Chapter 1

Introduction

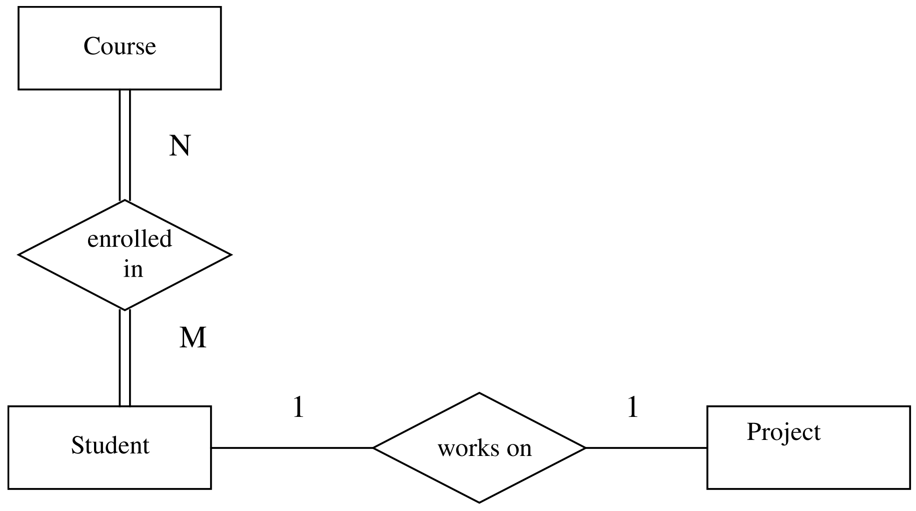
Entity-relationship (ER) modelling (Chen, 1976) focuses on high level conceptual models designed to facilitate database design in the development of information systems and relational databases. The success of the design of these models is measured by the level of accuracy with which they can reflect the real world environment (Dullea et al., 2003). Entity-relationship modelling can be a daunting task (Storey and Goldstein, 1988; Batra and Antony, 1994; Moody, 1996; Marsden and Staniforth, 1996; Antony and Batra, 2002) to both designers and students alike due to its abstract nature. However, the conceptual phase, which involves ER modelling, is considered to be one of the most critical tasks to the overall success of the system (Carolyn and Begg, 1999). Any errors, mistakes or inconsistencies incurred at this stage can be very costly later especially when a system has already been implemented. Boehm (1981) reported that the cost difference to correct an error in the early phases of software development as opposed to post-implementation phase is on the order of 1:100. In addition to the abstract nature of ER modelling, most of the input to this task involves natural language such as English, documented as requirements' specifications, which are also inherently ambiguous.

Much research has attempted to apply natural language processing (NLP) to extract knowledge from requirements' specifications with the aim of designing databases. Recent advances in this field suggest promising approaches that may assist database designers or novices learning database modelling concepts. Although research on NLP techniques in database design has been extensive (e.g. Eick and Lockemann, 1985; Tseng et al., 1992; Tjoa and Berger, 1993; Meziane, 1994; Buchholz et al., 1995; Burg et al., 1996; Gomez et al., 1999; Harmain and Gaizauskas, 2003; Meziane and Vadera, 2004), research on the formation and use of heuristics to aid the construction of logical databases from natural language has been scarce. In general, human experts draw on their own heuristics to decide whether an element should be represented as an entity or a relationship, for instance, in a conceptual model. It is desirable for a database design tool to be capable of imitating the way a human expert carries out the design task (Storey, 1993). The main goal of this thesis is to introduce new heuristics to assist the automated production of an ER model from a natural language specification. To realize the utilities of these heuristics, a tool called *ER-Converter* has been implemented. Results generated by ER-Converter are evaluated against human performance and also existing systems. This research can be utilized, for instance, as part of a domain model in an intelligent tutoring system (ITS) for Databases.

1.1 Overview of Data Modelling

The first step in designing a database application is to understand what information the database must store (Ramakrishnan, 1998). This is known as requirements analysis. The information gathered in this step is used to develop a high-level description of the data to be stored in the database, along with the data constraints. This step is referred to as conceptual design, and it may be completed out using the ER model.

An ER model is built around the basic concepts of *entities*, *attributes, relationships* and *cardinality.* An *entity* is an object that exists in the real world and is distinguishable from other objects. Examples of entities include a “student”, an “employee” and a “book”. These are typically derived from nouns. A collection of similar entities is called an *entity set*. An entity is described using a set of *attributes*. Attributes may be derived from adjectives and adverbs. The attributes of an entity reflect the level of detail at which we wish to represent information about entities. For example, the student entity set may have “id_number”, “name”, “address”, “course” and “year” as its attributes. A *relationship* is an association among two or more entities. Relationship can be derived typically from verbs in the requirements’ specifications. For example, we may have a relationship from this sentence: “A student may be enrolled in many courses”. The verb phrase “enrolled in” implies a relationship between the entity student and course. *Cardinality* represents the *key constraint* in a relationship. In the previous example, the cardinality is said to be many-to-many, to indicate that a student can take many courses and a course can be taken by many students. In an ER diagram, an entity is normally represented by a rectangle. An ellipse usually represents an attribute and a diamond shape shows a relationship. *Cardinality* is represented by '1' for one-sided, 'M' and 'N' for many-sided. For example, in a one to one relationship, the cardinality may be presented as 1:1 while in a many to many relationship, it may be represented as M:N. Figure 1.1 shows an example of an ER diagram. The double line



represents the participation of the relationship, i.e. whether it is optional or mandatory.

Figure 1.1. Example of an ER diagram using the Chen notation (Chen, 1976)

Conceptual database design produces a set of relation schemas and integrity constraints that can be regarded as a good starting point for the final database design. However, the relation schemas may suffer from inconsistencies and redundancies. Thus, given a relation schema, a decision has to be made as to whether it is a good design or if further decomposition is needed. The decomposition process into several classes of relational schemas that obey some set of rules is referred to as *normalization*.

