Hierarchy-aware Learning of Sequential Tool Usage via Semi-automatically Constructed Taxonomies

Nima Nabizadeh
Institute of Communication Acoustics, Ruhr University Bochum, Germany
nima.nabizadeh @rub.de

Martin Heckmann
Aalen University of Applied Sciences, Honda Research Institute Europe GmbH, Germany
martin.heckmann @hs-aalen.de

Dorothea Kolossa
Institute of Communication Acoustics, Ruhr University Bochum, Germany
dorothea.kolossa @rub.de

Abstract

When repairing a device, humans employ a series of tools that corresponds to the arrangement of the device components. Such sequences of tool usage can be learned from repair manuals, so that at each step, having observed the previously applied tools, a sequential model can predict the next required tool. In this paper, we improve the tool prediction performance of such methods by additionally taking the hierarchical relationships among the tools into account. To this aim, we build a taxonomy of tools with hyponymy and hypernymy relations from the data by decomposing all multi-word expressions of tool names. We then develop a sequential model that performs a binary prediction for each node in the taxonomy. The evaluation of the method on a dataset of repair manuals shows that encoding the tools with the constructed taxonomy and using a top-down beam search for decoding increases the prediction accuracy and yields an interpretable taxonomy as a potentially valuable byproduct.

1 Introduction

Humans perform various tasks that have an inherent sequential nature comprising several steps; repairing a device is one of them. An AI agent serving as a cooperative assistant in such a task should be provided with contextual knowledge about the pertinent sequence of steps. The importance of such knowledge in cooperative situations has been shown in (Salas et al., 1995; Marwell and Schmitt, 2013). An example of using sequential context knowledge in a cooperative situation is found in Whitney et al. (2016). Here, it can be seen that learning the dependencies among the ingredients in cooking recipes helps with resolving the user's request for ingredients, since the system can anticipate what may be needed next.

There have been numerous efforts to acquire task knowledge from available sources of instructional data, for instance from the web. Related work on extracting the workflow from instructional text, such as (Maeta et al., 2015; Yamakata et al., 2016), have not built a sequential model for generalizing the obtained knowledge to unseen tasks. Working on data collected from the wikiHow website as the resource, Chu et al. (2017) and Zhou et al. (2019) developed models that learn the temporal order of steps but only take one previous step into account and ignore the higher-order dependencies.

Nabizadeh et al. (2020a) compared sequence learning methods for modeling long-term dependencies among the used tools in various steps of repair tasks, showing the advantage of Recurrent Neural Network (RNN) models for this purpose. Their results revealed that the similarity among the sequence of used tools in different repair manuals makes it possible to predict the next required tool on unseen repair tasks. In their approach, each tool is represented as a distinct class, while the relationships among the related types of the tools are not considered. As a result, the input does not provide the model with any information about different types associated with a class of tool. For instance, the model has no clue that different types of screwdrivers, such as Phillips and Torx, are all screwdrivers, and that different sizes of Phillips screwdrivers belong to the same category of Phillips screwdrivers. Such information is also missing in calculating the cross-entropy loss of the RNN. However, we posit that for instance the penalty for predicting a 4mm Nut Driver instead of a 5mm Nut Driver should be less than the penalty of predicting a hammer, instead.

This paper, therefore, extends their work by taking such missing information into account. Specifically, we develop a sequential model for predicting the tool usage in unseen repair tasks, where we encode the tools using a semi-automatically constructed taxonomy. Previous work has shown the advantage of hierarchy-aware neural networks for different tasks, such as audio event detection (Jati et al., 2019) and entity type classification (Xu and Barbosa, 2018), which both benefit from predefined hierarchies.

The tool names in repair tasks are often compound nouns, containing information about the main class of the tool and its detailed attributes. We use the arrangement of words in the tool names for building a tool taxonomy, with different types and sizes of a parent node tool arrayed as its child nodes. Applying a binary classifier for predicting each node in the taxonomy, i.e., predicting the main class of the tool and its details separately, we show the advantages of hierarchy-aware prediction model over the flat one.

2 Method

This section introduces the proposed approach for producing the tool taxonomy from data and modeling the dependencies among the used tools in the steps of the repair tasks.

