Honda Research Institute EU
Honda Research Institute Europe GmbH
https://www.honda-ri.de/

Target-aware Prediction of Tool Usage in Sequential Repair Tasks

Nima Nabizadeh, Martin Ernst Heckmann, Dorothea Kolossa

2020

Preprint:

This is a post-peer-review, pre-copyedit version of an article published in Machine Learning, Optimization, and Data Science. The final authenticated version is available online at: https://doi.org/10.1007/978-3-030-64580-9

Target-aware Prediction of Tool Usage in Sequential Repair Tasks

Nima Nabizadeh[1], Martin Heckmann[2], and Dorothea Kolossa[1]

[1] Cognitive Signal Processing Group, Ruhr University Bochum, Germany
{nima.nabizadeh, dorothea.kolossa}@rub.de
[2] Honda Research Institute Europe GmbH, Germany
martin.heckmann@honda-ri.de

Abstract. Repairing a device is usually a sequential task comprising several steps, each potentially involving a different tool. Learning the sequential pattern of tool usage would be helpful for various assistance scenarios, e.g. allowing a contextualized assistant to predict the next required tool in an unseen task. In this work, we examine the potential of this idea. We employ two prominent classes of sequence learning methods for modeling the tool usage, including Variable Order Markov Models (VMMs) and Recurrent Neural Networks (RNNs). We then extend these methods by also conditioning them on additional information from the repair manuals that represents the repair target. This information is present, for example, in the title of manuals. We investigate the effect of long-term dependencies and of target-awareness on the prediction performance and compare the methods in terms of accuracy and training time. The evaluation using an annotated dataset of repair manuals shows that both target-awareness and long-term dependencies have a substantial effect on the tool prediction. While the RNN has slightly more accurate predictions in most scenarios, the VMM has a lower training time and is beneficial when the prediction needs to be restricted with respect to the device category.

Keywords: Target-awareness · Sequence learning · Repair tasks · Variable Order Markov Model · Recurrent Neural Network .

1 Introduction

An increasing demand exists for collaborative artificial agents that help humans to accomplish tasks and achieve common goals. Since performing many of these tasks requires interactions with objects in the environment, it is useful to equip the AI agent with knowledge about the possible object interactions. As an example, Whitney et al. [24] demonstrated that in a cooking task, modeling the sequence of mentioned ingredients in the recipes improves the performance of a robot when resolving the user's references to ingredients and helps in the disambiguation of the user's requests for such objects.

Typically, repairing a device is a complex sequential task, in which each step has to be completed before the next one can be started. In carrying out such repairs, a user works on a series of components employing a sequence of tools.

The order in which the tools are applied can be obtained from available repair manuals. Repair manuals for related device categories usually present similar patterns of tool use that correspond to the arrangement of components within the device. For example, most repairs of MacBook Pro devices include a first step of opening the lower case with a pentalobe screwdriver. The common next step is detaching the battery connections using a spudger, and then removing the logic board with a T6 Torx screwdriver. An AI agent capable of analyzing the repair manuals and modeling the tool usage could utilize this knowledge to predict the required tools during task execution, even if the exact ongoing task does not yet exist in the training data. For example, when a repair manual for replacing the hard drive on a 2017 Macbook Pro is not available, there may be manuals for other repair tasks on a Macbook Pro 2017, as well as manuals for hard drive replacement for other Macbook models. By leveraging this knowledge, an agent could disambiguate tool requests more easily and provide or suggest the needed tool to the user more quickly and proactively.

In this paper, we examine two well-known classes of sequence learning models for our purpose of predicting the required tools in unseen repair tasks: generative sequence modeling with a Variable Order Markov Model (VMM) and a discriminatively trained Recurrent Neural Network (RNN). We investigate a scenario, where the agent not only observes the sequence of used tools but is also informed about the task target. In this scenario, we assume that the user specifies the target, including information about the component under repair, e.g., "battery replacement," and the device category, e.g., "Macbook Pro Retina display." Based on the intuition that the training examples with more similar repair targets should provide more useful information for an accurate tool prediction, we propose methods for conditioning the predictor models on non-sequential information from repair manuals that represents the repair target.

