

# MyFixit: An Annotated Dataset, Annotation Tool, and Baseline Methods for Information Extraction from Repair Manuals

Nima Nabizadeh<sup>1</sup>, Dorothea Kolossa<sup>1</sup>, Martin Heckmann<sup>2</sup>

<sup>1</sup>Cognitive Signal Processing Group, Ruhr University Bochum, Germany,

<sup>2</sup>Honda Research Institute Europe GmbH, Germany

<sup>1</sup>{nima.nabizadeh, dorothea.kolossa}@rub.de

<sup>2</sup>martin.heckmann@honda-ri.de

## Abstract

Text instructions are among the most widely used media for learning and teaching. Hence, to create assistance systems that are capable of supporting humans autonomously in new tasks, it would be immensely productive, if machines were enabled to extract task knowledge from such text instructions. In this paper, we, therefore, focus on information extraction (IE) from the instructional text in repair manuals. This brings with it the multiple challenges of information extraction from the situated and technical language in relatively long and often complex instructions. To tackle these challenges, we introduce a semi-structured dataset of repair manuals. The dataset is annotated in a large category of devices, with information that we consider most valuable for an automated repair assistant, including the required tools and the disassembled parts at each step of the repair progress. We then propose methods that can serve as baselines for this IE task: an unsupervised method based on a bags-of- $n$ -grams similarity for extracting the needed tools in each repair step, and a deep-learning-based sequence labeling model for extracting the identity of disassembled parts. These baseline methods are integrated into a semi-automatic web-based annotator application that is also available along with the dataset.

**Keywords:** semi-structured dataset, information extraction, instructional text, procedural task, repair manual, semi-automatic annotation web tool

## 1. Introduction

Since the emergence of the earliest writing systems, textual instructions have always been among the most ubiquitous means of transferring procedural knowledge. Nowadays, free instructions are available on the Web for numerous tasks of human life, be they easy every-day hacks, recipes, or professional-level instructions on using or repairing technical devices of the highest complexity. For the latter, instructions might be written in a complex and technical style, and they, in general, often use situated language (Malmaud et al., 2014). The instructional text is regularly divided into multiple steps, which should be performed in a specified sequence. Consequently, there could be a sequence of objects that the person interacts with during the process. Extracting such practical information from text presents a unique challenge in the domain of information extraction (IE).

The type and structure of the extracted information also depend on the characteristics of the task and the intended use of the information. In this work, we focus on IE from a highly complex type of instructions: repair manuals. More specifically, we focus on extracting the pieces of information that a collaborative repair assistant would benefit most from, in order to support a human in a repair task.

A fruitful collaboration among agents depends on multiple prerequisites, one of them being sufficient shared knowledge among the agents. Many researchers have stressed the importance of shared knowledge and representation structure in cooperative situations (Grice, 1975; Salas et al., 1995). In our case, we are mainly interested in how this information builds up and is spread across the different steps as well as across the different sentences in a single step. More precisely, we focus on the sequence and identity of the required tools and disassembled parts during the repair

process. This information can help to estimate the state of the task and workstation environment, serving as the “situational context” of task-oriented dialogue, as it is defined in Deutsch (1974).

When a task-oriented collaboration involves objects in the shared workspace of the agents, an essential communicative function is resolving the partner’s references to objects in the environment. Wilkes-Gibbs and Clark (1992) indicated that when people collaborate on referring expressions, they issue full noun phrases initially, after which they begin to shorten the phrases. Thus, in the process of repair, one could say “Give me the T8 Torx screwdriver”, and then “Give me the screwdriver”, or just “next one”. In such situations, the task knowledge helps the partner to anticipate the required objects and disambiguate the references, so that the speaker can be less explicit in referring expressions (Whitney et al., 2016). Generally, when the agents share a common source of knowledge, such as the task instructions, it can be expected that their cooperation becomes smoother and more intuitive.

An assistance system can use the extracted information to obtain better estimates of a task’s progress. Compared to the recognition of user activities, or perception of the visual state of the device, observing the sequence of requested tools and detached parts on the workspace can provide a more accessible means of narrowing down the estimation of the current step. The human actions during the repair process may include subtle or sophisticated motions, sometimes on small and occluded components that would be more difficult to recognize in a non-contextualized setup.

In a robotic scenario, an intelligent repair assistant can utilize the extracted information to offer fast or proactive help to the human. It can hand and fetch the objects of each

repair step, practicing its knowledge to predict or disambiguate the correct object. The robot can make use of its knowledge of which part will be detached, or tool needed to prepare its gripper for, or providing a fitting container. For example, screws of different sizes would go in different containers. More generally, repair assistants (both virtual or embodied) could use such information to provide contextualized help and to estimate the task progress.

In this paper, we present a semi-structured dataset of repair manuals. To the best of our knowledge, this is the first dataset in the domain of repair. Information extraction from repair manuals brings several challenges to this task, as they usually consist of long and complex passages with a situated and technical language. Frequently, relevant parts of connected information are distributed across several passages. The dataset is enriched with human annotations in a category of devices, and the amount of annotations is considerably larger than the existing datasets (cf. Sec. 6. for a comparison with similar datasets). For addressing the proper information in the underspecified text, we propose two different annotation schemes for word- and step-level information. In the word-level annotation, we assume that a good description of the disassembled part exists in the text, while at the step-level, we deal with the imperfections in the natural language, such as omissions and implicit actions. In addition to the annotated data, we introduce an unsupervised method for extracting the sequence of required tools that uses the manual’s pre-specified set of tools, its *toolbox*, and device category in addition to the text description of the steps. Moreover, we test a state-of-the-art sequence labeling method for the extraction of the disassembled parts and the removal verbs; i.e., the verbs which describe the removal of a part from its location, and we report the performance of the model. The model has a bidirectional long short-term memory architecture with a subsequent conditional random field (BiLSTM-CRF) for labeling the word-level information. These baseline methods are integrated into a semi-automatic web-based annotator, which is also freely available in addition to the dataset<sup>1</sup>.

