLB (Puduppully et al., 2019b) and ROTOWIRE (Wiseman et al., 2017) which consist of tables of records r paired with a gold summary y (see Tables (A)–(C) in Figure 1). We now describe our method for obtaining macro plans x from r and y.

Similar to Moryossef et al. (2019), we define macro plans to be conformant with gold summaries such that (1) they have the same splits into paragraphs—entities and events within a paragraph in y are grouped into a paragraph plan in x; and (2) the order of events and entities in a paragraph and its corresponding plan are identical. We construct macro plans by matching entities and events in the summary to records in the tables. Furthermore, paragraph delimiters within summaries form natural units which taken together give rise to a high-level document plan.

We match entities in summaries with entities in tables using exact string match, allowing for some degree of variation in the expression of team names (e.g., *A's* for *Athletics* and *D-backs* for *Diamondbacks*). Information pertaining to innings appears in the summaries in the form of ordinal numbers (e.g., *first, ninth*) modifying the noun *inning* and can be relatively easily identified via pattern matching (e.g., in sentences like ''*Dozier led off the **fifth** inning*''). However, there are instances where the mention of innings is more ambiguous (e.g., ''*With the scored tied 1–1 in the **fourth**, Andrew Cashner (4–13) gave up a sacrifice fly*''). We could disambiguate such mentions manually and then train a classifier to learn to predict whether an inning is mentioned. Instead, we explore a novel annotation-free method that makes use of the pretrained language model GPT2 (Radford et al., 2019). Specifically, we feed the context preceding the ordinal number to GPT2 (i.e., the current paragraph up to the ordinal number and the paragraph preceding it) and if *inning* appears in the top 10 next word predictions, we consider it a positive match. On a held-out dataset, this method achieves 98% precision and 98% recall at disambiguating inning mentions.

To resolve whether the summary discusses the top or bottom side of an inning, we compare the entities in the paragraph with the entities in each half-inning (play-by-play Table (B) in Figure 1) and choose the side with the greater number of entity matches. For instance, *Andrew Cashner, Merrifield* and *fourth* inning uniquely resolves to the bottom half of the *fourth* inning.

3.3 Paragraph Plan Construction

Figure 1 shows the macro plan we obtain for game summary (C). Importantly, macro plan (E) is the outcome of a content selection process after

514

considering several candidate paragraph plans as input. So, what are the candidate paragraph plans that give rise to macro plan (E)? To answer this question, we examined the empirical distribution of paragraph plans in MLB and RotoWire (training portion). Interestingly, we found that ∼79% of the paragraph plans in MLB refer to a single event or a single player (and team(s)). In RotoWire, ∼92% of paragraphs are about a singleton player (and team(s)) or a pair of players.

Based on this analysis, we assume that paragraph plans can be either one (verbalized) entity/event or a combination of at most two. Under this assumption, we explicitly enumerate the set of candidate paragraph plans in a game. For the game in Figure 1, candidate paragraph plans are shown in Table (D). The first table groups plans based on individual verbalizations describing the team(s), players, and events taking place in specific innings. The second table groups pairwise combinations thereof. In MLB, such combinations are between team(s) and players. In RotoWire, we also create combinations between players. Such paragraph plans form set $\mathcal{E}$ based on which macro plan x is constructed to give rise to game summary y.

4 Model Description

The input to our model is a set of paragraph plans, each of which is a sequence of tokens. We first compute paragraph plan representations $\in \mathbb{R}^n$, and then apply a contextualization and content planning mechanism similar to planning modules introduced in earlier work (Puduppully et al., 2019a; Chen and Bansal, 2018). Predicted macro plans serve as input to our text generation model, which adopts an encoder-decoder architecture (Bahdanau et al., 2015; Luong et al., 2015).

4.1 Macro Planning

Paragraph Plan Representation We encode tokens in a verbalized paragraph plan e_i as $\{\mathbf{e}_{i,j}\}_{j=1}^{|e_i|}$ with a BiLSTM (Figure 2, bottom part). To reflect the fact that some records will be more important than others, we compute an attention weighted sum of $\{\mathbf{e}_{i,j}\}_{j=1}^{|e_i|}$ following Yang et al. (2016). Let $\mathbf{d} \in \mathbb{R}^n$ denote a randomly initialized query vector learnt jointly with the rest of parameters. We compute attention values $\alpha_{i,j}$ over $\mathbf{d}$ and paragraph plan token representation $\mathbf{e}_{i,j}$:

$$\alpha_{i,j} \propto \exp(\mathbf{d}^\mathsf{T}\mathbf{e}_{i,j}) \tag{1}$$
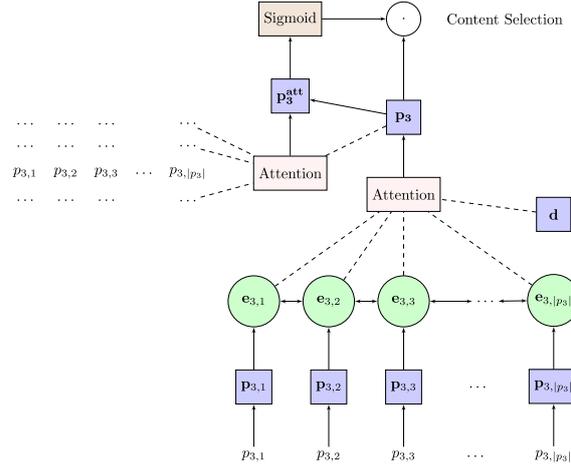

Figure 2: Paragraph plan representation and contextualization for macro planning. Computation of $\mathbf{e}_3$ is detailed in Equations (1) and (2), $\mathbf{e}_3^{att}$ in Equation (3), and $\mathbf{e}_3^c$ in Equation (4).

