Take a Break in the Middle: Investigating Subgoals towards Hierarchical Script Generation

Xinze Li[1], **Yixin Cao**[2†], **Muhao Chen**[3], **Aixin Sun**[1†]

[1] S-Lab, Nanyang Technological University
[2] Singapore Management University [3] University of Southern California
{xinze.li, axsun}@ntu.edu.sg
yxcao@smu.edu.sg muhaoche@usc.edu

Abstract

Goal-oriented Script Generation is a new task of generating a list of steps that can fulfill the given goal. In this paper, we propose to extend the task from the perspective of cognitive theory. Instead of a simple flat structure, the steps are typically organized hierarchically — Human often decompose a complex task into subgoals, where each subgoal can be further decomposed into steps. To establish the benchmark, we contribute a new dataset, propose several baseline methods, and set up evaluation metrics. Both automatic and human evaluation verify the high-quality of dataset, as well as the effectiveness of incorporating subgoals into hierarchical script generation. Furthermore, We also design and evaluate the model to discover subgoal, and find that it is a bit more difficult to decompose the goals than summarizing from segmented steps.

1 Introduction

Scripts are forms of knowledge representations for ordered events directed by particular goals (Herman, 1997). As shown in Figure 1, *to obtain a Ph.D degree* (i.e., goal), one shall follow specific events step-by-step, including *do the research*, *write the paper*, etc. Such procedure knowledge not only provides a process of problem solving, but also benefits many real-world applications, such as narrative understanding (Chaturvedi et al., 2017), task bots (Peng et al., 2021), and diagnostic prediction (Zhang et al., 2020c). Therefore, the task of script generation is proposed to automatically generate events given a goal (Lyu et al., 2021).

Existing works typically assume that events are sequentially arranged in a script, while we argue that this assumption leads to linear generation that is far from enough for comprehensively acquiring the representation about how events are organized towards a task goal. When humans compose a script, the underlying procedure of a task is often not a simple, flat sequence. As suggested by cognitive studies (Botvinick, 2008; Zhang and Norman, 1994), human problem solving often involves hierarchical decomposition of the task. That is to say, a more complicated task is often decomposed into subgoals, and each subgoal can be further decomposed into more fine-grained steps. For instance, the process of *obtaining a Ph.D. degree* can divide into subgoals like *publishing research papers*, *passing the qualification exam*, and *defending the thesis*. (Figure. 1). The subgoal *publishing research papers* thereof further consists of more fine-grained steps like *doing the research*, *writing the paper*, and *passing peer review*. Accordingly, a proper way of acquiring script knowledge should also hierarchically capture different levels of task subgoals.

Figure 1: An illustration of the hierarchical decomposition of the goal "*How to obtain a Ph.D. degree*". We use blue for the steps, and yellow for the subgoals. Conventional task focuses on steps only, while we highlight the breaking in the middle (red arrows), which refers to the switch between higher-level subgoals.

In this paper, we propose to investigate subgoals as an intermediate between goals and steps. Given

[†]Co-corresponding Author.

Findings of the Association for Computational Linguistics: ACL 2023, pages 10129–10147

a *goal*, instead of generating steps in a linear manner, our task seeks to generate scripts at two levels. The first level consists of subgoals, and the second level is detailed steps, where each subgoal contains several steps. Such a new setting not only accords to people's cognitive process (Antonietti et al., 2000) but also investigates the model's abilities to understand knowledge from two aspects: problem decomposition and summarization. We propose three research questions to assist the investigation: 1) How to identify the subgoals? 2) How to effectively introduce subgoals to models to improve script generation? 3) Does the generated hierarchy align with human intuition?

To answer the first question, we construct a new dataset, namely "Instructables"[1], about D.I.Y projects involving goals, subgoals, and steps. Besides, we extend the existing wikiHow dataset (Zhang et al., 2020b) with the subgoals. To automatically obtain subgoals, we deploy a segmentation method that separates steps in training data. For each segment, we further leverage a prompt-based fine-tuning (Lester et al., 2021) method to learn subgoal generation. We have designed quantitative metrics for evaluation, which verifies the reasonability of the two-level hierarchy of script.

For the second question, we build the benchmark with multiple baselines. The basic idea is to incorporate the goal, subgoals, and steps into one prompt template, and use special tokens to preserve structure information. Then, we finetune Pre-trained Language Models (PLMs) to generate in a top-down or interleaving manner. This allows the model to *take a break* to conclude on each subgoal before generating the succeeding steps. We have conducted extensive experiments. The results show that by including subgoals, the model generates scripts with better soundness, diversity, and making better sense to achieve the goal. In fact, given gold standard segmentation and subgoals, the improvement is more substantial, indicating space for improvement in our predicted subgoals.

To address the third question, we conduct human evaluation to assess the quality of both steps and subgoals, as well as the above two types of model abilities. We observe that the language model shows a weaker ability to decompose the goals than to summarize the steps. We have also analyzed the errors in detail and found that the models sometimes generate repetitive subgoals and low-quality steps. The models still have much room for improvement in generating high-quality hierarchical scripts if the goal is too complicated.

To summarize, our work has three contributions: 1) We construct the dataset "Instructables" and extend the wikiHow dataset to investigate the subgoals as intermediate between goals and steps. 2) We build a benchmark with several baselines towards hierarchical script generation. 3) We conduct extensive evaluation to investigate the effects of subgoals qualitatively and quantitatively.

2 Related Work

Procedural Knowledge Acquisition Early research on the script and procedural knowledge is usually formulated as a classification or ranking problem. For example, Modi and Titov (2014) and Pichotta and Mooney (2016) predict a score for each given event to determine their relative order based on event representation learning. P2GT (Chen et al., 2020) and APSI (Zhang et al., 2020a) further analyze the intention of events and conduct a joint learning-to-rank for better ordering. Thanks to the success of the Pre-trained Language Model (PLM), recent work GOSC (Lyu et al., 2021) proposes to generate steps in a simple, flat manner for any goal. Another line of works (Pareti et al., 2014; Lagos et al., 2017) have attempted to establish a hierarchical structure among scripts by linking their steps. Given any event of goal *A*, Zhou et al. (2022) compute a similarity score to find the most semantically close goal *B*, so that all events of *B* can be regarded as detailed subevents of the given event at the lower level. This approach, although effective, has an exceptionally high demand on the dataset to cover a wide range of goals. In many cases, there is no reasonable goal for alignment, which results in a deviation in the meanings between the linked sentences. Therefore, we neither regard steps in a flat manner, nor link steps and goals with the retrieve-then-rerank approach. Instead, we propose a task and model targeting the inner hierarchy of a script during generation and are complementary to the above works.

Controlled NLG Script generation is a form of controlled text generation task (Hu et al., 2017) since the generated scripts are attributed to the given goal. To increase the controllability of text generation, research efforts investigate the ways of constrained decoding. NeuroLogic Decod-

[1]Our code and dataset are available at: https://github.com/lixinze777/Hierarchical-Script-Generation

ing (Lu et al., 2021) improves controlled generation upon semantic constraints by enforcing predicate logic formula at the decoding stage. NeuroLogic A*esque Decoding (Lu et al., 2022) further incorporates a lookahead heuristic to estimate future constraint satisfaction. Controlled text generation tasks can take other forms like generating descriptions conditioned on subparts of a table (Wang et al., 2022). Another classic application of controlled text generation is storytelling, whereby stories are generated based on a prompt or a storyline. (Fan et al., 2018) generate hierarchical stories conditioned on a prompt that was generated first. (Fan et al., 2019) enhance the coherence among different levels of a hierarchical story with a verb-attention mechanism. Unlike tasks like storytelling, script generation is not open-ended since it is goal-oriented.

3 Task and Dataset

In this section, we first formulate the new task setting and then introduce the new dataset that we constructed, namely Instructables.