The paper is organized as follows: We introduce the dataset and annotation guidelines in Sec. 2. Then, we propose methods for automatically extracting the tool and part information of interest in Sec. 3. and we evaluate their performance in Sec. 4. In Sec. 5. we introduce the annotator app and Sec. 6. discusses related work. The paper is concluded by an analysis of the introduced task and future work in Sec. 7.

## 2. Dataset and Annotation Scheme

In this section, we introduce the dataset and our proposed annotation schemes, addressing the problems such as elided nouns and implicit actions of the steps. The data is collected from the iFixit<sup>2</sup> website. iFixit is a wiki-based site, in which users collaboratively generate and modify repair manuals for many everyday appliances, creating one of the largest collections of free online manuals.

<sup>1</sup><https://github.com/rub-ksv/MyFixit-Dataset>

<sup>2</sup><https://www.ifixit.com/>

## 2.1. Data Collection

We gathered the manuals from iFixit along with their metadata in JSON-like objects, where each document has self-explaining tags regarding the *Title*, *Category*, a list of all required tools provided by the instructor, which we refer to as the *Toolbox*, a list of *Steps*, and a derived list of hierarchical categories for the device (*Ancestors*). In most cases, the instructors show step-by-step procedures for opening the device and removing or repairing a broken component. Since reassembly is usually the reverse of disassembly, it is not included in the guides of iFixit. Each step has an attribute *Lines*, which contains the text description of the step and, if available, the attribute *Image* with the link(s) to the provided image(s). The fact that 98.7 % of the steps have one or more images also makes this data suitable for multimodal studies. Figure 1 shows an instance of the dataset, including the extra tags we added to the data that contain the annotated information in this work.

### 2.1.1. Data Statistics

In total, 31,601 repair manuals were collected from the iFixit API in 15 basic categories, see Figure 2. There is a high variation in the number of steps (average=9.68, median=7.00, variance = 109.95), depending on the category of the device and the difficulty level of the task. However, there is less variation in the number of tools pre-specified in the toolboxes, where the average number is 2.42 tools per manual with median is 2.00 and variance is 3.98.

## 2.2. Annotation Guidelines

For the evaluation of methods proposed in Sec. 3., we manually annotated a subset of the data with the required tool, disassembled parts, and removal verbs. The selected subset contains all the manuals in the category of *Mac Laptop*, that includes *MacBook Pro*, *MacBook Air*, *PowerBook*, *iBook*,

```
{
  "Title": "PowerBook G4 Aluminum 12" 867 MHz Logic
  Board Replacement", "Guideid": 213, "Category":
  "PowerBook G4 Aluminum 12" 867 MHz", "Url":
  <link to the guide>,
  "Ancestors": ["PowerBook G4 Aluminum 12" 867 MHz",
  "G4 Aluminum Series", "PowerBook", "Mac Laptop",
  "Mac", "Root"],
  "Toolbox": ["4mm Nut Driver", "Phillips #00
  Screwdriver", "spudger", "t6 torx screwdriver",
  "Arctic Silver ArcticClean", "coin"]
  "Steps": [ {"Order": 1,
  "lines": ["Remove the four Phillips from the memory
  door.", "Slide the memory door away from the
  memory compartment"]
  "Image": [<links to the images>], "StepId": 43711},
  "Tools": ["Phillips #00 Screwdriver"],
  "Word_level_parts":
  [{"Name": "four phillips", "Span": [2,3]},
  {"Name": "memory door", "Span": [10,11]}],
  "Step_level_parts":
  ["four phillips screws", "memory door"]
  "Verbs": [{"Name": "remove", "Span": [0,0], "Part_indx": 0},
  {"Name": "slide away", "Span": [8,8], "Part_indx": 1}
  ,...]
```

Figure 1: An example of annotated data from iFixit.com. The gray area shows the annotation values. Only Step 1 out of 48 in this guide is shown here.

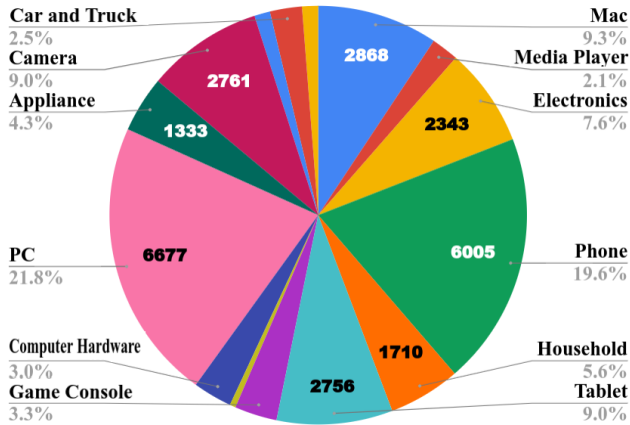


Figure 2: Number of collected manuals in different device categories.

and some other varieties of Mac laptops. This selection was due to it having the highest average number of steps and tools per manual, 24.5 and 4.3, respectively. With an approximate estimation based on the specified level of difficulty and the required repair time in iFixit, these devices appeared to be among the most complicated devices for repair. One should note that in iFixit, the instructors often copy the text description of some steps from other manuals in order to create a new guide, and hence, there are steps with identical text descriptions. In fact, there are 4,350 steps with unique text descriptions in this category. Therefore, we manually annotated the steps with unique text descriptions and then applied the same annotation to the steps with identical text. Table 1 shows the statistics of annotated data.

