Distilling Task Knowledge from How-To Communities

Cuong Xuan Chu
Max Planck Institute for Informatics
Saarbrücken
Germany
cxchu@mpi-inf.mpg.de

Niket Tandon
Allen Institute for Artificial Intelligence
Seattle
USA
nikett@allenai.org

Gerhard Weikum
Max Planck Institute for Informatics
Saarbrücken
Germany
weikum@mpi-inf.mpg.de

ABSTRACT

Knowledge graphs have become a fundamental asset for search engines. A fair amount of user queries seek information on problem-solving tasks such as building a fence or repairing a bicycle. However, knowledge graphs completely lack this kind of how-to knowledge. This paper presents a method for automatically constructing a formal knowledge base on tasks and task-solving steps, by tapping the contents of online communities such as WikiHow. We employ Open-IE techniques to extract noisy candidates for tasks, steps and the required tools and other items. For cleaning and properly organizing this data, we devise embedding-based clustering techniques. The resulting knowledge base, HowToKB, includes a hierarchical taxonomy of disambiguated tasks, temporal orders of sub-tasks, and attributes for involved items. A comprehensive evaluation of HowToKB shows high accuracy. As an extrinsic use case, we evaluate automatically searching related YouTube videos for HowToKB tasks.

Keywords

HowTo Commonsense, Knowledge base construction

1. INTRODUCTION

Motivation and Problem In the last decade, knowledge graphs about entities and their relationships have become key assets for search engines, to answer entity-centric queries, and as semantic features for recommendations. However, a good fraction of user queries are not about entities, but rather seek information on problem-solving tasks, so-called *how-to queries*. Examples are: How do I build a fence around my house? How do I repair the flat tire on my bicycle? How can I avoid high altitude sickness on my upcoming trip to the Himalayas? Knowledge graphs do not support this class of queries at all. Conventional Internet search, returning ten blue links, does a decent job, but a knowledge base with well organized knowledge on tasks and their problem-solving steps could boost the search result quality and user satisfaction.

Recently, online communities have compiled extensive information about such how-to tasks. Examples are WikiHow, snapguide, eHow, Howcast, and others. In this paper, we leverage WikiHow (`www.wikihow.com`), probably the largest of these communities. WikiHow contains a wealth of instructive contents on a wide variety of tasks, breaking them down into sub-tasks with detailed descriptions and pictorial illustrations. This is a great source for humans; however, it is merely in textual form and far from rigorously represented machine knowledge that can directly support how-to queries. The problem addressed and solved in this paper is to distill and semantically organize task knowledge from the raw input of the WikiHow community.

Approach and Contribution We have developed methods for extracting and cleaning task knowledge from WikiHow. The resulting knowledge base, called *HowToKB*, contains 1.29 Million tasks and sub-tasks organized into a clean taxonomy. Each task or sub-task is represented by a frame with attributes for parent task, preceding sub-task, following sub-task, required tools or other items, and linkage to visual illustrations. As the text in the raw input is often noisy and ambiguous, we have devised techniques for canonicalizing tasks and items: grouping all information for the same meaning into one frame, and distinguishing different meanings by different frames. For example, the WikiHow phrase "use a keyboard" leads to one frame about tasks that involve a computer keyboard and another frame about playing music on a keyboard.

Prior work along these lines is scarce. The most related papers are the work on procedural knowledge by [56] and our own work on Knowlywood [49]. [56] also leverage WikiHow, but mostly treats it as a text corpus for query expansion. There is no attempt to canonicalize sub-tasks (i.e., identify the same-meaning sub-tasks across different tasks) and to infer structured attributes for tasks. [49] constructs an activity knowledge bases from narrative texts like movie scripts. These general activities are not related to how-to tasks, and the methodology in this work is very different from the current paper. [49] uses computationally expensive semantic-parsing techniques, whereas HowToKB is largely built by light-weight clustering.

Our methodology first applies Open-IE techniques [12] to WikiHow articles, in order to extract – noisy and ambiguous – candidates for task and sub-tasks. Subsequently, we use judiciously devised clustering techniques to clean and organize these candidates, and to infer attribute values. To canonicalize tasks and sub-tasks, we leverage word embed-

WWW 2017, April 3–7, 2017, Perth, Australia.
ACM 978-1-4503-4913-0/17/04.
http://dx.doi.org/10.1145/3038912.3052715

dings to distinguish different meanings of the same phrase (e.g., "use keyboard").

The quality of the constructed HowToKB knowledge base has been evaluated by crowdsourcing. For all task attributes, we achieve precision above 80 percent, and well above 90 percent for some attributes. As an extrinsic use case, we developed a technique for searching relevant YouTube videos to tasks in HowToKB. HowToKB datasets are available at `http://www.mpi-inf.mpg.de/yago-naga/webchild/HowToKB`.

The salient contributions of this paper are:

- the first method for the automatic construction of a large knowledge base on how-to tasks,
- the publicly available knowledge base HowToKB,
- experiments on the construction methodology, the quality of HowToKB, and its benefits for YouTube search.

2. RELATED WORK

Mining procedural knowledge has been pursued across several domains (e.g., [3, 24]). The input to mine this knowledge has come from multiple languages [17], multiple modalities [22] and, when necessary, compiled by experts [3] or through crowdsourcing [40, 54]. In order to represent this knowledge, both formal [43, 14] and less formal representations [7, 52] have been proposed. This effort is cast into several tasks that aim at mining and organizing this knowledge based on these representations [49, 26, 40]. Various methods at different levels of supervision have been proposed to solve these tasks [40]. To cover the related work, we divide it into: input and representations, and task and models.

Input and Representations.

In terms of input, the closest to this paper is the prior work on process knowledge by [56]. That work also tapped into the contents of the WikiHow community. In representing this knowledge, unlike our representation, they made no attempt to canonicalize tasks and semantically organize their attributes into frames. In terms of representation, the closest to our frame based representation is Knowlywood [49]. Knowlywood frames are organized similar to ours. However, they do not contain the important sub-task relationship because they aimed at general activities such as romantic dinners, wedding speeches, etc., rather than how-to knowledge for task solving. By the nature of its inputs, Knowlywood resulted in a heavy bias towards activities that are salient in movies, such as dramatic farewells or killings. This is very different from the nature, and scope of the knowledge in HowToKB.

