Show Me More Details: Discovering Hierarchies of Procedures from Semi-structured Web Data

Shuyan Zhou‡* **Li Zhang**†* **Yue Yang**† **Qing Lyu**†

Pengcheng Yin‡ **Chris Callison-Burch**† **Graham Neubig**‡

‡Carnegie Mellon University † University of Pennsylvania

{shuyanzh,pcyin,gneubig}@andrew.cmu.edu

{zharry,yueyang1,lyuqing,ccb}@seas.upenn.edu

Abstract

Procedures are inherently hierarchical. To *"host a party"*, one may need to *"clean the house"*, which in turn may require *"putting away the clothes"*. While such hierarchical knowledge is critical for reasoning about complex procedures, most existing works treat procedures as shallow structures without modeling the hierarchical dependency between them. In this work, we attempt to construct an open-domain hierarchical knowledge-base (KB) of procedures based on wikiHow, a website containing more than $110k$ instructional articles, each documenting the steps to accomplish a complex procedure. To this end, we develop a simple and efficient method that links steps (*e.g.* *"clean the house"*) in an article to other articles with similar intents (*e.g.* *"how to deep clean your house"*), which proceeds recursively to form the KB. Our method significantly outperforms several strong baselines according to automatic evaluation, human judgment, and application to downstream tasks such as instructional video retrieval.[1]

1 Introduction

Procedural knowledge is the knowledge that describes *steps* needed to achieve a particular *goal*. It is inherently hierarchical: a high-level procedure is composed of many lower-level procedures. For example, a procedure with the goal *"host a party"* consists of steps like *"clean the house"*, *"send the invitation"*, and so on, where each step itself is a procedure as well. Such hierarchical relations between procedures are typically recursively defined, with lower-level procedures further decomposed into more fine-grained steps: one may have to *"put away the clothes"* and *"vacuum the floor"* in order to *"clean the house"*.

Although there has been increasing interest in discovering relations between procedures, the majority of works have focused on temporal (Pustejovsky et al., 2003; Ning et al., 2018), causal (Hashimoto et al., 2014; Caselli and Vossen, 2017) and spatial (Glavaš et al., 2014; Liu et al., 2014) relations, which are also called *horizontal* relations (Zhang et al., 2020a). Fewer works have attempted to discover the *vertical* relations between complex procedures and their fine-grained child steps. Those that do mostly perform a shallow one-level decomposition at best, and often require costly resources such as human expert annotation (Chu et al., 2017; Zhang et al., 2020a, 2021). One exception is Lagos et al. (2017), which links action phrases in one procedure to another in how-to articles. Such linking is crucial to provide detailed explanations of complex steps to a less knowledgeable audience.

In this paper, we revisit this important but understudied task to develop a simple and effective algorithm (Figure 1) to construct a hierarchical KB for over $110k$ complex procedures spanning a wide range of topics from wikiHow[2], a how-to website with a large number of instructional articles that has recently become a widely-used resource in NLP (Zhang et al., 2020d,c; Zhou et al., 2019; Zellers et al., 2019). We treat each wikiHow article[3] as the source of procedural knowledge, where we obtain the title of the article to represent the goal (*e.g.* g_1 in Figure 1), and a list of headline paragraphs of steps as the list of steps (*e.g.* $s_1 \ldots s_n$) following Zhang et al. (2020d). Next, we decompose the obtained steps by linking them with the goals expressing the same intent (*e.g.* s_1 to g_2). The steps of that goal are treated as the finer-grained steps of the obtained steps (B4). In this way, the procedure knowledge of

[*] Equal contribution.

[1] A demo with partial data is at wikihow-trees.web.app. The code and some sample data are attached to the submission. The complete web application, data, and code will be released upon paper acceptance.

[2] www.wikihow.com

[3] www.wikihow.com/Host-a-Good-Party

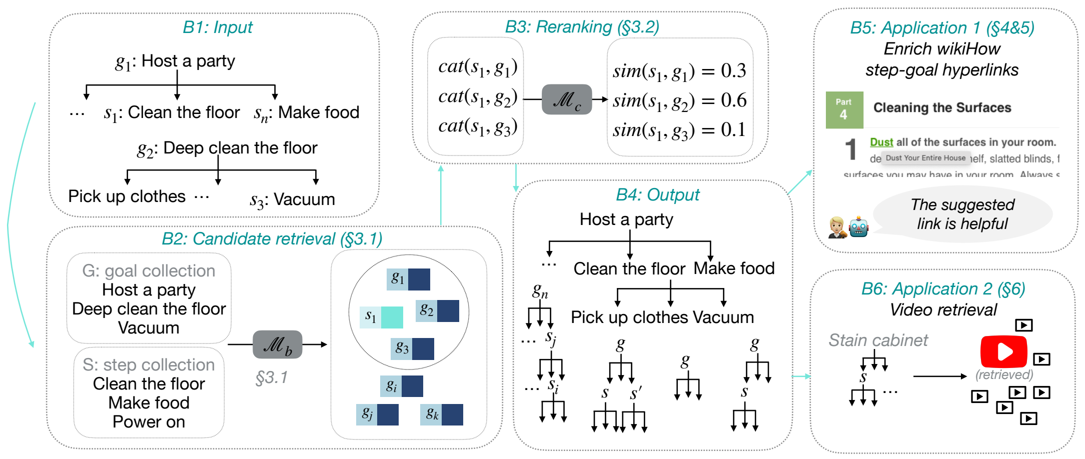

Figure 1: The overview of our proposed method. The input (Block1) and output (B4) of the hierarchy discovery model (B2, B3) and the applications (B5, B6) of the hierarchical knowledge base.

goals goes from shallow hierarchies (B1) to deeper hierarchies. To link steps with goals, our hierarchy discovery model (§3) first independently encodes each step and goal in wikiHow and searches the k nearest goals of similar semantics for each step (B2). Then, it applies a joint encoder to calculate the similarity score between the step and each candidate goal (B3). This pipeline can efficiently search over a large candidate pool while accurately measuring the similarity between steps and goals.

To assess the downstream utility of discovering hierarchies of procedures, we discuss several potential use cases and propose corresponding evaluation metrics, both automatic and manual. First, the discovered hierarchies could be straightforwardly used to complete missing step-goal hyperlinks in wikiHow, which are currently curated manually by the community (B5). In §4, we automatically evaluate the performance of our proposed method on step-goal linking and find that our method outperforms a few strong baselines, including a lexical overlap method analogous to Lagos et al. (2017), by significant margins. In §5, we ask the crowdworkers to indicate whether the decomposed, finer-grained steps are helpful to assist accomplish a step and find that our method produces much more helpful decompositions than the baselines. Finally, our hierarchy could bridge the abstract natural language description and the concrete executions of a procedure. We use video retrieval as a demo extrinsic evaluation task (§6), and observe that queries that encode deeper hierarchies are better at retrieving instructional videos than those without.

2 Problem Formulation

We represent a procedure as a tree where the root node n represents a goal and its children nodes $\texttt{Ch}(n)$ represent the steps of n. We formulate the hierarchy discovery task as first identifying the steps among $\texttt{Ch}(n)$ that can themselves be a goal of some other finer-grained steps (sub-steps), and then inserting these sub-steps into the tree.