Number of manuals	1,497
Number of steps	36,973
Number of sentences	81,083
Number of unique tools	81
Number of unique disassembled parts	950
Number of unique verbs	49

Table 1: Statistics of dataset with human annotation.

### 2.2.1. Annotation of Required Tools

The goal of this annotation is to specify the tools that are needed for each step. During the annotation, the annotator observed the corresponding text and image of a step and chose a tool from the toolbox of the manual. When a tool was utilized in a step without being mentioned in the toolbox, the annotator entered the tool name manually. Each step in iFixit is typically focused on one particular part of the device, and even if multiple tools are applied, it is not always clear whether the tools are used simultaneously or sequentially. Therefore, we decided to retain the multi-tool steps (2.8 % of the annotated steps) instead of splitting them into multiple steps. In cases where the instructor had referred to another related step that should also be performed, we annotated the current step with the tool associated with the related step. Additionally, we used the label *no tool* for the steps that do not require any tool. To obtain consistent tool names, we hand-crafted a small set of rules for standardizing the entered names and fixing spelling errors, e.g., changing the string "Ph0 screwdriver" to the more popular

name "Phillips 00 screwdriver". When it was not possible to distinguish the exact required tool, neither from text nor from the image, we annotated the step with the general category of the tool, based on the available information. For example, when we know the needed screwdriver is of type Torx, but its size is not apparent, and there is a *T8* and *T6 Torx screwdriver* in the toolbox, we annotate the step only with *Torx screwdriver*.

### 2.2.2. Annotation of Disassembled Parts

Concerning the disassembled parts, only the parts which are entirely removed from the mainframe were considered as disassembled. For example, when a component is removed from its holder, but a cable or ribbon still connects it to the device, we do not label it as disassembled. When a part is removed from another component of the device, e.g., removing the bracket from the hard drive, it (bracket) is also labeled as disassembled. Apart from the parts, we also annotated the removal verbs referring to the act of detachment of the part. These action verbs are labeled in the text with an index pointing to the corresponding disassembled part in the step. In the case of a verb phrase ellipsis, we did not extend the verb index to the parts with ellipted verbs. For example, in the sentence "*Remove two black Phillips screws from the right side of the board. One 6 mm Torx screw from the left side*", the verb *remove* is indexed only for the *Two black Phillips screws*. We included the verb particles in the annotation of phrasal verbs, as they play a crucial role in the meaning. For instance, the particles *off*, *away*, and *out* are fairly explicit indications of detaching, e.g., in *lift-off*, *slide-away*, and *pull-out*.

Our proposed annotation of disassembled parts consists of two schemes:

1. **Word-level Annotation:** In this approach, we want each word in the text to be labeled with one of the three tags: 1- Disassembled part, 2- Removal verb, and 3- Other. The noun phrase representing a part, including the quantity and modifiers of the noun, is considered as the part's name, e.g., *Two silver Phillips screws*. Still, when a modifier is not referring to the intrinsic properties of the part, we did not include it in the label, e.g., the word *following* in the phrase *following 5 mm Hex nuts*. A part can be mentioned multiple times before the instructor refers to its detachment, and we only label it once it is disassembled.
2. **Step-level Annotation:** In some cases, the information in the word-level annotation is not sufficient for representing the detached part. For such steps, we also added a "Step-level" annotation. The main uses of this schema can be categorized into the following cases:

- (a) **Noun Ellipsis:** In the cases of ellipted nouns (the *zero anaphora* problem), we manually write the complete noun in the step-level annotation. E.g., in the sentence *Remove the following P5 pentalobe screws securing the lower case to the MacBook Pro: Eight 3.0 mm, Two 2.3 mm.*, the step-level annotation is *Eight 3.0 mm P5 pentalobe screws*, and *Two 2.3 mm P5 pentalobe screws*.

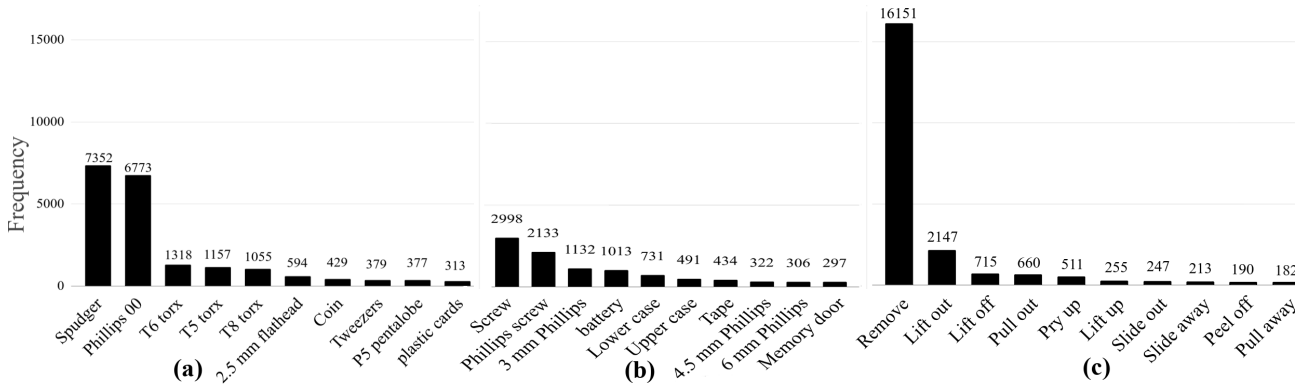


Figure 3: Distribution of the 10 most common tools (a), disassembled parts (b) and removal verbs (c).

