Take a Break in the Middle: A Hierarchical Generation Approach for Script Knowledge Acquisition

Anonymous ACL submission

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 a hierarchical generation approach, and set up evaluation metrics. In experiments, both automatic and human evaluation verify the high-quality of dataset and the effectiveness of our proposed method, as well as the correctness of the paper’s main claim — the generated script becomes better, even using a simple strategy of taking a break between subgoals in the middle of sequential generation. Further ablation study also provides valuable insights for future research. Codes and data will be released later.

1 Introduction

Scripts are forms of knowledge representations that model ordered events or actions for particular goals (Salveter, 1982; Herman, 1997). As shown in Figure 1, to obtain a Ph.D degree (i.e., goal), one shall follow specific events\(^1\) step-by-step, including do the research, write research 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), diagnostic prediction (Zhang et al., 2020c), and biological process comprehension (Berrant et al., 2014). Therefore, the task of script generation is proposed to automatically generate events given any goal (Lyu et al., 2021).

Existing works typically assume that all events are sequential, while we argue that such linear generation is far from enough for comprehensively acquiring script knowledge. 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 such as publish research paper, passing the qualification exam, and defending the thesis. (Figure. 1). The subgoal publishing research paper thereof consists of more fine-grained steps such as do the research, write the paper, and pass peer review. Accordingly, a proper way of acquiring script knowledge should also hierarchically capture different levels of task subgoals.

In this work, we extend the task, contribute a

\(^1\)Event denotes a short sentence in this area.
new dataset, and build the benchmark with a novel hierarchical generation approach for goal-oriented script generation. The extended task of hierarchical script generation brings along the challenge of acquiring knowledge about the internal structure of a script. Given a goal, instead of generating in a linear manner, our task seeks to generate scripts in two levels. The first level consists of subgoals, and the second level is detailed steps, where each subgoal contains several steps. Note that subgoals are not necessarily sequential but can be achieved in parallel. Such a new setting not only accords to people’s cognitive process (Antoni et al., 2000) but also provides a more flexible way of content organization than pure linearity, offering another possibility in dealing with topic shift and expansion. It will thus benefit natural language generation, especially for long documents.

Existing script generation tasks mostly used the dataset wikiHow (Koupaee and Wang, 2018; Zhang et al., 2020b), collected from wikiHow website\(^2\), an online wiki-style community containing how-to guideline articles. To verify the generalizability of the proposed methods over more than one dataset, we construct a new dataset “Instructables” based on Instructables website\(^3\), a community for users to share their D.I.Y projects. 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.

To tackle the task, we propose a novel hierarchical generation approach. The first challenge is how to deal with the lack of subgoals, if the training data is in a flat structure in practice. To address this problem, we deploy a segmentation method that automatically separates steps in training data, and generates their subgoals by reusing the data in a dual learning way. Another challenge is that text generation is, by nature, sequential instead of hierarchical. Therefore, it is difficult for models to generate texts directly at multiple levels without seeing the whole picture at decoding time. Against this issue, we leverage the prompt tuning (Lester et al., 2021) method to preserve structure information via special tokens. By incorporating the goal, subgoals, and steps in one template, we allow the model to take a break to conclude each subgoal before generating the succeeding steps. This method leads to an interleaved generation process where each subgoal is generated before its steps guide their decoding.

We evaluate our method on the quality of both the steps and the subgoals. For the steps, we conduct both automatic and human evaluations to judge if our method outperforms the baseline. For the subgoals, we conduct human evaluation to assess if the generated subgoals are valid to the goal and representative of the steps. The experiment result shows that our method generates hierarchically structured scripts with comparable or better quality than the scripts generated by baseline methods. In fact, given gold standard segmentation and subgoals, the improvement is more substantial, indicating space for improvement in our segmentation and subgoal labeling methods. Further, the ablation study shows that our method brings more improvement to longer scripts with more segments, which is the weakness of the baseline method.

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 events in a simple, flat manner for any goal. Besides, a few 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, instead of linking steps and goals with the retrieve-then-rerank approach, Our proposed task and model target 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 Decoding (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. Different from tasks like storytelling, script generation is not open-ended since the generated texts have to comply with common sense.

