Incorporating Task-Specific Concept Knowledge into Script Learning

Chenkai Sun, Tie Xu, ChengXiang Zhai, Heng ji
University of Illinois at Urbana-Champaign, IL, USA
Alibaba, Hangzhou, China
{chenkai5, czhai, hengji}@illinois.edu
xutie.xt@alibaba-inc.com

Abstract

In this paper, we present TETRIS, a new task of Goal-Oriented Script Completion. Unlike previous work, it considers a more realistic and general setting, where the input includes not only the goal but also additional user context, including preferences and history. To address this problem, we propose a novel approach, which uses two techniques to improve performance: (1) concept prompting, and (2) script-oriented contrastive learning that addresses step repetition and hallucination problems. On our WikiHow-based dataset, we find that both methods improve performance.

1 Introduction

A Goal-Oriented Script refers to a sequence of events that describe some stereotypical activities for achieving a specified goal (Feigenbaum et al., 1981; Lyu et al., 2021). It is important to study how to automatically construct instructional scripts because it enables many high-impact applications such as robot action planning (Conti et al., 2020; Mohanan and Salgoankar, 2018), causal reasoning (Guo et al., 2020), and task-oriented dialogue generation (Chen et al., 2017). While previous work has shown that neural models are capable of constructing the entire script given a goal (Lyu et al., 2021; Sakaguchi et al., 2021), their proposed tasks are highly restrictive and based on overly simplified assumptions about the application because it ignored both the usage context (e.g., what is a preferred way of solving the problem, and what steps have been executed already) and the variable personal preferences that a user may have (e.g., a goal might need be achieved in different ways). Take "Make Eggless Cupcakes" as an example, the user might want to make it with different equipment (e.g., by using a rice cooker instead of an oven) or styles (e.g., to have some banana flavor), or the user has already completed some steps but would like to seek information on what to proceed next; a machine learning model trained on goal-to-script tasks as done in the existing work cannot adaptively provide solutions under such situations.

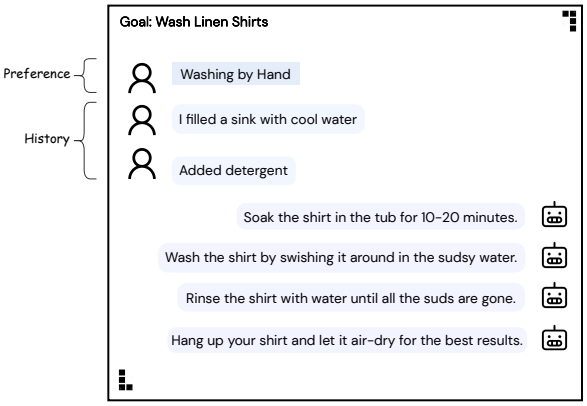

Figure 1: An example illustrating the task. The input consists of a specified goal (e.g., wash line shirts), an optional preference (e.g., washing by hand), and a history of steps. The model is asked to generate the remaining steps to achieve the goal.

We address the limitations of the previous task formulation and propose TETRIS, a more general task of goal-oriented script learning that allows flexible usage scenarios, and therefore it accommodates more realistic needs in downstream applications. More specifically, as shown in Figure 1, the task considers both user preference and history as input in addition to the goal. From a probabilistic perspective, instead of modeling the problem with $p(\text{Script}|\mathcal{G})$, we model it as

$$p(\text{Completion of Script}|\mathcal{H}, \mathcal{P}, \mathcal{G}) = \prod_i p(\text{Script}_i|\text{Script}_{<i}, \mathcal{P}, \mathcal{G})$$

where i indicates each step index in the script, $\mathcal{P}$ denotes user preference, $\mathcal{H}$ indicates History, and $\mathcal{G}$ abbreviates Goal.

To solve the new problem, we can use a com-

https://github.com/chenkaisun/Tetris

Proceedings of the 17th Conference of the European Chapter of the Association for Computational Linguistics, pages 3026–3040
May 2-6, 2023

monly used baseline method, i.e., using a seq2seq model (Lewis et al., 2019; Raffel et al., 2019), but a direct application of such a model would not be optimal because the model's understanding of the context of the goal is inevitably shallow and thus unlikely be able to fully exploit the extra context information we have in the problem setup. To address this limitation, we propose to enrich the representation with additional task-specific concepts so as to enable the model to understand the context more deeply and leverage relevant knowledge about those concepts encoded in the training data the model has been exposed to. To obtain task-specific knowledge needed for solving script learning, we introduce TASK CONCEPT DICTIONARY (TCD), a novel Key-Value Knowledge Base (KB) that consists of task phrases as the key and the associated concepts involved in the solution process for each task as the value. The concepts contain not only items needed for the task but also concerned attributes (e.g., "thickness") and intermediate products during the process (e.g., "dough" appears during Making Rice Noodles). In our work, as a proof of concept, we automatically construct the knowledge base from WikiHow[1], one of the most used and actively updated How-to websites.

Once we construct a TCD, it can be potentially used for solving the problem of script completion in many ways. In this paper, we focus on exploring how to leverage it to acquire task-specific relevant concepts for enriching the task representation used in a baseline model for script completion. Specifically, inspired by how humans use mind-map to construct solutions, we introduce a novel "concept prompting" framework that first acquires relevant concepts from TCD and then connects them with the input as a prompt for the language generator to complete the script (Figure 2). We further introduce two complementary ways of concept acquisition. The first method retrieves concepts of close neighbors from TCD. The second method involves a generative model pretrained on TCD that is capable of generating associated concepts for a given goal, which can generalize beyond similar cases. Discovering that the model often repeats historical steps and hallucinates actions and entities, we also developed a script-oriented contrastive learning approach to address these issues by constructing corresponding negative samples.

We perform experiments on a dataset constructed from WikiHow and find that our method gives a consistent improvement on both automatic and human evaluations in comparison to the baseline, demonstrating the benefit of TCD for script completion. Moreover, we find that concept acquisition quality has a large impact on this task.

To summarize, we make the following contributions:

- We introduce TETRIS, the task of Goal-Oriented Script Completion that asks a model to complete the task based on a goal, and optionally preference and history. The task poses interesting new challenges and enables new applications.
- We propose TASK CONCEPT DICTIONARY, a novel task-specific knowledge base that encodes knowledge about the associations between goals and concepts and a method composed of (1) extraction and integration of the relevant concepts from TCD, and (2) script learning-oriented contrastive learning objective to enhance the correctness of model-produced scripts.
- We experiment with the proposed methods on the WikiHow dataset and show that our methods outperform the state-of-the-art baseline models consistently, demonstrating the need for task-specific knowledge and the benefit of TCD for improving the performance of script completion.