3.1 Task Definition

The original *goal-oriented script generation* (Lyu et al., 2021) (GOSC) focuses on generating a *sequence* of steps (or events) that accomplishes the given goal. In contrast, the proposed *hierarchical script generation* conducts hierarchical generation and models the script as multiple levels of events. Formally, given a *goal* g as input, it is to generate L levels of events as output, where the events at the 1-st to the $(L-1)$-th levels are called *subgoals (s)* and the events at the L-th (most fine-grained) level are called *steps (t)* . Within each level, the list of children events should fulfill the objective of their parent event. Note that the number of events at each level is not fixed, and the model is required to decide the number by itself. Based on our observation, two levels of events are sufficient for most scripts (i.e., $L = 2$) in reality. For example, both websites, wikiHow and Instructables, define several sections for each goal, and each section contains multiple steps. These task instruction resources are all organized in two levels. Thus, in the rest of the paper, we define $L = 2$.

This task inherently include 2 settings. The input for both settings are the goal g. Setting 1 takes the sequence of events from the lowest level of the hierarchy as output, which is the same as the GOSC task. Through this setting, we investigate whether including subgoals improve the traditional script generation task. Setting 2 takes the entire hierarchy of events as output. Through this new setting, we investigate the language model's ability to generate high-quality subgoals and steps.

3.2 Dataset

We use two datasets, wikiHow and Instructables, for the studied task. The wikiHow dataset (Zhang et al., 2020b) is a collection of how-to articles crawled from the wikiHow website. Each article describes methods to achieve a given goal in the title. The articles are written in sections, each of which comes with a few steps. In this work, we consider one section of steps as one segment and a section name as a subgoal. Due to the lack of resources, many research works on script use wikiHow as the only dataset.

To verify the model's generalizability over more than a single dataset, we construct another dataset based on Instructables[2] — a website specializing in user-created do-it-yourself (DIY) projects.

Instructables Construction The data construction process consists of two stages. The first is raw data preparation. We collect the content of each project according to its category (Circuits, Workshop, Craft, Cooking, Living, and Outside) using Scrapy.[3] Each project consists of a title showing the item that the author sought to make (i.e., a toy rocket) and the instructions to make this item. In most cases, authors write the instruction in a few sections, each with a step-by-step description. During crawling, We take each section name as a subgoal and every sentence as a step.

The second stage is filtering. The raw data is inconsistent in text style, section format, article length, etc. We hereby carry out seven steps to filter out noisy data. We remove: 1) Non-English projects using Python library langdetect.[4] 2) The section on Supplies or Materials, which describes composed materials instead of events/actions. 3) The section numbers (e.g., *"section 3: draw a line" -> "draw a line"*). 4) Unnecessary spaces and characters like Line Feeder to maintain the human-readable text. 5) Projects with empty text content in any section since these sections are usually presented as figures or videos. 6) Projects with any overly lengthy section. Many authors post

[2] https://www.instructables.com/
[3] https://github.com/scrapy/scrapy
[4] https://pypi.org/project/langdetect/

Category	Scripts	Subgoals	Steps
Circuits	22,437	109,917	282,685
Workshop	16,991	94,554	257,248
Craft	24,874	137,471	365,244
Cooking	12,916	69,633	189,371
Living	23,204	114,113	291,682
Outside	6,986	34,391	92,439
TOTAL	107,408	560,079	1,478,669

Table 1: Number of total scripts, subgoals, and steps in dataset Instructables by category.

stories or anecdotes to convey the rationale they came up with the project, and we find 128 words a good threshold to filter them out. 7) Projects that build the same item as others. We remove repeated articles about seen items to eliminate possible redundancy or anomalies in data distribution. Finally, we unify the format of project titles to make them consistent. By performing Part-of-Speech (POS) Tagging on titles, we prefix "How to make" to noun phrases (e.g., *How to make* Kung Pao Tofu), we recover the verb from stemming and prefix "How to" to verb phrases (e.g., *How to* Build a Toy Rocket), and we retain the How-to questions (e.g., How to Create a Puppet).

Dataset Statistics Table 1 shows the statistics of Instructables by category. In total, we obtain 107,408 scripts, 560,079 subgoals, 1,478,669 steps, and 26,813,397 words. We analyze the difference between Instructables and wikiHow in appendix A

4 Method

In this section, we present our series of methods to build the benchmark for the hierarchical script generation task. Note that we do not target a best-performing model but aim at a reasonable baseline to shed light on future research. We first introduce a segmentation method that automatically segment the steps and generate their subgoals, in case there is no ground truth hierarchy available for training. Then, given the goal, subgoals, and steps, we introduce the proposed framework for training. Finally, given a goal, we detail the inference process.

4.1 Segmentation

The dataset wikiHow and Instructables naturally manifest an easy-to-consume hierarchical layout, as we explained before. However, for better generalization, we do not assume that all sources of script data possess the privilege of a hierarchical layout with sections and subgoals. Formally, given an ordered list of steps, we seek to find segmentation points between steps to separate them into relatively concrete segments. Each segment should inherently represent a subgoal. To find these segmentation points in an unsupervised manner, we propose four methods. The first method finds low probability scores between consecutive steps via BERT **next sentence prediction** (Devlin et al., 2019). The second method measures the plausibility of a list of steps with **perplexity** and locates the abnormal ones. The third method applies **clustering** algorithm to group steps based on their sentence embeddings. The last method locates segmentation points upon multiple **topics detected** via fastclustering (Reimers and Gurevych, 2019). We explain these methods in detail in appendix B.

4.2 Subgoal Labeling

Given steps and their segmentation, we are to generate an event for each segment as their parent subgoal, where the subgoal is a high-level summarization of its children steps. Due to the lack of annotations, we perform the labeling in a self-supervised manner. That is to say, we regard it as a dual problem of script generation. Given a goal and all the steps in a flat format, instead of training a model to generate steps, we fine-tune a T5-base model (Raffel et al., 2020) to generate the goal using the list of steps as inputs. Specifically, We convert the question-format goal into a verb phrase by removing the "How to" prefix, which is more suitable as a subgoal. Note that we did not include any additional training data but reused the training dataset for the script generation task. This practice ensures that the system is not leaked with any sentences in the development or testing data to gain unfair information at training time.

4.3 Hierarchical Generation

Training Given a goal, we train a model to generate a varying number of subgoals, and each subgoal has its own steps. Thanks to the recent progress of prompt-based fine-tuning (Lester et al., 2021), we use the special token `<section>` to preserve the structure information (Yao et al., 2019) and take advantage of PLMs to generate such a two-level structure by re-formatting a prompt:

> [Goal], `<section>` With [Subgoal], [steps]. `<section>` With [Subgoal], [steps]

An example prompt for the goal *How to learn Web Design* is as below:

> *To learn Web Design, <section> with Finding Web Design Resources, check online for web design courses and tutorials. Look into taking a class at a local college or university... <section> With Mastering HTML, familiarize yourself with basic HTML tags. Learn to use tag attributes...*

Intuitively, there are two typical generation sequences for a multi-granular generation task, **interleaving** and **top-down**. The above template adopts an interleaving generation sequence. In terms of the auto-regressive decoding sequence, a subgoal is generated, followed by its affiliated steps, the next subgoal, and so on. We also propose a template incorporating a top-down generation sequence where all subgoals are generated first followed by all the steps. The prompt is as follows:

> [Goal], the subgoals are: [Subgoal], [Subgoal]. <section>, [steps]. <section>, [steps]

For both generation sequences, we add a special token <section> as a delimiter between two segments to denote the hierarchy information. Of course, for more levels, one can add more special tokens as delimiters between different levels. Inspired by Yao et al. (2019), we use a special token to delimit different parts of the structure instead of conducting a complex hierarchy decoding process. This technique leads to two benefits. First, it allows a sequential generation process that aligns well with the pre-training objective of PLM, therefore facilitating knowledge prompting from the PLM. Second, the hierarchy information improves long-text generation (e.g., sometimes there are many steps to decode) because the subgoal shortens the dependency between steps by providing a high-level summarization. We leave the exploration for other long-text generation tasks in the future.