(b) **Implicit Detachment of the Parts:** There are steps in the dataset, in which the detachment of the part is not explicitly mentioned in the text, although it can be inferred by human readers through other pieces of information. For example, there could be only some wires that keep a component attached to the device, and the instructor may only mention disconnecting the wires, while the detachment of the part is not explicitly mentioned in the text. In some steps, identifying the detached part requires understanding the next step or the title—which can be interpreted as the goal—of the task. For example, when the goal of the task is the detachment of a part, and the instructor only explains detachments of the blocking components, the actual detachment of the target part may not be explained, assuming it to be obvious. Such steps with an implicit act of detachment are annotated with the name of the disassembled part at the step level.

(c) **Pronominal References to the Parts:** When a pronoun receives a removal action in a sentence, and the referent of the pronoun exists in a separate sentence, we label the pronoun in the word-level and the referent in the step-level annotation. E.g., in the step, *grasp the optical drive by tilting it from the right-hand side. Then lift it gently out of the lower case.*, at the step level *optical drive* and at the word-level, *it* in the last sentence is the annotation of disassembled parts.

Figure 3 (a) shows the number of steps with the ten most frequently used tools. The distribution of the ten most common disassembled parts and the disassembly verbs is shown in Figure 3 (b) and (c), respectively.

### 3. Methods for Information Extraction

In this section, we introduce methods for extracting the information from each step. The proposed methods can serve as the baselines for this IE task and also provide means to automatically annotate other parts of the data set, which are so far not annotated.

#### 3.1. Extracting the Required Tool

In repair manuals, including the iFixit data, it is common that the required tools are specified for the overall repair task. Here, however, we are interested in knowing the tools

that are needed for each step of the task. Therefore, we utilize the toolbox provided by the instructors as a gazetteer<sup>3</sup> and search for the complete and/or partial mention of tools in the text. In many cases, the tool names include size and/or type information, possibly shared between the tool name and the name of the corresponding object, e.g., Torx/Phillips/Tri-point/Flathead screwdrivers corresponds to Torx/Phillips/Tri-point/Flathead screws. The instructors often hint at a suitable tool by mentioning sizes or types.

We use a measure of text similarity to search for the gazetteer values, i.e., tools in the toolbox, in the text. Broder (1997; Kondrak (2005) compared the sets of  $n$ -grams ( $n$ -shingles) in two documents for calculating a text similarity. We extend this approach by using  $n$ -grams of variable size to search for the tool names in the text. If  $T$  is the set of all required tools in the manual’s toolbox, for each tool  $t \in T$  we generate a **bag of  $n$ -grams** ( $\text{BoN}(t)$ ). A bag of  $n$ -grams, defined as in (Arora et al., 2018), is the collection of all  $w$ -grams for  $w \leq n$  appearing in the text. We set  $n$  as the number of tokens in the tool name. Accordingly, a bag of  $n$ -grams ( $\text{BoN}(l)$ ) with the same maximum size ( $n$ ) is produced from the list of singularized word tokens in the text description ( $l$ ) after removing some frequent words with a stop list. The similarity between  $t$  and  $l$   $\text{Sim}(t, l)$  is calculated by the Jaccard coefficient, with the exception that we do not need to normalize the scores over the size of  $\text{BoN}(l)$ , as we only need to find the most fitting tool in each section of the text  $l$ :

$$\text{Sim}(t, l) = \frac{|\text{BoN}(t) \cap \text{BoN}(l)|}{|\text{BoN}(t)|}. \quad (1)$$

To find the most fitting tool(s) for each step, we calculate the similarity score between each line in the step and every tool in the toolbox. If all the scores are zero, we return “no tool” for that step. In the case of multiple maxima in the similarity score of a line, the model first tries to extract the more general category of the tools. The general category of tools is derived from the longest common subsequence with a possible gap, among the words in the tool names. For example the general category of *Phillips #00 screwdriver* and *Phillips #1 screwdriver* would be considered as

<sup>3</sup>A gazetteer is a set of lists containing names of entities such as cities, organizations, days of the week, etc., which are used to find occurrences of these names in text (Cunningham et al., 2009)

*Phillips screwdriver*. If there is no common sub-sequence among the tools, we return the most frequent tool in the device category. The algorithm 1 shows the procedure of tool extraction.