Early work on the formal representation of procedural knowledge was pioneered by Minsky [32], who introduced semantic frames with slots to represent common situations. Building on these ideas, Schank and Abelson [43] introduced ordering in frames with scripts and plans, to understand stories and actions. While a script can relate one event to a subsequent event, Fikes and Nilsson [14] introduced the Strips action representation that relates an event to a state of the world before and after the event. Fillmore's frame semantics [15] linked frames to lexical words, leading to the development of FrameNet [1], containing scenario based frames that are evoked by certain verbs. These complex and rich representations were populated manually.

Often, these frame based roles do not generalize, and so, annotated corpora cannot be shared across domains, e.g., the semantic roles in the science domain [29] are very different from the news domain [9]. With the aim of generalizability and automated learning of representation, syntactic structures were used for representation. Chambers et. al [7, 8] describe a statistical co-occurrence model of (verb, dependency) pairs, and Rudinger et. al introduce a language modeling approach [42] that is fairly generalizable. However, these approaches achieve generalizability at the expense of decreased expressive power. These learned representations, often lead to noise due to over-generalization [2].

Maintaining a middle ground, PropBank [25] reduced the representation complexity by introducing generic roles, while adding a layer of semantic annotation over the syntactic structure of Penn TreeBank. This eases representation learning, while keeping much of the expressive power. The generic roles `ARG0-ARG5`, `ARGMs` often exhibit inconsistent interpretations for different verbs, but `ARG0` and `ARG1` preserve the semantics of `agent` and `patient`, respectively [51]. HowToKB's frame roles can be seen as a subset of PropBank roles, because Open IE tuples [12] can be aligned to PropBank roles (typically, `ARG0` is the Open IE subject, `ARG1` is the Open IE object, and additional roles `ARG2-ARG5` are the Open IE context, while `ARG-TIME` and `ARG-DIR` can be aligned to Open IE time and location roles).

To learn such representations, we do not rely on either of PropBank, FrameNet, or VerbNet [44], as they do not contain sufficient training examples on how-to tasks and are costly to extend [21]. In an attempt to overcome this sparsity, [38] describes a method to construct WordNet-based distributional profiles for frames to link out-of-vocabulary lexical units to existing frames. Other domain adaptation methods such as [10] make similar efforts, with limited success and marginal improvements. Besides, these repositories are verb-specific, whereas HowToKB would require a verb-phrase repository. For example, they contain an entry `dig` or perhaps `dig into`, but not the phrase `dig into literature`. Verbal phrases, with direct object, dramatically reduce the ambiguity of the event.

Task and Models.

Populating machine-readable, organized events has been cast into different tasks, related to schema-based or schema-free extraction and organization of the extracted information. Our task is a schema-based (i.e., with fixed roles). The closest to our task is our own prior work Knowlywood [49]. However, Knowlywood required a very specific methodology for extraction. In particular, we used computationally expensive techniques for semantic parsing that jointly perform word sense disambiguation via WordNet and VerbNet. HowToKB is based on much more light-weight methods to induce distributional senses of a task. As for a downstream application, [56] also tapped into WikiHow, but for a very different task. They support keyword search for how-to contents by means of query expansion in an IR-style manner. Our use case (see Section 6) resembles this, but our task of organizing WikiHow contents is very different.

Among the different related tasks, Semantic Role Labeling (SRL) [18] has been most widely studied. SRL systems are typically trained over PropBank annotated corpus [25]. Early SRL methods, such as [53], jointly model the SRL task, by considering global features looking at all arguments of a verb together. More recently, neural network models [27, 41] have gained popularity. [41] build on syntactic dependency paths to learn embedding representations for semantic role labeling. Unsupervised SRL [26, 52] induces semantic roles automatically from unannotated data. This line of work may be useful in discovering new semantic frames and roles, but does not provide us with the frame attributes that we need. A related, but more difficult task than SRL is Frame Semantic Parsing (FSP) [11], for aligning sentences to FrameNet frames, which are much richer than PropBank. Early work on this task includes rule-based system and classifiers to predict frames and their arguments in text. [23] divided the task as identifying targets that could evoke frames in a sentences, identifying the correct semantic frame for a target, and determining the target words that fill the semantic roles of a frame.

When extracting semantic roles, our task is quite different from FSP and SRL. The latter are designed for verbs, and not the kind of verb phrases that we consider. SRL and FSP systems have the disadvantage that they do not generalize, while our domain for HowTo tasks is very different from the available annotated training corporas. While FSP operates on one sentence at a time, it relies on the FrameNet repository that already organizes frames [11]. We do not assume the existence of a repository like FrameNet. SRL typically operates one sentence at a time; so these methods do not infer connections across sentences.

Frames have been organized temporally, hierarchically, and as clusters. Temporally organizing events has been widely studied in the context of causal relations [46]. Girju et. al [19] use modified Hearst patterns to extract a large number of potential cause-effect tuples, but focus only on nominal events. This line of work does not focus on constructing rich frames of events, or temporal ordering of frames. For temporal order learning, graph algorithms [40], Markov Models [35], and permutation priors capturing event ordering constraints [16], and most recently neural models [39] have been proposed. As our input source provides very rich temporal structure, we do not rely on these models for temporal learning, though they might help to improve the temporal edges in our graph. However, this would be a non-trivial extension because these methods operate on events and not event frames.

Besides temporal organization, the semantic organization of events is typically approached as a graph based problem. Several methods use the WordNet taxonomy to organize events, such as in the denotation graph of [57], which are constructed over crowdsourced simple sentences describing images. The work by [40] is particularly interesting because it performs both temporal, and clustered hierarchical organization that we need. There is work on semantically organizing verbs and verb phrases in general, such as [28, 34, 4, 20]. However, in all these methods, the graph nodes are merely verbs, and not the rich frames as in HowToKB.

3. INFORMATION EXTRACTION FOR TASK KB CONSTRUCTION

3.1 Concepts and Notation

In natural language, how-to tasks are typically referred to by verbal phrases (e.g., "repair" or "repair a tire"). To a first degree, WordNet [13] provides a repository of such phrases. However, WordNet mostly contains single verbs and misses out on many compound phrases. Also, noun phrases may express tasks as well (e.g., "tire repair"). Therefore, we consider extended phrases, not necessarily present in WordNet, as cues for expressing tasks.

Definition 3.1 (Normalized extended phrase)**.** *A phrase (noun phrase or verb phrase) whose head-word is present in WordNet, is called an extended phrase (e.g., "tire repair" has head-word "repair"). These phrases can be normalized either strongly or weakly. Strong normalization reduces the phrase to its head word, while weak normalization performs stemming and removes leading articles.*

Definition 3.2 (Task phrase)**.** *A* task phrase *consists of (v, prep, o) where v is a verb or a normalized extended verb phrase and o is a noun or a normalized extended noun phrase and prep is a preposition linked with v. For example,* paint on wall .

