Automated Generation of E-R Diagram from a Given Text in Natural Language

Sutirtha Ghosh
Dept. of Computer Science and Engineering
National Institute of Technology, Durgapur, India
ghsutirtha@gmail.com
Rezaul Bashar
Science, Technology and Management Crest
Sydney, Australia
rezaul.bashar2016@gmail.com

Prasenjit Mukherjee and Baisakhi Chakraborty
Dept. of Computer Science and Engineering
National Institute of Technology
Durgapur, India
prasen.mscit09@gmail.com
baisakhichak@yahoo.co.in

Abstract—This paper proposes an Automated E-R Diagram Generation (AGER) System that can generate E-R Diagram from a given text in Natural Language given as input. Natural language texts are used to perform the information extraction by parsing the syntax of the sentences and semantically analyzing their content. The AGER System parses an input text in natural language, semantically analyzes it and internally uses some domain specific database and POS tagging to detect Entity and Relations from the given passage and builds a graph that represents the E-R Diagram. The E-R Diagram can be traversed to generate Data Definition Language to create the actual relations in any RDBMS system. The abstract nature of entity relationship diagram makes it difficult for database designers to directly create E-R Diagram from the natural language text input statement which is used to create the physical model of the database. In the requirement analysis phase of any software design, often Databases designers need to have elaborate discussions with the client on the use cases of the databases and at the end of the work they come up with a neat E-R Diagram which shall be used in subsequent phases to physically realize the relations and implement them. The AGER system proposed here is aimed to assist database designers to create E-R Diagram directly from Client's requirements in natural language.

***Keywords*— E-R Diagram; Natural Language Processing; NL Sentence; ER-Generator.**

I. INTRODUCTION

An Entity Relationship (ER) data model describes the information about entities, attributes and relationships. Entity relationship model is a high level conceptual form which facilitates design of a database. This paper proposes a model which takes text description from the client as input and then generates an E-R Diagram after processing the text. The DBMS designer then can further interact to refine the model and remove any inconsistency. This will reduce initial burden and speed up the requirement analysis phase. This model uses some internal databases to store the frequently appearing domain specific entity names and attribute names along with their synonyms. Entity names and attribute names correspond to nouns and the several relation names correspond to verbs. The database contains Tables of Entity Names, Attribute Names, Relation names and their corresponding Synonym Names. Pronouns at the beginning of the sentence are resolved using the nearest noun that appeared in the previous statement. The number of entity names in any domain is in no way exhaustive so we cannot store all of them in a database, even new entity names are created daily such as new institution names may be created daily in the domain of educational institutes. The database of the AGER System uses POS tagger to detect Entities and their corresponding relations. Section 2 describes the related works, Section 3 describes the architecture of the system, Section 4 elaborates the Algorithm and implementation details. Section 5 winds up with conclusion and future scope of work.

II. RELATED WORKS

The construction of E-R Diagram is an important step in the requirement analysis phase of database designing. Peter Chen has proposed 11 rules to convert Natural Language to Entity Types and Relation Types in [1]. Commercial product for ER model generation includes Microsoft's Visio, Tech's ER Studio and Dia [2]. Most of the research work focuses on developing methodologies and tools to draw ER diagram or generate a UML model from a given description in natural Language as in [2]. Hatem Herchi has proposed a DC builder system to generate UML class diagram from user requirements given in natural language in [3]. Hatem Herchi has presented 12 rules to extract class names from natural language and then generated an XML file that was refined to generate UML class diagram. A heuristics-based approach has been used to generate E-R diagram from natural language specification. ER elements like entities, attributes and relationships are determined using semantic heuristics as in [4]. Large scale Object-based Language Interactor, translator and Analyzer (LOLITA) [5] is a NLP system that produces an object model from natural language specification that considers nouns as objects. The system is written in Haskell. This approach is limited to identify classes and cannot extract objects in different NL specification in [6].

