Procedures are inherently hierarchical. To make videos, one may need to purchase a camera, which in turn may require one to set a budget. While such hierarchical knowledge is critical for reasoning about complex procedures, most existing work has treated procedures as shallow structures without modeling the parent-child relation. 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 carry out a complex procedure. To this end, we develop a simple and efficient method that links steps (e.g., purchase a camera) in an article to other articles with similar goals (e.g., how to choose a camera), recursively constructing 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 Introduction

A procedure includes some steps needed to achieve a particular goal (Momouchi, 1980). Procedures are inherently hierarchical: a high-level procedure is composed of many lower-level procedures. For example, a procedure with the goal make videos consists of steps like purchase a camera, set up lighting, edit the video, and so on, where each step itself is a procedure as well. Such hierarchical relations between procedures are recursive: the lower-level procedures can be further decomposed into even more fine-grained steps: one may need to arrange the footage in order to edit the video.

Relatively little attention has been paid to hierarchical relations in complex procedures in the field of NLP. Some work performed a shallow one-level decomposition and often required costly resources such as human expert task-specific annotation (Chu et al., 2017; Zhang et al., 2020a, 2021). More attention has been paid in fields adjacent to NLP. For example, Lagos et al. (2017) and Pareti et al. (2014) both create hierarchical structures in how-to documents by linking action phrases in one procedure to another procedure or by linking steps in how-to articles to resources like DBPedia (Auer et al., 2007). This kind of linking is helpful for explaining complex steps to readers who do not have prior knowledge of the topic being explained.

In this paper, we revisit this important but understudied task to develop a simple and effective algorithm (Figure 1) to construct a hierarchical knowledge-base (KB) for over 110k complex procedures spanning a wide range of topics from wikiHow, a large-scale how-to website that has recently become a widely-used resource in NLP (Zhou et al., 2019; Zellers et al., 2019; Zhang et al., 2020d,c). From each wikiHow article which represents a procedure, we follow Zhang et al. (2020d) and extract the title as the goal (e.g., g₁ in Figure 1), and the paragraph headlines as steps (e.g., s₁ ... sn). Next, we decompose the steps by linking them to articles with the same or a similar goal (e.g., s₁ to g₂). The steps of the linked article are treated as the finer-grained steps (s₁ to s_j) of the linked step (s₁). In this way, the procedural hierarchies go from shallow (B1) to deep (B4).

To link steps and article goals, we employ a retrieve-then-rerank approach, a well-established paradigm in related tasks (Wu et al., 2019; Humeau et al., 2019). Our hierarchy discovery model (§3) first independently encodes each step and goal in wikiHow and searches the k nearest goals of similar meaning for each step (B2). Then, it applies a dedicated joint encoder to calculate the similarity score between the step and each candidate goal,
thus reranking the goals (B3). This pipeline can efficiently search over a large candidate pool while accurately measuring the similarity between steps and goals. With each step linked to an article goal, a hierarchical KB of procedures is thus constructed.

We evaluate our KB both intrinsically and extrinsically. Intrinsically, the discovered links can be directly used to complete missing step-goal hyperlinks in wikiHow, which have been manually curated (B5). Our proposed method outperforms strong baselines (e.g., Lagos et al. (2017)) according to both automatic and human evaluation, in terms of recall and usefulness respectively (§4, §5). Extrinsically, we consider the task of retrieving instructional videos given textual queries. We observe that queries that encode deeper hierarchies are better than those that do not (§6). This provides evidence that our KB can bridge the high-level instructions and the low-level executions of procedures, which is important for applications such as robotic planning.

2 Problem Formulation

We represent a procedure as a tree where the root node $n$ represents a goal and its children nodes $Ch(n)$ represent the steps of $n$. We formulate the hierarchy discovery task as identifying the steps among $Ch(n)$ that can themselves be a goal of some other finer-grained steps (sub-steps), which are inserted into the tree.