2 Task Formulation

We define a Goal-Oriented Script to include three types of elements, **Goal**, **Preference**, and **Step**. A **Goal** represents the task desired to be completed. An optional **Preference** defines in what ways the goal should be completed. A **Step** is one procedural event toward solving the task, conditioned on a preference. In this work, given a goal $\mathcal{G}$, a preference $\mathcal{P}$, and a history of steps $\mathcal{H}$ already done, we aim to generate the remaining steps that accomplish the goal in natural language. An example is shown in Figure 1. For convenience, we use $\mathcal{G}_p$ to denote the goal conditioned on $\mathcal{P}$, and $\mathcal{I}$ to denote the entire input (consisting of $\mathcal{G}$, $\mathcal{P}$, and $\mathcal{H}$).

3 Task Concept Dictionary

As in virtually all NLP applications, a main technical challenge in solving the problem of script

[1] www.wikihow.com

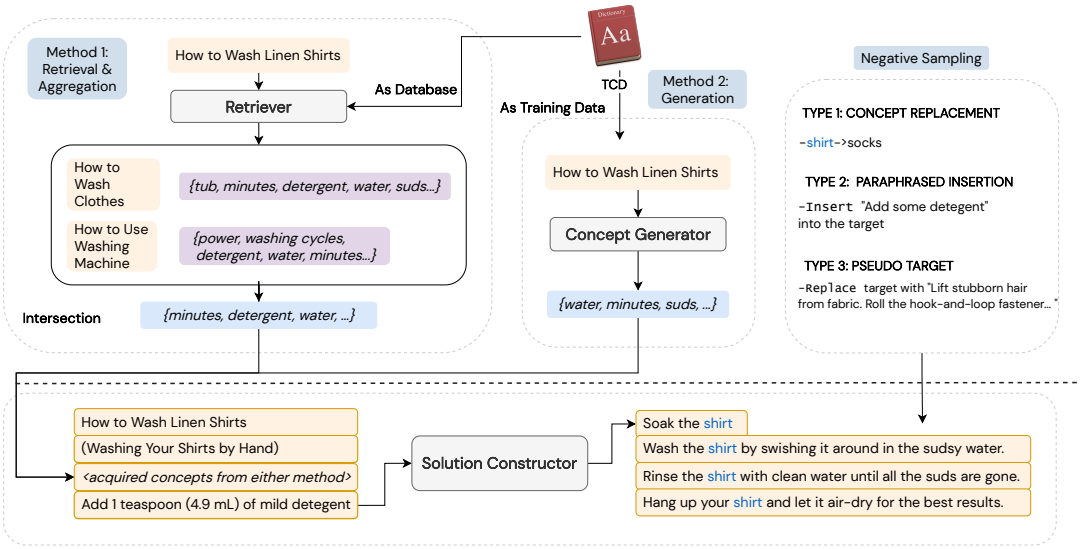

Figure 2: Our goal-oriented script completion framework. The top part shows the methods for automatically acquiring relevant concepts from the Task Concept Dictionary and negative sampling strategies. The bottom part shows the solution constructor, which uses the acquired concepts as an informative prompt to complete the script.

completion is how to acquire the relevant knowledge and leverage the knowledge to generate the remaining steps in a script. In our case, a critical question is: What kind of knowledge representation is likely helpful for solving the script completion problem? In our work, we make a hypothesis that creating task-concept association would help the model generalize better on script completion and propose Task Concept Dictionary (TCD). In TCD, each task key $\mathcal{G}$ from the collection of the keys $\mathcal{D}^G$ (e.g., "How to Modify the Navigation System of an Acura") is mapped to a set of relevant concepts $\mathcal{D}_C^{\mathcal{G}}$ during its solution process. Each $c \in \mathcal{D}_C^{\mathcal{G}}$ can be associated with $\mathcal{D}^{\mathcal{G}}$ in four types of roles of a concept in that task: as preparation material needed for achieving the task (like ingredients for recipe), as attributes/aspects that need to be considered, as intermediate entities generated during the process, and others. In this work, we made a primal construction of TCD (TCD.v0) as a proof of concept, that is, the association types are not considered between $\mathcal{G}$ and $\mathcal{D}_C^{\mathcal{G}}$.

The design of TCD is motivated by the observation that humans can often form solutions through brainstorming and mind-mapping with knowledge about the tasks and relevant concepts involved in the solution process. Analogously, TCD is meant to encode such task-specific knowledge about associations between goals and concepts so as to assist downstream script learning applications. As will be shown in our experiment results, TCD is indeed beneficial for improving performance for script completion by supplying useful relevant concepts to a baseline model.

In general, TCD can be constructed by using any instructional content on the Web. In our experiments, to construct TCD.v0, we first collect all articles from WikiHow. We chose to use WikiHow as the seed corpus for its richness in user preference data. Each article consists of a goal, a list of preferences, and solution steps under each preference. Let $\mathcal{G}_i^{p_j}$ be the goal from an article i conditioned on the j-th preference, we then proceed to augment the set $\mathcal{D}^G$ with a key $\mathcal{G}_i^{p_j}$, and augment $\mathcal{D}_C^{\mathcal{G}_i^{p_j}}$ with noun phrases (by using Spacy[2] tagger for extraction) in the solution steps under preference j. We create an association between goals and concepts if they co-occur in the same article. We show the statistics of TCD in Appendix A.1.

4 Method

In this section, we discuss our design of the method for TETRIS. The framework contains two parts, concept acquisition from TCD and solution construction. The first part uses TCD as an aid to acquire concepts relevant to the current input. Specifically, we introduce two different methods (Figure 2), one based on retrieval of closely related tasks and aggregation on the associated concepts, while the other involves training a concept generator on TCD. As shown in Section 5, both of

[2] https://spacy.io/

them have a positive impact on performance, yet they also introduce different benefits. The concepts fetched from TCD using either method form a concept prompt, which is combined with the input $\mathcal{I}$ (consisting of goal, preference, and history) as an augmented input $\mathcal{I}^*$. $\mathcal{I}^*$ is then fed into an encoder-decoder language model $\mathcal{M}_{\text{EncDec}}$ to complete the script. Below we describe the two methods for the generation of relevant concepts using TCD.

4.1 Concept Retrieval & Aggregation

The intuition of the retrieval method comes from observing how human refers to their past knowledge or the web for similar instructional sources (even though they might not address the need exactly) to achieve the current goal. For instance, if someone has experience in baking cheesecake and chocolate cake, it would be a breeze for them to make a strawberry cake by adapting the knowledge from their past experience. Building on the intuition, we propose to use retrieval as an interpretable way to retrieve similar tasks from the set of keys $\mathcal{D}^G$ in TCD and use their associated concepts to aid the downstream script completion task.