2.1 Semi-automatic Construction of Tool Taxonomy

The head-modifier principle (Spärck Jones, 1983) inspired our approach for constructing a taxonomy, stating that the linear arrangement of the elements in a compound reveals the kind of information it conveys. The head of a compound, which is usually the right-most word for compound nouns in the English language, serves as the general (semantic) category to which the entire compound belongs. Other elements, i.e., modifier, distinguish this member from other members of the same category. The automatic process of constructing the taxonomy contains two main stages: 1- branching and 2- merging. In the branching stage, we split the head and the modifier of the multi-word tool names and arrange the modifiers as the child nodes of the node corresponding to the head of the compound noun. In the merging step, a constructed node with only one child is merged with its child node, making a single node for both. E.g., the node "1.5mm Hex" in Figure 1, is produced by merging the parent node ”Hex” and its child node ”1.5mm”. The inherent structure of the tool names in the repair tasks allows us to perform the above process for more than one level. The multi-word tool names usually follow the pattern <size, type, main class>, as is the case, for example, in ”t6 Torx screwdriver.” However, we still needed to apply several handcrafted rules, e.g. via regular expressions, to standardize the tool names that were entered with a different pattern. For instance, ”Phillips #00 screwdriver” was changed to the equivalent and normalized name ”Ph00 Phillips screwdriver” to follow a unified pattern of left-branching compounds. It is worth mentioning that on the iFixit website, the instructors usually link the tools to the iFixit tool store; therefore, in most cases, different manuals use a unique name for a specific tool. In the process of constructing the taxonomy, the single-word tool names, and the heads of the multi-word tool names are grouped under the root node with the compound modifiers as the child nodes. This process is repeated for creating a taxonomy with up to three levels. Figure 1 shows the instances of the produced taxonomy, where the leaves, i.e., terminal nodes, are marked in gray. The constructed taxonomy is a non-ultrametric tree, i.e., the distance between the leaves and the root node is not the same for all leaves.

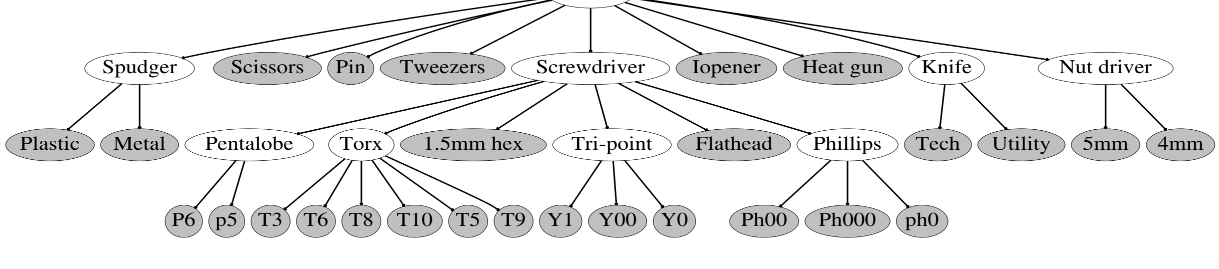

Figure 1: Instances of the tool taxonomy constructed from the MyFixit dataset

2.2 Sequential Models of Tool Usage

A repair manual can be understood as a list of steps; each step might require a different tool. Moreover, each tool is a node in the constructed taxonomy, with one or more parent nodes representing the more general categories of the tool. Let $\mathcal{O}$ denote the set of all nodes in the taxonomy except for the root node. The model is trained to predict the probability of observing each node from $\mathcal{O}$ in the following step, based on the sequence of prior, observed tools, i.e., based on the sequence of the taxonomy nodes seen in the preceding steps. We represent each node in the set $\mathcal{O}$ with a one-hot vector. A tool is then encoded by the sum of its active node vectors. The resulting multi-hot vector is later used as the input of the sequential model at each timestep. Our sequential model consists of a Long-Short-Term Memory (LSTM) (Hochreiter and Schmidhuber, 1997) layer with state size 256, followed by a Fully Connected (FC) hidden layer of the same size. The LSTM layer takes the encoded tools from the input and generates a representation of the used tools at each step. The LSTM output is fed to a fully-connected hidden layer with a hyperbolic tangent activation function while its parameters are shared among all the timesteps. The output layer has $|\mathcal{O}|$ neurons and a sigmoid activation function that estimates the probability of observing each node in $\mathcal{O}$. The model takes the multi-hot vector of the next tool as the ground truth during training. Its parameters are learned by minimizing the binary cross-entropy loss in Equation (1) using the Adam optimizer (Kingma and Ba, 2015) with a learning rate of 0.001.

$$\mathrm{H}(y, \hat{y}) = -\frac{1}{|\mathcal{O}|} \sum_{i \in \mathcal{O}} y_i \cdot \log(\hat{y}_i) + (1 - y_i) \cdot \log(1 - \hat{y}_i) \tag{1}$$

Here, $\hat{y}_i$ denotes the probability of i-th node in the model output and y_i is the corresponding target value. Figure 2 illustrates the unrolled graph of the proposed sequential model.

