International Journal of Information, Control and Computer Sciences
ISSN: 2517-9942
Vol:9, No:1, 2015

Implementing a Database from a Requirement Specification

M. Omer, D. Wilson

***Abstract*—**Creating a database scheme is essentially a manual process. From a requirement specification the information contained within has to be analyzed and reduced into a set of tables, attributes and relationships. This is a time consuming process that has to go through several stages before an acceptable database schema is achieved. The purpose of this paper is to implement a Natural Language Processing (NLP) based tool to produce a relational database from a requirement specification. The Stanford CoreNLP version 3.3.1 and the Java programming were used to implement the proposed model. The outcome of this study indicates that a first draft of a relational database schema can be extracted from a requirement specification by using NLP tools and techniques with minimum user intervention. Therefore this method is a step forward in finding a solution that requires little or no user intervention.

***Keywords*—**Information Extraction, Natural Language Processing, Relation Extraction.

I. Introduction

DERIVING a data storage system is a subsection of a software development life cycle and goes through several stages before a physical database is created. There are many manual techniques that can help the system analyst in the early stages of database development such as data gathering techniques, designing and analyzing Data Flow Diagrams (DFDs) and Enhancement Entity Relational Diagrams (EERDs). There are also several tools which can automatically help in creating a database schema such as CM_Bulider [9] and Class_Gen [12]. The purpose of this paper is to demonstrate a Natural Language Processing tool that will auto-generate a relational database from a requirement specification. This paper is organized as follows: work related to NLP is discussed in Section II. The proposed extraction method, identification algorithms, and an analysed case study are discussed in Section III. Section IV describes the evaluation methodology; case studies; result summary with discussion; conclusion and future work.

II. Related Work

In 1976 the Entity Relationship Model (ERM) was introduced by Chen [1]. This model gives clear illustration about how the information stored in a database is derived. It shows entities, attributes of each entity, and the relationship between entities. The ERM is obtained through several stages of development, including data gathering and data analyzing. These stages are often time consuming as they are preformed manually. An updated model of the ERM was proposed, known as the Enhancement Entity Relationship Model (EERM).

Chen explained that an ERM gives a clear idea about the system, not just for system and database analysers but also for database managers and database users. Chen declared that, there is a relationship between the Entity Relationship Model and English Sentences Structure. He gave 11 rules to express the relationship; they are described as guides rather than as rules [2]. By using Chen's rules several research projects have attempted to build an automatic system to derive an ERM, as will be shown later.

Following on from Chen, an expert system for creating a relational database was created. The system in question starts by collecting information about the database by asking a client questions related to the nature of the database. For example, details of the information which should be stored in the database, the relations between database entities and the attributes of each entity. Next, the information is analyzed and an Entity Relation Diagram (ERD) is designed. Finally, the normalized fourth form is established by applying functional dependencies [3]. The major problem with this system is that it relies on the interaction between the client and the analyst and in many cases, a considerable amount of time.

Analyzing documentation is not a new concept. In 1994 a tool by the name of Data Module Generator (DMG) was implemented [4]. A requirement specification is written in a natural language such as English, Chinese, or Arabic. German was used as a Natural Language to be entered into the DMG and an EERM data model is output from the tool. The DMG extracts the information which is needed to create the EERM; the extraction depends on the relationship between the natural language and the ERM as per the rules discussed by Chen. Because of errors and incomplete information that is often contained within requirement specification, the DMG cannot achieve a satisfactory output without human intervention. For this reason, the DMG is considered as a semi-automatic system and more research needs to be done in this area. However, with continuing improvements in Natural Language Processing, we propose that that NLP tools and techniques can be implemented to analyze a data requirements specification with a high degree of accuracy with little or no human intervention.

Meziane explained that a requirement specification is described in a natural language because it is understood by

M. Omer is PhD student at the University of Huddersfield, UK (e-mail: muss.omer@hud.ac.uk).

D. Wilson is a senior lecturer and Course Leader (MSc Advanced Computer Science, Network Technology and Management, Information Systems Management) at the University of Huddersfield, UK (e-mail: d.r.wilson@hud.ac.uk).

users and developers alike [5]. However, a requirement specification can often be incomplete and contain errors. As such, any errors or limitations in a requirement specification can lead to errors in meeting system requirements. Meziane converted a requirements specification into a logical form using language representations to remove errors and incomplete aspects to obtain entities and relationships. Meziane's system was implemented by using Prlog-2 and a case study was provided [5], [6].

Buchholz proposed a Dialogue Tool to design a database, specifically the EER model. The functionality of the tool is such that a moderated dialogue (in the form of a question-answer) begins when information is entered into the system. There are three types of questions: Content Questions (CQ) asking whether there is information that needs to be added; Linguistic Clarification (LQ) which describes how an event will be carried out; and Pragmatic Clarification (PC) that defines the way the objects are described. Secondly, syntax analysis (grammar lexicon and parse) and semantic analysis (meaning of the sentences) are achieved. Finally, programmatic interpretation is carried out to convert natural language to an ERD [7]. This particular tool still requires a significant amount of user input however; it requires much less involvement than the expert system devised by Storey and Goldstein [3].