Paragraph plan vector $\mathbf{e}_i$ is the attention weighted sum of $\mathbf{e}_{i,j}$ (with $\sum_j \alpha_{i,j} = 1$):

$$\mathbf{e}_i = \sum_j \alpha_{i,j}\mathbf{e}_{i,j} \tag{2}$$

Next, we contextualize each paragraph plan representation vis-a-vis other paragraph plans (Figure 2, top left part). First, we compute attention scores $\beta_{i,k}$ over paragraph plan representations to obtain an attentional vector $\mathbf{e}_i^{att}$ for each:

$$\beta_{i,k} \propto \exp(\mathbf{e}_i^\mathsf{T}\mathbf{W}_a\mathbf{e}_k)$$
$$\mathbf{c}_i = \sum_{k \neq i} \beta_{i,k}\mathbf{e}_k$$
$$\mathbf{e}_i^{att} = \mathbf{W}_g[\mathbf{e}_i; \mathbf{c}_i] \tag{3}$$

where $\mathbf{W}_a \in \mathbb{R}^{n \times n}, \mathbf{W}_g \in \mathbb{R}^{n \times 2n}$ are parameter matrices, and $\sum_{k \neq i} \beta_{i,k} = 1$. Then, we compute a content selection gate, and apply this gate to $\mathbf{e}_i$ to obtain new paragraph plan representation $\mathbf{e}_i^c$:

$$\mathbf{g}_i = \text{sigmoid}\left(\mathbf{e}_i^{att}\right)$$
$$\mathbf{e}_i^c = \mathbf{g}_i \odot \mathbf{e}_i \tag{4}$$

where $\odot$ denotes element-wise multiplication. Thus, each element in $\mathbf{e}_i$ is weighted by corresponding element of $\mathbf{g}_i \in [0, 1]^n$ to obtain a contextualized paragraph plan representation $\mathbf{e}_i^c$.

Content Planning Our model learns to predict macro plans, after having been trained on pairs of sets of paragraph plans and corresponding
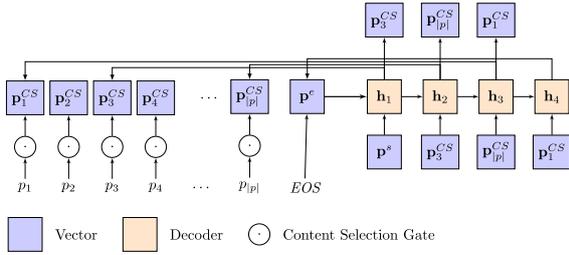
515

Figure 3: Macro planning model; paragraph plan representation and contextualization mechanism are detailed in Figure 2. The output points to e_3, $e_{|\mathcal{E}|}$, and e_1 (see Equations (5) and (6)). *EOM* is end of macro plan token.

macro plans (Sections 3.2 and 3.3 explain how we obtain these for data-to-text datasets like ROTOWIRE and MLB). More formally, we model macro plan $z = z_1 \ldots z_{|z|}$ as a sequence of pointers, with each z_k pointing to an input paragraph plan, i.e., $z_k \in \{e_i\}_{i=1}^{|\mathcal{E}|}$. We decompose $p(z|\mathcal{E})$, the probability of macro plan z given paragraph plans $\mathcal{E}$, as:

$$p(z|\mathcal{E}) = \prod_{k=1}^{|z|} p(z_k|z_{<k}, \mathcal{E}) \qquad (5)$$

where $z_{<k} = z_1 \ldots z_{k-1}$.

We use Pointer Networks (Vinyals et al., 2015) to model $p(z_k|z_{<k}, \mathcal{E})$ as:

$$p(z_k = e_i|z_{<k}, \mathcal{E}) \propto \exp(\mathbf{h}_k^\intercal \mathbf{W}_b \mathbf{e}_i^c) \qquad (6)$$

where $p(z_k|z_{<k}, \mathcal{E})$ is normalized to 1 and $\mathbf{W}_b \in \mathbb{R}^{n \times n}$. Rather than computing a weighted representation, Pointer Networks make use of attention to point to specific elements in the input (see Figure 3). We use a decoder LSTM to compute hidden representation $\mathbf{h}_k$ at time step k. We initialize $\mathbf{h}_0$ with the mean paragraph plan representation, $\mathrm{avg}(\{\mathbf{e}_i^c\}_{i=1}^{|\mathcal{E}|})$. Once the output points to e_i, its representation $\mathbf{e}_i^c$ is used as input to the next step of the LSTM decoder. The process stops when the model points to *EOM*, a token indicating end of the macro plan.

4.2 Text Generation

Recall that z is a sequence of pointers with each entry z_k pointing to a paragraph plan, namely, $z_k \in \{e_i\}_{i=1}^{|\mathcal{E}|}$. We can deterministically obtain macro plan x from z by retrieving the paragraph plans being pointed to, adding <P>

separators in between. The conditional output probability $p(y|x)$ is modeled as:

$$p(y|x) = \prod_{t=1}^{|y|} p(y_t|y_{<t}, x)$$

where $y_{<t} = y_1 \ldots y_{t-1}$.

To compute $p(y|x)$, we use an encoder-decoder architecture enhanced with an attention mechanism (Bahdanau et al., 2015; Luong et al., 2015). We encode macro plan x with a bidirectional LSTM (Hochreiter and Schmidhuber, 1997). At time step t, we lookup the embedding of the previously predicted word y_{t-1} and feed it as input to the decoder, which is another LSTM unit. The decoder attends over hidden states of the macro plan to predict y_t. We further incorporate a copy mechanism (Gulcehre et al., 2016) in the decoder to enable copying values directly from the macro plan.

We expect the text generation model to learn to generate summary tokens while focusing on the corresponding macro plan and that the output summary will indeed follow the plan in terms of the entities and events being described and their order. At the same time, we believe that text generation is relatively easier as the encoder-decoder model is relieved from the tasks of document structuring and information selection.

4.3 Training and Inference

We train two independent models for macro planning and text generation. Our training objective for macro planning aims to maximize the log likelihood of the macro plan given the paragraph plans:

$$\max_\theta \sum_{(\mathcal{E}, z) \in \mathcal{D}} \log p\left(z|\mathcal{E}; \theta\right)$$

where $\mathcal{D}$ is the training set consisting of pairs of (sets of) paragraph plans and macro plans, and θ are model parameters.

Our training objective for text generation aims to maximize the log likelihood of the output text given the macro plan:

$$\max_\phi \sum_{(x, y) \in \mathcal{F}} \log p\left(y|x; \phi\right)$$

where $\mathcal{F}$ is the training set consisting of pairs of macro plans and game summaries, and ϕ are model parameters.

516

	RotoWire	MLB
Vocab Size	11.3K	38.9K
# Tokens	1.5M	14.3M
# Instances	4.9K	26.3K
# Record Types	39	53
Avg Records	628	565
Avg Paragraph Plans	10.7	15.1
Avg Length	337.1	542.05

Table 1: Dataset statistics for RotoWire and MLB. Vocabulary size, number of tokens, number of instances (i.e., table-summary pairs), number of record types, average number of records, average number of paragraph plans, and average summary length.