Task phrases are the anchors for compiling knowledge on how-to tasks into semantic frames.

Definition 3.3 (Task frame)**.** *A* task frame *is a task phrase enhanced with*

- *attributes: location, time, participating agent, participating object, category;*
- *hierarchical relations: parent task and sub-tasks;*
- *temporal relations: previous task and next task.*

Finally, images or videos showing the tasks are also stored in a frame. Each attribute can have zero, one or multiple entries with confidence scores $\in [0, 1]$.

Table 1 shows an example for a task frame.

attribute	notation	value	type
location	$loc(a)$	*house*	noun-phrase
time	$time(a)$	*weekend*	WN-time
parti. agent	$parta(a)$	*student*	WN-living
parti. object	$parta(a)$	*brush*	WN-non-living
category	$cat(a)$	*house*	WH-categories
parent-task	$parent(a)$	*decorate house*	*task*
sub-task	$sub(a)$	*clean wall*	*task*
prev-task	$prev(a)$	*buy paint*	*task*
next-task	$next(a)$	*dry the wall*	*task*

Table 1: Example frame on task $a =$ *paint a wall* .
Acronyms used: WN = WordNet, WH = WikiHow

The WikiHow community organizes its contents with one article dedicated to one task. The articles are semi-structured, containing: (i) textual and visual content, (ii) article level statistics like number of views and quality ratings and (iii) a category selected from a hierarchy of more than 3000 categories. A WikiHow article describes various how-to methods to solve a task. Each method mentions the required tools and presents a step-by-step procedure. A step contains a heading, denoting the step's main task, followed by a textual description and representative images.

To distill knowledge from a WikiHow article into machine-readable form, we posit the following structure:

- Hierarchical structure: The task in an article is split into several solution methods. The method has a heading that indicates the main task for the solution method, which can be seen as a sub-task of the article-level task. Similarly, a step-level headings can be interpreted as sub-tasks of the per-method sub-tasks.

- Temporal structure: Method-level and step-level tasks have temporal ordering. Suppose a method consists of a sequence of three step-level tasks, a_1, a_2 and a_3. We interpret this as a_1 preceding a_2 and a_2 preceding a_3.
- Categorical structure: Tasks at all levels can be assigned to the same category as the category for the entire article.
- Frame attribute structure: Every method contains a list of required tools and other objects which become the participants of a task. Steps are depicted by images or videos; these visual contents become the visual representative of a step-level task.

We obtain sub-tasks at all levels from the step-by-step headings in an article. We next describe our IE model (Section 3.2), and organization model (Section 4). Fig. 1 provides a high-level architecture of our two-stage approach.

3.2 Information Extraction Method

Our IE goal is to extract tasks, participants, location and time from a WikiHow paragraph heading. For example, given, heading: *Patch up any holes or breaks in the wall with compound or caulk*; extract tasks: *patch up hole* and *patch up break*, with the participating object *compound, caulk* and location *the wall*, respectively. The method to perform these extractions must meet the following requirements:

- *Semantic role labels.* We have five roles: task, participating object, participating agent, location and time.
- *Open domain.* Task knowledge is evolving and open domain; so the IE method must be domain agnostic.

There are three methodology choices.

- Knowlywood semantic parsing pipeline [49]: Carrying over our prior work to the new setting of distilling task knowledge would provide the semantic role labels, but rely on VerbNet [44] and WordNet dictionaries for additional sense disambiguation, and not fully covering the new domain.
- SRL systems [37]: Even if it is possible to align our role labels to existing PropBank roles, e.g. `ARG0` aligns with agent, SRL systems are difficult to adapt to a new domain like ours. This necessitates retraining that requires training data, that is not available. Thus, using an existing SRL system for acquiring task knowledge would need a large amount of labeled training data that we do not have.
- Open IE systems: ReVerb [12] and Ollie [30] are open domain. In an Open IE tuple, the subject is the agent, object(s) are participating objects, while, location and time are provided as its pipeline includes an SRL system for location and time. Tasks can be easily derived as `subject + object`.

This leaves us with Open IE as our primary method for task IE. Specifically, we use the Open IE 4.2 software because it is the best performing Open IE system (cf. Section 5.1), with some extensions as described next, related to preprocessing the input, and post-processing the role values, and, esp. type restricting `location`, and `time` to overcome Open IE's SRL noise.

Open IE 4 expects sentences or phrases that begin with a subject, while WikiHow headings are imperatives, typically starting with a verb. We supplement the input WikiHow headings with a prefix subject *You*, before passing to Open IE (e.g. *You* `paint a wall`). This simple change greatly increases the coverage of triples.

Open IE 4 yields subject-predicate-object tuples, where each of the three components is a lexical phrase. We normalize the output from Open IE in two different ways: For *weak normalization* (see Definition 3.1), we drop stop-words from the task phrase. For *strong normalization* (see Definition 3.1), we concatenate the head verb and head noun making a task name. For example, the original task derived from Open IE tuple extracted phrase *paint a bedroom wall* has the weak normalization *paint bedroom wall* and the strong normalization *paint wall* .

To overcome Open IE's frequent errors in distinguishing `location`, and `time` (cf. Section 5.1), we impose semantic type restrictions on the head word of the role value. For example, we restrict that the head word of `time`, could only be one of the hyponyms of WordNet time synsets (providing us with a lexicon of time including day, night, summer, etc.). Prepositions such as `in`, and `at` often cause the SRL to fail, and by restricting `time` with a lexicon, we fall back to `location` whenever a role value is not acceptable as a time.

4. ORGANIZING THE TASK KB

Semantically organizing the extracted task phrases and their candidate frames is crucial for building a clean and consistent KB. The KB construction, as described in the previous section, has redundancy from the WikiHow contents. For instance, considering two frames for tasks *paint a wall* and *color a ceiling* , they are highly related to each other and heavily overlap in their sub-tasks and frame attributes. Grouping such nearly synonymous task frames together can make the KB more precise and consistent. A second issue to address in KB organization is the disambiguation of task names. For instance, the task *use a keyboard* can can be interpreted as *use a music keyboard* or *use a computer keyboard* . It is important to separate these two senses, and have one frame for each of the two meanings.

To cope with these issues, we cast semantic organization into a clustering problem. We devised a hierarchical clustering algorithm, with a computationally inexpensive bottom-up phase followed by a top-down algorithm.

