Computer and Information Science; Vol. 8, No. 4; 2015
ISSN 1913-8989 E-ISSN 1913-8997
Published by Canadian Center of Science and Education

A Smart Algorithm for USE-Cases Production Based on Name Entity Recognition

Rafeeq Al Hashemi[1], Moha'med Al-Jaafreh[2], Tahseen Al-Ramadin[3] & Ayman AL Dmour[3]

[1] Graduate School of Management, Robert Morris University, Greater Chicago Area, United States

[2] Department of Software Engineering, Al-Hussein Bin Talal University, Jordan

[3] Department of Computer Information Systems, Al-Hussein Bin Talal University, Jordan

Correspondence: Ayman AL Dmour, Department of Computer Information Systems, Al-Hussein Bin Talal University, Jordan. Tel: 962-799-375-581. E-mail: ayman70jo@yahoo.com

Received: September 12, 2015 Accepted: October 12, 2015 Online Published: November 6, 2015

doi:10.5539/cis.v8n4p51 URL: http://dx.doi.org/10.5539/cis.v8n4p51

Abstract

Use case analysis is a common technique used to ascertain the functional requirements of a software system. A use case diagram is a kind of Unified Modeling Language (UML) diagram created for use case analysis. Creating effective use cases can be a determining factor in building a system that meets users' needs. However, writing use cases is a difficult and time-consuming process, requiring the user to manually fill out a form or write text in a specific, pre-stipulated format. Many students lack the technical knowhow to do this. Our research offers a software solution that resolves this issue. By combining natural language algorithms, such as Part Of Speech (POS) and Name Entity recognition (NE), with a set of grammatical rules created and implemented as a Finite State Machine (FSM), our system extracts the relevant items from the text and automatically translates the plain or unstructured text into a structured one. The paper has been tested on standard examples with excellent results.

Keywords: use cases, information extraction, name entity recognition, finite state machine

1. Introduction

The use case model is composed of two parts: The use case diagrams and the use case specifications. The use case model comprises actors, use cases, and associations, all of which are depicted in a use case diagram. According to (A. Cockburn, 2000), each use case represents a major piece of functionality that is complete from beginning to end, and is described within a use case specification. The specification includes the following processes: the basic flow of the use case, the alternative flows, involved actors and stakeholders, conditions, and references to other related use cases. The business rules associated with use case interactions must be specified or at least indicated by these rules (F. Dias, A. Schmitz, M. Campos, A. Correa, A. Alencar, 2008).

Even experienced analysts find it very challenging to generate use case models based on analysis of the functional requirements of a natural language. There are numerous issues that give rise to this difficulty: complexity, imprecision, and ambiguity of natural language; combined with a lack of domain knowledge, object oriented experiences, and effective object-oriented methods. More research is needed in the area of methodologies and CASE tools.

Because use case specifications remain largely textual, ambiguities are predictable (B. Dano, H.Briand, and F.Barbier, 1997). According to (R. Denney, 2005) and (D. Jagielska, P. Wernick, M. Wood, and S. Bennett, 2006), the resulting descriptions usually suffer from problems such as redundancies, ambiguities, inconsistencies, conflicts with domain terminology, and implementation details (terminology contaminated use cases). These issues make it difficult for customers to understand and maintain the specifications.(El-Attar, Miller, 2006 ; El-Attar, 2009; El-Attar, Miller, 2012) state that these issues result in low quality information systems.

One of the causes of imprecision and ambiguity in use case specifications is the unrestricted textual nature of the specifications. The elicitation process is a major determinant in the quality of the use case models. (J. Kim, P. Sooyong, and S. Vijayan, 2004) state that the lack of support for a systematic requirements elicitation process is probably one of the main drawbacks of use-case-driven analysis. This lack of elicitation guidance sometimes results in an ad-hoc set of use cases without a consistent underlying rationale.

Several existing approaches use natural-language parsing techniques to retrieve the use case elements from pre-existing documents. This method is based on either a predefined SRS template or a use case template. However, this process suffers from ambiguities, redundancies and inconsistencies present in such documents, and is therefore unreliable.