During inference, we employ beam search to find the most likely macro plan $\hat{z}$ among candidate macro plans z' given paragraph plans as input.

$$\hat{z} = \arg\max_{z'} p(z'|\mathcal{E}; \theta)$$

We deterministically obtain $\hat{x}$ from $\hat{z}$, and output summary $\hat{y}$ among candidate outputs y' given macro plan $\hat{x}$ as input:

$$\hat{y} = \arg\max_{y'} p(y'|\hat{x}; \phi)$$

5 Experimental Setup

Data We performed experiments on the RotoWire (Wiseman et al., 2017) and MLB (Puduppully et al., 2019b) benchmarks. The details of these two datasets are given in Table 1. We can see that MLB is around 5 times bigger, has a richer vocabulary, and has longer game summaries. We use the official splits of 3,398/727/728 for RotoWire and 22,821/1,739/1,744 for MLB. We make use of a tokenization script[1] to detokenize and retokenize the summaries in both RotoWire and MLB.

We reconstructed the MLB dataset, as the version released by Puduppully et al. (2019b) had removed all paragraph delimiters from game summaries. Specifically, we followed their methodology and downloaded the same summaries from the ESPN Web site[2] and added the `<P>` delimiter to paragraphs in the summaries.[3]

RotoWire does not have paragraph delimiters in game summaries either. We reverse engineered these as follows: (1) we split summaries into sentences using the NLTK (Bird et al., 2009) sentence tokenizer; (2) initialized each paragraph with a separate sentence; (3) merged two paragraphs into one if the entities in the former were a superset of entities in the latter; (4) repeated Step 3 until no merges were possible.

Training Configuration We tuned the model hyperparameters on the development set. For training the macro planning and the text generation stages, we used the Adagrad (Duchi et al., 2011) optimizer. Furthermore, the text generation stage made use of truncated BPTT (Williams and Peng, 1990) with truncation length 100. We learn subword vocabulary (Sennrich et al., 2016) for paragraph plans in the macro planning stage. We used 2.5K merge operations for RotoWire and 8K merge operations for MLB. In text generation, we learn a joint subword vocabulary for the macro plan and game summaries. We used 6K merge operations for RotoWire and 16K merge operations for MLB. All models were implemented on OpenNMT-py (Klein et al., 2017). We add to set $\mathcal{E}$ the paragraph plans corresponding to the output summary paragraphs, to ensure full coverage during training of the macro planner. During inference for predicting macro plans, we employ length normalization (Bahdanau et al., 2015) to avoid penalizing longer outputs; specifically, we divide the scores of beam search by the length of the output. In addition, we adopt bigram blocking (Paulus et al., 2018). For MLB, we further block beams containing more than two repetitions of a unigram. This helps improve the diversity of the predicted macro plans.

System Comparisons We compared our model against the following systems: (1) the **Template**-based generators from Wiseman et al. (2017) for RotoWire and Puduppully et al. (2019b) for MLB. Both systems apply the same principle, they emit a sentence about the teams playing in the game, followed by player-specific sentences, and a closing sentence. MLB additionally contains a description of play-by-play; (2) **ED+CC**, the best performing system in Wiseman et al. (2017), is a vanilla encoder-decoder model equipped with an attention and copy mechanism; (3) **NCP+CC**, the micro planning model of Puduppully et al.

---

[1] https://github.com/neulab/DGT.

[2] http://www.espn.com/mlb/recap?gameId={gameid}.

[3] Although our model is trained on game summaries with paragraph delimiters, and also predicts these at generation time, for evaluation we strip `<P>` from model output.

(2019a), generates content plans from the table by making use of Pointer networks (Vinyals et al., 2015) to point to records; content plans are encoded with a BiLSTM and the game summary is decoded using another LSTM with attention and copy; (4) **ENT**, the entity-based model of Puduppully et al. (2019b), creates dynamically updated entity-specific representations; the text is generated conditioned on the data input and entity memory representations using hierarchical attention at each time step.

6 Results

Automatic Evaluation For automatic evaluation, following earlier work (Wiseman et al. 2017; Puduppully et al. 2019a,b, inter alia) we report BLEU (Papineni et al., 2002) with the gold summary as reference but also make use of the Information Extraction (IE) metrics from Wiseman et al. (2017), which are defined over the output of an IE system; the latter extracts entity (players, teams) and value (numbers) pairs in a summary, and then predicts the *type* of relation. For instance, given the pair *Kansas City Royals, 9*, it would predict their relation as *TR* (i.e., Team Runs). Training data for the IE system is obtained by checking for matches between *entity, value* pairs in the gold summary and *entity, value, record type* triplets in the table.

Let $\hat{y}$ be the gold summary and y the model output. *Relation Generation* (RG) measures the precision and count of relations extracted from y that also appear in records r. *Content Selection* (CS) measures the precision and recall of relations extracted from y that are also extracted from $\hat{y}$. *Content Ordering* (CO) measures the normalized Damerau-Levenshtein distance between the sequences of relations extracted from y and $\hat{y}$.

We reused the IE model from Puduppully et al. (2019a) for RotoWire but retrained it for MLB to improve its precision and recall. Furthermore, the implementation of Wiseman et al. (2017) computes RG, CS, and CO excluding duplicate relations. This artificially inflates the performance of models whose outputs contain repetition. We include duplicates in the computation of the IE metrics (and recreate them for all comparison systems).