4.1 Similarity Measures

Clustering the task frames requires computing pairwise similarity measures. We define a multi-dimensional similarity model, comprising similarity measures per frame attribute.

Categorical Similarity: The WikiHow category taxonomy is a tree, containing over 3000 categories with a maximum depth of 8 levels. We use the Wu-Palmer measure [55] to compute the similarity between two categories c_1, c_2:

$$f_{cat}(c_1, c_2) = \frac{2 * depth(lca(c1, c2)) + 1}{depth(c1) + depth(c2) + 1} \tag{1}$$

Lexical Similarity: This measure computes the lexical similarity between two task names (surface forms of two tasks a_1, a_2) as:

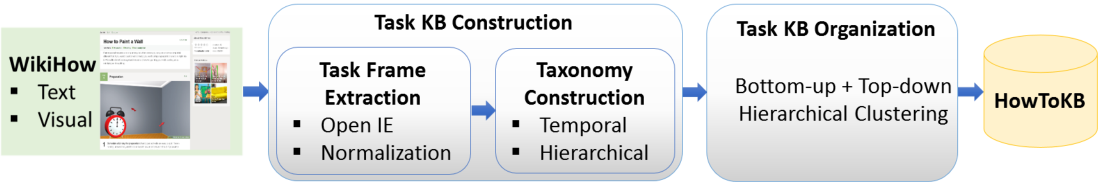

Figure 1: HowToKB System Overview

$$sim(surface(a1), surface(a2)) \\ = f_{w2v}(verb(a1), verb(a2)) * f_{w2v}(noun(a1), noun(a2)) \tag{2}$$

Here, f_{w2v} is an embedding-based similarity. Specifically, we are using Word2Vec [31]. To this end, we train a distributional representation over part of speech (POS)-tagged WikiHow data. The similarity between two POS-tagged words is the cosine similarity between their embeddings:

$$f_{w2v}(w_1, w_2) = cosine(E_{w_1}, E_{w_2}) \tag{3}$$

where E_w is the embedding of word w.

Vector similarity: Location, time, participating agent and participating object are all vectors of strings. To compute the similarity of two vectors $[L_1, L_2]$ (e.g., location vectors of two task frames), we use the weighted Jaccard coefficient, with weights coming from equation 3, as:

$$f_{list}(L_1, L_2) = \frac{\sum_{w_i \in L_1} \sum_{w_j \in L_2} f_{w2v}(w_i, w_j)}{|L_1||L_2|} \tag{4}$$

Analogously, for attributes that are lists of tasks, like prev-task, next-task, sub-task and parent-task, we plug the task name similarity from Equation 2) into Equation 4.

4.2 Combining Features

The similarity between two task frames can now be computed as follows:

$$sim(a_1, a_2) = \frac{1}{1 + exp(-f_{sim}(a_1, a_2))} \tag{5}$$

Here, $f_{sim}(a_1, a_2)$ is a linear combination of all features:

$$f_{sim}(a_1, a_2) = w_0 + \sum_{i=1}^{N} w_i f_i(a_1, a_2) \tag{6}$$

where $f_i(a_1, a_2)$ is the similarity measure for the i^{th} attribute of the task frames. We learn the parameters $w_i, i \in [0, N]$, by training a logistic regression model.

For this purpose, we manually create a training set of 5500 data points (5000 training points, 500 test points) for the logistic regression model, using two labels: near-synonymous frames or dissimilar frames.

For efficient estimation of the – very sparse – pairwise similarity matrix between all task frames, we introduce pruning of dissimilar pairs. The empirical observation on which this heuristics is founded, is:

Two task frames are dissimilar with an empirical confidence of 99.9% if a combination of their categorical and lexical similarity is less than a threshold.

Based on this observation, we first compute the overall similarity of two task frames by using only two of the basic similarity measures, a fairly light-weight computation. Only for the pairs that survive this filter step (i.e., could potentially be near-synonymous), we need to perform the more expensive full computation using all similarity measures.

4.3 Clustering Algorithm

As we do not know the appropriate number of clusters, we use hierarchical clustering until reaching a stopping criterion. There are two approaches to hierarchical clustering: agglomerative bottom-up and divisive top-down. These have cubic or quadratic complexity, though faster approximation algorithms are known. This is still too slow when the number of nodes is large, as in our case. Therefore, we devise a two-phase method, which first performs an aggressive bottom-up clustering to construct *super-clusters* and then splits these in a top-down manner as needed. This allows for substantial pruning of the search space, in the bottom-up phase, which implies starting with much smaller input in the top-down phase.

Bottom-up phase.

The bottom-up algorithm starts with lexical grouping based on Equation 2. This clusters together all task frames with the same strongly normalized name. For instance, two frames *paint the living room wall* and *paint the bedroom wall* have the same normalization *paint wall* and are grouped together. As a result, we can group over 1.2M task frames into approximately 375K groups, which we call *lexical clusters*. For each of the resulting clusters, we use the strong normalization as a key for subsequent steps.

Next, the bottom-up algorithm merges lexical clusters by the similarity of their keys, using Equation 2. This results in *distributional clusters*. By our empirical observation on dissimilarity (see above), we avoid almost all false negatives this way. However, we produce false positives: frames in the same cluster that should really be kept apart. This is taken care of in the following top-down phase.

Top-down phase.

Each of the coarse-grained super-clusters is now refined by recursive splitting. Different super-clusters can be processed in parallel.

We use a simple heuristics for splitting clusters. We identify the two most dissimilar frames within a cluster, use them as seeds of two sub-clusters, and then assign all other frames to the closer one of the seeds.

These steps are repeated for each of the clusters until a stopping criterion is satisfied. We considered the Bayesian Information Criterion [45] to decide when to stop splitting, but it turned out that simple thresholding on average intra-cluster similarity is very effective. So we chose this alternative, giving us more flexibility in producing coarser or more fine-grained clusters (depending on the threshold value).

5. EXPERIMENTS

Table 2 summarizes the size of the HowToKB: the number of HowTo tasks, before and after organization (ungrouped and grouped task), their attributes and the associated images. We estimate the precision by extensive sampling, with assessment by crowdsourcing. Table 3 gives an anecdotal example from HowToKB.

We conducted extensive experiments to assess the viability of our approach and the quality of the resulting HowToKB. Our experiments cover the two modules developed in this paper: knowledge extraction and KB organization. Subsection 5.1 reports on the quality of the extraction module. Subsection 5.2 reports on the quality of the KB organization method. Subsection 5.3 reports on the overall accuracy of the resulting HowToKB. For each module, we compare against various baseline competitors.

