Automatically Extracting Procedural Knowledge from Instructional Texts using Natural Language Processing

Ziqi Zhang[1], Philip Webster[1], Victoria Uren[2], Andrea Varga[1], Fabio Ciravegna[1]
[1] University of Sheffield
[1]Initial.Surname@dcs.shef.ac.uk
[2]University of Aston
[2]v.uren@aston.ac.uk

Abstract

Procedural knowledge is the knowledge required to perform certain tasks, and forms an important part of expertise. A major source of procedural knowledge is natural language instructions. While these readable instructions have been useful learning resources for human, they are not interpretable by machines. Automatically acquiring procedural knowledge in machine interpretable formats from instructions has become an increasingly popular research topic due to their potential applications in process automation. However, it has been insufficiently addressed. This paper presents an approach and an implemented system to assist users to automatically acquire procedural knowledge in structured forms from instructions. We introduce a generic semantic representation of procedures for analysing instructions, using which natural language techniques are applied to automatically extract structured procedures from instructions. The method is evaluated in three domains to justify the generality of the proposed semantic representation as well as the effectiveness of the implemented automatic system.

Keywords: procedural knowledge, information extraction, instructional text

1. Introduction

Procedural knowledge is the knowledge required to perform certain tasks. It is extensively used in our daily life, and forms an important part of expertise. There is a broad range of research studies addressing the usage of procedural knowledge, such as automatic user documentation generation (Paris et al., 2002; Paris et al., 2005), or to support procedural question answering (Delpech and Saint-Dizier, 2008). In the area of process automation, procedural knowledge can be modelled as workflows or programs to enable workflow engines or programs to control the process.

Due to the importance of procedural knowledge, continuous efforts have been dedicated to the research related to the acquisition of procedural knowledge. One popular direction is learning procedures from natural language instructions or procedural texts, which are descriptions of steps for performing certain tasks written in human natural language. Examples of such include recipes, teaching texts, and product usage manuals. These are commonly used by human learners to carry out tasks and acquire new skills, and are usually well documented and easily available. However, automatically extracting procedural knowledge from texts is a challenging task. Firstly, natural language texts must be processed and transformed into well-formed structured procedures interpretable by programs. Secondly, human instructions are naturally plagued with imperfections such as ambiguities, omissions, unintentional inconsistencies and errors (Gil, 2010), which must be resolved.

Among the related work, many (Bielsa and Donnell, 2002; Aouladomar, 2005; Gil, 2010) have studied the topological, grammatical and semantic patterns of instructional language to identify essential components of procedures and their interrelated information. However, very few have addressed automatically acquiring procedural knowledge from instructions, except Paris et al. (2002) and Gil et al. (2011), which are limited in certain ways.

This paper makes two contributions towards the research on automatic procedural knowledge acquisition from texts. Firstly, we introduce a formal representation of procedural knowledge and we describe how the representation can be used to facilitate tasks such as question answering, documentation generation, and procedure automation. Secondly, we introduce a method of automatically extracting procedural knowledge from natural language instructional texts based on natural language processing techniques. This is then evaluated in three domains. The experiments have shown that, despite the high level of regularities in instructional language as discovered in previous studies, annotating instructions is a non-trivial task. Even human annotators have found it difficult to annotate certain procedural elements. As a result, the automatic extraction system has achieved good accuracy on some types of procedural elements, but also suffered from reduced accuracy where substantial exceptions are found. To encourage future research, we make available some of these corpora and their annotations. To our knowledge, this is the first annotated instruction corpus that is publicly available.

The remainder of this paper is organised as follows. Section 2 discusses the motivations for this work in details, and proposes requirements for building automated procedural knowledge acquisition systems. Section 3 reviews state-of-the-art and discusses their limitations. Section 4 introduces the proposed semantic representation and Section 5 proposes a method for automatic procedural knowledge acquisition based on NLP techniques. Section 6 describes corpus preparation, experiments and discussion, followed by a conclusion in Section 7.

