Low-resource Data-to-Text Generation Using Pretrained Language Models

Anonymous ACL submission

Abstract

Expressing natural language descriptions of structured facts or relations – data-to-text generation – increases the accessibility of a diverse range of structured knowledge repositories. End-to-end neural models for this task require a large training corpus of relations and corresponding descriptions. While such resources are unrealistic for every domain, we do not fully understand how well different data-to-text generation models can generalize to new relations. This work presents an analysis of data-to-text models for unseen relations based on two pre-trained language models (PLMs): T5 and GPT-2. We consider different strategies, including few-shot learning, prompt-tuning, and incorporating other domain knowledge (natural language description of the unseen relations) to identify effective strategies and remaining challenges for improving performance of PLMs on new relations.

1 Introduction

Structured data repositories, or knowledge bases, contain a wealth of information organized to facilitate automated access and analysis. Automated data-to-text systems can transform and organize this knowledge into natural language text snippets that broaden access (Gatt and Krahmer, 2018). The input to these systems takes the form of relations, or triples, and systems process triple sets that consist of sets of subject, predicate and object. Applications of this technology include story or dialogue generation (Moon et al., 2019), open-domain question-answering (Ma et al., 2021; Fan et al., 2019), and text summarization (Wiseman et al., 2017). Domains span journalism (Leppänen et al., 2017), weather forecasts (Ramos-Soto et al., 2014; Mei et al., 2015), financial and sport casting (Plachouras et al., 2016; Chen and Mooney, 2008; van der Lee et al., 2017), and summarizing patient medical histories (Portet et al., 2009).

Historically, data-to-text systems included pipeline approaches with customized models (Gardent et al., 2017). In recent years, pretrained Transformer-based language models (Devlin et al., 2018; Liu et al., 2019; Radford et al., 2019) have come to dominate this task, just as they have other NLP tasks. Recent examples include Mager et al. (2020) and Kale and Rastogi (2020), who use models like GPT-2 (Radford et al., 2019) and T5 (Raffel et al., 2019) to generate language descriptions for relations. To support these types of systems, Nan et al. (2020) introduce DART, an open-domain and large data-to-text generation corpus. Models trained on DART, both larger and more diverse than previous corpora, improve the performance of BART (Lewis et al., 2019) and T5 on the standard WebNLG challenge (Gardent et al., 2017).

This approach requires a PLM to be fine-tuned on a task-specific in-domain dataset (Howard and Ruder, 2018; See et al., 2019; Keskar et al., 2019). The promising results using this paradigm belie the reality that in spite of its aspirations, most domains and relations that one could express fail to appear in DART. Furthermore, the extensive development effort behind DART, and other similar datasets, underscores the challenge of creating an in-domain dataset for each task of interest. Unfortunately, PLMs fine-tuned on a specific domain often do not generalize to a new domain (Harkous et al., 2020). For example, a model trained to generate text for sports relations (DEFEATED, COACHED) is unlikely to generate sensible text for medical relations (DIAGNOSED, INFLAMES).

A variety of methods have emerged within PLM research to address domain or task adaptation. For example, GPT style models have demonstrated improved performance on a new task via few-shot learning with a handful of examples (Chen et al., 2019). Other strategies, such as prompt tuning (Lester et al., 2021), by only updating a small subset of model parameters, can adapt PLMs to perform specific down-stream tasks.

While great progress has been made in utilizing PLMs for data-to-text generation, questions regarding their adaptation to new domains are unanswered. More specifically, it is not clear how well data-to-text models generalize to new relations and how effective these adaptation strategies are at mitigating the challenges of adapting to a low-resource setting. We conduct an evaluation of PLMs for data-to-text generation focused on new (*unseen*) relations (*predicates*). We consider how GPT-2 coupled with strategies such as few-shot training, prompt tuning, and predicate description augmentation performs on new domains as compared to a baseline (state-of-the-art) T5 model fine-tuned on an open-domain dataset. We show that while an out-of-the-box GPT-2 model performs poorly on DART, its performance can be drastically improved by these adaptation methods. We make the following contributions:

- We evaluate GPT2-XL for data-to-text generation. While the zero-shot model performs poorly, we evaluate several strategies to improve performance, including few-shot learning and prompt tuning. Both provide significant improvements on the DART dataset.

- We propose a post hoc re-ranking strategy for GPT-2 that further improves results without requiring additional training data.

- We show how T5 performance compares to GPT2-XL depending on the amount of supervised training data available.

- We evaluate all models on unseen predicates and show how various approaches enable generalization to new relations.

- We evaluate models separately on easy and hard instances to highlight remaining challenges for this task.

- We conduct a qualitative evaluation of the models to identify pathological behaviors.

We provide recommendations for future model and dataset research.

2 Background and Related Work

In the task of data-to-text generation, we are provided a set of triples that include a predicate, subject, and object. The system then produces a text snippet expressing the predicate in natural language. Figure 2 shows examples of predicates from sports domains. The system can be given a set of triples with related predicates (e.g., CLUB, LEAGUE, FORMER_TEAM) and must generate text that expresses the facts encoded by these relations. The resulting text is typically evaluated by comparison to a set of reference texts, which represent various ways of expressing this triple set.

Variations in the formulation of this task depend on the structure of the relations (e.g., tables, triples), the domain of the task (single or open domain), and the source of the data (manually created, automatically derived).

Harkous et al. (2020) follow a generate-and-rerank paradigm to improve the semantic fidelity of the generated text by fine-tuned GPT-2 model. More recently, Ribeiro et al. (2020) propose a new task-adaptive pretraining strategy to adapt BART (Lewis et al., 2019) and T5 (Raffel et al., 2019) models for data-to-text generation. They show that adding an intermediate task-adaptive pretraining step between the task-independent pretraining and fine-tuning further improves the performance of these models on data-to-text generation.

Creating a large enough dataset for fine-tuning PLMs for data-to-text generation is not feasible or cost-efficient.[1] Weakly supervised annotation methods (e.g., based on identifying sentences in a corpus that are likely to express a data record) also include a significant amount of effort and often result in annotations that are low in fidelity between data records and the corresponding textual expression (Mintz et al., 2009). Training NLG models on such data can result in pathological outputs with missing information or hallucination (Dušek et al., 2019).

3 Model Adaptation

As a supervised task, data-to-text generation systems rely on previously observed examples to learn the correct generation for a predicate. What happens when the model encounters a new predicate? What about predicates from a new domain?

Previous work included separate evaluations for "unseen" predicates (Gardent et al., 2017). However, strategies to improve unseen predicates focused on data augmentation: finding new training

[1]Throughout the paper we use the term *low-resource domains* to refer to domains and applications for which a human annotated data-to-text dataset is not readily available. This includes domains such as finance and medicine.