	Ungrouped	Grouped	Precision
#Tasks	≈1.2M	≈0.5M	0.90
#Attributes per task	4.8	12.0	0.90
#Images per task	0.8	2.0	1.00

Table 2: HowToKB statistics

Task: ***paint a bedroom wall***	
Location	bedroom, house
Time	day, weekend
Agents	man, husband
Prop	paint, wall, paint brush
Parent	decorate the house
Previous	clean the wall, cover floor
Next	dry wall, redo floor
Sub-tasks	hit the edges first with paint, dip the roller into the paint, roll the paint onto the wall

Table 3: Anecdotal examples from HowToKB

5.1 KB Construction

To construct a task-frame from WikiHow contents, we rely on an Open IE system. Assuming that the input WikiHow text is perfectly correct, we measure the noise introduced while structuring the WikiHow text into task-frames using Open IE vs. other baselines.

Baselines.

Open IE systems process sentences to extract SPO triples, where O is a list of objects that can additionally include context like a role label (e.g. location, time). We consider different Open IE systems: ReVerb, Ollie, and Open IE 4.2. We compare these against our wrapper that post-processes Open IE 4.2 results.

Evaluation setup.

Our KB construction pipeline consists of extracting tuples from sentences, and structuring them into frames. We evaluate both these aspects. For triple extraction evaluation, we construct a dataset of 50 random sentences sampled from WikiHow. Three human judges extract 110 SPO triples from these 50 sentences, forming the ground truth for triple extraction. To measure the noise introduced by the Open IE systems over these 50 sentences, we use the standard F1 measure.

To construct a ground truth for task-frames, we crowdsource the validation of 150 task-frames on crowdflower.com. We validate every attribute value pair in a task frame, by asking the turkers if the attribute value for a task is *one of the most likely* values for the attribute, under a given context. The context provides additional background information to the turkers about the task, this could be the title of the WikiHow article from which we mined a task. We aggregate the scores from three judges and retain those frame attribute values where at least two judges agree, resulting in the ground truth that we will refer to as $G_{validation}$. A sample question posed to turkers is *In some context like* decorate your house *, one of the most likely previous tasks when we* paint a wall *could be* pick a color .

Results.

Which Open IE system is the best? We compare the F1 scores against the triple ground truth for each of the Open IE systems and find that Open IE 4.2 has the maximum F1 of 72.07%, followed by ReVerb at 58.59%, and, Ollie at 45.38%. To confirm if these different systems extract complementary triples, we perform ensembling of these systems, which leads to a collective F1 of 73.01%. The ensemble is slower, and provides marginal gains to Open IE 4.2. Open IE 4.2 has the additional benefit of semantic role labeling (location, and time). Therefore, Open IE 4.2 is the best performing system.

Open IE systems attach a confidence score to every SPO triple. We empirically investigate the F1 scores at thresholds between [0,1] at a step size of 0.05. We found the optimal confidence score at 0.45 which gives the best F1 score of 72.07% (precision of 75.47% and recall of 68.9%). We observe that the maximum precision at any threshold is 82.5% (at a lower recall level) showing that the default confidence score does not yield very high quality results. As our goal in KB construction is to minimize the noise, the default confidence score does not support our goal. Our Open IE wrapper post-processes the triples (cf. Section 3.2) leading to a marginal increase in F1 to 72.34%. Importantly, the precision increases dramatically from 75.4% to 97.1% at the cost of being more selective (recall drops from 68.9% to 57.6%). So our additional wrapper on top of Open IE 4.2 is decisive for building a high-quality KB.

Our Open IE 4.2 wrapper leads to 1.29 million task-frames, derived from 1.94 million high quality triples from 1.96 million sentences of 168,697 WikiHow articles. Table 4 reports the high quality of the extracted task-frames measured against the ground truth $G_{validation}$. This demonstrates that our KB construction stage is robust, eliminating most of the noise.

5.2 KB Organization

The basis of KB organization is clustering which in turn relies on the quality of the similarity function. As discussed in Section 4.2, our similarity function achieves very high accuracy. Using all the features, it reaches a precision of 97.6% on the test set.

Task attribute	Total	# Correct	Precision
Task phrase meaningfulness	150	149	0.99
Location	3	3	1.0
Participating Object (prop)	188	185	0.98
Participating Living Being	17	15	0.88
Parent task	106	106	1.0
Previous task	106	101	0.95
Next task	103	100	0.97
Sub task	108	107	0.99

Table 4: Quality of task-frames extracted by our Open IE 4.2 wrapper

In the experiment described in this section, we assess the quality of the clusters obtained through our method and through the baselines.

Baselines.
As we devised a hybrid bottom-up/top-down clustering method, a natural baseline thus is a traditional top-down divisive algorithm. To highlight the effectiveness of the bottom-up stage in our hybrid method, we compare against a top-down only algorithm . As baseline, we use a single-linkage based top-down algorithm [33] (which was also a part of our hybrid algorithm), with the same threshold parameters are in our model. While faster approximations of the single-linkage method exist [47], we choose the basic method for simplicity, and because the choice of heuristic does not affect the relative difference between the baseline and our method.

Evaluation setup.
In order to measure the quality of the clusters, we construct a crowdsourced ground truth. We begin by randomly selecting 100 task-frame pairs (each task-frame in a pair belongs to the same cluster), and 100 task-frame pairs (each task-frame pair belonging to a different cluster). For each of these 200 pairs, CrowdFlower workers assess if a task-frame pair is similar enough to be clustered together. We display a maximum of top-k (k=5) values (sorted by frequency) for every frame attribute, in order to provide sufficient context to the turkers. This ground truth enables us to compare the accuracy (i.e. the ratio of the number of correctly clustered pairs against the total number of pairs) of the our KB organization method and the baseline. We also compare their running time.

Results.
Our clustering algorithm outperforms top-down clustering both in terms of speed and precision. Figure 2 shows that this gap keeps increases with the size of the dataset. The gains in speed are due to the pre-clustering in the bottom-up phase, leading to several, smaller clusters as starting point. The hybrid approach also stymies the baseline in terms of precision: 91.2% vs. 71.6%.