1.2 Difficulty in ER Modelling

The difficulties in dealing with conceptual and logical design of databases have been well documented in past research studies (Storey and Goldstein, 1988; Batra and Antony, 1994; Moody, 1996; Marsden and Staniforth, 1996; Antony and Batra, 2002). Some educators also expressed their concern on whether they are teaching database design properly to students (Carpenter, 1992; Kleen, 1993). As most information systems need a reliable database, the correctness of database design is significant in the quality of these systems.

However, due to its complexity, database design can be error-prone, especially when handled by novice designers (Batra and Antony, 1994).

Batra and Antony (1994) have studied the processes employed by novice designers engaged in database design to gain an understanding of error causing factors. These factors are important for system developers and researchers in building tools and techniques that could prevent database design errors and thereby enhance the quality of information systems. Their study, which focussed on conceptual data modelling, showed that there are three factors that account for errors in database design:

1) Complexity of the task

Past experiments (Batra et al., 1990) suggested that novices face much more difficulty in modelling relationships rather than entities. One of the reasons stems from the fact that given a set of entities, there are potentially a very large number of possible relationships.

2) Use of heuristics that lead to biases

Heuristics are simple procedures which are often guided by common sense, meant to provide good but not optimal solutions to difficult problems, easily and quickly (Zanakis and Evans, 1981). In general, heuristics are often useful, but sometimes they may lead to severe and systematic errors called *biases*. An example of a useful heuristic is a requirement that specifies that the degree of a relationship should be a minimum (binary). Another useful heuristic is that if it has been determined that two entities have a binary relationship, they will not be involved in another higher degree relationship. Although these are valid most of the time, the designer should be careful to ensure that the heuristics do not lead to a bias in modelling the relationships.

3) Inexperience and incomplete knowledge of the novice

Given the restricted length of training, it is not surprising that a novice has limited knowledge and skills. Experts often draw from past experiences, but whether their experience can be applied to novices still remains a question. The important issue is how can the novices be trained *effectively* and *efficiently*?

Moody (1996) suggested four reasons why ER models are difficult to understand:

1. ER models "look" technical

To the average user, the meaning of an ER model is not obvious. In addition, they do not look very different from the other technical diagrams used in the system development process like network communication diagrams or system architecture diagrams – they consist of similar graphical representations such as geometrical shapes connected with lines.

2. The ER model does not handle complexity well

ER models are not able to cope with the large size and complexity of data models encountered in real world situations. When the number of entities becomes very large, the ER model becomes difficult to understand and manage.

3. Users find ER models abstract and difficult to relate to

Classification and *generalization* in ER models are two mechanisms that are used to derive entities. *Classification* is an abstraction used for grouping real world instances into classes or concepts. Entities (e.g. student, book) represent classes of things rather than instances. However, users often find abstract representations of requirements difficult to relate to and need concrete examples to understand what they mean.

Generalization is a mechanism to construct more abstract concepts based on similar properties (attributes or relationships) of more specific entities (Elmasri and Navathe, 2004). This is represented through the use of subtypes and supertypes. In Figure 1.2 Postgraduate and Undergraduate are subtypes of Student. Student is the supertype of Postgraduate and Undergraduate. The symbol 'o' means 'overlapping' which indicates that there may be cases where a postgraduate can also be an undergraduate, possibly in a different course. The symbol '$\cup$' means that a subtype is part of a supertype. For example, Postgraduate is part of Student. However users often find that highly generalised representations of data are difficult to understand. The level of abstraction of these models is a major barrier to their acceptance and understanding in practice.

4. ER models are focused on design rather than analysis

ER models have been used to define the structural aspects of data, for the purpose of design. The current modelling representations have been focused on effective design rather

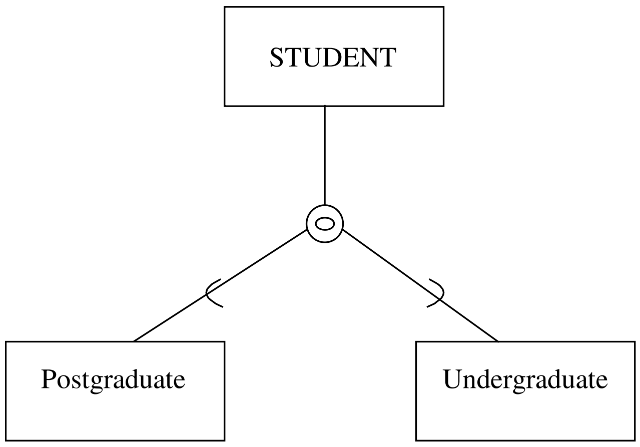


Figure 1.2. Representation of subtypes and supertypes

than effective analysis. This raises a question: for whose benefit are data models anyway? For the technicians or users? The reason for having ER models is to represent information requirements in a way that could be understood by users. It was not meant to describe the way in which data would be stored in the computer (Elmasri and Navathe, 2004).

The discussion above gives some evidence of the difficulty designers and students face in database design, specifically in Data Modelling. To further support this evidence, a survey has been carried out as part of this research to determine the perceived difficulty of the subject area. The motivation was to find out whether students still find Data Modelling a challenging subject.

1.2.1 Analysis on the difficulty of Database Modelling subject

A survey was carried out to determine how difficult students found the subject of Databases, particularly in the areas of Data Modelling. A questionnaire was used and is given in Appendix A.

The survey was conducted in the School of Computer Science at the Queen's University of Belfast. Thirty nine students participated in this study. They were undertaking the Masters course in *Computer Science and Applications*, a computing course for graduates with primary degrees in disciplines other than Computer Science. The questionnaires were distributed during the *Databases* lecture. At that time, the lectures in ER modelling and Normalization had been presented and the students had completed practical exercises on those subjects.

The majority of the participants (71.8%) had more than 3 weeks of experience in using databases. 81.6% of the participants had used INGRES, a database management system (DBMS). Most of the participants (92.3%) had used SQL (Structured Query Language), one of the query languages in databases.

When asked about the importance of the Database Systems course, 56.4% agreed that it is important while 25.6% of them thought that is a very important subject. Thus, it can be concluded that the majority of these students believe that the Database course is one of the important subjects in computing. Carpenter (1992) also states that formal database courses are needed for proper training of database designers to ensure a proper database design as this stage is a very critical stage in the development of an information system.