Chen showed, as mentioned earlier, there is a direct correspondence between the English Language and the ERM. He states there is also a direct correspondence between the Chinese language and the ERM. The output of Chen's work can be used to clarify how a requirement specification could be analyzed in many languages such as English and Chinese [8].

Harmain developed a natural language base case tool known as Class Model Builder (CM_Builder). Its purpose was to help extract classes, attributes and relationships automatically rather than done manually by a system analyst. In other words it produces a class diagram as represented by the Unified Modelling Language (UML). The CM-Builder is a semi-automatic system. There are two versions of CM_Builder. A better performance and less human intervention is available in CM_Bulider version 2 [9]. The purpose of CM-Builder was not for producing class models automatically without human intervention, but was developed to show that NLP can help in producing an initial class diagram, which can then be reviewed and refined by software engineers to produce a final version for the class diagram.

Heuristics were used to design a semi-automated tool known as ER_converter. ER_Converter will assists in producing the Entity Relationship Diagram (ERD) from a requirement specification. The process starts when a requirement specification is read by the system. The system then uses heuristics and human intervention to build the ERD [10]. However, the ER_converter still requires some human intervention; although less than CM-Builder.

Al-Safadi claimed that there are three ways to implement Part Of Speech (POS) extraction: Firstly, create a lists of nouns, verbs, pronouns, adjectives, and adverbs, and then match the requirements specification to the list to find the POS. Al-Safadi argued that the problem with this is that some words can be found in more than one category, for instance, 'book' can be both a noun and a verb. Secondly, POS can be achieved by words with special endings. For instance, one of the well-known noun's special ending is (- al), depending on that, the word (several) and the word (rental) are categorized as nouns. Finally, POS can be implemented by using English language grammar. Al-Safadi showed that this is an optimal approach and used it to design a tool known as the Database Design Tool (DBDT) [11]. DBDT extracts entities, attributes, relationships between entities, cardinality, and multiplicity. However, Al-Safadi set rules about how a requirement specification file should be written.

Elbendak designed a NLP tool known as (Class_Gen) to produce a class model. This tool produces a 'first cut' for the class model. The software engineer then reviews and refines it to produce the final class model [12].

Slankas is working to implement a tool to extract access control policies from unconstrained natural language text. The input of this system is a natural language text, which describes a relational database and said access control polices. The system output will be a set of Structured Query Language (SQL) commands, which describe the required access control polices in the database [13]. This work is related to our research as it intends to extracts a relational database from a requirement specification as the initial step before extracting the access control polices (*project target*).

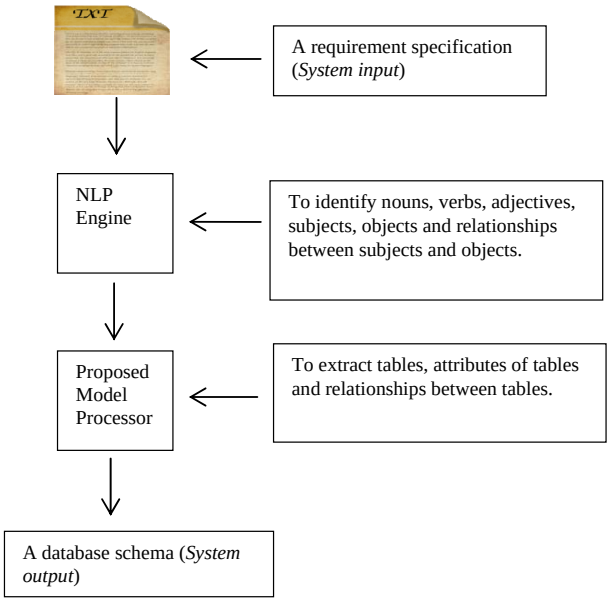

Fig. 1 System architecture

III. Proposed Extraction Method

A requirement specification is passed into the system (*system input*); it is analyzed by a NLP tool. The Stanford CoreNLP version 3.3.1 [14], works as a NLP engine and identifies nouns, adjectives, verbs, subjects and objects; it also

detects the relationship between subjects and objects. The output of the NLP engine is fed into our proposed model processor. The proposed model processor generates an initial database. The initial database is reviewed and refined by a system analyst before a database schema is created (*system output*). Fig. 1 represents system architecture.