2. Motivations

Automatic procedural knowledge acquisition has broad application in practice. From the human users'

perspective, procedural knowledge can be used in automatic user documentation generation (Paris et al., 2002; Paris et al., 2005), or support procedural question answering (Murdock et al., 2007). In the area of process automation, procedural knowledge can be formalised as workflows or programs, which are used by a workflow engine or procedure program to automatically monitor context parameters and control the procedure execution. Usually, such data must be authored manually and require certain level of expertise. The capability of automatically acquiring procedural knowledge from instructions and generating executable procedures would be a desirable feature (Fritz and Gil, 2011; Gil, 2010) in automation systems.

Although traditionally workflows and automated procedures are primarily used in business contexts for process controlling (Tang and Hwang, 1996; Muehlen, 2001), recent research (Mühlhäuser, 2008) has introduced similar ideas in consumer environments. It has been argued that new products today are equipped with increasing amount of functionalities, accompanied with increasing complexity and difficulty of tasks associated with them. Researchers of smart products and smart environments have advocated equipping products with procedural knowledge and reasoning capabilities to enable them to guide users through complex tasks and provide necessary support, such as handling exceptions, proposing substitutions, and delegating sub-procedures (e.g., "pre-heat the oven" in a recipe) to intelligent agents that can automate certain parts of a procedure. For instance, in a ubiquitous "smart" kitchen environment, a central workflow engine monitors the state of a procedure, while "talking" to other intelligent devices such as oven, kettle, and food processor, to request for certain service and delegate parts of the procedure. In such an environment, different "smart" consumer devices possess specific procedural knowledge about carrying out certain tasks (e.g., pre-heat oven, blending), and manipulates parts of a procedure. The central management agent (workflow engine) controls the overall process, being able to request certain services and delegates part of a procedure; while the execution logic of each part is fulfilled by each distributed devices. To realise this scenario, an effective approach to acquiring procedural knowledge from traditional sources – instructions – is critical.

However, as mentioned before, acquiring procedural knowledge from human natural language instructions is a challenging task. On the one hand, natural language instructions are not interpretable by machines; on the other hand, they often contain imperfections, which may hinder the automatic acquisition process. In the following, we propose several requirements for a procedural knowledge acquisition system:

- **A formal structured representation of procedure** defines elements of procedures and their relations. This structured representation can support enabling machine interpretation (e.g., reasoning, monitoring), and the elements may correspond to programmable elements in a procedure program. Ideally, the representation should be domain-independent so it can be applied in any domains.
- **Natural language processing** techniques that can automatically process instructions and generate structured representation of procedures according to the semantic representation.
- **Robustness in handling imperfections** in instructional language: The system should capture, and ideally rectify errors or missing information found in instructions.

3. Related Work

Many works (Kosseim and Lapalme, 1998; Bielsa and Donnell, 2002; Aouladomar, 2005) have analysed instructional texts to study the topological and grammatical structures of instructions, which identify semantic elements of a procedure. These are often referred to as "semantic analysis" of instructions. The studies resemble an annotation process, where instructions are segmented into related expressions according to their underlying procedural semantics or functions. It has been found that instructional language is often highly organised and consists of a confined set of grammar and vocabularies. Despite the extensive coverage of domains and large amount of documents studied in these works, none has studied the difficulty of annotating human instructions, or made the corpora available as reference for future work. Gil (2010) analysed instructions as means of learning procedures and has shown that procedures can be very complex and instructions can be difficult to interpret. Human instructions are naturally plagued with imperfections, such as omissions, inconsistencies, and errors. These have created challenges in learning procedures from instructions.

While these studies address language analyses of instructions and representation of procedures, research on automatic procedural knowledge acquisition from instructions is scarce. Brasser and Linden (2002) and Paris et al. (2002) applied natural language processing techniques to automatically analyse instructions, extract lexical segments corresponding to certain procedural elements and create a structured representation in the form of "task models" – a key element of model-based user interface design. The representational model builds on top of state-of-the-art representations of procedures of instructions, including concepts such as task elements (steps), conditions, purposes, actions, objects of actions, their locations and instruments. Among these, only a limited set are dealt with by the system and tested. The outputs however, are not structured workflows or procedures, but rather semi-structured texts.

