Improving Data-to-Text Generation via Preserving High-Frequency Phrases and Fact-Checking

Ethan Joseph*
Rensselaer Polytechnic Institute

Julian Lioanag**
Rensselaer Polytechnic Institute

Mei Si†
Rensselaer Polytechnic Institute

Transforming numerical data into natural language descriptions (data-to-text) requires presenting the data in the correct context, supplementing plausible details, and creating an overall coherent and non-conflicting narrative. In this work, we propose a generate-extract-correct pipeline for the task. We use transfer learning with an auxiliary task of keeping high-frequency word sequences from the training data for text generation. We then apply information extraction to the generated text to check its accuracy, followed by correction, and thus ensure the coherence of the generated narrative. We demonstrate the effectiveness of this approach with both objective and subjective evaluations. Using an empirical evaluation, we show that people rated our system's outputs similarly to human-written text regarding its coherence, conciseness, and grammar.

Keywords: language generation; data-to-text; transfer learning

1. Introduction

In recent years, there has been an increasing interest in automatically generating text descriptions or dialogue from structured data (Puduppully, Dong, and Lapata 2019; Wiseman, Shieber, and Rush 2017; Rebuffel et al. 2020; Kale 2020). Data-to-text, broadly speaking, refers to tasks where a system is provided with data in a machine-readable format, e.g., RDF or tabular data, and needs to produce human-readable text based on the data. Because data-to-text techniques can enable machines to communicate with people in a natural, narrative way, they have enormous potential for real-world applications, especially with the fast development of semantic web, knowledge graph, and automated data analysis tools in recent years.

Given its primary function of communicating data with people, we infer three desiderata for data-to-text generation techniques. First of all, the data-to-text generation needs to ensure it conveys accurate information. This requires providing correct data and avoiding confusion in the generated text. Confusion can come from multiple sources, including redundancy and inconsistency in information, violating common sense, and incorrect grammar. Since people read sequentially, data-to-text generation can be viewed as an iterative grounding process, where the beginning part should ground the latter part of the generated text. Secondly, the generated narrative needs to be relatively concise while not hurting readability. Being concise can help

* Dept. of Computer Science - Rensselaer Polytechnic Institute. E-mail: josepe2@rpi.edu
** Dept. of Computer Science - Rensselaer Polytechnic Institute. E-mail: lioanj@rpi.edu
† Dept. of Cognitive Science - Rensselaer Polytechnic Institute. E-mail: sim@rpi.edu

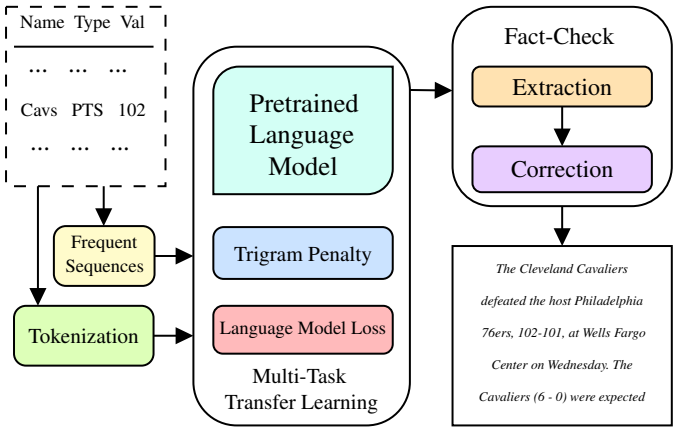

Figure 1
Overview of the approach

deliver information more effectively and reduce the inclusion of data that does not exist in the input, and is hallucinated by the language model. Finally, the generated text should typically follow the same writing style and have the same topic and word choice preferences as the training examples to provide a familiar reading experience.

Multiple datasets have been used for exploring the data-to-text task, including RotoWire (Wiseman, Shieber, and Rush 2017), WebNLG (Gardent et al. 2017), and E2E (Novikova, Dušek, and Rieser 2017). We choose to work with the RotoWire dataset. The RotoWire dataset contains statistics of NBA basketball games with corresponding human-written narratives. This dataset presents a unique challenge by requiring models to form relatively long narrative descriptions with many numbers embedded in them (14.25 sentences and 25.49 numbers per description on average). Previous works on this dataset utilized explicit content planning, attention, and copy mechanisms, but can still suffer from insufficient fluency and accuracy in the generated text. Both hurt peoples' reading experience and prevent them from understanding the data without confusion. Section 2 summarizes related work, and Section 3 discusses the imperfections in their generated text.

We propose a three-step generate–extract–correct pipeline for the data-to-text task as shown in Figure 1. This model helps to ground the generated text by emphasizing its accuracy and reducing potential confusion. It does so by having specific fact-checking and correction procedures after text generation. For generating the text, we investigate two techniques for enhancing transfer learning-based data-to-text techniques. First, to improve the language model's capacity to learn the local structure and word choices from the training data, we mine high-frequency trigrams from the training data. During the transfer learning process, we add an auxiliary task of learning these trigram word combinations. This technique helps our model produce text written in the same style as the training samples, and hence, helps the readers comprehend them. Secondly, instead of directly outputting the generated text, we employ an extract-correct post-processing step to improve the generated text's accuracy. First, information extraction is applied to the generated text for retrieving the information mentioned in it. The retrieved contents are then compared with the input data for checking their accuracy. If mistakes are found, they are fixed in the subsequent correction step. This can significantly reduce generation errors introduced by the pre-trained language models.

2. Related Work

Early approaches to data-to-task have relied on domain-specific knowledge and curation by experts. Such techniques can generate coherent narratives, but suffer from lacking flexibility and variations in the generated text. These approaches often involve developing complex rule-based templates in collaboration with experts in the field, as in (Reiter et al. 2005). More recently, deep learning techniques have been employed to encode data records into a semantic vector space, which can then be decoded and translated into output summaries. Early work in deep-learning-based data-to-text models often linearizes the input records, encoding them as a sequence of facts. (Wiseman, Shieber, and Rush 2017) shows the limitations of using recurrent architectures on such large structured data, which often fails to capture long-term relationships in the data. More recently and in contrast to the practice of linearly encoding records, (Puduppully, Dong, and Lapata 2019; Rebuffel et al. 2020) have used more complex schemes to encode input records, taking into account content planning and the structure of the input records. These models focus on end-to-end training and utilize planning or attention mechanisms, arguing that the previous linear encoding of input records has prevented models from extracting meaningful relationships hidden in the data.

Many recent advances in natural language processing have been attributed to the Transformer architecture (Vaswani et al. 2017), which not only have a strong language comprehension capacity but are also able to leverage language modeling skills to generate fluent text (Radford et al. 2019). Transfer learning, in which models are pre-trained on an unrelated, data-rich task, and later finetuned on a downstream task, has been shown to be very effective in many tasks (Raffel et al. 2020). In particular, (Kale 2020) demonstrates that finetuning the T5 model outperformed many other multi-stage pipelined approaches in three data-to-text benchmarks. The tasks in (Kale 2020) only require short-scale generations. In contrast, the RotoWire dataset contains longer narrative descriptions with many numbers (average of 24), posing a very different challenge.

The idea of rewriting part of the generated text for achieving a better quality has been explored in a few works. (He, Peng, and Liang 2019) used rewriting to increase the "surprise" factor of a generated sentence, and thus make the sentences more fun to read. (Song et al. 2020) rewrites the generated dialogue to make its tone consistent with the speaker's personality profile. In this work, we apply the rewriting idea to improve the accuracy of the generated text.