---

**Algorithm 1** Extracting the tool(s) in a step

---

**Procedure** step\_tool\_extraction  
**Input**  $T \leftarrow$  toolbox,  $c \leftarrow$  dev category,  $s \leftarrow$  step lines  
1: extracted\_tools  $\leftarrow$  []  
2: **for each**  $l \in s$  **do** ▷ Each line in step  
3:   scores  $\leftarrow$  []  
4:   **for each**  $t \in T$  **do** ▷ Each tool in toolbox  
5:     append  $sim(t, l)$  to scores  
6:   **if**  $\sum_T scores \neq 0$  **then**  
7:      $\hat{t} \leftarrow \arg \max_{t \in T} scores$   
8:     **if**  $len(\hat{t}) = 1$  **then** ▷ One maximum  
9:        $t^* \leftarrow \hat{t}$   
10:    **else** ▷ Multiple maxima  
11:      $t_g \leftarrow$  general category of tools in  $\hat{t}$   
12:     **if**  $t_g = \emptyset$  **then** ▷ No common sub-sequence  
13:        $t^* \leftarrow$  most frequent tool of  $\hat{t}$  in  $c$   
14:     **else**  
15:        $t^* \leftarrow t_g$   
16:     **if**  $t^* \notin$  extracted\_tools **then**  
17:       append  $t^*$  to extracted\_tools  
18: **if** extracted\_tools =  $\emptyset$  **then**  
19:   **return** “no tool”  
20: **else**  
21:   **return** extracted\_tools

---

### 3.2. Extracting the Disassembled Parts

We propose a baseline model for the extraction of disassembled parts and their corresponding verbs in the word-level annotation. The model employs a BiLSTM-CRF sequence labeling architecture (Huang et al., 2015), receiving the *pooled contextualized embeddings* introduced in (Akbik et al., 2019b) as input. This design was selected because of its outstanding performance in sequence labeling tasks. Moreover, the pooled contextualized embeddings showed a good performance in representing the semantics of the rare words, which can appear in our data, e.g., the technical names of components. The BiLSTM layer is used to obtain the semantics of both past and future text, and the CRF layer is used to predict the tags of the entire step jointly by considering the dependencies of output tags. For each word, the model produces a probability distribution over the possible tags, given the surrounding context and the predicted tags.

For training the model, the parts and verbs of the word-level annotations are converted into a BIOE format, where B-, I- and E- tags indicate beginning, intermediate, and end positions of entities and O-tags represent every other word outside the entities, e.g., *B-part*, *E-verb*, etc. The input of the model is the concatenation of FLAIR contextualized embeddings (Akbik et al., 2019a), and Stanford’s trained glove embedding (Pennington et al., 2014). For the training and evaluation, we randomly divided the annotated “Mac Laptop” data into 80% training set, 10% development, and 10% test set. The hyper-parameters are set as in the best configuration suggested in Akbik et al. (2019b) for the named

#	Error type	Percentage
1	Needs picture	23.5
2	Misleading words	21.9
3	Needs extra knowledge	16.2
4	Tool not in the toolbox	13.7
5	Reassembly tip	10.7
6	Referring to another step	5.8
7	Multiple tools in a line	4.8
8	Other	3.4

Table 2: Percentage of error types for the tool extraction baseline.

entity recognition tasks.

## 4. Evaluation

In this section we evaluate the performance of the proposed models in extracting the relevant information.

### 4.1. Tool Extraction

To evaluate the tool extraction, we compare the extracted tool according to Algorithm. 1 with the human annotations described in Sec. 2.2. The multi-tool steps are only considered as correct if the set of extracted tools matches the annotation set. This approach produced **94.3%** accuracy on the entire annotated data set (*Mac Laptop* category).

#### 4.1.1. Analysis of Tool Extraction Errors

To better understand the method’s performance, we manually analyzed all errors of the method and categorized them. Table 2 shows the distribution of error types. Here is the explanation of each error type:

1. When there is not enough information about the tool in the text, however, the tool is clearly recognizable from the picture.
2. When there is some notion of a tool or related object in the text, but the tool is not used. For example, the instructor refers to a *Phillips screw* to specify the location of another component.
3. When the required tool is recognizable from the text but by using extra knowledge. An ontology of relations between tools and corresponding objects and actions, e.g., the relation between *Wrench* and *Bolt*, seems to be necessary for such cases.
4. When a tool is used without being mentioned in the toolbox. This mainly happens for general-purpose tools such as *needle* or *coin*.
5. Since the reassembly part is not included in the manuals, the instructors sometimes give tips for reassembly inside a disassembly step, and they might also refer to an extra tool.
6. When the instructor referred to another step that should also be performed and it needs a tool.
7. When multiple tools are used in a single line of instruction, which will lead to errors as the model only extracts one tool from each line in the step.
8. Other types of errors, mainly typos in the tool names.

Handling these situations requires a more complex model capable of semantic analysis of the text or the inclusion of other information sources, such as ontologies or images.

Despite the limitations and the simplicity of the model, we have already achieved a good accuracy, as evaluated in Section 4.1.

## 4.2. Part Extraction

The evaluation of the baseline part extraction is performed by calculating the model’s precision and recall in labeling the entities in the text. In the annotated data, 6.1% of the steps have a step-level annotation. In those steps, the word-level annotation might be empty, e.g., in the cases of implicit detachments, or can be incomplete, e.g., in the presence of noun ellipsis. In any case, the evaluation in this paper only takes the word-level information into account, and we postpone a more detailed evaluation for future work. Table 3 shows the precision, recall, f1-score values of the model. The final scores are calculated by micro averaging over the steps.

Entity	Precision	Recall	F1-score
Parts	0.87	0.90	0.88
Verbs	0.95	0.94	0.95

Table 3: Precision, recall and f1-scores of part and verb extraction in word-level annotation.

The model shows a better performance in labeling the verbs. The reason can be the lower diversity of verbs, as it is observable from Figure 3 and table 1.