The proposed solution to the task of auto-generating a database schema is defined in three stages:

A. Identify the process by which auto-generation will be achieved (*The identification algorithm*)
B. Select a suitable case study
C. Apply the identification algorithm

A. The Identification Algorithm

1) Each direct subject, direct object, passive subject, and passive object, is considered as a candidate table.
2) Each verb is considered as a candidate relation and its arguments are found.
3) For each candidate table find its frequency.
4) Find compound nouns in sentences and attach them to their candidate tables and candidate relations.
5) Find some kind of relations, which can give an indication to attributes such as possession, something has something and something includes something.
6) Find some relations which can raise primary keys such as 'identified by'.
7) Remove vague candidate tables.
8) Remove each vague argument in candidate relations.
9) Remove each table which has no attributes from candidate tables and candidate relations.
10) Remove each candidate table which has no relations.
11) Remove each candidate relation that has less than two tables as arguments.
12) Assess the initial database; it is reviewed and refined by a system analyst to produce a final database schema (System output).

B. Case Study with Analysis (Store Problem)

As in [9], a case study (Store Problem) was taken as it is, see below

A store has many branches. Each branch must be managed by at most 1 manager. A manager may manage at most 2 branches. The branch sells many products. Product is sold by many branches. Branch employs many workers. The labour may process at most 10 sales. It can involve many products. Each Product includes product code, product name, size, unit_cost and shelf_no. A branch is uniquely identified by branch_number. Branch has name, address and phone_number. Sale includes sale_number, date, time and total_amount. Each labour has name, address and telephone. Worker is identified by id'.

C. Applying the Identification Algorithm to Define a Database Schema

By applying step 1 of the identification algorithm on the first sentence, it is found that (store) is a candidate table because it is the subject of the sentence, this appears in Stanford CoreNLP in the relation nusubj(has-3,store-2). (Branches) is also a candidate table because it is a sentence object, this appears in the relations dobj(has-3, branches-5), see below:

A store has many branches.
root(ROOT-0, has-3)
det(store-2, A-1)
nsubj(has-3, store-2)
amod(branches-5, many-4)
dobj(has-3, branches-5)

A full description of the Collapsed dependencies processed by the Stanford CoreNLP for the Store problem is given in appendix A.

Applying step 2 shows that the verb (has) is a candidate relation and its arguments are store and branches; this appears in the CoreNLP in the relations nsubj(has-3,store-2) and dobj(has-3, branches-5), the relation name is (has-3). The output when the first sentence is analyzed by Stanford is shown in Tables I and II.

TABLE I
CANDIDATE TABLES FOR FIRST SENTENCE

Candidate table name	Frequency
Store	1
Branches	1

TABLE II
NOMINEE RELATIONS FOR FIRST SENTENCE

Candidate relation
Candidate_ relation(has, store, branches)

Steps 1 and 2 are applied on the sentences from 1 to 14 of Store problem, then the frequency of each candidate table is found (step 3) as in Tables III and IV.

TABLE III
ALL CANDIDATE TABLES AND THEIR FREQUENCY AT STEP 3

Candidate table	frequency
Branch	7
Product	4
Address	2
Labour	2
Name	3
Sale	2
Worker	2
Code_no	1
Date	1
It	1
Manager	1
Phone_number	1
Code	1
Sale_number	1
Shelf_no	1
Size	1
Store	1
Time	1
Total_amount	1
Unit_cost	1

TABLE IV
ALL CANDIDATE RELATIONS AT STEP 3

Candidate relations
Candidate_relation(has, store, branches)
Candidate_relation(managed, branch)
Candidate_relation(manage, manager, branches)
Candidate_relation(sell, branch, products)
Candidate_relation(sold, product)
Candidate_relation(employs, branch, workers)
Candidate_relation(process, labour, sales)
Candidate_relation(involve, it, products)
Candidate_relation(include, product, code, name, size, unit_cosit, shelf_no)
Candidate_relation(identified, branch)
Candidate_relation(has, branch ,name, address, phone_number)
Candidate_relation(include, sale , sale_number, date, time, total_amount)
Candidate_relation(has, labour, name, address, telephone)
Candidate_relation(identified, Worker

TABLE V
ALL CANDIDATE TABLES AND THEIR FREQUENCY AT STEP 4

Candidate table	frequency
Branch	7
Product	2
Address	2
Labour	2
Name	2
Sale	2
Worker	2
Code_no	1
Date	1
It	1
Manager	1
Phone_number	1
Ptoduct Code	1
Sale_number	1
Shelf_no	1
Size	1
Store	1
Time	1
Total_amount	1
Unit_cost	1
Product name	1

TABLE VI
ALL CANDIDATE RELATIONS AT STEP 4

Candidate relations
Candidate_relation(has, store, branches)
Candidate_relation(managed, branch)
Candidate_relation(manage, manager, branches)
Candidate_relation(sell, branch, products)
Candidate_relation(sold, product)
Candidate_relation(employs, branch, workers)
Candidate_relation(process, labour, sales)
Candidate_relation(involve, it, products)
Candidate_relation(include, product code, product name, size, unit_cosit, shelf_no)
Candidate_relation(identified, branch)
Candidate_relation(has, branch ,name, address, phone_number)
Candidate_relation(include, sale , sale_number, date, time, total_amount)
Candidate_relation(has, labour, name, address, telephone)
Candidate_relation(identified, Worker

By applying step 4, there are two compound nouns including product code and product number. Compound nouns are expressed in Stanford CoreNLP as nn, see sentence 9 in appendix A. This affects the candidate tables and relations, as in Tables V and VI.