This work seeks to combine multiple aspects of recent advances in data-to-text and broader text generation by performing a multi-task (Luong et al. 2015) transfer learning on transformer architectures for the data-to-text task, and by introducing a post-processing module to improve the accuracy of generated descriptions. In contrast to previous work, we argue that the transformer model would be good at extracting latent relationships in input data due to their strong language and understanding skills, even if that data is encoded linearly. Our results show we can dependently improve transfer learning for data-to-text tasks based on multiple language models, including T5 (Raffel et al. 2020).

3. Case Studies of Generation Errors

The sentences in the generated text need to be grounded in their context, i.e., they need to be accurate and consistent with each other. Unfortunately, because of the complexity of the task, existing models often cannot ensure self-consistency, contain inaccurate records, and suffer other readability issues. This section provides examples of these challenges in generations achieved with previous SOTA models on the RotoWire dataset. Complete examples can be found in Appendix A.

Table 1
Duplicate percentage, average numbers of records, sentences, erroneous records, and duplication per generated description on the test set. Compared between human written descriptions (**Gold**), [Wiseman, Shieber, and Rush 2017]'s template-based model (**Template**) and neural model (**WS-2017**), [Rebuffel et al. 2020]'s best model (**Heir-k**), [Puduppully, Dong, and Lapata 2019]'s best model (**NCP+CC**), and our best model (**Bart**$_{Tri+Fact}$).

Model	Dup %	# Rec	# Sent	# Err	DupSent
Gold	0.14 %	25.49	14.25	1.49	0.05
Template	0.01 %	54.26	8.11	0.59	0.88
WS-2017	30.58 %	45.18	15.19	11.23	1.69
Heir-k	13.34 %	32.61	14.10	6.38	0.21
NCP+CC	15.77 %	45.96	12.11	5.52	0.89
Bart$_{\text{Tri+Fact}}$	1.27%	55.38	13.03	5.10	0.07

3.1 Duplicate Information

A common issue with the generated text is that it includes redundant or repeated information. Take, for example, the following excerpt generated from the model defined in (Puduppully, Dong, and Lapata 2019):

Tristan Thompson chipped in with seven points and 13 rebounds, marking his first double-double of the year. Tristan Thompson chipped in seven points and 13 rebounds as the starting power forward. **Ersan Ilyasova had a solid game off the bench with 21 points (8-13 FG, 4-6 3Pt) and four rebounds**. Gerald Henderson scored 11 (5-9 FG , 1-4 3PT) and **Ersan Ilyasova had a team-high of 21 points (8-13 FG, 4-6 3Pt) and grabbing four rebounds.** It was a season-high for Ilyasova, who hadn't reached double figures in points twice this season. Gerald Henderson had 11 points (5-9 FG, 1-4 3Pt) as well.

Sets of duplicate information are highlighted with italics, boldface, and underlines respectively. To get an estimate of the number of semantically similar sentences in the generated descriptions, we run a simple cosine similarity test. Two sentences are considered duplicate if the cosine similarity of their average word2vec embeddings (Rehurek and Sojka 2011) is greater than 0.9. Using this technique, we get an average of 0.89 pairs of redundant sentences per description on the test set for (Puduppully, Dong, and Lapata 2019) (See Table 1 for full statistics), implying that almost every generated description has some form of duplicate information. Further, by extracting records from generated descriptions using an information extraction system, we see 15.77% duplicate records for (Puduppully, Dong, and Lapata 2019) and 30.58% for (Wiseman, Shieber, and Rush 2017) as shown in Table 1.

The duplication can affect the overall readability of the generated descriptions, impacting their coherency and conciseness. We address this issue by finetuning large transformer language models, which have been shown to generate consistent text with minimal duplicates.

3.2 Erroneous Information

In many cases, SOTA models generate sentences with erroneous information, such as records that didn't exist in the data, or incorrect scores. Below are excerpts from (Puduppully, Dong, and Lapata 2019) and (Rebuffel et al. 2020) that show these inconsistencies, highlighted in bold:

1. **Greg Beasley** led the bench with 17 points, two rebounds, two assists and one steal.

2. Kobe Bryant led the Lakers with **26** points (**10 - 20** FG, **2** - 4 3Pt, **4** - 4 FT), 12 rebounds, four assists, one steal and one block in 38 minutes.

3. The Memphis Grizzlies (5 - **2**) defeated the Phoenix Suns (3 - 2) Monday **1 - 2** at the Talking Stick Resort Arena in Phoenix.

In the first example, "Greg Beasley" is not an actual player in the NBA, and in the second and third examples, incorrect scores were generated. Using an information extraction system (see Section 5.1 for details) and comparing extracted records to actual input records, we found an average of 11.23 incorrect records per generated description for (Wiseman, Shieber, and Rush 2017)'s neural model, and 5.52 incorrect records per generated description for the (Puduppully, Dong, and Lapata 2019) model. We address this issue by post-processing generated descriptions and correcting erroneous information in an ad hoc fashion.

A more significant issue is that the text descriptions used in training often contain sentences that refer to information not existing in the input data and, therefore, are not grounded by data. For example, the text in Table 2 mentions, "The Sixers will return to action on Wednesday, when they host the Sacramento Kings for their next game." Data-to-text models often learn the "need" of adding sentences like this due to their prevalence in the text used for training. However, the RotoWire dataset does not contain data on each team's next match, so the model ends up making up the information in the generated text. This issue where models generate text but cannot relate it with real-world data is a severe limitation of many data-to-text models. Since writers often utilize information outside of the paired data in their writings, it is hard for machine learning models to address this without external knowledge. In (Reiter et al. 2005) where human authored templates are used for text generation, this problem is particularly avoided by generating more concise descriptions and including more real data in the generated text. As shown in Table 1, the average number of records mentioned in the generated text is 54.26 in (Reiter et al. 2005), while the average number of sentences used is only 8.11. In Gold, i.e., the training data, the average number of records is only 25.49, and the average number of sentences is 14.25. Other models typically also generate text that includes more records than Gold. Our model generates text with a very similar number of records as the template model. We believe, as a result, our generated text contains less made-up information and is more grounded.

3.3 Grammar and Consistency of Text

Sometimes, generated text can be awkwardly phrased, affecting readability. Excerpts from descriptions generated using the model from (Puduppully, Dong, and Lapata 2019) display this:

1. However, a standout effort in the second half was the play of the dynamic duo of **D'Angelo Russell and D'Angelo Russell**, who combined for 51 points on the night.

2. **Derrick Favors (knee) sat this one out with a sore back**, while Gordon Hayward returned ...

3. **The Pacers are now 2 - 3 in the first three games of their nine - game homestand.** They are now 2 - 3 on the road this season.

The first example duplicates the same entity in the same sentence. The last two examples contain contradictions: Derrick Favors injured his knee but sat out with a sore back, and the Pacers' win-loss ratio is 2 - 3, when they are described to have only played three games so far. We address this issue by adding an auxiliary objective while finetuning the language model, which is designed to help the model keep high-frequency sequences of words together and better learn the writing styles of professional sports summaries.

Table 2
Sample data-record table (top) paired with a truncated human-written news summary (bottom). Corresponding records are bolded.

Team	WIN	LOSS	PTS	DREB	FG3_PCT	...
Raptors	11	6	122	34	68	...
76ers	4	14	95	26	41	...