Because of the above-mentioned limitations, the analyst must be an expert to obtain the use case elements correctly and completely from existing documents. Irrespective of the analyst's ability, this procedure is extremely time consuming. One approach involves developing use case models from scratch, by using the classical methodology of open-ended questions that lack specificity and formality (V. Gervasi and D. Zowghi, 2005). The result of this method is a document with ill-defined requirements that needs to be reorganized and readjusted, in order for the analyst to efficiently derive the use case elements. Other approaches, such as natural language parsing and open questions, are generally applied in re-engineering to derive textual requirements. These approaches also result in ill-defined requirements, as they have the same weaknesses mentioned above.

Informality is the principal difficulty with the use of automated tools for use case modeling. Informal natural language is inherently complex, fuzzy and ambiguous. As a result, use case elements are difficult to identify completely and correctly. In sum, there is a scarcity of approaches that automatically generate use case models.

Use case model schemas are sometimes used to normalize the use case specifications. A use case processing method is introduced for use case realization. The association relationships between formalized action statements and behavior types are identified and applied, in order to automate the use-case analysis.

(Hasan Al-Shalabi and colleges Hasan Al Shalabi, Rafeeq Al-Hashemi and Tahseen A. Al-Ramadin, 2013) proposed an Automatic Cover Letter Generator System from CVs (ACLGS) that solved the problem of writing a C.V. Cover letters require some linguistic skills and some experience in this domain. By building a template as a frame of slots, with each slot containing a required skill for the job, the ACLGS extracts the required information from the user C.V and fills the slots automatically.

ACLGS then applies information retrieval methodologies to extract information with intelligence trends. Using this information, it mines the user C.V. in terms of parts-of-speech tags and some indicator words that the system uses to recognize the proper data and required information. In addition, the system specifies a set of features for each slot in the form.

2. Methodology

Our paper examines a new field of information extraction, one that utilizes the power of the computer to perform one of the most critical tools of use cases for system analysis. The proposed method allows the computer to automatically generate accurate and complete use cases, thereby eliminating the need to do it manually. This research was implemented on plain text documents, using standard examples of this field, both from the Internet and from books, such as Summerfield's. The proposed system uses the Rapid Miner application, which helps complete the research with fewer resources and in less time than the classical method. Our new method also works well with the most challenging aspect of the problem, unstructured data.

The proposed Algorithm is illustrated in figure (1). In the beginning, the algorithm converts the unstructured text into structured tables that extract the meaningful terms and objects. This is achieved through the following sequence of steps: preprocessing to filter the entered text; tokenizing the text; stop-word removal, and word stemming. The next step is to identify the Part-of-Speech tag for each word, in order to discriminate the hot keywords.

The Porter stemming algorithm (or 'Porter stemmer') is the term used in linguistic morphology, it is a process for removing the commoner morphological , suffix and prefix from the word, then by changing some part of the words (phase to phase) to get the root (word stem) of the words. In literature, many stemming algorithm are used. However, we employ Porter stemmer, because it works well with our algorithm and offers excellent results. About 500 documents and examples were selected from the database and analyzed manually to identify the keywords and sentences that play the primary role in building use cases. We then initiated a set of grammar rules that covers all of these sentences and keywords. Figure 2 illustrates the block diagram of the proposed method.

Algorithm: *Use-Case Production*

Input: *Document*

Output: *Use Case Template:*

1. *Preprocessing*
   a. *Tokenization*
   b. *Stop words filtering*
   c. *P.O.S. Tagging*
   d. *Stemming*
2. *Implementing Regular Expressions for Information Extraction*

