Information Systems 40 (2014) 153–167

Contents lists available at ScienceDirect

Information Systems

journal homepage: www.elsevier.com/locate/infosys

Automatic case acquisition from texts for process-oriented case-based reasoning

Valmi Dufour-Lussier [a,b,c,*], Florence Le Ber [d], Jean Lieber [a,b,c], Emmanuel Nauer [a,b,c]

[a] Université de Lorraine, LORIA, UMR 7503 — Vandœuvre-lès-Nancy F-54506, France
[b] CNRS, LORIA, UMR 7503 — Vandœuvre-lès-Nancy F-54506, France
[c] Inria — Villers-lès-Nancy F-54602, France
[d] ICUBE Université de Strasbourg/ENGEES, CNRS —Strasbourg F-67000, France

ARTICLE INFO

Available online 20 December 2012

Keywords:
Automatic case acquisition
Information extraction
Natural language processing
Process-oriented case-based reasoning
Textual case-based reasoning
Workflow generation

ABSTRACT

This paper introduces a method for the automatic acquisition of a rich case representation from free text for process-oriented case-based reasoning. Case engineering is among the most complicated and costly tasks in implementing a case-based reasoning system. This is especially so for process-oriented case-based reasoning, where more expressive case representations are generally used and, in our opinion, actually required for satisfactory case adaptation. In this context, the ability to acquire cases automatically from procedural texts is a major step forward in order to reason on processes. We therefore detail a methodology that makes case acquisition from processes described as free text possible, with special attention given to assembly instruction texts. This methodology extends the techniques we used to extract actions from cooking recipes. We argue that techniques taken from natural language processing are required for this task, and that they give satisfactory results. An evaluation based on our implemented prototype extracting workflows from recipe texts is provided.

1. Introduction

This paper introduces a method for the automatic acquisition of a rich case representation from free text for process-oriented case-based reasoning. Case-based reasoning (CBR) [1] is a reasoning paradigm used to solve problems by retrieving similar problem–solution pairs from a case base and adapting the solutions. Because case engineering is a complicated and costly process, automatic case acquisition from text has been a major focus of research in the last 15 years. Methods used in textual CBR vary widely in their scope and their use of natural language processing (NLP) methods, depending on the intended applications. At one end of the spectrum, a text is seen as a bag-of-words and is mostly represented as a term vector. Weights are assigned according to the relative frequency of the terms in a text versus its frequency in all the texts of the case base, possibly taking synonymy into account. This method does not allow for sophisticated inferences, such as adaptation of highly structured cases, taking into account domain knowledge. At the other end, Gupta and Aha [2] propose using a full-fledged NLP solution to translate free text into predicate logic. While such a system would indeed allow for very sophisticated adaptation (including, if combined with a natural language generation system, unlimited possibilities in terms of textual adaptation), existing NLP systems translating from text to logic are far from being accurate.

In this paper, we argue in favour of using limited NLP techniques to extract, from procedural texts, the verbs, their complements and their relevant modifiers, in order

* Corresponding author at: Université de Lorraine, LORIA, UMR 7503 — Vandœuvre-lès-Nancy F-54506, France. Tel.: +33 3 54 95 86 43; fax: +33 9 57 92 62 82.
E-mail addresses: valmi.dufour@loria.fr (Dufour-Lussier), florence.leber@engees.unistra.fr (F. Le Ber), jean.lieber@loria.fr (J. Lieber), emmanuel.nauer@loria.fr (E. Nauer).

0306-4379/$ - see front matter © 2012 Elsevier Ltd. All rights reserved.
http://dx.doi.org/10.1016/j.is.2012.11.014

to acquire rich case representations for processes. We are more open to linguistic-inspired techniques than approaches to textual CBR based on information retrieval and even information extraction. By limiting ourselves to procedural texts (an important but not very varied type of text) and trying to extract only domain-relevant meaning from it, we are able to achieve encouraging results. The approach presented herein is a generalisation of our previous work in defining a case representation for recipes and implementing software to acquire cases automatically from a recipe book [3,41]. This domain shows numerous examples where case representations richer than a vector space and NLP are a necessity.