Player	H/V	PTS	AST	REB	FG	TO	...
Carroll	H	10	3	5	4	0	...
Siakam	H	8	0	3	4	1	...
Henderson	V	0	2	1	0	1	...

The host Toronto **Raptors** defeated the Philadelphia **76ers**, **122 - 95**, at Air Canada Center on Monday. The Raptors came into this game as a monster favorite and they didn't leave any doubt with this result. Toronto just continuously piled it on, as they won each quarter by at least four points. 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**. ... The Sixers will return to action on Wednesday, when they host the Sacramento Kings for their next game. ...

3.4 Balance of Statistics vs. Descriptors

While it is often beneficial for the descriptions to include many statistics about a game, there has to be a balance between the number of records and descriptive sentences about the game or the players. If a description contains too many records, it can often feel like reading a wall of data, in which case the information would be better conveyed through a table. However, if there are too few records, readers may not be satisfied. (Wiseman, Shieber, and Rush 2017)'s Template model skews towards "wall of data", containing over 54 records in only 8.11 sentences on average (Table 1). We rely on our language model and the auxiliary training objective to learn the correct ratio of records to descriptors.

4. Proposed Approach

To address the challenges presented in data-to-text generation, we decompose the generation task into three steps: generation, information extraction, and correction. The generation pass involves four steps as shown in Figure 1:

1. Tokenize the input record table R, extract frequent sequences (trigrams) from R.
2. Use transfer learning to finetune a language model with an auxiliary task of learning high-frequency word sequences from training data via trigram penalty.
3. Feed tokenized input into the finetuned language model, generate text y.
4. Feed y into the Fact-Check module, receive final text y'.

We hypothesize that a pretrained language model will be able to overcome the duplication and fluency challenges identified in Sections 3.1 and 3.3. We test our approach on three state of the art language models: T5 (Raffel et al. 2020), Bart (Lewis et al. 2019), and Pegasus (Zhang et al. 2020). T5 was employed by (Kale 2020) for data-to-text tasks. BART and Pegasus are selected because we believe their BERT style bi-directional encoders can efficiently attend to our input records, and their GPT-2 style auto-regressive decoders are ideal for generating fluent text.

While our generation system is not trained end-to-end, it is automated and does not require human intervention during execution. We also argue for its simplicity. Retraining the system for

working with another dataset will only require a quick finetuning pass (averaging about 1 hour on an Nvidia Titan RTX) rather than the full training process from scratch.

4.1 Tokenization and Notation

To pass a table of records R to a language model, we first tokenize the data by prefacing records with "special field" tokens. Our finetuning pass then optimizes a cross-entropy loss between the model's output y and professionally written text $\hat{y}$, with an added auxiliary task for learning high-frequency word sequences in the training data.

Adopting the notation from (Puduppully, Dong, and Lapata 2019), the input to our model, R, is a table of records from match m (see the top of Table 2 for an example.) Each data record r_j has 5 features: the entity which it belongs to ($r_{j,1}$; e.g. Cavaliers, Stephen Curry), its value ($r_{j,2}$; e.g. 102, Golden State), its relation type ($r_{j,3}$; e.g. POINTS, REBOUNDS), whether the record belongs to the home or away team ($r_{j,4}$; HOME or AWAY), and whether the record belongs to a team or a player ($r_{j,5}$; TEAM or PLAYER), represented as $\left\{r_{j,k}\right\}_{k=1}^{5}$. The total number of records is given by $|R|$. The output y is a text description of R containing words $y_1 \cdots y_{|y|}$, where $|y|$ is the length of the text. The gold text description paired with each R in the dataset is then $\hat{y}$. See Table 2 for an example record table (top) and paired text description (bottom).

Records and descriptions are tokenized using byte-pair encoding (BPE). To model each record r_j, we introduce multiple special field tokens that each correspond to a specific record relation type and whether the record belongs to a team or a player ($r_{j,3}$ and $r_{j,5}$). This ensures that the representation for a record type is never split by the tokenizer, and reduces the total size of our tokenized input (at the cost of a slightly increased vocabulary), allowing us to pass a longer context to the model.

For each match m's table of records R, we start by tokenizing the team-level records such as team-wins and team-points, then we follow with the records for all the players. We also add special "HOME" and "AWAY" tokens that separate each new entity and gives the model information about which team each record belongs to ($r_{j,4}$). For each match, we first convert the table of records to an easily tokenizable string. For example, the table of records given by the top part of Table 2 would be converted to the following string:

```
<|HOME|> Raptors <|TM-PTS|> 122 <|TM-REB|> 42
<|TM-AST|> 22 <|TM-WINS|> 11 <|TM-LOSSES|> 6
<|AWAY|> 76ers <|TM-PTS|> 95 <|TM-REB|> 38
... ...
```

4.2 FineTuning and Trigram Penalty

Using the tokenized dataset, we finetune large transformer models to generate the text description y given the tokenized input R. Finetuning is the process of taking a model that was initially trained (pre-trained) on a large dataset, and further training (finetuning) it using the same objective on a different and typically smaller dataset. This is known as transfer learning (Raffel et al. 2020), and it allows the language model to learn vocabulary, grammar, structure, and linguistic features of language during the pre-training step on a vast amount of data, then further finetuning on the data-to-text dataset allows the model to generate fluent and consistent text with linguistic features of the new dataset (in our case, the descriptions y).

To better learn common phrases in the paired gold text descriptions, we add an auxiliary task during the finetuning step to increase the likelihood of generating word sequences frequent in $\hat{y}$. We hypothesize that this objective can help the language model generate text that more closely follow the language patterns in $\hat{y}$, and improve the frequency of expressions commonly

Algorithm 1: Trigram loss penalty

Data: x, x_{t-1}, x_{t-2}, TG
Result: Penalty: p
$p \leftarrow 1$;
foreach $R \in TG$ **do**
  if $(x_{t-2}, x_{t-1}) \in R$ **then**
    if $x \notin R$ **then**
      $p \leftarrow p + 1$;
    end
  end
end

seen in the training data. After some experimentation, we chose to focus on trigram sequences. To generate a list of frequent trigrams, we comb through gold human written text in the training set, create a count of each sequence of 3 words, and choose the 100 most common sequences. For now, we ignore any word sequences that contain data records, e.g. "scored 2 points". Examples of enforced trigrams include: "double-digit favorite", "led the way", "triple - double", and "of the season". Because of the inclusion of such word sequences, this task may indirectly help the model with topic selection as well.

During finetuning, we minimize a cross-entropy loss with label smoothing (Pereyra et al. 2017) combined with a penalty factor that scales the loss if frequent trigrams aren't being generated (or are only partially generated). Given target word y, output token x, the previous two tokens x_{t-1} and x_{t-2} and a list of frequent trigrams TG, we minimize

$$\mathcal{L}(x, x_{t-1}, x_{t-2}, y, TG) = \text{Cross Entropy}(x, y) + \alpha \log f(x, x_{t-1}, x_{t-2}, TG)$$

where α is a hyperparameter to scale the trigram penalty $f(x, x_{t-1}, x_{t-2}, TG)$, given by Algo. 1.

We also enforce a minimum and maximum length penalty. Output texts are generated using beam search with a beam size of 4, and we remove duplicate trigrams during the search to avoid repetition following (Paulus, Xiong, and Socher 2017).

To examine how robust this proposed auxiliary task is for improving the performance of transfer learning, we test our approach on three state-of-the-art language models: T5 (Raffel et al. 2020), Bart (Lewis et al. 2019), and Pegasus (Zhang et al. 2020) in Table 3.