Zero-shot Prompt

Translate Graph to English:

Graph: *<H>* Alan Martin (footballer) *<R>* CLUB *<T>* Hamilton Academical F.C.
English:

Zero-shot Prompt + Relation Description

Translate Graph to English:

Definition: club is an organization of players and managers associated with a particular football team.
Graph: *<H>* Alan Martin (footballer) *<R>* CLUB *<T>* Hamilton Academical F.C.
English:

Figure 1: A customized 0-shot prompt for GPT

Few-shot Prompt

Translate Graph to English:

Graph: *<H>* Paulo Sousa *<R>* CLUB *<T>* ACF Fiorentina
English: Paulo Sousa plays for ACF Fiorentina.
###
Graph: *<H>* Dave Challinor *<R>* CLUB *<T>* Colwyn Bay F.C.
English: Dave Challinor plays for Colwyn Bay F.C.
###
Graph: *<H>* Alan Martin (footballer) *<R>* CLUB *<T>* Hamilton Academical F.C.
English:

Figure 2: A customized 3-shot prompt for GPT

data that included the unseen predicates rather than different modeling techniques. Evaluation of different models will be our focus.

How should we conceptualize unseen predicates? Are these out of vocabulary tokens, where we could expect a model to generalize? For example, the new predication MANAGER can be informed by a seen predicate of COACH. Alternatively, unseen predicates may be a new task, e.g., the predicate CLINICAL_DIAGNOSIS when training data included only sports relations.

We study this problem using PLMs like GPT-2, which excel at adapting to new tasks. In contrast to "supervised" models like T5,[2] which expect task-specific training data, generative PLMs can obtain reasonable performance in a few shot setting. Therefore, we will evaluate their efficacy for data-to-text generation in a low-resource setting: unseen predicates.

While PLMs can be fine-tuned on new data, their increasing size and training requirements disfavors this approach. Instead, current work assumes a single PLM capable of performing multiple downstream tasks (Lester et al., 2021). We adopt GPT2-XL, a decoder-only Transformer (Vaswani et al., 2017) with 1.5B parameters pre-trained for language modeling (Radford et al., 2019).[3] We utilize GPT2-XL as a data-to-text generation model in various low-resource settings. Instead of fine-tuning the language model to predict a textual description given the input data record (Mager et al., 2020; Nan et al., 2020; Ribeiro et al., 2020), we investigate customized prompting and tuning GPT2-XL (Radford et al., 2019), which is better suited to applications for which little to no data is available.

3.1 Zero-shot Setting

We start by evaluating GPT2-XL in the zero-shot setting, an especially challenging setting due to a lack of coverage in the training data of pairings between structured records and unstructured text (Gong et al., 2020). Ribeiro et al. (2020) handled this by including an additional pretraining step. Our focus is on an off-the-shelf GPT2-XL model. We format the input data using the data-to-text generation infix and prefix formatting of Ribeiro et al. (2020) (example in Figure 1). We provide no additional context or task-specific training.

3.2 Few-shot Setting

We next consider a few-shot setting by augmenting the format of the zero-shot input with reference generations from the training corpus. We evaluate inputs with three examples (3-shot). See Figure 2 for an example. For predicates "seen" in the training set, we select at random three examples of the same predicate. For "unseen" predicates – not examples in the training set – we randomly select three examples. Other work has found that careful shot selection based on input text similarity can be beneficial (Liu et al., 2021a). However, it's less clear how this would apply to unseen predicates. We leave this for future work.

[2] We note that new findings (Sanh et al., 2021) has demonstrated T5 can handle zero-shot task adaptation with the right prompts; this is an evolving issue.

[3] WebText (the training dataset) includes content of more than 8 million documents with outbound links from Reddit, a social media platform. Wikipedia (the main data source for DART) is excluded.

3.3 Prompt Tuning

The expected task for a PLM is indicated by the choice of prompt; ours (Figure 1) follows prior work (Ribeiro et al., 2020; Nan et al., 2020). The prompt includes a prefix ("Graph") and infix token ("English") that indicate the start of the input and the start of the expected output. Auto-regressive language models are sensitive to the choice of prompt, and significant effort is needed to craft effective prompts (Liu et al., 2021b).

Lester et al. (2021) proposed an alternative method: prompt tuning. Instead of using discrete prompt tokens, "soft-prompts" are embeddings that are learned through back-propagation. We follow previous work (Lester et al., 2021; Chowdhury et al., 2022) and use a generic sequence of tokens to denote the prompt prefix $p_{1:s} = (p_1, p_2, ...p_s)$ and infix $q_{1:t} = (q_1, q_2, ...q_t)$. The model observes input $p_{1:s}$ <*H*> x_1 <*R*> x_2 <*T*> x_3 $q_{1:t}$, where x_1, x_2 and x_3 are strings from the example.

The objective during prompt-tuning is to maximize the probability of output sequence $y_{1:m}$ given input data record, prefix $p_{1:s}$, and infix $q_{1:t}$. During training however, only the embedding of the prompt tokens can be updated. Unlike fine-tuning which updates all model parameters on the target task, prompt tuning tunes a small number of parameters while keeping most of the language model fixed. Prompt tuning updates less than 0.01% of the model parameters, whereas other methods like prefix tuning (Li and Liang, 2021) update 0.1–1% of the model parameters. While this requires use of the full training set, as opposed to few shot training, it illuminates the abilities of GPT2-XL given access to such data.

3.4 Domain Knowledge

We explore another way of improving model performance in a low resource setting: providing definitions for predicates. In many domains, we may find a knowledge base containing many predicates, and definitions for those relations, but no examples of sentences expressing those relations. In these cases, we want to enhance the context of the PLM with predicate definitions. For examples, for the tuple *<H> Genuine Parts <R> DISTRIBUTOR <T> automotive and industrial replacement parts* we may know that DISTRIBUTOR means *"someone who markets merchandise"*. This may be helpful to a model that was never exposed to this predicate at training time.

We source predicate definitions for our data from WordNet, a lexical database in English (Miller, 1995), and WikiData.[4] We use WikiData since Wikipedia was the source of many relations in the DART data.[5]. An example of the input prompt enhanced with the "definition" appears in Figure 1. We also consider using predicate descriptions in combination with prompt tuning.

3.5 T5

We compare various settings of GPT2-XL with T5$_{\text{large}}$ (Raffel et al., 2019), a Transformer encoder-decoder architecture with 770M parameters for text generation. The model is pretrained with a denoising objective on a variety of NLP tasks using the web-extracted C4 corpus. Unlike a GPT style model, the denoising objective means an off-the-shelf model does poorly on unseen tasks, such as data-to-text generation (Raffel et al., 2019; Lester et al., 2021). Therefore, we follow Nan et al. (2020) and fine-tune T5$_{\text{large}}$ on the task-specific data. While this model requires a large amount of supervised examples, it attains state of the art performance on this task.

4 Dataset

For our experiments we use DART (Nan et al., 2020), the largest publicly available open-domain data-to-text generation corpus. DART relies on data from Wikipedia as well as two other commonly used data sets for this task: WebNLG (Gardent et al., 2017) and E2E (Novikova et al., 2017). Each instance includes a triple set (a set of one or more predicates and their labels) and a text snippet that expresses all relations in the triple set in natural language. We choose DART due to its size and wide coverage of predicate types. Relevant DART statistics appear in Table 1. We use the original train, development, and test splits for our experiments.[6]

Data Splits: The DART test set includes 5097 examples, of which 4826 (94.4%) include at least one relation type that appears in the training set. We represent this subset as the SEEN partition. The remaining 271 instances (5.3%) are considered UNSEEN. Note that the Nan et al. (2020) include an

[4]wikidata.org

[5]DART includes predicates such as *MARGIN_OF_VICTORY* and *INTERMEDIATE_(SOUTH)_WINNERS* Since descriptions for such relations cannot be found verbatim in WordNet or WikiData, no description is added to those cases.