Step 5, there are three (something has something) relations:

1) Candidate_relation (has, store, branches);
2) Candidate_relation (has, branch, name, address, phone_number);
3) Candidate_relation (has, labour, name, address, telephone).

There are also two (something includes something) relations:

1) Candidate_relation (include, product, product code, product name, size, unit_cost, shelf_no)
2) Candidate_relation (include, sale, sale_number, date, time, total_number).

Those relationships produce attributes. This means (branches) is an attribute for the store table. Name, address and phone_number are attributes for the branch table. Name, address and telephone are attributes for the labour table. Product code, product name, size, unit_cost and shelf_no are attributes for the product table. Finally, sale_number, date, time and total_number are attributes for the sale table. All these attributes will be discarded from being candidate tables and are attached to their tables as attributes.

Step 6, there are two (identified by) relations:

1) Candidate_relation (identified, branch)
2) Candidate_relation (identified, worker).

The expression agent (identified-5, branch_number-7) in sentence 10 shows that, branch_number is a primary key for branch table, and the expression agent(identified-3, id-5) in sentence 14 shows that Id is a primary key for the worker candidate table. Candidate tables and candidate relations after the filtering are affected as in Tables VII and VIII.

TABLE VII
CANDIDATE TABLES AFTER APPLYING STEPS 5 AND 6

Symbol	Quantity	Attributes
Branch	7	Branch_number (Primary key), name, address and phone_number.
Product	2	Product code, product name, size , unit_cost and shelif_no
Labour	2	Name, address and telephone
Sale	2	Sale_number, date, time and total_number.
Store	1	Branches
Worker	2	Id (primary key)
It	1	
Manager	1	

TABLE VIII
CANDIDATE RELATIONS AFTER APPLYING STEPS 5 AND 6

Candidate relations
Candidate_relation(managed, branch)
Candidate_relation(manage, manager, branches)
Candidate_relation(sell, branch, products)
Candidate_relation(sold, product)
Candidate_relation(employs, branch, workers)
Candidate_relation(process, labour, sales)
Candidate_relation(involve, it, products)

TABLE IX
CANDIDATE TABLES AFTER APPLYING STEPS FROM 7 TO 11

Symbol	Quantity	Attributes
Branch	7	Branch_number (Primary key), name, address and phone_number.
Product	4	Product code, product name, size , unit_cost and shelif_no
Labour	2	Name, address and telephone
Sale	2	Sale_number, date, time and total_number.
Store	1	Branches
Worker	2	Id (primary key)

TABLE X
CANDIDATE RELATIONS AFTER APPLYING STEPS FROM 7 TO 11

Candidate relations
Candidate_relation(sell, branch, products)
Candidate_relation(employs, branch, workers)
Candidate_relation(process, labour, sales)

As in Tables IX and X steps from 7 to 11 remove the candidate table (It) from the tables list, and remove the argument (It) from candidate relations at Candidate_relation (involve, it, products) because (it) is vague and because (it) has no attributes. The (manager) table is also discarded from candidate tables and candidate relations because it has no attributes:

1) Candidate_relation(managed, branch)
2) Candidate_relation(manage, branches)
3) Candidate_relation(sold, product)
4) Candidate_relation(involve, products)

are removed from candidate relations because they have one argument.

Step 12 offers an opportunity for system analysts to review the initial database and finalize it. Database names, tables, attributes, attributes type, relations and constraints can be added or removed. The system produces plain text file that contains SQL statements to describe a database schema. SQL statements are chosen as an output because they are understood by many databases. The final output after step 12 is listed below:

CREATE TABLE branch(address varchar(50), phone_number int, name varchar(50));
CREATE TABLE labour(address varchar(50), telephone int, name varchar(50));
CREATE TABLE product(size varchar(50), shelf_no varchar(50), unit_cost varchar(50),product name varchar(50),product code varchar(50));
CREATE TABLE sale(sale_number int, time datetime, total_amount int, date datetime);

IV. EVALUATION METHODOLOGY

This approach attempts to extract an initial database from a problem description, therefore, it is considered as Information Extraction (IE) software. An IE system is usually evaluated by comparing the result of the proposed system with the manual result. In our case the result of our system is referred to as (proposed system result) and the result which is already produced manually by the system analyst is called (manual result). Recall and precision are also important factors to measure the performance of our system. They were designed for evaluating Information Retrieval (IR) systems but they are widely used in IE systems. Recall is used to measure the extent the proposed system result is completed, compared to a manual result. "Using (1), we calculated the recall."

