Research Journal of Applied Sciences, Engineering and Technology 12(9): 926-932, 2016
DOI:10.19026/rjaset.12.2810
ISSN: 2040-7459; e-ISSN: 2040-7467

Submitted: October 9, 2015 Accepted: October 30, 2015 Published: May 05, 2016

Research Article

Automatic Domain Knowledge Extraction from Requirements Specification Text

[1]S. Geetha, [2]G.S. Anandha Mala and [3]Suresh Kumar Sanampudi
[1]JNTUH, Hyderabad, India
[2]Department of CSE, Easwari Engineering College, Chennai,
[3]Department of IT, JNTU College of Engineering Jagtial, Telangana, India

Abstract: The automated generation of sequence of actions and schema extraction from requirements specification text written in natural language is really challenging and has limited explorations so far. It's an attempt to automatically retrieving some useful elements from the requirements specification text, which is required for the software development process. This study proposes a system to simplify requirements analysis process and identify the interaction between the class objects using Natural Language Processing (NLP) techniques and crafted rules. The Automatic Domain Knowledge Extraction (ADKE) system does the step by step elicitation of classes, attributes, relationships, actors, relational schema, use cases and interaction between the classes. Finally deploys relational schema into the database, which allows user to acquire domain knowledge by applying queries and quick traceability link between the use cases. The experimental results of ADKE system are compared with other available tools and it gives better accuracy also.

Keywords: Key attributes, natural language processing, sequence of actions, relational schema

INTRODUCTION

This study aims at presenting an approach to change the Natural Language Text (NLT) into structured database representation, though transformation of natural language requirements specification text into usable software is challenging. Most of the applications possess the information in the form of NLT, not structured database. Extracting knowledge from NLT has become a vital characteristic of Knowledge Discovery Database (KDD). Application of text mining could result in productive use of NLT. Procedures like structuring of input, extraction of patterns and interpreting the output are discussed in Jekub and Roman (2013). Development of software mainly focuses on apprehending natural language requirements specification text. Software development process is initiated by capturing the requirements; this provides a total understanding of the system giving an overview of tasks to be accomplished. The process of requirements capture demands the requirements to be structured. This in turn leads to the translation of unstructured requirements into a structured one. A number of algorithms have been reported to automate or semi-automate the translation of the natural language requirements specification text into a structured requirements specification. Development of software is made easy as the database representation and knowledge attainment is possible with the application of queries. Having known this, a technique for translating natural language requirements specification text into a relational schema is possible by determining the schema for the tables and the relationship corresponding to all the tables mined from the requirements specification text and sequence of messages shared between the class objects are discussed here.

Various concerns are to be dealt with in devising the relational schema of a database by locating the referential integrity constraints that are enforced on a database. Automated discovery of semantic associations involved in the table schema elements, i.e., primary key and foreign key are exigent. Techniques for translating natural language requirements into relational schema and generation of sequence of actions shared among the objects are discussed in this study. Inferring from the previous observation the primary key and the foreign key for construction of relational schema are identified by the mapping rules and rule sets approaches. The automated formation of the relational schema from natural language requirements specification provides us with a platform to extract the information for specific domain. An automated approach of identifying sequence of action shows the dynamic behavior of the system which maintains the quick traceability link between problem statements in the natural language requirements specification text. This study discusses related work, proposed methodology, experimental results and discussion of the proposed system. Finally concludes the proposed work.

Corresponding Author: S. Geetha, JNTUH, Hyderabad, India, Dr. M.G.R. E & R I, Chennai

LITERATURE REVIEW

Requirements Specification is an essential artifact in the software Industry. Requirements specification specifies the behaviour of the system using the information obtained from the user to develop the software with the help of the natural language. NLP interprets the natural language requirements specification text into processed or structured information to the software development. NLP techniques are used to extract the useful information from requirements specification text. Many researchers have experimented with different methods to achieve the above.