The participants in the survey were asked to rate the difficulty of the selected topics from Databases on a scale of very easy (1) to very difficult (4). Table 1.1 shows the results obtained:

Subject	*Very easy (%)*	*Easy (%)*	*Difficult (%)*	*Very difficult (%)*
Introduction to Databases	7.7	76.9	7.7	-
Entity-Relationship Modelling	2.6	48.7	48.7	-
Normalization	2.6	12.8	71.8	12.8
The Relational Model	2.6	25.6	71.8	-
SQL	5.0	48.7	41.0	2.6

Table 1.1: Difficulty of Database subject areas

With regards to the *ER Modelling*, nearly half of the students find the topic difficult. The participants were also asked a few questions on general understanding of both subjects. When asked whether they understand the basic concepts about entities, relationships and attributes, 74% of them find the concepts clear to them. However, when asked about their ability to construct an ER model for a given problem, only 31% reported that they are able to do it *most of the time,* another 59% could do it *sometimes* while 10% reported that they are s*eldom* capable of constructing it. 74% of them scan the sentences for nouns, verbs and other part of speech tags when they are determining the entities and relationships in an ER model. This is an important source of information as it provides some knowledge on how a similar task could be performed by an ER-Converter tool. This analysis also confirms that currently, students still perceive Databases to be a difficult subject which has been reported in the literature.

1.3 Objectives of research

The main aim of this research is to develop heuristics, algorithms and software to transform natural language input text of database problems into ER models. The primary objectives are summarized as below:

- Develop new heuristics to transform natural language specifications of database problems into Entity-Relationship (ER) models
- Design and implement, *ER-Converter*, a tool to assist the transformation
- Evaluate the approach against human performance and compare to other work in the field

The overall aim of this thesis is not to produce a fully automatic tool that will replace human analysts but rather to assist them by performing semi-automatic analysis of the requirements' specifications and produce an ER model for the analysts to review and refine. In the educational context, ER-Converter could serve as part of a dynamic domain model of an ITS.

1.4 Thesis Structure

Chapter 2 discusses previous work that applies natural language processing to database modelling, Intelligent Tutoring System (ITS) and ITS in Databases. The approaches and techniques used in processing natural language requirements' specifications for conceptual modelling are reviewed. As ITS is an area where this research can be applied, related ITSs in Databases are also reviewed.

Following on from this, issues relating to Natural Language Processing (NLP) in database design are discussed in Chapter 3. Problems encountered whilst processing natural language like the presence of ambiguities and solutions to them are presented. This chapter also elaborates on the parser used in this research, namely Memory Based Shallow Parser (MBSP) (Zavrel and Daelemans, 2003). This parser is used to tag words in natural language specifications to determine appropriate parts of speech (POS). The tagged text will act as an input to the ER-Converter tool.

The next step involves the development of a major focus in the thesis, i.e. to propose new heuristics to be utilized in the transformation from natural language to ER models. This is presented in Chapter 4. Existing heuristics in the literature are also discussed. Weights associated with each heuristic are also discussed. Before the proposed and existing heuristics were selected for implementation, a manual test was carried out to determine the contribution of the heuristics. The results on the training dataset are presented. The selection of the final set of heuristics is justified, based on a number of criteria.

Once the final set of heuristics is selected, the next step is to implement these heuristics to assess their utility in a practical environment in *ER-Converter*, a tool to transform natural language requirements' specifications to an ER model which is discussed in Chapter 5. Heuristics that have been selected as outlined in Chapter 4 are implemented in ER-Converter. Each of the steps involved in the production of an ER model are elaborated through sample sentences.

Having implemented ER-Converter, the results produced by the system are discussed in Chapter 6. The basic measures used in this research are *recall* and *precision*. New measures like *ask_ user* are also introduced in this chapter. ER-Converter produces favourable results though it requires limited human intervention.

Chapter 7 summarizes the work carried out in this research with comparison to other related work. The performance of ER-Converter is compared against other systems where figures are available. However, due to the different datasets used in the evaluation, direct comparison cannot be carried out. Future research directions are also presented in this chapter. Among the extensions suggested are the integration of WordNet and semantic interpretation.

Chapter 2

Literature review

This chapter surveys work on a range of systems that apply natural language processing in databases and Intelligent Tutoring Systems (ITS) for Databases. The ITS literature is reviewed since it provides one of the contexts where this research work could be applied.

2.1 Application of natural language processing (NLP) to database design

Natural languages are common tools for people to describe and communicate their understanding of the world. Because both ER diagrams (ERD) and natural languages satisfy similar human needs, their correspondence has been studied (Chen, 1983). Chen (1983) proposed some basic rules for translation between English sentences and ER models (ERM). A summary of the basic translation rules is shown in Table 2.1. These translation rules can be used in the conversion of an English language description of database requirements into ER models. Though this mapping can be performed by a human, there are certain limitations in machines which prevent them from carrying out this task. One of the reasons for this stems from the limitations of current technologies in NLP in matching human knowledge, for example in identifying a specific category for a word. For example, the words “Pat Clooney” can be easily identified as a possible candidate for an entity but is not easily identifiable by a machine. Thus, this mapping can only serve as a basis for a manual or semi-automatic process of transforming an English specification into an ER model (Chen, 1998).

Much work has tried to apply natural language to extract knowledge from requirements' specifications or dialogue sessions with designers, with the aim to design databases (Eick and Lockemann, 1985; Storey, 1988; Tseng et al., 1992; Tjoa and Berger, 1993; Meziane, 1994; Buchholz et al., 1995; Burg et al., 1996; Gomez et al., 1999; Harmain and Gaizauskas, 2003; Meziane and Vadera, 2004). The relevant tools and methodologies, which specifically analyse natural language requirements as input for conceptual design, are now discussed in the following sub-sections.