## 5. Semi-Automatic Annotation Web Tool

We offer the MyFixit dataset along with a web-based annotation tool <sup>4</sup> that facilitates the annotation of data. Compared to the available annotation tools such as Gate (Cunningham et al., 2009) and Brat (Stenetorp et al., 2012), in which the user first selects the span of words and then chooses a label, our application speeds up the annotation by suggesting candidates in the form of checkboxes. The checkboxes are checked by default for candidates that are extracted using the proposed methods in this paper. The user can yet manually modify the information in case of mistakes in suggested candidates.

As for the required tools of the repair tasks, the suggested candidates are the tools in the toolbox, while the extracted tools with the method in Section 3.1. are checked by default. Regarding the disassembled parts and removal verbs, the user can choose to utilize the supervised method proposed in Section 3.2., or a simple unsupervised approach that leverages the information from each annotated step to the next ones.

In this approach, the app first employs a deep learning-based shallow parser, implemented in the Flair framework (Akbi et al., 2019a), to extract the noun and verb phrases from sentences. The nouns are further filtered by Wordnet (Miller, 1995) so that only the nouns that are hyponyms of “Artifact” will be suggested as the part candidates. When the user annotates some parts and verbs in a step, those parts

<sup>4</sup><https://github.com/rub-ksv/MyFixit-Annotator>

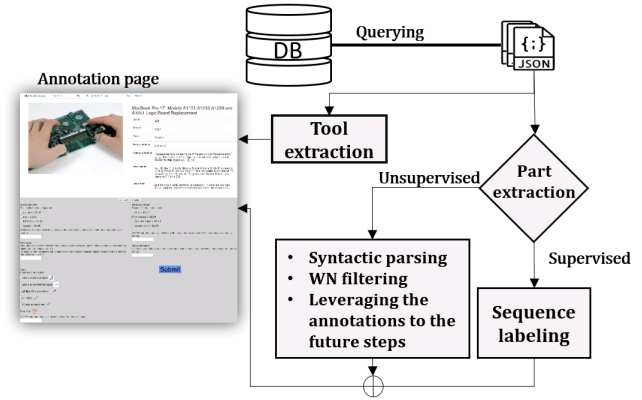


Figure 4: The process of data preparation for the annotation tool.

and verbs would be checked by default if they appear in the succeeding steps, and the Wordnet filtering will not affect them. The app also tokenizes the sentences in each annotated step and uses the annotation of the sentences for the coming steps. If any of the annotated sentences appear in the next steps, it would be removed from the description of the step and will be shown in a separate section. Consequently, as the user proceeds with the annotation, she has fewer sentences to read and annotate, while at each step, it is possible to control and change each piece of the information. Figure 4 shows the flow of data in the annotation tool. For more detailed information about the annotation tool, please refer to its Git repository.

## 6. Related Work

Related work on IE from instructional texts can be compared in several aspects, including the domain of instructions, the representation of the information, and the methods for extracting the proposed representation from the instructions.

The choice of representation structure is often influenced by the domain of data and the intended application of extracted knowledge. For example, in the domain of cooking, Mori et al. (2014) collected 200 Japanese recipes and annotated them with several concepts in the text, including the foods, utensils, actions and durations (e.g., the water must boil for 5 minutes), or the state of the food (e.g., taste and color) at certain steps. These concepts are extracted with supervised named-entity recognition techniques and are later represented as the nodes in a directed acyclic graph (Maeta et al., 2015; Yamakata et al., ). A similar dataset of 260 English cooking recipes was developed by Tasse and Smith (2008) for studying segmentation models. In this work, the so-called *Minimal Instruction Language for the Kitchen* (MILK) was also introduced. Such representations are proposed for and specialized to cooking and are hence not easily applicable in other domains.

While probabilistic graphs provide an overall representation of a workflow, it is sometimes desirable to obtain intrinsic properties of individual entities along with their relations, for example, the fact that the location of some ingredients is in the fridge, or that certain knives are intended for cutting specific ingredients. With such a goal, Kaiser et al. (2014) extracted a domain ontology of common

sense knowledge by parsing cooking recipes. In the field of robotics, extracted knowledge bases can help to reduce the amount of hard-coded information needed for planning tasks such as cooking pancakes or biscuits (Tenorth et al., 2011; Bollini et al., 2012). An ontology in the domain of repair manuals promises similar benefits, for instance, for disambiguating the required tool, countering the 3rd type of extraction error in Sec. 4.1.1.

Despite the many efforts made towards information extraction from kitchen recipes, only a few works have explored instructional text in other domains. In one example, Zhang et al. (2012) proposed a more general representation of steps in procedural tasks, including the semantic elements *ActionVerb*, *Actee*, *Instrument*, *Purpose* and *pre/post-condition*. Apart from recipes, they collected several car and aircraft maintenance instructions, 74 in total, to examine the performance of their syntactic parser for extracting such a representation. Parsing-based methods are most beneficial when the task is to find every action in a text, rather than some particular actions. Moreover, mapping the result of syntactic parsers to limited extraction patterns is more prone to the risk of missing entities in under-specified or ungrammatical language.

Perhaps the most similar data to iFixit is found on Wikihow. On the data collected from wikihow.com, Chu et al. (2017) used an Open IE system for the extraction of general-purpose semantic frames from the semi-structured text. The target of IE in this work was to build a knowledge base (the *HowToKB*) for expediting the search process and answering “how-to queries”. In our work, we are interested in the specific entities with fixed relations, while Open IE systems extract all potential relationships with no pre-specified types. The published data in this work also differ in that they do not have any human ground-truth annotation.