[6]Nan et al. (2020) use version v1.0.0 of DART, whereas we use the publicly available version, v1.1.1.

	Train	Dev	Test
Size	30,526	2768	5097
#Unique relation types	4221	419	494
#Ref per example min/avg/max	1/2.0/48	1/2.5/33	1/2.4/35
#Triples per record min/avg/max	1/3.3/10	1/3.7/8	1/3.6/7

Table 1: Descriptive statistics of the DART version 1.1.1

evaluation on the "unseen" portion of WebNLG. However, in that case "unseen" means that the relations do not appear in the WebNLG training data, while they may still appear in the DART training data. Our splits ensure that the UNSEEN partition only contains predicates not seen during training.

To support additional system analysis, we create an additional partition of the test data: EASY and HARD. We determine whether an instance is HARD based on the similarity of the input relation to the reference text. In many cases the generation has high lexical overlap with the input data, while in other cases the generation is non-trivial. Examples of these in shared in Appendix A. To identify these easy and hard cases, we use BERTScore (Zhang et al., 2019) to measure the similarity of the input data records with respect to the reference. We rank the input/output pairs based on the computed BertScore (F1) and include the top 10% (510 examples) in the EASY partition and bottom 10% in the HARD partition.

5 Experimental Setup

Model Training Our experiments use the DART dataset with existing train/dev/test splits.[7] Following Harkous et al. (2020), we add special tokens <*H*>, <*R*>, and <*T*> before the head entity, the predicate and tail entity of each triple respectively. In our experiments, we use the pretrained models GPT2-XL and T5$_{\text{large}}$ released by Hugging Face (Wolf et al., 2019).

For the few-shot experiments, we use GPT-2 tokenizer to split input data records into special symbols and subword units. We use beam search with beam size of three for decoding. We apply light post-processing to the generated text to remove the input prompt from the newly generated tokens and truncate generated text at newline characters. We set maximum generated tokens to 100 and repetition penalty to 1.01 for our experiments.

For our prompt tuning experiments we train the GPT2-XL for auto-regressive language modeling on one NVIDIA V100 GPU with 32GB of memory, for a single epoch on DART train set with prefix and infix length of 8, respectively. We use the Adam optimizer (Kingma and Ba, 2014) with learning rate 0.1 and 100 warm up steps for the linear learning rate scheduler. We use a training batch size of 2, and accumulate the gradient for 32 steps before updating weights (effective batch size of 64). For decoding, we use the same parameters as the previous setting.

We use the scripts from Ribeiro et al. (2020) to finetune T5 on DART, using identical hyperparameter settings.[8] We use the Adam optimizer with an initial learning rate of 3e-5 and a linearly decreasing learning rate schedule. We fine-tune the model on four GPUs for a maximum of 100 epochs and stop training if the performance does not improve on the dev set for 15 epochs. We decode with beam search with beam size 3. Each epoch of training takes approximately 2 hours for each model.

Finally, we include a baseline system to benchmark performance of our machine learning models. In a "copy baseline" we simply copy the input text and remove the prefix tokens (<H>, <R>, <T>) as well as special characters (e.g., underscores) common in DART predicates. This method performs well for examples with high lexical overlap between triple set and reference generation.

Evaluation Metrics Following previous work, we use automated metrics such as BLEU (Papineni et al., 2002), METEOR (Denkowski and Lavie, 2014), TER (Snover et al., 2006), and chrF++ (Popović, 2015) for evaluating our generation results. In addition, we also report BERTScore (Zhang et al., 2019) and BLEURT (Sellam et al., 2020). These metrics go beyond surface form similarities and use contextual embeddings to measure semantic similarity between the generated and reference text.[9]

6 Experiments

We evaluate GPT2-XL with various input types and T5$_{\text{large}}$ to answer several empirical questions. First, how well does GPT2-XL perform on the data-

[7]In the DART dataset, some data records are paired with more than 30 references. Nan et al. (2020) do not report the number of references used for their experiments. However in their adaptation of Ribeiro et al's fine-tuning script (Ribeiro et al., 2020) they only use three references. We follow their methodology and only use up to three references per example.

[8]`https://github.com/UKPLab/plms-graph2text` (Apache 2.0 license)

[9]We use the evaluation scripts provided in the official WebNLG challenge: `https://github.com/WebNLG/GenerationEval` (MIT license)

ID	Model	BLEU ↑			METEOR ↑			TER ↓		
		SEEN	UNSEEN	ALL	SEEN	UNSEEN	ALL	SEEN	UNSEEN	ALL
1	copy baseline	4.48	5.07	4.50	0.28	0.31	0.28	0.92	0.86	0.92
2	GPT2-XL (0-shot)	13.13	13.88	13.26	0.23	0.27	0.23	0.69	0.78	0.70
3	GPT2-XL(3-shot)	26.74	23.72	26.65	0.29	0.28	0.29	0.85	0.78	0.84
4	GPT2-XL-PT	33.55	29.86	33.41	0.24	0.28	0.24	0.65	0.61	0.65
5	GPT2-XL-PT + Reranking	31.03	31.67	31.09	0.28	0.30	0.28	0.63	0.58	0.63
6	$T5_{large}$	48.41	43.48	48.25	0.39	0.40	0.39	0.46	0.44	0.46
+Descriptions										
7	GPT2-XL(0-shot)	11.45	8.05	11.4	0.20	0.19	0.20	0.70	1.00	0.72
8	GPT2-XL(3-shot)	26.32	21.30	26.14	0.28	0.27	0.28	0.83	0.89	0.83
9	GPT2-XL-PT	33.96	31.37	33.85	0.24	0.28	0.24	0.66	0.59	0.66
10	$T5_{large}$	48.56	43.82	48.4	0.39	0.39	0.39	0.46	0.45	0.46

Table 2: Model results on test set of the DART dataset. ↑: Higher is better. ↓: Lower is better.