Gil et al. (2011) proposed an interactive system that helps non-programmer users to translate natural language instructions to executable procedures, while handling certain ambiguities and omissions in instructions. The system relies on substantial prior domain knowledge, in the forms of controlled vocabularies and paraphrase

patterns. A set of controlled vocabularies (e.g., actions, objects) are pre-defined with respect to the domain of interest, such that the system prompts users to use "known" vocabularies rather than introducing new terms when writing instructions. Paraphrase patterns are lexical patterns to match segments of instructions to pre-defined program code. For example "descend to a new position with latitude=[X]" is mapped to a function that takes "latitude" as argument and serves the "descending" action. Heuristic and deductive reasoning are applied to fill or prompt for missing information in original instructions. The system is later combined with learning-by-demonstration in a framework for assisting end-user programming in Fritz and Gil (2011). The major limitation of this approach is its heavy dependence on prior domain knowledge. Although ultimately it is essential to encode domain knowledge for process automation, we view the learning process as two independent sub-processes: one identifies generic procedural elements and creates a structured representation; the other further processes them by attaching domain-specific knowledge to enable automatic execution. The first process can be generalisable to any instructions thus benefiting any domains; also, it can facilitate the next sub-process.

4. Representation of Procedural Knowledge

In this section, we discuss a semantic representation of procedural knowledge in order to support various tasks. Previous studies such as Aouladomar (2005), Bielsa and Donnell (2002), Delpech and Saint-Dizier (2008), Gil et al. (2011), Kosseim and Lapalme (1998) and Paris et al. (2002) have proposed different formalisations of procedures. The formalisation defines a structured representation of procedures by specifying semantic elements of a procedure and their interrelated information. We propose a generic representation shown in Figure 1, which as we shall show, can support a number of tasks related to procedural knowledge.

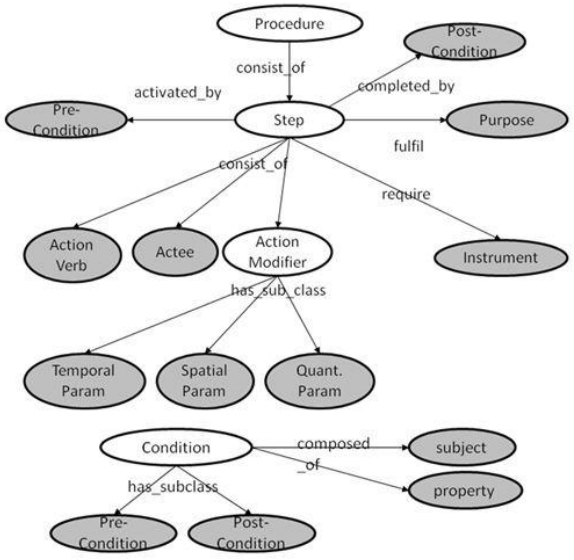

Figure 1. The formal representation of a procedure.

We define a procedure as an ordered sequence of *steps*. A *step* may be activated when certain *pre-condition* is satisfied. Likewise, the step may reach its end state when certain *post-condition* is satisfied. Pre-condition and post-condition are further decomposed to *subject* (e.g., oven) and *property* (e.g., temperature=200C). A step may have an individual objective to achieve, which is represented by *purpose*. A step consists of: an *action verb*, which represents the core activity or action (e.g., blend, pre-heat); *actee*, which refers to the object(s) involved; *instrument*, which refers to the mechanisms used for performing the activity; and *action modifier*, which are constraints on the activity. Action modifiers are sub-classed into three generic types, *temporal parameters* (TemporalParam) describe the duration of an action, such as "10 minutes"; *spatial parameters* (SpatialParam) describe directional and location change, such as "pull the handle upwards", "strain the juice into the glass", "; *quantitative parameters* (Quant.Param) generalize expressions of quantitative amount, such as "10 ml", "five strawberries", "click twice. Figure 2 shows example instructions and their structured representation.