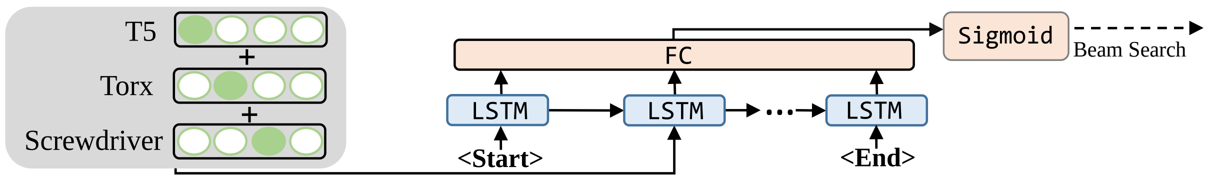

Figure 2: Proposed sequential model for learning the sequence of tools in repair tasks with an example of an encoded tool input.

2.3 Inference

To infer the required tool from the model's predicted distribution of taxonomy nodes, we used beam search, a search algorithm that generates the sequence of nodes one-by-one while keeping a fixed number of active candidates, the *beam size*, denoted by m. For each example in the test set, starting from the root node, we take m child nodes of the root node with the highest probability scores. For each node candidate, we expand it if it is not a leaf node and take its m child nodes with the highest probability. This process continues until we have expanded all the non-leaf-node candidates. Finally, the tool with the highest probability is returned as the prediction for the next step. The probability associated with a tool prediction is the average of the probabilities of its corresponding nodes in the taxonomy.

3 Experiment

3.1 Dataset Description

MyFixit is a collection of repair manuals collected by Nabizadeh et al. (2020b) from the iFixit website. The manuals are divided into several steps by the instructors, where at each step, the user should typically detach a specific component of the device under repair. Each step of the manuals in the "Mac Laptop" category is manually annotated with the required tools of the steps. In total, 1,497 manuals with 36,973 steps are annotated with the required tools. The authors also proposed an unsupervised method for the automatic annotation of tools from each step description. Their method utilizes the Jaccard similarity

Model	Annotation	Taxonomy Levels			
		1	2	3	Total
Flat	Manual	50.56 ± 2.0	83.69 ± 1.1	78.90 ± 1.3	76.30 ± 0.8 [1]
	Automatic	29.31 ± 1.6	72.08 ± 0.9	65.68 ± 1.9	62.39 ± 0.9
Hierarchy-aware	Manual	49.97 ± 1.0	86.29 ± 0.6	88.33 ± 0.5	81.78 ± 0.5
	Automatic	30.68 ± 0.9	72.87 ± 0.9	80.97 ± 1.0	70.42 ± 0.8

Table 1: Average accuracy of the tool prediction (%) with standard deviation, using the flat and hierarchy-aware model trained on automatically and manually annotated data.

between the bags of n-grams of the text description of steps and each tool name to return the tool with the highest similarity as the required tool of the step. The automatically annotated tools were reported to be correct in 94% of the steps. In addition to the sequential model trained on human-annotated data, we also evaluate the models trained on automatically annotated tools but tested on human annotations. This allows us to investigate the effect of hierarchy-aware prediction in the presence of annotation errors. Among the total steps of the annotated data, 51.8% of the used tools have three levels, 38.1% have two levels, and 10.1% have only one level in the constructed taxonomy.

3.2 Baseline Model

We compare the result of our proposed hierarchy-aware prediction to the baseline flat prediction of (Nabizadeh et al., 2020a). In this model, each tool is independently encoded with a one-hot vector, and the model is trained to reduce the cross-entropy loss between the predicted and ground-truth distribution.