5.3 Resulting HowToKB

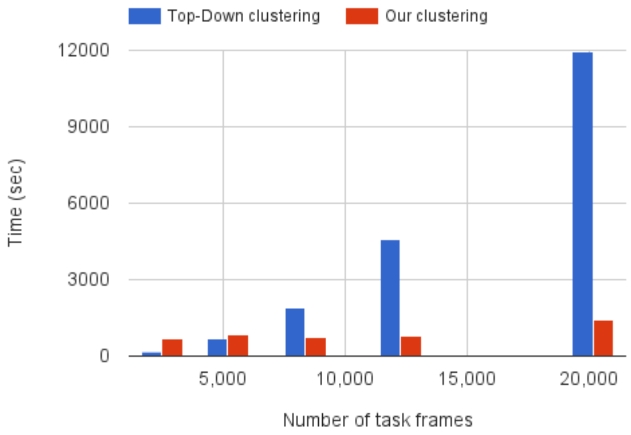

Figure 2: Running time of top-down vs. our clustering

ConceptNet relations	alignment
IsA, InheritsFrom	type of
Causes^{-1}, ReceivesAction^{-1}, RelatedTo^{-1}, CapableOf^{-1}, UsedFor^{-1}	participant
AtLocation, LocationOfAction^{-1}, LocatedNear	location
HasPrerequisite,	next
HasSubevent, Has First|Last Subevent	sub task
MotivatedByGoal	prev
SimilarTo, Synonym	similar

Table 5: ConceptNet as KB baseline

Baselines.
While there is no direct competitor that provides semantically organized task-frames, the knowledge bases with the most comparable content are Knowlywood [49] and ConceptNet [48]. They neither distinguish task knowledge nor provide task attribute structure, necessary for comparison with HowToKB. Even though the frame attributes in Knowlywood are same as in HowToKB, it does not contain **subtask** attribute; we treat that as an empty valued attribute for Knowlywood. ConceptNet represents knowledge as uncanonicalized triples, without any distinction between task, activity, and concepts: making it uncomparable to HowToKB. To provide structure to ConceptNet, similar to the approach in [49], we impose the constraints that the left argument of the ConceptNet triple must be syntactically similar to a task `verb [preposition] object`, and the right argument can be either a task or a noun concept. If necessary, we invert the relations in ConceptNet in order to map the relations to task-frame attributes (see Table 5).

Evaluation setup.
In contrast to the ground truth $G_{validate}$, that validated the frames, we crowdsource the task of acquiring perfect frames. This is a stricter evaluation of the final frames in HowToKB, and it is suitable because we are no longer evaluating the extraction from text, rather we are evaluating the overall quality of the task-frames. We randomly select 50 task-frames from $G_{validate}$ and asked turkers to fill the values in the attribute, instead of asking them to validate. This will likely provide another perspective where the turkers are not biased by our proposals. A sample question we posed is as follows: In some context such as *decorate the house* , the

most likely `location` when we *paint a wall* is *(blank)* . The resulting ground truth is called $G_{acquire}$, which is used to measure the precision of the HowTo task-frames (i.e. coverage of the crowdsourced values).

Results.

The evaluation results are shown in Table 6. We report the statistical significance by Wilson score confidence intervals for $\alpha = 95\%$ [6]. The results show that the clustered task-frames in HowToKB have very high quality.

The activities derived from ConceptNet and Knowlywood were assessed based on the ground truth. ConceptNet has low coverage containing only 1 of the 50 ground-truth tasks, and Knowlywood has 4 activities that match any of the 50 tasks. These numbers confirm that the kind of how-to task knowledge that HowToKB possesses is not captured by any prior work on commonsense knowledge acquisition.

Attribute	Precision	Size
Meaningful task phrases	0.94±0.04	0.51M
Location	0.70±0.27	18K
Participating object	0.89±0.04	1.4M
Participating living being	0.88±0.11	136K
Parent task	0.81±0.0	0.92M
Previous task	0.83±0.05	0.91M
Next task	0.84±0.04	0.91M
Sub task	0.88±0.03	0.92M
Images	0.88±0.03	1.03M

Table 6: Quality and Size of HowToKB

6. USE CASE: YOUTUBE VIDEO SEARCH

In order to investigate the usefulness of HowToKB, we present a use case to automatically retrieve relevant YouTube videos D for a query q, where q is a task title, e.g., *paint a wall* . The usefulness of HowToKB would be evident if the search results improve by expanding q to q' using relevant context from task frames (such as location attributes like `house`, or sub-tasks). If q is absent in HowToKB, this context can come from task frames which are semantically related to q (e.g., expansion using the context of *color a wall* in the absence of *paint a wall*). It should be noted that in this task the frames are simplified into bags of words, and the frame attribute labels are not leveraged. We perform this relaxation due to the absence of a baseline retrieval approach that could leverage structure.

Baselines.

As the task of query expansion has been well-studied, we propose several baselines that capture different aspects of relatedness, taxonomic, distributional, and structured frame attributes;.

noexpand As an obvious baseline, *noexpand* does not expand q and uses the terms in q to match against a YouTube video's title and description. This is already a strong baseline because some instructional videos tend to have informative titles, and descriptions.

wn As a second baseline, we leverage WordNet for query expansion, as it is a popular choice for expansion [36]. To expand q that consists of a verb and noun, we consider the Cartesian product of taxonomically similar verbs and taxonomically similar nouns. To control the size of expanded queries, we limit the expansion to the top-2 taxonomic neighbors. We follow the `similar-to` edges, and `hypernymy` edges between WordNet synsets. To convert synsets to words, we consider the two most frequent senses that should adequately capture the most important senses [50]. For each synset, we pick the more commonly occurring lexical word (in an external corpus such as Google N-grams [5]). For example, for the animal sense of `tiger`, we select the synset word `tiger`, instead of the less frequent `Panthera tigris`. This provides us with the top-k neighbors for every word that is used to expand the query.

w2v As a third baseline, we use a Word2Vec model 4.1, trained over WikiHow contents, to expand queries. For each query, we expand every word in the query by adding the two most related words according to the Word2Vec model.

Our system expands q, systematically using two sets of attributes from the task frames:

I_{attr} Inter-task attributes, consisting of parent task, and previous or next tasks.

C_{attr} Contextual attributes, consisting of location, time and participant information.

Our full system, called *ht* (HowTo), is the combination of these two.

Evaluation Setup.