ID	Model	BLEU ↑		METEOR ↑		chrF++ ↑		TER ↓		BERTScore(F1) ↑		BLEURT ↑	
		EASY	HARD	EASY	HARD	EASY	HARD	EASY	HARD	EASY	HARD	EASY	HARD
11	copy baseline	18.00	2.01	0.41	0.23	0.45	0.32	0.79	0.99	0.88	0,80	0.12	-1.00
12	GPT2-XL (0-shot)	22.20	6.92	0.34	0.18	0.47	0.31	0.83	0.64	0.90	0.88	-0.09	-0.54
13	GPT2-XL (3-shot)	34.97	1.88	0.34	0.06	0.54	0.07	0.82	0.38	0.92	0.93	-0.09	-0.11
14	GPT2-XL-PT	42.81	31.78	0.35	0.23	0.57	0.39	0.48	0.69	0.94	0.92	0.31	-0.17
15	GPT2-XL-PT + Reanking	43.35	25.79	0.37	0.29	0.60	0.48	0.47	0.66	0.94	0.93	0.34	-0.04
16	$T5_{large}$	70.54	38.34	0.51	0.35	0.80	0.57	0.23	0.59	0.97	0.94	0.70	0.20
+Descriptions													
17	GPT2-XL (0-shot)	19.00	6.43	0.30	0.17	0.42	0.31	0.93	0.65	0.89	0.88	-0.20	-0.54
18	GPT2-XL (3-shot)	34.19	20.54	0.38	0.26	0.61	0.44	0.92	0.81	0.93	0.91	0.07	-0.26
19	GPT2-XL-PT	42.52	33.1	0.34	0.23	0.56	0.39	0.5	0.69	0.93	0.91	0.28	-0.21
20	$T5_{large}$	70.06	38.49	0.51	0.34	0.80	0.57	0.23	0.60	0.97	0.94	0.69	0.20

Table 3: Model results on EASY and HARD partitions of the DART test set. ↑: Higher is better. ↓: Lower is better.

to-text task? Second, how well do GPT2-XL and $T5_{large}$ do on relations that does not appear in the training set? Third, can we improve GPT2-XL through the strategies proposed in §3?

6.1 Results

Table 2 presents model performance on the SEEN and UNSEEN partitions. For evaluation results based on chrF++, BERTScore, and BLEURT see Table 5 in the Appendix B. As expected, the copy baseline (row 1) does poorly across all conditions, but consistently in the SEEN and UNSEEN partitions. As reported previously (Nan et al., 2020), T5 (row 6) does well at this task. Performance drops significantly on the UNSEEN data because the model does not observe these predicates during training.

We now turn to GPT2-XL, which is evaluated on this task without any training data. Following previous work we find that GPT2-XL makes an effective zero-shot model, with results easily surpassing the copy baseline. Notably, GPT2-XL does similarly on either partition, since it was not trained on any task data. Examining the output more closely, we find that GPT2-XL mostly copies the input; while it outperforms the copy baseline, its strategy is largely the same. We include examples in Appendix C.

Task Prompting GPT2-XL with a 3-shot prompt (row 3) does much better than the 0-shot case. Differences between the SEEN and UNSEEN settings are mixed across metrics, despite the unseen prompts including unrelated predicates; the model still benefits from multiple shots even if they do not contain the same predicates. While few-shot prompting leads to a boost in BLEU and METEOR, the translation edit rate (TER) increases by 0.14 point. We conjecture that this is due to an increase in hallucinated content in this setting. We take a closer at these pathological behaviors in §7. Critically, the performance gap between T5, which is trained on thousands of examples, and GPT2-XL (0-shot), which is trained on non, is noticeably reduced with just three shots of in-context examples.

We next consider prompt tuning, which utilizes all of the available training data to tune prompts for GPT2-XL. In contrast to T5 training, which modifies all model parameters, prompt tuning adapts only a tiny fraction of the model's parameters ($< 0.01\%$). Despite this difference, we still see another gain in performance (row 4). Not surprisingly, utilizing the training data does better than using just a few examples in the prompt. Additionally, prompt tuning also hallucinates less, as evidenced by a lower TER score (0.65 vs 0.84 for ALL). The prompt tuned GPT2-XL achieves the highest BLEU score (29.86) on UNSEEN predicates in comparison to the other variations of prompting. Overall, it is clear that in resource limited settings, GPT2-XL can be improved with even a few training examples, and substantially improved with prompt tuning, despite keeping most of the model's parameters unchanged.

Predicate Descriptions We next turn to evaluating models with predicate descriptions. As described in §3.4, we augment each prompt with a description of the predicate. We evaluate this augmentation in the 0-shot (row 7), 3-shot (row 8) and prompt tuning (row 9) settings, as well as in T5 training (row 10). We observe very small improvements on the UNSEEN partition and only in cases where model parameters are updated (rows 9 and 10). We suspect that as descriptions are sourced from WordNet and WikiData, their format may not be helpful for this task or our predicates could be largely self-explanatory already. We conjecture that in the 0-shot setting, conditioning the generation on descriptions might distract the model from the head and tail entity. However, we suspect that specialized domains such as finance or medicine would benefit from added descriptions.

Adding predicate descriptions in the few-shot setting improves the BLEU score to 20.54 on the HARD partition (Table 3, row 18). For the prompt tuned GPT2-XL, BLEU score improves to 33.1 (row 19). However, we do not see any gains for 0-shot GPT or T5 (row 17 and 20). Overall, GPT2-XL benefits from predicate descriptions on examples where significant re-writing is needed, even when additionally prompt tuned. GPT2-XL with prompt tuning achieves competitive results with benchmark T5 on the HARD partition (33.1 vs 38.49 BLEU).

Generation Difficulty We now turn to a deeper analysis of the models and their behavior. Table 3 shows the performance of all models on the EASY and HARD partitions. All models have noticeably worse performance on HARD examples, where more abstraction is needed – the performance gaps are very large. For example, the BLEU gap between the two partitions for T5 (row 16) is similar to the gap between T5 and GPT2-XL 0-shot. The best performing model T5 (row 16), has a gap of 0.16 METEOR between the EASY and HARD partition, while the GPT2-XL prompt tuned (row 14) has the smallest difference in performance between the partitions. In terms of generalizing to new relation types and domains where more abstraction is needed, prompt tuning may be a better approach.[10]

It is clear that these models do well overall in their ability to copy the input, but do poorly when significant rewriting is required. In many domains, we may prefer models with more "interesting" rewrites, a task at which these models do not do well. On the other hand, DART is a mostly automatically derived dataset, with significant errors in some examples. These examples may pervade the HARD partition.

Reranking GPT2-XL prompt tuned is both parameter efficient and generalizes very well to new predicates. It also comes closest to the performance of the state-of-the-art fine-tuned T5$_{\text{large}}$. During manual evaluation, we observe that this model would often miss subject or object of the predicate in its generations (see §7 for details). We can mitigate this problem without additional model training through a reranking strategy to ensure that the selected generation contains all relevant information.

We first create multiple candidate generations by increasing beam size during decoding. Next, we compute the percentage of head and tail entities covered in the text. Finally, we pick the candidate with the highest score.[11] Row 5 and 15 show the results of reranking a GPT2-XL prompt tuned model. Reranking moderately improves performance on all partitions, and across all metrics except BLEU.

Training Curves Our experiments so far have focused on GPT2-XL, demonstrating how effective this model can be at utilizing small amounts of data to improve on this task. We now turn to T5 and ask a similar question: how much data does T5 require to do well on this task? Specifically, how many examples are required for T5 to exceed the performance of GPT2-XL with just three shots?

We fine-tune T5 on increasingly larger amounts of training data. We start off with an off-the-shelf T5 model with no additional training. We then vary

[10]Note that our goal here is not to beat the previous state-of-the-art but rather to make recommendations for adapting PLMs for low-resource data-to-text generation

[11]We use a beam size of 20 during decoding. Prior to measuring the entity coverage in the candidates, we normalize the text by lower casing and removing special characters.