Retrieval. First we encode $\mathcal{G}_p$ (i.e., the preference-conditioned goal in $\mathcal{I}$) into a dense vector $\mathbf{e}_\text{g}$ with an encoder model. We similarly encode $\mathcal{D}^G$ into $\{\mathbf{e}_j\}_{j=1}^{|\mathcal{D}^G|}$. A cosine similarity score s_{gj} is computed between $\mathbf{e}_\text{g}$ and each $\mathbf{e}_\text{j}$. The top-K related tasks $\mathcal{N}^K$ are then retrieved based on the scores. We further obtain a set of concepts $\mathcal{C}_i$ for i-th task $\mathcal{N}^K$. In our experiment, we use SBERT (Reimers and Gurevych, 2019) (pretrained for semantic search) as the encoder. We use FAISS (Johnson et al., 2019), an efficient similarity search library, to perform top-k search.

Aggregation. The retrieved neighborhood of concepts, however, may sometimes introduce contextual noise (e.g., the grape and chocolate-related concepts are not useful in strawberry cake baking). To tackle this issue, we additionally perform operations on the retrieved concepts using the set intersection. The concept set for an input $\mathcal{I}$ is computed as $\mathcal{C}^s = \bigcap_{i=1}^{i=K} \mathcal{C}_i$. We finally map the set $\mathcal{C}^s$ into a list $\mathcal{C}$, neglecting ordering information.

4.2 Concept Generation

While the retrieval method allows an explicit concept acquisition process, it is limited by the width of TCD. For example, if one would like to complete a task from a domain that is not covered well by TCD, the retrieved neighbors can introduce more noise than help. To address this limitation of the retrieval method and enhance generalization, we propose to use language modeling as an alternative way of acquiring concepts. Specifically, inspired by the work (Bosselut et al., 2019), in which a concept is generated given an edge type, we propose to directly generate a set of concepts relevant to a given task and preference. We directly train the model on TCD, where the model is asked to generate the set $\mathcal{D}_C^t$ given each key t. In the inference stage, by feeding $\mathcal{G}_p$ into the trained concept generator, we can then obtain a list of concepts $\mathcal{C}$.

	F&E	F&B	C&V
# Train Samples	10600	2824	2214
# Dev Samples	3449	873	741
# Test Samples	3567	930	714
# Articles	2201	802	481
Avg # Tokens/Step	8.66	7.25	9.31
Avg # Steps/Article	10.42	8.66	10.41

Table 1: Statistics summarizing the WikiHow-based dataset for TETRIS, where F&E indicates Food and Entertainment, F&B indicates Finance and Business, and C&V indicates Cars and Other Vehicles

4.3 Solution Constructor

In this step we aim to encode both of the information from $\mathcal{C}$ (using either of the above methods) and $\mathcal{I}$ to generate the remaining steps that accomplish the goal specified in $\mathcal{I}$. The TCD has enabled us to enrich the representation by augmenting $\mathcal{I}$ with $\mathcal{C}$ and such an augmented representation can then be fed into any baseline script completion method to enable the baseline method to have (indirect) access to the knowledge encoded in TCD, thus improving accuracy of script completion.

Input Formation. An encoder-decoder model $\mathcal{M}_g$ is used for generation. In our experiment, we choose BART (Lewis et al., 2019) as the base model for its impressive performance on other NLP tasks (Liu et al., 2021; Lewis et al., 2020). Clearly, the framework accommodates any other models as well. Given the list of concepts $\mathcal{C}$ generated from either Section 4.1 or 4.2, we form the input $\mathcal{I}_f$ to the encoder as "<s>Goal (Preference) ### $\mathcal{C}$ ### </s>Step$_1$</s>Step$_2$...Step$_{|\mathcal{H}|}$</s>", where <s> is a special token to represent the start of the sentence and </s> is for separation.

Model Generation. The input is fed through the standard tokenization, embedding mapping, and transformer to produce encoder hidden states. In

Method	BERTScore	BARTScore	BLEU 1	BLEU 2	BLEU 3	BLEU 4	METEOR	ROUGE 2
BART	86.76	-4.65	8.45	4.40	2.21	1.13	14.67	4.6
GPT2	86.75	-4.59	17.89	8.29	3.44	1.52	15.61	3.1
CRA-3	87.18	-4.41	22.59	10.95	4.86	2.10	18.07	4.9
CRA-1	86.74	-4.50	21.87	10.82	4.85	2.21	17.55	5.3
CG	86.86	-4.45	24.34	11.84	5.26	2.43	17.88	5.1
CG+SOCL	86.77	-4.47	**26.18**	**13.15**	**6.18**	**3.07**	18.11	**5.7**
CRA-2	**87.21**	**-4.37**	23.95	11.36	5.06	2.35	**18.65**	4.9
CRA-2+SOCL	87.19	-4.38	24.25	11.58	5.25	2.42	18.36	5.0

Method	BERTScore	BARTScore	BLEU 1	BLEU 2	BLEU 3	BLEU 4	METEOR	ROUGE 2
BART	87.87	-4.37	16.61	8.31	4.67	2.92	18.92	5.6
GPT2	88.06	-4.30	17.13	7.27	3.58	2.10	18.61	3.6
CRA-3	88.37	-4.28	14.73	6.95	3.77	2.36	19.17	4.6
CRA-1	87.66	-4.37	17.68	8.57	4.63	2.87	18.95	5.4
CG	88.35	**-4.17**	22.87	10.97	6.09	3.94	20.88	5.4
CG+SOCL	**88.44**	-4.18	**23.42**	**11.12**	**6.19**	**4.02**	**21.20**	5.7
CRA-2	88.28	-4.26	18.78	9.00	4.94	3.18	19.73	5.2
CRA-2+SOCL	88.42	-4.20	20.48	10.04	5.80	3.96	20.89	**6.3**

Method	BERTScore	BARTScore	BLEU 1	BLEU 2	BLEU 3	BLEU 4	METEOR	ROUGE 2
BART	87.95	-3.88	28.39	14.88	7.76	4.30	21.99	7.3
GPT2	87.88	-3.94	22.73	11.06	5.43	2.91	19.95	5.7
CRA-3	88.05	-3.89	26.43	14.18	7.62	4.32	21.39	7.4
CRA-1	87.94	**-3.85**	29.09	15.07	7.76	4.18	22.05	7.4
CG	87.84	-3.90	**29.69**	**15.70**	**8.27**	**4.60**	21.96	7.4
CG+SOCL	87.92	-3.87	27.97	14.66	7.81	4.42	21.93	**7.6**
CRA-2	88.06	**-3.85**	28.48	14.71	7.64	4.17	21.89	7.3
CRA-2+SOCL	**88.08**	**-3.85**	29.40	15.54	8.21	4.58	**22.41**	7.5

Table 2: Automatic metrics for evaluating language generation performance on the test set of Vehicles (top), Finance (middle), and Food (bottom). Our main models **CRA-2** and **CG** outperform the baseline consistently, demonstrating the effectiveness of our mechanism. We also include the performance of **CRA-1** and **CRA-3** to show that the performance varies as the number of neighbors changes. Moreover, we show that our contrastive learning objective (**SOCL**) almost always helps the model to have better performance.

the decoder end, the decoder hidden states (from the previous token) additionally attend to encoder hidden states to produce the next token. The model computes generation probability by taking the dot product between the decoder output and the tokens embeddings from the vocabulary.