The performance of the proposed methods is evaluated with a web-based dataset comprised of semi-structured repair manuals, in which the necessary tool for each repair step is annotated. We employ the titles of the repair manuals as a proxy to the full target of the repair tasks, containing the specific model of the device and the component that is to be repaired. Additionally, we utilize other information from the repair manuals, such as the hierarchy of the device categories, or the name of the component under repair. We are also concerned with the training time of the models because data available from the Internet can be updated frequently, and a model featuring a high training speed can leverage new instances of the source data faster.

2 Related Work

There have been many studies on obtaining and structuring knowledge from instructional data on the web, mainly in the domain of kitchen activities. The derived knowledge is sometimes common-sense knowledge about the individual objects that are involved in a task, e.g., "milk is in the refrigerator", or tool-action relations, e.g., "knives are used for cutting" [11, 18]. However, the

constructed knowledge-bases do not cover the temporal relation among the used objects in a task.

The temporal ordering of the task steps and their corresponding objects is taken into account in various studies on interpreting instructional texts and extracting a workflow from them [26, 16, 14, 25]. However, in these studies the generalization of learned sequences to previously unseen tasks is not considered. Such generalization is achieved in [1], where authors proposed a graphical model to determine the likely temporal order of events, i.e., actions together with the objects that they operate on. In [5], the authors extracted step-level semantic frames of the tasks in the Wikihow website, and temporally arranged them by clustering the frames' embeddings. In [27], Zhou et al. utilized the task titles in addition to the descriptions of the steps for predicting the order of steps. Still, none of the mentioned works accounts for long-term dependencies among the steps in a task.

RNNs and VMMs have been employed for sequence learning considering long-term dependencies in various applications, such as recommendation systems [9, 22, 3], human activity recognition [6, 17] or a wide range of natural language processing tasks, such as script learning [21]. Our work presents similar sequential models that are additionally conditioned on non-sequential information.

3 Models

In this section, we describe the proposed methods for modeling the dependencies among the used tools in various steps of the repair tasks along with the methods for conditioning the predictor models on the task target.

3.1 Problem Definition

The problem of tool prediction can be defined as follows: at each time step i, given a sequence of observed tools from k preceding steps $< t_{i-k}^{i-1} >$, we wish to predict the expected tool of the current step ($\hat{t}_i$) from a finite set of tools in the training set (T). At each time step, the models produce a probability distribution over T, and the required tool is obtained by

$$\hat{t}_i = \underset{t \in T}{\operatorname{argmax}} P(t_i = t | t_{i-k}^{i-1}), \tag{1}$$

The symbol in the first time step represents the start symbol and all sequences terminate in an end symbol, shared among all repair manuals.

3.2 Variable Order Markov Model

Variable order Markov models are an extension of first-order and fixed-order Markov models. In fixed-order Markov models, each random variable in a sequence depends on a specific number of preceding random variables. In VMMs, in contrast, the order of the model may vary in response to the statistics in the data. As a result of this, VMMs can be used to model sequential data of considerable complexity.

Begleiter et al. in [2] studied several prominent algorithms for the prediction of discrete sequences over a finite alphabet, using variable-order Markov models. Their results showed that in most cases the Prediction-by-Partial-Matching (PPM) algorithm performed best. In this paper, we use their implementation of the so-called method C (PPM-C)[3]. The value k in equation (1) can be interpreted as the upper bound for the maximal Markov order the model constructs and it limits the history to the k previous observations. When $k = 0$, it can be interpreted as a memoryless model which always predicts the most frequent tool in the training set, irrespective of any observation. For $0 < k \leq i$ the probability $P(t|t_{i-k}^{i-1})$ depends on the count of the sequence ($N(t_{i-k}^{i-1}, t)$) in the training set. If the count is zero, the model simply ignores the first observation in the sequence and recursively backs off to a lower order using an escape (E) probability:

$$P\left(t|t_{i-k}^{i-1}\right) = \begin{cases} P_k\left(t|t_{i-k}^{i-1}\right), & \text{if } < t_{i-k}^{i-1}, t > \text{ exists in training set} \\ P_k\left(E|t_{i-k}^{i-1}\right) P\left(t|t_{i-k+1}^{i-1}\right), & \text{otherwise.} \end{cases} \tag{2}$$

