Selecting, Planning, and Rewriting: A Modular Approach for Data-to-Document Generation and Translation

Lesly Miculicich[*†] **Marc Marone** [*‡]
Hany Hassan[‡]

[†] Idiap Research Institute, Ecole Polytechnique Federale de Lausanne (EPFL), Switzerland
lmiculicich@idiap.ch

[‡]Microsoft, 1 Microsoft Way, Redmond, WA 98121, USA
{v-mamaro,hanyh}microsoft.com

Abstract

In this paper, we report our system submissions to all 6 tracks of the WNGT 2019 shared task on Document-Level Generation and Translation. The objective is to generate a textual document from either structured data: *generation task*, or a document in a different language: *translation task*. For the translation task, we focused on adapting a large scale system trained on WMT data by fine tuning it on the RotoWire data. For the generation task, we participated with two systems based on a selection and planning model followed by (a) a simple language model generation, and (b) a GPT-2 pre-trained language model approach. The selection and planning module chooses a subset of table records in order, and the language models produce text given such a subset.

1 Introduction

Data-to-text generation focuses on generating natural text from structured inputs such as table records. Traditional data-to-text systems used a pipelined approach for data selection followed by planning and text generation. Recently, End-to-End neural generation systems have been proposed such as (Wiseman et al., 2017). While such systems generate more fluent output, they are not faithful to the facts in the structured data.

One of the difficulties of the generation task is that the models have to learn two different aspects: "what to say" (i.e. selecting relevant information) and "how to say it" (i.e. generating the actual text based on these facts). We believe that available data-sets are too small for allowing current neural network models, based on encoder-decoder architectures, to capture the complexity of the problem. Recently, (Puduppully et al., 2019) proposed an end-to-end system that explicitly models a content selection and planning module separated from the text generation, showing improvements with respect to previous end-to-end systems (Wiseman et al., 2017). We adopt this approach and divide our model into two parts noted as: content selection and planning, and text generation. This division helps the system to produce more coherent document structure. One drawback of this approach is the limitation of the language model generation coverage. We tackle this limitation by adopting pre-trained language models such as OpenAI's GPT-2 (Radford et al., 2019). Following the shared task guidelines (Hayashi et al., 2019), we evaluate our models using an information extraction (IE) system.

2 Translation Tasks

We submitted translation results for both directions: English-to-German and German-to-English. Our models are based on the transformer architecture trained with multi-agent dual learning (MADL) (Xia et al., 2019). This system uses the *transformer_big* configuration (modelsize 1024, filter size 4096, 6 blocks for each encoder and decoder) from (Vaswani et al., 2017), using dropout of 0.2. It is trained with 3 primary translations models and 3 dual translation models (for details refer to (Xia et al., 2019)). The base models were trained with 20M filtered sentence-pairs from WMT 2019 shared translation task (Bojar et al., 2019), plus 120M English, and 120M German monolingual sentences from *Newscrawl* (Bojar et al., 2019). The vocabulary is shared and composed of word segments obtained by applying the BPE algorithm with 35K merge operations.

We fine-tuned the base models with 3K sentence-pairs of the Rotowire English-German parallel data. We use batches of 4096 tokens and optimize with different learning rates. The best result was obtained with learning rate of 10^{-5} for both directions. Additionally, for the German-to-English translation task, we back-translate the

[*]Equal contribution. Work done while interning at Microsoft

monolingual Rotowire English corpus. For this purpose, the documents were split into sentences[1] to obtain 45K English sentences in total, then we use the MADL model which was fine-tuned with parallel Rotowire data to obtain their respective German translations. Finally, we fine-tune the MADL system again using the concatenation of original parallel sentences and the back-translated corpus. Since we do not have an in-domain monolingual German corpus, we ensemble 3 replicas of the fine-tuned MADL models trained with 10, 20, and 30% dropout for the English-German task. Additionally, we back-translate 1M monolingual sentences from *Newscrawl* which were selected based on their similarity to Rotowire data following (Moore and Lewis, 2010). However, this did not lead to any further improvements in the systems.

All translation systems used text summaries only. We did not use the additional data tables in the submitted results for the MT + NLG track. In our experiments, we find that adding the structured data did not lead to improvements over the baseline systems.