4.3 Post-Processing Fact-Check

For post-processing, we employ a two-step, extract-and-correct process shown in Figure 2, relying on an information extraction system to extract records from the generated text y, then passing these extracted relations along with the accurate input R to a correction module that replaces those incorrect values in y.

4.3.1 Information Extraction

For the information extraction (IE) component of our fact-check system, we finetune a RoBERTa (Liu et al. 2019) transformer with a classifier head to predict $r_{j,3}$ (i.e. the relation type between an entity and value) given all pairs of word-spans in an input. Unrelated pairs are

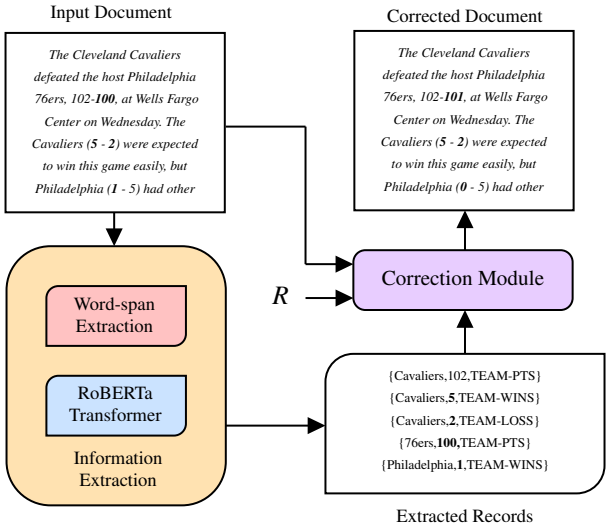

Figure 2
Extraction and Correction

predicted as ε and ignored. Thus, the model learns to minimize

$$\mathcal{L}(\theta) = -\sum_{j} \log \sum p(r_{j,3} = r'_{j,3} \mid r_{j,1}, r_{j,2}; \theta)$$

for all text spans $\{r_{j,1}, v\}$. The training dataset for this task was developed in the same way as the IE dataset proposed in (Wiseman, Shieber, and Rush 2017). We programmatically extract text spans in the gold summaries by looping through each entity and number in each sentence, then search the records in R for a relation $r_{j,3}$ that corresponds to the extracted span. If an entity and number aren't found together in R, we give a ε label. A RoBERTa model trained on this dataset achieves an 83.6% accuracy when evaluated on the test set. This is sufficient to improve generations as part of the fact-check module, despite being lower than the 90% accuracy claimed by the CNN/LSTM ensemble IE system (Wiseman, Shieber, and Rush 2017) used to calculate the objective evaluation metrics.

4.3.2 Correction

Given a *sentence*, trained RoBERTa *model*, and corresponding table of records R, we begin the fact-check by looping through the sentence to extract text spans. For each word in tokenized *sentence*, we first check if the word corresponds to an entity $r_{j,1}$ in R. Next, we loop through each number v in the sentence and construct a span from the entity and that value. This span is passed to *model* along with *sentence*, which predicts the relation between the entity and value $r'_{j,3}$. Finally, we check the correct value of v (i.e. $r_{j,2}$) given $r_{j,1}$ and the predicted $r'_{j,3}$, and replace v in *sentence* with $r_{j,2}$ if the values diverge. This way, we can find sentences with incorrect values in y, replace the wrong values with the correct ones from corresponding input records, and finally rewrite to new output text y'. Pseudocode of the whole extraction/correction process is provided by Algo. 2.

Algorithm 2: Fact-Check

Data: $R, model, sentence$
Result: Corrected: $sentence$
$p \leftarrow 1$;
$S \leftarrow tokenize(sentence)$;
foreach $ent \in S$ **do**
  if $ent \in [\, r_{j,1}$ ***for*** $j \in |R|\,]$ **then**
    foreach $value \in S$ **do**
      if $isNumber(value)$ **then**
        $span \leftarrow \{ent, value\}$;
        $rel \leftarrow model.forward(S, span)$;
        $r_{j,2} \leftarrow R[ent, rel]$;
        if $value \neq r_{j,2}$ **then**
          $sentence[value] \leftarrow r_{j,2}$
        end
      end
    end
  end
end
return $sentence$

5. Evaluation

We train and evaluate our model on the RotoWire data from the BoxScore dataset (Wiseman, Shieber, and Rush 2017). There are a total of 4853 distinct text descriptions covering basketball games played between 1/1/2014 and 3/28/2017. Each game is paired with an average of 628 records (with an average of 28 separate entities). The descriptions are relatively long, averaging 337 words in 14 sentences. We followed the same split introduced in the dataset, training on 3398 data/description pairs, using 727 for validation, and 728 for testing.

To show that the effectiveness of our approach is model-agnostic, we ran objective evaluations on each of the pretrained T5, Bart, and Pegasus models (See Table 3 for comparisons.) Note that these models follow the encoder-decoder transformer architecture. Tests with encoder or decoder only models such as BERT and GPT2 were unable to generate grammatical text for this task. We believe that having a separate encoder and decoder is ideal for the data-to-text task as it allows for the model to better learn an internal representation for the input records R, then separately focus on translating that representation into a text description y. The subscript **Base** models were trained without trigram loss and unprocessed. Subscript **Tri** models were trained with trigram loss but generated without fact-checking. Finally, subscript **Tri+Fact** models utilize our full pipeline, and were trained with trigram loss and processed with fact-checking.

In addition, we compared the performance of our best model with that of (Wiseman, Shieber, and Rush 2017)'s template-based (**Template)** model and neural model (**WS-2017**), (Puduppully, Dong, and Lapata 2019)'s best model (**NCP+CC**) and (Rebuffel et al. 2020)'s best model (**Heirk**). Results on the test set can be found in Tables 4-5 (SOTA results bolded).

Table 3
Transformer Model Comparison on Test Set.

Model	RG		CS		CO
	#	P%	P%	R%	DLD%
$T5_{Base}$	15.33	46.71	22.76	29.4	10.81
$T5_{Tri}$	19.33	66.56	29.47	33.67	12.38
$T5_{Tri+Fact}$	25.97	77.68	28.99	35.64	12.67
$Psus_{Base}$	21.06	54.88	24.85	37.61	14.52
$Psus_{Tri}$	31.31	72.92	28.53	48.43	16.21
$Psus_{Tri+Fact}$	33.31	87.17	**31.67**	47.33	**17.06**
$Bart_{Base}$	44.10	80.89	26.66	57.25	14.09
$Bart_{Tri}$	46.19	86.14	27.62	58.47	16.56
$Bart_{Tri+Fact}$	**50.60**	**89.90**	27.60	**60.49**	16.18

5.1 Objective Evaluations

We evaluated model outputs on the validation and test sets using the metrics defined in (Wiseman, Shieber, and Rush 2017). These metrics use a neural ensemble IE system to extract records from gold description $\hat{y}$ and our models' output y. This system ensembles the predictions from 3 CNN based architectures and 3 Bi-Directional LSTM based architectures trained to predict relations given all pairs of word-spans in an input. We then compared whether the extractions align or diverge from the gold summaries. The following metrics are used:

Relation Generation (RG): measures the "correctness" of the records extracted from y, as the proportion of extracted records that is also in R, given in terms of precision **P%** and number of unique generations **#**.