Lastly, we use negative log-likelihood loss during training for each sample

$$\mathcal{L}_G = -\sum_{i=1}^{|S|} \log P\Big(s_i | s_{<i}, T\Big) \qquad (1)$$

where S denotes the tokens for the remaining steps.

4.4 Script-Oriented Contrastive Learning

To further improve the accuracy of the generated scripts, we design a contrastive learning framework that addresses deficiencies discovered in the model outputs from 4.1 and 4.2.

Negative Sampling In contrastive learning(Hu et al., 2022), hard negative samples are constructed to guide the model to better distinguish between the incorrect samples and the desired outcome. To construct such negatives, we gain insights from the model output, from which we found that the model elicits two typical types of erroneous behavior: (1) repetition of steps from the history and (2) hallucination of non-relevant actions/concepts for a given task (e.g., sago appeared in the process of making kimchi). Based on the observation, we propose the following strategies: (1) CONCEPT REPLACEMENT, where we randomly replace the concepts in the positive sample with concepts from other tasks in TCD under the same category (e.g., Food category in WikiHow), (2) PARAPHRASED INSERTION, where we paraphrase history steps and insert them into target steps, and (3) PSEUDO TARGETS, where we construct pseudo target by sampling steps from the same category and glue them into a sequence.

Contrastive Loss In computing the contrastive loss, we generate negative targets from the strategy above for each positive script target (composition

discussed in the Appendix). To allow the model to distinguish between correct and incorrect script completions, we use the model's hidden states to compute a score to compute the correctness score for each sample, which is then used to compute the triplet loss (Schroff et al., 2015). More specifically, for each training sample

$$\mathcal{L}_{\text{CL}} = \sum_{k} \max\left(0, \phi + c_k^- - c^+\right),$$
$$c^+ = \sigma\left(\text{AvgPool}\left(\boldsymbol{W}_c \boldsymbol{H}^+ + \boldsymbol{b}_c\right)\right)$$
$$c_k^- = \sigma\left(\text{AvgPool}\left(\boldsymbol{W}_c \boldsymbol{H}_k^- + \boldsymbol{b}_c\right)\right)$$

Where $\boldsymbol{H}^+$ and $\boldsymbol{H}_k^-$ indicate the decoder hidden states for the positive and k-th negative sample, σ is the sigmoid function, AvgPool is the average pooling function, $\boldsymbol{W}_c$ and $\boldsymbol{b}_c$ are learnable parameters.

4.5 Training Objective

We jointly optimize the model on cross-entropy loss from generation and triplet loss from contrastive learning

$$\mathcal{L} = \mathcal{L}_G + \beta \mathcal{L}_{CL}$$

where β is hyper-parameter.

5 Experiments

5.1 Dataset

We collect data from Food & Entertaining (abbreviated as Food), Finance & Business (abbreviated as Finance), and Cars & other Vehicles (abbreviated as Vehicles) categories of WikiHow, which have varying data scales and therefore allow comparison of models' generalization ability. Each article from WikiHow contains a goal, (optionally) several preferences for completing the goal, and steps under each direction. The details of the dataset are shown in table 1. More processing details are in Appendix A.1.

5.2 Implementation and Training Detail

Our model is implemented using Pytorch (Paszke et al., 2019) and Huggingface Transformers (Wolf et al., 2020) with BART-base as the base generator. The reproducibility and hyperparameter details can be found in Appendix A.2.

5.3 Compared Methods

We compare our framework and methods with the state-of-the-art text generation baselines **BART** (Lewis et al., 2019) and **GPT2** (Radford et al., 2019), which don't use any of our proposed methods. The solution constructor in our framework also uses BART. We use **CRA**-k to denote top-k neighbors in 4.1 and set k to be 1,2, or 3 in our experiment. We use **CG** to denote the concept generator in 4.2. Since TCD.v0 is also based on WikiHow as source data, we make some modifications to our methods in the experiment so that we can test the scenarios where the new unseen task cannot be exactly found in the knowledge base. For concept generation, we exclude the evaluation set related articles in the dataset from its training data. For **CRA**-k, when we retrieve relevant tasks from TCD for a given goal, we remove its own key from TCD. **SOCL** indicates that script-oriented contrastive learning is used during the training.

5.4 Automatic Evaluation

Evaluation Metrics We use both deep learning and n-gram-based evaluation strategies. The metrics include BERTScore (Zhang et al., 2019), BARTScore (Yuan et al., 2021), BLEU (Papineni et al., 2002), METEOR (Banerjee and Lavie, 2005), and ROUGE (Lin, 2004). Note that BARTScore computes the log-likelihood of producing the reference text given the generated text using a BART model pretrained on ParaBank2[3]. BERTScore computes an embedding matching-based score.