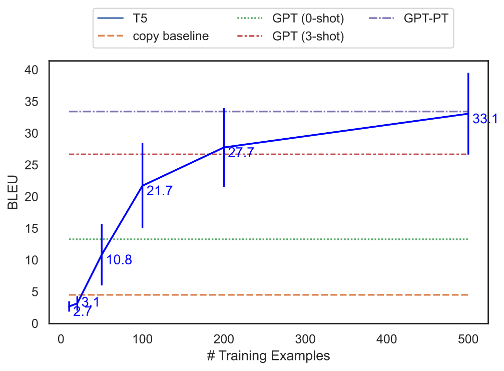

Figure 3: Impact of fine-tuning data size on performance of T5. Numbers reflect average performance over 5 different data samples, with standard error of the mean indicated by bars.

the number of training examples in {10, 20, 50, 100, 200, 500}.[12] We repeat each setting five times by resampling a training set and fine-tuning T5, and report results for each training set size averaged cross all test partitions. Figure 3 shows the BLEU performance (y-axis) of T5 as a function of number of training examples (x-axis). Performance of the copy baseline, 0-shot, 3-shot, and prompt tuned GPT2-XL are indicated by horizontal lines. Without any task-specific fine-tuning, T5 does slightly worse than the copy baseline, easily outperformed by 0-shot GPT2-XL. In settings without training data, GPT2-XL is the clear choice. T5 continues to lag behind GPT2-XL 3-shot until trained on at least 200 examples, and meets the performance of GPT2-XL prompt tuned after training on 500.

7 Error Analysis

To further examine the pathological behaviors of the models, we randomly sampled 50 examples from the DART test set for manual evaluation. For each example, the output of T5 and GPT2-XL in the 3-shot, prompt tuned, and reranked settings were presented to two annotators.[13] We also showed the reference text as another candidate, with the generating model identity hidden. Annotators evaluated output quality based on three criteria: (1) whether it contains hallucinated content *(hallucination)* (2) whether the text is missing information from the input records *(missing info)*, and (3) *fluency*. Annotators indicated agreement with each of these Likert items on an ordinal scale

[12]We use the same hyper-parameters as before except for the number of training epochs and batch size. To avoid overfitting on small data, we only fine-tune for 1 epoch. We use batch size of 2.

[13]Performed by two of the paper authors.

Source	Hallucination ↓	Missing Info ↓	Fluency ↑
Reference	1.53	1.19	4.51
GPT2-XL(3-shot)	3.26	3.61	3.17
GPT2-XL-PT	1.73	3.35	4.64
GPT2-XL-PT + Ranking	1.73	2.79	4.75
T5 $_{large}$	1.16	1.23	4.79
Agreement	0.64	0.77	0.50

Table 4: Results of the qualitative evaluation. ↓: Lower is better. ↑: Higher is better. Inter-annotator agreement is measured by Kendall's τ rank correlation coefficient.

from 1 (strongly disagree) to 5 (strongly agree).

Table 4 presents average annotator score according to each of these Likert items. GPT2-XL in the 3-shot setting often misses information. Notably, both variations of the prompt-tuned generate very fluent text. Reranking improves the quality of the generations by decreasing the amount of missing information and improving fluency. While the best GPT2-XL model does very similar to T5$_{large}$ in terms of fluency, on average it hallucinates or misses information more often.

8 Conclusion and Future Work

We systematically analyze the performance of a generative language model (GPT2-XL) for data-to-text generation in a low-resource setting by examining performance on unseen examples. Custom prompting and domain knowledge (predicate descriptions) can improve the performance of off-the-shelf GPT2-XL in a data- and parameter-efficient manner. We conduct experiments with varying training set sizes to make recommendations on a suitable approach for data-to-text generation depending on the amount of available training data.

When training data is unavailable, GPT2-XL (0-shot) is better than T5$_{large}$. With a small number of examples (3-shot), GPT2-XL outperforms T5$_{large}$ until at least 200 training examples are available. We also perform an error analysis and find that prompt tuned GPT2-XL generations can be improved by decreasing the incidence of missing information. We also find that the performance gap between easy and hard DART examples is massive for T5$_{large}$. These findings suggest that future work should consider more challenging examples, and should consider ways in which to generate larger variations for expressing a predicate type. This should include considerations of more challenging and disparate domains, such as finance or medicine. In these cases, we may see benefits from our proposed predicate descriptions, which did best in the low-resource, hard examples.

References

David L Chen and Raymond J Mooney. 2008. Learning to sportscast: a test of grounded language acquisition. In *Proceedings of the 25th international conference on Machine learning*, pages 128–135.

Zhiyu Chen, Harini Eavani, Wenhu Chen, Yinyin Liu, and William Yang Wang. 2019. Few-shot nlg with pre-trained language model. *arXiv preprint arXiv:1904.09521*.

Jishnu Ray Chowdhury, Yong Zhuang, and Shuyi Wang. 2022. Novelty Controlled Paraphrase Generation with Retrieval Augmented Conditional Prompt Tuning. In *Association for the Advancement of Artificial Intelligence (AAAI)*. AAAI.

Michael Denkowski and Alon Lavie. 2014. Meteor universal: Language specific translation evaluation for any target language. In *Proceedings of the ninth workshop on statistical machine translation*, pages 376–380.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Ondřej Dušek, David M Howcroft, and Verena Rieser. 2019. Semantic noise matters for neural natural language generation. *arXiv preprint arXiv:1911.03905*.

Angela Fan, Claire Gardent, Chloé Braud, and Antoine Bordes. 2019. Using local knowledge graph construction to scale seq2seq models to multi-document inputs. *arXiv preprint arXiv:1910.08435*.

Claire Gardent, 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.

Albert Gatt and Emiel Krahmer. 2018. Survey of the state of the art in natural language generation: Core tasks, applications and evaluation. *Journal of Artificial Intelligence Research*, 61:65–170.

Heng Gong, Yawei Sun, Xiaocheng Feng, Bing Qin, Wei Bi, Xiaojiang Liu, and Ting Liu. 2020. Tablegpt: Few-shot table-to-text generation with table structure reconstruction and content matching. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 1978–1988.

Hamza Harkous, Isabel Groves, and Amir Saffari. 2020. Have your text and use it too! end-to-end neural data-to-text generation with semantic fidelity. *arXiv preprint arXiv:2004.06577*.

Jeremy Howard and Sebastian Ruder. 2018. Universal language model fine-tuning for text classification. *arXiv preprint arXiv:1801.06146*.

Mihir Kale and Abhinav Rastogi. 2020. Text-to-text pre-training for data-to-text tasks. In *Proceedings of the 13th International Conference on Natural Language Generation*, pages 97–102, Dublin, Ireland. Association for Computational Linguistics.

Nitish Shirish Keskar, Bryan McCann, Lav R Varshney, Caiming Xiong, and Richard Socher. 2019. Ctrl: A conditional transformer language model for controllable generation. *arXiv preprint arXiv:1909.05858*.

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Leo Leppänen, Myriam Munezero, Mark Granroth-Wilding, and Hannu Toivonen. 2017. Data-driven news generation for automated journalism. In *Proceedings of the 10th International Conference on Natural Language Generation*, pages 188–197.