While this formulation could potentially be used on any large collection of described procedures, we specifically focus on wikiHow, a website containing over $110,000$ how-to articles describing complex procedures. As in B1 of Figure 1, each article comprises a goal (g) represented by the title of the article, and a series of steps ($\texttt{Ch}(g)$) represented by the headlines of paragraphs describing steps. Therefore, each article forms a procedure tree of depth one.

We denote the collection of all goals and steps in wikiHow as G and S respectively. Our hierarchy discovery aims to link a step $s_i \in S$ to a goal $g \in G$ such that g corresponds to the same procedure as s_i. It then treats $\texttt{Ch}(g)$ as $\texttt{Ch}(s_i)$. Given that g and s_i are both represented by textual descriptions, the discovery process can be framed as a *paraphrase detection task*, as detailed below. This discovery process can be applied recursively on the leaf nodes until the resulting leaf nodes reach the desired granularity, effectively growing a hierarchical procedure tree, *e.g.* B4 of Figure 1.

3 Hierarchy Discovery Model

For each of the 1.5 million steps in the wikiHow corpus, we aim to select one goal that expresses the same procedure as the step from over $110k$ goals.

We propose a simple and efficient method to deal with such a large search space through a two-stage process. First, we perform *search*, encoding each step and goal *separately* and select the k most similar goals for each step s. Second, we perform *re-ranking*, *jointly* encoding a step with each of its candidate goals to allow for more expressive contextualized embeddings. The goal with the highest similarity score is selected and the step is expanded accordingly, as in B4 of Figure 1. Previous works on entity linking (Wu et al., 2019) and information retrieval (Humeau et al., 2019) have adopted similar methods and achieved strong performance.

3.1 Candidate Goal Retrieval

In the first stage, we independently encode each step $s \in S$ and goal $g \in G$ with a model $\mathcal{M}_b$, resulting in embeddings $\boldsymbol{e}_{s_1}$, $\boldsymbol{e}_{s_2}$, ..., $\boldsymbol{e}_{s_n}$ and $\boldsymbol{e}_{g_1}$, $\boldsymbol{e}_{g_2}$, ..., $\boldsymbol{e}_{g_n}$. The similarity score between s and g is calculated as the cosine similarity between $\boldsymbol{e}_s$ and $\boldsymbol{e}_g$. We denote this first-stage similarity score as $\text{sim}_1(s, g)$. Using this score, we can obtain the top-k most similar candidate goals for each step s, and we denote this candidate goal list as $\texttt{C}(s) = [g_1, ..., g_k]$. To perform this top-$k$ search, it is possible to use efficient similarity search libraries such as FAISS (Johnson et al., 2017).

We instantiate $\mathcal{M}_b$ with two learning-based paraphrase encoding models. The first is the SP model (Wieting et al., 2019, 2021), which encodes a sentence as the average of the sub-word unit embeddings generated by SentencePiece (Kudo and Richardson, 2018). The second is SBERT (Reimers and Gurevych, 2019), which encodes a pair of sentences with a siamese BERT model that is finetuned on paraphrase corpus. For comparison, we additionally experiment with using search engines as $\mathcal{M}_b$, specifically Elasticsearch with the standard BM25 weighting metric (Robertson and Zaragoza, 2009) where we index each article with its title only or with its full article and the Bing Search API where we limit the search to wikiHow website only[4]. The BM25 with the former setting resembles the method proposed by Lagos et al. (2017).

3.2 Reranking

While efficient, encoding steps and goals independently is likely sub-optimal as information in the steps cannot be used to determine the encoding of the goals and vice-versa. Therefore, we concatenate a step with each of its top-k candidate goals in $\texttt{C}(s)$ and feed them to a model $\mathcal{M}_c$ that jointly encodes each step-goal pair. Concretely, we follow the formulation of Wu et al. (2019) to construct the input of each step-goal pair as:

`[CLS]` *ctx* `[ST]` *step* `[ED]` *goal* `[SEP]`

where `[ST]` and `[ED]` are two reserved tokens in the vocabulary of a pretrained model, which mark the location of the step of interest. *ctx* are the context for a step (*e.g.* its surrounding steps or its goal) that could provide additional information. The hidden state of the `[CLS]` token is taken as the final contextualized embedding. The second-stage similarity score is calculated as follows:

$$\text{sim}_2(s, g_i) = \text{proj}(\mathcal{M}_c(s, g_i)) + \lambda \text{sim}_1(s, g_i) \quad (1)$$

where $\text{proj}(\cdot)$ takes an d-dimension vector and turns it to a scalar with weight matrix $W \in \mathcal{R}^{d\times 1}$, λ is the weight for the first-stage similarity score. Both W and λ are optimized through backpropagation.

With labeled data, we finetune $\mathcal{M}_c$ to minimize the negative log-likelihood of the correct goal among the top-k candidate goal list, where the log-likelihood is calculated as:

$$ll(s, g_i) = -\log\left(\text{softmax}\left(\frac{\text{sim}_2(s, g_i)}{\sum_{g_j \in \texttt{C}(s)} \text{sim}_2(s, g_j)}\right)\right) \quad (2)$$

Compared to the randomly sampled in-batch negative examples, the top-k candidate goals are presumably harder negative examples (Karpukhin et al., 2020) and thus the model must work harder to distinguish between them. We will explain the extraction of the labeled step-goal pairs used to train this model in §4.1.

Concretely, we experiment with two pretrained models as $\mathcal{M}_c$, specifically BERT-base (Devlin et al., 2018) and DeBERTa-large finetuned on the MNLI dataset (He et al., 2020). We pick the first because it is standard, and the second due to its high position on the performance chart of BERTScore (Zhang et al., 2019). [5]In addition, we consider including different *ctx* in the reranking input. For each step, we experiment with including no context, the goal of the step, and the surrounding steps of the step within a window-size n (n=1).

[4]www.bing.com

[5]https://cutt.ly/oTx5gMM. Both BERTScore and our reranker aim at measuring the semantic similarity between a pair of texts, motivating this choice.

3.3 Unlinkable Steps

Some steps in wikiHow could not be matched with any goal. Such steps are *unlinkable* because of several reasons. First, the step itself might be so fine-grained that further instructions are unnecessary (e.g. *"Pick up clothes"*). Second, although wikiHow spans a wide range of complex procedures, it is far from comprehensive. Some goals simply do not exist in wikiHow. Hence, we design a mechanism to predict whether a step is linkable or not explicitly. More specifically, we add a special token `unlinkable`, taken from the reserved vocabulary of a pretrained model, as a placeholder "goal" to the top-k candidate goal list $\texttt{C}(s)$, and this placeholder is treated as the gold-standard answer if the step is determined to be unlinkable. The similarity score between a step and this placeholder goal follows Equation 1 and $\text{sim}_1(s, \texttt{unlinkable})$ is set to the lowest first-stage similarity score among the candidate goals retrieved by the first-stage model. Accurately labeling a step as `unlinkable` is non-trivial – it requires examining whether the step can be linked to any goal in G. Instead, we train the model to perform this determination by assigning `unlinkable` to steps that have a ground-truth goal but this goal does not appear in the top-k candidate goal list. The loss follows Equation 2.

4 Automatic Step Prediction Evaluation

To evaluate how well our hierarchy discovery model can link steps to goals, we leverage existing annotated step-goal links.