2.1 Results

Table 1 shows the results of our systems for both directions measured with sacre-BLEU. Fine tuning with Rotowire parallel data brings an improvement of 7.9 BLEU points for English-to-German and 9.3 for German-to-English in the test set. Further improvement of 1.9 BLEU points is obtained with back-translation of monolingual Rotowire data for the latter direction. The dropout ensemble adds a very small gain of 0.2 BLEU. We found that selected data from *Newscrawl* does not add any significant improvement.

We also evaluate our German-to-English system with the content oriented metrics provided by the organizers of the shared-task. Table 2 shows the values for development and test sets. We show the results measured with the ground-truth translation for comparison. The content generation (RG) of the best system reaches two percentage points higher than the ground-truth. The translation model produces fewer referring expressions, and morphological variations than the ground-truth to refer to entities, which makes it easier for the information extraction tools to recognize them. The Content selection (CS) reaches high precision (92%) and recall (93%), and the content order (CO) score is 89. Further manual analysis indicates that the main issues are the translation of textual numbers, and the morphological variation of entities.

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

	EN → DE	DE → EN
MADL	39.99	48.71
+ RW parallel	47.90	57.99
+ RW monolingual ⋆	–	59.94
+ Ensemble ⋆	48.09	–

Table 1: Machine translation results measured with sacre-BLEU and task-specific tokenization[1]. ⋆ denotes a late entry, not in the official evaluation.

3 Generation Task

One of the difficulties of the data-to-document generation task, as formulated by (Wiseman et al., 2017), is that the models have to learn to select a relatively small portion of table records and their order for generating text. We argue that the size of available data-sets for training (i.e. Rotowire with 3.4K samples) is too small for allowing the network to capture the complexity of the task. In consequence, following (Puduppully et al., 2019; Moryossef et al., 2019), we divide the work in two parts : (a) content selection and planning, and (b) text generation. The idea is to introduce a direct signal to the system i.e. adding a loss function that guides an orderly selection of records, alleviating the work of the text generator.

Our system is based on (Puduppully et al., 2019) with two main differences. First, we use a transformer network for encoding each record in relation to other records, instead of a more complex gated approach as previous work. Second, we share the vocabularies of record *values* and summary text, thus the final estimated distribution for prediction over the whole vocabulary is summed instead of concatenated. Figure 1 shows the architecture of the model. In the following, we describe each component in detail:

3.1 Content selection and Planning

Given a set of records $r = r_1, r_2, ..., r_M$, the objective is to select a subset of them in the correct order $\tilde{r} \in r$. We use an encoder-decoder architecture to model this problem. Similar to (Wiseman et al., 2017), we create embeddings of each feature record (e.g. value, key, etc.) and concatenate

	Dev			Test		
	RG (P%/#)	CS (P%/R%)	CO	RG (P%/#)	CS (P%/R%)	CO
Ground truth	92.0 (23.1)	100 / 100	100	92.3 (22.6)	100 / 100	100
MADL	90.4 (19.2)	90.6 / 78.8	74.4	91.7 (19.3)	89.6 / 77.1	75.1
+ RW parallel + RW mono.	94.4 (24.2)	93.6 / 93.8	89.9	94.1 (23.3)	92.6 / 93.1	89.1

Table 2: Content evaluation of the German-to-English translation models on test and dev-sets from parallel Rotowire using the IE models of (Puduppully et al., 2019). RG:Content generation, CS: Content selection, CO: Content order.