Automated Knowledge Provider System has been developed to extract knowledge from knowledge database. This system is able to process Natural Language queries in

978-1-7281-0404-1/19/$31.00 ©2019 IEEE
DOI 10.1109/iCMLDE.2018.00026

Authorized licensed use limited to: Indian Institute of Technology - Jodhpur. Downloaded on October 01,2025 at 17:58:36 UTC from IEEE Xplore. Restrictions apply.

assertive sentence or in interrogative sentence. A rule based parsing technique has been applied on assertive and interrogative queries to convert those queries into SQL queries which is then executed on a database to fetch the answer(s) as in [7]. Requirement Analysis and Class Diagram Extraction (RACE) is a desktop instrument to assist requirement analysis. It extracts class diagram from a given requirement specification in natural language using Natural Language Processing and Domain Ontology techniques. It uses rules to identify classes, attributes and relationships and Domain Ontology to improve the performance of concept identification as in [8]. A method to convert Natural Language description to UML has been proposed in [9] to provide automatic assistance to developers. This paper focuses on generation of Activity Diagram and Sequence Diagram through Natural Language Processing. This system has proposed a novel technique that enhances the generation of UML model and provides automatic assistance to developers. Usama Iqbal and Imran Sarwar Bajwa has proposed a method to translate SBVR (Semantics of Business Vocabulary and Business Rules) rules into UML activity diagram. This method takes the system requirements specifically in SBVR syntax then parses the whole input and chooses UML ingredients like Noun, Verb, Fact type etc. and at last generates the visual representation of selected information as in [10]. The dependency analysis based approach is efficient to derive UML class automatically from Natural Language requirements. Requirement statements are transformed to an intermediary frame-based structured representation using dependency analysis of requirements statements and Grammatical Knowledge patterns. Class diagram using rule based approach is derived from the frame-based structure created previously as in [11]. Richa Sharma Et al. have proposed a method that uses Grammatical Knowledge Patterns and lexical and syntactic analysis of requirement text to create UML diagrams. First the text is transformed to intermediary structured representation-frames and then use the knowledge stored in frames to generate the UML diagrams as in [12]. M. O. Muss A and D. Wilson have described many existing rules that convert natural language to formal models such as Enhanced Entity Relationship Diagrams, Class Diagrams and Relational Database Schemas and explains their limitations. Also, many new rules are proposed for more accurate conversion. The rules are divided into three parts i.e. rules for entity/classes/tables, rules to detect relationship between entities and rules to define attributes of entities as in [13]. OWL (Web Ontology Language) is a language for knowledge representation in a semantic perspective. A good technique has been discussed for converting OWL to ERD (Entity Relationship Diagram). The most likely composition of ERD constructs that correspond to a given sequence of OWL axioms are selected using a Hidden Markov Model (HMM). OWL inputs are the observable states and ERD structures are the hidden states. Transmission and emission probabilities are heuristically set up through analysis of appropriate grammar describing the ERD syntax. Several OWL to ERD conversion rules are presented in [14].

III. ARCHITECTURE

The architecture of the proposed uses both domain specific databases and rules to convert text to ER diagram. We have used database specific to educational institutions for our work. The domain may refer to any other specifications like health care organizations, railways systems, travel and tourism domain and so on. The general outline of the system architecture is given below followed by explanation of each of the steps.

A. Description of Steps

1. Text in Natural Language: User inputs Text in Natural Language (NL) in a text box provided.

2. Sentence Segmentation: User generally inputs a set of NL sentences. NL Sentences are separated by full-stop and stored in a string array. The segmentation has been done in such a way that it will not detect cases like Mr. XYZ where (.) appears after Mr. as the end of sentence.

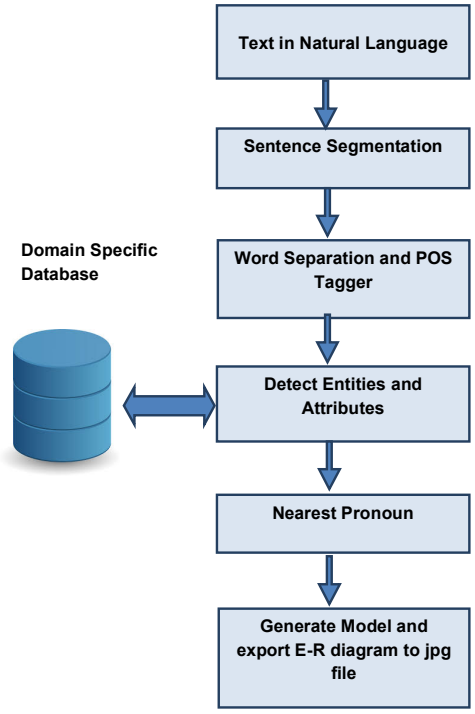

Fig. 1. Basic Diagram of Proposed System

3. Word Separation and POS tagging: Single and multiple spaces are removed from NL sentences and words are separated. Separated Words are inserted into the 2-D string array. These words are now passed to a rule based POS tagger (the NLTK POS tagger has been used through python computer language) for POS tagging to identify Noun, Pronoun, Verb, Adjective, Preposition, etc..