While this formulation could potentially be used on any large collection of procedures, we specifically focus on wikiHow. As shown in B1 of Figure 1, each article comprises a goal ($g$), and a series of steps ($Ch(g)$). 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 algorithm aims to link a step $s_i \in S$ to a goal $g \in G$ such that $g$ has the same meaning as $s_i$. It then treats $Ch(g)$ as $Ch(s_i)$. Given that $g$ and $s_i$ are both represented by textual descriptions, the discovery process can be framed as a paraphrase detection task. 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 (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 retrieval, encoding each step and goal separately in an unsupervised fashion and select the $k$ most similar goals for each step $s$. This process is fast at the expense of accuracy. Second, we perform reranking, jointly encoding a step with each of its candidate goals in a supervised fashion to allow for more expressive contextualized embeddings. This process is more accurate at the expense of speed, since calculating each similarity score requires a forward pass in the neural network. The goal with the highest similarity score is se-

Figure 1: The overview of our proposed method. The input (B1) and output (B4) of the hierarchy discovery model (B2, B3) and the applications (B5, B6) of the hierarchical knowledge base.
lected and the step is expanded accordingly, as in B4 of Figure 1.

3.1 Retrieval

In the first stage, we independently encode each step $s \in S$ and goal $g \in G$ with a model $M_b$, resulting in embeddings $e_{s_1}, e_{s_2}, \ldots, e_{s_n}$ and $e_{g_1}, e_{g_2}, \ldots, e_{g_m}$. The similarity score between $s$ and $g$ is calculated as the cosine similarity between $e_s$ and $e_g$. 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 $C(s) = [g_1, \ldots, g_k]$. To perform this top-$k$ search, we use efficient similarity search libraries such as FAISS (Johnson et al., 2017).

We instantiate $M_b$ 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 search engines as $M_b$, specifically Elasticsearch with the standard BM25 weighting metric (Robertson and Zaragoza, 2009). We index each article with its title only or with its full article. We also experiment with Bing Search API where we limit the search to wikiHow website only3. 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 encode the goals and vice-versa. Therefore, we concatenate a step with each of its top-$k$ candidate goals in $C(s)$ and feed them to a model $M_c$ 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:

$$[CLS] \text{ctx} [ST] \text{step} [ED] \text{goal} [SEP]$$

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 is 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:

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

where $\text{proj}(\cdot)$ takes an $d$-dimension vector and turns it to a scalar with weight matrix $W \in \mathbb{R}^{d \times 1}$, and $\lambda$ is the weight for the first-stage similarity score. Both $W$ and $\lambda$ are optimized through back-propagation (see more about labeled data in §4.1).

With labeled data, we finetune $M_c$ to minimize the negative log-likelihood of the correct goal among the top-$k$ candidate goal list, where the log-likelihood is calculated as:

$$\ell(s, g_i) = -\log \left( \text{softmax} \left( \frac{\text{sim}_2(s, g_i)}{\sum_j \text{sim}_2(s, g_j)} \right) \right) \quad (2)$$

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 $M_c$, specifically BERT-base (Devlin et al., 2019) and DeBERTA-large finetuned on the MNLI dataset (He et al., 2021). We pick them due to their high performance on various tasks (Zhang et al., 2020e).4

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 $n$ ($n=1$).

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. Go to a store). 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 \texttt{unlinkable},

3\url{www.bing.com}

4\url{https://cutt.ly/oTx5gMM}. BERTScore measures the semantic similarity between a pair of texts, similar to the objective of our reranking.
taken from the reserved vocabulary of a pretrained model, as a placeholder “goal” to the top-\(k\) candidate goal list \(C(s)\), 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 \(\text{sim}_1(s, \text{unlinkable})\) is set to the lowest first-stage similarity score among the candidate goals retrieved by the first-stage model. Accurately labeling a step as \text{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 classification by assigning \text{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 train our models and 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 21\(k\) 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 Create a channel is Make a Youtube Channel. 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. Models based on search engines 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.