4.1 Labeled Step-goal Construction

In wikiHow, there are around $21k$ steps that already have a hyperlink redirecting it to another wikiHow article, populated by editors. We treat the title of the linked article as the ground-truth goal for the step. For example, as in B5 of Figure 1, the ground-truth goal of the step "Dust all the surface in your room" is "Dust your entire house". We build the training, development and test set with a 7:2:1 ratio.

4.2 Results

Table 1 lists the recall of different models without or with the reranking. Precision is immaterial here since each step has *only one* linked article.

Candidate Retrieval The SP model achieves the best recall of all models, outperforming SBERT by a significant margin. Search-engine-based models with various configurations, including the commercial Bing Search are less effective. In addition, BM25 (goal only), which does not consider any article content, notably outperforms BM25 (article) and Bing Search, implying that the full articles may contain undesirable noise that hurts the search performance. This interesting observation suggests that while commercial search engines are powerful, they may not be the best option for specific document retrieval tasks such as ours.

Model	R@1	R@10	R@30
SP	35.8	64.4	72.5
SBERT	30.6	53.3	63.4
BM25 (goal only)	30.5	51.6	61.1
BM25 (article)	9.3	35.3	49.2
Bing Search	28.0	47.9	-
BERT	50.7	69.4	-
DeBERTa	**55.4**	**71.9**	-
− surr	54.3	71.6	-
− goal	55.0	71.5	-
− both	52.4	71.0	-
+ unlinkable	50.4	71.6	-
+ $\lambda = 0$	51.9	71.4	-

Table 1: The recall@n for different models on the test set. The top half are with paraphrase retrieval only and the bottom half are with taking the top-30 candidate goals generated by the best model (SP) and adding the reranking model. The best performance recall is **bold**. "surr" denotes the surrounding steps of the query step.[6]

Reranking We select the top-30 candidate goals predicted by the SP model as the input to the reranking stage. The recall@30 of the SP model is 72.5%, which bounds the performance of any reranker.[7] As seen in the bottom half of Table 1, reranking is highly effective, as the best configuration brings 19.6% improvement on recall@1 and the recall@10 almost reaches the upper bound of this stage. We find that under the same configuration, DeBERTa-large finetuned on MNLI (He et al., 2020) outperforms BERT by 1.7% on recall@1 (52.4-50.7), matching the reported results from BERTScore.[5]

To qualitatively understand the benefit of the reranker, we further inspect random predictions of SP and DeBERTa. We find that the reranker largely resolves *partial matching* problems observed in SP. As shown in **C1** of Table 2, SP tends only to consider the action (*e.g.* learn) or the object (*e.g.* bike) and mistakenly ranks those partially

[6] We are unable to get the top-30 results from Bing search because the web queries only return top-10 search results.

[7] We only experiment with SP because it is the best retrieval model, providing larger improvement headroom.

	Step	**Retrieval Prediction**	**Reranking Prediction** (GT)	**Context**
C1	Learn to chop properly	Learn Editing	Chop Food Like a Pro	Use a Knife
	Acquire a bike	Get on a Bike	Buy a Bicycle	Commute By Bicycle
	Get some vinyl records	Cut Vinyl Records	Buy Used LP Records	Buy a Turntable
C2	Open your coordinates	Read UTM Coordinates	Find Your Coordinates in Minecraft	Find the End Portal in Minecraft
	Fill in sparse spots	Remove Set in Stains	Fill in Eyebrows	Shape Eyebrows (*g*) Trim your brows (*surr*) Use a clear gel to set (*surr*)

Table 2: The main failure modes of the candidate retrieval model (SP) that could be recovered by the reranking model. **Step**: the query step; **Retrieval Prediction**: the top-1 prediction of the best retrieval model SP; **Reranking Prediction**: the top-1 prediction of the best reranking model DeBERTa, it is also the ground-truth goal. By default, the **Context** refers to the goal of the query step. The last case lists both goal (*g*) and the surrounding steps (*surr*).

matched goals the highest. In contrast, the reranker makes fewer mistakes. In addition, we observed that the reranker performed better on rare words or expressions. For example, as shown in the last column of **C1**, the reranker could finds "vinyl records" is closely related to "LP records" and predicts the correct goal while SP could not.

Second, we observe that the surrounding context of a query step and the goal of the step is helpful in general. Incorporating both contexts brings a 3% improvement in recall@1. While steps are informative, they could be highly dependent on the contexts. For example, some steps are underspecified, using pronouns to refer to previously occurring contents or simply omit them. The additional information introduced by the context helps resolve these uncertainties. In the first example of **C2**, the context "minecraft" is absent in the query step but present in the goal of that step. Similarly, in the second example, the context "eyebrows" is absent in the query step but present in both the goal and the surrounding steps.

Finally, adding unlinkable prediction harms the recall@1 due to its over-prediction of `unlinkable` for steps whose ground-truth goal exists in the top-k candidate list. This is an expected trade-off between the recall of linkable steps and the recall of unlinkable steps. To investigate the effectiveness of having this explicit unlinkable prediction, we conduct an extra experiment where we take the predictions of the best DEBERTA model and assign a step as `unlinkable` if the similarity score of its highest-ranked goal is under a threshold t. We find that by setting $t = 0.9$, the recall@1 degrades from 55.4% to 41.9%. The recall of the unlinkable steps whose ground-truth goal is not in the candidate goal list is 45.0%. On the other hand, the explicit unlinkable prediction results in 50.4% recall@1 and 64.5% unlinkable recall. Therefore, this explicit prediction yields more balance between the trade-offs. In §5, we will demonstrate that this explicit unlinkable prediction is overall informative to distinguish steps of the two types through crowdsourcing annotations. We empirically find that setting the weight of $\text{sim}_1(s, g)$ (λ) to 0 is beneficial in the unlinkable prediction setting.

5 Manual Step Prediction Evaluation

Although the automatic evaluation strongly indicates the effectiveness of our proposed hierarchy discovery model, it is not comprehensive. We complement our evaluation with crowdsourced human judgements via Amazon Mechanical Turk (AMT).

Each example to annotate is a tuple of a step, its original goal from wikiHow, and the top-ranked goal predicted by one of our models. For each example, we ask three AMT crowd workers to judge whether the steps in the article of the linked goal are exact, helpful, related, or unhelpful with regard to accomplishing the queried step. Details about the task design, task requirements, worker pay, example sampling, etc. are in Appendix A.

We select SP, DEBERTA, and DEBERTA with unlinkable prediction and $\lambda = 0$ (DEBERTA-UL) for comparison. We attempt to answer the following questions: (1) does the performance trend shown in automatic evaluation hold in human evaluation? and (2) is the unlinkable prediction helpful to eliminate predictions that are likely to fail (an important ability to avoid providing users with misleading information (Rajpurkar et al., 2018))?