### 3 Task and Dataset

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

#### 3.1 Task Definition

The original goal-oriented script generation (Lyu et al., 2021) 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 $g$, it is to generate $L$ levels of events, where the events at the $1$ to $(L-1)$ levels are called subgoals ($s$) and the events at the $L$-th level are called steps ($t$). Within each level, the list of children events should fulfill their parent event. Note that the number of events at each level is not fixed, and the model is required to decide it 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. Thus, in the rest of the paper, we define $L = 2$.

#### 3.2 Dataset

We use two datasets for this task, wikiHow and Instructables. 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 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 — a website specializing in user-created and uploaded 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. Each project consists of a title showing the item that the author made (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 one step. The second stage is filtering. The raw data is inconsistent in text style, section format, article length, etc. We hereby carry out eight steps to filter out noisy data. We remove: 1) Non-English projects using Python library langdetect. 2) The section on Supplies or Materials, which includes items instead of events/actions. 3) The section counters (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 content in any section since they normally use figures or videos to illustrate their ideas. 6) Projects with any section overly lengthy. Many authors post stories or anecdotes to convey the rationale they came up with the project.
Table 1: Number of total scripts, subgoals, and steps in dataset Instructables by category

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

and we find 128 words a good threshold to filter them out. 7) Projects that build the same item as others. We remove repeated items after the first appearance to eliminate possible anomalies in data distribution. 8) We unify the format of project titles to make them consistent. By performing Part-of-Speech (POS) Tagging on titles, we prefix “How to” to verb 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).

Statistics Table 1 shows the statistics of Instructables. In total, we obtain 107,408 scripts, 560,079 subgoals, 1,478,669 steps, and 26,813,397 words. For wikiHow, there are 2.5 subgoals for each script, 7.4 steps per subgoal, and 6.0 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.

4 Methodology

In this section, we present our proposed method 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 can 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 will introduce the proposed framework for training. Finally, given a goal, we will 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 aim 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. 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. We explain these methods in detail in appendix A.

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, 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 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 the system does not peek at 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 tuning (Lester et al., 2021), we take advantage of PLMs to generate such a two-level structure
by re-formatting a prompt as follows:

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

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

```
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. According to the autoregressive 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, whereby all subgoals are generated first, followed by the steps. The prompt is as follows:

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

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