Knowledge about language structure like words, groups of words in sentences, matching the word meaning with sentence meaning, etc., are designed and proved by Mirzanur *et al*. (2009) using NL. The translation of requirements specification into schema generation for structural representation and identifying the sequence of actions is the critical step in the initial stage of software development process in order to bring out the functional view of the scenario. Delisle *et al*. (1999) proposed a semi automatic method to extract the Object Oriented Elements (OOE) which proved better for short inputs. Natural language requirements specification has been analyzed automatically by a variety of tools (Overmyer *et al*., 2001; Harmain and Gaizauskas, 2003; Anandha Mala and Uma, 2006; Hina and Imran, 2011; Bajwa and Choudhary, 2012) to generate the class models and the accuracy provided by these tools are very less.

Kate and Mooney (2010) identified and encoded through card-pyramid parsing which performed all the entities and identified the possible relations between them in a sentence and Xian-Yi *et al*. (2011) worked on extracting the relation and entity, to construct the database from natural language text. Divesh (2010) identified the relationship between the values of a set of attributes and Zhang *et al*. (2012) presented an approach to identify the key attributes within the database, interrelationship between the attributes using statistical measures from synthetic and real datasets, which is a data oriented approach. Sismanis *et al*. (2006) presented the Gordian - a scalable algorithm to extract the keys, including composite keys in large datasets. Adelfio and Samet (2013) extracted the schema from the datasets of spreadsheets and HTML tables. It is important to note that, the above mentioned ideas are focused on extracting the key attributes to construct the schema from datasets and not from the natural language requirements specification. Li (2000) depicted a semi automatic approach to generate a sequence diagram from the use case description and Yue *et al*. (2010) automated the elicitation of sequence diagram from use cases by using natural language parser thus identifying the objects and the interaction among them. Jitendra and Atul (2014) identified the interactions between objects from the specification of use case and not from the NL requirements specification.

The above survey details the extraction of class diagram from the requirements specification text, as specified in Semantic Based Vocabulary Rules (SBVR) using NLP techniques and defines the schema for the database using the real and synthetic data sets. The automatic and semi automatic generation of sequence of actions have rarely been explored by many investigators, shows the dynamic behaviour of the system. The proposed work presents the interaction between the objects, based on the use cases identified from the software requirements specification, constructing the schema and deployed into the database without manual intervention.

PROPOSED METHODOLOGY

The translation of natural language requirements specification into usable software poses quite a challenge. Analyzing the requirements to construct the logical design and sequence of message sharing for a particular course of action is essential step in understanding and working with natural language requirements specification text. The proposed work starts with automatic construction of schema by identifying the classes, attributes, key attributes and relationship between the classes. The extension of work includes automatic identification of sequence of actions to show how objects interact with one another in particular scenario (situation) using crafted rules used in the initial stage of software development process. The system architecture named as ADKE shown in Fig. 1.

Preprocessor: The preprocessor includes sentence splitting, tagging, noun phrase chunking and reference

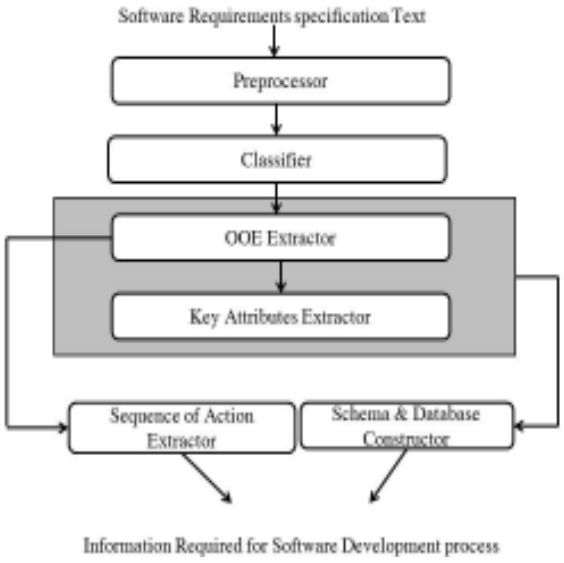

Fig. 1: System architecture for ADKE system

resolving. The input natural language requirements specification is split into sentences through sentence splitter for tagging and then each word in the sentences is tagged by using POS Brill Tagger (Brill, 1992). The nouns that are identified from the tags assigned by the tagger are chunked together depending on their positions with other words by using Ramshaw and Marcus (1995) chunking algorithm. Reference resolver resolves the pronouns to their respective noun phrases based on the antecedent indicators which are calculated for each pronoun by using robust pronoun resolution algorithm (Mitkov, 1998).