Content Selection (CS): measures how well y matches $\hat{y}$ in terms of selecting which records to generate, as the proportion of records extracted from y that are also in $\hat{y}$, given in terms of precision **P%** and recall **R%**.

Content Ordering (CO): measures how well the order of records in y matches the order of records in $\hat{y}$, given as the normalized Damerau-Levenshtein Distance **DLD%** between records extracted from y and $\hat{y}$.

(Wiseman, Shieber, and Rush 2017) notes that CS primarily targets the "what to say" aspect of evaluation, CO focuses on the "when to say it", and RG targets both.

In addition, we report BLEU, ROUGE-L, and METEOR, using paired human-written descriptions as a reference. A lot of work on this dataset only reports BLEU. Like BLEU, ROUGE-L and METEOR are commonly used metrics when evaluating automated text generation. ROUGE-L emphasizes recall, and METEOR has been shown to correlate better with human judgment and doesn't penalize using synonyms (Denkowski and Lavie 2014).

Table 3 shows that our proposed auxiliary task and post-processing procedures improved the performances of all three language models. Our pipeline improves evaluation results by 2.34% up to 26.4% on average comparing to finetuning the language models alone. Overall, the $Bart_{Tri+Fact}$ model performed the best and is what we will use to compare to the previous state-of-the-art.

Table 4 shows that the $Bart_{Tri+Fact}$ model performs better on RG# and RG P% than all other models except for the Template model. For CS, the $Bart_{Tri+Fact}$ has higher recalls than all other models, including the Template model.

To further investigate these results, we computed the average length and the amount of duplication that exists in each model's output on the test set. The results are shown in Table 1.

Table 4
Objective Evaluation on Test Set.

Model	RG		CS		CO
	#	P%	P%	R%	DLD%
Template	**54.23**	**99.95**	26.61	59.15	14.44
WS-2017	23.58	75.09	28.25	35.81	15.37
NCP+CC	34.12	88.12	**34.49**	51.13	**18.66**
Heir-k	22.83	79.22	34.12	37.88	17.10
$\text{Bart}_{\text{Tri+Fact}}$	*50.60*	*89.90*	27.60	**60.49**	16.18

Table 5
BLEU, ROUGE-L, and METEOR Scores.

Model	Validation			Test		
	BLU	ROG	MET	BLU	ROG	MET
Gold	100	100	100	100	100	100
Template	8.97	18.54	21.67	8.93	18.59	21.38
WS-2017	14.57	23.00	31.44	14.19	22.86	31.39
NCP+CC	16.19	23.69	32.06	**16.5**	23.67	31.81
Heir-k	**16.3**	23.27	33.26	**16.5**	23.33	33.53
$\text{Bart}_{\text{Tri+Fact}}$	14.19	**24.34**	**34.88**	14.52	**24.24**	**34.48**

$\text{Bart}_{\text{Tri+Fact}}$ only generated 0.07 pairs of duplicated sentences per description and 1.27% of duplicate records. This is a notable improvement compared to other models. While the total numbers of sentences generated by these models are similar, with less duplication, the descriptions generated by $\text{Bart}_{\text{Tri+Fact}}$ contain more unique records. This can explain why we have better results on RG metrics. Similarly, the more unique records can account for our higher CS recall. The fact that we have lower CS precision indicates our generated descriptions do not necessarily follow the same content plan that the gold descriptions use, and may generate more records that aren't mentioned in Gold. As shown in Table 1, generating more records than Gold is common; and having a higher number of records reduces the amount of made-up information in the generated text. In fact, our model generated a similar amount of records as the Template model. However, unlike Template, our model also generates sufficient descriptor text such that reading the generated descriptions doesn't feel like reading a wall of data, as shown by our conciseness and coherence scoring higher than Template in the subjective evaluations (Table 6). Therefore, we believe our generated text descriptions are better grounded for the readers.

As shown in Table 5, the $\text{Bart}_{\text{Tri+Fact}}$ model has higher METEOR and ROUGE-L scores, but slightly lower BLEU when compared to other models. This suggests that the text generated from our models contains a lot of synonyms, which is expected when using a pretrained language model.

Interestingly, $\text{Bart}_{\text{Tri+Fact}}$ improves the CS scores over the base Bart model in both test and validation sets, while in theory, the post-processing we perform should not affect content selection (CS). We suspect this may result from the IE model being able to extract more accurate information in the text generated by $\text{Bart}_{\text{Tri+Fact}}$. Further, the auxiliary task of learning high-frequency word sequences may have helped the model select more accurate records.

Table 6
Results from Subjective Evaluations

Model	Grammar	Coherency	Conciseness
Gold	**24.444**	**26.111**	-3.889
Template	-48.889	-44.444	-1.667
NCP+CC	-10.000	-1.111	-1.111
Heir-k	10.556	1.667	**1.667**
$\text{Bart}_{\text{Tri+Fact}}$	20.000	13.889	0.000

5.2 Subjective Evaluations

Using the same design as in (Puduppully, Dong, and Lapata 2019), we conducted a human evaluation study on Amazon Mechanical Turk (MTurk) to assess the subjectively perceived quality of the generated text. We randomly picked 30 basketball matches in the test set. We then asked crowd-workers to compare a human-written description (Gold), and descriptions generated by Template, NCP+CC, Heir-k, and our $\text{Bart}_{\text{Tri+Fact}}$ with each other. For each game, we arranged the 5-tuple of generated description into pairs for comparison, resulting in 10 pairs. Each pair was shown to 3 different crowd-workers. They were asked to choose the *better* description according to:

Coherence: Is the summary easy to read? Does it follow a logical order?
Conciseness: Is the summary concise? Does it avoid redundancy and repetition?
Grammar: Does the summary read fluently? Does it use proper grammar?

All of these questions are important for people's subjective experiences of whether the generated text is well-grounded. We recruited 450 subjects. Each made two comparisons. This results in a total of 900 comparisons. We then calculated the score of a system for each criterion as the difference between the percentage of times it was chosen as the *better* one and the percentage of times it was chosen as the *worse* one. The scores range from -100 (absolute worst) to +100 (absolute best).

The results of this study are displayed in Table 6. The evaluations for $\text{Bart}_{\text{Tri+Fact}}$ are similar to those for Gold with slightly lower Coherence and Grammar scores, but a better Conciseness score. Counting a score of 1 each time a description generated from an algorithm is selected, and 0 otherwise, we performed one-way ANOVA on the subjects' ratings of Grammar, Coherency, and Conciseness. The results show a significant difference ($p < .05$) among the subject's ratings for Grammar and Coherency, but not for Conciseness. We performed additional T-tests between the evaluations for $\text{Bart}_{\text{Tri+Fact}}$ and other algorithms using two-tailed unpaired T-tests. At the .05 level, there is no significant difference between $\text{Bart}_{\text{Tri+Fact}}$ and Gold or Heir-k. However, $\text{Bart}_{\text{Tri+Fact}}$ did perform significantly better than NCP+CC and Template in regards to Grammar and Coherency. Template performs significantly worse in Coherence and Grammar, probably because of its restricted and rigid sentence templates. Overall, $\text{Bart}_{\text{Tri+Fact}}$ was rated higher than the other generative models (Heir-k and NCP+CC). Our Conciseness is also slightly higher than every model except Heir-k. This may imply that our system strikes the right balance between data and descriptors.