```
To learn Web Design, the subgoals are: Finding Web Design Resources, Mastering HTML... <section> Check online for web design courses and tutorials. Look into taking a class at a local college or university... <section> Familiarize yourself with basic HTML tags. Learn to use tag attributes...
```

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 an autoregressive 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 T5 model as input, with the prefix “Ask question:” as common practice for Question Answering tasks using the T5 model. 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 each dataset 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 script generated by our framework.

**Metrics** For automatic evaluation metrics, we first follow prior works (Lyu et al., 2021) without considering the hierarchy information and report perplexity and BERTScore (Zhang et al., 2019) for steps. 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.

**Baseline** Given the different nature of the hierarchical script generation in contrast to the previous task, there is not a directly applicable base-
With segmentation, we take a break in the middle. We report the results of hierarchical script generation with the highest development performance to be designed for a multilingual corpus.

Model Configuration We fine-tune the T5 model from the Hugging-Face service. 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.

5.2 Automatic Evaluation

We report the results 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.

We can observe that the results from both datasets are generally consistent. Our method has outperformed the baseline in three metrics Perplexity, BLEU-1, and ROUGE-L score, indicating the effectiveness of our method in generating scripts with higher quality. 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. Between the top-down and interleaving approaches, the latter has slightly better or tied scores among almost all metrics for both datasets. The subgoals are in proximity to the corresponding steps for the interleaving approach, which better guides the step generation. It is notable 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. 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 self-predicted segmentation and subgoals.

We also acknowledge the limitations of automatic evaluation. The reference script is not the only solution to the goal, and an output script different from the reference may achieve the goal in alternative means. Hence, we conduct human evaluations to complement automatic evaluation.

5.3 Human Evaluation

We conduct human evaluations to assess the quality of our scripts. We evaluate the scripts on both the steps 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 valid components of the task goal. In the context of the main goal, the second criterion concerns if the generated subgoal properly represents the associated steps. We provide more details of human evaluation in appendix B.

Step Evaluation From the results in appendix C, 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. In addition, we realize that the proportion of favored scripts is higher on the Instructables dataset than on wikiHow. Such results are very much due to the high-quality wikiHow scripts generated by the baseline method. Our method has a greater improvement on the Instructables dataset, similarly, attributes to the low-quality scripts from the baseline. In extreme cases, we observed empty or very short output 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 D).

Subgoal Evaluation Regarding the question of whether subgoals are helpful to achieve the goal,
Table 3: Segment distances of the proposed segmentation methods on wikiHow dataset.

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 a different 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 prove that it is still a challenging task to generate high-quality subgoals especially for Instructables dataset. A good area to work on is to improve the subgoals’ association with the goals.

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 segments in ground truth. In addition, we impose an penalty score $P = k \cdot d$ 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 most scripts in our datasets have a uniform number of steps in different segments. In addition, 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 proven 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 Study

Script: 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 navig...
gating the impact of our solution on scripts with different lengths and 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 number of segments in Figure 2. 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 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, our method 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.

**Subgoal: Is it Necessary** Since the automatic evaluation showcases an improvement in the quality of generated scripts, we here 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 according to ground truth, but not adding the gold subgoals. We evaluated the scripts using the metrics in § 5.1. The result in Figure 3 shows that simply with a few special tokens separating the steps in training dataset, the generated text improves in quality. However, the results are slightly worse than those with gold 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 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 thousand hierarchical scripts. We further propose a method to address this task, which learns to segment steps before script generation and concludes the subgoal each segment represents. Then, we prompt tuning T5-base model using templates that combine subgoals and steps with special tokens. Experiment results from both automatic and human evaluation show that scripts generated by our method is of better quality than the baseline in achieving the given goal. Meanwhile, the gap towards the performance upper-bound still indicates much room for improvement. Future work will investigate the method to improve the generation quality further. Some potential areas for exploration include designing decoding algorithms to enhance generation conditioned on the subgoals.


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.


A Segmentation Methods

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

A.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 \([CLS]\text{step}1[SEP]\text{step}2[SEP]\) 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 one before. We determine \(K\) lowest probability scores corresponding to \(K\) segmentation points. In experiments, we heuristically find the best \(K\) between 2 to 3.

A.2 Perplexity

Another approach is to measure the plausibility of a list of steps with perplexity. Assume that for a list of steps \([S_x \rightarrow S_y]\), the gold segment position is between \(S_i\) and \(S_{i+1}\) \((x < i < y)\), separating the list into two segments \([S_x \rightarrow S_i]\) and \([S_{i+1} \rightarrow S_y]\). The perplexity of \([S_x \rightarrow S_i]\) should be greater than that of \([S_x \rightarrow S_i]\) since an additional sentence not belonging to the segment makes it less natural. Similarly, the perplexity of \([S_i \rightarrow S_y]\) should be greater than that of \([S_{i+1} \rightarrow S_y]\). 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.