<table>
<thead>
<tr>
<th>Model</th>
<th>R@1</th>
<th>R@10</th>
<th>R@30</th>
</tr>
</thead>
<tbody>
<tr>
<td>SP</td>
<td>35.8</td>
<td>64.4</td>
<td>72.5</td>
</tr>
<tr>
<td>SBERT</td>
<td>30.6</td>
<td>53.3</td>
<td>63.4</td>
</tr>
<tr>
<td>BM25 (goal only)</td>
<td>30.5</td>
<td>51.6</td>
<td>61.1</td>
</tr>
<tr>
<td>BM25 (article)</td>
<td>9.3</td>
<td>35.3</td>
<td>49.2</td>
</tr>
<tr>
<td>Bing Search</td>
<td>28.0</td>
<td>47.9</td>
<td>-</td>
</tr>
</tbody>
</table>

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.\(^5\)

**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.\(^6\) As seen in the bottom half of Table 1, reranking is highly effective, as the best configuration brings a 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., 2021) outperforms BERT by 1.7\% on recall@1, matching the reported trends from BERTScore.\(^5\)

To qualitatively understand the benefit of the reranker, we further inspect randomly sampled 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 to only consider the action (e.g., learn) or the object (e.g., bike) and mistakenly rank those partially 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 predicts that “vinyl records” is closely related to “LP records” and outputs the correct goal while SP could not.

Second, we observe that the surrounding context and the goal of the query step are helpful in general. Incorporating both contexts brings a 3\% improvement in recall@1. While steps are informative,

\(^5\)We are unable to get the top-30 results from Bing search because the web queries only return top-10 search results.\(^6\)We only experiment with SP because it is the best retrieval model, providing a larger improvement headroom.
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).

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

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.

5 Manual Step Prediction Evaluation

The automatic evaluation strongly indicates the effectiveness of our proposed hierarchy discovery model. However, it is not comprehensive because the annotated hyperlinks are not exhaustive. We complement our evaluation with crowdsourced human judgments via Amazon Mechanical Turk (MTurk).

Each example of annotating 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 MTurk 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 A.

We select SP, DeBERTa, and DeBERTa with unlinkable prediction and \( \lambda = 0 \) (DeBERTa-UL) for comparison. We attempt to answer the following questions. First, does the performance trend shown in automatic evaluation hold in human evaluation? Second, can the unlinkable predictions help 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 linkable examples. Namely, those whose top-1 prediction is not predicted as unlinkable by the DeBERTa-UL model. Ideally, the linked articles from these examples should be helpful. The other
group contains unlinkable examples. For these, we evaluate the second-highest ranked prediction of the DeBERTA-Ul model. Ideally, the linked articles from these examples should be unhelpful.

The corresponding crowd judgment is shown in Figure 2. Comparing 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 unlinked for a step, the suggested decompositions of all models are more likely to be unhelpful. This implies the high precision of the unlinked prediction, effectively avoiding misleading predictions. Note that our study does not explicitly require subjects to carry out the task, but only annotates whether they find the instructions helpful.

6 Application to Video Retrieval

In addition to intrinsic evaluation, we take a further step to study the usefulness of our open-domain hierarchical KB to downstream tasks. We select video retrieval as the extrinsic evaluation task, which aims at retrieving relevant how-to videos for a textual goal to visually aid users. More formally, given a textual goal $g$, the task is to retrieve its relevant videos $v_g$ from the set of all videos, with a textual query $q$. Intuitively, our KB can be useful because videos usually contain finer-grained steps and verbal descriptions to accomplish a task. Therefore, the extra information presented in decomposed steps could benefit retrieving relevant videos.

6.1 Dataset Construction

We use Howto100M (Miech et al., 2019) for evaluation. It is a dataset of millions of instructional videos corresponding to over 23k goals. We construct our video retrieval corpus by randomly sampling 1,000 goals (e.g., record a video) with their relevant videos. The relevant videos $v_g = \{v_{g_1}, v_{g_2}, \ldots, v_{g_n}\}$ of each goal $g$ in the dataset are obtained by selecting the top 150 videos among the search results of the goal on YouTube.\footnote{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 highly ranked videos likely demonstrate the queried goal.}

Table 3: The Recall/Precision@N (%, ↑) and mean rank (MR, ↓) 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**.