Inference At inference time, we feed the how-to question, which contains the goal, into the tuned PLM as input, with the prefix "Ask question:" as common practice for Question Answering tasks using the PLM. We fix the hyper-parameters the same as across training settings. The decoder generates subgoals and steps in an interleaving/top-down approach, and the output is the same as the prompt format we design for training sets. We leverage the special tokens in output as the beacon for extracting subgoals and subsequently expand the linear output into the hierarchical format.

5 Experiments

To evaluate the proposed framework for hierarchical script generation, we conduct extensive experiments on the presented wikiHow and Instructables datasets. We compare our proposed framework with prior strong baseline method and discuss the results (§ 5.2-§ 5.3). We have also investigated the best segmentation method as a secondary experiment. (§ 5.4).We conduct ablation studies (§ 5.5).

5.1 Experimental Setup

Dataset Following the setup from Lyu et al. (2021), we randomly separate datasets into training and test sets using a 90/10% split, and hold out 5% of the training set as the development set. We perform both automatic and human evaluations to assess the quality of the generated script.

Metrics For automatic evaluation metrics, we first follow prior works (Lyu et al., 2021) without considering the hierarchy information as our task setting 1 § 3.1 and report **perplexity** and **BERTScore** (Zhang et al., 2019) for steps. Perplexity is the exponential of log-likelihood of the sequence of steps assigned by the model. In addition, we compute three widely used generation metrics, **BLEU-1** (Papineni et al., 2002), **ROUGE-L** (Lin, 2004) and **Distinct-n metric** (Li et al., 2016) by taking average over all testing data. For our experiment that investigates the best segmentation strategy, due to the lack of measurements, we define a metric "segment distance" and the details can be found in § 5.4. When evaluating the hierarchical scripts, we remove the subgoals and prompt template to ensure fair comparison.

Baseline Given the different nature of the hierarchical script generation in contrast to the previous task, there is not a directly applicable baseline. Here, we choose a state-of-the-art model for conventional script generation, namely GOSC (Lyu et al., 2021), as a strong baseline. Note that we carefully re-implement the model according to the settings and parameters from (Lyu et al., 2021) and report a variance where the mT5-base model of GOSC is replaced as T5-base model (Raffel et al., 2020) to learn better on the English corpus, since GOSC was originally designed for a multilingual setting. Another baseline is a two-stage generation

Dataset	Method	Bert. ↑	Perp. ↓	BLEU-1↑	ROUGE-L ↑	Dist.-3 ↑
wikiHow	GOSC (mT5-base)	82.3	17.0	-	-	-
	GOSC	85.3	14.3	20.4	20.2	93.4
	GOSC (two-stage)	83.6	20.9	16.2	22.6	93.7
	HSG (top-down)	85.5	**12.8**	22.9	23.2	**94.0**
	HSG (interleaving)	**85.6**	15.4	**23.7**	**24.1**	**94.0**
	HSG w gold subgoal	85.8	12.7	31.2	27.0	95.9
Instructables	GOSC	82.2	31.3	16.6	16.0	92.5
	HSG (top-down)	82.0	**19.5**	17.4	17.8	94.4
	HSG (interleaving)	**82.8**	22.2	**18.9**	**21.8**	**94.5**
	HSG w gold subgoal	82.8	24.3	25.1	23.4	96.5

Table 2: Performance of our method (marked as HSG) in top-down and interleaving approach on test set of wikiHow and Instructables datasets. We also report the cases where gold segments and subgoals used in the generation process as a performance upper bound. We abbreviate BERTScore, Perplexity, and Distinct-3 to Bert., Perp., and Dist.-3 respectively. We **bold** the best performance.

process. First generating the steps in GOSC manner, then using these steps as subgoals to generate the actual steps.

5.2 Automatic Evaluation

We report the average results of 3 runs of hierarchical script generation on both datasets in Table 2. We compare the quality of the generated text according to the four metrics (§ 5.1). For each dataset, we also report the case where ground truth segmentation and subgoals (section name) are provided as a performance upper bound (of the proposed prompt-based fine-tuning method, as better performance may be achieved by more advanced model).

We can observe that the results from both datasets are generally consistent. **(1)** The two-stage baseline is weaker than the basic baseline as indicated by most metrics. This baseline has an error aggregation problem whereby the steps generated could be irrelevant after two stages of generation. **(2)** Our method has outperformed the baseline in three metrics (Perplexity, BLEU-1, and ROUGE-L scores), indicating the effectiveness of our method in generating scripts with higher quality. **(3)** The improvement in Distinct-3 metric on both datasets indicates that our method is capable of generating texts of greater variation. With segmentation, we take a break in the middle of the generation process, and provides the model a chance to conclude on steps of the current subgoal, and refer to the information from the upper level of the hierarchy. The model thereafter generates the script with better quality and less repetition. **(4)** Between the top-down and interleaving approaches, the latter achieves slightly better or tied scores among almost

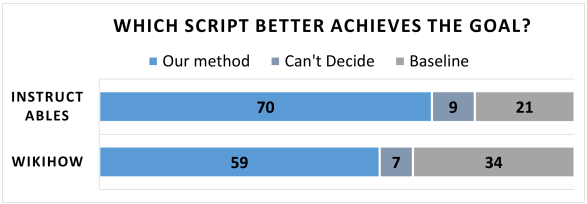

Figure 2: Stacked bar chart showing the result of human evaluation on question related to steps. Blue, grey and orange color respectively indicates the percentage that our script is preferred, baseline script is preferred, and cannot decide.

all metrics for both datasets. The subgoals are in proximity to the corresponding steps for the interleaving approach, which better guides the step generation. **(5)** It is noteworthy that our method with predicted subgoals outperforms the gold segmentation and subgoals on perplexity for Instructables dataset, showing that our generated subgoals might be closer to natural language compared to using gold subgoals from Instructables. **(6)** Except for this, using gold segmentation and subgoal leads to better results in all other metrics on both datasets, manifesting an enormous potential of our method, which also indicates an area of improvement for the accuracy of our predicted segmentation and subgoals. We acknowledge that existing metrics are unable to directly measure how well a script fulfills a goal. Hence, we conduct human evaluations to complement automatic evaluation.

5.3 Human Evaluation

We conduct human evaluations to assess the quality of our hierarchical scripts for task setting 2 § 3.1. We evaluate the scripts on both the steps

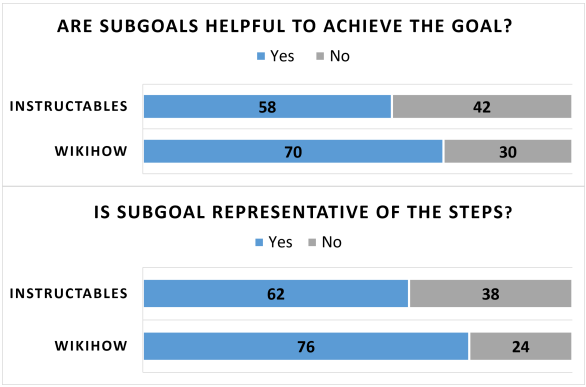

Figure 3: Results on subgoals. Blue and grey color respectively indicates the percentage of "Yes" and "No"

and the subgoals. The steps are assessed through direct comparison, where we ask the annotators to choose between scripts generated by our method (flattened) and by the baseline. In addition, we also evaluate the generated subgoals based on two criteria. The first criterion concerns whether the annotators consider the generated subgoals as valid components of the task goal, i.e., problem decomposition ability. In the context of the main goal, the second criterion concerns if the generated subgoal properly represents the associated steps, i.e., summarization ability. We provide more details (e.g., guideline and annotators) of human evaluation in appendix D.

Step Evaluation From Figure 2, the scripts generated by our method were more favored by annotators over the baseline scripts in both wikiHow (59%) and Instructables (70%) test sets than the baseline scripts, excluding 7% and 9% draw cases. In addition, we realize that the proportion of favored scripts is higher on the Instructables dataset than on wikiHow. Such results are due to the high-quality wikiHow scripts generated by the baseline method. Our method has a greater improvement on Instructables, similarly, attributes to the low-quality scripts from the baseline, mainly because Instructables' scripts are more difficult and longer, which requires the ability of problem decomposition. In extreme cases, we observed empty or very short outputs for Instructables using the baseline method, which did not appear in the scripts generated by our method. Further, we analyze more typical examples and mistakes in case study (appendix E).