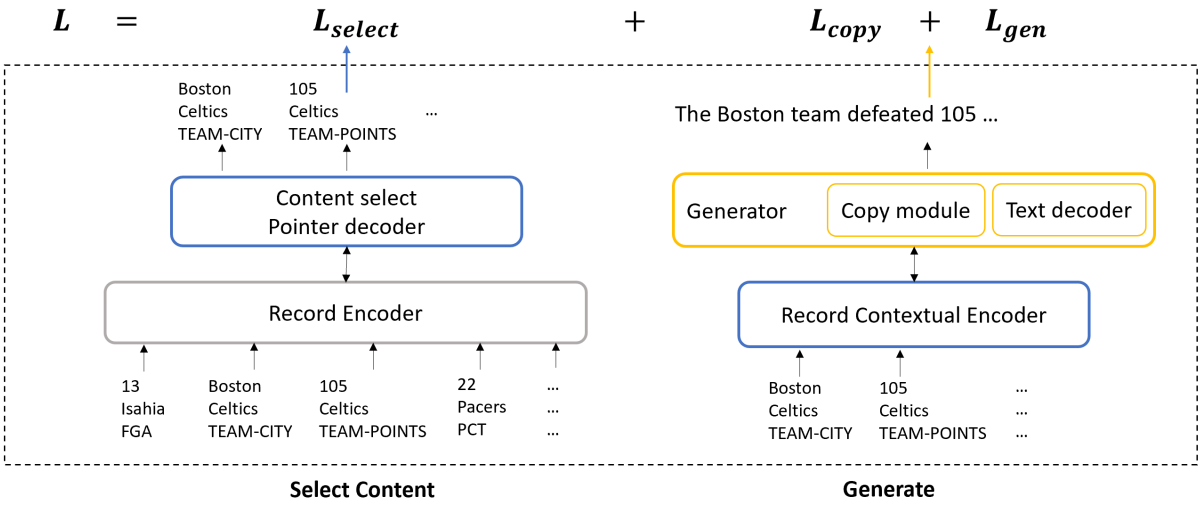

Figure 1: Content selection and generation

them. Then, we encode them using a transformer network. The transformer self-attention allows the network to capture the relationships among records and estimate their relative salience. However, we do not use positional embeddings as the records do not have a sequential nature. The decoder is a recurrent pointer network which predicts one record $\tilde{r}_t$ at each time-step based on previous predictions as follows:

$$\tilde{r}_t = softmax(\hat{h}_1, ..., \hat{h}_M) \quad (1)$$

$$\hat{h}_i = f(h_i, s_t) \quad (2)$$

$$s_t = g(s_{t-1}, \tilde{r}_{t-1}) \quad (3)$$

where f is a non-linear function, g is an auto-regressive network (i.e. LSTM, transformer) and h_i is the encoded state of the record r_i using the transformer. When using LSTM, the initial state is the average of the encoder output h. We optimize this sub-network with a cross-entropy loss L_{select}, and the ground truth targets are extracted following (Puduppully et al., 2019).

3.2 Text Generation

The text generator is also an encoder-decoder network. The encoder is a bi-directional LSTM or transformer that receives a set of record as input. During training the input are the ground truth targets $\tilde{r}_{gold}$, and during decoding the predictions of the *content selection and planning* $\tilde{r}$.

The decoder has two parts: a *text decoder* and a *copy module* that uses a copy mechanism to directly predict encoded records. We share the vocabulary of the copy-decoder and the record feature *value* of the encoder so the probability distributions of generating and copying are summed for each shared word, similar to (See et al., 2017). The embeddings of all record features are shared for both content selection, and text generation. The optimization is performed with a cross-entropy loss L_{gen} for the generation, and a binary cross-entropy loss L_{copy} for the copy module.

3.3 Joint regularized training (End-to-end)

We train the *content selector* and *text generator* in a joint manner by adding their losses $L = L_{select} + L_{gen} + L_{copy}$. This can be seen as a regularization process where the loss of one network regularizes the other. We observe that the performance of the separately trained networks are worse than the jointly trained ones. The input

	P	R	F1	DL
Single Baseline	41	68	51	0.76
Single CSP	43	67	52	0.75
Joint Baseline + TG	45	62	52	0.75
Joint CSP + TG	46	71	56	0.70

Table 3: Evaluation of Content Selection and Planning (CSP) module, with and without joint training with the Text Generator (TG). Baseline: (Puduppully et al., 2019). P: precision, R: recall, DL: Damerau-Levenshtein distance.

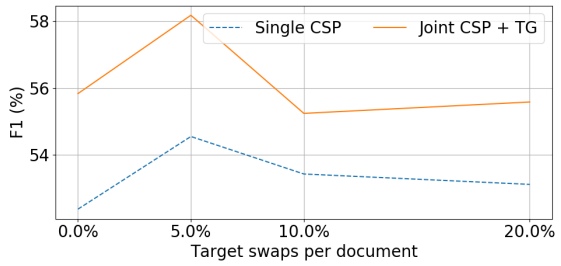

Figure 2: Augmenting data by swapping the target values of each document at different percentage rates. In each case, we doubled the training set.

for the text generator is ground truth during training. At decoding time, we perform 2 consecutive beam search of 5 beams, the first one for the content selection, and the second for generating text. We tuned the architecture using the development set. We evaluated different configurations of transformers and LSTMs, from 1 to 2 layers, with dimensions of 252 and 512, and dropout from 0.1 to 0.3. The best results are obtained using LSTMs decoders with 512 size dimension for all hidden layers and embeddings, each encoder and decoder has 2 layers, and we use dropout of 0.3. We also used data augmentations by swapping values in of the target at a range of 10, 20, and 30 percent of the values in each sample. Finally, we train the network with batches of 5 document samples, and updated the parameters each 20 batches. We use Adam optimizer at an initial learning rate of 10^{-3}.