For the purpose of the second question, we separate the examples into two groups. One contains the examples predicted as `unlinkable` by the DEBERTA-UL model, for which we consider the

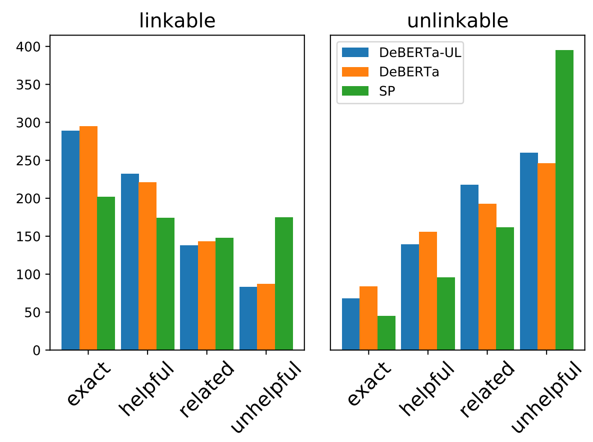

Figure 2: Crowd workers' ratings of step-goal links predicted by our models. The left graph shows steps linked to *some* goals by the DEBERTA-UL model, while the right shows steps those predicted as `unlinkable`.

Query	R/P@1	R/P@10	R/P@25	R/P@50	MR
L_0	2.2/89.2	19.2/78.1	39.9/66.0	56.6/48.2	79.49
L_1	2.2/88.0	19.2/78.0	40.1/66.4	58.1/49.6	75.79
FIL-L_1	2.2/**89.9**	20.2/81.7	43.1/71.2	63.2/53.8	66.32
FIL-L_2	2.2/89.4	**20.3/82.7**	**43.9/72.3**	**65.0/55.2**	**63.38**
L_0	12.1/81.7	59.8/42.8	71.9/20.8	77.9/11.3	41.60
L_1	11.8/79.7	61.2/43.9	74.1/21.4	80.5/11.6	36.70
FIL-L_1	12.4/83.7	66.0/47.3	77.4/22.4	82.9/**12.0**	33.35
FIL-L_2	**12.5/84.4**	**66.1/47.7**	**78.0/22.5**	**83.3/12.0**	**32.30**
L_0	11.4/82.6	59.2/45.2	71.8/22.1	77.8/12.0	43.11
L_1	11.2/81.3	60.4/46.2	73.8/22.7	79.9/12.3	38.19
FIL-L_1	**11.7/85.1**	64.8/49.5	77.2/23.8	82.2/**12.7**	34.76
FIL-L_2	11.6/84.5	**65.5/50.0**	**77.9/24.0**	**82.7/12.7**	**34.13**

Table 3: The Recall/Precision@N (%, $\uparrow$) and mean rank (MR, $\downarrow$) with different queries on the relevant video retrieval task on the training (top), development (middle) and the test set (bottom). The best performance on each set is **bold**.

second highest ranked prediction of DEBERTA-UL and the top-1 prediction of the other models. The other contains the rest, for which we consider the top-1 prediction of all models. The corresponding crowd judgement is shown in Figure 2. Comparing among the models, the DEBERTA model and the DEBERTA-UL model have similar performance, while greatly outperforming the SP model. This shows that our proposed model decomposes much more helpful finer-grained steps to assist users with tasks, similar to the trend observed in our automatic evaluation. Comparing the two graphs, it is apparent that when the DEBERTA-UL model predicts `unlinkable` for a step, the following ranked predictions and those of the other models are more likely to be unhelpful. This implies the high precision of the `unlinkable` prediction, effectively avoiding misleading predictions. Note that our study does not explicitly require subjects to carry out the task, but only annotate whether they find the instructions helpful.

6 Application to Video Retrieval

Previously, we measure the quality of discovered hierarchies through intrinsic evaluation. In this section, we take a further step to study the usefulness of applying our open-domain hierarchical knowledge base to downstream tasks. We select video retrieval as the extrinsic evaluation task, where we aim at retrieving relevant how-to videos for a stated goals. Intuitively, this is a suitable extrinsic evaluation task because videos usually contain finer-grained steps and instructions to accomplish a task, the extra information presented in decomposed steps of a goal could be helpful for retrieving its relevant videos.

6.1 Dataset Construction

From the Howto100M dataset (Miech et al., 2019), a large-scale dataset consisting of millions of instructional videos for over $23k$ goals, we construct our video retrieval corpus by randomly sampling $1,000$ goals (*e.g. "host a party"*) with their relevant videos. The relevant videos $\boldsymbol{v}_g = \{v_1, v_2, ..., v_n\}$ of each goal g in the dataset is obtained by selecting the top 150 videos among the search results of the goal on YouTube.[8] For each goal g, we randomly split its relevant videos $\boldsymbol{v}_g$ into three sub-sets $\boldsymbol{v}_g^{\text{tr}}$, $\boldsymbol{v}_g^{\text{dev}}$ and $\boldsymbol{v}_g^{\text{test}}$ with a ratio of 75%:12.5%:12.5%, which are included in the training, development, and testing sets, respectively.[9]

6.2 Setup

We use BM25 as the retrieval model, where videos are represented as their (automatically generated) captions and indexed using Elasticsearch.[10] We consider two baseline query strategies, together with a straightforward but effective video-oriented filtering method that generates more informative queries with hierarchical knowledge.

L_0: Goal only The query is the goal g itself. This is the minimal query without any additional hierarchical information.

[8]Although the relevance between a goal and a video is not explicitly annotated in the Howto100M dataset, we argue that with the sophisticated engineering of the YouTube video search API and hundreds of thousands user clicks, the high ranked videos are likely to be demonstrating the queried goal.

[9]We explain more about the appropriateness of the downstream video retrieval task setup in B.1.

[10]We found the performance of a neural model (BERT finetuned on query/video caption pairs) significantly lower than BM25 and therefore we only experiment with BM25.

L_1: Goal + $\texttt{Ch}(g)$ The query is a combination of the goal g and its immediate steps $\texttt{Ch}(g)$. This query encodes hierarchical knowledge that already exists in wikiHow. The final score between a combined query and a document is the weighted sum of scores between each individual query and the document. The weight is tuned on a development set and it is set to 1 for g and 0.1 for $\texttt{Ch}(g)$.

FIL-L_1, FIL-L_1: Filtered Steps While L_1 generates contextually richer queries by incorporating all the information in the child steps of a goal, this could also likely introduce noise, since the child steps have varying frequencies in the video demonstrations, some steps might not appear in any video at all. This could become more problematic as more levels of decomposed steps are included in the query. We therefore consider filtering the steps of a goal and only retain those that are most informative. Specifically, for each goal g, we use a hill-climbing algorithm to check each step s, and include s into the query if it yields better ranking results for videos in the training set $\boldsymbol{v}_g^{\text{train}}$. See algorithm 1 in Appendix for more details. We experiment with two scenarios: FIL-L_1, which filters steps in $\texttt{Ch}(g)$, and FIL-L_2, which filters steps in both $\texttt{Ch}(g)$ and their children $\texttt{Ch}(n_i)$ ($n_i \in \texttt{Ch}(g)$). FIL-L_2 encodes deeper hierarchical knowledge than FIL-L_1. Similarly, we tune the weight and it is set to 1 for g and 0.5 for the filter steps.