Figure 1. Use-Cases Production Algorithm

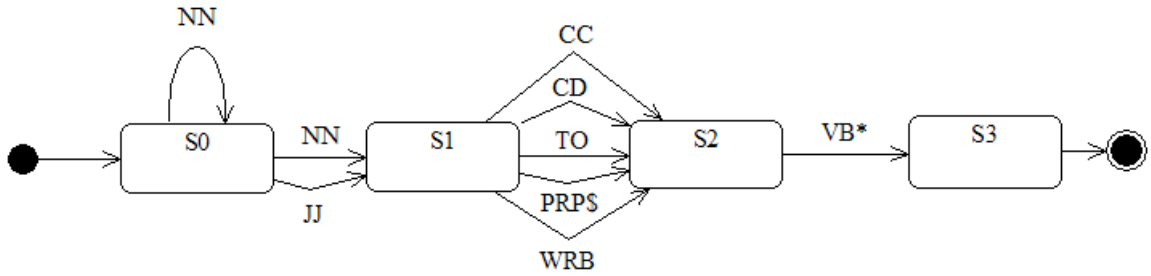

Figure 2. Finite state machine for the extracting rules

In this paper we used Name Entity Recognition, which significantly improved the accuracy of the algorithm by helping to identify the relationship between entities. Name Entity and Relation Entity Recognition are new features added to the grammar rules set (Carreras, X., Marquez, L., & Padro, L., 2003; Nadeau, D., Sekine, S., 2007). These two features extract items such as Names, Actors, Processes, and Relations between the Actors and the system. The use of these extracted objects enables the computer to produce the use case diagram automatically. This additional step yielded excellent results and added significant value to the algorithm.

Finally, a Preprocessing step added to this method also produces improved results. This step is achieved by employing a lookup dictionary that we created using a set of the most closely related keywords frequently employed in the use case diagram. This is an additional feature that we use to identify related keywords, Actors, Relations and the System Name.

3. Results and Discussion

The algorithm was implemented on standard scenarios, and it yielded more accurate results when compared to manually extracted sentences used in the use cases. The following is an example of the results that were achieved when testing the method.

Case Study:

Seats can be reserved by customers on the website of a bus company. The customer has the option to directly pay for the seat through the website. In that case, the seat cannot be canceled (neither by the customer nor by the bus company). If the customer has not paid for the seat, the bus company can cancel the seat, in the event that the customer does not show up one hour before the trip. When the reservation is canceled, the seat will become free and can then be sold to another customer. Both the customer and the company staff must authenticate themselves before performing operations within the system.

The algorithm implemented in this paper analyzes the entered unstructured text and extracts the most important sentences matching the set of features established by the algorithm.

i) Seats can be reserved by customers.

1. ➔(by part of speech) Seats/NNS can/MD be/VB reserved/VBN by/IN customers/NNS
2. ➔(Delete stop words) delete " can/MD" and" by/IN"
3. ➔ (follow state chart) S0(NN)➔S1(VB*)➔S3(NN)➔S4.

4. NN➔ Actor1" Seats", VB* ➔ use case(1)" be reserved" , NN ➔Actor2 " customers".

ii) The customer has the option to directly pay for the seat.

1. ➔(by part of speech) The/DT customer/NN has/VBZ the/DT option/NN to/TO directly/RB pay/VB for/IN the/DT seat/NN./.
2. ➔(Delete stop words) delete "The/DT", "to/TO", "for/IN", the/DT", " through/IN" and "the/DT".
3. ➔ (follow state chart) S0(NN)➔ S0(VBZ)➔ S0(NN)➔ S1(TO)➔ S1(RB)➔ S2(VB*)➔S3(NN)➔S4.
4. **NN➔ Actor1" Customer", VB* ➔ use case(2)" pay" , NN ➔Actor2 " seat".**

iii) the reservation is cancelled,

1. ➔(by part of speech) the/DT reservation/NN is/VBZ cancelled/VBN
2. ➔(Delete stop words) delete "The/DT".
3. (follow state chart) S0(NN)➔ S0(VBZ)➔ S1(VB*) ➔ S3.
4. **NN➔ Actor1" reservation", VB* ➔ use case(3)" Cancelled".**

iv) Both the customer and the company staff must authenticate themselves.

1. ➔(by part of speech) Both/PDT the/DT customer/NN and/CC the/DT company/NN staff/NN must/MD authenticate/VBP themselves/PRP.
2. ➔(Delete stop words) delete "Both/PDT", "the/DT" must/MD and themselves/PRP.
3. (follow state chart) S0(NN)➔ S0(CC)➔ S0(NN)➔ S0(NN)➔ S1(VB*)➔ S3.
4. **NN➔ Actor1" Customer", company staff, VB* ➔ use case(3)" Cancel".**

The output will be as follows :

1. NN➔ Actor1" Seats", VB* ➔ use case(1)" be reserved" , NN ➔Actor2 " customers".
2. NN➔ Actor1" Customer", VB* ➔ use case(2)" pay" , NN ➔Actor2 " seat".
3. NN➔ Actor1" reservation", VB* ➔ use case(3)" Cancelled".
4. NN➔ Actor1" Customer", company staff, VB* ➔ use case(3)" Cancel".