Information extraction from instructional text is often performed by driving a dense representation of meanings from every sentence of the instructions. Consequently, it is assumed that the steps are described with single actions in imperative sentences. With such an assumption, the first step in the manual shown in the appendix breaks down to multiple actions (remove, insert, search, pull). Extracting all these actions in isolation is not in the interest of this work. Instead, we seek representations that are capable of capturing the long-term dependencies across passages of text to derive higher-level information from the entire semantic block of each step, i.e., knowing that the *expansion bays* are detached without requiring any tool. We consider such integrated information as far more practical for a collaborative assistant. Repair instructions usually include long passages, and in the data we extracted, 60% of the steps have more than one sentence, while only 31% of steps start in an imperative form. In many cases, a step begins with precaution measures, and it contains extra explanations about the components, the devices, and sometimes about companies. Depending on the task, it might take several sentences, and even several steps, to detach a single component. Moreover, as we observed in the iFixit data, the text description of steps is not always sufficient for extracting the desired information, so the additional benefit can be drawn from

other attributes of the steps, such as the toolbox. Extracting the information into step-level annotations requires models capable of propagating the context of objects, along with the temporal order of steps and the other properties of the task. The partially observed Markov Decision Process proposed in Malmaud et al. (2014) for interpretation of cooking recipes seems to be a feasible solution for such problems, and we plan to incorporate similar strategies in future work on repair assistants, as well.

## 7. Conclusion

We have presented the first steps for information extraction from repair manuals and towards an intelligent repair assistant. We introduced a dataset of repair manuals with semi-structured information. The dataset is annotated in the category of *Mac Laptops*, with the objects that a user might interact with in the course of the repair task. The dataset is well-suited for studying information extraction from long and technical passages, where the arguments and actions might be elided, implicit and spread across several passages. We then proposed methods that can serve as baselines for determining the required tools and the disassembled parts in each repair step. The methods, along with multiple NLP techniques, are integrated into a semi-automatic annotation web-based tool, which is freely available in addition to the dataset.

In this paper, we only extract the disassembled parts using the word-level annotation of steps, i.e. when the identity of the disassembled part can be extracted by labeling the corresponding words in the text. For future work the method needs to be enhanced for learning the step-level information, where extracting the information goes beyond the word labeling in descriptions and often requires extra knowledge.



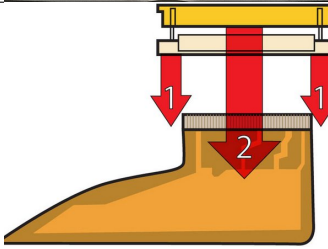

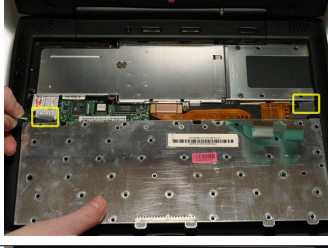

We imagine that an inverted sequence of the object interactions during the disassembly stage can serve as a prior during the reassembly stage. The long-term dependency among the steps and their corresponding objects is a distinctive property of repair manuals. When we detach a component from a device using a tool, eventually, typically after multiple intermediate steps, we have to reattach that component, most likely with the same tool. The specific order of components in the devices, which is partially shared among devices in a category, makes it feasible to attempt predicting the next step in the repair task, e.g., by using predictive models for learning statistical scripts, such as Hidden Markov Models (Orr et al., 2014) or LSTM Neural Networks (Pichotta and Mooney, 2016). Finally, we plan to use the extracted information as the context for a task-oriented dialog in a human-robot interaction scenario and we envisage that this dataset and associated baseline methods are similarly beneficial for work on other smart assistance systems and their constituent NLP and task-understanding components.

## 8. Acknowledgement

We thank Dirk Ruiken for assistance with the project, and for comments that greatly improved the manuscript.

## Appendix

The Complete Manual of “PowerBook G3 Wall-street Keyboard Replacement” in iFixit.

Step	Text	Image	Tool	Disassembled Parts
1	Remove both expansion bays using the levers on the front of the computer. Insert your index fingers inside the expansion bays and search for the two ribbed tabs on the underside of the upper case. The tabs are located near the bottom corners of the keyboard. Pull the tabs toward yourself and the keyboard will pop up.		No tool	<b>Expansion bays</b>
2	Pull the keyboard forward to disengage the tabs holding it in back and rotate it toward you. Rest the keyboard on the trackpad.		No tool	No part
3	This is a diagram of the ribbon clamp connectors you will disconnect in the next step. 1) With your fingernails, grasp the locking bar on either side and pull up a small amount (about 1/16” or 2 mm). 2) After disengaging the locking bar, slide the cable out of the connector.		No tool	No part
4	Disconnect the two keyboard connectors by disengaging the clamps and pulling the ribbons directly upward.		No tool	No part
5	Slide a spudger downward between each plastic strain relief cable and the wall of the case in order to bow out the cable beyond the small tab holding it in place. Once the strain relief cables are free, lift the keyboard off.		<b>Spudger</b>	<b>Keyboard</b>
6	Your laptop should look approximately like this.		No tool	No part