Results The test set results are shown in Table 2. We can see that the performance of our method variants (both CRA and CG) is consistently better than the baselines, empirically demonstrating the effectiveness of both concept acquisition methods and that the task-concept representation of the task-specific knowledge is helpful for downstream script learning. Moreover, while CG outperforms CRA-2 on all metrics in Finance, it is less clearly observed in the others, showing that the level of effectiveness also depends on the domain. The fact that CRA-2 is better than CRA-1 most of the time shows that the set operation in Section 4.1 does improve the retrieval quality for certain domains. Meanwhile, CRA-3 consistently performs worse than CRA-2, showing that setting $k = 3$ may have removed too much useful information. With contrastive learning, the model almost always gains enhancement in performance, showing the effectiveness of our sampling strategies; an exception is shown in the Food category for CG, where it shows positive im-

[3] https://github.com/neulab/BARTScore

Name	Correctness ↑	Fluency ↑	Average Rank ↓	Best ↑	Worst ↓
BART	1.72	2.57	3.57	4.13	72.73
CG	2.32	3.06	2.65	9.09	13.22
CRA-2	**2.52**	**3.15**	**2.34**	**14.88**	**9.92**
Gold-Concepts	**3.40**	**3.55**	**1.44**	**71.90**	**4.13**

Table 3: Human evaluation results for the WikiHow dataset on all three categories combined. Scripts completion outputs (from automatic analysis) from each model are presented to human judges, who are asked to rate on Correctness (i.e., at what level does the generated solve the problem) and Fluency of the output. We also report the average rank and percentage of the time that each model output is chosen as best or worst, after asking each judge to rank outputs from different methods for each input

pact only on half of the metrics, and this might be caused by that the CG itself is effective enough and contrastive learning introduces more noise than help. We also show in Appendix that adding SOCL alone is already effective. Lastly, we observe that the Vehicle and Finance categories contain much less training data than Food, yet the performance improvement is more notable, demonstrating the potential of the methods in low-resource settings.

5.5 Human Evaluation

Evaluation Metrics & Process As mentioned in (Zhang, 2022), automatic metrics do not correlate well with human judgments on script learning tasks due to the diversity of potential solutions. As a complement to automatic metrics, we also recruited five volunteers (who are not authors) and conducted the human evaluation. The volunteers were Master/Ph.D. students with enough background knowledge to rate output. We gave each candidate a quiz composed of 20 random samples from the dataset as filtering to see if the annotator can give scores close to the authors'. We made sure that each annotator understands the assigned data samples during evaluation. In our evaluation process, we compared the output generated by BART, CRA-2, and CG with SOCL, and Gold-Concepts (a variant where concepts come groundtruth in TCD). We presented the input and the corresponding generated outputs from each method to human judges and asked them to rate the Correctness and Fluency of the output, both on a scale from 0 to 4. Correctness (or usefulness) was defined as the level of confidence that the generated steps successfully complete the specified goal and preference. We additionally asked the judges to rank the outputs according to their overall preference and their rating on Correctness & Fluency; after this procedure, we reported the average rank and the percentage of the time that each model output is chosen as the best or worst.

Goal	**How to Store Peaches**
Preference	Keeping Peaches in the Fridge
History	Rinse the peaches to clean off any dirt or debris. Dry the peaches with clean paper towels or a clean hand towel.
BART	<eos>
CG	[*peach, paper bag, store peach, container, term storage*] **Store peaches in the fridge** for up to 3 months.
CRA-2	[*peach, freezer, container, half*] **Store the peaches in an airtight container** in the freezer for up to 3 months.
Reference	Place whole, uncut peaches in the fridge on their own or in a plastic bag. ... **Store sliced peaches in an airtight container** for 1-2 days.

Table 4: Example input and output from the Food category. The noun phrases contained in square brackets in CG and CRA-2 indicate the acquired concepts from TCD. <eos> indicates an empty completion (i.e., the history is believed to contain the complete script). With fridge storage-related concepts, our models correctly produce the storing step.

Results. 121 samples are randomly selected from automatic analysis for human evaluation, where 40 come from the Vehicle category, 40 come from Finance, and 41 from Food. We present the human evaluation results in Table 3. The results align with automatic metrics roughly in general. The Gold-Concept variant is usually the best since it has access to ground-truth noun phrases from the reference. CRA-2 often generates more helpful solutions for users. The outputs from BART are worse than others most of the time, confirming the belief that using acquired task knowledge as prompts can help the model to generate better solutions on average. The Fluency level for all methods is rated to be of acceptable quality most of the time, likely due to the strong generalization capability of the pretrained language model.

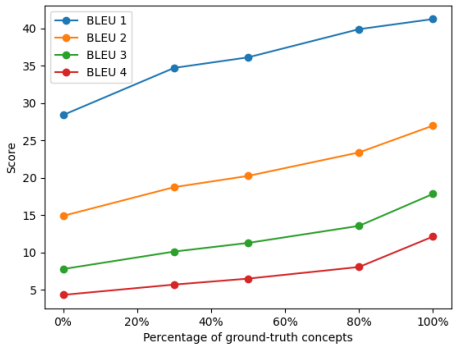

Figure 3: The plot shows that the automatic performance increases as the amount of ground-truth concepts increases, showing that deriving the right task-specific concepts can make the model generate better scripts.

5.6 Impact of Retrieval Quality

One question to be asked is how impactful it is to derive a concept prompt close to the groundtruth (i.e., concepts from TCD by using the current goal as the search key). To answer the question, we randomly draw concepts from the groundtruth at different thresholds and use each of them as a prompt to train a baseline model. From our result shown in Figure 3, we can see that the performance monotonically increases as the number of concepts covering the ground-truth concepts increases, showing the consistent benefit of using the concepts in the right context; this also demonstrates the promising direction of our methods and the significance of developing a concept deriver better at understanding task context.

5.7 Qualitative Analysis

We present an example of generated outputs by different methods in Table 4. Additional examples are available in Appendix A.3. We compare BART, CG, and CRA-2. From the generated output, we can see that BART ends the script immediately, missing the step of storing, likely due to the lack of access to knowledge about storage. On the other hand, with the acquisition from TCD, our methods can reach more contextual knowledge for the current task. Specifically, both methods are able to access storage-related concepts such as term storage, container, and freezer. As a result, the outputs match with the reference more closely. While the alignment with the reference isn't perfect (e.g., peaches are additionally placed in a bag in the reference), to most human judges, the script completion clearly achieves the goal already; this further demonstrates the difficulty of evaluating script completion.

6 Related Work

Script Learning: Scripts (Schank and Abelson, 2013; Feigenbaum et al., 1981; Yang et al., 2021; Zhang et al., 2020b,a) refers to the knowledge of stereotypical event sequences which human is constantly experiencing and repeating every day. One branch of works in script learning focuses on distilling *narrative* scripts from news or stories (Chambers and Jurafsky, 2008; Jans et al., 2012; Lee and Goldwasser, 2019), where the scripts are not goal-oriented. One of the most recent tasks in *narrative* script modeling is Multiple-Choice Narrative Cloze Test (Granroth-Wilding and Clark, 2016), where an event is removed from the chain and the model is asked to predict which event from the choices fills the blank. The other line of work, which is more closely related to our work, centers around *procedural* scripts, where a sequence of events happened often to achieve a goal. Recent work has introduced the task of constructing the entire script given a goal (Lyu et al., 2021) or choosing 1 out of 4 candidates' steps that most likely help achieve a goal (Zhang et al., 2020c). In this work, we propose a more general script learning setup that considers additionally the user preference and history, improving the generalization of models in script learning.