As the use-case of automatically aligning WikiHow tasks with YouTube has not been studied previously, there is no ground truth that ranks $d \in D$ for q. Such a ground truth is challenging, e.g. how to judiciously select the query set Q, and crowdsourcing the relevance judgment annotation for even 100 queries would run into thousands of dollars. To overcome these challenges, we instead leverage the YouTube references on a few WikiHow pages (these WikiHow article titles form our task query set Q). We crawl the metadata associated with these YouTube videos such as, title, description, tags, category, and, comments. The resulting dataset, referred to as $G_{howtube}$, consists of 18,380 q/d pairs, typically one relevant video per query (1.02 videos per q).

A useful retrieval method would find a ground truth relevant video for the query in the top results. Precision@K is equivalent to HITS@K in this setup because there is typically one relevant video per query. MRR (Mean Reciprocal Rank) is another relevant metric that measures the position of the first correct result. We consider both HITS and MRR as evaluation metrics.

Results.

Table 7 reports HITS and MRR metrics for all the baselines, and the full HowToKB based system $(I + C)_{attr}$, referred as `ht`. It is evident that the expansion of q to q' provides very rich context leading to higher values for the evaluation metrics. YouTube videos describe the tools or objects needed for the task, or present the related tasks by providing a step-by-step procedure. HowToKB based expansion gains advantage of these descriptions, because the C_{attr} captures these relationships. See Table 8 for examples of successful queries.

Metric	`noexpand`	`wn`	`w2v`	C_{attr}	I_{attr}	`ht`
HIT@1	1421	2087	1863	1867	2199	**2361**
HIT@3	2411	2952	2946	3061	3408	**3619**
HIT@5	3058	3496	3532	3779	4077	**4340**
HIT@10	4259	4278	4614	5036	5043	**5425**
MRR	0.076	0.114	0.101	0.101	0.12	**0.128**

Table 7: Results for YouTube search use case

q	`ht`	YouTube video description
make caramel corn	make caramel corn ... brown sugar ... popcorn ... syrup teaspoon.. salt teaspoon ...bake soda ... vanilla ...	**Title: Gourmet Caramel Popcorn "Thanks Monique"** gourmet caramel popcorn ...ingredient ...popcorn tablespoon butter cup pack brown sugar cup ...light karo syrup teaspoon ...vanilla teaspoon bake soda ...
dance salsa	dance salsa ... salsa master basics dance ...leader dance follower work ...cross body lead ... learn open break	**Title: Salsa Dancing for Beginners** salsa dancing... beginner ... basics step...dancing partner ... salsa right turn ...cross body lead ... turn partner work...

Table 8: Examples of query expansions and corresponding search results

`wn` and `w2v` perform marginally less than `ht`. To further understand the differences among their results, we study:

1 Are these baselines complementary?
2 Which of these is the best system for ambiguous queries such as *use keyboard* .

To answer the first question, we find that `w2v` and `wn` are not complementary, with approximately 3/4th of the times overlapping. `w2v` relatedness results are mostly similar to `wn`, because both encode some form of relatedness and similarity. In the remaining 1/4th cases, `wn` and `w2v` have different information, `w2v` has co-occurrence relatedness (e.g. paint and brush), while `wn` does not contain such information. However, it contains cleaner similarity relations.

On the other hand, `ht` helps with a different genre of query expansion (in more than 50% cases), containing knowledge beyond semantic relatedness, such as previous task. We also find that in the non-trivial queries (where `noexpand` fails), `ht` gets 3% higher HIT@1. This clearly establishes `ht` as the best method.

To answer the second question, we find that for ambiguous queries, `ht` elicits the different senses much better than any other baseline. Out of 161 such ambiguous queries (with >=4 WikiHow categories), `ht` has a hit rate of nearly 10%, while the other baselines do not reach even 1% hit rate.

7. CONCLUSION

We presented HowToKB, the first comprehensive knowledge base with fine-grained attributes about HowTo tasks. Our methodology combines information extraction from how-to communities with a judiciously designed form of hierarchical clustering. Experiments demonstrate that this methodology is fast and yields accurate results. HowToKB and its crowdsourced test benchmarks are publicly available as a resource for other researchers.

As for ongoing and future work, we plan to add value to HowToKB by integrating more multimodal data (e.g., videos) and turn this resource into an open-domain training dataset for task-oriented computer vision tasks.

8. REFERENCES

[1] C. F. Baker, C. J. Fillmore, and J. B. Lowe. The berkeley framenet project. In *COLING-ACL*, 1998.

[2] N. Balasubramanian, S. Soderland, Mausam, and O. Etzioni. Generating coherent event schemas at scale. In *EMNLP*, 2013.

[3] J. Berant, V. Srikumar, P.-C. Chen, A. V. Linden, B. Harding, B. Huang, P. Clark, and C. D. Manning. Modeling biological processes for reading comprehension. In *EMNLP*, 2014.

[4] C. D. Bovi, L. Telesca, and R. Navigli. Large-Scale Information Extraction from Textual Definitions through Deep Syntactic and Semantic Analysis. *TACL*, pages 529–543, 2015.

[5] T. Brants and A. Franz. Web 1t 5-gram v1. 2006.

[6] L. D. Brown, T. T. Cai, and A. DasGupta. Interval estimation for a binomial proportion. *Statistical Science*, 16(2):101–133, 2001.

[7] N. Chambers. Event schema induction with a probabilistic entity-driven model. In *EMNLP*, 2013.

[8] N. Chambers and D. Jurafsky. Unsupervised learning of narrative event chains. In *ACL*, 2008.

[9] N. Chinchor. Muc-4 evaluation metrics. In *MUC*, 1992.

[10] D. Dahlmeier and H. T. Ng. Domain adaptation for semantic role labeling in the biomedical domain. *Bioinformatics*, 26:1098–1104, 2010.

[11] D. Das, D. Chen, A. F. T. Martins, N. Schneider, and N. A. Smith. Frame-semantic parsing. *Computational Linguistics*, 40:9–56, 2014.

[12] O. Etzioni, A. Fader, J. Christensen, S. Soderland, and M. Mausam. Open information extraction: The second generation. In *IJCAI*, pages 3–10, 2011.

[13] C. Fellbaum and G. Miller. *WordNet: An Electronic Lexical Database.* MIT Press, 1998.

[14] R. E. Fikes and N. J. Nilsson. Strips: A new approach to the application of theorem proving to problem solving. *Artificial intelligence*, 2(3-4):189–208, 1971.

[15] C. J. Fillmore. Frame semantics and the nature of language. *Annals of the New York Academy of Sciences*, 280(1):20–32, 1976.

[16] L. Frermann, I. Titov, and M. Pinkal. A hierarchical bayesian model for unsupervised induction of script knowledge. In *EACL*, 2014.