Classifier: Classifier normalizes the preprocessed requirements specification text. The tagged, noun phrase chunked and reference resolved input is normalized or simplified into Subject-Verb-Object (SVO) pattern. SVO patterns used to identify the Objected Oriented Elements (OOE) as classes, attributes, actors, methods, use cases and their relationships (Anandha Mala and Uma, 2006).

OOE extractor: The OOE namely classes, attributes, methods, actors, use cases and relationships are identified based on simple rule based approach from the SVO patterns:

- Translating common, proper nouns to classes
- In noun-noun, nouns which are not followed by auxiliary verbs are translated as classes
- Nouns which are followed by auxiliary verbs are translated as class properties in noun - noun pattern
- Translating the lexical verb of a non-personal noun to a method of the first noun in noun-noun pattern
- Translating the lexical verb of a personal noun to a use case of the first noun in noun-noun pattern
- Matching a noun to a personal pronoun as the nouns of previous sentence.

As a result of using above rules, the OOE are extracted from input requirements specification text.

Key attributes extractor: Recognition of key attributes in a table come up with valuable/powerful and error free design. Each row in a table can be identified uniquely by primary key and the relevancy and relationship between tables maintained by using foreign key. The class attributes elicited by OOE extractor are used to construct database table with their attributes.

Identification of primary key attributes: Primary key allows maintaining the uniqueness of every record in the database. The main characteristics of the primary key as stable, unique, indexed and hidden information can be achieved with the help of rules as follows:

- When sentence is in the form of "subject+ auxiliary verb+ adjective+ object", then the object is a key attribute.
- When sentence is in the form of "subject+ auxiliary verb+ object", then the object is a key attribute, if the object is prefixed or suffixed with set of predefined words in the training data set.
- When the sentence is in the form of "subject+ auxiliary verb+object", then the object is a key attribute, if the object is in the training data set.
- When the sentence in the form of "subject+ auxiliary verb+object1+object2", prioritizing is done to the object based on the relevance between object1 and object2. Priority assigned to the objects based on the relevance between, the subject and object1, object2. The highest prioritized object is a key attribute. The sample table attributes taken for identification of the primary key attribute and observations found by applying the above hand crafted rules.

Identification of foreign key attributes: Foreign key references the primary key of the other tables to maintain the relationship between them. The referential actions of a foreign key can be achieved with the following rules as:

- Both foreign key and primary key attribute must be similar
- A table can have multiple foreign keys
- The foreign key must have 1:1, 1: N, M: N cardinality

The primary key in all the tables is referred to identify foreign keys by using above rules.

Schema and database constructor: Understanding the properties of the table and relationship between the tables is very important in data analysis to construct the schema which describes the structure of the tables in a database. Schema can be constructed using table attributes and its relationship extracted from input natural language requirements specification text. So every attribute in tables are associated with data types and range of values. Without violating the referential constraints, data types have to be assigned to all the attributes of all the tables. To assign the data type for the attribute the following procedure is followed as:

- When the attribute is prefixed or suffixed with date, then assign date as data type.
- When the attribute is prefixed or suffixed with no or number or num, then assign number as data type.
- When the attribute is not sticking with previous rule, then assign character as data type.

- When the attributes are key attributes, both the primary and foreign key must have same type based on the category of above mentioned rules

After assigning the data types to attributes of all the tables, then the size of the attribute could be fixed. The size of the attributes for number type as 20 and character type as 30 without violating the referential constraint, because both the keys primary/foreign key must have the same size depends upon the data type.