Subgoal Evaluation From Figure 3, regarding the question of whether subgoals are helpful to achieve the goal, 70% of the subgoals are given credit by the annotators for the wikiHow dataset, while this percentage is 58% for the Instructables dataset. For the other question assessing whether generated subgoal well-represents the associated steps, the percentage of positive responses for the wikiHow dataset (76%) also surpasses that of the Instructables dataset (62%). The results from these two questions accord with each other that the subgoals generated for the Instructables dataset are of worse quality than that of wikiHow. From another perspective, comparing the results between two questions, we find that the generated subgoals have a weaker degree of association with the goals than with the generated steps. The results demonstrate a great challenge on complex task decomposition.

Segmentation Method	k=3	k=4
2 Segments	4.17	5.20
3 Segments	4.23	5.16
Next Sentence Prediction	4.23	5.26
Perplexity	4.30	5.36
Agglomerative Clustering	4.08	4.89
Topic Detecting	**3.82**	**4.69**

Table 3: Segment distances of the proposed segmentation methods on wikiHow dataset.

5.4 Segmentation

To better understand the best segmentation strategy for our task, we assess distinct segmentation techniques and aim to find the one with the closest segment structure to the ground truth. In order to measure the affinity between predicted and gold segmentation points, we propose the metric "segment distance" in light of the metric "edit distance" in quantifying string similarity. Instead of calculating the number of minimal editing scripts, "segment distance" calculates the least number of steps to shift m predicted segmentation points to the actual ones, where $m = \min(p-1, g-1)$, p is the number of predicted segments, and g is the number of segments in ground truth. In addition, we impose of penalty score $P = k * d$ for difference of number of segments $d = |p - g|$ as a metric that encourages accurate estimation of number of segments. The k value is set between 3 and 4 as a fair penalization.

As a baseline, we take the average number of segments N (closet integer) from the dataset and carry out an N-equal splits on each script. This simple approach is in fact a strong baseline since

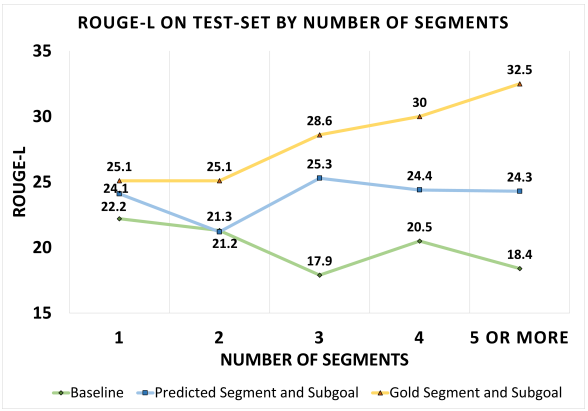

Figure 4: ROUGE-L Score on test set categorize by number of segments from "1" to "5 or more". Green color represents the baseline method. Blue color represents our method. Yellow color represents our method using gold segmentation and subgoals.

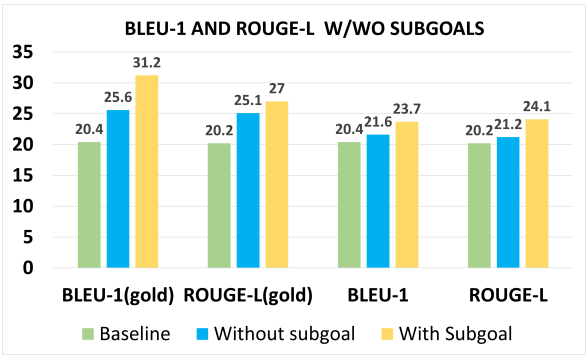

Figure 5: BLEU-1 and ROUGE-L scores for generated scripts. Green color represents the baseline method. Blue color represents our method using segmentation separated with special tokens. Yellow color represents our method using both segmentation and subgoals.

most scripts have a limited number of steps in different segments. This baseline inherently maintains a small penalty score P. We evaluate the segmentation performance on 1,000 random scripts.

In Table 3, we report the results of segmentation experiment on wikiHow dataset as a representation, where a smaller average segment distance indicates a structurally similar segmentation to the gold standard. We report two settings on $k = 3$ and 4 respectively. On the one hand, the baseline method is demonstrated to be strong since they produce comparable results with NSP and Perplexity methods. On the other hand, the Clustering and Topic Detecting methods outperform the baselines. Topic Detecting produces the best score of 3.82 when $P = 3$, and 4.69 when $P = 4$. As such, we chose it as our segmentation method for this work.

5.5 Ablation Studies

Scripts: Long vs. Short On wikiHow, about 25% of the scripts are written in one whole segment without subgoals. We are interested to see whether our method can improve the generation of these scripts as well. We extend this question by navigating the impact of our solution on scripts of different lengths and with different number of subgoals. The dividing of the segments can vary according to the complexity of the goal, and the author's writing style — some authors prefer to write scripts all in one segment. We categorize the test set according to number of segments in ground truth scripts into "1", "2", "3", "4" and "5 or above", where the number of steps are averaged as 9.5, 13.3, 16.1, 20.8 and 27.9, respectively. We select ROUGE-L as a representative metric, and plot a graph of performance with respect to the number of segments in Figure 4. From the result, it is evident that the improvement is outstanding with long scripts (more segments) in test set. Overall, the baseline scripts showcase a downward trend as the number of segments increases, since decoding is often more challenging when the texts are longer – this is a common difficulty for long text generation. Our methods tackle this problem by taking a break in the middle and providing room for adjustment at decoding time with segmentation and subgoals. Consequently, hierarchical generation manifests a rising trend, especially with gold segment and subgoals. Another interesting phenomenon is that the performance improves, although not significantly, for the single segment scripts. Although the authors do not divide segments when composing scripts, the tasks may still inherently manifest a hierarchical structure, as solving a task naturally falls into different stages.

Subgoals: Are these Necessary Since the automatic evaluation showcases an improvement in the quality of generated scripts, we hereby further investigate if such improvement is caused by the segmentation only. We experiment by formatting the training data scripts with special token at position of a new segment, but not adding the subgoals. We explored two settings. The first setting uses gold segmentation, and the second setting uses the predicted segmentation. We evaluated the scripts using the metrics in § 5.1. The result in Figure 5 shows that under both settings, simply with a few special tokens separating the steps in training dataset, the generated text improves in quality. However,

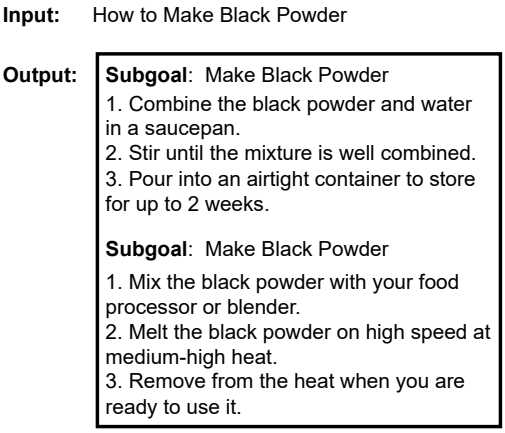

Figure 6: The "How to make black powder" script (from wikiHow)

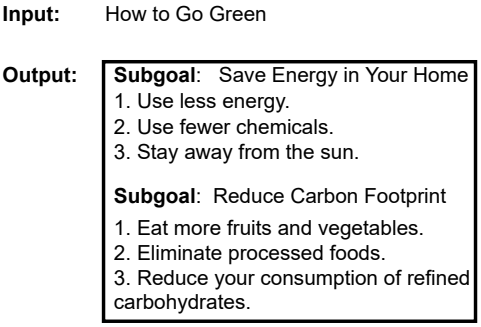

Figure 7: The "How to go green" script (from wikiHow)

the results are still worse than those with subgoals, explaining the significance of including subgoals in training data. In addition, for human, subgoals provide explainability for why models choose to make specific segments during script generation.