Authorized licensed use limited to: Indian Institute of Technology - Jodhpur. Downloaded on October 01,2025 at 17:58:36 UTC from IEEE Xplore. Restrictions apply.

4. Detect Entities, Attributes and Relations: The detection of entities attributes and relationship has been done with the help of database and SVM classifier in this step. Detailed explanation of actual algorithms is provided in later sections. After this phase, the Entities will be replaced by "<Eid>" where "id" is the corresponding id in the Entity table and similarly for attribute and relations. E.g.: <E1>

5. Nearest Pronoun Detection: The pronouns like he, she, it is already detected in the POS tagging phase. In this phase, the pronoun will be identified from nearest previous Entity that is present either in the same sentence or in the previous sentences.

E.g.: The sentence "*Students takes classes and they stay in hostel*." will be converted to "E<1> takes classes and they stay in hostel" after step 4. Again, "E<1> takes classes and they stay in hostel" will be processed and converted to "E<1> takes classes and E<1> stay in hostel" in step 5.

6. Generate the ER model: NL sentences will be short assertive sentences. System will extract entities, attributes and relationships through the steps 1 thru 5 in section Description of Steps. The extracted entities, attributes and relationships will contribute towards designing of database via E-R diagram generation.

B. Domain Specific Database

Six tables have been implemented which contain domain specific words and synonyms which are related to educational institutions. The tables are as follows:

a) Entity Name Table: This table contains entity names for a particular domain. The Educational Institutions has been selected as a domain; and Teacher, Student, Classroom etc. are examples of entities.
The sample table has been given in Table 1.

TABLE I: ENTITY NAME TABLE

Id	Entity Name
1	student
2	employee
3	course

b) Entity Synonym Database: This table contains synonym words for each entity which are assigned in Entity Name Table.
Example – If an entity "Teacher" is in the Entity Name Table, then all the synonyms of "teacher" word and "teacher" word itself should have in Entity Synonym table.
The sample table has been given in Table 2.

The id of each synonym indicates the word it points to in the Entity name table. That is Entity_Synonym.id is a foreign key to Entity_Name.id.

TABLE II: ENTITY SYNONYM TABLE

Id	Synonym
1	student
1	Boy
1	Girl
2	employee
2	professor
2	Associate Professor

c) Attribute Name Table: This table contains names of all the domain specific attributes. This table has been implemented because once a word is detected as Noun we need to disambiguate between whether it is an Attribute or an Entity. The structure is similar to Entity Name Table.

d) Attribute Synonym Table: This table stores the synonym of corresponding attributes present in Attribute Name Table. Structure is similar to Entity Synonym Table.

e) Relation Name Table: This table contains basically verbs defining relation between two entities E.g. Works, teaches, takes etc.

f) Relation Synonym Table: This table contains synonyms of the corresponding words present in Relation Name Table. These databases will be used in the Detect Entities, Attributes and Relationship phase.

IV. ALGORITHM AND IMPLEMENTATION

The user will post NL sentences. These sentences are assumed to be simple, assertive sentences for the best functioning to generate the ER model of the database. Interrogative, exclamatory sentences are exempted while specifying the requirements of a database. Assertive sentences are generally in the form of Subject + Verb + Object. Subject is considered to be Noun which specifies Entity [1]. Verb makes the relationship between Subject and Object [1], Object is also a Noun but it can be an Entity or Attribute. If the object is identified as an adjective then it will be treated as an attribute.

Sentence Processing Steps:

The paragraph (NL sentences) is represented as a 2D array of words where every line contains words from a single sentence. This process has been described in 3rd step of description steps.

General Rules:

i. If an entity and an attribute appear in a single sentence, then they are related to each other because the sentences that we dealing with are simple assertive sentences.

Authorized licensed use limited to: Indian Institute of Technology - Jodhpur. Downloaded on October 01,2025 at 17:58:36 UTC from IEEE Xplore. Restrictions apply.

ii. If two entities are linked by a verb which is detected as relation, then these two entities must be related.

iii. If a pronoun is present, then it represents the nearest subject appearing in the previous sentence.

iv. Multiple attributes appearing in a sentence will be linked to the first entity that appears in that sentence (applicable if only one entity is present in that sentence). Generally, the first entity is the subject.