If Φ_k is the set of tools appearing after the history $< t_{i-k}^{i-1} >$ in the training set, the probability estimates based on the PPM-C method are:

$$\begin{aligned} P_k(t|t_{i-k}^{i-1}) &= \frac{N(t_{i-k}^{i-1}, t)}{|\Phi_k| + \sum_{t' \in \Phi_k} N\left(t_{i-k}^{i-1}, t'\right)} \quad , \text{ if } t \in \Phi_k \\ P_k(\mathrm{E}|t_{i-k}^{i-1}) &= \frac{|\Phi_k|}{|\Phi_k| + \sum_{t' \in \Phi_k} N\left(t_{i-k}^{i-1}, t'\right)} \end{aligned} \tag{3}$$

3.3 Recurrent Neural Network Model

Our RNN model consists of a Long-Short-Term Memory (LSTM) [10] layer, followed by a Fully Connected (FC) layer. The LSTM layer takes the tools encoded in 1-hot encoding from the input and transforms them into an embedding vector. The model estimates the probability of all possible tools in T, given the sequence of previous tools, by using a final soft-max layer that takes the 1-hot vector of the next tool as the ground truth during training. We use the cross-entropy loss

$$\mathrm{H}(p, q) = -\sum_{x \in T} p(x) \log\left(q(x)\right) \tag{4}$$

between the predicted p and target q probabilities of the tools using the Adam optimizer [13] with a learning rate of 0.001 over batches of 10 sequences. The LSTM layer has 256 units attached to a fully connected hidden layer of size 256. During training, we also use dropout with a rate of 0.2. The training process benefits from an early stopping mechanism that watches the accuracy of the development set in each epoch and ends the training if the accuracy is not increased in 10 consecutive epochs.

[3] We tried two other VMM algorithms using the SPMF open-source library, including compact prediction tree [8] and LZ78 [28], but they did not show a better performance than PPM and were therefore not considered in the later evaluation.

3.4 Integrating the Task Target in a VMM

For integrating the task target in a VMM, we additionally condition the probability of observing a tool on the auxiliary information that represents the repair target (G).

Assume that the observed target of the task during test time is a string of words $G = w_1, w_2, .., w_m$. The probability of using tool t in the current step after observing the history of used tools and the task target can then be estimated via

$$P\left(t|t_{i-k}^{i-1}, G\right) = \frac{P(t|t_{i-k}^{i-1})P(G|t_{i-k}^{i-1}, t)}{P(G|t_{i-k}^{i-1})} \propto P(t|t_{i-k}^{i-1})P(G|t_{i-k}^{i-1}, t). \tag{5}$$

The first term in Equation (5), $P(t|t_{i-k}^{i-1})$, is derived similarly to the model without target awareness, using Equation (3) independently of the task target. Similarly if $< t_{i-k}^{i-1}, t >$ does not exist in the training set, the model recursively ignores the earliest observation and repeats the process. The second term $P(G|t_{i-k}^{i-1}, t)$ is the probability of observing the target G given the history of tool usage and the tool candidate t. Let us assume that the sequence $< t_{i-k}^{i-1}, t >$ appears in one or more repair manuals in the training set. A set of all targets, e.g., titles of these manuals containing $< t_{i-k}^{i-1}, t >$, forms our reference for estimating $P(G|t_{i-k}^{i-1}, t)$. This set is denoted as D. We assume that the more similar G is to D, the higher the probability $P(G|t_{i-k}^{i-1}, t)$ should be. We use the Jaccard similarity between the sets of n-grams of various orders as the measure of text similarity between G and D. If G_n and D_n are the sets of all n-grams of order n in the strings of words in G and D respectively, the similarity score is: $\text{sim}(G_n, D_n) = |D_n \cap G_n|/|D_n \cup G_n|$. The final value is estimated by a weighted interpolation of the similarity scores of various n-gram orders:

$$P(G|t_{i-k}^{i-1}, t) \propto \epsilon + \sum_{n=1}^{m} \lambda(n)\text{sim}(D_n, G_n), \tag{6}$$

where m is the total number of words in G , ϵ is a small smoothing constant to avoid zero values in case there is no common n-gram between G and D, and $\lambda(n)$ is the interpolation weight when $\sum_{n=1}^{m} \lambda(n) = 1$. The interpolation weight makes it easy to investigate the effect of various n-gram orders on the result. We examine three different approaches for λ, a constant weight ($\lambda_c(n) = 1/m$), a linear function that assigns higher weights to the higher-order n-gram similarities ($\lambda_l(n) = n/\sum_{j=1}^{m} j$), and a softmax function that exponentially grows with the order of n-gram similarities ($\lambda_s(n) = e^n/\sum_{j=1}^{m} e^j$).

3.5 Integrating the Task Target in an RNN Model

For conditioning RNN models on auxiliary features, several approaches have been proposed in the literature. In the *contextual recurrent networks*, see [15, 7], a context vector is added to the input of the RNN cells at each time step. Unlike to our case, the context vector may change from one time step to the next. In contrast, in [23, 12], the hidden state of the RNN is affected by an

image embedder only at time zero for generating short image descriptions. For longer sequences, however, it may be difficult for the network to remember the auxiliary information until the end of the sequence. In our experiments, we achieved the best results when we included a representation of the repair target into the output of the LSTM hidden states in every time step, feeding into a fully connected layer. An advantage of this approach is that we do not need to re-train the LSTM weights, and the LSTM weights of the model in Section 3.3 can be used directly.

For generating a representation of the repair task, we employed another RNN with Gated Recurrent Unit (GRU) cells [4] of size 50, receiving the pretrained GloVe [20] word embeddings of the repair target description as input. The GRU cells receive the word embeddings and produce a representation of the repair target, which is concatenated to the representation of the sequence of the tools produced by the LSTM cells of the core tool prediction network. Fig. 1 shows the architecture of this model.

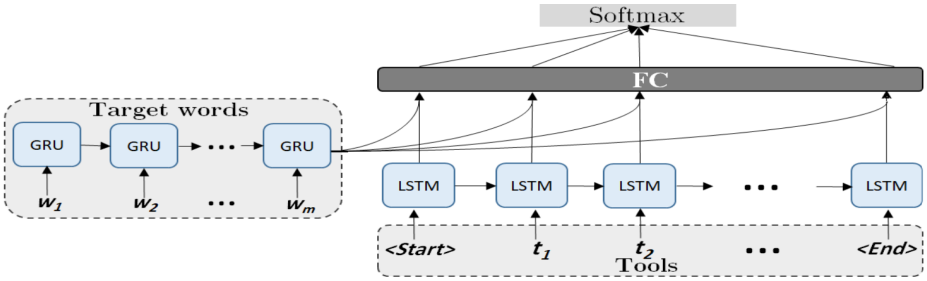

Fig. 1. Unrolled graph of the RNN model conditioned on the task target. The tools are encoded as 1-hot vectors and the target words are encoded with pre-trained GloVe embeddings.

4 Experiments

In this section, we describe the dataset we used for evaluating the performance of all considered sequence prediction models. First, we assess the models without an integrated target to analyze the effect of long-term dependencies, then we evaluate the performance of the new, target-aware predictors.

4.1 Dataset Description

We evaluate the methods of Section 3 on the human-annotated part of the MyFixit dataset[4]. MyFixit is a collection of repair manuals collected from the iFixit website [19]. The iFixit instructors split the manuals into their constituent steps, where at each step, the user typically works on one particular device component.