Our method is related to Retrieval-augmented text generation (Li et al., 2022; Zhang et al., 2022; Wu et al., 2021), a paradigm that aims to combine deep learning models with retrieval methods for text generation and has gained significant attention in recent years. For example in (Rubin et al., 2022), the author retrieves similar training examples as prompt to the current input. In our work, we are the first to introduce a framework that retrieves concepts from the concept dictionary as prompts for script learning. Apart from acquiring concepts, we also introduce a way to cancel contextual noise from neighbors by set intersections for better prompt quality. Furthermore, our method is related to contrastive learning (Wang et al., 2022; Hu et al., 2022; An et al., 2022), a line of self-supervised learning methods that improve representation learning by compacting positive samples while contrasting them with negative samples (Cao and Wang, 2021). Previous methods used negative sampling approaches such as replacing the words

with synonyms and shuffling target sentences. In our work, we introduced a new script-oriented contrastive learning objective to address script-specific issues and enhance the quality of generated scripts.

7 Conclusion

In this work, we propose TETRIS, the new task of Goal-Oriented Script Completion, which allows a model to produce the remaining steps given a user-specified goal, preference, and history of steps. We also present TASK CONCEPT DICTIONARY (TCD), a knowledge base representing task and concept association, to enable knowledge-based methods for the task. We introduce different methods to acquire concepts as prompts for the downstream text generator. We also introduce a contrastive learning strategy for script learning. The methods present consistently better performance on both automatic and human evaluation, clearly demonstrating the benefit of TCD for improving the performance of script completion. The qualitative analysis further shows that the task-specific knowledge can indeed benefit goal-oriented script learning tasks by feeding relevant knowledge about task completion. Future work could explore how to improve the acquisition quality from TCD and applications of TCD on task-oriented dialog systems.

Limitations

While TCD paired with concept acquisition methods can aid downstream script learning tasks, it doesn't consider the inclusion of actions of each step event, which can potentially benefit the script learning tasks. A possible direction is to extend the design of TCD and the concept prompt to include the semantics of actions and their orders.

Meanwhile, the concepts extracted by our method do not overlap with the ground-truth concepts (i.e., the set of concepts that appear in the reference) very well (e.g., <20% in Jaccard Index). The gap in performance between our methods and the Gold-Concept variant shows that improving the concept derivation quality might be the next step.

Furthermore, because our dataset is constructed from the English version of WikiHow, the benefits of our methods shown in the experiments are only empirically proved to work for English. We plan to further test our methods in multiple languages.

Ethics Statement

The study aims to extend deep learning-based models on the ability to generalize scripts under different user contexts. The script learning models introduced in the work can potentially be helpful for task-oriented dialog systems to suggest solutions to users. The dataset on which we base our experiments is constructed automatically from the publicly available website WikiHow. Since the website is primarily crowdsourced, the models trained on the data might incur subjective bias. During the human evaluation phase of the experiment, all involved human judges participated voluntarily and received decent payment.

Acknowledgements

We would like to thank the anonymous reviewers for their constructive suggestions on our work and thank the raters for their help on manual evaluation. This research is based upon work supported in part by U.S. DARPA KAIROS Program No. FA8750-19-2-1004. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the of ficial policies, either expressed or implied, of DARPA, or the U.S. Government. The U.S. Government is authorized to reproduce and distribute reprints for governmental purposes notwithstanding any copyright annotation therein.

References

Chenxin An, Jiangtao Feng, Kai Lv, Lingpeng Kong, Xipeng Qiu, and Xuanjing Huang. 2022. Cont: Contrastive neural text generation.

Satanjeev Banerjee and Alon Lavie. 2005. Meteor: An automatic metric for mt evaluation with improved correlation with human judgments. In *Proceedings of the acl workshop on intrinsic and extrinsic evaluation measures for machine translation and/or summarization*, pages 65–72.

Antoine Bosselut, Hannah Rashkin, Maarten Sap, Chaitanya Malaviya, Asli Celikyilmaz, and Yejin Choi. 2019. Comet: Commonsense transformers for automatic knowledge graph construction.

Shuyang Cao and Lu Wang. 2021. Cliff: Contrastive learning for improving faithfulness and factuality in abstractive summarization. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 6633–6649, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Nathanael Chambers and Dan Jurafsky. 2008. Unsupervised learning of narrative event chains. In *Proceedings of ACL-08: HLT*, pages 789–797, Columbus, Ohio. Association for Computational Linguistics.

Hongshen Chen, Xiaorui Liu, Dawei Yin, and Jiliang Tang. 2017. A survey on dialogue systems: Recent advances and new frontiers. *Acm Sigkdd Explorations Newsletter*, 19(2):25–35.

Christopher J Conti, Aparna S Varde, and Weitian Wang. 2020. Robot action planning by commonsense knowledge in human-robot collaborative tasks. In *2020 IEEE International IOT, Electronics and Mechatronics Conference (IEMTRONICS)*, pages 1–7. IEEE.

Edward A Feigenbaum, Avron Barr, and Paul R Cohen. 1981. The handbook of artificial intelligence.

Mark Granroth-Wilding and Stephen Clark. 2016. What happens next? event prediction using a compositional neural network model. *Proceedings of the AAAI Conference on Artificial Intelligence*, 30(1).

Ruocheng Guo, Lu Cheng, Jundong Li, P Richard Hahn, and Huan Liu. 2020. A survey of learning causality with data: Problems and methods. *ACM Computing Surveys (CSUR)*, 53(4):1–37.

Zhe Hu, Hou Pong Chan, Jiachen Liu, Xinyan Xiao, Hua Wu, and Lifu Huang. 2022. Planet: Dynamic content planning in autoregressive transformers for long-form text generation. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2288–2305, Dublin, Ireland. Association for Computational Linguistics.

Bram Jans, Steven Bethard, Ivan Vulic, and Marie-Francine Moens. 2012. Skip n-grams and ranking functions for predicting script events. In *Proceedings of the 13th Conference of the European Chapter of the Association for Computational Linguistics (EACL 2012)*, pages 336–344. ACL; East Stroudsburg, PA.

Jeff Johnson, Matthijs Douze, and Hervé Jégou. 2019. Billion-scale similarity search with gpus. *IEEE Transactions on Big Data*, 7(3):535–547.

I-Ta Lee and Dan Goldwasser. 2019. Multi-relational script learning for discourse relations. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4214–4226, Florence, Italy. Association for Computational Linguistics.

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*.

Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, et al. 2020. Retrieval-augmented generation for knowledge-intensive nlp tasks. *Advances in Neural Information Processing Systems*, 33:9459–9474.