A. Algorithmic Steps:

Entity and Attribute Detection:

1) Get the NOUN tags and search the word in the Entity Synonym Database. If it is present in the database then take the id of that word from the Synonym Database and fetch the word with same id from Entity Name Database. Now check whether that word is present in the Dictionary; if present then do nothing, else store it in the dictionary as a key.

2) System will fetch all adjectives from the same sentence that denotes the attributes of the entity [1] detected in step 1 of Algorithmic steps section. Sometimes nouns represent attributes, so nouns are also considered to be searched for attribute matching in the Attribute Database. If all the attributes detected in this phase corresponds to the entity detection in previous step, then store all the attributes in a list (say list L1) and store it as a value in the dictionary corresponding to the entity. Eg dict[student] = [[Name,Roll_no,Section]].

3) Verb makes the relationship between two entities in ER diagram. Entity name and Attributes have been detected in previous step. AGER system will take the verb tagged word and search for another entity name in the same sentence. Verify verb and entity words that exist in the database. Verb connects both the entities. Now we store the verb entity pair in the second list i.e. dict[key]= [[contains attributes] [(verb, entity)]], here key is the entity detected in step 1 of Algorithmic steps section.

4) The dictionary will be populated after step 3 of Algorithmic steps and system will generate the graph of ER model of database after using the entries present in dictionary.

i. Create an entity node (Rectangle shape) with the attribute node (Ellipse shape) for every key present in the dictionary.

ii. Traverse all the entities and make the relation (Rhombus) and connect the participating entities with it.

iii. Create the svg image. (graphviz has been used)

B. Algorithm

```
for sentence in sentences
        entities = getNouns(sentence)
        attributes = getAdjs(sentence)
        relations = getverbs(sentence)
        root_entity = entities [0]
        if root_entity not in dictionary
        dictionary[root_entity] = [list(),list()]
        for word in sentence
if (word in entities or word in EnitySynonymDatabase) and word
not in dictionary.keys():
        dictionary[word] = [ list(), list() ]  # two empty list in a
list
        if word in attributes or word in
AttrbuteSynonymDatabase:

        dictionary[root_entity][0].append(word)
                if word in relations:
find the other entity(say E2) except the root_entity which
participates in the relation
Make a tuple t = (word, E2)
dictionary[root_entity][1].append(t)
```

TABLE III: COMMON TAGS FOR POS TAGGER

Tag	Meaning	Examples	Tag	Meaning	Examples
ADJ	Adjective	new,very, good	ADP	Adposition	on,at,with
ADV	Adverb	Still, early	CONJ	Conjunction	And, or
DET	Determiner, Article	The, a , an, some	NOUN	Noun	year,home
VERB	Verb	is,takes, gives	PRON	pronoun	He, their, it

C. Example

An example has been given below.

Student has name and address. Every boy has roll number. Student takes courses. Course has title, level and credits. Course has course_id.

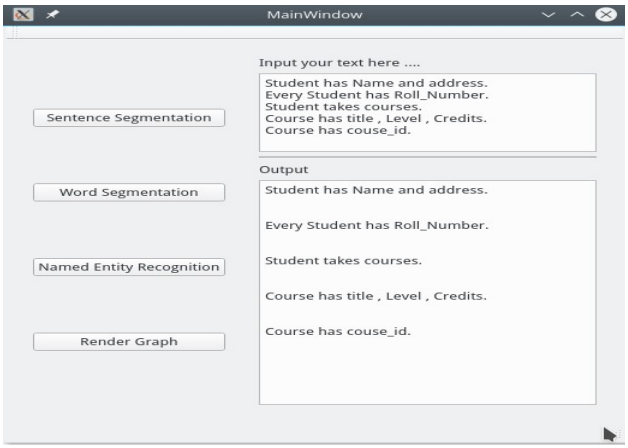

Fig. 2. Sentence Segmentation shown in Output block

Authorized licensed use limited to: Indian Institute of Technology - Jodhpur. Downloaded on October 01,2025 at 17:58:36 UTC from IEEE Xplore. Restrictions apply.

Step 1: First the paragraph (NL sentences) text is segmented into individual sentences (Fig. 2) and stored in an array. Statement segmentation has been done on the basis of placement of full stop (.) but at the same time considering some special uses of (.) like Mr., etc.

Step 2: In this step, the sentences are segmented into individual words considering spaces as the separator between individual words.