Since reassembly is often the reverse of disassembly, it is not included in the manuals of the iFixit website. Every step of manuals in the "Mac Laptop"

[4] https://github.com/rub-ksv/MyFixit-Dataset

category is annotated with the required tools of the step. In total, 1,497 manuals and 36,973 steps are annotated with 81 unique tools. Since our task is to predict the required tools, the steps that do not need a tool are not included in the manuals. On the other hand, more than one tool is used in 2.8% of the steps, which we split into multiple consecutive steps that only require one tool. Each repair manual has a title that usually contains detailed information about the device and the broken component, e.g., *"MacBook Pro 15" Retina Display Mid 2014 Battery Replacement"*. We interpret the title of manuals as the full target of the task. Additionally, every manual has a hierarchical list of device *categories* and a *subject* field that refers to the component that needs to be repaired or replaced. We treat the name of device categories or the broken component as a limited target of the task. Fig. 2 shows an instance of the device categories and components in MyFixit data. The name of the device categories in the data include the information from upper-level categories, for example, the complete name of *"Late 2013"* category in Fig. 2 is *"MacBook Pro 15" Retina Display Late 2013"* in the data.

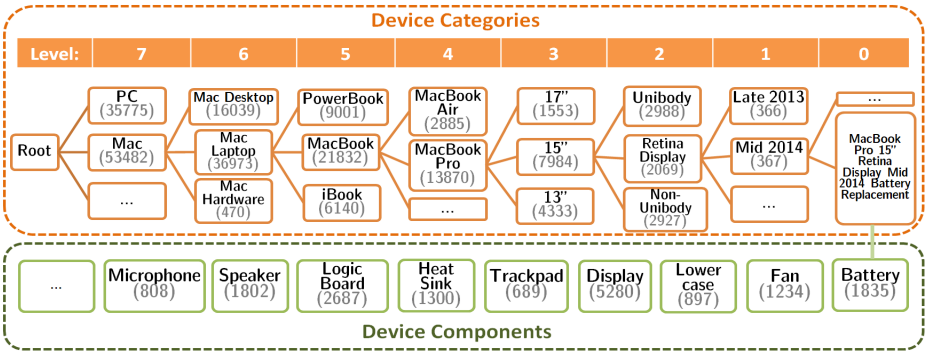

Fig. 2. Instances of the hierarchical categories of devices (above) and the component to be repaired (below) in the MyFixit dataset. The number in the parentheses shows the total number of steps within the device category or with same component to be repaired.

It can be seen in Fig. 2 that we identify each level of the device categories with a number, starting from zero on the leaf nodes that belong to the complete title of the tasks. As the category level increases, we have a more general information about the device being repaired. One should note that the number of hierarchy levels is not the same for all the manuals; for example, MacBook Air devices do not have the level two in this figure relating to the Unibody/Retina display, and the level two for those devices corresponds to the level three in the figure, indicating the size of the screen.

4.2 Prediction without Target-awareness

In this section, we evaluate the models of Section 3.2 and 3.3 for the prediction of expected tools based only on the observation of previously applied tools. We

use 10 folds of cross-validation; in each fold, we randomly split the data into 80% training and 20% test set. For the RNN-based model, we utilize 10% of the training set as a development set for hyper-parameter tuning and early stopping. In each cross-validation fold, the accuracy is calculated by counting the number of correct predictions divided by the total number of steps in the test set. By varying the maximum order of the VMM, i.e., k, we can scrutinize the effect of long-term dependencies on the prediction. When $k = \max$, the maximum order of the VMM is equal to the length of the longest sequence in the training set (61 on average). Training the RNN takes around 8 minutes on a single GTX TITAN X GPU and training the VMM on one core of a 2.10GHz CPU with 11 MB cache takes 0.01 to 1.3 seconds depending on the maximum model order (k). Fig. 3 shows the accuracy of the proposed models in percent.