Brian Lester, Rami Al-Rfou, and Noah Constant. 2021. The power of scale for parameter-efficient prompt tuning. *arXiv preprint arXiv:2104.08691*.

Mike Lewis, 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*.

Xiang Lisa Li and Percy Liang. 2021. Prefix-tuning: Optimizing continuous prompts for generation. *arXiv preprint arXiv:2101.00190*.

Jiachang Liu, Dinghan Shen, Yizhe Zhang, Bill Dolan, Lawrence Carin, and Weizhu Chen. 2021a. What makes good in-context examples for gpt-3? *arXiv preprint arXiv:2101.06804*.

Xiao Liu, Yanan Zheng, Zhengxiao Du, Ming Ding, Yujie Qian, Zhilin Yang, and Jie Tang. 2021b. Gpt understands, too. *arXiv preprint arXiv:2103.10385*.

Yinhan Liu, 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. *arXiv preprint arXiv:1907.11692*.

Kaixin Ma, Hao Cheng, Xiaodong Liu, Eric Nyberg, and Jianfeng Gao. 2021. Open domain question answering over virtual documents: A unified approach for data and text. *arXiv preprint arXiv:2110.08417*.

Manuel Mager, Ramón Fernandez Astudillo, Tahira Naseem, Md Arafat Sultan, Young-Suk Lee, Radu Florian, and Salim Roukos. 2020. Gpt-too: A language-model-first approach for amr-to-text generation. *arXiv preprint arXiv:2005.09123*.

Hongyuan Mei, Mohit Bansal, and Matthew R Walter. 2015. What to talk about and how? selective generation using lstms with coarse-to-fine alignment. *arXiv preprint arXiv:1509.00838*.

George A Miller. 1995. Wordnet: a lexical database for english. *Communications of the ACM*, 38(11):39–41.

Mike Mintz, Steven Bills, Rion Snow, and Dan Jurafsky. 2009. Distant supervision for relation extraction without labeled data. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP*, pages 1003–1011.

Seungwhan Moon, Pararth Shah, Anuj Kumar, and Rajen Subba. 2019. Opendialkg: Explainable conversational reasoning with attention-based walks over knowledge graphs. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 845–854.

Linyong Nan, Dragomir Radev, Rui Zhang, Amrit Rau, Abhinand Sivaprasad, Chiachun Hsieh, Xiangru Tang, Aadit Vyas, Neha Verma, Pranav Krishna, et al. 2020. Dart: Open-domain structured data record to text generation. *arXiv preprint arXiv:2007.02871*.

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

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting of the Association for Computational Linguistics*, pages 311–318.

Vassilis Plachouras, Charese Smiley, Hiroko Bretz, Ola Taylor, Jochen L Leidner, Dezhao Song, and Frank Schilder. 2016. Interacting with financial data using natural language. In *Proceedings of the 39th International ACM SIGIR conference on Research and Development in Information Retrieval*, pages 1121–1124.

Maja Popović. 2015. chrf: character n-gram f-score for automatic mt evaluation. In *Proceedings of the Tenth Workshop on Statistical Machine Translation*, pages 392–395.

François Portet, Ehud Reiter, Albert Gatt, Jim Hunter, Somayajulu Sripada, Yvonne Freer, and Cindy Sykes. 2009. Automatic generation of textual summaries from neonatal intensive care data. *Artificial Intelligence*, 173(7-8):789–816.

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

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. 2019. Exploring the limits of transfer learning with a unified text-to-text transformer. *arXiv preprint arXiv:1910.10683*.

Alejandro Ramos-Soto, Alberto Jose Bugarin, Senén Barro, and Juan Taboada. 2014. Linguistic descriptions for automatic generation of textual short-term weather forecasts on real prediction data. *IEEE Transactions on Fuzzy Systems*, 23(1):44–57.

Leonardo FR Ribeiro, Martin Schmitt, Hinrich Schütze, and Iryna Gurevych. 2020. Investigating pretrained language models for graph-to-text generation. *arXiv preprint arXiv:2007.08426*.

Victor Sanh, Albert Webson, Colin Raffel, Stephen H Bach, Lintang Sutawika, Zaid Alyafeai, Antoine Chaffin, Arnaud Stiegler, Teven Le Scao, Arun Raja, et al. 2021. Multitask prompted training enables zero-shot task generalization. *arXiv preprint arXiv:2110.08207*.

Abigail See, Aneesh Pappu, Rohun Saxena, Akhila Yerukola, and Christopher D Manning. 2019. Do massively pretrained language models make better storytellers? *arXiv preprint arXiv:1909.10705*.

Thibault Sellam, Dipanjan Das, and Ankur P Parikh. 2020. Bleurt: Learning robust metrics for text generation. *arXiv preprint arXiv:2004.04696*.

Matthew Snover, Bonnie Dorr, Richard Schwartz, Linnea Micciulla, and John Makhoul. 2006. A study of translation edit rate with targeted human annotation. In *Proceedings of the 7th Conference of the Association for Machine Translation in the Americas: Technical Papers*, pages 223–231.

Chris van der Lee, Emiel Krahmer, and Sander Wubben. 2017. Pass: A dutch data-to-text system for soccer, targeted towards specific audiences. In *Proceedings of the 10th International Conference on Natural Language Generation*, pages 95–104.

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 M Shieber, and Alexander M Rush. 2017. Challenges in data-to-document generation. *arXiv preprint arXiv:1707.08052*.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, et al. 2019. Huggingface's transformers: State-of-the-art natural language processing. *arXiv preprint arXiv:1910.03771*.

Tianyi Zhang, Varsha Kishore, Felix Wu, Kilian Q Weinberger, and Yoav Artzi. 2019. Bertscore: Evaluating text generation with bert. *arXiv preprint arXiv:1904.09675*.

A Data Splits

Examples from the EASY and HARD partitions are shown in Figure 4. The copy baseline achieves good results on the EASY examples. On the other hand, the examples from the HARD partition are more abtractive – generating descriptions for these examples requires substantial rewriting. In several cases, the reference text has a low fidelity with respect to the input record. For example, when one or more triples in the input are not described in the reference text. This is a data quality issue and is a common occurrence in DART.

B Results

Experimental results on SEEN and UNSEEN partitions are presented in Table 5. As reported in § 6, T5 performs well on this task (row 6). The 0-shot GPT2-XL outperforms the copy baseline in terms of all metrics except for chrF++ (row 2). GPT2-XL with a 3-shot prompt does much better than the 0-shot case. Prompt tuning improves the results both in terms of BertScore and BLEURT (row 4). We see another gain in the performance by adding reranking (row 5). These trends are consistent with what we observed for BLEU, METEOR, and TER in Table 2.

We do not see a consistent performance drop going from SEEN to the UNSEEN partition when looking at chrF++, BertScore, and BLEURT. This is somewhat surprising, but also hard to interpret given that chrF++ relies on character n-gram and BertScore and BLEU rely in contextualized embeddings.

C Sample Model Output

In this section, we share a few samples from the DART test set as well as outputs generated by different models. We qualitatively compare different models and highlight a few of their common errors.