6.3 Results

We report the precision@N, recall@N and mean rank (MR) following existing works on video retrieval (Luo et al., 2021) (Equations listed in Appendix). Table 3 lists the results. First, queries that encode hierarchies of goals (L_1, FIL-L_1 and FIL-L_2) are generally more beneficial than queries that do not (L_0). The steps of goals enrich a query and assist the retrieval. Second, video-oriented filtering produces a set of more generalizable steps that are shared among multiple videos and yields significant improvement over the un-filtered L_1 queries. This is somewhat expected. Although steps of a goal in wikiHow articles are contributed by human writers, they are not grounded to real-world executions of that goal. Many steps do not have corresponding executions in the videos and become noisy steps in the L_1 queries. More interestingly, we observe that queries using deeper hierarchies (FIL-L_2) outperform the shallower ones (FIL-L_1) in most cases. This is probably due to that how-to videos usually contain detailed (verbal) instructions of a procedure, which are better aligned with more fine-grained steps found in FIL-L_2.

Goal	Stain Cabinet
FIL-L_1	Purchase some stain colors to test
FIL-L_2	FIL-L_1 + Buy cloth with which to apply the stain Unscrew the cabinet from the wall Clean your workspace
KM	Remove the doors Sanding the front Top coat Finished look

Goal	Make Avocado Fries
FIL-L_1	Bake the avocado fries until they are golden Dip the avocado wedges into the egg and then the breadcrumbs
FIL-L_2	FIL-L_1 + Preheat the oven Peel and pit the avocados Cut your avocado in half and remove the stone Let rise Finished, cool and enjoy
KM	2 large avocados ... pinch of salt, pinch of pepper two eggs, beaten ... bake at 425F 20 min until golden bros ...

Table 4: The queries and the key moments (KM) for two goals. "..." represents the omission of steps that describe the ingredients to save space. The first selected video is `h9k0T25_NxA` and the second is `o7uVUmPph6I`.

In our qualitative study, we investigate how FIL-L_2 queries with deeper hierarchies help retrieval. Table 4 list FIL-L_1 and FIL-L_2 queries for two goals. We find that the FIL-L_2 queries are more informative and cover more aspects. For example, the FIL-L_2 queries for *"stain cabinet"* and *"make avocado fries"* consist of the preparation, actual operations, and the post-processing steps while the FIL-L_1 query only contains the first one. In addition, we list the key moments that textually describe the important clips of the videos from some randomly selected YouTube videos by searching the goal on Google with the assumption that steps grounded in videos can also serve as the query for the goal.[11] We find that the FIL-L_2 query of *"make avocado fries"* explains a few necessary steps to accomplish this goal, while the key moment is mostly composed of the ingredients of this dish. This comparison suggests the potential integration of our induced hierarchical knowledge to identify key moments in videos in the future.

[11]Key moments are either identified manually, or are extracted automatically by YouTube. https://cutt.ly/qTcxSi6

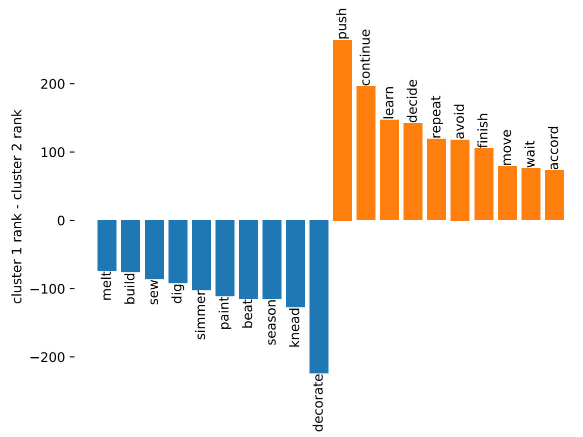

Figure 3: The verbs with largest rank difference in two clusters. The blue bars are words becoming less frequent in cluster 2 (decomposed steps) and the orange bars are words becoming more frequent.

6.4 Decomposition Analysis

In this part, we study the properties of the hierarchies: (1) what kind of steps are likely to be linked to another goal and are thus decomposed? and (2) what do the decomposed steps look like?

We group steps to two clusters. The first contains the immediate steps of a goal ($s \in \mathtt{Ch}(g)$) whose prediction is not `unlinkable`. The second contains the decomposed steps of the steps in the first cluster ($s' \in \mathtt{Ch}(s)$). We use spaCy (Honnibal et al., 2020) to extract and lemmatize the verb in each step and rank the verbs by their frequency in each cluster. Next, the top-100 most frequent verbs in each cluster are selected and we measure the rank difference of these verbs in the two clusters. Figure 3 plots the verbs with largest rank difference and the full figure is in Figure 4. We observe that verbs that convey complex actions and intuitively consist of many other actions become less frequent after the decomposition (*e.g.* decorate). On the other hand, verbs that describe the action itself gain in frequency after the decomposition (*e.g.* push, hold, press). This observation follows with our assumption that the decomposition would leads to more fine-grained realizations of a complex procedure. Some other more abstract actions such as "learn" and "decide" also increase in frequency, as they are part of planning processes that go into more complex actions.

7 Related Work

Linking Procedural Events Lagos et al. (2017) share the same task formulation as ours. Both works try to link steps (or spans within steps) to other wikiHow articles. We make several important contributions over their work: (1) a search-then-rerank method significantly increases linking recall; (2) more comprehensive experiments with manual and downstream evaluation that showcases the quality and usefulness of the linked data and (3) experiments and data with broader coverage over all of WikiHow, not just the Computer domain.

Procedural Knowledge Procedural knowledge can be seen as a subset of knowledge pertaining to *scripts* (Abelson and Schank, 1977; Rudinger et al., 2015), *schemata* (Rumelhart, 1975) or events. A small body of previous work (Mujtaba and Mahapatra, 2019) on procedural events includes extracting them from instructional texts (Paris et al., 2002; Delpech and Saint-Dizier, 2008; Zhang et al., 2012), reasoning about them (Takechi et al., 2003; Tandon et al., 2019; Rajagopal et al., 2020), or showing their downstream applications (Pareti, 2018; Zhang et al., 2020d; Yang et al., 2021; Zhang et al., 2020b; Lyu et al., 2021), specifically on intent reasoning (Sap et al., 2019; Dalvi et al., 2019; Zhang et al., 2020c). Most procedural datasets are collected by crowdsourcing then manually cleaned (Singh et al., 2002; Regneri et al., 2010; Li et al., 2012; Wanzare et al., 2016; Rashkin et al., 2018) and are hence small. Existing works also leverage wikiHow for large-scale knowledge-base construction (Jung et al., 2010; Chu et al., 2017; Park and Motahari Nezhad, 2018), but our work is the first to provide comprehensive intrinsic and extrinsic evaluation of the resulting knowledge-base.

8 Conclusion

We propose a search-then-rerank algorithm to effectively construct a hierarchical knowledge-base of procedures based on wikiHow. Our hierarchies are shown to help users accomplish tasks by accurately providing decomposition of a step and improve performance of downstream tasks such as retrieving instructional videos. One interesting extension is to further study and improve the robustness of our two-stage method to tackle more complex linguistic structures of steps and goals (*e.g.* negation, conjunction). Another direction is to enrich the resulting knowledge-base by applying our method to other web resources.[12] Future works could also explore other usages such as comparing and clustering procedures based on their deep hierarchies.

[12] *e.g.* https://www.instructables.com/, https://www.diynetwork.com/how-to

Acknowledgments

This research is based upon work supported in part by the DARPA KAIROS Program (contract FA8750-19-2-1004), the DARPA LwLL Program (contract FA8750-19-2-0201), the IARPA BETTER Program (contract 2019-19051600004), and the Amazon Alexa Prize TaskBot Competition. Approved for Public Release, Distribution Unlimited. The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright notation thereon. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies, either expressed or implied, of Amazon, DARPA, IARPA, or the U.S. Government.