[17] N. Garg and J. Henderson. A bayesian model of multilingual unsupervised semantic role induction. *CoRR*, abs/1603.01514, 2016.

[18] D. Gildea and D. Jurafsky. Automatic labeling of semantic roles. *Computational Linguistics*, 2000.

[19] R. Girju and D. I. Moldovan. Text mining for causal relations. In *FLAIRS Conference*, 2002.

[20] A. Grycner and G. Weikum. Poly: Mining relational paraphrases from multilingual sentences. In *EMNLP*, 2016.

[21] L. He, M. Lewis, and L. S. Zettlemoyer. Question-answer driven semantic role labeling: Using natural language to annotate natural language. In *EMNLP*, 2015.

[22] T.-H. Huang, F. Ferraro, N. Mostafazadeh, I. Misra, A. Agrawal, J. Devlin, R. B. Girshick, X. He, P. Kohli, D. Batra, C. L. Zitnick, D. Parikh, L. Vanderwende, M. Galley, and M. Mitchell. Visual storytelling. In *HLT-NAACL*, 2016.

[23] R. Johansson and P. Nugues. Lth: Semantic structure extraction using nonprojective dependency trees. In *SemEval@ACL*, 2007.

[24] C. Kiddon, G. T. Ponnuraj, L. S. Zettlemoyer, and Y. Choi. Mise en place: Unsupervised interpretation of instructional recipes. In *EMNLP*, 2015.

[25] P. Kingsbury and M. Palmer. Propbank: the next level of treebank. In *Proceedings of Treebanks and lexical Theories*, volume 3. Citeseer, 2003.

[26] J. Lang and M. Lapata. Unsupervised semantic role induction with graph partitioning. In *EMNLP*, 2011.

[27] M. Lewis, L. He, and L. S. Zettlemoyer. Joint a* ccg parsing and semantic role labelling. In *EMNLP*, 2015.

[28] D. Lin and P. Pantel. Dirt@ sbt@ discovery of inference rules from text. In *KDD 2001.*

[29] S. Louvan, C. Naik, S. Kumaravel, H. Kwon, N. Balasubramanian, and P. Clark. Cross sentence inference for process knowledge. In *EMNLP*, 2016.

[30] Mausam, M. Schmitz, S. Soderland, R. Bart, and O. Etzioni. Open language learning for information extraction. In *EMNLP-CoNLL*, 2012.

[31] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean. Distributed representations of words and phrases and their compositionality. In *NIPS*, pages 3111–3119, 2013.

[32] M. Minsky. A framework for representing knowledge. In *MIT-AI Laboratory Memo 306*, 1974.

[33] F. Murtagh and P. Contreras. Algorithms for hierarchical clustering: an overview. *Wiley Interdisc. Rew.: Data Mining and Knowledge Discovery*, 2:86–97, 2012.

[34] N. Nakashole, G. Weikum, and F. Suchanek. Patty: a taxonomy of relational patterns with semantic types. In *EMNLP*, pages 1135–1145, 2012.

[35] J. W. Orr, P. Tadepalli, J. R. Doppa, X. Z. Fern, and T. G. Dietterich. Learning scripts as hidden markov models. In *AAAI*, 2014.

[36] D. Pal, M. Mitra, and K. Datta. Improving query expansion using wordnet. *Journal of the Association for Information Science and Technology*, 65(12):2469–2478, 2014.

[37] M. Palmer, I. Titov, and S. Wu. Semantic role labeling. In *Tutorials, NAACL-HLT*, 2013.

[38] M. Pennacchiotti, D. D. Cao, R. Basili, D. Croce, and M. Roth. Automatic induction of framenet lexical units. In *EMNLP*, 2008.

[39] K. Pichotta and R. J. Mooney. Statistical script learning with recurrent neural networks. *EMNLP*, page 11, 2016.

[40] M. Regneri, A. Koller, and M. Pinkal. Learning script knowledge with web experiments. In *ACL*, pages 979–988, 2010.

[41] M. Roth and M. Lapata. Neural semantic role labeling with dependency path embeddings. *CoRR*, abs/1605.07515, 2016.

[42] R. Rudinger, P. Rastogi, F. Ferraro, and B. V. Durme. Script induction as language modeling. In *EMNLP 2015.*

[43] R. C. Schank and R. P. Abelson. Scripts, plans and knowledge. In *IJCAI*, 1975.

[44] K. K. Schuler. Verbnet: A broad-coverage, comprehensive verb lexicon. *Doctoral Dissertation, University of Pennsylvania*, 2005.

[45] G. Schwarz. Estimating the dimension of a model. *The Annals of Statistics 6(2)*, pages 461–464, 1978.

[46] R. Sharp, M. Surdeanu, P. Jansen, P. Clark, and M. Hammond. Creating causal embeddings for question answering with minimal supervision. In *EMNLP*, 2016.

[47] R. Sibson. Slink: an optimally efficient algorithm for the single-link cluster method. *The computer journal*, 16(1):30–34, 1973.

[48] R. Speer and C. Havasi. Representing general relational knowledge in conceptnet 5. In *LREC*, pages 3679–3686, 2012.

[49] N. Tandon, G. de Melo, A. De, and G. Weikum. Knowlywood: Mining activity knowledge from hollywood narratives. In *CIKM*, pages 223–232, 2015.

[50] N. Tandon, G. de Melo, and G. Weikum. Acquiring comparative commonsense knowledge from the web. In *AAAI*, 2014.

[51] S. ting Yi, E. Loper, and M. Palmer. Can semantic roles generalize across genres? In *HLT-NAACL*, 2007.

[52] I. Titov and A. Klementiev. A bayesian approach to unsupervised semantic role induction. In *EACL*, 2012.

[53] K. Toutanova, A. Haghighi, and C. Manning. Joint learning improves semantic role labeling. In *ACL 2005.*

[54] L. D. A. Wanzare, A. Zarcone, S. Thater, and M. Pinkal. A crowdsourced database of event sequence descriptions for the acquisition of high-quality script knowledge. In *LREC*, 2016.

[55] Z. Wu and M. Palmer. Verb semantics and lexical selection. In *ACL*, pages 133–138, 1994.

[56] Z. Yang and E. Nyberg. Leveraging procedural knowledge for task-oriented search. In *SIGIR 2015.*

[57] P. Young, A. Lai, M. Hodosh, and J. Hockenmaier. From image descriptions to visual denotations: New similarity metrics for semantic inference over event descriptions. *TACL*, 2:67–78, 2014.