6 Case Study and Error Analysis

We present two example scripts generated using our method with the wikiHow dataset. Through these two examples, We analyze the common errors and the typical mistakes encountered. We put two more example scripts from Instructables dataset in appendix E since they are rather long. These Instructables scripts manifest not only the common errors in this section but also a third typical mistake.

Repetitive Subgoals According to our observation, the pervasive problem is that the generated subgoals are repeats of one another or the goal. This error appears in examples in Figure 6. One cause of this error is the inaccurate segmentation in the training dataset, which raises the difficulty of the subgoal prediction. Frequently, the subgoal labeled to a segment is no other but this segment's main goal. The reason is that the segment is a fraction of the script used as input when training the subgoal predictor. Moreover, the frequent occurrence of repetitive subgoals in the training dataset may seem like a pattern for the generation model, generating more scripts with repetitive subgoals. A revised loss function that penalizes repetition among goals and subgoals is a possible solution.

Irrelevant or Low-Quality Steps Another mistake with the generated scripts resides in the quality of the steps. For instance, in Figure 6, the model possibly mistakes "black powder" for "black pepper" and generates steps related to cooking. This mistake could originate from the lack of weapon-related knowledge in the training dataset. Figure 7 shows the script of "go green," with two subgoals. Despite the reasonable subgoals generated, the steps under "reduce carbon footprint" are irrelevant. The correct interpretation of "go green" is about environmental-friendly measures, while the steps discuss "green and healthy lifestyle." Since both wikiHow and Instructables use pictures to supplement text descriptions, a multi-modal approach may reduce ambiguity in the goal interpretation.

7 Conclusion

This work studies a new task of hierarchical script generation. To facilitate the research on this task, we contribute a new dataset Instructables to supplement the existing wikiHow resource, gathering over 100,000 hierarchical scripts. We further propose a method to build the benchmark, which learns to segment steps before script generation and concludes the subgoal each segment represents. Then, we finetune a T5-base model using prompts that combine subgoals and steps with special tokens. Experiment results from both automatic and human evaluation show that scripts generated by our method are in better quality than the baseline. Meanwhile, the gap towards the performance upper-bound still indicates much room for improvement. In the future, we are interested in exploring the idea of hierarchical generation in dealing with paragraphs in document generation.

Acknowledgement

We appreciate the reviewers for their insightful comments and suggestions. Xinze, Yixin, and Aixin were supported under the RIE2020 Industry Alignment Fund – Industry Collaboration Projects

(IAF-ICP) Funding Initiative, as well as cash and in-kind contribution from the industry partner(s). Yixin is supported by the Singapore Ministry of Education (MOE) Academic Research Fund (AcRF) Tier 1 grant. Muhao Chen was supported by the National Science Foundation of United States Grant IIS 2105329, an Amazon Research Award and a Cisco Research Award.

Limitation

The dataset Intructables, same as wikiHow, has a two level hierarchy of goal-subgoals-steps. Such structure is because of the writing habits of human authors. Due to the lack of datasets with deeper nested hierarchy, our work does not investigate the cases when there are multiple levels of subgoals. In addition, this work focuses on the presentation of task and dataset, and does not explore the performance of more advanced language models on the task. Relevant studies can be conducted in future works.

Ethics Statement

Our method is capable of generating large number of hierarchical scripts. Learning knowledge from different sources of information, the model might be misused into generating unsafe contents upon asking inappropriate questions. For now this seems unlikely since there is no offensive content in out collected dataset.

References

Alessandro Antonietti, Sabrina Ignazi, and Patrizia Perego. 2000. Metacognitive knowledge about problem-solving methods. *British Journal of Educational Psychology*, 70(1):1–16.

Matthew M Botvinick. 2008. Hierarchical models of behavior and prefrontal function. *Trends in cognitive sciences*, 12(5):201–208.

Snigdha Chaturvedi, Haoruo Peng, and Dan Roth. 2017. Story comprehension for predicting what happens next. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1603–1614, Copenhagen, Denmark. Association for Computational Linguistics.

Muhao Chen, Hongming Zhang, Haoyu Wang, and Dan Roth. 2020. What are you trying to do? semantic typing of event processes. In *Proceedings of the 24th Conference on Computational Natural Language Learning*, pages 531–542, Online. Association for Computational Linguistics.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Angela Fan, Mike Lewis, and Yann Dauphin. 2018. Hierarchical neural story generation. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 889–898, Melbourne, Australia. Association for Computational Linguistics.

Angela Fan, Mike Lewis, and Yann Dauphin. 2019. Strategies for structuring story generation. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 2650–2660, Florence, Italy. Association for Computational Linguistics.

David Herman. 1997. Scripts, sequences, and stories: Elements of a postclassical narratology. *PMLA/Publications of the Modern Language Association of America*, 112(5):1046–1059.

Zhiting Hu, Zichao Yang, Xiaodan Liang, Ruslan Salakhutdinov, and Eric P Xing. 2017. Toward controlled generation of text. In *International conference on machine learning*, pages 1587–1596. PMLR.

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

Nikolaos Lagos, Matthias Gallé, Alexandr Chernov, and Ágnes Sándor. 2017. Enriching how-to guides with actionable phrases and linked data. In *Web Intelligence*, volume 15, pages 189–203. IOS Press.

Brian Lester, Rami Al-Rfou, and Noah Constant. 2021. The power of scale for parameter-efficient prompt tuning. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 3045–3059, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Jiwei Li, Michel Galley, Chris Brockett, Jianfeng Gao, and Bill Dolan. 2016. A diversity-promoting objective function for neural conversation models. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 110–119, San Diego, California. Association for Computational Linguistics.

Chin-Yew Lin. 2004. ROUGE: A package for automatic evaluation of summaries. In *Text Summarization Branches Out*, pages 74–81, Barcelona, Spain. Association for Computational Linguistics.

Ximing Lu, Sean Welleck, Peter West, Liwei Jiang, Jungo Kasai, Daniel Khashabi, Ronan Le Bras, Lianhui Qin, Youngjae Yu, Rowan Zellers, Noah A. Smith, and Yejin Choi. 2022. NeuroLogic a*esque decoding: Constrained text generation with lookahead heuristics. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 780–799, Seattle, United States. Association for Computational Linguistics.

Ximing Lu, Peter West, Rowan Zellers, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. 2021. NeuroLogic decoding: (un)supervised neural text generation with predicate logic constraints. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 4288–4299, Online. Association for Computational Linguistics.

Qing Lyu, Li Zhang, and Chris Callison-Burch. 2021. Goal-oriented script construction. In *Proceedings of the 14th International Conference on Natural Language Generation*, pages 184–200, Aberdeen, Scotland, UK. Association for Computational Linguistics.

Ashutosh Modi and Ivan Titov. 2014. Inducing neural models of script knowledge. In *Proceedings of the Eighteenth Conference on Computational Natural Language Learning*, pages 49–57, Ann Arbor, Michigan. Association for Computational Linguistics.

Daniel Müllner. 2011. Modern hierarchical, agglomerative clustering algorithms. *arXiv preprint arXiv:1109.2378*.

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

Paolo Pareti, Benoit Testu, Ryutaro Ichise, Ewan Klein, and Adam Barker. 2014. Integrating know-how into the linked data cloud. In *International Conference on Knowledge Engineering and Knowledge Management*, pages 385–396. Springer.

Baolin Peng, Chunyuan Li, Jinchao Li, Shahin Shayandeh, Lars Liden, and Jianfeng Gao. 2021. Soloist: Building task bots at scale with transfer learning and machine teaching. *Transactions of the Association for Computational Linguistics*, 9:807–824.

Karl Pichotta and Raymond J. Mooney. 2016. Using sentence-level LSTM language models for script inference. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 279–289, Berlin, Germany. Association for Computational Linguistics.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, Peter J Liu, et al. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *J. Mach. Learn. Res.*, 21(140):1–67.

Nils Reimers and Iryna Gurevych. 2019. Sentence-bert: Sentence embeddings using siamese bert-networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics.