We thank Ziyang Li and Ricardo Gonzalez for developing the web demo. We also thank **[TODO: people]** for their valuable feedback.

References

Robert Abelson and Roger C Schank. 1977. Scripts, plans, goals and understanding. *An inquiry into human knowledge structures New Jersey*, 10.

Fabian Caba Heilbron, Victor Escorcia, Bernard Ghanem, and Juan Carlos Niebles. 2015. Activitynet: A large-scale video benchmark for human activity understanding. In *Proceedings of the ieee conference on computer vision and pattern recognition*, pages 961–970.

Tommaso Caselli and Piek Vossen. 2017. The event storyline corpus: A new benchmark for causal and temporal relation extraction. In *Proceedings of the Events and Stories in the News Workshop*, pages 77–86.

Cuong Xuan Chu, Niket Tandon, and Gerhard Weikum. 2017. Distilling task knowledge from how-to communities. In *Proceedings of the 26th International Conference on World Wide Web*, pages 805–814.

Bhavana Dalvi, Niket Tandon, Antoine Bosselut, Wen-tau Yih, and Peter Clark. 2019. Everything happens for a reason: Discovering the purpose of actions in procedural text. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 4496–4505, Hong Kong, China. Association for Computational Linguistics.

Estelle Delpech and Patrick Saint-Dizier. 2008. Investigating the structure of procedural texts for answering how-to questions. In *Proceedings of the Sixth International Conference on Language Resources and Evaluation (LREC'08)*, Marrakech, Morocco. European Language Resources Association (ELRA).

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

Goran Glavaš, Jan Šnajder, Parisa Kordjamshidi, and Marie-Francine Moens. 2014. Hieve: A corpus for extracting event hierarchies from news stories. In *Proceedings of 9th language resources and evaluation conference*, pages 3678–3683. ELRA.

Chikara Hashimoto, Kentaro Torisawa, Julien Kloetzer, Motoki Sano, István Varga, Jong-Hoon Oh, and Yutaka Kidawara. 2014. Toward future scenario generation: Extracting event causality exploiting semantic relation, context, and association features. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 987–997.

Pengcheng He, Xiaodong Liu, Jianfeng Gao, and Weizhu Chen. 2020. Deberta: Decoding-enhanced bert with disentangled attention. *arXiv preprint arXiv:2006.03654*.

Matthew Honnibal, Ines Montani, Sofie Van Landeghem, and Adriane Boyd. 2020. spaCy: Industrial-strength Natural Language Processing in Python.

Samuel Humeau, Kurt Shuster, Marie-Anne Lachaux, and Jason Weston. 2019. Poly-encoders: Transformer architectures and pre-training strategies for fast and accurate multi-sentence scoring. *arXiv preprint arXiv:1905.01969*.

Jeff Johnson, Matthijs Douze, and Hervé Jégou. 2017. Billion-scale similarity search with gpus. *arXiv preprint arXiv:1702.08734*.

Yuchul Jung, Jihee Ryu, Kyung-min Kim, and Sung-Hyon Myaeng. 2010. Automatic construction of a large-scale situation ontology by mining how-to instructions from the web. *Web Semantics: Science, Services and Agents on the World Wide Web*, 8(2-3):110–124.

Vladimir Karpukhin, Barlas Oğuz, Sewon Min, Patrick Lewis, Ledell Wu, Sergey Edunov, Danqi Chen, and Wen-tau Yih. 2020. Dense passage retrieval for open-domain question answering. *arXiv preprint arXiv:2004.04906*.

Taku Kudo and John Richardson. 2018. SentencePiece: A simple and language independent subword tokenizer and detokenizer for neural text processing. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 66–71, Brussels, Belgium. Association for Computational Linguistics.

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.

Boyang Li, Stephen Lee-Urban, Darren Scott Appling, and Mark O Riedl. 2012. Crowdsourcing narrative intelligence. *Advances in Cognitive systems*, 2(1).

Zhengzhong Liu, Jun Araki, Eduard H Hovy, and Teruko Mitamura. 2014. Supervised within-document event coreference using information propagation. In *LREC*, pages 4539–4544.

Huaishao Luo, Lei Ji, Ming Zhong, Yang Chen, Wen Lei, Nan Duan, and Tianrui Li. 2021. CLIP4Clip: An empirical study of clip for end to end video clip retrieval. *arXiv preprint arXiv:2104.08860*.

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

Antoine Miech, Dimitri Zhukov, Jean-Baptiste Alayrac, Makarand Tapaswi, Ivan Laptev, and Josef Sivic. 2019. Howto100m: Learning a text-video embedding by watching hundred million narrated video clips. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 2630–2640.

Dena Mujtaba and Nihar Mahapatra. 2019. Recent trends in natural language understanding for procedural knowledge. In *2019 International Conference on Computational Science and Computational Intelligence (CSCI)*, pages 420–424.

Qiang Ning, Hao Wu, and Dan Roth. 2018. A multi-axis annotation scheme for event temporal relations. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1318–1328, Melbourne, Australia. Association for Computational Linguistics.

Paolo Pareti. 2018. *Representation and execution of human know-how on the Web*. Ph.D. thesis.

Cécile Paris, Keith Vander Linden, and Shijian Lu. 2002. Automated knowledge acquisition for instructional text generation. In *Proceedings of the 20th Annual International Conference on Computer Documentation*, SIGDOC '02, page 142–151, New York, NY, USA. Association for Computing Machinery.

Hogun Park and Hamid Reza Motahari Nezhad. 2018. Learning procedures from text: Codifying how-to procedures in deep neural networks. In *Companion Proceedings of the The Web Conference 2018*, pages 351–358.

James Pustejovsky, Patrick Hanks, Roser Sauri, Andrew See, Robert Gaizauskas, Andrea Setzer, Dragomir Radev, Beth Sundheim, David Day, Lisa Ferro, et al. 2003. The timebank corpus. In *Corpus linguistics*, volume 2003, page 40. Lancaster, UK.

Dheeraj Rajagopal, Niket Tandon, Peter Clark, Bhavana Dalvi, and Eduard Hovy. 2020. What-if I ask you to explain: Explaining the effects of perturbations in procedural text. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 3345–3355, Online. Association for Computational Linguistics.

Pranav Rajpurkar, Robin Jia, and Percy Liang. 2018. Know what you don't know: Unanswerable questions for SQuAD. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 784–789, Melbourne, Australia. Association for Computational Linguistics.

Hannah Rashkin, Maarten Sap, Emily Allaway, Noah A. Smith, and Yejin Choi. 2018. Event2Mind: Commonsense inference on events, intents, and reactions. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 463–473, Melbourne, Australia. Association for Computational Linguistics.

Michaela Regneri, Alexander Koller, and Manfred Pinkal. 2010. Learning script knowledge with web experiments. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 979–988.

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.

Stephen Robertson and Hugo Zaragoza. 2009. The probabilistic relevance framework: Bm25 and beyond. *Found. Trends Inf. Retr.*, 3(4):333–389.