6. Conclusion and Future Work

We aim at generating well-grounded text for the data-to-text task by emphasizing its accuracy, coherency, and conciseness. We propose a generate-extract-correct pipeline and incorporate an

auxiliary task of learning high-frequency word sequences. Evaluations on the RotoWire dataset demonstrate the auxiliary task and the ad hoc extract-correction processes improved transfer learning performances using all three language models – BART, T5, and Pegasus. Subjective evaluation using mTurk show that the results generated by our model are comparable to Gold descriptions.

For future work, we want to look further into the consistency of the generated text. Minimally, the usage of transition phrases, e.g., "also" and "but" should be consistent with the conjunction or contradiction relationship between sub-sentences. Furthermore, the sentiment of a sentence, should be consistent with the comparison in it. This means that the attitude towards subjects in a sentence should correlate with the generated text. For instance, if A defeats B, then A's score should be higher than B's. We are also interested in connecting this work with common sense reasoning. One limitation of work in this area is the generated text can only state factual information, but not offer any explanations while human written text often involves some form of explanations and inferences.

References

Denkowski, Michael and Alon Lavie. 2014. Meteor universal: Language specific translation evaluation for any target language. In *Proceedings of the EACL 2014 Workshop on Statistical Machine Translation*.

Gardent, Claire, Anastasia Shimorina, Shashi Narayan, and Laura Perez-Beltrachini. 2017. The WebNLG challenge: Generating text from RDF data. In *Proceedings of the 10th International Conference on Natural Language Generation*, pages 124–133.

He, He, Nanyun Peng, and Percy Liang. 2019. Pun generation with surprise. *arXiv preprint arXiv:1904.06828*.

Kale, Mihir. 2020. Text-to-text pre-training for data-to-text tasks. *arXiv*, pages arXiv–2005.

Lewis, Mike, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Ves Stoyanov, and Luke Zettlemoyer. 2019. Bart: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. *arXiv preprint arXiv:1910.13461*.

Liu, Yinhan, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized bert pretraining approach.

Luong, Minh-Thang, Quoc V Le, Ilya Sutskever, Oriol Vinyals, and Lukasz Kaiser. 2015. Multi-task sequence to sequence learning. *arXiv preprint arXiv:1511.06114*.

Novikova, Jekaterina, Ondřej Dušek, and Verena Rieser. 2017. The e2e dataset: New challenges for end-to-end generation. *arXiv preprint arXiv:1706.09254*.

Paulus, Romain, Caiming Xiong, and Richard Socher. 2017. A deep reinforced model for abstractive summarization. *CoRR*, abs/1705.04304.

Pereyra, Gabriel, George Tucker, Jan Chorowski, Łukasz Kaiser, and Geoffrey Hinton. 2017. Regularizing neural networks by penalizing confident output distributions. *arXiv preprint arXiv:1701.06548*.

Puduppully, Ratish, Li Dong, and Mirella Lapata. 2019. Data-to-text generation with content selection and planning. In *Proceedings of AAAI*, Honolulu, Hawaii.

Radford, Alec, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9.

Raffel, Colin, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer.

Rebuffel, Clément, 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.

Rehurek, Radim and Petr Sojka. 2011. Gensim–python framework for vector space modelling. *NLP Centre, Faculty of Informatics, Masaryk University, Brno, Czech Republic*, 3(2).

Reiter, Ehud, Somayajulu Sripada, Jim Hunter, Jin Yu, and Ian Davy. 2005. Choosing words in computer-generated weather forecasts. *Artif. Intell.*, 167(1–2):137–169, September.

Song, Haoyu, Yan Wang, Wei-Nan Zhang, Xiaojiang Liu, and Ting Liu. 2020. Generate, delete and rewrite: A three-stage framework for improving persona consistency of dialogue generation. *arXiv preprint arXiv:2004.07672*.

Vaswani, Ashish, 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.
Wiseman, Sam, Stuart M. Shieber, and Alexander M. Rush. 2017. Challenges in data-to-document generation. *CoRR*, abs/1707.08052.
Zhang, Jingqing, Yao Zhao, Mohammad Saleh, and Peter Liu. 2020. Pegasus: Pre-training with extracted gap-sentences for abstractive summarization. In *International Conference on Machine Learning*, pages 11328–11339. PMLR.

Appendix A: Qualitative Comparisons

Below we provide two examples of outputs generated by different systems and manually mark issues identified in Section 3. Erroneous information (3.1) in red, duplicate information (3.2) in blue, inarticulate/illogical sentences (3.3) in green, and inconsistent sentences (3.3) in orange.

Note that there is a small amount of error, even in gold. For duplication, we only marked the places where the information appeared at its second or third times. For spotting inconsistencies, the context needs to be taken into consideration. For example, in the NCP+CC version of the first example, it first says the game's date is Friday, then, later on, says it is Monday and then Tuesday. Similarly, in the $\text{Bart}_{\text{Tri+Fact}}$ version, the Raptors are said to play with both the Knicks and the Nets. In the NCP+CC version of the second example, Middleton never surpassed the 20-points mark based on prior information in the paragraph.

Gold – Summary 1

DeMar DeRozan and Terrence Ross combined for a whopping 55 points on 17-of-28 shooting, helping lead the Raptors to outstanding 51 and 56 percent success rates from the field and three-point range, respectively. Kyle Lowry went for 18 points, while DeMarre Carroll supplied 13 points of his own. Jonas Valanciunas registered an 11-point, 13-rebound double-double. The top-heavy Bucks got their own stellar 30-point performance from Giannis Antetokounmpo, while Jabari Parker followed with 27 points. However, only one other Bucks player managed to get into double digits in the scoring column, and that effort came from the bench in the person of Greg Monroe (11 points). Tony Snell (eight points), John Henson (seven points) and Matthew Dellavedova (two points) considerably underwhelmed, while Milwaukee particularly struggled from long range, shooting just 17 percent from beyond the arc.

Template – Summary 1

The Toronto Raptors (17-7) defeated the Milwaukee Bucks (11-12) 122-100. Giannis Antetokounmpo scored 30 points (13-23 FG, 0-3 3Pt, 4-6 FT) to go with 9 rebounds. DeMar DeRozan scored 30 points (7-11 FG, 1-1 3Pt, 15-15 FT) to go with 3 rebounds. Jabari Parker scored 27 points (11-17 FG, 2-3 3Pt, 3-5 FT) to go with 3 rebounds. Terrence Ross scored 25 points (10-17 FG, 4-6 3Pt, 1-3 FT) to go with 5 rebounds. Kyle Lowry scored 18 points (5-9 FG, 4-7 3Pt, 4-4 FT) to go with 3 rebounds. DeMarre Carroll scored 13 points (5-12 FG, 3-7 3Pt, 0-0 FT) to go with 6 rebounds. The Bucks' next game will be at home against the Dallas Mavericks, while the Raptors will travel to play the Bulls.