Fei Wang, Zhewei Xu, Pedro Szekely, and Muhao Chen. 2022. Robust (controlled) table-to-text generation with structure-aware equivariance learning. *arXiv preprint arXiv:2205.03972*.

Liang Yao, Chengsheng Mao, and Yuan Luo. 2019. Kg-bert: Bert for knowledge graph completion. *arXiv preprint arXiv:1909.03193*.

Hongming Zhang, Muhao Chen, Haoyu Wang, Yangqiu Song, and Dan Roth. 2020a. Analogous process structure induction for sub-event sequence prediction. *arXiv preprint arXiv:2010.08525*.

Jiaje Zhang and Donald A Norman. 1994. Representations in distributed cognitive tasks. *Cognitive science*, 18(1):87–122.

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, Online. Association for Computational Linguistics.

Tianran Zhang, Muhao Chen, and Alex AT Bui. 2020c. Diagnostic prediction with sequence-of-sets representation learning for clinical events. In *International Conference on Artificial Intelligence in Medicine*, pages 348–358. Springer.

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

Shuyan Zhou, Li Zhang, Yue Yang, Qing Lyu, Pengcheng Yin, Chris Callison-Burch, and Graham Neubig. 2022. Show me more details: Discovering hierarchies of procedures from semi-structured web data. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2998–3012, Dublin, Ireland. Association for Computational Linguistics.

A Comparison between wikiHow and Instructables

In this section we compare the features and statistics between dataset Instructables and wikiHow in detail.

The two datasets are different in multiple aspects. In terms of content, Instructables includes innovative thoughts on building concrete objects (e.g., *toy rocket*), While wikiHow incorporates daily-life experiences for possibly abstract concepts (e.g., *live healthily*). In terms of language style, Instructables is subjective (e.g., *I built it with ...*), and wikiHow is relatively objective with the use of imperatives (e.g., *Take a good sleep*). Regarding domain, Instructables involves six domains like circuits and craft, while wikiHow spans over 19 domains like arts and sports.

For Instructables dataset, on average, there are 5.2 subgoals for each script, 2.6 steps per subgoal, and 18.1 words per step. We also collect this statistics for wikiHow dataset, which includes 112,451 scripts, 278,680 subgoals, 2,057,088 steps, and 12,281,074 words. For wikiHow, there are 2.5 subgoals for each script, 7.4 steps per subgoal, and 6.0 words per step on average. The average sentence length of Instructables is much longer due to its narrative-based language style. Compared to wikiHow, which focuses on daily-life experiences, Instructables is more challenging since many items to build are highly professional and complicated (i.e., a Blind Assist Ultrasonic Navigator), which also explains the reason for the large average number of subgoals per script.

B Segmentation Methods

In this section, we formally explain the algorithms and implementations of each segmentation method in detail.

B.1 Next Sentence Prediction

We separate two consecutive steps if their continuity is predicted as negative via next sentence prediction — the two steps are talking about different topics. Given a list of ordered steps, we concatenate every two consecutive steps as *[CLS]step1[SEP]step2[SEP]* and calculate the probability score using BERT-base (Devlin et al., 2019) model. Specifically, we assume that a higher probability score indicates that the latter step is more rational than the previous one. We determine K lowest probability scores corresponding to K segmentation points. In experiments, we heuristically find the best K between 2 to 3.

Algorithm 1 Finding segmentation points with topic detecting. N is the number of steps in the script. x and y are start and end positions of subset. S is the list of segmentation points we look for.

Require: $N \geq 3$ ▷ At least 3 steps in a script
  $x \leftarrow 0$
  $y \leftarrow 2$
  $S \leftarrow list$
  while $y < N$ **do**
    if $topicNumber(x, y) < 2$ **then**
      $y \leftarrow y + 1$
    else if $topicNumber(x, y) \geq 2$ **then**
      $S \leftarrow y - 2$
      $x \leftarrow y - 1$
      $y \leftarrow y + 1$
    end if
  end while

B.2 Perplexity

Another approach is to measure the plausibility of a list of steps with **perplexity**. Assume that for a list of steps [S_x to S_y], the gold segment position is between S_i and S_{i+1} ($x < i < y$), separating the list into two segments [S_x to S_i] and [S_{i+1} to S_y]. The perplexity of [S_x to S_{i+1}] should be greater than that of [S_x to S_i] since an additional sentence not belonging to the segment makes it less natural. Similarly, the perplexity of [S_i to S_y] should be greater than that of [S_{i+1} to S_y]. We iterate from i = 0 to $i = N - 1$ (number of steps), and mark the i as a segmentation point if it satisfies both perplexity requirements.

B.3 Agglomerative Clustering

Instead of looking for segmentation points, we apply hierarchical agglomerative clustering (HAC) (Müllner, 2011) to group steps based on their sentence embeddings using SentenceBert (Reimers and Gurevych, 2019). Specifically, we merge two steps if their euclidean distance falls below a threshold while maintaining variance within all clusters minimized. Since HAC does not guarantee consecutive steps in the same cluster (e.g., if steps 1,2,4,5 are in cluster A, step 3 could be in step B), we make adjustment by recursively sending each step to the cluster with most of its neighbours, and sort the steps in the end.

B.4 Topic Detecting

While NSP compares topics locally between 2 steps, we design this method to detect topics glob-

ally among multiple steps. As shown in algorithm 1, starting with the first two steps, we add one step each time. A segmentation point is marked before the new step if more than one topic is detected. The topic detecting is implemented using fastclustering[5], which calculates cosine-similarity scores among the steps based on their sentence embeddings. Assuming that steps that share a topic have higher similarity scores, steps are assigned to the same community if their scores are above a threshold. In practice, we find 0.65 a reasonable threshold.

C Model Configuration

We fine-tune the T5 model from the Hugging-Face service[6]. We use Adam (Kingma and Ba, 2014) for optimization with the learning rate of 1e-4. We set the batch size 16 to fit the memory of one NVIDIA Tesla v100 GPU. The number of epochs is limited to 3 for models to converge within a reasonable running time. Training takes around 6 hours to finish on wikiHow and 12 hours on Instructables. We choose the model with the highest development performance to be evaluated on the test set.

D Human Evaluation Details

In this section, we explain in detail our human evaluation settings. For step evaluation, each question provides a goal and two scripts, asking the annotator which script better achieves the goal. Three options are provided, "A is better", "B is better", and "Not sure". Note that we flatten the script generated by our method and randomize the positions (A or B) of scripts in different questions to prevent the annotators from possibly identifying which script is ours from non-content information. For subgoal evaluation, we evaluate the association between 1) goal and subgoal and 2) subgoal and step. For goal-subgoal evaluation, each question provides a goal and a list of subgoals generated, asking the annotators if the subgoals are valid components of the goal and assist in achieving the goal. For subgoal-step evaluation, each question provides a goal, a subgoal, and a list of steps, asking the annotators if the subgoal is representative of the steps considering the goal.Three options are provided, "Yes", "No", and "Can't decide".

[5] https://www.sbert.net/examples/applications/clustering/README.html

[6] https://huggingface.co/t5-base

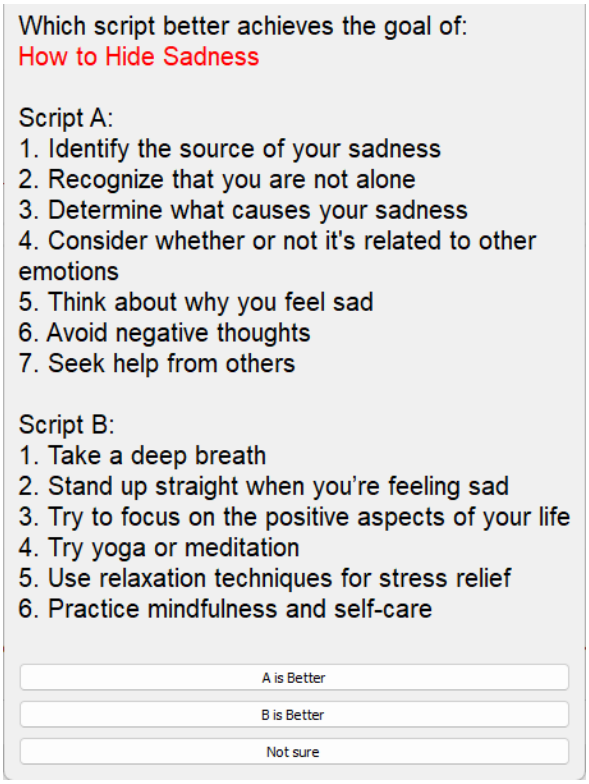

Figure 8: Screenshot of a question shown to the annotators, asking them to select the script that achieves the goal better from the two.