After extracting the schema from the requirements text is to be converted into intermediate representation as XSD (XML Schema Definition) to construct database. Finally Jackcess a java library file is used to transform XSD structure into the database Microsoft Access including attributes, key attributes, type and size of all the tables represented in schema.

Sequence of actions extractor: The sequence of actions shows messages exchanged between objects to carry out the functionality of particular scenario (situation) and useful in documenting how a future system behaves. The use cases elicited from OOE extractor are analyzed to identify sequence of messages that are communicated between objects. The possible steps are used to create the sequences of actions between objects are:

- Identify the actors and objects who are taking part
- Identify the sequence of interaction
- Describe the messages exchanged between actors and objects
- The actor can send the message to an object
- The object can also send the message to another object

This can be attained with the rules of:

Rule 1: If use cases are scanned in the sentence flow, then apply usecases to generate sequences of messages that are shared between objects, referred as method of classes

Rule 2: If source objects, messages, target objects are identified and then use the messages (methods) to communicate between the objects. The above said rules used to show how objects are interacting with each other in a particular situation.

RESULTS AND DISCUSSION

The main idea of the proposed ADKE system is to identify the elements as classes, attributes, actors, methods relationships, use cases, sequence of actions, construction of database includes the identification of primary and foreign key attributes, data type assigned to all the attributes by using crafted rules and natural language processing techniques to ease the process of software development. The performance of ADKE system is evaluated by measuring the precision to check the retrieved information is relevant and recall checks the completeness of the result.

The results obtained by the ADKE system are compared with results produced by the human experts with the same set of requirements specification text. The performance evaluation of ADKE system for OOE and key attributes extractor samples are shown in Table 1.

The tool was implemented by using java and it was validated using 100 samples of different in document size. The sample result shows better in accuracy, precision and recall of key attribute extractor in Fig. 2.

A case study is discussed from the domain of airline reservation and the problem statement is given as input for the ADKE system to construct database and sequence of actions.

A result of schema constructor which includes key attributes information for airline reservation shown by Fig. 3.

The database schema converted into XML is imported into relational database Microsoft-Access without violating the integrity constraints.

The result produced by ADKE system is compared with other available tools that perform automatic extraction of knowledge from requirement specification text. The recall and precision values of the various tools are compared with ADKE system and the different criteria supported by various tools are also shown in Table 2.

The message communications between the objects are generated by sequence of actions extractor is shown in Table 3.

Table 1: Performance evaluation of OOE and key attributes extractor

	Classes			Attributes			Primary Key		
Doc. No.	Precision	Recall	Accuracy	Precision	Recall	Accuracy	Precision	Recall	Accuracy
Airlines	88.5	100	88.5	100	100	100	80	100	80
Banking	86.5	100	86.5	100	100	100	90.9	100	90.9
Garage	88.9	100	88.9	100	100	100	81.8	90	81.8
Hospital	90.9	100	90.9	100	100	100	88.9	100	88.9
Super market	86.8	100	86.8	100	100	100	84.6	100	84.6
Hotel	89.3	100	89.3	100	100	100	85.7	100	85.7
Kiosk	92.9	97.5	92.9	100	100	100	83.3	100	83.3
Library	88.5	97.9	88.5	100	100	100	84.2	100	84.2
Railway	88.5	100	88.5	100	100	100	87.5	100	87.5
Student	88.9	100	88.9	100	100	100	88.9	100	88.9