Set the oven to 200C/Gas mark 6. ⇨ ActionVerb= "set" Actee = "oven" ActionModifier-Quant.Param= "200C" OR "gas
Turn on the heater until temperature reaches 26C. ⇨ ActionVerb= "turn on" Actee = "heater" Post-condition= Subject: temperature; Property: 26C
To make the paste, blend the chick pea for 1 minute using a blender. ⇨ Purpose= "make the paste" ActionVerb= "blend" Instrument= "blender" Actee = "chick pea" ActionModifier-TemporalParam= "1 minute"

Figure 2. Example instructions and corresponding structured representation

This generic representation of procedures can be used to support different tasks related to procedural knowledge, such as documentation generation, procedural question answering and procedure automation.

Documentation generation: According to Gil (2010), creating comprehensive and comprehensible instructions and user documentations is not a trivial task. Procedures can be very complex and natural language instructions are "naturally plagued with omissions, oversights, unintentional inconsistencies, errors, and simply poor design". We argue that the proposed formal representation of procedures can support human authors in this process by specifying the necessary elements in a procedure and their interrelated information. In fact, some work on Natural Language Generation has already explored the usage of a "task model" – simplified procedure formalism

– to guide users in the creation of documentation (Paris et al., 2002; Paris et al., 2005).

Question answering: Recent research has shown that procedural questions are the second largest set of queries on search engines (De Rijke, 2005). Delpech and Saint-Diezer (2008) has proposed to parse procedural texts into structured representations to support procedural question answering. The representation identifies key information elements in a procedure such as titles, advices and goals, which can be used by search engines to locate important information. Our proposed representation is even more fine-grained, supporting more specific questions related to a procedure such as "what tools are required to perform the task" (instrument), and "how do I know when to move on to the next step" (pre-/post-condition). These are crucial information to support users through a task.

Procedure automation: The generic representation can be coupled with domain-specific knowledge in order to support procedure automation as illustrated in the scenarios of SmartProducts (Mühlhäuser, 2008). The scenarios depict a ubiquitous environment in which a number of smart devices that each has certain functionalities can communicate and collaborate to assist users to complete a procedure in a distributed manner. For example, in a smart kitchen environment, the overall sequence of a recipe process can be coordinated by a central device. For each step, it broadcasts the metadata across the network to other devices. Devices are equipped with domain knowledge, and are able to recognise data that describe its functionalities. For example the step with an ActionVerb "blend" and a parameter "1 minute" can be recognised by a smart food processor which matches the action to its functionality "blend" with the timing parameter of "1 minute". It then sets up these parameters automatically and responds to the central coordinating device to request for user attention and task delegation. Upon receiving the nonfiction the user simply places the ingredients into the food processor to trigger the process, and then continues with other sub-tasks.

One essential step to enable this intelligent behaviour is a structured representation of procedures that specifies key information with adequate level of details, as well as the capability to automatically extract such structured representations from legacy data. As it is clear in the example, our formalisation of procedures provides essential support for this purpose.

5. Automatic Extraction from Texts

Our method of automatic procedural knowledge extraction from texts employs natural language parsing to partition sentences into related segments, from which a set of finite-state grammars are applied to extract procedural elements defined in Figure 1. Next, simple rule-based reasoning is applied to resolve certain types of omissions and ambiguities in instructions.

Natural language parsing: In this process, full syntactic parsing is applied to individual sentences to identify the grammatical segments and their hierarchical relations. The parsing process takes an input sentence and produces a tree representation of the grammatical segments of a sentence, as shown in Figure 3.

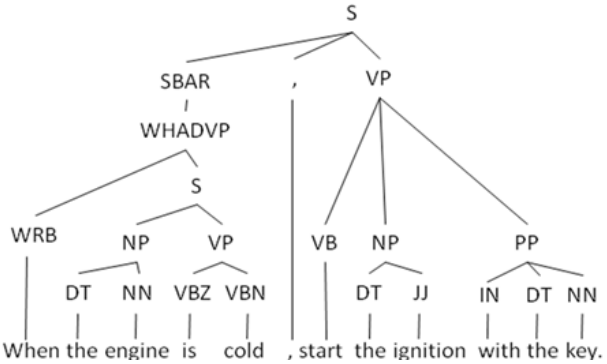

Figure 3. Parsing result of an example sentence using the Stanford Parser. S=sentence; SBAR=subordinate clause; VP=verb phrase; NP=noun phrase; PP=propositional phrase; CC=Coordinating conjunction. For the complete reference of the tagset, see Taylor et al. (2003).