Task Prompting As seen in Examples 1 and 2, GPT2-XL in the 0-shot setting often copies the input. GPT2-XL with a 3-shot prompt generates a much more fluent text than the 0-shot case. This can be seen in Examples 2, 4, and 5. Although GPT2-XL with few-shot prompting generates more fluent text, it often generates hallucinated content (see Example 3).

We see that prompt tuning further boosts our performance and generates a more coherent text in comparison to few-shot GPT2-XL (see Example 1 and 3). Moreover, it hallucinates much less than the few-shot setting (e.g. see Example 3). We also saw this previously in Table 2, as the prompt tuned GPT2-XL achieved lower TER score. In contrast to T5 training, in which all model parameters are updated, prompt tuning adapts only a small fraction of the model parameters. However, in many cases the generated text is as good as the benchmark T5 (see Example 2). Despite generating very fluent text, prompt tuned GPT2-XL often misses information from one or more relations (Examples 1, 3, and 4).

Reranking Reranking based on entity coverage solves the missing information issue in several cases. For example, in Example 3, the entity *Alvis Speed 25* which is missed by the prompt tuned GPT2-XL, is covered after reranking. The benefit of reranking also can be seen in Example 4. On the other hand, in Example 2, ranking does not solve the missing information issue. This is because argument "yes" of "family-friendly" probably would not naturally appear in generated text (e.g., "Yes, this is a family-friendly restaurant"). For such cases, the reranking heuristic will not provide useful feedback.

Predicate Descriptions As mentioned in Section 6.1, in several cases, the description extracted from WordNet and WikiData are trivial. In Example 2, the definition of relations *food*, *area*, and *near* add no information beyond the word itself, and therefore not helpful for the model. On the other hand, it seems like defining relation *MANUFACTURER* in Example 3 has improved generations of GPT2-XL in both the few-shot and prompt-tuned settings. In some cases, while the predicate description can be potentially useful, the model ignores the augmented description. For example, in 4, the definition of relation *GENRE* is not covered in the generated text of any of models.

ID	Model	chrF++ ↑			BERTScore(F1) ↑			BLEURT ↑		
		SEEN	UNSEEN	ALL	SEEN	UNSEEN	ALL	SEEN	UNSEEN	ALL
1	copy baseline	0.33	0.34	0.33	0.83	0.85	0.83	-0.59	-0.29	-0.58
2	GPT2-XL (0-shot)	0.34	0.34	0.34	0.88	0.87	0.88	-0.46	-0.30	-0.46
3	GPT2-XL (3-shot)	0.48	0.44	0.48	0.91	0.91	0.91	-0.19	-0.17	-0.19
4	GPT2-XL-PT	0.40	0.44	0.40	0.92	0.92	0.92	-0.11	0.06	-0.10
5	GPT2-XL-PT + Reranking	0.46	0.47	0.46	0.92	0.92	0.92	-0.01	0.12	0.00
6	T5$_{\text{large}}$	0.64	0.64	0.64	0.95	0.95	0.95	0.38	0.44	0.39
	+ Description									
7	GPT2-XL (0-shot)	0.31	0.23	0.30	0.88	0.86	0.88	-0.46	-0.54	-0.46
8	GPT2-XL (3-shot)	0.47	0.42	0.46	0.91	0.90	0.91	-0.19	-0.16	-0.19
9	GPT2-XL-PT	0.39	0.45	0.39	0.91	0.92	0.91	-0.14	0.09	-0.13
10	T5$_{\text{large}}$	0.64	0.63	0.64	0.95	0.95	0.95	0.38	0.43	0.38

Table 5: Performance on the DART test set, partitioned by whether predicates are SEEN, UNSEEN, and overall. ↑: Higher is better.

Easy **Examples**

Input: <H> Adolfo Suárez Madrid–Barajas Airport <R> LOCATION <T> Madrid, Paracuellos de Jarama, San Sebastián de los Reyes and Alcobendas

Reference: Adolfo Suárez Madrid–Barajas Airport can be found in Madrid, Paracuellos de Jarama, San Sebastián de los Reyes and Alcobendas.'

###

Input: <H> Alaa Abdul-Zahra <R> CLUB <T> Sanat Mes Kerman F.C.

Reference: Alaa Abdul-Zahra's club is Sanat Mes Kerman F.C.

###

Input: <H> Alderney Airport <R> RUNWAY_NAME <T> "14/32"

Reference: Alderney Airport runway name is 14/32

###

Input: <H> Asunción <R> IS_PART_OF <T> Gran Asunción

Reference: Asunción is a part of Gran Asunción.

###

Input: <H> Airey Neave <R> AWARD <T> Military Cross

Reference: Airey Neave was awarded the Military Cross.

Hard **Examples**

Input: <H> 2004 <R> MOVEMENTS <T> Promotion Playoffs - Promoted <H> 2004 <R> POSITION <T> 1st

Reference: Sports stats for Ljungskile SK

###

Input: <H> Khokhan Sen <R> MATCHES <T> 14 <H> Khokhan Sen <R> INNINGS <T> 21 <H> Khokhan Sen <R> RANK <T> 9 <H> Khokhan Sen <R> CAUGHT <T> 20 <H> Khokhan Sen <R> STUMPED <T> 11 <H> Khokhan Sen <R> DISMISSALS <T> 31

Reference: The innings when caught was 20 was 21

###

Input: <H> thierry morin <R> POSITION <T> defender <H> [TABLECONTEXT] <R> NAME <T> thierry morin <H> [TABLECONTEXT] <R> [TITLE] <T> Players

Reference: Thierry Morin was a defender for Paris Saint-Germain.

###

Input: <H> ALV X-1 <R> COUNTRY_ORIGIN <T> United States <H> United States <R> ETHNIC_GROUP <T> African Americans <H> United States <R> DEMONYM <T> Americans

Reference: Originating in the United States and by Americans, some of African decent is the ALVX-1.', 'ALVX-1 comes from the US where Americans live and African Americans are an ethnic group

###

Input: <H> past tense <R> SEASON_# <T> 4 <H> past tense <R> ORIGINAL_AIR_DATE <T> october29,2008 <H> past tense <R> NO._IN_SERIES <T> 13 <H> past tense <R> U.S._VIEWERS_(MILLIONS) <T> 7.93 <H> past tense <R> DIRECTED_BY <T> michael pressman <H> past tense <R> WRITTEN_BY <T> craig turk

Reference: Past Tense was the 13th episode in the series.

Figure 4: Examples from the EASY and HARD partition

Example #1

Input: <H> Aleksey Chirikov (icebreaker) <R> SHIP_BEAM <T> 21.2 <H> Aleksey Chirikov (icebreaker) <R> STATUS <T> ""In service"" <H> Aleksey Chirikov (icebreaker) <R> BUILDER <T> Helsinki <H> Aleksey Chirikov (icebreaker) <R> BUILDER <T> Arctech Helsinki Shipyard

Reference: The icebreaker Aleksey Chirikov was built at the Arctech Helsinki Shipyard in Helsinki. The Aleksey Chirikov (icebreaker) is in service and its ship beam is 21.2

Description: BUILDER means a person who constructs something

GPT2-XL (0-shot): <H> Aleksey Chirikov (icebreaker) <R> SHIP_BEAM <T> 21.2 <H> Aleksey Chirikov (icebreaker) <R> STATUS <T> "In service" <H> Aleksey Chirikov (icebreaker) <R> BUILDER <T> Helsinki <H> Aleksey Chirikov (icebreaker) <R> BUILDER <T>

GPT2-XL (3-shot): Aleksey Chirikov (icebreaker) is of type 21.2.