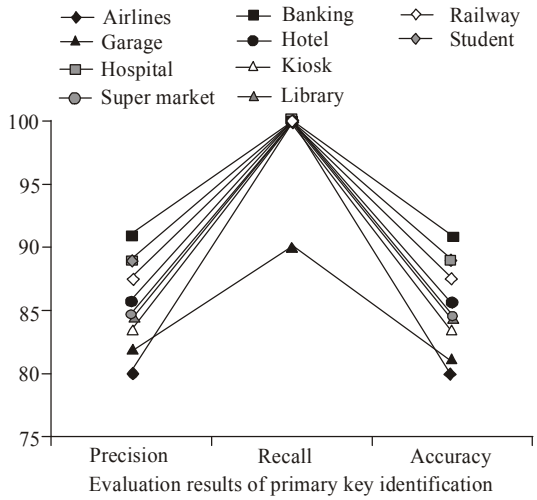

Fig. 2: Results obtained on various documents by key attribute extractor

```
The Structure for all the Tables are as Follows...
**************************************************
Table Name : passenger
**********************
Column Name     - Data Type            - Size  - IntegrityConstraint
= = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = =
  id            - String               - 30    - Primary Key
  name          - String               - 30
  address       - String               - 30
  age           - String               - 30
  ticke_no      - String               - 30
  journey_date  - Date                 - 8
= = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = =

Table Name : details
********************
Column Name     - Data Type            - Size  - IntegrityConstraint
= = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = =

Table Name : ticket
*******************
Column Name     - Data Type            - Size  - IntegrityConstraint
= = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = =
  ticket_no             - String               - 30    - Primary Key
  seat_no               - String               - 30    - Foreign Key(seat)
  flight_no             - String               - 30    - Foreign Key(flight)
  date                  - Date                 - 8
  source                - String               - 30
  destination           - String               - 30
= = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = =
```

Fig. 3: Structure of database generated by schema constructor

Table 2: Comparison of ADKE system with other tools

Criteria	LIDA	CM BUILDER	NL-OOML	SBVR2UML	ADKE
Classes	Yes	Yes	Yes	Yes	Yes
Attributes	Yes	Yes	Yes	Yes	Yes
Key Attributes	No	No	No	No	Yes
Seq. of Actions	No	No	No	No	Yes

Table 3: Results produced by sequence of actions extractor

S. No.	Syn. Str.	Sender	Receiver	Action	Argument
5.1		Passenger	Receptionist	Meets	
6		Receptionist	Passen Ger-details	Gets	
7		Passenger	Details	Gives	
8		Passenger	Ticket	Request	
9		Receptionist	Request	Receives	
10		Receptionist	Aircraft	Checks	
11		Receptionist	Rout	Checks	
12.1		Receptionist	Seat	Checks	
13		Passenger	Ticket	Gets	
14		Receptionist	Ticket	Issues	
15	7	Receptionist	Database	Stores ticket	Ticket
16		Receptionist	Seat	Blocks	
17.1		Receptionist	Ticket	Reserve	
18			Ticket	Cancel	
20.1	12	Receptionist	Seat	Allotted	Passenger
22.1			Seat	Frees	
23.1	12		History	Created	Database

The result of the performance shows that the ADKE system does not miss to identify classes, attributes, methods, use cases and relationships. The system missed out some methods only if the tagger assigns the wrong tag to each word.

CONCLUSION

This work presents an approach to automatically retrieving useful elements from the natural language requirements specification text, which is helpful for the initial stage of software development process in order to improve the accuracy of the machine processing. The ADKE system extracts domain knowledge such as attributes, actors, relationships, relational schema, use cases and interaction between the classes from natural language requirements specification with natural language processing techniques and set of hand crafted rules. The system can be extended to overcome the defectness in the tagger and reference resolver by building a knowledge base which can also increase the effectiveness of generation of system elements.

REFERENCES

Adelfio, M.D. and H. Samet, 2013. Schema extraction for tabular data on the web. Proc. VLDB Endowment, 6(6): 421-432.

Anandha Mala, G.S. and G.V. Uma, 2006. Automatic Construction of Object Oriented Design Models [UML diagrams] from Natural Language Requirements Specification. In: Yang, Q. and G. Webb (Eds.), PRICAI 2006: Trends in Artificial Intelligence. Lecture Notes in Computer Science, Springer-Verlag, Berlin, Heidelberg, 4099: 1155-1159.

Bajwa, I.S. and M.A. Choudhary, 2012. From Natural Language Software Specifications to UML Class Models. In: Zhang, R. *et al*. (Eds.), Enterprise Information Systems. Lecture Notes in Business Information Processing, Springer-Verlag, Berlin, Heidelberg, 102: 224-237.