In Figure 3, a sentence (S) is divided into branches of compositional segments, each of which is then further divided into lower level grammatical segments. We extract *verb phrases (VP)* from the first level branch of the parse tree as expressions of core activities, denoted by *VP-expr*. Additionally, each *VP-expr* may have associated contextual information, such as the SBAR ("When the engine is cold") segment in Figure 3. These segments are also extracted with respect to their *VP-expr*, and we define these expressions as "*VP contextual expression*", denoted by *VP-cntxt*. Other types of grammatical segments that may serve as *VP-cntxt* include prepositional phrases (PP), adverb phrases (ADVP), simple declarative clauses (S), and noun phrases (NP) in case the concerning verb phrase has an actor. Figure 4 shows some examples.

<PP>For safety reasons</PP >, disable the front airbag system by…
<S>To enable debugging mode</S>, hit "Ctrl + F9".

Figure 4. Examples of *VP-cntxt*. All parsing results are obtained using the Stanford Parser.

Note that *VP-cntxt* may be at the sibling level of a *VP-expr* (SBAR in Figure 3), or as branch of a *VP-expr* (PP in Figure 3). Apparently, the examples show that *VP-expr* and *VP-cntxt* contain lexical information that can be processed to identify the structured elements of a procedure. For this purpose, *VP-expr* and associated *VP-cntxt* are submitted to the next extraction process, which is a finite-state grammar that makes use of the resulting parse tree.

The finite-state extraction grammar consists of sets of rules that exploit the grammatical stereotypes discovered in earlier studies. For example, an SBAR that begins with "if" or "when" often indicates an expression of conditions; a sentence that starts with "to" followed by a

verb phrase often indicates an expression of a purpose. We refer to these words as *markers*. The extraction grammar searches for markers within *VP-expr* and *VP-cntxt* using regular expression patterns. If a marker is found, the segment is further analysed grammatically to classify the expressions into one of the elements in Figure 1, and extract structured contents. Figure 5a and 5b show examples of simplified extraction grammar for *VP-expr* and *VP-cntxt*.

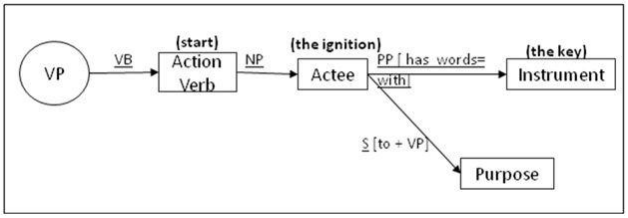

Figure 5a. Example extraction grammar for *VP-expr* and the *VP-cntxt* it contains. Rectangles represent components defined in Figure 1; underlines represent segments in the parse tree; [square brackets] represent a pattern to be fulfilled; (round brackets) represent words and phrases from sentences; bold texts correspond to the texts shown in Figure 3.

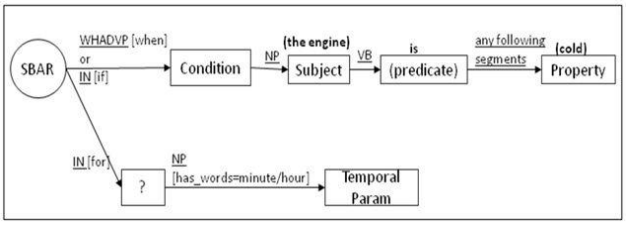

Figure 5b. Example extraction grammar for *VP-cntxt* using same notations as in Figure 5a.

Although our method for procedural elements extraction is based on the similar principles as Brasser and Linden (2002) and Paris et al. (2002), the essential difference is that it relies on syntactic parsing and analyses texts based on constituents of a parsing tree while Brasser and Linden, and Paris et al. employed WordNet to classify verbs and nouns and grammatical heuristics to recognise procedural elements at per-token level. The advantages of the parsing-based method are two-fold. Firstly, different parsers built for particular domains can be easily plugged in to cope with varying vocabularies (especially verbs) across domains. Secondly, parsers based on statistical models generally have high accuracy in grammatical analysis, which improves the overall accuracy of procedural analysis. General purpose statistical parsers are usually available for many different languages, such as English, Spanish and Portuguese.