3.3 Evaluation

To evaluate the proposed methodology in Section 2 we used ten folds of cross-validation; in each fold, we randomly split the data into 70% training, 20% test, and 10% development set. The development set is used for an early stopping mechanism. In our experiments, the best result is achieved with beam size 3. For the evaluation metric, we report the per-level leaves' accuracy, standard deviation, and total accuracy of the leaf nodes. The accuracy of each level is the number of correct predictions of leave nodes in a taxonomy level, divided by the total number of tools having leaf nodes at that level in the test set. Per-level leaves' accuracy can be calculated similarly for the flat predictor. The total accuracy is the count of all correct predictions divided by the size of the test set. Table 1 shows the result of our evaluation. It can be seen that the hierarchy-aware model improves the total accuracy by 5.48% for the manually annotated tools and 8.03% for the automatically extracted ones. The accuracy improvement achieved for predicting the tools with three levels in taxonomy is considerably higher than for the tools with a lower number of levels. Moreover, the hierarchy-aware model has a lower average standard deviation, and using this model helps the most for the prediction with automatically annotated data. This could be due to the fact that in the hierarchy-aware encoding of the tools, even if the annotation of the tool's detailed characteristics is wrong, the model can still be provided with correct information about the more general category of the tool. In 36.6% of the automatic annotation errors, the automatically and manually annotated tools have a common parent in the taxonomy.

4 Conclusion

In this paper, we utilize the head-modifier principle to decompose the multi-word expressions of tool names and build a taxonomy for the used tools in a dataset of repair manuals. We noted that utilizing the constructed taxonomy in sequential modeling of the used tools improves the tool prediction performance, especially when the data is annotated automatically and includes annotation errors. We imagine that hierarchy-aware modeling also helps when we have an imperfect observation of the used tools, e.g., when the model is uncertain about the size of used screwdrivers in the past. In the future, we plan to study the effect of such observation uncertainty on the prediction performance.

[1]Compared to (Nabizadeh et al., 2020a), we achieved a slightly higher accuracy for the flat predictor, due to the standardization of the tool names that led to a lower number of unique tools.

References

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

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9(8):1735–1780.

Arindam Jati, Naveen Kumar, Ruxin Chen, and Panayiotis Georgiou. 2019. Hierarchy-aware loss function on a tree structured label space for audio event detection. In *In Proceedings of IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 6–10. IEEE.

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *3rd International Conference on Learning Representations, ICLR*.

Hirokuni Maeta, Tetsuro Sasada, and Shinsuke Mori. 2015. A framework for procedural text understanding. In *Proceedings of the 14th International Conference on Parsing Technologies, IWPT*, pages 50–60. ACL.

Gerald Marwell and David R Schmitt. 2013. *Cooperation: An experimental analysis*. Academic Press.

Nima Nabizadeh, Martin Heckmann, and Dorothea Kolossa. 2020a. Target-aware prediction of tool usage in sequential repair tasks. In *Proceedings of The Sixth International Conference on Machine Learning, Optimization, and Data Science*, pages 869–880. Springer.

Nima Nabizadeh, Dorothea Kolossa, and Martin Heckmann. 2020b. Myfixit: An annotated dataset, annotation tool, and baseline methods for information extraction from repair manuals. In *Proceedings of Twelfth International Conference on Language Resources and Evaluation*, pages 2120–2128. European Language Resources Association.

Eduardo Salas, Carolyn Prince, David P Baker, and Lisa Shrestha. 1995. Situation awareness in team performance: Implications for measurement and training. *Human factors*, 37(1):123–136.

Karen Spärck Jones. 1983. Compound noun interpretation problems. Technical report, University of Cambridge, Computer Laboratory.

David Whitney, Miles Eldon, John Oberlin, and Stefanie Tellex. 2016. Interpreting multimodal referring expressions in real time. In *Proceedings of International Conference on Robotics and Automation (ICRA)*, pages 3331–3338. IEEE.

Peng Xu and Denilson Barbosa. 2018. Neural fine-grained entity type classification with hierarchy-aware loss. *arXiv preprint arXiv:1803.03378*.

Yoko Yamakata, Shinji Imahori, Hirokuni Maeta, and Shinsuke Mori. 2016. A method for extracting major workflow composed of ingredients, tools, and actions from cooking procedural text. In *Proceedings of International Conference on Multimedia & Expo Workshops (ICMEW)*, pages 1–6. IEEE.

Yilun Zhou, Julie Shah, and Steven Schockaert. 2019. Learning household task knowledge from wikihow descriptions. In *Proceedings of the 5th Workshop on Semantic Deep Learning*, pages 50–56. ACL.