For each criterion (step, goal-subgoal, and subgoal-step), we randomly generate 100 questions from the test set of each dataset (wikiHow, Instructables), giving a total of 600 questions. We employ four human annotators in total. All annotators are graduate students and native or proficient speakers of English. All annotators possess adequate real-life knowledge and experiences to make reasonable judgments about the provided goals and have no potential conflicts of interest in this work. Each set of questions is answered by two different annotators, and for any disagreement, a third annotator will provide the final answer.

We hereby present the screenshots of the human evaluation questions. Figure 8 corresponds to the questions which compare the script generated by our method with the baseline. Figure 9 and 10 correspond to the questions which evaluate the quality of generated subgoals

Since each question is answered by two annotators, we report Inter Annotator Agreement (IAA) by providing the number of questions (out of 100) that a third annotator is not required. For the question "Which script better achieves the goal", the IAA are 72 for wikiHow and 81 for Instructa-

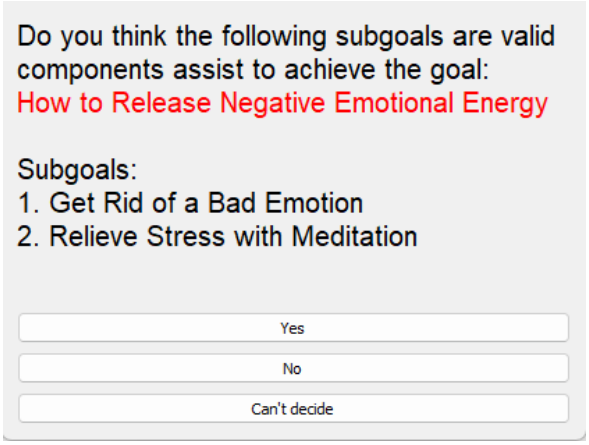

Figure 9: Screenshot of a question shown to the annotators, asking them to judge if the subgoals generated are valid components of the given goal, and are helpful in achieving the goal.

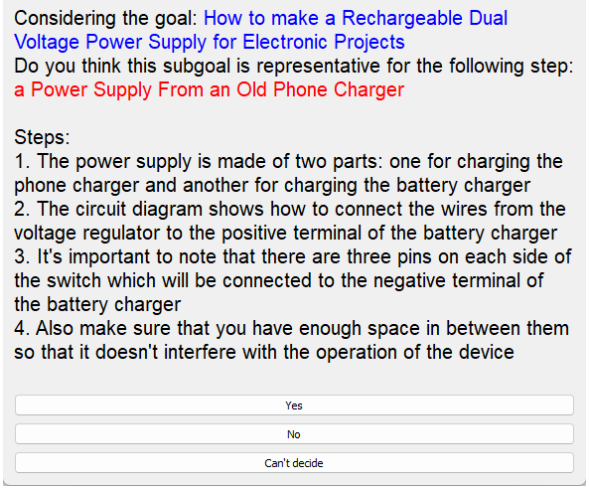

Figure 10: Screenshot of a question shown to the annotators, asking them to judge if the generated subgoals are representative of the steps, considering the provided goal.

bles. For the question "Are the subgoals helpful to achieve the goal", the IAA are 78 for wikiHow and 50 for Instructables. For the question "Is the subgoal representative of the steps", the IAA are 69 for wikiHow and 70 for Instructables.

E More Case Studies

In this section we further analyze two hierarchical scripts generated using our method from Instructables dataset.

The example in Figure 11 also manifests the two common errors presented in § 6, **repetitive subgoals** and **low-quality steps**. To elaborate, the steps ambiguously mention the concepts of processing the woods, painting, and assembling the

Input: How to Make Wooden Toy Trucks

Output:

Subgoal: a Wooden Toy Truck
1. The first step is to cut the wood into pieces.
2. You can use any size you want but I used 2x4's for this project.
3. Once you have all of your pieces cut out, it's time to glue them together.
4. Now that everything is dry, we are going to start painting!
5. We will be painting the sides of the trucks with white paint and then adding some more details on each side so they don't look too messy.
6. After applying the paint, let it dry completely before moving onto the next step.

Subgoal: a Wooden Truck
1. This part is pretty simple: just take one piece of wood and put it in between the two pieces of wood.
2. Make sure not to cover the whole thing or else it won't hold its shape well.
3. Next, add another piece of wood around the top of the truck where the wheels go.
4. It should now look like the picture above.

Figure 11: The "How to make wooden toy trucks" script (from Instructables)

wheel and truck body. However, the overall quality of the steps are rather low and does not lead to the fulfillment of the goal. A goal like this with complicated building steps may still be challenging for models at T5-base level.

Another problem observed is **inaccurate segmentation or subgoal**. Each subgoal is supposed to be a good summary of the corresponding steps. The example in Figure 12 shows an incorrect subgoal, "Yummy Eggs", while the steps are about preparation. The "egg" possibly comes from the keyword "breakfast" at decoding time. The second subgoal is not perfect either, whereby the last three steps discuss the idea of "baking" instead of "making pizza crust"; hence they should be separated into a third segment. This error manifests as an area of improvement in models' ability to understand the big picture of the event. It might be helpful to revise the decoding strategy for beam search in the future. By adding heuristics (Lu et al., 2022) in beam search, it is possible to improve the constrained generation process to look into the goal globally instead of locally.

F More Qualitative Examples

Aside from the examples shown in section appendix E, We provide 3 examples scripts generated and compare them with the ground truth. The outputs on the left hand side are generated with our method and the scripts on the right hand side are

Input: How to Make Breakfast Pizza

Output:

Subgoal: Yummy Eggs
1. You will need: 1 cup of flour 2 tablespoons of baking powder 1/2 teaspoon of salt 3/4 cup of water (I used my hands) Salt to taste Pinch of pepper
2. To make the dough you will need a large bowl, a small mixing bowl, an electric mixer, a pizza cutter, some scissors

Subgoal: Easy Pizza Crust
1. Add your eggs and mix them together.
2. I like to use an electric mixer because it is easier to work with.
3. It's also easy to do when you are making this crust.
4. If you don't have one, just put some on top of each other so that they won't get stuck in the pan.
5. This makes for a nice crisper crust.
6. Now let's start by adding our toppings; cheese, tomato sauce, or whatever else you would like.
7. For mine I added about 14 cup of milk at a time.
8. Once everything is mixed up we can begin cutting into slices.
9. When all the pieces are cut out take off the foil from the oven and place onto a plate.
10. After placing the slicer over the pizza then flip it over and repeat steps 2-5 until you reach the desired thickness.
11. Take care not to burn yourself while doing this step.

Figure 12: The "How to make breakfast pizza" script (from Instructables)

the gold references composed by human authors on wikiHow and Instructables websites respectively. Please see Figure 13 and 14 for wikiHow, and Figure 15 for Instructables.

Input: How to Clean a Cloudy Mirror

Generated Output:

Subgoal: Clean a Mirror with Vinegar
1. Mix 1 cup (240 ml) of vinegar and 2 cups (470 ml) of water.
2. Scrub the mirror with your fingernail until it is completely dry.
3. Rinse off the soapy solution using warm water.
4. Wipe down the mirror with a clean cloth to remove any dirt or debris.