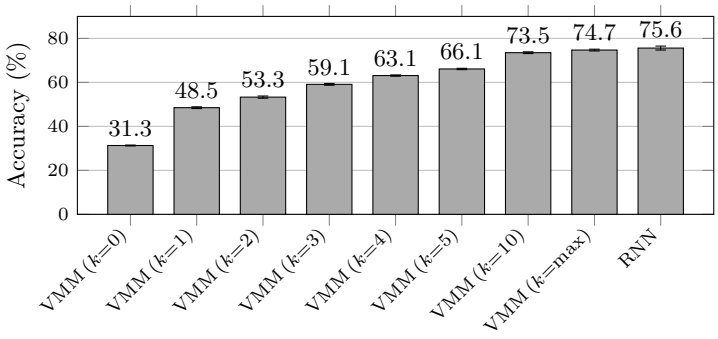

Fig. 3. Accuracy (%) of tool prediction for VMM with different maximal Markov order (k) and RNN. The error bars show the standard deviation between the results of the 10 folds of cross validation.

As can be seen in Fig. 3, both approaches perform comparably well, where the RNN has a slightly higher accuracy, which is also accompanied by a somewhat higher standard deviation between the results of the 10 folds. The accuracy of the VMM with $k = 0$ reveals the percentage of the most frequent tool (the spudger) in the dataset. In contrast to Whitney et al, [24], who showed that increasing the order of the model to more than two, i.e., using bi-grams, does not have a considerable impact on deriving the correct ingredient, our result demonstrates that incorporating the longer history can significantly increase the accuracy of tool prediction. However, the effect of expanding the history to include more than 10 previous observations only leads to marginal improvements. Using an RNN model instead of a VMM results in a further small improvement. Yet this comes at the cost of considerably longer training times for the RNN.

4.3 Target-aware Prediction

In the following, we assume that apart from the history of tool usage, the model has access to other pieces of information regarding the repair task. In one scenario, this information is the full task target, containing both the name of the

device and the broken component (level 0 in Fig. 2). In another scenario, we assume that the information is limited either to the device category at various levels of generalization or to the name of the component being replaced or repaired.

Full Target: The models with target-awareness proposed in Section 3.4 and 3.5 are evaluated similarly to the model without target-awareness (Section 4.2) using 10 folds of cross-validation. Training the RNN model took around 4 minutes, while training and testing the VMM (k=max) conditioned on the task target took around 95 seconds for the constant and linear interpolation weight and approximately two minutes for the exponential weight. The computational hardware is the same as the ones in the previous section. Nearly one-third of this time was spent on the training phase when we build the transition matrix, and for each entry, a set of n-grams in the title of the manual is created. In the test phase, the similarity scores are computed in an online fashion. Table. 1 shows the accuracy of the models.

Table 1. Accuracy (%) of the models without target-awareness, compared to the accuracy of the target-aware VMM with various interpolation weights (λ) of n-gram similarities, along with the RNN conditioned on the repair targets.

Target-awareness	Model	Accuracy	
		mean	std
Without	VMM	74.7	0.43
	RNN	75.6	0.88
With	VMM (Constant λ)	80.5	0.51
	VMM (Linear λ)	81.6	0.48
	VMM (Softmax λ)	83.4	0.36
	RNN	**86.3**	0.75

One can see that integrating target awareness improves the accuracy by 10.7% for RNN and 8.7% for VMM models. From Table 1 it can be seen that assigning larger weights to the higher n-gram similarities in VMM models yields a higher accuracy and lower standard deviation. The model based on an RNN gives the highest accuracy by producing a high-dimensional representation of the task target and the sequence of tools.

Limited Target: In this experiment, we investigate the cases with limited target knowledge, i.e., only knowing the general category or the broken component of the device. For the VMM, we compare two different strategies. In the first, similar to the models with full target-awareness, we condition the model's posterior probability on the sequence of words in the limited target, using Equation (5) and the softmax interpolation weights. Since the similarity scores are calculated in an online manner, the model does not need extra training. In the second

strategy, for each test instance, we train the VMM without an integrated target directly on the subset of the training data that shares the repair target. These subsets are denoted as "target sets". Because the training of the VMM, especially on small datasets, usually takes only a fraction of a second, this approach appears to be feasible in practice. If the target set of the category of a test manual is empty, i.e., there is no manual with the same category in the training set, the target set is constructed using a higher level category of the devices in the training set. For the components, we skip the target-awareness and use the complete training set if the target set becomes empty.