Table 2 (top) presents our results on the RotoWire test set. In addition to Templ, NCP+CC, ENT, and ED+CC we include the

RotoWire	RG		CS			CO	BLEU
	#	P%	P%	R%	F%	DLD%	
Templ	**54.3**	**99.9**	27.1	57.7	36.9	13.1	8.46
WS-2017	34.1	75.1	20.3	36.3	26.1	12.4	14.19
ED+CC	35.9	82.6	19.8	33.8	24.9	12.0	14.99
NCP+CC	40.8	87.6	28.0	51.1	36.2	15.8	16.50
ENT	32.7	91.7	**34.7**	48.5	40.5	16.6	16.12
RBF-2020	44.9	89.5	23.9	47.0	31.7	14.3	**17.16**
Macro	42.1	97.6	34.1	**57.8**	**42.9**	**17.7**	15.46
−Plan(4)	36.2	81.3	22.1	38.6	28.1	12.1	14.00

MLB	RG		CS			CO	BLEU
	#	P%	P%	R%	F%	DLD%	
Templ	**62.3**	**99.9**	21.6	**55.2**	31.0	11.0	4.12
ED+CC	32.5	91.3	27.8	40.6	33.0	17.1	9.68
NCP+CC	19.6	81.3	**44.5**	44.1	44.3	**21.9**	9.68
ENT	23.8	81.1	40.9	49.5	44.8	20.7	11.50
Macro	30.8	94.4	40.8	54.9	**46.8**	21.8	**12.62**
−Plan(SP,4)	25.1	92.7	40.0	44.6	42.2	**21.9**	11.09

Table 2: Evaluation on RotoWire and MLB test sets; relation generation (RG) count (#) and precision (P%), content selection (CS) precision (P%), recall (R%) and F-measure (F%), content ordering (CO) in normalized Damerau-Levenshtein distance (DLD%), and BLEU.

best performing model of Wiseman et al. (2017) (WS-2017; note that ED+CC is an improved re-implementation of their model), and the model of Rebuffel et al. (2020) (RBF-2020), which represents the state of the art on RotoWire. This model has a Transformer encoder (Vaswani et al., 2017) with a hierarchical attention mechanism over entities and records within entities. The models of Saleh et al. (2019), Iso et al. (2019), and Gong et al. (2019) make use of additional information not present in the input (e.g., previous/next games, summary writer) and are not directly comparable to the systems in Table 2. Results for the MLB test set are in the bottom portion of Table 2.

Templ has the highest RG precision and count on both datasets. This is not surprising, by design Templ is always faithful to the input. However, notice that it achieves the lowest BLEU among comparison systems, indicating that it mostly regurgitates facts with low fluency. Macro achieves the highest RG precision among all neural models for RotoWire and MLB. We obtain an absolute improvement of 5.9% over ENT for RotoWire and 13.3% for MLB. In addition, Macro achieves the highest CS F-measure for both datasets. On RotoWire, Macro achieves the highest CO score, and the highest BLEU on MLB. On RotoWire, in terms of BLEU, Macro is worse

than comparison models (e.g., NCP+CC or ENT). Inspection of the output showed that the opening paragraph, which mostly describes how the two teams fared, is generally shorter in Macro, leading to shorter summaries and thus lower BLEU. There is high variance in the length of the opening paragraph in the training data and Macro verbalizes the corresponding plan conservatively. Ideas such as length normalization (Wu et al., 2016) or length control (Kikuchi et al., 2016; Takeno et al., 2017; Fan et al., 2018) could help alleviate this; however, we do not pursue them further for fair comparison with the other models.

The Contribution of Macro Planning To study the effect of macro planning in more detail, we further compared Macro against text generation models (see Section 4.2) which are trained on verbalizations of the tabular data (and gold summaries) but do not make use of document plans or a document planning mechanism. On RotoWire, the model was trained on verbalizations of players and teams, with the input arranged such that the verbalization of the home team was followed by the visiting team, the home team players and the visiting team players. Mention of players was limited to the four best ones, following Saleh et al. (2019) (see −Plan(4) in Table 2). For MLB, we additionally include verbalizations of innings focusing on scoring plays which are likely to be discussed in game summaries (see −Plan(SP,4) in Table 2). Note that by preprocessing the input in such a way some simple form of content selection takes place simply by removing extraneous information which the model does not need to consider.

Across both datasets, −Plan variants appear competitive. On RotoWire −Plan(4) is better than ED+CC in terms of content selection but worse compared to ENT. On MLB, −Plan(SP,4) is again superior to ED+CC in terms of content selection but not ENT whose performance lags behind when considering RG precision. Taken together, these results confirm that verbalizing entities and events into a text sequence is effective. At the same time, we see that −Plan variants are worse than Macro across most metrics which underlines the importance of an explicit planning component.

Table 3 presents intrinsic evaluation of the macro planning stage. Here, we compare the inferred macro plan with the gold macro plans, CS and CO metrics with regard to entities and events

Macro	CS-P	CS-R	CS-F	CO
RotoWire	81.3	73.2	77.0	45.8
MLB	80.6	63.3	70.9	31.4

Table 3: Evaluation of macro planning stage; content selection precision (CS-P), recall (CS-R), F-measure (CS-F), and content ordering (CO) between the inferred plans and gold plans in terms of entities and events for RotoWire (RW) and MLB test sets.

instead of relations. We see that our macro planning model (Macro) achieves high scores for CS and CO for both RotoWire and MLB. We further used the CS and CO metrics to check how well the generated summary follows the (predicted) plan. We followed the steps in Section 3.2 and reverse engineered macro plans from the model summaries and compared these extracted plans with the original macro plans with regard to entities and events. We found that Macro creates summaries that follow the plan closely: For RotoWire, the CS F-score and CO are greater than 98%; for MLB, the CS F-score is greater than 94% and CO is greater than 89%. We show an output summary for Macro in Table 4, together with the predicted document plan.

Human-Based Evaluation We also asked participants to assess model output in terms of relation generation, grammaticality, coherence, and conciseness (Wiseman et al., 2017; Puduppully et al., 2019a; Puduppully et al., 2019b), For RotoWire, we compared Macro against RBF-2020,[4] ED+CC, Gold, and Templ. For MLB, we compared Macro against ENT, ED+CC, Gold, and Templ.

We conducted our study on the Amazon Mechanical Turk (AMT) crowdsourcing platform, following best practices for human evaluation in NLG (van der Lee et al., 2019). Specifically, to ensure consistent ratings, we required crowdworkers to have an approval rating greater than 98% and a minimum of 1,000 previously completed tasks. Raters were restricted to English-speaking countries (i.e., US, UK, Canada, Ireland, Australia, or NZ). Participants were allowed to provide feedback on the task or field questions (our interface accepts free text).

---

[4]We are grateful to Clément Rebuffel for providing us with the output of their system.