Rule-based reasoning is applied to resolve omissions and ambiguous references in instructions and populate corresponding procedural components:

- Missing actees, arguments of actions.
- Missing subjects in conditions.
- Ambiguous references like “it”, “them”.

We adopt a simple rule that makes inference based on the context of a step. If an actee is missing, or denoted by a reference word, the rule simply assigns the actee from the previous step; similarly, for subject of pre- and post-conditions, the rule simply assigns the actee from the main step that the condition is associated with. For example, the second instruction in Figure 6 will be assigned the actee “chicken” found in the first instruction to form a complete step of a procedure.

1. Season the chicken with salt and thyme.
2. Griddle for 10 minutes.

Figure 6. Example instruction with missing *actee*

6. Evaluation and Discussion

This section describes experiments for evaluating the automatic procedure extraction. As mentioned before, there are no public data available for the studies of procedural knowledge acquisition. Therefore, we firstly describe the corpus preparation process, in which we gather instructions from three domains, and create gold standard annotations according to the proposed representation; next, we explain the methodology for evaluating the automatic procedural knowledge extractor and show results.

6.1 Data preparation

We collected 30 cooking recipes from the BBC recipe website[1], 30 car maintenance instructions from Fiat car manual [2] and eHow.com, and 14 example procedure descriptions from the A380 maintenance manual[3]. The instructions in the Aircraft Manufacturing domain are more complicated since they often involve multiple sub-procedures. For instructions with sub-procedures, we separated the descriptions of the sub-procedures from their parent procedures, thus transforming the 14 examples into 25 instructions. On average, each instruction contains 5.3 statements, each containing an average of 13 tokens (i.e., words and symbols separated by spaces). For the kitchen and the car domains, on average, each instruction contains 9 statements, each containing an average of 12 tokens.

In order to evaluate the automatic extraction system, we ask human annotators to create a gold standard against which the automatically generated content is compared. Since the system automatically segments instructions and classifies them into the components defined in the semantic representation, human annotators are requested to do the same by annotating the segments using an annotation tool. For each domain, three annotators are invited to perform the task, and a subset (25%) of the corpus is used for studying Inter-Annotator-Agreement following the approach in Hripcsak and Rothschild (2006).

[1] http://www.bbc.co.uk/food/recipes/

[2] http://aftersales.fiat.com/elum/SelectModel.aspx?mark ID=1&lan guageID=1

[3] Due to data distribution policies, this part of the data is unavailable to the public.

Essentially, this is similar to the analyses in previous studies such as Kosseim and Lapalme (1998), Brasser and Linden (2002), Aouladomar (2005) and Gil (2010), who classified segments of instructions according to their semantics and studied their statistical distribution. However, as mentioned before, none of these have made their corpora or annotations available, nor studied the difficulty of annotation in the process. In fact, as indicated by the IAA for annotating procedural elements in Table 1, annotating instructions is a non-trivial task. Even human annotators have found it difficult to annotate certain procedural elements, particularly instrument and spatial parameter.

Procedural component	Total in the corpora	Sampled average IAA
Condition (pre, post)	113	0.82
Purpose	75	0.82
Instrument	84	0.78
ActionVerb	751	0.95
Actee	732	0.89
TemporalParam	51	0.96
QuantitativeParam	24	0.89
SpatialParam	149	0.81

Table 1. Data statistics and IAA

Examples of ambiguities include "Brush the chicken with olive oil", for which one annotator considers "chicken" and "olive oil" both being the actee of the action verb "brush", while another treats "olive oil" as instrument. Another typical example is "Heat the oil on a griddle pan over a high heat", for which annotators disagree on annotating instrument (griddle pan v.s. high heat or both). These have shown that analysing and annotating instructions is not an easy task, and in many cases, cannot be accomplished by simple pattern matching as suggested in previous works. On the contrary, annotators apply their background knowledge to make inferences and reasoning, which will cause inter-annotator disagreement. There is no clear difference in IAA between the three domains, which possibly suggests that the representational model and their distribution in corpus are generic.