Step 3: In step 3 of Description step, the words are sent to POS tagger as argument and it returns the words along with their tags. The common tags have been used by POS tagger that is shown in Table 3. Below the full data structure has been give that is returned by the POS tagger after POS tagging.

TABLE IV: COMMON TAGS FOR POS TAGGER

Sentences	Words1	Words2	Words3	Words4	Words5
Sentence 1	Student, Noun	Has, Verb	Name, Verb	And, Conj.	Address, Noun
Sentence 2	Every, Det	Student, Noun	Has, verb	Roll Number, Noun Phrase	
Sentence 3	Student, Noun	Takes, Verb	Courses, Noun		
Sentence 4	Course, Noun	Has, Verb	Title, Noun	Level, Noun	Credits, Noun
Sentence 5	Course, Noun	Has, Verb	Course_id, Noun		

A domain specific database has been defined with Entities, Attributes, Relationships and its synonyms. After this step, Tags from Table 4 and the domain specific database have been used to generate the dictionary table. We specifically need domain specific database to remove ambiguities because only rules based on parts of speech tags can have ambiguities for e.g. 3rd element in first row "Name" is detected as a verb which represents an attribute, thus to remove this form of ambiguities we use the domain specific database. A simplified algorithm that runs on the above data structure represented in Table 4 is explained in algorithms and implementation details, section 4.
The structure of the dictionary table has been given in Table 5.

After First Statement:

Sentence: “Student has Name and Address” - The "Name" is detected as verb but since it is present in Attribute Synonym Database we consider it as an attribute.

TABLE V: DICTIONARY AFTER PROCESSING SENTENCE 1

Entity Name	Attributes		Relationships
Student	Name	Address	-

After Second Statement:

Sentence: “Student takes courses”, here takes is a verb which relates two entities Student and courses. Student is already present in the dictionary where courses is not present and will be included in the dictionary. Relationship is stored in form of two value tuple like (takes, Courses). Therefore, the dictionary will look like-

TABLE VI: DICTIONARY AFTER PROCESSING SENTENCE 2

Entity Name	Attributes		Relationships	
Student	Name	Address	(takes, course)	-
Courses	-		-	

After third statement:
Sentence: “Courses has title, level, credits”, courses entity already exists in dictionary and now 3 attributes will be added to it since they are present in Attribute Synonym Database. Even if they were not and they were adjectives we could certainly say that these words are adjectives because adjectives represent attributes.

TABLE VII: DICTIONARY AFTER PROCESSING SENTENCE 3

Entity Name	Attributes			Relationships	
Student	Name	Address		(takes, course)	-
Courses	Title	Level	Credits	-	

After fourth statement:
Sentence: “Courses has course_id” - courses is already present in the dictionary so it is detected as attribute of Courses entity.

TABLE VIII: DICTIONARY AFTER PROCESSING SENTENCE 4

Entity Name	Attributes				Relationships	
Student	Name	Address			(takes, course)	-
Courses	Title	Level	Credits	Course_id	-	

The final dictionary state contains all the information to generate the ER Diagram after all the sentences are processed. Graph-viz has been used to render the graph into a SVG file. The dictionary as shown in Table 8 is traversed to create the image. After processing every entity, we mark it as done. First the entity in row 1 is processed and then we move on to the entity that is present in Relationship section of the current entity. If no relationship section exists then move on to next entity and at end ensure all the entities are processed. This is a Breadth First Search with minor variations.

Authorized licensed use limited to: Indian Institute of Technology - Jodhpur. Downloaded on October 01,2025 at 17:58:36 UTC from IEEE Xplore. Restrictions apply.