Huayang Li, Yixuan Su, Deng Cai, Yan Wang, and Lemao Liu. 2022. A survey on retrieval-augmented text generation.

Chin-Yew Lin. 2004. Rouge: A package for automatic evaluation of summaries. In *Text summarization branches out*, pages 74–81.

Liyuan Liu, Haoming Jiang, Pengcheng He, Weizhu Chen, Xiaodong Liu, Jianfeng Gao, and Jiawei Han. 2019. On the variance of the adaptive learning rate and beyond. *arXiv preprint arXiv:1908.03265*.

Ye Liu, Yao Wan, Lifang He, Hao Peng, and S Yu Philip. 2021. Kg-bart: Knowledge graph-augmented bart for generative commonsense reasoning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 6418–6425.

Qing Lyu, Li Zhang, and Chris Callison-Burch. 2021. Goal-oriented script construction. *arXiv preprint arXiv:2107.13189*.

MG Mohanan and Ambuja Salgoankar. 2018. A survey of robotic motion planning in dynamic environments. *Robotics and Autonomous Systems*, 100:171–185.

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.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. 2019. Pytorch: An imperative style, high-performance deep learning library. *arXiv preprint arXiv:1912.01703*.

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*.

Nils Reimers and Iryna Gurevych. 2019. Sentence-bert: Sentence embeddings using siamese bert-networks. *arXiv preprint arXiv:1908.10084*.

Ohad Rubin, Jonathan Herzig, and Jonathan Berant. 2022. Learning to retrieve prompts for in-context learning.

Keisuke Sakaguchi, Chandra Bhagavatula, Ronan Le Bras, Niket Tandon, Peter Clark, and Yejin Choi. 2021. proscript: Partially ordered scripts generation. In *Findings of the Association for Computational Linguistics: EMNLP 2021*, pages 2138–2149, Punta Cana, Dominican Republic. Association for Computational Linguistics.

Roger C Schank and Robert P Abelson. 2013. *Scripts, plans, goals, and understanding: An inquiry into human knowledge structures*. Psychology Press.

Florian Schroff, Dmitry Kalenichenko, and James Philbin. 2015. Facenet: A unified embedding for face recognition and clustering. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 815–823.

Liang Wang, Wei Zhao, Zhuoyu Wei, and Jingming Liu. 2022. Simkgc: Simple contrastive knowledge graph completion with pre-trained language models. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 4281–4294, Dublin, Ireland. Association for Computational Linguistics.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Remi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander Rush. 2020. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45, Online. Association for Computational Linguistics.

Zeqiu Wu, Michel Galley, Chris Brockett, Yizhe Zhang, Xiang Gao, Chris Quirk, Rik Koncel-Kedziorski, Jianfeng Gao, Hannaneh Hajishirzi, Mari Ostendorf, and Bill Dolan. 2021. A controllable model of grounded response generation. *Proceedings of the AAAI Conference on Artificial Intelligence*, 35(16):14085–14093.

Yue Yang, Artemis Panagopoulou, Qing Lyu, Li Zhang, Mark Yatskar, and Chris Callison-Burch. 2021. Visual goal-step inference using wikihow. *arXiv preprint arXiv:2104.05845*.

Weizhe Yuan, Graham Neubig, and Pengfei Liu. 2021. Bartscore: Evaluating generated text as text generation. *Advances in Neural Information Processing Systems*, 34:27263–27277.

Li Zhang. 2022. Reasoning about procedures with natural language processing: A tutorial.

Li Zhang, Qing Lyu, and Chris Callison-Burch. 2020a. Intent detection with wikihow. *arXiv preprint arXiv:2009.05781*.

Li Zhang, Qing Lyu, and Chris Callison-Burch. 2020b. Reasoning about goals, steps, and temporal ordering with wikihow. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 4630–4639.

Li Zhang, Qing Lyu, and Chris Callison-Burch. 2020c. Reasoning about goals, steps, and temporal ordering with wikihow. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 4630–4639.

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

Yizhe Zhang, Siqi Sun, Xiang Gao, Yuwei Fang, Chris Brockett, Michel Galley, Jianfeng Gao, and Bill Dolan. 2022. Retgen: A joint framework for retrieval and grounded text generation modeling.

A Appendix

A.1 Dataset & TCD

The dataset statistics are shown in Table 1. The raw corpus comes from the 07/20/21 snapshot of WikiHow. We filter out unordered scripts by using both the classification results from (Lyu et al., 2021) and the WikiHow section type. We perform a 6:2:2 split on the articles to create a train, development, and test set. We create a history $\mathcal{H}$ in data samples for TETRIS by randomly splitting a sequence of steps under each preference into two halves. For TCD, we have 206621 keys in total and 10.37 concepts per key on average.

A.2 Implementation Details

We implement the models using the 4.8.2 version of Huggingface Transformer library[5](Wolf et al., 2020). We use the Oct 1, 2021 commit version of the BART-base model (139M parameters) from Huggingface[6]. The contrastive learning variants has 140M parameters. For SBERT in Section 4.1, we use the all-mpnet-base-v2 checkpoint from sentence transformer library [7]. We use Huggingface datasets[8] for automatic evaluation metrics. The BART Score comes from the author's repository[9] and we used the one trained on ParaBank2. The hyperparameters for the experiment (non-contrastive

[5] https://github.com/huggingface/transformers

[6] https://huggingface.co/facebook/bart-base/commit/ea0107eec489da9597e9eefd095eb691fcc7b4f9

[7] https://www.sbert.net/

[8] https://github.com/huggingface/datasets

[9] https://github.com/neulab/BARTScore

Method	BERTScore	BARTScore	BLEU 1	BLEU 2	BLEU 3	BLEU 4	METEOR	ROUGE 2
BART	**87.87**	-4.37	16.61	8.31	4.67	2.92	18.92	5.6
BART+SOCL	87.725	**-4.349**	**19.478**	**9.441**	**5.216**	**3.236**	**19.198**	**5.73**

Table 5: On the Finance dataset, our result shows that SOCL alone already brings consistent benefits to the performance.

Table 6: Hyperparameters for non-contrastive learning (CL) models (some are introduced and changed for CL experiment). The ones below the mid-line are generation related. Batch size is changed for contrastive learning.