For the RNN model, we fine-tune the conditioned model (Section 3.5) on a training set with limited target information. We re-train the parameters of the GRU and fully connected layer on different variants of the training sets; in each one, the target of the manuals includes either the name of the component to be repaired or different levels of device categories. The models are evaluated on three levels of categories. The average numbers of manuals in the target set are 16.63, 72.59, and 149.99 for the category level 1, 2, and 3, respectively, and 22.16 for the device components. Table 2 represents the average accuracy of the suggested models for different levels of target-awareness. The result shows that

Table 2. Average accuracy (%) of proposed VMM and RNN with full target-awareness (category level 0) and limited target-awareness (category 1, 2, 3 and device components).

Model	Category Level				Component
	0	1	2	3	
VMM conditioned on target	83.4	82.6	80.4	78.1	76.2
VMM trained on target set	N/A	**83.3**	**81.1**	79.0	75.8
RNN conditioned on target	**86.3**	81.4	80.0	**79.3**	**77.5**

the VMM trained on target sets performs slightly better when there is more explicit information about the category of the device, i.e., the category level is lower and there are fewer samples in the target set. For the most general category (level 3) and the device components, the RNN performs best.

5 Conclusion

In this paper, we investigate the use of sequence models for predicting tool usage from repair manuals. We introduce approaches to additionally condition the models on the repair task.

Our results show that the prediction of tool usage can be improved notably by considering the long-term dependencies among the repair steps. VMMs and RNNs are comparable for this task, while VMMs exhibit a considerably shorter training time and do not need the effort for hyper-parameter tuning that is required with neural network models. We have also shown that our approaches for

including target awareness increase accuracy, and that this increase depends on the specificity of the provided repair target. Not surprisingly, the best performance is achieved when the target includes the exact model of the device and of the component that is to be replaced. For our targeted application domain, it could be beneficial to use both VMM and RNN models in parallel, as the VMM has the advantage that it can be trained on a smaller sub-set of the training set corresponding to the specified category of the device, while the RNN can offer performance benefits on larger data.

In this work, we have focused on task-conditional modeling of tool sequences, all the while assuming that the models can rely on a perfect observation of the ground-truth tool usage. In the future, we plan to explore how observation errors can influence the performance of each of the predictor models, with the goal of developing our methods towards real-life scenarios, where a repair assistant would only have imperfect data, e.g. from video recordings of the on-going repair process, on which to base its tool predictions and suggestions.

References

1. Abend, O., Cohen, S.B., Steedman, M.: Lexical event ordering with an edge-factored model. In: The 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies. pp. 1161–1171. ACL (2015)
2. Begleiter, R., El-Yaniv, R., Yona, G.: On prediction using variable order markov models. Journal of Artificial Intelligence Research **22**, 385–421 (2004)
3. Cai, R., Wang, Q., Wang, C., Liu, X.: Learning to structure long-term dependence for sequential recommendation. arXiv preprint arXiv:2001.11369 (2020)
4. Cho, K., van Merrienboer, B., Gülçehre, Ç., Bahdanau, D., Bougares, F., Schwenk, H., Bengio, Y.: Learning phrase representations using RNN encoder-decoder for statistical machine translation. In: Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing. pp. 1724–1734. ACL (2014)
5. Chu, C.X., Tandon, N., Weikum, G.: Distilling task knowledge from how-to communities. In: Proceedings of the 26th International Conference on World Wide Web. pp. 805–814. ACM (2017)
6. Ding, W., Liu, K., Cheng, F., Zhang, J.: Learning hierarchical spatio-temporal pattern for human activity prediction. Journal of Visual Communication and Image Representation **35**, 103–111 (2016)
7. Ghosh, S., Vinyals, O., Strope, B., Roy, S., Dean, T., Heck, L.P.: Contextual LSTM (CLSTM) models for large scale NLP tasks. CoRR (2016)
8. Gueniche, T., Fournier-Viger, P., Tseng, V.S.: Compact prediction tree: A lossless model for accurate sequence prediction. In: International Conference on Advanced Data Mining and Applications. pp. 177–188. Springer (2013)
9. He, Q., Jiang, D., Liao, Z., Hoi, S.C., Chang, K., Lim, E.P., Li, H.: Web query recommendation via sequential query prediction. In: 25th international conference on data engineering. pp. 1443–1454. IEEE (2009)
10. Hochreiter, S., Schmidhuber, J.: Long short-term memory. Neural computation **9**(8), 1735–1780 (1997)
11. Kaiser, P., Lewis, M., Petrick, R.P., Asfour, T., Steedman, M.: Extracting common sense knowledge from text for robot planning. In: International Conference on Robotics and Automation (ICRA). pp. 3749–3756. IEEE (2014)