<V(Rays)> <P> <V(8-B)> <P> <V(Rays)> <P> <V(Rays)> <V(Red Sox)> <P> <V(8-B)> <P> <V(8-T)> <P> <V(9-T)> <P> <V(Clay Buchholz)> <P> <V(5-T)> <P> <V(Edwin Jackson)> <P> <V(5-T)> <P> <V(8-T)>

ST. PETERSBURG, Fla. (AP) – The **Tampa Bay Rays** are making the most of it. <P> Akinori Iwamura hit a two-run homer in the **eighth** inning and the Rays beat the Boston Red Sox 2-1 on Sunday to complete a three-game sweep. <P> The **Rays**, who have the best record in the majors, have won six of their last seven games. <P> The **Rays** have won four of their last five series, including three in a row against the **Red Sox**, who have won six of their last seven overall. <P> Dioner Navarro singled with one out in the **eighth** off Clay Buchholz (1-2) and moved to third on Jason Bartlett's flyout to center. Iwamura then drove a 1-1 pitch into the left-field stands for his second homer of the season. <P> Scott Dohmann (2-0) got the win in relief, striking out Manny Ramirez with runners on first and third to end the **eighth**. <P> Troy Percival worked the **ninth** for his fifth save in five opportunities. <P> **Clay Buchholz** (1-2) gave up two runs and three hits in eight innings. He struck out nine and walked two. <P> The Red Sox loaded the bases with one out in the **fifth** on a single by Coco Crisp, a wild pitch and a walk to Jed Lowrie. Jacoby Ellsbury drove in Crisp with a two-out single to center. <P> **Jackson** struck out four and walked three. <P> The Red Sox loaded the bases with one out in the **fifth** on a single by Coco Crisp, a walk to Jed Lowrie and a one-out walk to Jed Lowrie. Jackson struck out Julio Lugo, but Jacoby Ellsbury singled to center to put the Red Sox up 1-0. <P> The Red Sox threatened in the **eighth** when J. D. Drew drew a two-out walk against Trever Miller, but Ramirez struck out to end the inning.

Table 4: Predicted macro plan (top) with corresponding model output (bottom). Entities and events in summary corresponding to those in the macro plan are boldfaced.

RotoWire	#Supp	#Contra	Gram	Coher	Concis
Gold	3.63	0.07	38.33	46.25*	30.83
Templ	7.57*	0.08	−61.67*	−52.92*	−36.67*
ED+CC	3.92	0.91*	5.0	−8.33	−4.58
RBF-2020	5.08*	0.67*	13.33	4.58	3.75
Macro	4.00	0.27	5.0	10.42	6.67

MLB	#Supp	#Contra	Gram	Coher	Concis
Gold	3.59	0.14	21.67	30.0	26.67
Templ	4.21	0.04	−51.25*	−43.75*	7.5
ED+CC	3.42	0.72*	−22.5*	−12.08*	−39.17*
ENT	3.71	0.73*	5.83*	−0.83*	−22.08*
Macro	3.76	0.25	46.25	26.67	27.08

Table 5: Average number of supported (#Supp) and contradicting (#Contra) facts in game summaries and *best-worst scaling* evaluation (higher is better). Systems significantly different from Macro are marked with an asterisk * (using a one-way ANOVA with post hoc Tukey HSD tests; $p \leq 0.05$).

In our first study, we presented crowdworkers with sentences randomly selected from summaries along with their corresponding box score (and play-by-play in case of MLB) and asked them to count supported and contradicting facts (ignoring hallucinations, i.e., unsupported facts). We did not require crowdworkers to be familiar with NBA or MLB. Instead, we provided a cheat sheet explaining the semantics of box score tables. In addition, we provided examples of sentences with supported/contradicting facts. We evaluated 40 summaries from the test set (20 per dataset), 4 sentences from each summary and elicited 3 responses per summary. This resulted in 40 summaries × 5 systems × 3 raters, for a total of 600 tasks. Altogether, 131 crowdworkers participated in this study (agreement using Krippendorff's α was 0.44 for supported and 0.42 for contradicting facts).

As shown in Table 5, Macro yields the smallest number of contradicting facts among neural models on both datasets. On RotoWire the number of *contradicting* facts for Macro is comparable to Gold and Templ (the difference is not statistically significant) and significantly smaller compared to RBF-2020 and ED+CC. The count

of *supported* facts for Macro is comparable to Gold, and ED+CC, and significantly lower than Templ and RBF-2020. On MLB, Macro has significantly fewer *contradicting* facts than ENT and ED+CC and is comparable to Templ and Gold (the difference is not statistically significant). The count of *supported* facts for Macro is comparable to Gold, ENT, ED+CC, and Templ. For both datasets, Templ has the lowest number of contradicting facts. This is expected as Templ essentially parrots facts (aka records) from the table.

We also conducted a second study to evaluate the quality of the generated summaries. We presented crowdworkers with a pair of summaries and asked them to choose the better one in terms of *Grammaticality* (is the summary written in well-formed English?), *Coherence* (is the summary well structured and well organized and does it have a natural ordering of the facts?), and *Conciseness* (does the summary avoid unnecessary repetition including whole sentences, facts or phrases?). We provided example summaries showcasing good and bad output. For this task, we required that the crowdworkers be able to comfortably comprehend NBA/MLB game summaries. We elicited preferences with Best-Worst Scaling (Louviere and Woodworth 1991; Louviere et al., 2015), a method shown to be more reliable than rating scales. The score of a system is computed as the number of times it is rated best minus the number of times it is rated worst (Orme, 2009). The scores

520

range from −100 (absolutely worst) to +100 (absolutely best). We divided the five competing systems into ten pairs of summaries and elicited ratings for 40 summaries (20 per dataset). Each summary pair was rated by 3 raters. This resulted in 40 summaries × 10 system pairs × 3 evaluation criteria × 3 raters, for a total of 3,600 tasks. A total of 206 crowdworkers participated in this task (agreement using Krippendorff's α was 0.47).

As shown in Table 5, on ROTOWIRE, Macro is comparable to Gold, RBF-2020, and ED+CC in terms of *Grammaticality* but significantly better than Templ. In terms of *Coherence*, Macro is comparable to RBF-2020 and ED+CC but significantly better than Templ and significantly worse than Gold. With regard to *Conciseness*, Macro is comparable to Gold, RBF-2020, and ED+CC, and significantly better than Templ. On MLB, Macro is comparable to Gold in terms of *Grammaticality* and significantly better than ED+CC, ENT, and Templ. Macro is comparable to Gold in terms of *Coherence* and significantly better than ED+CC, ENT and Templ. In terms of *Conciseness*, raters found Macro comparable to Gold and Templ and significantly better than ED+CC and ENT. Taken together, our results show that macro planning leads to improvement in data-to-text generation in comparison to other systems for both ROTOWIRE and MLB datasets.