Brill, E., 1992. A simple rule-based part of speech tagger. Proceeding of the 3rd Conference on Applied Natural Language Processing (ANLC, 1992), pp: 152-155.

Delisle, S., K. Barker and I. Biskri, 1999. Object-Oriented Analysis: Getting Help from Robust Computational Linguistic Tools. In: Friedl, G. and H.C. Mayr (Eds.), Application of Natural Language to Information Systems, Oesterreichische Computer Gesellschaft, Vienna, Austria, pp: 167-172.

Divesh, S., 2010. Schema extraction. Proceeding of the 19th ACM International Conference on Information and Knowledge Management (CIKM'10). ACM, New York, pp: 3-4.

Harmain, H.M. and R. Gaizauskas, 2003. CM-builder: A natural language-based CASE tool for object-oriented analysis. Automat. Softw. Eng., 10(2): 157-181.

Hina, A. and S.B. Imran, 2011. Generating UML class models from SBVR software requirements specifications. Proceeding of the 23rd Benelux Conference on Artificial Intelligence (BNAIC, 2011). Gent, Belgium, pp: 23-32.

Jekub, P. and Y. Roman, 2013. Information Extraction: Past, Present and Future. In: Poibeau, T., H. Saggion, J. Piskorski and R. Yangarber (Eds.), Multi-source Multilingual Information Extraction and Summarization. Theory and Applications of Natural Language Processing, Springer-Verlag, Berlin, Heidelberg, pp: 23-49.

Jitendra, S.T. and G. Atul, 2014. Automatic generation of sequence diagram from use case specification. Proceeding of the 7th India Software Engineering Conference, New York, USA.

Kate, R.J. and R.J. Mooney, 2010. Joint entity and relation extraction using card-pyramid parsing. Proceeding of the 14th Conference on Computational Natural Language Learning (CoNLL, 2010), pp: 203-212.

Li, L., 2000. Translating use cases to sequence diagrams. Proceeding of the 15th IEEE International Conference on Automated Software Engineering (ASE '00), IEEE Computer Society, Washington, USA, pp: 293.

Mirzanur, R., D. Sufal and S. Utpal, 2009. Parsing of part-of-speech tagged assamese texts. Int. J. Comput. Sci. Issues, 6(1): 28-34.

Mitkov, R., 1998. Robust pronoun resolution with limited knowledge. Proceeding of the 18th International Conference on Computational Linguistics (COLING'98)/ACL'98 Conference. Montreal, Canada, pp: 869-875.

Overmyer, S.P., L. Benoit and R. Owen, 2001. Conceptual modeling through linguistic analysis using LIDA. Proceeding of the 23rd International Conference on Software Engineering, pp: 401-410.

Ramshaw, L.A. and M.P. Marcus, 1995. Text chunking using transformation-based learning. Proceeding of the 3rd Annual Workshop on Very Large Corpora, pp: 82-94.

Sismanis, Y., P. Brown, P.J. Haas and B. Reinwald, 2006. GORDIAN: Efficient and scalable discovery of composite keys. Proceeding of the 32nd International Conference on Very Large Data Bases (VLDB, 2006). ACM, Seoul, Korea, pp: 691-702.

Xian-Yi, C., C. Xiao-Hong and H. Jin, 2011. The Overview of Entity Relation Extraction Methods. In: Chen, R. (Ed.), Intelligent Computing and Information Science. Communications in Computer and Information Science, Springer-Verlag, Berlin, Heidelberg, 134: 749-754.

Yue, T., L.C. Braind and Y. Labiche, 2010. Automatically deriving UML sequence diagrams from use cases. Technical Report, TR-SCE-10-03, Carleton University, Canada, pp: 1-56.

Zhang, M., M. Hadjieleftheriou, B.C. Ooi, C.M. Procopiuc and D. Srivastava, 2012. Automatic discovery of attributes in relational databases. Proceeding of the 2011 ACM SIGMOD International Conference on Management of Data, ACM, New York, pp: 109-120.