Some background knowledge is required to understand certain choices made in designing this extraction process. This is introduced briefly in Sections 2 and 3. Section 2 presents two formalisms that are used in process case representations: workflows and qualitative algebras. Some shortcomings of both are pointed out, which justifies making sure that the proposed method can extract both workflows and qualitative constraints from text. Additionally, two running examples are introduced: a cooking recipe and a scientific experimental protocol. Section 3 presents different usages that are expected for the case representation, and shows which effect they have on the extraction process. More specifically, two approaches for textual case adaptation are presented: one based on "grafting", and one based on belief revision theory.

Section 4 is the core of the paper. It describes in detail the process through which case representations are extracted from texts. A text is analysed in many passes. All but the last we use implement fairly typical NLP methods, and are detailed in Section 4.1. The last one, introduced in Section 4.2, is an anaphoric reference solver that we designed specifically for procedural texts. Section 4.3 explains how a structured case representation is engineered from the information extracted in the previous steps. This is sufficient to allow for adaptation of the formalised cases, but additional concerns that must be kept it mind to facilitate the reuse of the actual text are introduced in Section 4.4. All the examples in this section come from recipes, but we show in Section 4.5 that the method presented is applicable to different types of texts by applying them to the scientific protocol.

Section 5 presents a first evaluation of the extraction process by comparing the case representations obtained automatically against a gold standard. Sections 6 and 7 finally discuss related and future work.

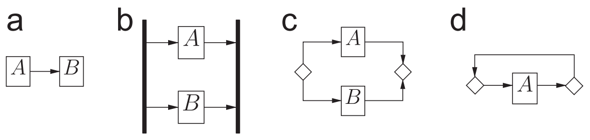

Fig. 1. Workflow patterns: (a) sequence; (b) conjunction; (c) disjunction; and (d) loop.

2. Formalisms for process representation

The most important method used to model cases representing processes is workflows. For instance, all the papers presented at the first Process-oriented Case-based Reasoning workshop [4] were based on workflows, except two, that used workflow-like graphs. Workflows efficiently model the actions required to complete a process: a workflow defines a partial order over the actions and offers the possibility to express options (disjunctions), simultaneity (conjunctions) and repetition of actions (loops). There exist different formal languages in which workflow knowledge can be represented. In this paper, UML Activity Diagrams are used.

The most basic control flow structure is the sequence (shown in Fig. 1a), which means that a task B is ready to begin execution as soon as a task A is finished. Other control statements are the fork, indicating the concurrent execution of workflows, the join, synchronising them, the decision, leading to the exclusive execution of one workflow from a set, and the merge, ending such a conditional execution [5]. The fork and join control statements are used to create a conjunction control structure (Fig. 1b), while the decision and merge statements can be used to create either a disjunction (Fig. 1c) or a loop structure (Fig. 1d).

Workflows have some limitations, apparent in the representation of the biochemistry procedure shown in Fig. 2. Temporal aspects that are important in certain

1. Suspend evenly 1 kg. of yeast in 4 liters of 0.5 M Na_2HPO_4 and boil for 3 hours.
2. Cool to 37 °and add 0.5 gm. of trypsin (Wilson 1:250) on the 1st, 3rd, 6th, and 10th day of incubation at 37 °. Add 5 ml. of toluene on the 1st day, and 2.5 ml. of toluene on the 3rd, 7th, 11th, and 15th day of incubation. Incubate for 16 days at 37 °, with occasional stirring, and adjust pH daily to 7.8–8.0 by the careful addition of NaOH.
3. Collect precipitate by centrifugation. Discard supernatant.
   [. . .]
7. The precipitate from step 6 is added rapidly with vigorous stirring:–
   (a) to 8 liters of *absolute ethyl alcohol*; stir for 1 hour; centrifuge.
   (b) The precipitate is added to 4 liters of *absolute ethyl alcohol* and treated as in (step a).
   (c) The residue is dried *in vacuo*.
   (d) The dried powder is refluxed for 3 hours with 500 ml. of *absolute ethyl alcohol.*
   (e) Centrifuge and dry immediately *in vacuo* and store *in vacuo*.

Fig. 2. Excerpts from a scientific experimental protocol for preparing zymosan [6].