N correct = total of correct answers produced by the system. N missed = total of correct answers from a manual result and are missed from a proposed result.

$$\text{recall} = \frac{\text{N correct}}{\text{N correct + N missed}} * 100 \quad (1)$$

The precision reflects the extend the information which is extracted by the proposed system is correct, "Using (2), we calculated the precision."

$$\text{precision} = \frac{\text{N correct}}{\text{N correct + N incorrect}} * 100 \quad (2)$$

N incorrect refers to the total of incorrect answers which are extracted by the proposed system.

In our case, there is no right or wrong answer, but there is a good answer and a bad answer. Occasionally the proposed system gives correct answers but they are not included in a manual answer, for this reason over-specification is measured.

Over-specification displays to what extent the proposed system extracts correct answers, which are not found by system analyst in the manual answer [12]. "Using (3), we calculated the over-specification."

$$\text{over specification} = \frac{\text{N extra}}{\text{N correct + N incorrect}} * 100 \quad (3)$$

N extra = total of correct answers which are not found in a manual answer.

One of four classifications will be given to each answer. CORrect (COR) if it is found in both a proposed system result and a manual result. INCorrect (INC) when it is not found in a manual result and both a problem description and our own argument show it is incorrect. Extracted answer is marked as EXTra (EXT) when it is not included in a manual result, but both a problem description and our own argument show it is correct. MISSed (MISS) if it is a correct answer but it is not included in the proposed system result.

A. Case Studies

All case studies in appendix B were taken from [12] as they are with their manual result. At the moment the evaluation is a partial evaluation as only table names are evaluated. Tables are considered the most important part for the relational database. In future, attributes, primary keys, and relations between tables will be evaluated. Each case study is given a table to make comparison of the answer, which was obtained by the proposed system and the answer proposed by the system analyst. Each table has three columns, the first column for the table name which are proposed by our system, the second column for table name which are given by the system analyst and third column for answer classification. See Tables XI-XVI.

TABLE XI
COMPARISON OF THE PROPOSED SYSTEM RESULT AND THE MANUAL RESULT FOR ATM PROBLEM

Proposed system result	Manual result	Classification
Account, account data	Account	COR
Teller machine	ATM	COR
Bank	Bank	COR
Cash card	Cash Card	COR
Cashier	Cashier	COR
Cashier Station	Cashier Station	COR
Process transaction	Remote Transaction	COR
Transaction	Cashier Transaction	COR
Computer	Bank Computer	COR
	Customer	MISS
	Consortium	MISS
	Central Computer	MISS
Banking network		EXT
Prints receipt		EXT
Security provision		EXT
Access		EXT
Cash		INC
Cost		INC
Recordkeeping		INC

TABLE XII
COMPARISON OF THE PROPOSED SYSTEM RESULT AND THE MANUAL RESULT FOR ORGANIZATION PROBLEM

Proposed system result	Manual result	Classification
Department	Department	COR
Project	Project	COR
Staff	Staff	COR
Manager	Manager	COR
Staff Member		EXT

TABLE XIII
COMPARISON OF THE PROPOSED SYSTEM RESULT AND THE MANUAL RESULT FOR ELECTRICAL FILLING PROGRAM PROBLEM

Proposed system result	Manual result	Classification
Text document	Text Document	COR
Keyword	Keyword	COR
Character	ASCII Character	COR
Index	Index	COR
Word	word	COR
	Junk word	MIS
	Abstract	MIS
	page	MIS
	Author	MIS
	Line	MIS
Search criteria		EXT
Document		EXT
Document description		EXT
User		EXT
Document filed		EXT
Search		EXT
Retrieval		EXT
Example user		INC
Filing program		INC
EFP		INC

TABLE XIV
COMPARISON OF THE PROPOSED SYSTEM RESULT AND THE MANUAL RESULT FOR LIBRARY SYSTEM PROBLEM

Proposed system result	Manual result	Classification
Book	Book	COR
Catalogue note	Catalogue_note	COR
Delivery note	Deleivery_note	COR
Note	Note	COR
Invoice	Invoice	COR
Order	Order	COR
	Enquiry Note	MIS
	Person	MIS
Publisher		EXT
Library		EXT
Letter		EXT
Cheque		EXT

TABLE XV
COMPARISON OF THE PROPOSED SYSTEM RESULT AND THE MANUAL RESULT FOR JOURNAL REGISTRATION PROBLEM

Proposed system result	Manual result	Classification
Article	Article	COR
Journal	Journal	COR
Reader	Reader	COR
	Issue	MIS
	Topic	MIS
Permission		EXT
Access		EXT
Department		EXT