GPT2-XL (3-shot) + Description: Aleksey Chirikov (icebreaker) was a Russian icebreaker.

GPT2-XL-PT: Aleksey Chirikov (icebreaker) was built by Arctech Helsinki Shipyard.

GPT2-XL-PT + Description: Aleksey Chirikov (icebreaker) was built by Arctech Helsinki Shipyard.

GPT2-XL-PT + Reranking: Aleksey Chirikov (icebreaker) is in service at Arctech Helsinki Shipyard.

T5$_{\text{large}}$: The ship Aleksey Chirikov was built by Arctech, Helsinki Shipyard and has a ship beam of 21.2 metres. It is currently in service.

Example #2

Input: <H> The Mill <R> eatType <T> pub <H> The Mill <R> food <T> Fast food <H> The Mill <R> area <T> city centre <H> The Mill <R> familyFriendly <T> no <H> The Mill <R> near <T> Café Sicilia

Reference: The Mill pub is near Café Sicilia in the city centre that isn't family friendly, is average and has Fast food.

Description: eatType means type of food served by a restaurant or restaurant chain. food means any substance that can be metabolized by an animal to give energy and build tissue. area means area occupied by an object. near means move towards.

GPT2-XL (0-shot): Graph: <H> The Mill <R> eatType <T> pub <H> The Mill <R> food <T> Fast food <H> The Mill <R> area <T> city centre <H> The Mill <R> familyFriendly <T> no <H> The Mill <R> near <T> Café Sicilia
English:

GPT2-XL (3-shot): The Mill is located in the city centre near Café Sicilia. It serves fast food in the price range of £20-25 with a moderate customer rating

GPT2-XL (3-shot) + Description: The fast food restaurant, The Mill, has an average rating of 3 out of 5 perhaps as it is not friendly to kids. The Mill is located in the city centre area near Café Sicilia.

GPT2-XL-PT: The Mill is a fast food pub in the city centre near Café Sicilia.

GPT2-XL-PT + Description: The Mill is a family friendly pub near Cafe Sicilia.

GPT2-XL-PT + Reranking: The Mill is a fast food pub in the city centre near Café Sicilia.

T5$_{\text{large}}$: The Mill is a fast food pub in the city centre near Café Sicilia.

Example #3

Input: <H> Alvis Speed 25 <R> MANUFACTURER <T> Alvis Car and Engineering Company <H> Alvis Car and Engineering Company <R> LOCATION_CITY <T> Coventry

Reference: Alvis Speed 25 was made by the Alvis Car and Engineering Company of Coventry.

Description: MANUFACTURER means manufacturer or producer of this product.

GPT2-XL (0-shot): <H> Alvis Speed 25 <R> MANUFACTURER <T> Alvis Car and Engineering Company <H> Alvis Car and Engineering Company <R> LOCATION_CITY <T> Coventry

GPT2-XL (3-shot): Alvis Speed 25 won the race in Coventry on April 25, 1996.

GPT2-XL (3-shot) + Description: Alvis Speed 25 is a car manufactured by Alvis Car and Engineering Company in Coventry, England

GPT2-XL-PT: Alvis Car and Engineering Company is located in Coventry.

GPT2-XL-PT + Description: Alvis Speed 25 is manufactured in Coventry.

GPT2-XL-PT + Reranking: Alvis Speed 25 is manufactured by Alvis Car and Engineering Company in Coventry.

T5$_{\text{large}}$: Alvis Car and Engineering Company is located in Coventry and manufactures the Alvis Speed 25.

Example #4

Input: <H> Anders Osborne <R> ASSOCIATED_BAND_ASSOCIATED_MUSICAL_ARTIST <T> Galactic <H> Anders Osborne <R> ASSOCIATED_BAND_ASSOCIATED_MUSICAL_ARTIST <T> Tab Benoit <H> Anders Osborne <R> GENRE <T> Blues <H> Anders Osborne <R> ASSOCIATED_BAND_ASSOCIATED_MUSICAL_ARTIST <T> Billy Iuso <H> Anders Osborne <R> ACTIVE_YEARS_START_YEAR <T> 1989

Reference: Anders Osborne musical career began in 1989. His career began in the blue's genre. He has been associated with the band, Galactic as well as Tab Benoit. Osborne also has ties to fellow musician Billy Iuso.

Description: GENRE means creative work's genre or an artist's field of work

GPT2-XL (0-shot): <H> Anders Osborne <R> ASSOCIATED_BAND_ASSOCIATED_MUSICAL_ARTIST <T> Galactic <H> Anders Osborne <R> ASSOCIATED_BAND_ASSOCIATED_MUSICAL_ARTIST <T> Tab Benoit <H> Anders Osborne <R> GENRE <T> Blues <H> Anders Osborne <R> ASSOCIATED_BAND_ASSOCIATED_M

GPT2-XL (3-shot): Anders Osborne is an English singer, songwriter and guitarist.

GPT2-XL (3-shot) + Description: Anders Osborne is an active musician and composer.

GPT2-XL-PT: Tab Benoit and Anders Osborne are associated with Galactic.

GPT2-XL-PT + Description: Tab Benoit and Anders Osborne are associated with Galactic.

GPT2-XL-PT + Reranking: Tab Benoit is associated with Galactic and Anders Osborne is associated with Billy Iuso.

T5$_{\text{large}}$: Anders Osborne is associated with the band Galactic, Billy Iuso, Tab Benoit and the genre of Blues. His active years start in 1989.

Example #5

Input: <H> Nurburgring <R> FASTEST_LAP <T> PSV Eindhoven <H> Nurburgring <R> WINNING_CLUB <T> A.C. Milan

Reference: PSV Eindhoven had the fastest lab on the Nurburgring, and A.C. Milan was the winning club at that race.

Description: FASTEST_LAP means person who does the fastest lap during a motorsport race.

GPT2-XL (0-shot): <H> Nurburgring <R> FASTEST_LAP <T> PSV Eindhoven <H> Nurburgring <R> WINNING_CLUB <T> A.C. Milan

GPT2-XL (3-shot): A.C. Milan had the fastest lap in the race won by PSV Eindhoven.

GPT2-XL (3-shot) + Description: A.C. Milan set the fastest lap at the Nurburgring when the winning team was PSV Eindhoven.

GPT2-XL-PT: Nurburgring is the fastest lap at A.C. Milan.

GPT2-XL-PT + Description: PSV Eindhoven is the winning club at the Nurburgring.

GPT2-XL-PT + Reranking: Nurburgring is the fastest lap at A.C. Milan

T5$_{\text{large}}$: A.C. Milan won the race where PSV Eindhoven had the fastest lap.