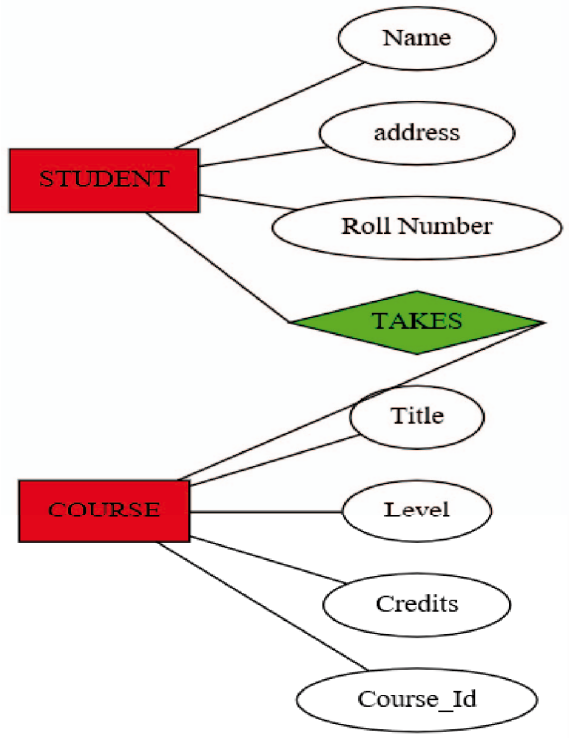

Fig. 3. ER Diagram generated after processing

V. Conclusion and Future Scope

The main use of the database in the AGER System is to automatically create E-R Diagram from natural language texts given as simple assertive sentences. The AGER system is intended to assist the database designers to create E-R Diagram directly from Client's requirements in natural language in Requirement analysis phase However, the system is not suitable for inputs that are in form of compound and complex sentences which shall be considered as a future work. Also, there shall be provision for database designers to include data definition language (DDL) in the proposed system.

Work is still being done on automatic extraction of Named Entities from an annotated domain specific database which will automatically generate the database. Another area for improvement is using a Support Vector Machine to chunk multiword entity names.

References

[1] P.P.S. CHEN, "English Sentence Structure and Entity-Relationship Diagrams", *Elsevier Science publishing Company*, 52 Vanderbilt Ave., New York, 10017, 1983.

[2] C. Elena , C. Dolores, M. Paloma and I. Ana, "Integrating Intelligent Methodological and Tutoring Assistance in a CASE Platform: The PANDORA Experience", *Informing Science*, pp.261-269, 2002.

[3] H. Herchi and B. A. Wahiba, "From user requirement to UML class diagram", *International Conference on Computer Related Knowledge*, 2012.

[4] F. Gomez, C. Segami and C. Delaune, "A system for the semiautomatic generation of E-R models from natural language specifications", *Data and Knowledge Engineering,* Vol. 99, no.1, pp. 57-81, 1999.

[5] L. Mich, "NL-OOPS: from natural language to object oriented requirements using the natural language processing system LOLITA", *Natural Language Engineering*, pp. 161-187, 1996.

[6] S. B. Eman and M. H. Mustafa, "Generating ER Diagram from Requirement Specification Based on Natural Language Processing", *International Journal of Database Theory and Application*, vol. 8, pp. 61-70, 2015.

[7] P. Mukherjee and B. Chakraborty, "Automated Knowledge Provider System with Natural Language Query Processing", *IETE Technical Review*, vol. 33, no. 5, pp. 525-538, 2016.

[8] M. Ibrahim and A. Rodina, "Class Diagram Extraction from Textual Requirements Using Natural Language Processing(NLP) Techniques", *Second International Conference on Computer Research and Development*, Kuala Lumpur, Malaysia, 2010.

[9] S. Gulia and T. Choudhury, "An efficient automated design to generate UML diagram from Natural Language Specification", *6th International Conference Cloud System and Big Data Engineering (Confluence),* IEEE, Noida, India, 2016.

[10] U. Iqbal and I. S. Bajwa, "Generating UML activity diagram from SBVR rules", *Sixth International Conference on Innovative Computing Technology (INTECH)*, IEEE, Dublin, Ireland, 2016.

[11] R. Sharma, P. K. Srivastava and K. K. Biswas, "From natural language requirement to UML class diagrams", *IEEE Second International Workshop on Artificial Intelligence for Requirements Engineering (AIRE)*, IEEE, Ottawa, ON, Canada, 2015.

[12] R. Sharma, G. Sarita and K.K. Biswas, "Automated generation of activity and sequence diagrams from natural language requirements", *International Conference on Evaluation of Novel Approaches to Software Engineering (ENASE)*, IEEE, Lisbon, Portugal, 2014.

[13] M. O. Muss and D. Wilson, "New rules for deriving formal models from text", *International Conference for Students on Applied Engineering (ICSAE)*, IEEE, Newcastle upon Tyne, UK , 2016.

[14] A.Pipitone and R. Pirrone, "A Hidden Markov Model for Automatic Generation of ER Diagrams from OWL Ontology", *IEEE International Conference on Semantic Computing (ICSC)*, IEEE, Newport Beach, CA, USA, 2014.

Authorized licensed use limited to: Indian Institute of Technology - Jodhpur. Downloaded on October 01,2025 at 17:58:36 UTC from IEEE Xplore. Restrictions apply.