English Grammar	*Entity Relationship Model (ERM) Structure*
Common noun	Entity type (a possible candidate)
Proper noun	Entity (candidate)
Transitive verb	Relationship type (candidate)
Intransitive verb	Attribute type (candidate)
Adjective	Attribute for entity
Adverb	Attribute for relationship
Gerund (a noun converted from a verb)	An entity type converted from a relationship type
Clause	A high-level entity type which hides a detailed ERM

Table 2.1 Correspondence between English structure and ERM constructs (Chen, 1998)

2.1.1 ANNAPURNA

Eick and Lockemann (1985) proposed concepts, methods and tools to support the extraction, integration, transformation and evaluation of terminological knowledge (obtained from natural language statements) that is based on database design techniques in a project called ANNAPURNA. This project aims to provide a computerized environment for semi-automatic database design covering all phases from knowledge acquisition obtained from the experts up to generating an optimal database schema for a given database management system. ANNAPURNA concentrates on the phases concerned with acquiring the terminological rules. The term ‘terminological rules’ here refers to the terminology used to describe the universe of discourse (UoD). The need for acquiring the terminological knowledge was driven by the fact that different experts and the knowledge designer may use different terminologies and will represent rules concerning the same objects in a different way.

The first step in acquisition of the terminological knowledge involves extracting the knowledge from queries and rules that have the form of natural language expressions. The queries and rules are usually obtained from a user group who are assumed to share the same terminological knowledge. The knowledge obtained would then be put into the form of S-diagrams. Figure 2.1 shows an example of an S-diagram. An S-diagram is a graphical data model which can be used to specify *classes* (for example 'room', 'door' and 'physical_object'), *subclass connections* between classes (for example 'rooms' and 'doors' are 'physical_objects') and *attributes*, which describe the properties of members of the classes (for example 'from_room', 'to_room' and 'by_door'). An attribute has a *domain* class and a *range* class. For example, the attribute 'by_door' has the domain_class 'connect' and the range_class 'door'. Cardinalities can be associated with the attributes and may be restricted using the *labels* (represented by arrows) multivalued, unique, optional and onto. A double arrow, as shown in Figure 2.1, indicates a label 'onto'. An arrow with the letter 'S' shows a subclass connection. A tool, AISCHYLOS, which is part of the ANNAPURNA project, has been developed to generate S-diagrams from the grammatical structure of a natural language sentence using heuristic rules.

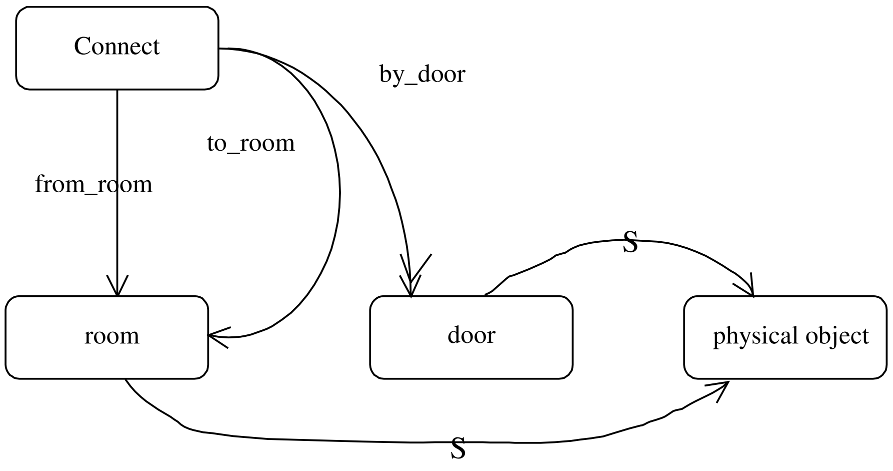


Figure 2.1. An example of an S-diagram (Eick and Lockemann, 1985)

Once the formalization is completed using S-diagrams, complete collective S-diagrams have to be derived for each user group from the individual S-diagrams (integration phase). To improve the process of knowledge integration, *quality* and *similarity* measures are used.

A *quality* measure will translate the S-diagrams to be integrated into a canonical form (by applying S-diagram transformations). This process guarantees that entities with similar structural properties will be described in the same way and entities that are structurally different will be represented in a different way. Similarity measures are used to detect synonyms, homonyms and subclasses. The limitation of this work is that the use of S-diagrams performs best when the complexity is small.

2.1.2 VCS (View Creation System)

VCS (View Creation System) (Storey, 1988) elicits the ER entities, attributes and relationships from the user by posing questions formulated in English. VCS formalizes, as a set of rules, a methodology for creating user views. These rules are encoded to form the knowledge base of VCS. VCS engages the user and poses queries in a dialogue session designed to elicit information requirements while simultaneously trying to detect and resolve inconsistencies and ambiguities. This task requires a considerable amount of user participation while the elicitation process is taking place.

The system selects primary keys from candidate keys using heuristics. All the entities and relationships are transformed into an initial set of relations. Each entity is then converted into a separate relation whilst each relationship is represented by a foreign key or separate relation (depending upon the cardinalities) (Storey, 1993). Once the relations are established, VCS, with the aid of the user, eliminates partial and transitive dependencies. The final output is a set of relations in the 4th normal form.

VCS's knowledge is represented as a set of procedural and production rules. It employs around five hundred rules, stored in the knowledge base. These rules are obtained from database design experts in consultation sessions in which they were asked to design a database for a hypothetical application. An example of a rule based on database theory is shown as follows:

IF: a relationship is of the form A is_a B

THEN: add the key of entity B to the set of candidate keys of entity A

VCS has been tested on eight real-world problems with users of varying skills in designing databases. The result shows that VCS is useful for users with some prior knowledge of database theory, but it did not perform well for those that did not. Another limitation is that VCS has a very limited capability in learning about which attributes are commonly used as keys in a particular domain. In terms of differing views from different experts in their own areas in an organization, VCS could perhaps consider resolving this through view integration in order to avoid bias.

2.1.3 Tseng et al. (1992)

Tseng et al. (1992) studied the inter-relationship between natural language constructs and the Entity-Relationship (ER) conceptual schema. A methodology is presented whereby it maps natural language constructs (in terms of queries) into relational algebra through an ER representation. The language processing follows three stages. First, the sentence is parsed according to a predefined grammar. Semantic roles are then built. These semantic roles are then mapped to an ER schema. The mapping is done by referring the corresponding verbs and nouns to the data dictionary. Each of the semantic roles is mapped into an entity relation and its *headnoun* and *modifiers* are mapped into the corresponding attributes of that entity based relation. A headnoun is a main noun in a phrase; for example, "The Irish supplier" has a headnoun 'supplier' and 'Irish' is the modifier of the headnoun. Verbs that relate to these semantic roles are then mapped into relationship relations that associate the entity relations. A logical form is developed by extending the ER representations to capture natural language semantics. This logical form can also be represented in a form similar to ERM and can be transformed into relational algebra. Figure 2.2 illustrates how a logical form can be represented as an ER schema from a natural language query, "List the suppliers who supply red parts".