WS-17 – Summary 1

The Toronto Raptors defeated the visiting Milwaukee Bucks, 122-100, at Air Canada Centre on Friday. The Raptors (17-7) checked in to Saturday's contest with only two wins in their last 11 games, but the Bucks (11-12) checked in to Friday's contest with a 33-point first quarter, and the Bucks (11-12) checked in to Wednesday's contest with a 33-point first quarter. However, the Bucks (11-12) battled back in the second half, outscoring the Raptors by a 33-18 margin over the final 12 minutes. However, the Raptors battled back to make it a one-point game heading into the fourth, but the Raptors battled back to make it a one-point game at the end of the third quarter. The Raptors were led by Giannis Antetokounmpo's 30 points, which he supplemented with nine rebounds, five assists, three steals and three blocks. Parker was next with 27 points, three rebounds, two assists and one steal. Parker was next with 27 points, three rebounds, two assists, one steal and one block. Greg Monroe was next with 11 points, two rebounds, two assists

and a steal. Greg Monroe supplied 11 points, two rebounds, two assists and a steal. Greg Monroe led the second unit with 11 points, two rebounds, two assists and a steal. Malcolm Brogdon led the second unit with 23 points, two assists, two rebounds and three steals. Milwaukee was led by a pair of 30-point efforts from Kyle Lowry and DeMar DeRozan, with the former adding three rebounds and three assists, and the latter supplying three rebounds, three assists, three steals and a block. Norman Powell was next with a bench-leading 25 points, which he supplemented with five assists, five rebounds and a steal. Jonas Valanciunas posted a 11-point, 12-rebound double-double that also included an assist, a steal and a block. Cory Joseph led the second unit with 27 points, seven assists, four rebounds and two steals. The Bucks remain in last place in the Eastern Conference's Atlantic Division. They head to Detroit to take on the Pistons on Tuesday.

NCP+CC – Summary 1

The Toronto Raptors defeated the Milwaukee Bucks, 122-100, at Air Canada Centre on Friday. The Raptors (11-12) checked in to Monday's contest having lost four of their last six games, but the Raptors (17-7) were able to pull away in the second half. The Raptors (17-7) checked in to Tuesday's contest with only two road wins in 17 tries, but the Raptors (17-7) were able to prevail with a solid road victory. The Raptors (17-7) were led by Jabari Parker's 27 points, which he supplemented with three rebounds, two assists, a steal and a block. Giannis Antetokounmpo was right behind him with 30 points, nine rebounds, five assists, three steals and a block. Jabari Parker was next with 27 points, three rebounds, two assists and a steal. Matthew Dellavedova was next with a bench-leading 27 points, which he supplemented with three rebounds, two assists and a steal. Matthew Dellavedova supplied two points, 10 assists and a rebound. The Raptors remain in last place in the Eastern Conference's Central Division. They head to Cleveland to take on the Cavaliers on Sunday night. DeRozan led the way for the Raptors with 30 points, three assists, three rebounds and a steal. Kyle Lowry was next with 18 points, seven assists, three rebounds and a steal. DeMarre Carroll registered 13 points, six rebounds, two assists and a steal. Jonas Valanciunas posted a 11-point, 12-rebound double-double that also included an assist, a steal and a block. The Bucks head back home to take on the Atlanta Hawks on Monday night, while the Raptors visit the Cleveland Cavaliers on Friday night as well.

Heir-k – Summary 1

The Toronto Raptors defeated the Milwaukee Bucks, 122-100, at Air Canada Centre on Monday. The recent form of these teams would indicate an easy Milwaukee win, but Toronto had other ideas. In fact, the Raptors led for nearly every minute of this game, with the Raptors having their only lead in the first quarter. Both offenses were extremely efficient, as they each shot over 53 percent from the field. Toronto dominated in the interior, as they won the rebounding battle, 46-37 and outscored Milwaukee, 62-37 in the paint. The assist-to-turnover ratio was a huge factor as well, with the Raptors recording 26 assists to six turnovers, while the Bucks committed 11 turnovers to 27 assists. Toronto (11-12) has been regressing over recent weeks and this was a tough task north of the border. The bench was the only thing keeping them around, as Terrence Ross and Terrence Ross both scored in double figures off the bench. In fact, Toronto led for the entirety of the second half, while holding a double-digit lead for the majority of that stretch. DeMar DeRozan once again led the team in scoring, as he tallied 30 points, three rebounds and three assists. Kyle Lowry was second on the team, finishing with 18 points, three rebounds and seven assists. Jonas Valanciunas provided 11 points and 12 rebounds. Terrence Ross was a nice spark off the bench, scoring 25 points on 10-of-17 shooting. Giannis Antetokounmpo led the Bucks in scoring, tallying 30 points, nine rebounds and three assists. Kyle Lowry was second on the team, with 18 points, three rebounds, seven assists and two steals.

$\text{Bart}_{\text{Tri+Fact}}$ – Summary 1

The Toronto Raptors (17-7) defeated the Milwaukee Bucks (11-12) 122-100 on Wednesday at the Air Canada Centre in Toronto. The Raptors got off to a hot start in this one, out-scoring the Bucks 33-23 in the first quarter and 36-18 in the second quarter. They were able to coast to a comfortable lead for the rest of the game, as the Bucks weren't able to keep up with the Raptors' high-powered offense. The Bucks were led by Giannis Antetokounmpo, who tallied 30 points (13-23 FG, 0-3 3Pt, 4-6 FT), nine rebounds, five assists and three steals in 39 minutes. Jabari Parker followed up with 27 points of his own, going 11-for-17 from the field and 2-of-3 from the three-point line to score 27 points, while also adding three rebounds, two assists and one steal in 35 minutes. The only other player to score in double digits for the Bucks was Malcolm Brogdon, who finished with 4 points (2-6 FG, 2-1 FT) in 23 minutes off the bench. Meanwhile, DeMar DeRozan led the way for the Raptors, scoring a game-high of 30 points on 7-for- 11 shooting from the floor and 15-for 15 from the free throw line. He added three rebounds and three assists in 32 minutes. Kyle Lowry finished with 18 points, seven assists and three rebounds in 34 minutes. DeMarre Carroll and Jonas Valanciunas added 11 points each, while Terrence Ross chipped in 25 points (10-17 FG) and five rebounds in 21 minutes as a reserve. Jonas Valancunas recorded a double-double of 11 points and 12 rebounds in 23 minutes, while Cory Joseph added seven assists, four rebounds and two steals in 27 minutes. For the Bucks, it was a forgettable night for the starting five, as John Henson, Matthew Dellavedova and Tony Snell combined for just nine points on 1-for 8 shooting in 28 minutes. Up next, the Bucks will head home to take on the Wizards on Friday, while the Raptors will head to New York to play the Knicks on Friday. The Raps will look to extend their winning streak to five games as they travel to Brooklyn on Friday to face the Nets.

Gold – Summary 2

The Milwaukee Bucks (18-17) defeated the New York Knicks (5-31) 95-82 on Sunday at Madison Square Garden in New York. The Bucks were able to have a great night defensively, giving themselves the scoring advantage in all four quarters. The Bucks showed superior shooting, going 46 percent from the field, while the Knicks went only 41 percent from the floor. The Bucks also out-rebounded the Knicks 48-36, giving them in an even further advantage which helped them secure the 13-point victory on the road. Brandon Knight led the Bucks again in this one. He went 6-for-14 from the field and 1-for-3 from beyond the arc to score 17 points, while also handing out five assists. He's now averaging 21 points per game over his last three games, as he's consistently been the offensive leader for this team. Zaza Pachulia also had a strong showing, finishing with 16 points (6-12 FG, 4-4 FT) and a team-high of 14 rebounds. It marked his second double-double in a row and fourth on the season, as the inexperienced centers on the Knicks' roster weren't able to limit him. Notching a double-double of his own, Giannis Antetokounmpo recorded 16 points (6-9 FG, 1-1 3Pt, 3-6 FT) and 12 rebounds. The 12 rebounds matched a season-high, while it was his second double-double of the season. Coming off the bench for a big night was Kendall Marshall. He went 6-for-8 from the field and 3-for-3 from the free throw line to score 15 points in 20 minutes. The Knicks really struggled to score without Carmelo Anthony and Amare Stoudemire. Tim Hardaway Jr led the team as the starting shooting guard, going 6-for-13 from the field and 3-for-5 from the three-point line to score 17 points, while also adding four assists. He's now scored 17 or more points in three out of his last four games, as he has put it on himself to pick up the slack with other key players sitting out. J.R. Smith also put together a solid outing as a starter. He finished with 15 points and seven rebounds

in 37 minutes. Like Haradaway Jr, he's also benefitted from other guys sitting out, and has now combined for 37 points over his last two games. While he didn't have his best night defensively, Cole Aldrich scored 12 points (6-10 FG) and grabbed seven rebounds in 19 minutes. The only other Knick to reach double figures in points was Jason Smith, who came off the bench for 10 points (3-11 FG, 4-4 FT). The Bucks' next game will be at home against the Phoenix Suns on Tuesday, while the Knicks will travel to Memphis to play the Grizzlies on Monday.