7 Discussion

In this work we presented a plan-and-generate approach for data-to-text generation that consists of a macro planning stage representing high-level document organization in terms of structure and content, followed by a text generation stage. Extensive automatic and human evaluation shows that our approach achieves better results than existing state-of-the-art models and generates summaries which are factual, coherent, and concise.

Our results show that macro planning is more advantageous for generation tasks expected to produce longer texts with multiple discourse units, and could be easily extended to other sports domains such as cricket (Kelly et al., 2009) or American football (Barzilay and Lapata, 2005). Other approaches focusing on micro planning (Puduppully et al., 2019a; Moryossef et al., 2019) might be better tailored for generating shorter texts. There has been a surge of datasets recently focusing on single-paragraph outputs and the task

of content selection such as E2E (Novikova et al., 2017), WebNLG (Gardent et al., 2017), and WikiBio (Lebret et al., 2016; Perez-Belrachini and Lapata, 2018). We note that in our model content selection takes place during macro planning *and* text generation. The results in Table 2 show that Macro achieves the highest CS F-measure on both datasets, indicating that the document as a whole and individual sentences discuss appropriate content.

Throughout our experiments we observed that template-based systems score poorly in terms of CS (but also CO and BLEU). This is primarily due to the inflexibility of the template approach which is limited to the discussion of a fixed number of (high-scoring) players. Yet, human writers (and neural models to a certain extent), synthesize summaries taking into account the particulars of a specific game (where some players might be more important than others even if they scored less) and are able to override global defaults. Template sentences are fluent on their own, but since it is not possible to perform aggregation (Reiter, 1995), the whole summary appears stilted, it lacks coherence and variability, contributing to low BLEU scores. The template baseline is worse for MLB than ROTOWIRE which reflects the greater difficulty to manually create a good template for MLB. Overall, we observe that neural models are more fluent and coherent, being able to learn a better ordering of facts which is in turn reflected in better CO scores.

Despite promising results, there is ample room to improve macro planning, especially in terms of the precision of RG (see Table 2, P% column of RG). We should not underestimate that Macro must handle relatively long inputs (the average input length in the MLB development set is ∼3100 tokens) which are challenging for the attention mechanism. Consider the following output of our model on the MLB dataset: *Ramirez's two-run double off Joe Blanton tied it in the sixth, and Brandon Moss added a two-out RBI single off Alan Embree to give Boston a 3-2 lead*. Here, the name of the pitcher should have been *Joe Blanton* instead of *Alan Embree*. In fact, *Alan Embree* is the pitcher for the following play in the half inning. In this case, attention diffuses over the relatively long MLB macro plan, leading to inaccurate content selection. We could alleviate this problem by adopting a noisy channel decomposition

(Yee et al., 2019; Yu et al., 2020), that is, by learning two different distributions: a conditional model that provides the probability of *translating* a paragraph plan to text and a language model that provides an unconditional estimate of the output (i.e., the whole game summary). However, we leave this to future work.

For RotoWire, the main source of errors is the model's inability to understand numbers. For example, Macro generates the following output: *The Lakers were the superior shooters in this game, going 48 percent from the field and 30 percent from the three-point line, while the Jazz went 47 percent from the floor and 30 percent from beyond the arc*. Here, *30 percent* should have been *24 percent* for the Lakers but the language model expects a higher score for the *three-point line*, and since *24* is low (especially compared to 30 scored by the Jazz), it simply copies *30* scored by the Jazz instead. A mechanism for learning better representations for numbers (Wallace et al., 2019) or executing operations such as argmax or minus (Nie et al., 2018) should help alleviate this problem.

Finally, although our focus so far has been on learning document plans from data, the decoupling of planning from generation allows to flexibly generate output according to specification. For example, we could feed the model with manually constructed macro plans, consequently controlling the information content and structure of the output summary (e.g., for generating short or long texts, or focusing on specific aspects of the game).

Acknowledgments

References

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*.

Regina Barzilay and Mirella Lapata. 2005. Collective content selection for concept-to-text generation. In *Proceedings of Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing*, pages 331–338, Vancouver, British Columbia, Canada. Association for Computational Linguistics. **DOI:** `https://doi.org/10.3115/1220575.1220617`

Steven Bird, Ewan Klein, and Edward Loper. 2009. *Natural Language Processing with Python*, O'Reilly Media.

Thiago Castro Ferreira, Chris van der Lee, Emiel van Miltenburg, and Emiel Krahmer. 2019. Neural data-to-text generation: A comparison between pipeline and end-to-end architectures. 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 552–562, Hong Kong, China. Association for Computational Linguistics. **DOI:** `https://doi.org/10.18653/v1/D19-1052`

Wallace L. Chafe. 1979. The flow of thought and the flow of language. In Talmy Givón, editor, *Syntax and Semantics*, volume 12, pages 159–181, Academic Press Inc. **DOI:** `https://doi.org/10.1163/9789004368897_008`

Yen-Chun Chen and Mohit Bansal. 2018. Fast abstractive summarization with reinforce-selected sentence rewriting. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 675–686, Melbourne, Australia. Association for Computational Linguistics. **DOI:** `https://doi.org/10.18653/v1/P18-1063`

Robert Dale. 1989. Generating referring expressions in a domain of objects and processes.

Pablo Duboue and Kathleen McKeown. 2002. Content planner construction via evolutionary

algorithms and a corpus-based fitness function. In *Proceedings of the International Natural Language Generation Conference*, pages 89–96, Harriman, New York, USA. Association for Computational Linguistics.

Pablo A. Duboue and Kathleen R. McKeown. 2001. Empirically estimating order constraints for content planning in generation. In *Proceedings of the 39th Annual Meeting of the Association for Computational Linguistics*, pages 172–179, Toulouse, France. Association for Computational Linguistics. **DOI:** `https://doi.org/10.3115/1073012.1073035`

John C. Duchi, Elad Hazan, and Yoram Singer. 2011. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research*, 12:2121–2159.

Angela Fan, David Grangier, and Michael Auli. 2018. Controllable abstractive summarization. In *Proceedings of the 2nd Workshop on Neural Machine Translation and Generation*, pages 45–54, Melbourne, Australia. Association for Computational Linguistics.