The predicate "sname= ?" is defined as a pseudo predicate. It represents the target attribute which is to be output to the user. The natural language conjunctives *'and'* and '*or*' were mathematically analysed in Tseng's study. The logical form can finally be transformed into relational algebra for query execution.

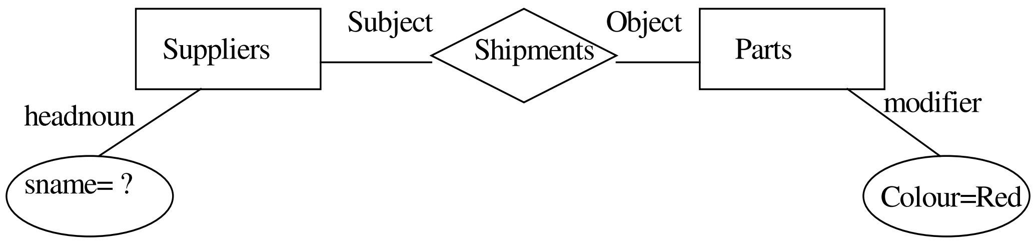


Figure 2.2. An example of the logical form (Tseng et al., 1992)

The methodology presented in this work argues that other intermediate forms suffer from bias to natural language constructs and much effort is needed to transform them into database query languages like SQL. Using the logical form in this methodology, the mapping using a representation similar to the ERD can efficiently transform a logical form to relational algebra. However, the data dictionary has to be changed every time a new database is used.

2.1.4 DMG (Data Model Generator)

Tjoa and Berger (1993) proposed a tool called Data Model Generator (DMG) which transforms requirements' specifications in natural language into concepts of an EER model. The transformation is based on the assumption that syntactic structures of the language can be translated into data modelling concepts. German was chosen as the input language.

DMG is a rule based design tool which maintains rules and heuristics in several knowledge bases. A parsing algorithm, which accesses information from a grammar and a lexicon, is designed to meet the requirements of the tool. During the parsing phase, the sentence is parsed by retrieving necessary information from the grammar, represented by syntactic rules and the lexicon. Word categories, word phrases and semantic roles are constituents which describe the sentence at different levels of detail. The syntactic structure of the sentence is represented graphically as a parse tree or by the flat structure of the linguistic concepts in the Linguistic Base. The parsing results are processed by rules and heuristics which set up a relationship between linguistic and design knowledge. DMG has to interact with the user if the word does not exist yet in the lexicon or the input of the mapping rules is ambiguous. In the *sentence transformation* phase, the parser triggers rules to determine *linguistic concepts* or *relationships*, which become entries in the Linguistic Base. The linguistic structures are then transformed by heuristics and rules into EER concepts. A selection of syntax and semantic-based heuristics to determine entity types, attributes, generalization hierarchies, relationships and cardinalities is presented in Tjoa and Berger (1993). An example below shows the appropriate heuristics applied to the given sentence:

"Every project has a project number"

H_E1: enttype: {project, project_number};
H_A1: project number → is attribute of → project;

H_E1 states that all nouns in the text are converted into entity types. H_A1 is a heuristic to determine attribute type. It states that if a sentence includes a main verb which is a modal verb with infinitive "have", then all nouns of the semantic role ACCOBJ (accusative object) are attributes of the noun which is part of the semantic role SUBJ (subject). DMG needs to interact with the user if there are any ambiguities or the input test does not represent the requirements of the user completely. Once the transformation rules and heuristics are applied, any occurring conflicts due to synonyms, homonyms or structural conflicts like connection traps are resolved by the designer before the final EER data model is produced.

Though the work has presented a selection of heuristics and rules for the transformation of natural language specifications to EER models, DMG has not been developed into a practical tool. The utility of the presented heuristics in DMG is not evaluated. User interaction can be extensive especially if the input text does not represent the complete requirements of the user.

2.1.5 FORSEN

FORSEN (Meziane, 1994; Meziane and Vadera, 2004) is an interactive approach for producing formal specifications from natural language requirements' specifications in English. The aims are to identify ambiguities present in natural language specifications and to identify the entities and relationships. The system first generates an entity-relationship

(ER) model from the input text. The entities and relationships are used as a basis for producing Vienna Development Method (VDM) (Jones, 1990) data types.

The ER entities and relationships are identified based on the view that nouns denote entities and verbs may indicate relationships. However, as this does not hold true for all cases, this method is inadequate. Arguments and the degrees of the relationships also need to be identified. Thus, the approach begins by using natural language techniques to translate sentences to a meaning representation called *logical form* language. The logical forms are the basis for identifying the entities and relationships. The quantifiers in the logical forms are used to identify suitable degrees for the identified relationships. Figure 2.3 shows the approach used by FORSEN in identifying ER models semi-automatically using formal specifications.

The translation process takes English sentences as inputs and produces the logical form in the following structure:

determiner (Base; Focus)

Nouns are usually represented as one place predicates. For example, "aircraft" is represented by *aircraft(X)*. For verbs, depending on their category, they may be represented by predicates having nil, one, two or three arguments. For example, the verb 'give' with three predicates can be represented as *give(X; Y; Z)*. The translation process is done in two phases. Firstly, a syntax analysis is performed to produce all possible parsings in terms of a syntax tree, according to the defined grammar. Next, the syntax tree is then transformed into a unique logical form. Once the logical forms are produced, the relevant ER elements are identified by mapping the suitable terms and predicates of the logical form.

One of the limitations of FORSEN is that it does not handle conjunctions or pronoun references. The requirements' specifications need to be analysed manually in order to solve these problems before FORSEN can be used. As sentences may be ambiguous, this may result in alternative logical forms. The analyst has to manually select the intended meaning before the task can be resumed. FORSEN also has not been evaluated in a practical environment.