3.4 Results

We test the *content selection and planning* module by comparing the output subset of records with the ground-truth records (i.e. the records extracted from the summary text with the IE tools). We use F1 and Damerau-Levenshtein (DL), the later to evaluate the correct order of records. The workshop metrics are not used here because the independently trained models do not output text. Results in Table 3 show that our model outperforms the baseline (Puduppully et al., 2019), and the joint training helps to further improve over the single models. Figure 2 shows the F1 scores while augmenting data by varying the percentage of swaps in the target training set. Adding samples with 5% of random swaps in each sample document helps both single and jointly trained models. Finally, Table 4 shows the evaluation results of the final joint system with the workshop metrics.

During a qualitative evaluation, we noticed that the *content selection and planning* module learns to output the most common sequence of records in the training data. In general, the sequence of records depends on the style of the commentator (e.g. describing first match results and then important player's scores). Our system mismatches less common styles, which affects the scoring of testing and development that contain different distribution of record sequences.

4 Generation with Pretrained LM

We experiment with using pretrained language models to enhance coverage and fluency of the text generation, since the amount of training data available for the generation the task is relatively small. In particular, we use a pretrained PyTorch implementation[2] of the GPT-2 (Radford et al., 2019) language model. The original GPT-2 description showed that this large scale language model additionally learned to complete other tasks such as summarization and translation. For example, the language model can be used for summarization, by providing an input document concatenated with the string `TL;DR:` and the output is the generated summary. Inspired by these results, we propose our summary *rewrite model*. Our model is a two phases approach: the first is the content selection model proposed in 3.1, the second is a GPT2-based generation module. Based on the output of the content selection module, our model provides a rough summary as input to GPT-2 model which generates the desired summary.

The baseline results in (Wiseman et al., 2017) show that simple templates are very effective at conveying information from the data tables. We use similar templates to generate a rough summary that is used as input in our *rewrite model*. The model takes input

[2] https://github.com/huggingface/pytorch-transformers

	Data to EN				Data to DE			
	BLEU	RG	CS (P/R)	CO	BLEU	RG	CS (P/R)	CO
End-to-end	15.03	93.38	32.34 / 58.04	18.52	11.66	80.30	27.89 / 48.96	16.43
GPT-50	15.17	94.35	33.84 / 53.82	19.26	11.84	82.79	34.23 / 42.32	16.93
GPT-90	13.03	88.70	32.81 / 50.64	17.34	10.43	75.05	30.97 / 41.48	16.27

Table 4: Generation results of our submitted systems as reported by the shared task organizers (Hayashi et al., 2019). RG: Relation Generation precision, CS: Content Selection (precision/recall), CO: Content Ordering.

of the form `template summary <R> gold summary`, which is used to fine-tune the pre-trained GPT-2 model. The templates consist of simple sentences involving a single record from the dataset, such as the number of points scored by a player. At training time we generate templates from the ground truth records following (Puduppully et al., 2019). At test time, we use the content selection module to select appropriate records. This effectively replaces the original generator network with the GPT-2 model, using text as an intermediate encoding. See Appendix sections A.1 and A.2 for a full example.

4.1 Decoding

Recently, (Holtzman et al., 2019) suggested that top-k token sampling (as used in the original GPT-2 results) is not ideal for generating realistic text from likelihood trained language models. They instead propose *Nucleus Sampling* or top-p sampling, which samples tokens from the top p portion of the output distribution. We experiment with several values of p and find that this provides an effective way to control generation quality. Our submitted models (GPT-50 and GPT-90) sample from the top 50% and 90% of the output distribution when decoding.

4.2 Results

We find that the template rewriting approach is competitive with the end-to-end trained models in terms of content metrics (Table 4), and subjectively appears to create natural sounding generations.