\[
<table>
<thead>
<tr>
<th>Query</th>
<th>R/P@1</th>
<th>R/P@10</th>
<th>R/P@25</th>
<th>R/P@50</th>
<th>MR</th>
</tr>
</thead>
<tbody>
<tr>
<td>L0</td>
<td>12.1/81.7</td>
<td>59.8/42.8</td>
<td>71.9/20.8</td>
<td>77.9/11.3</td>
<td>41.60</td>
</tr>
<tr>
<td>L1</td>
<td>11.8/79.7</td>
<td>61.2/43.9</td>
<td>74.1/21.4</td>
<td>80.5/11.6</td>
<td>36.70</td>
</tr>
<tr>
<td>FIt.-L1</td>
<td>11.6/84.4</td>
<td>66.1/47.7</td>
<td>78.0/22.5</td>
<td>83.3/12.0</td>
<td>32.30</td>
</tr>
<tr>
<td>FIt.-L2</td>
<td>11.8/79.7</td>
<td>61.2/43.9</td>
<td>74.1/21.4</td>
<td>80.5/11.6</td>
<td>36.70</td>
</tr>
<tr>
<td>L0</td>
<td>11.2/82.6</td>
<td>59.2/45.2</td>
<td>71.8/22.1</td>
<td>77.8/12.0</td>
<td>43.11</td>
</tr>
<tr>
<td>L1</td>
<td>11.2/82.6</td>
<td>59.2/45.2</td>
<td>71.8/22.1</td>
<td>77.8/12.0</td>
<td>43.11</td>
</tr>
<tr>
<td>FIt.-L1</td>
<td>11.6/84.4</td>
<td>66.1/47.7</td>
<td>78.0/22.5</td>
<td>83.3/12.0</td>
<td>32.30</td>
</tr>
<tr>
<td>FIt.-L2</td>
<td>11.8/79.7</td>
<td>61.2/43.9</td>
<td>74.1/21.4</td>
<td>80.5/11.6</td>
<td>36.70</td>
</tr>
</tbody>
</table>

6.2 Setup

Since our KB is fully textual, we also represent each video textually with its automatically generated captions. For the search engine, we use Elasticsearch with the standard BM25 metric (Robertson and Zaragoza, 2009).\footnote{We find the performance of a neural model (BERT fine-tuned on query/video caption pairs) significantly lower than BM25 and therefore, we only report the results with BM25.} We denote the relevance score calculated by BM25 between the query $q$ and a textually represented video $v$ as $\text{Rel}(q, v)$.

We experiment with four different methods, which differ in how they construct the query $q$:

**L0**: **Goal only**. The query is the goal $g$ itself. This is the minimal query without any additional hierarchical information. The relevance score is simply $\text{Rel}(q, v) = \text{Rel}(g, v)$.

**L1**: **Goal + Children**. The query is a concatenation of the goal $g$ and its immediate children steps $\text{Ch}(g)$. This query encodes hierarchical knowledge that already exists in wikiHow. The relevance score is then defined as a weighted sum, $\text{Rel}(q, v) = w_g \text{Rel}(g, v) + w_s \sum_{s \in \text{Ch}(g)} \text{Rel}(s, v)$.

The weights $w_g$ and $w_s$ are tuned on a development set and set to 1.0 and 0.1 respectively.

**FIt.-L1**: **Goal + Filtered children**. The query is a concatenation of the goal $g$ and a filtered sequence of its children $\text{Ch}(g)$. Intuitively, decomposing a goal introduces richer information
Therefore, we perform filtering and only retain queries since it produces a set of more generalizable steps that are shared among multiple videos. Although steps in wikiHow articles are human-written, 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 L1 queries. More interestingly, we observe that queries using deeper hierarchies (Fil-L2) outperform the shallower ones (Fil-L1) in most cases. This is probably due to the fact that how-to videos usually contain detailed (verbal) instructions of a procedure, which are better aligned with more fine-grained steps found in Fil-L2.