12. Karpathy, A., Fei-Fei, L.: Deep visual-semantic alignments for generating image descriptions. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 3128–3137. IEEE (2015)
13. Kingma, D.P., Ba, J.: Adam: A method for stochastic optimization. In: 3rd International Conference on Learning Representations, ICLR (2015)
14. Maeta, H., Sasada, T., Mori, S.: A framework for procedural text understanding. In: Proceedings of the 14th International Conference on Parsing Technologies, IWPT. pp. 50–60. ACL (2015)
15. Mikolov, T., Zweig, G.: Context dependent recurrent neural network language model. In: Spoken Language Technology Workshop (SLT). pp. 234–239. IEEE (2012)
16. Mori, S., Maeta, H., Yamakata, Y., Sasada, T.: Flow graph corpus from recipe texts. In: Proceedings of the Ninth International Conference on Language Resources and Evaluation. pp. 2370–2377. ELRA (2014)
17. Murad, A., Pyun, J.: Deep recurrent neural networks for human activity recognition. Sensors **17**(11), 2556 (2017)
18. Nanba, H., Doi, Y., Tsujita, M., Takezawa, T., Sumiya, K.: Construction of a cooking ontology from cooking recipes and patents. In: Proceedings of the 2014 ACM International Joint Conference on Pervasive and Ubiquitous Computing: Adjunct Publication. pp. 507–516 (2014)
19. Nima Nabizadeh, Dorothea Kolossa, M.H.: 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 (2020)
20. Pennington, J., Socher, R., Manning, C.D.: Glove: Global vectors for word representation. In: Empirical Methods in Natural Language Processing. pp. 1532–1543. ACL (2014)
21. Pichotta, K., Mooney, R.J.: Learning statistical scripts with LSTM recurrent neural networks. In: Schuurmans, D., Wellman, M.P. (eds.) Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence. pp. 2800–2806. AAAI Press (2016)
22. Quadrana, M., Cremonesi, P., Jannach, D.: Sequence-aware recommender systems. ACM Computing Surveys (CSUR) **51**(4), 1–36 (2018)
23. Vinyals, O., Toshev, A., Bengio, S., Erhan, D.: Show and tell: A neural image caption generator. In: IEEE Conference on Computer Vision and Pattern Recognition, CVPR. pp. 3156–3164 (2015)
24. Whitney, D., Eldon, M., Oberlin, J., Tellex, S.: Interpreting multimodal referring expressions in real time. In: International Conference on Robotics and Automation (ICRA). pp. 3331–3338. IEEE (2016)
25. Yamakata, Y., Imahori, S., Maeta, H., Mori, S.: A method for extracting major workflow composed of ingredients, tools, and actions from cooking procedural text. In: International Conference on Multimedia & Expo Workshops (ICMEW). pp. 1–6. IEEE (2016)
26. Zhang, Z., Webster, P., Uren, V.S., Varga, A., Ciravegna, F.: Automatically extracting procedural knowledge from instructional texts using natural language processing. In: Proceedings of the Eighth International Conference on Language Resources and Evaluation. pp. 520–527. ELRA (2012)
27. Zhou, Y., Shah, J.A., Schockaert, S.: Learning household task knowledge from wikihow descriptions. arXiv preprint arXiv:1909.06414 (2019)
28. Ziv, J., Lempel, A.: Compression of individual sequences via variable-rate coding. IEEE transactions on Information Theory **24**(5), 530–536 (1978)