Rachel Rudinger, Vera Demberg, Ashutosh Modi, Benjamin Van Durme, and Manfred Pinkal. 2015. Learning to predict script events from domain-specific text. In *Proceedings of the Fourth Joint Conference on Lexical and Computational Semantics*, pages 205–210.

David E Rumelhart. 1975. Notes on a schema for stories. In *Representation and understanding*, pages 211–236. Elsevier.

Maarten Sap, Ronan Le Bras, Emily Allaway, Chandra Bhagavatula, Nicholas Lourie, Hannah Rashkin, Brendan Roof, Noah A. Smith, and Yejin Choi. 2019. ATOMIC: an atlas of machine commonsense for if-then reasoning. In *The Thirty-Third AAAI Conference on Artificial Intelligence, AAAI 2019, The Thirty-First Innovative Applications of Artificial Intelligence Conference, IAAI 2019, The Ninth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2019, Honolulu, Hawaii, USA, January 27 - February 1, 2019*, pages 3027–3035. AAAI Press.

Push Singh, Thomas Lin, Erik T Mueller, Grace Lim, Travell Perkins, and Wan Li Zhu. 2002. Open mind common sense: Knowledge acquisition from the general public. In *OTM Confederated International Conferences" On the Move to Meaningful Internet Systems"*, pages 1223–1237. Springer.

Mineki Takechi, Takenobu Tokunaga, Yuji Matsumoto, and Hozumi Tanaka. 2003. Feature selection in categorizing procedural expressions. In *Proceedings of the Sixth International Workshop on Information Retrieval with Asian Languages*, pages 49–56, Sapporo, Japan. Association for Computational Linguistics.

Niket Tandon, Bhavana Dalvi, Keisuke Sakaguchi, Peter Clark, and Antoine Bosselut. 2019. WIQA: A dataset for "what if..." reasoning over procedural text. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 6076–6085, Hong Kong, China. Association for Computational Linguistics.

Yansong Tang, Dajun Ding, Yongming Rao, Yu Zheng, Danyang Zhang, Lili Zhao, Jiwen Lu, and Jie Zhou. 2019. Coin: A large-scale dataset for comprehensive instructional video analysis. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1207–1216.

Ji Wan, Dayong Wang, Steven Chu Hong Hoi, Pengcheng Wu, Jianke Zhu, Yongdong Zhang, and Jintao Li. 2014. Deep learning for content-based image retrieval: A comprehensive study. In *Proceedings of the 22nd ACM international conference on Multimedia*, pages 157–166.

Lilian DA Wanzare, Alessandra Zarcone, Stefan Thater, and Manfred Pinkal. 2016. A crowdsourced database of event sequence descriptions for the acquisition of high-quality script knowledge. In *Proceedings of the tenth international conference on language resources and evaluation (LREC'16)*, pages 3494–3501.

John Wieting, Kevin Gimpel, Graham Neubig, and Taylor Berg-Kirkpatrick. 2019. Simple and effective paraphrastic similarity from parallel translations. In *Proceedings of the Association for Computational Linguistics*.

John Wieting, Kevin Gimpel, Graham Neubig, and Taylor Berg-Kirkpatrick. 2021. Paraphrastic representations at scale. *arXiv preprint arXiv:2104.15114*.

Thomas Wolf, Julien Chaumond, Lysandre Debut, Victor Sanh, Clement Delangue, Anthony Moi, Pierric Cistac, Morgan Funtowicz, Joe Davison, Sam Shleifer, et al. 2020. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45.

Ledell Wu, Fabio Petroni, Martin Josifoski, Sebastian Riedel, and Luke Zettlemoyer. 2019. Zero-shot entity linking with dense entity retrieval. corr abs/1911.03814 (2019). *arXiv preprint arxiv:1911.03814*.

Yue Yang, Artemis Panagopoulou, Qing Lyu, Li Zhang, Mark Yatskar, and Chris Callison-Burch. 2021. Visual goal-step inference using wikihow. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, Punta Cana, Dominican Republic. Association for Computational Linguistics.

Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. 2019. HellaSwag: Can a machine really finish your sentence? In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4791–4800, Florence, Italy. Association for Computational Linguistics.

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*.

Hongming Zhang, Muhao Chen, Haoyu Wang, Yangqiu Song, and Dan Roth. 2020b. Analogous process structure induction for sub-event sequence prediction. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1541–1550, Online. Association for Computational Linguistics.

Li Zhang, Qing Lyu, and Chris Callison-Burch. 2020c. Intent detection with WikiHow. In *Proceedings of the 1st Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics and the 10th International Joint Conference on Natural Language Processing*, pages 328–333, Suzhou, China. Association for Computational Linguistics.

Li Zhang, Qing Lyu, and Chris Callison-Burch. 2020d. 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.

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

Yi Zhang, Sujay Kumar Jauhar, Julia Kiseleva, Ryen White, and Dan Roth. 2021. Learning to decompose and organize complex tasks. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 2726–2735.

Ziqi Zhang, Philip Webster, Victoria Uren, Andrea Varga, and Fabio Ciravegna. 2012. Automatically extracting procedural knowledge from instructional texts using natural language processing. In *Proceedings of the Eighth International Conference on Language Resources and Evaluation (LREC'12)*, pages 520–527, Istanbul, Turkey. European Language Resources Association (ELRA).

Yilun Zhou, Julie Shah, and Steven Schockaert. 2019. Learning household task knowledge from WikiHow descriptions. In *Proceedings of the 5th Workshop on Semantic Deep Learning (SemDeep-5)*, pages 50–56, Macau, China. Association for Computational Linguistics.

A Crowdsourcing Details

As discussed in section 5, we use Amazon Mechanical Turk (mTurk) to collect human judgements of linked wikiHow articles. Our mTurk task design HTML is attached in the supplementary materials. Each task includes an overview, examples of ratings, and 11 questions including 1 control question. Each question has the following prompt:

> Imagine you're reading an article about the goal `c_goal`, which includes a step `step`. Then, you're presented with a new article `r_goal`. Does this new article help explain how to do the step `step`?

where `c_goal` is the original corresponding goal of the `step`, and `r_goal` is the retrieved goal by the model. Both `c_goal` and `r_goal` have hyperlinks to the wikiHow article. The options of rating are:

1. The article explains exactly how to do the step.
2. The article is helpful, but it either doesn't have enough information or has too much unrelated information.
3. The article explains something related, but I don't think I can do the step with the instructions.
4. The article is unhelpful/unrelated.
5. I don't know which option to choose, because: [text entry box]

The control question contains either a `step` and `r_goal` with the exact same texts once lowercased (in which case the expected answer is always #1), or a `step` and a randomly selected unrelated `r_goal` (in which case the expected answer is always #4). We estimate that answering each question would take 30 seconds, with a pay of \$0.83 per task which equates to an hourly rate of \$9.05. We require workers to be English-speaking, with the mTurk Master qualification and a lifetime approval rate of over 90%.