In our qualitative study, we investigate how Fil-L2 queries with deeper hierarchies help retrieval. Table 4 list Fil-L1 and Fil-L2 queries for two goals. We find that the Fil-L2 queries are more informative and cover more aspects. For example, the Fil-L2 queries for stain cabinet and make avocado fries consist of the preparation, actual operations, and the post-processing steps, while the Fil-L1 query only contains the first one. In addition, we search the goals on Google and list the key moments of some randomly sampled videos. These key moments textually describe the important clips of the videos, and therefore they presumably also serve as the query for the goal. We find that the Fil-L2 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.

### 7 Decomposition Analysis

In this section, we study the properties of the hierarchies. First, what kind of steps are likely to be linked to another goal and are thus decomposed?  

### 6.3 Results

We report the precision@N, recall@N and mean rank (MR) following existing work on video retrieval (Luo et al., 2021) (see §B.2 for metric definitions). Table 3 lists the results. First, queries that encode hierarchies of goals (L1, Fil-L1 and Fil-L2) are generally more beneficial than queries that do not (L0). The steps of goals enrich a query and assist the retrieval. Second, video-oriented filtering yields significant improvement over the unfiltered L1 queries since it produces a set of more generalizable steps that are shared among multiple videos. Although steps in wikiHow articles are human-written, 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 L1 queries. More interestingly, we observe that queries using deeper hierarchies (Fil-L2) outperform the shallower ones (Fil-L1) in most cases. This is probably due to the fact that how-to videos usually contain detailed (verbal) instructions of a procedure, which are better aligned with more fine-grained steps found in Fil-L2.

<table>
<thead>
<tr>
<th>Goal</th>
<th>Stain Cabinet</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fil-L1</td>
<td>Purchase some stain colors to test</td>
</tr>
<tr>
<td>Fil-L2</td>
<td>Buy cloth with which to apply the stain</td>
</tr>
<tr>
<td></td>
<td>Unscrew the cabinet from the wall</td>
</tr>
<tr>
<td></td>
<td>Clean your workspace</td>
</tr>
<tr>
<td>KM</td>
<td>Make Avocado Fries</td>
</tr>
<tr>
<td>Fil-L1</td>
<td>Bake the avocado fries until they are golden</td>
</tr>
<tr>
<td></td>
<td>Dip the avocado wedges into the egg and then the breadcrumbs</td>
</tr>
<tr>
<td>Fil-L2</td>
<td>Preheat the oven</td>
</tr>
<tr>
<td></td>
<td>Peel and pit the avocados</td>
</tr>
<tr>
<td></td>
<td>Cut your avocado in half and remove the stone</td>
</tr>
<tr>
<td></td>
<td>Let rise</td>
</tr>
<tr>
<td></td>
<td>Finished, cool and enjoy</td>
</tr>
<tr>
<td>KM</td>
<td>2 large avocados ...</td>
</tr>
<tr>
<td></td>
<td>pinch of salt, pinch of pepper</td>
</tr>
<tr>
<td></td>
<td>two eggs, beaten ...</td>
</tr>
<tr>
<td></td>
<td>bake at 425F 20 min until golden brow ...</td>
</tr>
</tbody>
</table>

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 o7uVUmPph61.

but also introduces noise, since certain steps may not visually appear at all (e.g., enjoy yourself). Therefore, we perform filtering and only retain the most informative steps, denoted by $Ch'(g)$. Specifically, to construct $Ch'(g)$ for a goal $g$, we use a hill-climbing algorithm to check each step $s$ from $Ch(g)$, and include $s$ into the query only if it yields better ranking results for the ground-truth videos in the training set $v_{train}^{10}$. The relevance score is defined as $Rel(q, v) = w_qRel(g, v) + w_s \sum_{s \in Ch'(g)} Rel(s, v)$, where $w_q$ is set to 1.0 and $w_s$ is set to 0.5 after similar tuning.

### Fil-L2: Goal + Filtered children + Filtered grand-children

The query is the concatenation of the goal $g$ and a filtered sequence of its immediate children $Ch(g)$ and grandchildren $Ch(s)$ ($s \in Ch(g)$). These filtered steps are denoted by $Ch'(g + Ch(g))$. This two-level decomposition uses the knowledge from our KB, therefore including lower-level information about the execution of the goal. We perform the same filtering algorithm as in Fil-L1, and we define $Rel(q, v) = w_qRel(g, v) + w_s \sum_{s \in Ch'(g+Ch(g))} Rel(s, v)$. $w_q$ is set to 1.0 and $w_s$ is set to 0.5.