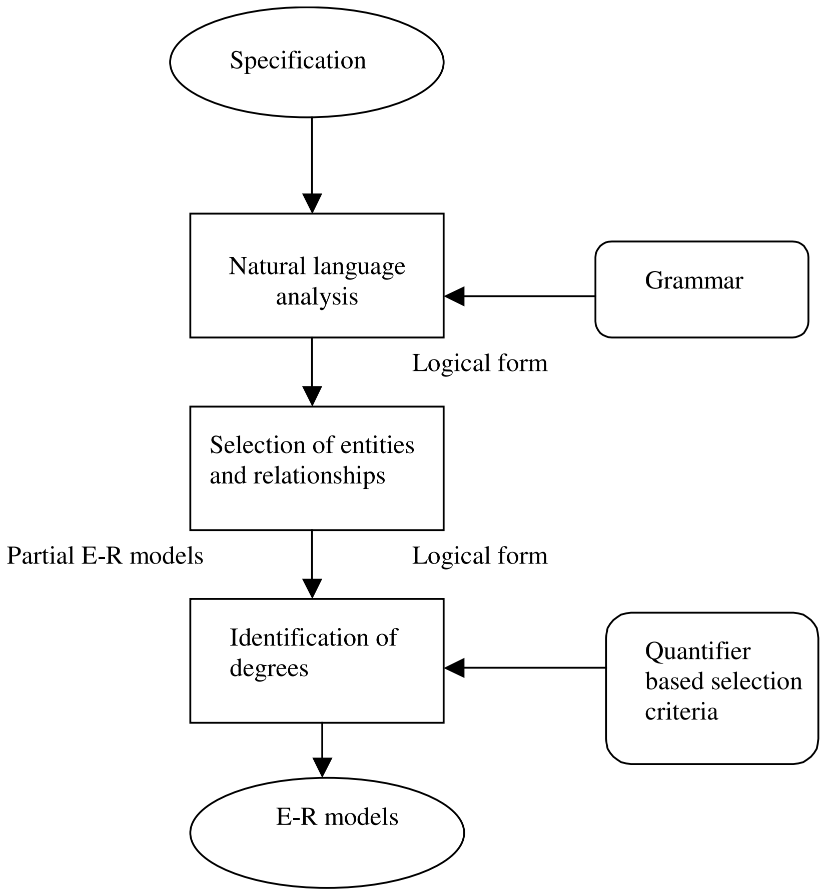

Specification
Natural language analysis
Grammar
Logical form
Selection of entities and relationships
Partial E-R models
Logical form
Identification of degrees
Quantifier based selection criteria
E-R models

Figure 2.3. FORSEN approach (Meziane and Vadera, 2004)

2.1.6 RADD (Rapid Application and Database Development)

Buchholz et al. (1995) developed a knowledge-based dialogue tool in German for producing a skeleton diagram of an Enhanced Entity-Relationship (EER) model. This tool is part of a larger database design system known as Rapid Application and Database Development (RADD), which consists of other components that form a complex tool. In order to obtain knowledge from the designer, a moderated dialogue is carried out during the design process. This moderated dialogue can be regarded as a question and answer session. During the session, the designer describes the structure of an application in natural language (German) and the dialogue tool reacts appropriately to each input sentence. The result of the syntactic, semantic and pragmatic analysis is used for controlling the dialogue. For example, if the designer's input is incomplete, a question will be initiated by RADD.

Once the knowledge is acquired through the dialogue, the input will undergo syntactic and semantic analysis. A special phrase structure grammar which uses the ID/LP (Immediate Dependence/Linear Precedence) format was developed for the syntactic analysis. The grammar formalism describes part of the German language based on analysis of user-input. The grammar analyses main and subsidiary clauses, relative clauses, prepositional clauses and basic verb phrases. The lexicon contains lexeme and morphological rules. A special parser has also been implemented which uses the grammar as well as the lexicon and transforms natural language input into syntax trees. The linguistic corpus was obtained by carrying out a number of interviews with librarians and library users ('Library' has been chosen as the domain knowledge base). It consists of more than 12,000 lexical units. Semantic analysis will then be performed to identify the meaning of sentences. A model of semantic roles based on Jackendoff's hypothesis (Jackendoff, 1983) is used for this purpose. It consists of the following roles which refer to the objects partaking in the action: Cause, Theme, Result/Goal, Source, Locative, Temporal, Mode, Voice/Aspect. The roles of a sentence are used to clarify linguistic completeness and to support the extraction of the design. The following example shows the semantic roles of the sentence; "The user borrows a book with a borrowing-slip":

Verb type: verb of movement (borrow)
Cause (subject): the user
Theme (object): a book
Mode: with a borrowing-slip

The transformation of the structure of natural language sentences into EER model structures is a process which is based on heuristic assumptions and pragmatic interpretation. The aim of the pragmatic interpretation is to map the natural language input onto an EER model structure using the results of syntactic and semantic analysis. Common rules are used for making general assumptions about how information gained from general sentences is related to entities, relationships, sets, keys and other EER structures. The results of the transformation processes are then transferred into a Data Dictionary which has been developed as part of RADD database design system.

One major limitation in RADD is that the accuracy of the EER model produced depends on the size and complexity of the grammar used and the scope of the lexicon. An extension of the lexicon is necessary to ensure a high level of accuracy of the result. Another open problem is the ‘integrity’ of the designer description of an application. A contradiction in the designer’s own views can cause conflict and affect the end result.

2.1.7 COLOR-X

In Burg and van de Riet (1996), a natural language and scenario-based approach to requirements engineering is proposed. In this context, a *scenario* refers to a sequence of events, describing the behaviour of parts of the system and its environment. Starting with an informal description of a scenario, formal event models are developed which reflect the information of the scenarios in a natural way. This is part of a larger project entitled COLOR-X (an acronym for COnceptual Linguistically based Object-Oriented Representation for Information and Communication Systems) which is based on strong linguistic theories and addresses both the issues of dynamic and static aspects of the system. COLOR-X has the main objective of generating object-oriented programming code from a natural language based modelling technique. The COLOR-X project is divided into several parts. Color-X Static Object Model (CSOM) represents the static models. Models define the overall structure of the system to be built by showing the objects, classes of objects and the relationships between them. The CSOM model contains the overall structure of the Universe of Discourse (UoD) for the programming code generator.