Claire Gardent, Anastasia Shimorina, Shashi Narayan, and Laura Perez-Beltrachini. 2017. Creating training corpora for NLG microplanners. 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. **DOI:** `https://doi.org/10.18653/v1/P17-1017`

Albert Gatt and Emiel Krahmer. 2018. Survey of the state of the art in natural language generation: Core tasks, applications and evaluation. *J. Artif. Intell. Res.*, 61:65–170. **DOI:** `https://doi.org/10.1613/jair.5477`

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. **DOI:** `https://doi.org/10.18653/v1/D19-1310`

Jiatao Gu, Zhengdong Lu, Hang Li, and Victor O. K. Li. 2016. Incorporating copying mechanism in sequence-to-sequence learning. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1631–1640, Berlin, Germany. Association for Computational Linguistics.

Caglar Gulcehre, Sungjin Ahn, Ramesh Nallapati, Bowen Zhou, and Yoshua Bengio. 2016. Pointing the unknown words. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 140–149, Berlin, Germany. Association for Computational Linguistics. **DOI:** `https://doi.org/10.18653/v1/P16-1014`

M. A. K. Halliday and Ruqaiya Hasan. 1976. *Cohesion in English*, London. Longman. **DOI:** `https://doi.org/10.1162/neco.1997.9.8.1735`

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural Computation*, 9:1735–1780.

Eduard H. Hovy. 1993. Automated discourse generation using discourse structure relations. *Artificial Intelligence*, 63(1-2):341–385. **DOI:** `https://doi.org/10.1016/0004-3702(93)90021-3`

Hayate Iso, Yui Uehara, Tatsuya Ishigaki, Hiroshi Noji, Eiji Aramaki, Ichiro Kobayashi, Yusuke Miyao, Naoaki Okazaki, and Hiroya Takamura. 2019. Learning to select, track, and generate for data-to-text. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 2102–2113, Florence, Italy. Association for Computational Linguistics. **DOI:** `https://doi.org/10.18653/v1/P19-1202`

Min-Yen Kan and Kathleen R. McKeown. 2002. Corpus-trained text generation for summarization. In *Proceedings of the International*

Natural Language Generation Conference, pages 1–8, Harriman, New York, USA. Association for Computational Linguistics.

Colin Kelly, Ann Copestake, and Nikiforos Karamanis. 2009. Investigating content selection for language generation using machine learning. In *Proceedings of the 12th European Workshop on Natural Language Generation (ENLG 2009)*, pages 130–137, Athens, Greece. Association for Computational Linguistics. **DOI:** `https://doi.org/10.3115/161 0195.1610218`

Yuta Kikuchi, Graham Neubig, Ryohei Sasano, Hiroya Takamura, and Manabu Okumura. 2016. Controlling output length in neural encoder-decoders. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1328–1338, Austin, Texas. Association for Computational Linguistics. **DOI:** `https://doi.org /10.18653/v1/D16-1140`

Guillaume Klein, Yoon Kim, Yuntian Deng, Jean Senellart, and Alexander Rush. 2017. OpenNMT: Open-source toolkit for neural machine translation. In *Proceedings of ACL 2017, System Demonstrations*, pages 67–72, Vancouver, Canada. Association for Computational Linguistics. **DOI:** `https://doi.org /10.18653/v1/P17-4012`

Ioannis Konstas and Mirella Lapata. 2013. Inducing document plans for concept-to-text generation. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1503–1514, Seattle, Washington, USA. Association for Computational Linguistics.

Karen Kukich. 1983. Design of a knowledge-based report generator. In *21st Annual Meeting of the Association for Computational Linguistics*. **DOI:** `https://doi.org /10.3115/981311.981340`

Anirban Laha, Parag Jain, Abhijit Mishra, and Karthik Sankaranarayanan. 2020. Scalable micro-planned generation of discourse from structured data. *Computational Linguistics*, 45(4):737–763. **DOI:** `https://doi .org/10.1162/coli_a_00363`

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. **DOI:** `https://doi.org /10.18653/v1/D16-1128`

R. E. Longacre. 1979. The paragraph as a grammatical unit. In Talmy Givón, editor, *Syntax and Semantics*, volume 12, Academic Press Inc., pages 115–133.

Jordan J. Louviere, Terry N. Flynn, and A. A. J. Marley. 2015. *Best-Worst Scaling: Theory, Methods and Applications*, Cambridge University Press. **DOI:** `https://doi.org /10.1017/CBO9781107337855`

Jordan J. Louviere and George G. Woodworth. 1991. Best-worst scaling: A model for the largest difference judgments. University of Alberta: Working Paper.

Thang Luong, Hieu Pham, and Christopher D. Manning. 2015. Effective approaches to attention-based neural machine translation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1412–1421, Lisbon, Portugal. Association for Computational Linguistics. **DOI:** `https://doi.org/10.18653/v1/D15 -1166`

Kathleen R. McKeown. 1992. *Text Generation*. Studies in Natural Language Processing, Cambridge University Press.

Kathleen R. McKeown, Desmond A. Jordan, Shimei Pan, James Shaw, and Barry A. Allen. 1997. Language generation for multimedia healthcare briefings. In *Fifth Conference on Applied Natural Language Processing*, pages 277–282, Washington, DC, USA. Association for Computational Linguistics. **DOI:** `https://doi.org/10.3115/974557 .974598`

Hongyuan Mei, Mohit Bansal, and Matthew R. Walter. 2016. What to talk about and how? Selective generation using LSTMs with coarse-to-fine alignment. In *Proceedings of the*

2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, pages 720–730, San Diego, California. Association for Computational Linguistics.

Amit Moryossef, Yoav Goldberg, and Ido Dagan. 2019. Step-by-step: Separating planning from realization in neural data-to-text generation. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 2267–2277, Minneapolis, Minnesota. Association for Computational Linguistics.