TABLE XVI
COMPARISON OF A PROPOSED SYSTEM RESULT VERSUS A MANUAL RESULT FOR LOCAL HOSPITAL PROBLEM

Proposed system result	Manual result	Classification
Doctor	Doctor	COR
Nurse	Nurse	COR
Patient	Patient	COR
	Ward	MIS
	Prescription	MIS
Responsibility		EXT
Drug		EXT
Hospital		EXT

TABLE XVII
RESULT SUMMARY AND DISCUSSION

Case study name	Recall	Precision	Over-specification
ATM	75%	75%	33%
Organization problem	100%	100%	25%
Electrical Filling Program	50%	62%	87%
Library system	75%	100%	67%
Journal registration	60%	100%	100%
Local hospital problem	60%	100%	100%

B. Result Summary and Discussion

In the first case study (ATM problem) the recall is 75% because there are three tables that are missed. The three missed tables are customer, central computer and consortium; all of them have no attributes which mean even if the system extracts them successfully they will be removed as they have no attributes (see step 9 of the identification algorithm). The precision in this case study is 75% because three tables are

extracted incorrectly, they include: cash, cost, and recordkeeping and all of them will be discarded later according to step 9.

In the second case-study (Organization problem), the result is 100% for recall and precision, it is the same as the manual result.

In the third case-study, Electrical Filling Program (EFP) recall is 50% because there are five missed tables including the page table and the line table which were not mentioned in problem statement. They also include: junk word, abstract, and author which have no attributes mentioned in the text. The precision is 62% because three tables were extracted incorrectly, they include: example user, filing program, and EFP. They have no attributes.

In the fourth case-study (Library System problem), recall is 75% because there are two missed tables, they include person table, and enquire note table; both of them were not mentioned clearly in the given text.

In the fifth case-study (Journal Registration problem), recall is 60% because there are two missed tables, they include: issue table and topic table. Issue table has no clear attributes and no clear relations, and topic table also has no attributes.

In the sixth case-study (Local Hospital problem), the recall is 60% because there are two missed tables, they include: ward, which has no attributes, and prescription, which was not mentioned in the given text, see Table XVII.

The result is encouraged, especially as it is clear that it is difficult to obtain a perfect database from a requirement specification. There are fundamental reasons for this; each analyst has his/her own viewpoint, and what is seen correct and important by one system analyst may not be seen important and correct by others. The given cases-studies do not cover the entirety of all systems. For instance, the first case-study ATM, which contains 166 words, does not cover everything about the ATM system, as in the other cases. Consequently, at this time, the extracted database will be incomplete. With information extraction, it is improbable that we would have guaranteed 100% accurate answers without implementing an Artificially Intelligent system. Typically, a computer program would need to think like a human to obtain consistently good results. This is a difficult task or may be impossible.

C. Conclusion and Future Work

To the best of our knowledge, no automated NLP based tool that targets to replace the system analyst is expected to be successful at present due to the current capability of NLP tools and techniques. However NLP can help in generating a relational database schema from a requirement specification. The proposed system is able to extract an initial database from a requirement specification with some degree of accuracy. A requirement specification may be incomplete or contain ambiguous information which requires extra analysis in order to extract the relational schema. To overcome this problem, relation extraction techniques including supervised and semi supervised methods will be used to build patterns, which can help us in extracting attributes, relations between tables, primary keys and other constraints from unstructured text. Specific domain knowledge could be used to extract a relational database from a problem description, for instance, related knowledge and information about education could be used to analyse the requirements for education systems.

APPENDIXES

A. Collapsed Dependencies Processed in Stanford CoreNLP for Store Problem

A store has many branches.
root(ROOT-0, has-3)
det(store-2, A-1)
nsubj(has-3, store-2)
amod(branches-5, many-4)
dobj(has-3, branches-5)

Each branch must be managed by at most 1 manager.
root(ROOT-0, managed-5)
det(branch-2, Each-1)
nsubjpass(managed-5, branch-2)
aux(managed-5, must-3)
auxpass(managed-5, be-4)
quantmod(1-9, at-7)
mwe(at-7, most-8)
num(manager-10, 1-9)
agent(managed-5, manager-10)

A manager may manage at most 2 branches.
root(ROOT-0, manage-4)
det(manager-2, A-1)
nsubj(manage-4, manager-2)
aux(manage-4, may-3)
quantmod(2-7, at-5)
mwe(at-5, most-6)
num(branches-8, 2-7)
dobj(manage-4, branches-8)

The branch sells many products.
root(ROOT-0, sells-3)
det(branch-2, The-1)
nsubj(sells-3, branch-2)
amod(products-5, many-4)
dobj(sells-3, products-5)