The formal event models are known as CEMs (COLOR-X event models). These mainly show the sequence of actions and events that take place in a particular UoD. The following example illustrates a trace of events that could and should be performed in the UoD. Figure 2.3 shows the corresponding CEM model:

Requirements document: “A user can borrow a book from a library. If the user has borrowed a book he has to return it within three weeks, before he is allowed to borrow a book again.”

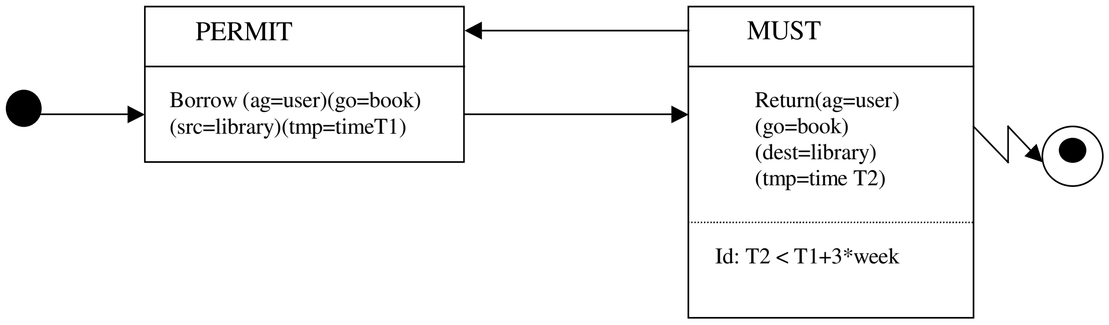


Figure 2.4. Example of a COLOR-X Event Model (Burg and van de Riet, 1995)

In Figure 2.4, a box represents an event that could or should take place, a straight arrow represents the actual occurrence of the event and a 'lightning' arrow shows the fact that the specific event did not take place. Modality of the sentences can be represented through the PERMIT-box and MUST-box. The PERMIT-box is triggered by the words 'can' and 'allowed to' in the requirements document. The MUST-box is caused by the word 'has to'. A MUST-event requires two outgoing arrows to succeeding events: the obligatory event has taken place or the obligation is violated. Since in the example there is no event specified that has to be completed when the book is not returned within three weeks, the outgoing 'lightning' arrow ends in an end-node (Burg and van de Riet, 1995). Table 2.2 shows the general form of the CPL (Conceptual Prototyping Language) specification language used in the CEM example:

CPL specification language	*Meaning/ Values*
Mode	FACTUAL I MUST I NEC IPERMIT
Tense	ACTION I DONE I PROSP I PERF I PRET
Predication	A relation between n terms $T_1 \ldots T_n$
T_i	A term denotes a (set, with cardinality *c*, of) object(s)
id	Identification of the objects
sit	Situation in which this CPL specification is supposed to hold

Table 2.2. General form of CPL specification language

The event models model the dynamics of the system as a whole, and their contents are paraphrased back into natural language sentences. Natural Language paraphrasing is used as one of the techniques to validate requirements. This similar paraphrasing approach is also used by Rolland and Proix (1992). Burg and his colleagues claimed that this technique tackles the problem caused by two conflicting concerns in requirements engineering namely the concern of the analyst to develop a formal requirements model and the user's need to communicate these requirements in their own terminology. A tool, CPL2nl (Conceptual Prototyping Language to natural language) has been developed to carry out the natural language paraphrasing task. This tool generates natural language sentences from specifications of CPL, a formal conceptual modelling technique. The output forms the underlying representation of CEMs. The validated CEM models are used as a basis for analysis and design models.

COLOR-X stems on strong linguistic theories and addresses both the dynamic and static aspects of systems through the utilization of CEM and CSOM models. One of the limitations however, is that though the transformation from informal scenarios into formal CEM models is supported with a lexicon, the process itself is not automated. A human analyst has to manually complete the creation of CEMs like defining the events and dependencies between events, though this task is supported by a CASE (Computer Aided Software Engineering) tool.

2.1.8 E-R Generator

E-R Generator (Gomez et al., 1999) is a rule-based system that takes natural language specifications as inputs and generates ER model elements such as entities, attributes and relationships for small domain applications. ER-Generator is part of a larger system which also consists of a parser and semantic interpreter known as a Natural Language Understander (NLU). Sentences entered by the user are parsed and semantically interpreted by the semantic interpreter, which then outputs the final knowledge representation structure. These final knowledge representation structures act as input to E-R Generator.

Figure 2.5 depicts the main components of E-R Generator. The E-R Generator consists of two kinds of rules: *specific rules* and *generic rules*. These rules are two major sources of knowledge used by the E-R Generator. *Specific rules* are linked to the semantics of some words in sentences. They are defined for a *verbal concept*, or predicates when its semantics

indicate that an action specific to the concept must be performed by the E-R Generator. The predicates are referred to as *NL-relations*. NL-relations may have one, two or more arguments. The arguments refer to the thematic roles of the NL-relation. They may be nouns or NL-relations themselves. All arguments that refer to physical objects like