To sample examples to annotate, we first obtain all the steps corresponding to the same 1000 goals as we did in subsection 6.1. To evaluate the DeBERTa-UL's ability to predict `unlinkable`, we randomly sample 500 steps predicted as `unlinkable` and another 500 predicted as otherwise. Then, for these 1000 steps, we obtain linked goal predictions of our three models:

Algorithm 1: Video-based filtering

```
Data: goal $g$, cost function $f$, candidate steps
      $\boldsymbol{p} = [p_1, ..., p_n]$, relevant videos $\boldsymbol{v}_g^{\text{tr}}$
Result: $best\_query$
$k \leftarrow 15$;
$best\_query \leftarrow [g]$;
$min\_cost \leftarrow f(best\_query, \boldsymbol{v}_g^{\text{tr}})$;
$r \leftarrow \min(n, k)$;
while $r \geq 0$ do
    $in\_cost \leftarrow 1e10$;
    for $p$ in $\boldsymbol{p}$ do
        if $p$ not in $best\_state$ then
            $query \leftarrow [best\_query, p]$;
            $cost \leftarrow f(query, \boldsymbol{v}_g^{\text{tr}})$;
            if $cost < in\_cost$ then
                $in\_cost \leftarrow cost$;
                $in\_query \leftarrow query$;
            end
        end
    end
    if $in\_cost < min\_cost$ then
        $min\_cost \leftarrow in\_cost$;
        $best\_query \leftarrow in\_query$;
    else
        break
    $r = r - 1$;
end
```

DeBERTa-UL, DeBERTa, and the SP model. If DeBERTa-UL predicts a step to be `unlinkable` by ranking the placeholder token first, the second ranked goal is instead considered. After removing duplicates of predicted step-goal pairs, we are left with 1448 examples.

When performing analyses, we only consider the responses from crowdworkers that pass more control questions than they fail.

B Video Retrieval Setup

B.1 Dataset Construction

Existing works also practice similar data splits that share the labels of videos/images across the training, development and the test set. For example, image retrieval tasks use the same objects labels for training and evaluations (Wan et al., 2014); Activity Net (Caba Heilbron et al., 2015), a popular benchmark for human activity understanding, uses the same 203 activities across different splits; Yang et al. (2021) trains a step inference model with a training set that shares the same goals with the test set.

This data split is meaningful on its own. We can view the original queries as initial schemas for complex procedures. Then we induce more *generalizable* schemas by matching them with schema instantiations (in our case, the videos that display

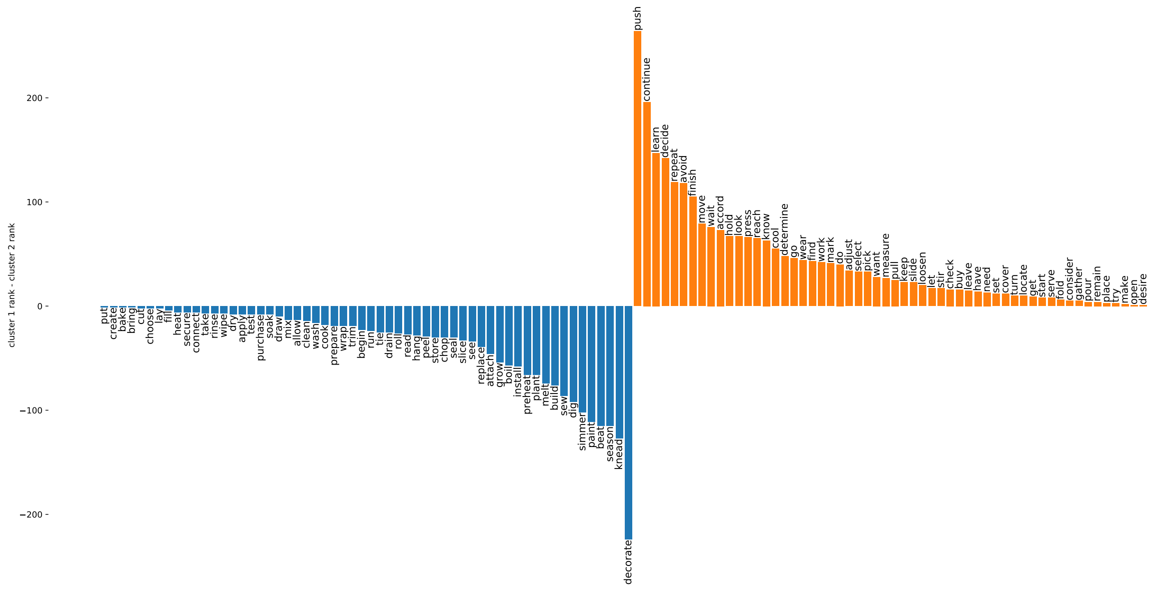

Figure 4: The full version of Figure 3

the procedures). We evaluate the quality of the induced schemas by matching them with *unseen* instantiations. The large-scale DARPA KAIROS project[13] adopted a similar setup, which we believe indicates its great interest to the community.

In terms of the scale of the video retrieval dataset, though we only select 1000 goals from $23k$ goals from Howto1M, there are already $150k$ videos in total while widely-used video datasets like COIN (Tang et al., 2019) only contain 180 goals and $10k$ videos. In addition, exiting works like (Yang et al., 2021) also experimented with a sampled dataset of similar scale.

B.2 Evaluation Metrics

We report precision@N, recall@N and mean rank (MR) following existing works on video retrieval (Luo et al., 2021)

$$\begin{aligned}
\texttt{recall@N} &= \frac{1}{M}\sum_{i=1}^{M}\frac{\sum_{v_j\in\boldsymbol{v}_{g_i}}\mathbb{1}(r(v_j)<=N)}{|\boldsymbol{v}_{g_i}|}\\
\texttt{precision@N} &= \frac{1}{M}\sum_{i=1}^{M}\frac{\sum_{v_j\in\boldsymbol{v}_{g_i}}\mathbb{1}(r(v_j)<=N)}{N}\\
\texttt{MR} &= \frac{1}{M}\sum_{i=1}^{M}\frac{\sum_{v_j\in\boldsymbol{v}_{g_i}}r(v_j)}{|\boldsymbol{v}_{g_i}|}
\end{aligned} \tag{3}$$

where M is the number of goals in total, $\boldsymbol{v}_{g_i}$ is a set of ground truth videos of goal g_i is the rank of video v and $\mathbb{1}$ is the indicator function.

[13]https://www.darpa.mil/program/knowledge-directed-artificial-intelligence-reasoning-over-schemas

C Experiment Reproducibility

Candidate Goal Retrieval The detailed parameter information of SP can be found in S5.1 in (Wieting et al., 2021). Encoding all steps and goals in wikiHow took around two hours on a 2080Ti (12GB) GPU. For SBERT, the encoding took around an hour on a v100 GPU (32GB).

Reranking We used the `transformers` library (Wolf et al., 2020) for re-ranking. The two re-ranking models we used are "bert-base-uncased" and "deberta-v2-large-mnli". We finetuned each model on our training set for five epochs and selected the best model on the validation set. Finetuning took around two hours on a 2080Ti (12GB) GPU for BERT and eight hours on a v100 GPU (32GB) for DeBERTa. We used the default hyperparameters provided by the `transformers` library.

D Risks

Our resulting hierarchy contains events from wikiHow, which may contain unsafe content that slip through its editorial process, although this is relatively unlikely.

E License of Used Assets

The wikiHow texts used in this work are licensed under CC BY-NC-SA 3.0.
FAISS is licensed under MIT License.
BERT is licensed under Apache License 2.0.
DeBERTa is licensed under MIT License.