Product is sold by many branches.
root(ROOT-0, sold-3)
nsubjpass(sold-3, Product-1)
auxpass(sold-3, is-2)
amod(branches-6, many-5)
agent(sold-3, branches-6)

Branch employs many workers
root(ROOT-0, employs-2)
nsubj(employs-2, Branch-1)
amod(workers-4, many-3)
dobj(employs-2, workers-4)

The labour may process at most 10 sales
root(ROOT-0, process-4)
det(labor-2, The-1)

nsubj(process-4, labor-2)
aux(process-4, may-3)
quantmod(10-7, at-5)
mwe(at-5, most-6)
num(sales-8, 10-7)
dobj(process-4, sales-8)

It can involve many products.
root(ROOT-0, involve-3)
nsubj(involve-3, It-1)
aux(involve-3, can-2)
amod(products-5, many-4)
dobj(involve-3, products-5)

Each product includes product code, product name, size,unit_costand shelf_no
root (ROOT-0 , includes-3)
det (Product-2 , Each-1)
nsubj (includes-3 , Product-2)
nn (code-5 , product-4)
dobj (includes-3 , code-5)
nn (name-8 , product-7)
dobj (includes-3 , name-8)
conj_and (code-5 , name-8)
dobj (includes-3 , size-10)
conj_and (code-5 , size-10)
dobj (includes-3 , unit_cost-12)
conj_and (code-5 , unit_cost-12)
dobj (includes-3 , shelf_no-14)
conj_and (code-5 , shelf_no-14)

A branch is uniquely identified by branch_number.
root(ROOT-0, identified-5)
det(branch-2, A-1)
nsubjpass(identified-5, branch-2)
auxpass(identified-5, is-3)
advmod(identified-5, uniquely-4)
agent(identified-5, branch_number-7)

Branch has name, address and phone_number.
root(ROOT-0, has-2)
nsubj(has-2, Branch-1)
dobj(has-2, name-3)
dobj(has-2, address-5)
conj_and(name-3, address-5)
dobj(has-2, phone_number-7)
conj_and(name-3, phone_number-7)

Sale includes sale_number, date, time and total_amount.
root(ROOT-0, includes-2)
nsubj(includes-2, Sale-1)
dobj(includes-2, sale_number-3)
dobj(includes-2, date-5)
conj_and(sale_number-3, date-5)
dobj(includes-2, time-7)
conj_and(sale_number-3, time-7)
dobj(includes-2, total_amount-9)
conj_and(sale_number-3, total_amount-9)

Each labour has name, address and telephone.
root(ROOT-0, has-3)
det(labor-2, Each-1)
nsubj(has-3, labor-2)
dobj(has-3, name-4)
dobj(has-3, address-6)
conj_and(name-4, address-6)
dobj(has-3, telephone-8)
conj_and(name-4, telephone-8)

Worker is identified by id'.
root(ROOT-0, identified-3)
nsubjpass(identified-3, Worker-1)
auxpass(identified-3, is-2)
agent(identified-3, id-5)

B. Case Studies

ATM problem: Design the software to support a computerized banking network including both human cashiers and automatic teller machines (ATMs) to be shared by a consortium of banks. Each bank provides its own computer to maintain its own accounts and process transactions against them. Cashier stations are owned by individual banks and communicate directly with own bank's computers. Human cashiers enter account and transaction data. Automatic teller machines communicate with a central computer which clears transactions with appropriate banks. An automatic teller machine accepts a cash card, interacts with the user, communicates with the central system to carry out the transaction, dispenses cash, and prints receipts. The system requires appropriate recordkeeping and security provisions. The system must handle concurrent access to the same account correctly. The banks will provide their own software for their own computer; you are to design the software for the ATMs and network. The cost of the shard system will be apportioned to the banks according to the number of customers with cash cards.

Organization problem: Each department in an organization consists of a manager and several departmental staff. Each manager is in charge of only one department and departmental staff is assigned to a single department. Several projects are attached to each department. All departmental staff is assigned to projects, with some staff being assigned to several projects, not necessarily in the same department. Each project is run by a management group that consists of the manager of the department together with a selection of staff working on the project. No departmental staff member is ever required to sit on more than one management group.

Electrical Filling Program problem: An electronic filing program (EFP) can be used to store and retrieve text documents. Any document created by a word processor, editor or other means may be stored in the electronic filing system. Documents may be filed along with keywords, authors and/or document description or abstract describing the document. Documents field in the system may also be removed or deleted. Documents stored in the EFP are indexed to enable rapid retrieval. Documents are retrievable according to convenient schemes not found in conventional classifications; example users may retrieve or locate documents based on their content, description, author(s), or a user defined keywords. Therefore, the document description, authors, keywords,

and/or the actual text document itself may be searched. A user may specify a search criterion, which results in a number of documents being found that meet the specified search criteria. The user may then continue to specify additional search criteria, successively narrowing down the search until the required documents are found. Documents found that meet the search criteria may then be viewed or printed. The user is provided with the capability of specifying any extraneous or junk words which if found in the content of the document will not be searched or indexed. The user can also specify which alphanumeric characters will be indexed and searched (the filing character set), thereby limiting the search and index to only portions of a document(s).