Template – Summary 2

The Milwaukee Bucks (18-17) defeated the New York Knicks (5-31) 95-82. Brandon Knight scored 17 points (6-14 FG, 1-3 3Pt, 4-5 FT) to go with 2 rebounds. Tim Hardaway Jr. scored 17 points (6-13 FG, 3-5 3Pt, 2-4 FT) to go with 3 rebounds. Giannis Antetokounmpo scored 16 points (6-9 FG, 1-1 3Pt, 3-6 FT) to go with 12 rebounds. Zaza Pachulia scored 16 points (6-12 FG, 0-0 3Pt, 4-4 FT) to go with 14 rebounds. Kendall Marshall scored 15 points (6-8 FG, 0-2 3Pt, 3-3 FT) to go with 2 rebounds. JR Smith scored 15 points (6-16 FG, 3-7 3Pt, 0-0 FT) to go with 7 rebounds. The Bucks' next game will be at home against the Dallas Mavericks, while the Knicks will travel to play the Bulls.

WS-17 – Summary 2

The Milwaukee Bucks (18-17) defeated the New York Knicks (5-31) 95-82 on Tuesday at Madison Square Garden in New York. The Bucks got off to a quick start in this one, out-scoring the Knicks 22-22 in the first quarter alone. The Bucks were able to use a strong first half, where they out-scored the Knicks 31-18 to seal the victory in front of their home crowd. The Bucks were the superior shooters in this game, going 46 percent from the field and 36 percent from the three-point line, while the Knicks went 41 percent from the floor and just 25 percent from deep. The Bucks were also able to force the Knicks into 16 turnovers, while committing just 16 of their own. The Bucks were led by the duo of Greg Monroe and Khris Middleton. Knight went 6-for-14 from the field and 1-for-3 from the three-point line to score a team-high of 17 points, while also adding five assists and two steals. He's now averaging 20 points and 8 rebounds on the year. Khris Middleton also had a solid showing, finishing with 8 points (2-6 FG, 1-2 3Pt, 3-3 FT) and five rebounds. He's now averaging 16 points and 6 rebounds on the year. The only other Knick to reach double figures in points was Brandon Knight, who chipped in with 17 points (6-14 FG, 1-3 3Pt, 4-5 FT) and five assists. The Knicks' next game will be on the road against the Cleveland Cavaliers on Friday, while the Knicks will be at home against the New York Knicks on Friday.

NCP+CC – Summary 2

The Milwaukee Bucks (18-17) defeated the New York Knicks (5-31) 95-82 on Wednesday at Madison Square Garden in New York. The Bucks were the superior shooters in this game, going 46 percent from the field and 25 percent from the three-point line, while the Knicks went just 41 percent from the floor and a meager 36 percent from beyond the arc. The Bucks were the superior shooters in this game, going 46 percent from the field and 25 percent from the three-point line, while the Knicks went just 41 percent from the floor and a meager 36 percent from deep. The Bucks also forced the Knicks into 18 turnovers, while committing just 11 of their own, which may have been the difference in this game, as the Bucks forced the Knicks into 21 turnovers, while committing just 11 of their own. The Bucks' frontcourt did most of the damage in this game. Giannis Antetokounmpo led the team with 16 points (6-9 FG, 1-1 3Pt, 3-6 FT), 12 rebounds and two blocked shots, while Middleton had eight points (2-6 FG, 1-2 3Pt, 3-6 FT

), five rebounds and two steals in 22 minutes. It was the first time he's surpassed the 20-point mark this season, so it was good to see him get things turned back around. Coming off the bench, Khris Middleton had eight points (2-6 FG, 1-2 3Pt, 3-3 FT), five rebounds and two steals in 22 minutes. The Bucks' next game will be on the road against the Boston Celtics on Friday, while the Knicks will travel to Brooklyn to play the Nets on Friday.

Heir-k – Summary 2

The Milwaukee Bucks (18-17) defeated the New York Knicks (5-31) 95-82 on Monday. Milwaukee has won four straight games, and the deeper they get into the season, the more believable the Bucks' turnaround under coach Jason Kidd appears to be a sustainable change. The Bucks gave all five of the bench players they used at least 21 minutes. Giannis Antetokounmpo led the way with a game-high 16 points and 12 rebounds, while Giannis Antetokounmpo had a double-double of his own with 16 points and 12 rebounds. It was an off-night for New York, as the team shot just 41 percent from the field and 36 percent from beyond the arc. Jared Dudley (12), Kendall Marshall (10) and Johnny O'Bryant (10) round out the six New York players who scored in double figures. Up next, the Bucks will stay home Wednesday to take on the 76ers, while the Knicks will head to Los Angeles on Saturday to take on the Clippers. As has been the regular season for the Knicks, but they didn't have enough swag to win the Bucks. They will hope to continue their hot start as they take on the Bulls in Madison Square Garden on Monday. The Bucks will also have a few days off before traveling to Orlando to take on the Magic on Wednesday. For the Knicks, meanwhile, the Knicks play their seventh straight loss as they take on the Milwaukee Bucks on Tuesday, in a total of 15 games.

Bart$_{\text{Tri+Fact}}$ – Summary 2

The Milwaukee Bucks (18-17) defeated the New York Knicks (5-31) 95-82 on Wednesday at Madison Square Garden in New York. The Bucks got off to a quick start in this one, out-scoring the Knicks 22-21 in the first quarter and never looking back after that. They were led by Giannis Antetokounmpo, who finished with 16 points (6-9 FG, 1-1 3Pt, 3-6 FT) and 12 rebounds in 30 minutes. He also added two assists, two steals and two blocks. Zaza Pachulia added 16 points and 14 rebounds of his own, while Brandon Knight chipped in 17 points and five assists in 32 minutes. It was a nice bounce back game for the Bucks, who had lost four of their last five games coming into Wednesday's contest. The Knicks, on the other hand, have now lost three straight games, as they continue to struggle on offense. They shot just 41 percent from the field and 36 percent from beyond the arc. Tim Hardaway Jr. was the high-point man for the Knicks, finishing with 17 points on 6-of-13 shooting, while J.R. Smith added 15 points, seven rebounds and four assists in 37 minutes. The only other Knick to reach double figures in points was Willy Hernangomez, who scored 10 points (3-11 FG, 4-4 FT) in 24 minutes off the bench. Up next, the Bucks will head home Friday to take on the road to play the Bulls, while the Knicks will travel to Boston on Friday to play against the Celtics.