For lower values of p in top-p sampling, we find that the model remains more true to the templates, tending to create short summaries that do not deviate much from the input facts. For larger values of p, where decoding is allowed to sample from more of the distribution, the output tends be longer but may deviate from the facts. We also note that when regenerating summaries for high values of p (with a different random seed), there are significant changes to the text but not to the facts reflected in the summary. See Appendix sections A.4 and A.5 for examples of generations at various p values. For both settings we observe occasional mistakes such as repetitions, suggesting that our values for p should have been tuned more carefully.

For the German generation track, we apply our model described in 2 to the English generations, since we did not have a GPT-2 scale language model for German.

5 Discussion and Conclusion

For the translation task we experimented with a simple fine tuning approach for a large scale system trained on general domain data. It proved very effective to fine tune a pre-trained system using RotoWire data. Our analysis indicates that the remaining problems are more related to number formatting which is a more generic issue for NMT systems and not a domain specific problem.

The generation task proved to be more challenging. Mainly generating faithful, accurate and fluent summaries can be a quite challenging task given the discrepancies between the provided data and the desired summaries. Our analysis indicates that there is a mismatch between the gold selection plan and the system output. The system outputs the most common sequence of facts whereas the gold presents more variety of fact sequences due to different writing styles. This issue should be further studied in future.

Utilizing large scale pre-trained LMs (such as GPT-2) is a very promising direction, since it decouples the dependency of selection and generation resources. Our current approach of feeding the template-based input to GPT2 is quite simple and efficient. We would like to investigate more principled methods of doing this in the future.

6 Acknowledgment

We would like to thank Tao Qin and his team at MSRA for providing the MADL translation baseline systems.

References

Ondřej Bojar, Rajen Chatterjee, Christian Federmann, Mark Fishel, Yvette Graham, Barry Haddow, Matthias Huck, Antonio Jimeno Yepes, Philipp Koehn, André Martins, Christof Monz, Matteo Negri, Aurélie Névéol, Mariana Neves, Matt Post, Marco Turchi, and Karin Verspoor. 2019. Proceedings of the fourth conference on machine translation (volume 2: Shared task papers, day 1). Florence, Italy. Association for Computational Linguistics.

Hiroaki Hayashi, Yusuke Oda, Alexandra Birch, Ioannis Constas, Andrew Finch, Minh-Thang Luong, Graham Neubig, and Katsuhito Sudoh. 2019. Findings of the third workshop on neural generation and translation. In *Proceedings of the Third Workshop on Neural Generation and Translation*.

Ari Holtzman, Jan Buys, Maxwell Forbes, and Yejin Choi. 2019. The curious case of neural text degeneration. *arXiv preprint arXiv:1904.09751*.

Robert C. Moore and William Lewis. 2010. Intelligent selection of language model training data. In *Proceedings of the ACL 2010 Conference Short Papers*, pages 220–224, Uppsala, Sweden. 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.

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

Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners.

Abigail See, Peter J. Liu, and Christopher D. Manning. 2017. Get to the point: Summarization with pointer-generator networks. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1073–1083, Vancouver, Canada. Association for Computational Linguistics.

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 *Advances in neural information processing systems*, pages 5998–6008.

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.

Yingce Xia, Xu Tan, Fei Tian, Fei Gao, Di He, Weicong Chen, Yang Fan, Linyuan Gong, Yichong Leng, Renqian Luo, Yiren Wang, Lijun Wu, Jinhua Zhu, Tao Qin, and Tie-Yan Liu. 2019. Microsoft research asia's systems for WMT19. In *Proceedings of the Fourth Conference on Machine Translation (Volume 2: Shared Task Papers, Day 1)*, Florence, Italy. Association for Computational Linguistics.

A Rewrite Model

Below are example generations from our models, with some highlighted comparisons. "..." indicates text omitted for brevity.

Each configuration correctly introduces the teams. However, note that the stadium differs. The End-to-end model claims that the game took place at the United Center. The GPT models claim that the game took place at the Air Canada Cent(er|re), the correct stadium for the home team (the Toronto Raptors). In fact, the GPT model uses what may be a more correct spelling (centre).

All models report player statistics (some example sentences are **bolded**) and game statistics. There are some occasional mistakes. Some are logical (the Raptors won), and others seems to be related to decoding.