## 9. Bibliographical References

- Akbik, A., Bergmann, T., Blythe, D., Rasul, K., Schweter, S., and Vollgraf, R. (2019a). FLAIR: An easy-to-use framework for state-of-the-art NLP. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics (Demonstrations)*, pages 54–59.
- Akbik, A., Bergmann, T., and Vollgraf, R. (2019b). Pooled contextualized embeddings for named entity recognition. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 724–728.
- Arora, S., Khodak, M., Saunshi, N., and Vodrahalli, K. (2018). A compressed sensing view of unsupervised text embeddings, bag-of-n-grams, and LSTMs. In *Proceedings of the 6th International Conference on Learning Representations, ICLR 2018*.
- Bollini, M., Tellex, S., Thompson, T., Roy, N., and Rus, D. (2012). Interpreting and executing recipes with a cooking robot. In *Proceedings of the 13th International Symposium on Experimental Robotics, ISER*, volume 88 of *Springer Tracts in Advanced Robotics*, pages 481–495.
- Broder, A. Z. (1997). On the resemblance and containment of documents. In Bruno Carpentieri, et al., editors, *Compression and Complexity of SEQUENCES*, pages 21–29. IEEE.
- Chu, C. X., Tandon, N., and Weikum, G. (2017). Distilling task knowledge from how-to communities. In *Proceedings of the 26th International Conference on World Wide Web*, pages 805–814. ACM.
- Cunningham, H., Maynard, D., Bontcheva, K., Tablan, V., Ursu, C., Dimitrov, M., Dowman, M., Aswani, N., Roberts, I., Li, Y., et al. (2009). *Developing Language Processing Components with GATE Version 5:(a User Guide)*. University of Sheffield.
- Deutsch, B. G. (1974). *The structure of task oriented dialogs*. Pittsburgh, PA: Carnegie-Mellon University.
- Grice, H. P. (1975). Logic and conversation. In *Speech acts*, pages 41–58. Brill.
- Huang, Z., Xu, W., and Yu, K. (2015). Bidirectional LSTM-CRF models for sequence tagging. *CoRR*, abs/1508.01991.
- Kaiser, P., Lewis, M., Petrick, R. P. A., Asfour, T., and Steedman, M. (2014). Extracting common sense knowledge from text for robot planning. In *Proceedings of 2014 IEEE International Conference on Robotics and Automation, ICRA*, pages 3749–3756.
- Kondrak, G. (2005). *N*-gram similarity and distance. In Mariano P. Consens et al., editors, *Proceedings of 12th International Conference on String Processing and Information Retrieval, SPIRE*, volume 3772, pages 115–126. Springer.
- Maeta, H., Sasada, T., and Mori, S. (2015). A framework for procedural text understanding. In *Proceedings of the 14th International Conference on Parsing Technologies, IWPT*, pages 50–60. Association for Computational Linguistics.
- Malmaud, J., Wagner, E., Chang, N., and Murphy, K. (2014). Cooking with semantics. In *Proceedings of the ACL 2014 Workshop on Semantic Parsing*, pages 33–38.
- Miller, G. A. (1995). WordNet: a lexical database for english. *Communications of the ACM*, 38(11):39–41.
- Mori, S., Maeta, H., Yamakata, Y., and Sasada, T. (2014). Flow graph corpus from recipe texts. In *Proceedings of the LREC*, pages 2370–2377.
- Orr, J. W., Tadepalli, P., Doppa, J. R., Fern, X. Z., and Dietterich, T. G. (2014). Learning scripts as hidden markov models. In *Proceedings of the Twenty-Eighth AAAI Conference on Artificial Intelligence*, pages 1565–1571. AAAI Press.
- Pennington, J., Socher, R., and Manning, C. (2014). GloVe: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543.
- Pichotta, K. and Mooney, R. J. (2016). Learning statistical scripts with LSTM recurrent neural networks. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence*, pages 2800–2806. AAAI Press.
- Salas, E., Prince, C., Baker, D. P., and Shrestha, L. (1995). Situation awareness in team performance: Implications for measurement and training. *Human factors*, 37(1):123–136.
- Stenetorp, P., Pyysalo, S., Topić, G., Ohta, T., Ananiadou, S., and Tsujii, J. (2012). BRAT: a web-based tool for NLP-assisted text annotation. In *Proceedings of the 13th Conference of the European Chapter of the Association for Computational Linguistics*, pages 102–107.
- Tasse, D. and Smith, N. A. (2008). SOUR CREAM: Toward semantic processing of recipes. *Carnegie Mellon University, Pittsburgh, Tech. Rep. CMU-LTI-08-005*.
- Tenorth, M., Klank, U., Pangercic, D., and Beetz, M. (2011). Web-enabled robots-robots that use the web as an information resource. *IEEE Robotics and Automation Magazine*, 18(2):58.
- Whitney, D., Eldon, M., Oberlin, J., and Tellex, S. (2016). Interpreting multimodal referring expressions in real time. In *2016 IEEE International Conference on Robotics and Automation (ICRA)*, pages 3331–3338.
- Wilkes-Gibbs, D. and Clark, H. H. (1992). Coordinating beliefs in conversation. *Journal of memory and language*, 31(2):183–194.
- Yamakata, Y., Imahori, S., Maeta, H., and Mori, S. ). A method for extracting major workflow composed of ingredients, tools, and actions from cooking procedural text. In *2016 IEEE International Conference on Multi-media & Expo Workshops, ICME*, pages 1–6.
- Zhang, Z., Webster, P., Uren, V. S., Varga, A., and Ciravegna, F. (2012). Automatically extracting procedural knowledge from instructional texts using natural language processing. In *Proceedings of the LREC*, volume 2012, pages 520–527.