A.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.

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.

\[
\begin{align*}
\text{Require: } & \quad N \geq 3 \quad \triangleright \text{ At least 3 steps in a script} \\
& x \leftarrow 0 \\
& y \leftarrow 2 \\
& S \leftarrow \text{list} \\
\text{while } y < N \text{ do} \\
& \quad \text{if } \text{topicNumber}(x, y) < 2 \text{ then} \\
& & \quad y \leftarrow y + 1 \\
& \quad \text{else if } \text{topicNumber}(x, y) \geq 2 \text{ then} \\
& & \quad S \leftarrow y - 2 \\
& & \quad x \leftarrow y - 1 \\
& & \quad y \leftarrow y + 1 \\
& \quad \text{end if} \\
\text{end while}
\end{align*}
\]

A.4 Topic Detecting

While NSP compares topics locally between 2 steps, we design this method to detect topics globally 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\(^6\), 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.

B 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

\(^6\)https://www.sbert.net/examples/applications/clustering/README.html
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”.

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 4 corresponds to the questions which compare the script generated by our method with the baseline. Figure 5 and 6 correspond to the questions which evaluate the quality of generated subgoals.

C Human Evaluation Results

In this section, we show the results of human evaluation in stacked bar charts. Note that for the question where we ask annotators to select the better script (Figure 7), we allow “not sure” to be the final result after coordination since it is a comparison question. It is possible that both scripts accomplish the given goal equally well or neither script accomplishes the goal. For the two questions evaluating the subgoals (Figure 8 and 9), all answers have to be solid “Yes” or “No” after coordination.
D Case Study and Error Analysis

We present example scripts generated using our method with both wikiHow and Instructables datasets. We analyze some common errors and typical mistakes encountered.

D.1 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 10 and 11. 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 future solution.

D.2 Irrelevant or Low-Quality Steps

Another mistake with the generated scripts resides in the quality of the steps. For instance, in Figure 10, 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. In Figure 11, the steps ambiguously mention the concepts of processing the woods, painting, and assembling the 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 now. Figure 12 shows the script of “go green”, with two subgoals. Despite the reasonable subgoals generated, the steps under “reduce carbon footprint” are irrelevant. The correct
The second subgoal is not perfect either, whereby instead of “making pizza crust”; hence they should.

**Figure 11:** The “How to make wooden toy trucks” script (from Instructables)

<table>
<thead>
<tr>
<th>Input: How to Make Wooden Toy Trucks</th>
</tr>
</thead>
<tbody>
<tr>
<td>Output:</td>
</tr>
<tr>
<td><strong>Subgoal:</strong> a Wooden Toy Truck</td>
</tr>
<tr>
<td>1. The first step is to cut the wood into pieces.</td>
</tr>
<tr>
<td>2. You can use any size you want but I used 2x4’s for this project.</td>
</tr>
<tr>
<td>3. Once you have all of your pieces cut out, it’s time to glue them together.</td>
</tr>
<tr>
<td>4. Now that everything is dry, we are going to start painting!</td>
</tr>
<tr>
<td>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.</td>
</tr>
<tr>
<td>6. After applying the paint, let it dry completely before moving onto the next step.</td>
</tr>
<tr>
<td><strong>Subgoal:</strong> a Wooden Truck</td>
</tr>
<tr>
<td>1. This part is pretty simple: just take one piece of wood and put it in between the two pieces of wood.</td>
</tr>
<tr>
<td>2. Make sure not to cover the whole thing or else it won’t hold its shape well.</td>
</tr>
<tr>
<td>3. Next, add another piece of wood around the top of the truck where the wheels go.</td>
</tr>
<tr>
<td>4. It should now look like the picture above.</td>
</tr>
</tbody>
</table>

**Figure 12:** The “How to go green” script (from WikiHow)

<table>
<thead>
<tr>
<th>Input: How to Go Green</th>
</tr>
</thead>
<tbody>
<tr>
<td>Output:</td>
</tr>
<tr>
<td><strong>Subgoal:</strong> Save Energy in Your Home</td>
</tr>
<tr>
<td>1. Use less energy.</td>
</tr>
<tr>
<td>2. Use fewer chemicals.</td>
</tr>
<tr>
<td>3. Stay away from the sun.</td>
</tr>
<tr>
<td><strong>Subgoal:</strong> Reduce Carbon Footprint</td>
</tr>
<tr>
<td>1. Eat more fruits and vegetables.</td>
</tr>
<tr>
<td>2. Eliminate processed foods.</td>
</tr>
<tr>
<td>3. Reduce your consumption of refined carbohydrates.</td>
</tr>
</tbody>
</table>

**Figure 13:** The “How to make breakfast pizza” script (from Instructables)

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

The interpretation of “go green” is about environmentally-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 help to reduce ambiguity in the goal interpretation.

D.3 Inaccurate Segmentation or Subgoals

A relatively less frequent but not to be ignored problem is the formation of subgoals. Each subgoal is supposed to be a good summary of the corresponding steps. The example in Figure 13 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.

E More Qualitative Examples

Aside from the examples shown in section appendix D, 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 the gold references composed by human authors on WikiHow and Instructables websites respectively. Please see Figure 14 and 15 for WikiHow, and Figure 16 for Instructables.

F Risks

Our method is capable of generating large number of hierarchical scripts. Learning knowledge from different sources of information, the model might be misled into generating unsafe contents upon asking inappropriate questions. For now this seems unlikely since there is no offensive content in our collected dataset.
### How to Clean a Cloudy Mirror

**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.

**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.

Figure 14: The “How to clean a cloudy mirror” script (from wikiHow)

### How to Have Flawless Skin in a Week

**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.

**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 15: The “How to have flawless skin in a Week” script (from wikiHow)
How to Make Heart Themed Gift Basket
With a Mirror As a Gift

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.
5. If you don't know how to do this, just look at the picture above for help.
6. After you have done that, take your scissors and cut off the top part of the heart.
7. 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. Just keep adding small amounts until you reach the desired height.
3. Finally, attach the basket to the wall where you want it to go.
4. And voila! You finished! Congratulations!

Figure 16: The “How to make heart themed gift basket with a mirror as a gift” script (from Instructables)