Name	Value
seed	42
learning rate	3e-5
batch size	16
weight decay	5e-4
RAdam epsilon	1e-8
RAdam betas	(0.9, 0.999)
scheduler	linear
warmup ratio (for scheduler)	0.06
number of epochs	25
metric for early stop	SacreBLEU[4]
patience (for early stop)	15
length penalty	1.2
max length	511
min length	2
beam search size	5

learning) are shown in Table 6 (applied to all models) and the ones not listed in the table are set to be default values from the transformer library. We use RAdam (Liu et al., 2019) as the optimizer. We perform hyperparameter search on batch size from {16, 32}, pretrained language model learning rate from {2e-5, 3e-5, 4e-5}, downstream learning rate for contrastive learning from {1e-3, 5e-4, 1e-4, 3e-5}, negative sample composition from type A "1 sample from each of type 1 2, filling empty case with type 3" and type BCD "2 samples from the type combination (2 3/1 2/1 3), filling empty case with empty string", and the number of epochs from {8, 15, 25} ({28, 32} for contrastive learning). For contrastive learning experiment negative sampling, we use type A for Vehicles and Food and type B for Finance. We perform our experiments on 40 GB A100 and 32 GB V100 GPU. For contrastive learning, we use batch size of 28 for Vehicles, 32 for Food, and 32 and 28 for CRA and CG respectively in Finance. The downstream learning rates are respectively 1e-3, 1e-3, 5e-4 for CRA-2+SOCL and 3e-5, 1e-4, 1e-3 for CG+SOCL in the datasets of Vehicles, Finance, and Food. We use 0.5 for the margin in the triplet loss function and 0.3 for the β in the training loss. The experiments can take up to 10 hours.

A.3 Additional Model Outputs and Analysis

We present examples in Table 7, 8, and 9, in addition to Section 5.7. From Table 7, we can see that while BART ends the script immediately after finding an agency, our methods are able to provide detail on the interaction with the agency. This shows that training with the acquired concepts as prompts allows the model to gain more task-specific contextual knowledge, which can be potentially shared in the token embeddings of concepts. In Table 8, the user preference in the example contains some ambiguity (i.e., it could mean making just the dough, or the complete process of making the pasta). While BART is missing the further processing on the *dough*, CG and CRA-2 are able to not only include detailed and reasonable steps that process the *dough* but also some information on what to do after the dough is being made, showing the usefulness of incorporating task knowledge. The reason for the improvement could be that the prompts of our method (all including the word *dough*) allow the model to be attentive to the processing of *dough*. In Table 9, the methods without gold concepts all believe the script has ended, which is a reasonable output. Yet they would be low on automatic generation metrics because of not matching reference well (which contains steps not necessarily needed to complete the goal). This shows the difficulty of automatically evaluating script generation since there can be many ways to solve a task and can be different in the amount of detail.

A.4 Discussions

While we show in the experiment that TCD can be used in the script completion task in our work to achieve better performance, it can also naturally be applied to other domains such as task-oriented

Goal	**How to Report Pet Store Neglect**
Preference	Reporting to Local Law Enforcement
History	Document the situation as extensively as possible Check your state's animal welfare law Find out which agency is responsible for enforcing animal laws
BART	Finished
CG	[dog] Report the situation to local law enforcement Follow up with any follow-up questions Finished
CRA-2	[police, report, animal control, authority] Contact local law enforcement agencies if necessary Follow up with any follow-up questions Finished
Gold-Concept	[situation, state, animal welfare law, agency, animal law, documentation, record, report, law enforcement] Gather documentation and record of the report to law enforcement Follow up with your report if necessary Finished
Reference	Gather your documentation and other records File your report with law enforcement Follow up on your report Finished

Table 7: Example output among different methods (Finance and Business)

dialog, event causal identification, and causal inference. Furthermore, since the two different concept acquisition methods in the paper contain different advantages, it would be an interesting task to devise a method that has both benefits of being explicit in the acquisition process and being parametrized.

One natural question to ask is why we represent the KB as concept-based instead of directly using paragraphs to record each goal. While How-to websites contain human-curated solutions, the way of annotation introduces canonicalization problems since different people might have 1. different orders of completing the same task and 2. different levels of detail in the expression; on the other hand, using concepts representation helps alleviate the problem since the core concepts (e.g., "butter", "temperature", "time" in "Baking a Cake") almost always participates regardless of how they are organized in each person's memory. Furthermore, TCD introduces an interesting function of automated self-enrichment. In hierarchically related goals, such as "how to bathe a dog" and "how to bathe a cat", we can use **WordNet Hierarchy** and **Concepts Intersection** to automatically derive new nodes. We leave the experimentation to future work.

Another important problem to be tackled is that, while human evaluation is more reliable than automatic evaluation, it is also much more costly. As mentioned in (Zhang, 2022), automatic metrics do not align with that of humans very well. One promising direction is to create an automatic metric that does not evaluate script generation methods only based on matching with the reference (since many equivalently valid references can exist).

Goal	**How to Make Macaroni Pasta**
Preference	Making Pasta Dough at Home
History	Mix the dry ingredients Create a hole or well in the dry ingredients Add three large eggs Whisk the eggs with a fork Place the dough on a clean counter
BART	Bake the pasta Serve the pasta while it is still warm Finished
CG	[flour, salt, bowl, butter, milk, egg, flour mixture, dough, surface] Roll the dough out into a rectangle Cut the dough into 1/4 inch (0.64 cm) circles Transfer the dough to a plate Repeat the process with the rest of the dough Cook the pasta Place the pasta on a plate to warm up Serve Finished
CRA-2	[dough] Roll the dough into a ball Repeat the process with the rest of the dough Transfer the dough to a baking sheet Bake the pasta Serve or refrigerate Finished
Gold-Concept	[ingredient, hole, egg, fork, dough, counter, surface, bowl] Place the dough on a floured surface Transfer the dough to a serving bowl Serve Finished
Reference	Fold the dough Look for a smooth surface Place the dough in a clean bowl to rest Finished

Table 8: Example output among different methods (Food and Entertaining category).

Goal	**How to Track a Flight in Progress**
Preference	Via Online Websites
History	Gather the information about the flight: Find a flight-tracking website to find out the information about your flight Enter your flight information Watch for plugins that you could need to add to your browsers functionality to run the website Watch a map come up on your screen showing the route the plane has taken, as well as the current location of the plane and the expected route ahead
BART	Finished
CG	[flyer, airline, website, flight, status, status update] Finished
CRA-2	[airline, flight, flight number] Finished
Gold-Concept	[information, flight, tracking website, flight information, plugin, browser functionality, website, map, screen, route, plane, location, auto, update feature, way, site, ability] Use the auto-update feature on the way out of the site Allow the site to continue running while you're not using it Check to see if the update feature is working properly
Reference	Figure out if the website you chose has an auto-update feature Look for ways to zoom in on the plane, if this site allows that ability Finished

Table 9: Example output among different methods (Cars & Vehicles category)