To our knowledge, this is the first work that has reported an analysis of the difficulty in annotating instructions. The corpora are available from:
http://staffwww.dcs.shef.ac.uk/people/Z.Zhang/resources/instructionAnnotation.zip.

6.1 Results and discussion

The data are used to evaluate the output produced by the finite-state extraction grammar. The results are shown in Table 2. In addition, as discussed before, the only other study that has reported a similar evaluation is Paris et al. (2002). The authors used a corpus of 9 instructions from the software documentation domain. As a comparison, we also summarise the evaluation in the rightmost column in Table 1. Although we were unable to obtain the corpus for a truly comparative evaluation, we believe the results are still useful reference.

Procedural component	F1	Paris et al. 2002
Condition (pre, post)	76.4	54.5
Purpose	76.1	30.2
Instrument	68.3	-
ActionVerb	89.4	77.6[4]
Actee	86.2	
TemporalParam	97.3	-
QuantitativeParam	84.1	-
SpatialParam	73.2	-

Table 2. Experiment Results

As shown in Table 2, our method in general has achieved high accuracy in this task. Compared against the results by Paris et al. (2002), the proposed extraction method achieved significant improvement in accuracy. We believe this is due to the robustness of syntactic parsing and the extraction grammar built on top of it. It is particularly difficult to extract instances of instrument and spatial parameter. Careful analysis show that a large proportion of errors are due to miss-alignment between the extraction patterns and their semantics. For instance, in examples like "fry … in a pan" and "blend … in a food processor", "pan" and "food processor" have been classified as spatial parameter due to the marker word "in". In some cases, the extraction patterns are insufficient. For example, in "spread the butter, followed by ketchup", only "butter" is recognized as actee. This caused a reduction in recall, and more complex patterns are required to handle such situations. Meanwhile, special phrases and terms have caused errors in syntactic parsing. For example, phrases such as "stir fry", "pan fry", "pour in", "scatter in" and "throw in" should be treated as verb phrases; instead, they are parsed as verbs followed by noun phrases, or prepositional phrases. These observations show that while instructional language does exhibit certain structural and lexical patterns, it is not always straightforward to apply these patterns in language analysis and procedural knowledge acquisition. An ideal system should be robust enough to account additional features and deal with exceptions. We also found no significant difference in the accuracies obtained from different domains, which suggests that the proposed representation and the extraction system are generic.

To study the accuracy of the rule-based reasoning, we

[4] Paris et al. (2002) reported 77.6 for "task element", which is in general equivalent to ActionVerb plus Actee in our model.

manually analysed the output against the natural language instructions. Out of all resolved omissions and ambiguous references, the simple rules have achieved an average of 57% accuracy across the three domains. A large proportion of errors are due to special verb phrases composed of a verb followed by a preposition, which have been treated as missing verb arguments. Examples of such include "pour in the water" and "scatter in the leaves".

7. Conclusion

This paper introduced an approach and an implemented solution for automatic procedural knowledge acquisition from natural language instructions. A generic representation of procedural knowledge has been proposed to support various tasks with procedural knowledge. An automatic extraction system built on natural language processing techniques processes instructional texts and generates procedural knowledge according to the proposed representation. Meanwhile, certain omissions and ambiguous references in the instructions are resolved by simple rules.

To verify this approach and evaluate the implementation, we firstly carried out corpus annotation exercise to create public, re-usable annotations for instructions. It has been shown that despite the regularities in instructional language discovered in the previous studies, annotating instructions remains a challenging task. Annotators apply their background knowledge and reasoning in annotation, which has caused inter-annotator-disagreement. We suggest that future research address the instruction annotation process with more transparency, to study the difficulty of such tasks.