Library System problem: When a library first receives a book from a publisher it is sent, together with accompanying delivery note, to the library desk. Here the delivery note is checked against a file of books ordered. If no order can be found to match the note, a letter of enquiry is sent to the publisher. If a matching order is found, a catalogue note is papered from the detail on the validated delivery note. The catalogue note, together with the book is sent to the registration department. The validated delivery note is sent to accounts department, where it is stored. On receipt of an invoice from the public accounts department checks its store of delivery notes. If the corresponding delivery note is found then an instruction to pay the publishers is made, and subsequently a cheque is sent. If no corresponding delivery note is found, the invoice is stored in pending file.

Journal Registration problem: The personal department of large research institute is responsible for the purchase and dissemination of journals to readers in the other department in the organization. Readers may be interested in certain specific topics relating to their research interests, while it is also possible to be pleased on a circulation list. Usually, reader het access to an issue of a journal for a fixed period of time, typically two weeks. It is possible to have access to an issue for a longer period of time, but permission must be granted from personal department. Journals appear on a regular basis each journal contain information on the publisher, language frequency of publication. The system should keep reader informed of the topics that are of interest to them and which appear in the different journals. Furthermore, it should be possible for reader to find articles which deal with topics they are interested in.

Local Hospital problem: A local hospital consists of many wards, each of which is assigned many patients. Each patient is assigned to one doctor, who has overall responsibility for the patients in his or her care. Other doctors are assigned on an advisory basis. Each patient is prescribed drugs by the doctor responsible for that patient. Each nurse is assigned to a ward and nurses all patients on the ward, though is given special responsibility for some patients. Each patient is assigned one nurse in this position of responsibility. One of the doctors is attached to each ward as an overall medical advisor.

REFERENCES

[1] P. P.-S. Chen, "The entity-relationship model_toward a unified view of data," ACM Trans. Database Syst., vol. 1, pp. 9-36, 1976.
[2] P. P.-S. Chen, "English sentence structure and entity-relationship diagrams," Information Sciences, vol. 29, pp. 127-149, 5// 1983.
[3] V. C. Storey and R. C. Goldstein, "A methodology for creating user views in database design," ACM Trans. Database Syst., vol. 13, pp. 305-338, 1988.
[4] A. M. Tjoa and L. Berger, "Transformation of requirement specifications expressed in natural language into an EER model," Proceeding of the 12th International Conference on ER-Approach, Airlington, Texas USA, 1994, pp. 206-217.
[5] F Meziane, "From English to formal specifications," PhD, Department of Mathematics and Computer Science, University of Salford, 1994.
[6] F. Meziane, S. Vadera, "Obtaining E-R diagrams semi-automatically from natural language specifications," presented at the Sixth International Conference on Enterprise Information Systems (ICEIS 2004), Universidad Portucalense, Porto, Portugal, 2004.
[7] E. Buchholz, H. Cyriaks, A. Düsterhöft, H. Mehlan, and B. Thalheim, "Applying a natural language dialogue tool for designing databases," in Proceedings of the First International Workshop on Applications of Natural Language to Databases. 1995.
[8] P. P.-S. Chen, "English, Chinese and ER diagrams, "Data & Knowledge engineering, vol. 23, pp. 5-16, 1997.
[9] H. M. Harmain and R. Gaizauskas, "CM-Builder: a natural language-based case tool for Object-Oriented analysis," Automated Software Engg., vol. 10, pp. 157-181, 2003.
[10] N Omar ,P. Hanna, P. M. Kevitt, "Heuristics-based entity relationship modelling through natural language processing," presented at the 15th Irish Conference on Artificial Intelligence and Cognitive Science (AICS'04), Castlebar, Ireland, 2004.
[11] L. A. E. Al-Safadi, "Natural language processing for conceptual modeling," International Journal of Digital Content Technology and its Applications, vol. 3, 2009.
[12] M. E. Elbendak, "Requirements-driven automatic generation of class models," PhD, School of Computing, Engineering and Information Sciences, Northumbria Univeristy, France, 2011.
[13] J. Slankas, "Implementing database access control policy from unconstrained natural language text," presented at the Proceedings of the 2013 International Conference on Software Engineering, San Francisco, CA, USA, 2013.
[14] C, D, Manning and etal, 2014." The Stanford CoreNLP natural language processing toolkit", In Proceedings of 52nd Annual Meeting of the Association for Computational Linguistics: System Demonstrations, pp. 55-60.