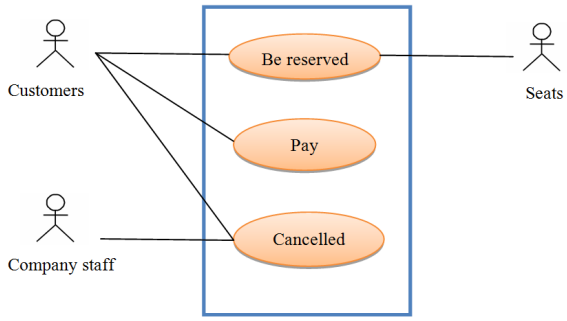

Figure 3. A bus company Use case

4. Conclusion

One important step in system analysis is building the use cases. This is a difficult process requiring considerable time and cost, as the user must manually fill out a form or write the text in a specific format with pre-stipulated constrains and limitations. Many students lack the technical capability to design use cases for their projects. In this paper, we produced an algorithm that helps both students and systems designers to overcome this problem, by constructing the use cases automatically. The algorithm builds the use cases by converting plain unstructured text into structured text, through swift and accurate extraction of the most meaningful items in the text.

References

Cockburn, A. (2001). Writing effective use cases. Reading, MA: Addison-Wesley.

Dano, B., Briand, H., & Barbier, F. (1997). A Use Case Driven Requirements Engineering Process. *Requirements Engineering, 2*(2), 79-91.

Carreras, X., Marquez, L., & Padro, L. (2003). A simple named entity extractor using adaboost. In W. Daelemans & M. Osborne (Eds.), Proceedings of the Seventh Conference on Natural Language Learning, Edmonton, Canada (pp. 798-803).

Jagielska, D., Wernick, P., Wood, M., & Bennett, S. (2006). How natural is natural language? How well do computer science students write use cases? Companion to the 21st ACM SIGPLAN Symposium on Object-oriented Programming Systems, Languages, and Applications (OOPSLA '06), (pp. 914-924). New York, NY: ACM Press.

Dias, F., Schmitz, A., Campos, M., Correa, A., & Alencar, A. (2008). Elaboration of use case specifications: an approach based on use case fragments. In: ACM Symposium on Applied Computing (SAC), Fortaleza, Ceara, Brazil, pp. 614-618.

Al-Shalabi, H., Al-Hashemi, R., & Al-Ramadin, T. A. (2013). Automatic Cover Letter Generator System from CVs (ACLGS). *Global Journal of Computer Science and Technology Software & Data Engineering, 13*(3), Version 1.0 Year.

Kim, J., Park, S., & Sugumaran, V. (2006b). Improving use case driven analysis using goal and scenario authoring: A linguistics-based approach. Data & Knowledge Engineering, pp.1-46.

El-Attar, M., & Miller, J. (2006). Matching Ant patterns to Improve the Quality of Use Case Models. The Fourteenth International Requirements Engineering Conference (RE 2006). Minneapolis-St. Paul, U.S.A. September 11-15.

El-Attar, M. (2009). Improving the Quality of Use Case Models and their Utilization in Software Development. Ph.D. Thesis. The University of Alberta. Edmonton, Canada.

El-Attar, M., & Miller, J. (2012). Constructing High Quality Use Case Models: A Systematic Review of Current Practices. *Requirements Engineering, 17*(3), 187-201.

Nadeau, D., & Sekine, S. (2007). A survey of named entity recognition and classification. *Linguistic Investigations, 30*(1), 3–26.

Denney, R. (2005). Succeeding with Use Cases Working Smart to Deliver Quality. Addison-Wesley Professional: Boston 2005.

Gervasi, V., & Zowghi, D. (2005). Reasoning about inconsistencies in natural language requirements. *ACM Transactions on Software Engineering and Methodology (TOSEM), 14,* 277- 330.

Copyrights

Copyright for this article is retained by the author(s), with first publication rights granted to the journal.

This is an open-access article distributed under the terms and conditions of the Creative Commons Attribution license (http://creativecommons.org/licenses/by/3.0/).