A.1 Gold Template Input

The Raptors are from Toronto. The Raptors were HOME. The Raptors scored 122 points. The 76ers are from Philadelphia. The 76ers were AWAY. The 76ers scored 95 points. ... Robert Covington scored 20 points. Robert Covington had 5 rebounds. Robert Covington had 2 assists. Robert Covington had 2 steals. Robert Covington made 7 shots. Robert Covington had 11 attempts. Jahlil Okafor scored 15 points. ...

A.2 Gold Summary

The host Toronto Raptors defeated the Philadelphia 76ers, 122 - 95, at Air Canada Center on Monday. ... The Raptors were lights - out shooting, as they went 55 percent from the field and 68 percent from three - point range. They also held the Sixers to just 42 percent from the field and dominated the defensive rebounding, 34 - 26. Fastbreak points was a huge difference as well, with Toronto winning that battle, 21 - 6. ... **Robert Covington had one of his best games of the season though, tallying 20 points, five rebounds, two assists and two steals on 7 - of - 11 shooting. Jahlil Okafor got the start for Embiid and finished with 15 points and five rebounds.** ...

A.3 End-to-End Summary

The Toronto Raptors defeated the visiting Philadelphia 76ers 122 - 95 , at United Center on Saturday evening. The Raptors (11 - 6) came in to Friday's contest having lost four of their last six games, and have now lost five of their last seven games. **They were led by Terrence Ross, who scored a game - high 22 points on 8 - of - 11 shooting from the field and 3 - of - 5 from three - point range. Kyle Lowry followed up with 24 points, eight assists and four rebounds, in 32 minutes. Jonas Valanciunas recorded a double - double with 12 points and 11 rebounds, in 22 minutes off the bench.** The Raptors shot 55 percent from the field and 68 percent from long range , while the 76ers shot 55 percent from the field and 68 percent from behind the arc . **Robert Covington was the high - point man for the 76ers, with 20 points , including 6 - of - 9 from three - point range.** ... The Sixers will look to bounce back when they visit the New York Knicks on Monday.

A.4 Top-0.5 Sampling (GPT-50)

The Toronto Raptors defeated the Philadelphia 76ers, 122 - 95, at Air Canada Centre on Saturday. The Raptors (11 - 6) were able to prevail despite being outshot, out rebounded and out - rebounded by the Sixers (7 - 15) in the contest. ... **The Raptors were led by Terrence Ross, who posted a game - high 22 points on 8 - of - 11 shooting, including 3 - of - 5 from long range. Kyle Lowry followed up with 24 points, eight assists and four rebounds, in 32 minutes. Jonas Valanciunas chipped in 12 points and 11 rebounds, in 22 minutes off the bench.** The Raptors shot a respectable 55 percent from the field and 68 percent from long range, but were out - rebounded by the Sixers by a 52 - 40 margin. Philadelphia had five players in double figures, **led by Robert Covington, who scored 20 points on 6 - of - 9 shooting from**

behind the arc. The Sixers will look to bounce back, as they travel to Boston to take on the Celtics on Monday. ... The Raptors will look to extend their winning streak to four on Monday against the Orlando Magic.

A.5 Top-0.9 Sampling (GPT-90)

We show summaries from two different seeds for the least restrictive sampling setting, $p = 0.9$. Some details change but most of the content supported by the data remains the same. Aggregated across the entire set, the information metrics remain nearly identical. This less restricted model more frequently outputs hallucinations and surrounding details not supported by the tabular data.

Sample One:

The Toronto Raptors defeated the Philadelphia 76ers, 122 - 95, at Air Canada Centre on Saturday. ... Lowry added 24 points, eight assists and four rebounds in 32 minutes. Jonas Valanciunas scored 12 points and grabbed 11 rebounds in 22 minutes as a starter. ...

Sample Two:

The Toronto Raptors defeated the Philadelphia 76ers, 122 - 95, in overtime at Air Canada Centre on Tuesday. ... After a rather lopsided first quarter, the Raptors came out flat for the first 40 - plus minutes of this game. Philadelphia ended up hanging around for the final four minutes of regulation, as Toronto took the lead for good with 16 seconds remaining in regulation. ... In overtime, the Raptors took a 13 - point lead into the locker room thanks to a clutch 30 - foot field goal from Terrence Ross with eight seconds remaining in regulation. ... Kyle Lowry added 24 points, eight assists and four rebounds in 32 minutes, while Jonas Valanciunas added a 12 - point, 11 - rebound double - double in just 22 minutes. ...