10See Algorithm 1 in Appendix for more details.
Second, what do the decomposed steps look like?

We group steps into two clusters. The first contains the immediate steps of a goal \( s \in Ch(g) \) whose prediction is not unlinkable. The second contains the decomposed steps of the steps in the first cluster \( s' \in Ch(s) \). 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 our assumption that the decomposition would lead to more fine-grained realizations of a complex procedure. Some other more abstract actions such as “learn” and “decide” also increase in frequency, as some low-level goals are explained with more complex steps.

8 Related Work

Linking Procedural Events To the best of our knowledge, two other pieces of work Pareti et al. (2014); Lagos et al. (2017) tackled the task of linking steps in procedures to other procedures. Both of them also drew the procedures from wikiHow. While we share the same task formulation, our work makes several additional contributions: (1) a retrieval-then-rerank method significantly increases linking recall; (2) more comprehensive experiments with the manual and the 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) and videos (Alayrac et al., 2016; Yang et al., 2021a), 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., 2021b; 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 work has also leveraged 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 a comprehensive intrinsic and extrinsic evaluation of the resulting knowledge-base.

9 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 the 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, or to other modalities (e.g., video clips). Future work

\[ \text{e.g., https://www.instructables.com/, https://www.diynetwork.com/how-to} \]
could also explore other usages such as comparing and clustering procedures based on their deep hierarchies; or applying the procedural knowledge to control robots in the situated environments.

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), the Amazon Alexa Prize TaskBot Competition, and the NSF (Award 1928631). 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, NSF, or the U.S. Government.

We thank Ziyang Li and Ricardo Gonzalez for developing the web demo, John Wieting for support on implementation, and the anonymous crowd workers for their annotations.

References


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:

\[ p = \{p_1, ..., p_n\}, \text{ relevant videos } v_p \]

Result: \[ \text{best_query} \]

\[ k \leftarrow 15; \]

\[ \text{best_query} \leftarrow [g]; \]

\[ \text{min_cost} \leftarrow f(\text{best_query}, v_p) ; \]

\[ r \leftarrow \min(n, k); \]

while \( r \geq 0 \) do

\[ \text{in_cost} \leftarrow 1e10; \]

for \( p \in p \) do

if \( p \notin \text{best_state} \) then

\[ \text{query} \leftarrow \text{best_query}, p; \]

\[ \text{cost} \leftarrow f(\text{query}, v_p); \]

if \( \text{cost} < \text{in_cost} \) then

\[ \text{in_cost} \leftarrow \text{cost}; \]

\[ \text{in_query} \leftarrow \text{query}; \]

end

end

if \( \text{in_cost} < \text{min_cost} \) then

\[ \text{min_cost} \leftarrow \text{in_cost}; \]

\[ \text{best_query} \leftarrow \text{in_query}; \]

else

1. break

r \rightarrow 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 (Heilbron et al., 2015), a popular benchmark for human activity understanding, uses the same 203 activities across different splits; Yang et al. (2021b) 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
the procedures). We evaluate the quality of the induced schemas by matching them with unseen instantiations. The large-scale DARPA KAIROS project\(^\text{13}\) 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., 2021b) 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)

\[
\begin{align*}
\text{recall}@N &= \frac{1}{M} \sum_{i=1}^{M} \sum_{v_j \in v_{gi}} \mathbb{1}(r(v_j) \leq N) \\
\text{precision}@N &= \frac{1}{M} \sum_{i=1}^{M} \sum_{v_j \in v_{gi}} \mathbb{1}(r(v_j) \leq N) \\
\text{MR} &= \frac{1}{M} \sum_{i=1}^{M} \sum_{v_j \in v_{gi}} r(v_j) \\
\end{align*}
\]

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

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. Fine-tuning 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.
The SP model is licensed under BSD 3-Clause "New" or "Revised" License ElasticSearch is licensed under Apache License 2.0. HowTo100M is licensed under Apache License 2.0.