Subgoal: Remove Stains from a Mirror
1. Use baking soda as an alternative to bleach for stained mirrors.
2. Use lemon juice in place of dishwashing detergent instead of alcohol.
3. Try apple cider vinegar on glass-colored mirrors when you want to keep them fresh.

Input: How to Clean a Cloudy Mirror

Gold Reference:

Subgoal: Clean a Mirror with Rubbing Alcohol
1. Pour rubbing alcohol into a dish and get out cotton pads.
2. Scrub the rubbing alcohol over cloudy areas or grime.
3. Clean the mirror with glass cleanser, shaving cream, or vinegar.

Subgoal: Clean a Mirror with Vinegar
1. Set a bowl of vinegar and a microfiber cloth near the mirror.
2. Dip the cloth into the vinegar and rub it over the surface of the mirror.
3. Wipe the mirror with water and glass cleanser to remove the smell.

Subgoal: Clean a Mirror with Shaving Cream
1. Squirt shaving cream on a microfiber cloth.
2. Rub the shaving cream over the surface of the mirror.
3. Wipe off the shaving cream using a clean microfiber cloth.

Figure 13: The "How to clean a cloudy mirror" script (from wikiHow)

Input: How to Have Flawless Skin in a Week

Generated Output:

Subgoal: Get Rid of Dry Skin
1. Use a moisturizer.
2. Use an exfoliator.
3. Apply sunscreen to your skin.

Subgoal: Prevent Acne on Your Face
1. Wash your face twice per day.
2. Avoid using harsh chemicals or products that can cause acne.
3. Stay away from alcohol and caffeine.

Input: How to Have Flawless Skin in a Week

Gold Reference:

Subgoal: Get Rid of Acne on Your Face
1. Determine your skin type.
2. Invest in a cleanser, toner, moisturizer and facial scrub suited to your skin type.
3. Buy an SPF 15+ sunscreen for daily use.
4. Use your face wash every day.
5. Wash off your makeup.

Subgoal: Get a Healthy Body
1. Eat properly.
2. Drink plenty of fluids.
3. Exercise.
4. Sleep.

Figure 14: The "How to have flawless skin in a Week" script (from wikiHow)

Input: How to Make Heart Themed Gift Basket With a Mirror As a Gift

Generated Output:

Subgoal: A Heart
1. The first thing you need to do is cut out the heart shape.
2. I used a ruler and then traced it onto my paper.
3. You can also use a pencil or something else that will be easier to work with.
4. Once you have all of your pieces cut out, start cutting them out.
4. If you don't know how to do this, just look at the picture above for help. 5. After you have done that, take your scissors and cut off the top part of the heart.
6. This step is pretty self-explanatory.

Subgoal: A Heart with Mirror
1. First, draw some lines on the back of the heart using an exacto knife.
2. Next, put glue on the inside of the heart so there is no gap between the mirror and the heart.
3. Make sure not to get glue on the outside of the heart.
4. When you are done, carefully place the center of the heart in the middle of the frame.
5. Take your time! It may seem like they aren't perfect but when you finish making these, you should end up with what looks like a real heart.
6. For me, mine was about 1/8" wide by 3/16".
7. To add more details, simply follow the pictures below.

Subgoal: With a Heart Basket From Scratch
1. Use any kind of painter's tape (I used spray paint) to cover the entire front of the heart.
2. Don't forget to leave enough space around the bottom of the heart as well. 3. Just keep adding small amounts until you reach the desired height.
4. Finally, attach the basket to the wall where you want it to go.
5. And voila! You finished! Congratulations!

Input: How to make Heart Themed Gift Basket With a Mirror As a Gift

Gold Reference:

Subgoal: Cardboard Basket
1. Take a piece of cardboard and cut a circle from it.
2. Take another piece of cardboard to make the walls of the basket.
3. Make cuts on the bottom of the cardboard and stick it on the bottom of the base.
4. You can use tape to re-enforce the base.
5. Make holes for handle on the walls

Subgoal: Yarn Basket
1. Make heart cut-outs and stick them on the basket
2. You can also use stickers and washi tape.
3. To make the handle, take yarn and measure how long you want it to be.
4. Make it a little longer than required.
5. Cut enough pieces of yarn to make it into 3 groups and braid it.
6. Put it through the holes on the basket and tie knots to make sure it doesn't slip off.

Subgoal: A Gift Basket and a Personalized Mirror
1. Take your thermacol and cut out the shape and make hole in the middle for the mirror.
2. Stick the cardboard on the back.
3. Decorate the base with colour paper, washi tape, and stickers.
4. Stick the mirror in the space. Put it all together.
5. You can add confetti to make it more decorative.
6. And there you go, you have made a gift basket and a customized mirror perfect for gifting!

Figure 15: The “How to make heart themed gift basket with a mirror as a gift” script (from Instructables)

ACL 2023 Responsible NLP Checklist

A For every submission:

☑ A1. Did you describe the limitations of your work?
9 Limitation

☑ A2. Did you discuss any potential risks of your work?
10 Ethics Statement

☑ A3. Do the abstract and introduction summarize the paper's main claims?
Abstract

☒ A4. Have you used AI writing assistants when working on this paper?
Left blank.

B ☑ Did you use or create scientific artifacts?

2.2 Dataset

☑ B1. Did you cite the creators of artifacts you used?
2.2 Dataset

☑ B2. Did you discuss the license or terms for use and / or distribution of any artifacts?
2.2 Dataset

☑ B3. Did you discuss if your use of existing artifact(s) was consistent with their intended use, provided that it was specified? For the artifacts you create, do you specify intended use and whether that is compatible with the original access conditions (in particular, derivatives of data accessed for research purposes should not be used outside of research contexts)?
2.2 Dataset

☑ B4. Did you discuss the steps taken to check whether the data that was collected / used contains any information that names or uniquely identifies individual people or offensive content, and the steps taken to protect / anonymize it?
2.2 Dataset

☑ B5. Did you provide documentation of the artifacts, e.g., coverage of domains, languages, and linguistic phenomena, demographic groups represented, etc.?
Appendix A

☑ B6. Did you report relevant statistics like the number of examples, details of train / test / dev splits, etc. for the data that you used / created? Even for commonly-used benchmark datasets, include the number of examples in train / validation / test splits, as these provide necessary context for a reader to understand experimental results. For example, small differences in accuracy on large test sets may be significant, while on small test sets they may not be.
4.1 Experiment Set Up

C ☑ Did you run computational experiments?

4 Experiments

☑ C1. Did you report the number of parameters in the models used, the total computational budget (e.g., GPU hours), and computing infrastructure used?
Appendix C

The Responsible NLP Checklist used at ACL 2023 is adopted from NAACL 2022, with the addition of a question on AI writing assistance.

☑ C2. Did you discuss the experimental setup, including hyperparameter search and best-found hyperparameter values?
4.1 Experiment Set Up

☑ C3. Did you report descriptive statistics about your results (e.g., error bars around results, summary statistics from sets of experiments), and is it transparent whether you are reporting the max, mean, etc. or just a single run?
4.2 Automatic Evaluation

☑ C4. If you used existing packages (e.g., for preprocessing, for normalization, or for evaluation), did you report the implementation, model, and parameter settings used (e.g., NLTK, Spacy, ROUGE, etc.)?
2.2 Dataset

D ☑ Did you use human annotators (e.g., crowdworkers) or research with human participants?

Appendix D

☑ D1. Did you report the full text of instructions given to participants, including e.g., screenshots, disclaimers of any risks to participants or annotators, etc.?
Appendix D

☑ D2. Did you report information about how you recruited (e.g., crowdsourcing platform, students) and paid participants, and discuss if such payment is adequate given the participants' demographic (e.g., country of residence)?
Appendix D

☑ D3. Did you discuss whether and how consent was obtained from people whose data you're using/curating? For example, if you collected data via crowdsourcing, did your instructions to crowdworkers explain how the data would be used?
Appendix D

☑ D4. Was the data collection protocol approved (or determined exempt) by an ethics review board?
Appendix C

☑ D5. Did you report the basic demographic and geographic characteristics of the annotator population that is the source of the data?
Appendix D