Using these annotations as gold standard, the automatic extraction system based on extraction grammar has been evaluated in three domains and obtained generally good accuracies, which significantly outperformed a similar previous work. This suggests that the proposed representation and the automatic extraction method are both generic, and can be applied or adapted to other domains for the problem of procedural knowledge acquisition. However, results are less satisfactory on instrument and spatial parameter, where IAA is the lowest. This shows that, despite the stereotypical patterns discovered in instructional languages, there is a sufficient level of exceptions that invalidate the method. Pattern based system may suffer from insufficient coverage, and less flexibility in coping with such exceptions. Instead, a more robust approach, such as statistical machine learning, may bring further improvement. Although this type of method requires a large amount of training data (i.e., in the form of annotated instructional texts), it is well known for its robustness in learning different models tailored for diverse data. This will be explored in the future. Concerning the proposed semantic representation, future work will aim to incorporate more complex procedural relations and controls, such as concurrency and alternatives. Attention will also be paid to supportive information for procedures, such as precautions and warnings. This kind of information is found to be very important in, e.g., the Aircraft Manufacturing domain. Often, the precaution notes include details of how a step of a procedure should be carried out and things to avoid. While these are not strictly part of a procedure, they comprise very important information that would ideally be captured. An adequate representation model should also capture such information.

8. References

Aouladomar, F. 2005. A preliminary analysis of the discursive and rhetorical structure of procedural texts. In Symposium on the Exploration and Modeling of Meaning

Bielsa, S., Donnell, M. 2002. Semantic functions in instructional texts: a comparison between English and Spanish. In Proceedings of the 2nd International Contrastive Linguistics Conference, p.723-732

Brasser, M., Linden, K. 2002. Automatically eliciting task models from written task narratives. Fourth International Conference on Computer-Aided Design of User Interfaces

Delpech, E., Saint-Dizier, P. 2008. Investigating the Structure of Procedural Texts for Answering How-to Questions, in LREC2008

De Rijke, M. 2005. Question Answering: What's Next? In the 6th International Workshop on Computational Semantics

Frantzi, K., Ananiadou, S. Tsujii, J. 1998. The C-value/NC-value method of automatic recognition for multi-word terms. Proceedings of the Second European Conference on Research and Advanced Technology for Digital Librarie

Fritz, G., Gil, Y. 2011. A formal framework for combining natural instruction and demonstration for end-user programming. In Proceedings of the 16th international conference on intelligent user interfaces

Gil, Y. 2010. Human Tutorial Instruction in the Raw. Submitted for publication, Available from http://www.isi.edu/~gil/gil-ker10.pdf.

Gil, Y., Ratnakar, V., Fritz, C. 2011. TellMe: Learning procedures from tutorial instruction. In Proceedings of the 16th international conference on intelligent user interfaces.

Hripcsak, G., Rothschild, A. 2005. Agreement, the F-measure and reliability in information retrieval: In Journal of the American Medical Informatics Association, 296-298.

Kosseim, L., Lapalme, G. 1998. Choosing Rhetorical Structures to Plan Instructional Texts. In Computational Intelligence, vol.16 (3), pp. 1-38.

Muehlen, M. 2001. Workflow based process controlling – or: what you can measure you can see. In Layna Fischer (Ed.): Workflow Handbook 2001. Future Strategies, Lighthouse Point, FL2001, pp. 61-77

Mühlhäuser, M. (2008). Smart Products: an introduction. In: Constructing Ambient Intelligence - AmI 2007 Workshops, pp. 158-164, Springer Verlag

Murdock, V., Kelly, D., Croft, W., Belkin, N., Yuan, X. 2007. Identifying and improving retrieval for

procedural questions. In Information Processing & Management, 43 (1), pp. 181-203
Paris, C., Colineau, N., Lu, S., Linden, K. 2005. Automatically generating effective online help. International Journal on E-learning, 4:83-103
Paris, C., Linden, K., Lu, S. 2002. Automated knowledge acquisition for instructional text generation. In Proceedings of SIGDOC'02.
Taylor, A., Marcus, M., Santorini, B. 2003. The Penn Treebank: An Overview. In Abeillé (2003)
Tang, J., Hwang, S. 1996. Handling uncertainties in workflow applications. In Proceedings of CIKM'96
Zhang. Z., Uren, V., Ciravegna, F. (2010). Position paper: A comprehensive solution to procedural knowledge acquisition using information extraction. In Proceedings of KDIR2010, Valencia, 2010.