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. **DOI:** https://doi.org/10.18653/v1/D18 -1422

Jekaterina Novikova, Ondřej Dušek, and Verena Rieser. 2017. The E2E dataset: New challenges for end-to-end generation. In *Proceedings of the 18th Annual SIGdial Meeting on Discourse and Dialogue*, pages 201–206, Saarbrücken, Germany. Association for Computational Linguistics. **DOI:** https://doi.org/10.18653/v1/W17 -5525

Bryan Orme. 2009. Maxdiff analysis: Simple counting, individual-level logit, and HB. *Sawtooth Software*.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. BLEU: a method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pages 311–318, Philadelphia, Pennsylvania, USA. Association for Computational Linguistics. **DOI:** https://doi.org/10.3115/1073083 .1073135

Romain Paulus, Caiming Xiong, and Richard Socher. 2018. A deep reinforced model for abstractive summarization. In *International Conference on Learning Representations.*

Laura Perez-Beltrachini and Mirella Lapata. 2018. Bootstrapping generators from noisy data. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 1516–1527, New Orleans, Louisiana. Association for Computational Linguistics. **DOI:** https://doi.org/10.18653/v1 /N18-1137

Ratish Puduppully, Li Dong, and Mirella Lapata. 2019a. Data-to-text generation with content selection and planning. In *Proceedings of the 33rd AAAI Conference on Artificial Intelligence*. Honolulu, Hawaii. **DOI:** https:// doi.org/10.1609/aaai.v33i01.330 16908

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. **DOI:** https://doi.org/10.18653/v1 /P19-1195

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9. **DOI:** https://doi.org/10.18653/v1/P19 -1195

Clément Rebuffel, Laure Soulier, Geoffrey Scoutheeten, and Patrick Gallinari. 2020. A hierarchical model for data-to-text generation. In *European Conference on Information Retrieval*, pages 65–80. Springer. **DOI:** https://doi.org/10.1007/978-3-030 -45439-5_5, **PMCID:** PMC7148215

Ehud Reiter. 1995. NLG vs. templates. *CoRR*, cmp-lg/9504013v1. **DOI:** https://doi.org /10.1017/S1351324997001502

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

525

Ehud Reiter and Robert Dale. 2000. *Building Natural Language Generation Systems*. Studies in Natural Language Processing, Cambridge University Press. **DOI:** `https://doi.org/10.1017/CBO9780511519857`

Fahimeh Saleh, Alexandre Berard, Ioan Calapodescu, and Laurent Besacier. 2019. Naver Labs Europe's systems for the document-level generation and translation task at WNGT 2019. In *Proceedings of the 3rd Workshop on Neural Generation and Translation*, pages 273–279, Hong Kong. Association for Computational Linguistics. **DOI:** `https://doi.org/10.18653/v1/D19-5631`

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Neural machine translation of rare words with subword units. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1715–1725, Berlin, Germany. Association for Computational Linguistics. **DOI:** `https://doi.org/10.18653/v1/P16-1162`

Zhihong Shao, Minlie Huang, Jiangtao Wen, Wenfei Xu, and Xiaoyan Zhu. 2019. Long and diverse text generation with planning-based hierarchical variational model. 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 3257–3268, Hong Kong, China. Association for Computational Linguistics. **DOI:** `https://doi.org/10.18653/v1/D19-1321`

Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. 2014. Sequence to sequence learning with neural networks. In *Advances in Neural Information Processing Systems*, volume 27, pages 3104–3112. Curran Associates, Inc.

Shunsuke Takeno, Masaaki Nagata, and Kazuhide Yamamoto. 2017. Controlling target features in neural machine translation via prefix constraints. In *Proceedings of the 4th Workshop on Asian Translation (WAT 2017)*, pages 55–63, Taipei, Taiwan. Asian Federation of Natural Language Processing.

Ran Tian, Shashi Narayan, Thibault Sellam, and Ankur P. Parikh. 2019. Sticking to the facts: Confident decoding for faithful data-to-text generation. *CoRR*, abs/1910.08684v2.

Chris van der Lee, Albert Gatt, Emiel van Miltenburg, Sander Wubben, and Emiel Krahmer. 2019. Best practices for the human evaluation of automatically generated text. In *Proceedings of the 12th International Conference on Natural Language Generation*, pages 355–368, Tokyo, Japan. Association for Computational Linguistics. **DOI:** `https://doi.org/10.18653/v1/W19-8643`

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, Curran Associates, Inc., pages 5998–6008.

Oriol Vinyals, Meire Fortunato, and Navdeep Jaitly. 2015. Pointer networks. In C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett, editors, *Advances in Neural Information Processing Systems 28*, pages 2692–2700, Curran Associates, Inc.

Eric Wallace, Yizhong Wang, Sujian Li, Sameer Singh, and Matt Gardner. 2019. Do NLP models know numbers? Probing numeracy in embeddings. 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 5307–5315, Hong Kong, China. Association for Computational Linguistics. **DOI:** `https://doi.org/10.18653/v1/D19-1534`

Ronald J. Williams and Jing Peng. 1990. An efficient gradient-based algorithm for on-line training of recurrent network trajectories. *Neural Computation*, 2(4):490–501. **DOI:** `https://doi.org/10.1162/neco.1990.2.4.490`

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. **DOI:** `https://doi.org/10.18653/v1/D17-1239`

Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V. Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, Jeff Klingner, Apurva Shah, Melvin Johnson, Xiaobing Liu, Lukasz Kaiser, Stephan Gouws, Yoshikiyo Kato, Taku Kudo, Hideto Kazawa, Keith Stevens, George Kurian, Nishant Patil, Wei Wang, Cliff Young, Jason Smith, Jason Riesa, Alex Rudnick, Oriol Vinyals, Greg Corrado, Macduff Hughes, and Jeffrey Dean. 2016. Google's neural machine translation system: Bridging the gap between human and machine translation. *CoRR*, abs/1609.08144v2.

Zichao Yang, Diyi Yang, Chris Dyer, Xiaodong He, Alex Smola, and Eduard Hovy. 2016. Hierarchical attention networks for document classification. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1480–1489, San Diego, California. Association for Computational Linguistics. **DOI:** `https://doi.org/10.18653/v1/N16-1174`

Kyra Yee, Yann Dauphin, and Michael Auli. 2019. Simple and effective noisy channel modeling for neural machine translation. 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 5696–5701, Hong Kong, China. Association for Computational Linguistics. **DOI:** `https://doi.org/10.18653/v1/D19-1571`

Lei Yu, Laurent Sartran, Wojciech Stokowiec, Wang Ling, Lingpeng Kong, Phil Blunsom, and Chris Dyer. 2020. Better document-level machine translation with Bayes' rule. *Transactions of the Association for Computational Linguistics*, 8:346–360. **DOI:** `https://doi.org/10.1162/tacl_a_00319`

Wlodek Zadrozny and Karen Jensen. 1991. Semantics of paragraphs. *Computational Linguistics*, 17(2):171–210.

527