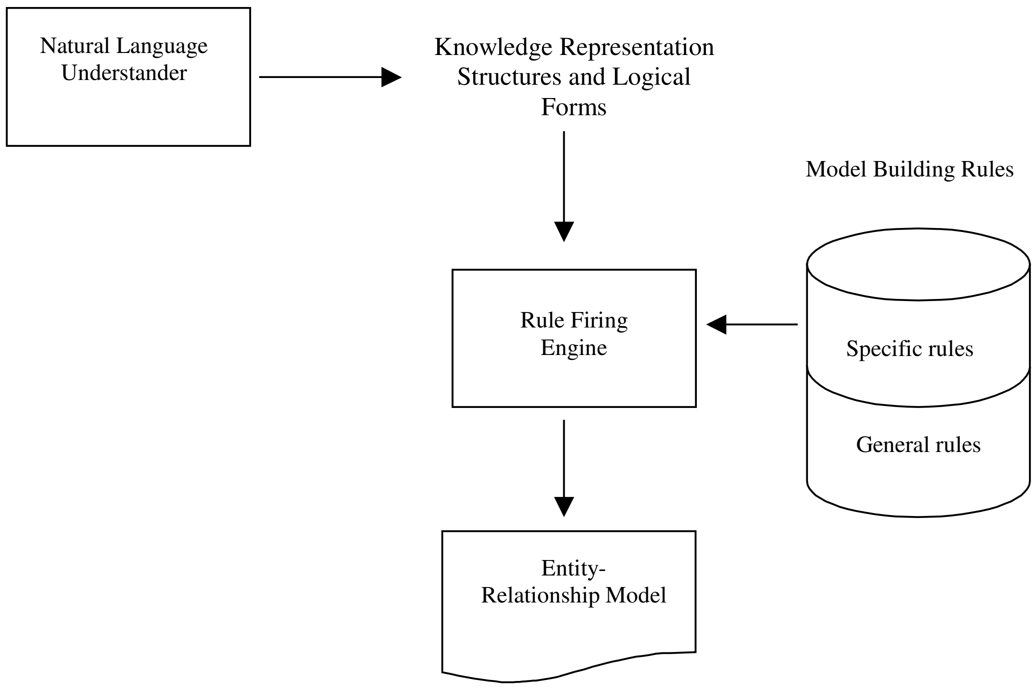

Natural Language Understander
Knowledge Representation Structures and Logical Forms
Model Building Rules
Rule Firing Engine
Specific rules
General rules
Entity-Relationship Model

Figure 2.5: The E-R Generator approach (Gomez et al., 1999)

'employee' are called *object structures*. Examples of specific rules are those that construct hierachical ER relationships among ER entities like the *is-a* verbal concept, or those that identify key attributes. The following example illustrates how a specific rule for defining hierarchical relations is established:

"Each person keeps a record of documents of interest. Documents may be books, identified by author, name and title, journal articles, identified by journal volume, number, author name, title and private correspondence, identified by sender and date."

E-R Generator creates the ER entities 'document', 'book', 'journal article' and 'private correspondence' and establishes the relations:

- book is-a document-of-interest
- journal article is-a document-of-interest
- private correspondence is-a document-of-interest

These relations are not translated into relationships, but are maintained by E-R Generator to keep track of the inheritance of attributes among ER entities.

Generic rules, on the other hand, identify ER entities and relationships on the basis of the logical form of the sentence and on the basis of the ER entities and relationships under construction. There are three types of generic rules: *unary*, *binary* or *n-ary* rules. In general, *unary* rules result in the definition of attributes; *binary* rules may define attributes, entities and relationships while *n-ary* rules result in the definition of relationships. Most sentences in a database description introduce *binary* ER relationships. Table 2.3 shows the binary rule cases. For example, in case 2, suppose there exists two entities, 'company' and 'books' in the database and no relation exists between them. If a user enters the sentence "the company sells books", E-R Generator creates the relationship 'sell' and relates it to 'company' and 'books'. Similarly, if both 'company' and 'books' do not exist in the database, E-R Generator then creates 'company' as an entity whilst 'books' becomes the attribute of 'company' (case 10).

E-R Generator identifies the entities, relationships and attributes based on the representation structures built by the NLU and on the current state of the database design. The task is carried out by accessing the structures that represent NL-relations, *a-structures* and the hierarchical NL-relation forms, *object structures*. All the structures are examined in two passes. In the first pass, some structures may result in the generation of ER elements. In the second pass, the saved structure from the first pass that caused no action may be considered here by some rules, particularly the unary rules. Specific rules are tried first before the generic rules. The generic rules are fired regardless of the verbal concept of an NL-relation. Their actions are based on the arguments of the NL-relation and on the entities and relationships currently defined in the database model.

	Argument 1	*Argument 2*	*Relation*
Case 1	Entity	Entity	Yes
Case 2	Entity	Entity	No
Case 3	Attribute	Entity	No
Case 4	Does not exist	Entity	No
Case 5	Entity	Attribute	No
Case 6	Attribute	Attribute	No
Case 7	Does not exist	Attribute	No
Case 8	Entity	Does not exist	No
Case 9	Attribute	Does not exist	No
Case 10	Does not exist	Does not exist	No

Table 2.3: Binary rule cases (Gomez et al., 1999)

One of the limitations of E-R Generator is that it may need user intervention in order to resolve ambiguities. This includes requesting assistance following concepts in the hierarchy, attaching attributes and confirming the suggestion of key attributes. The amount of user interaction increases depending on the type of the problem such as ambiguity caused by intersentential anaphora. Another limitation is the lack of background knowledge to describe the database application that E-R Generator needs as the techniques are based on semantic interpretation.

2.1.9 CM-Builder (Class Model- Builder)

CM-Builder (Harmain and Gaizauskas, 2003) is a natural language based CASE tool which aims to support the analysis stage of software development in an object-oriented framework. The tool uses natural language processing techniques to analyse software requirements documents and produce initial conceptual models represented in Unified Modelling Language (Booch et al., 1999).

There are two versions of CM-Builder, i.e. CM-Builder 1 and CM-Builder 2. In CM-Builder 1, a significant amount of user interaction is needed to select the correct candidate classes, attributes and relationships. This has been improved in CM-Builder 2 where the system uses several modules to process the natural language specification text to produce a conceptual model without user interaction. The *OOA (Object-Oriented Analysis) Module* basically converts all nouns into candidate classes and verbs into relationships. For every candidate class, its frequency in the text is considered before the item is selected. The most frequent candidates are the most likely classes. Attributes are found from simple heuristics like possessive relationships and the use of verb phrase like 'to have'. However, no detailed information is given in the literature on the heuristics applied in determining the attributes. WordNet (Fellbaum, 1998) is also employed to help determine attribute names from adjectives.

CM-Builder still has some limitations in its linguistic analysis. For example, attachment of postmodifiers such as prepositional phrases and relative clauses is limited. Other shortcomings include the state of the knowledge bases which are static and not easily updateable nor adaptive.

2.2 Summary of systems that apply NLP to database design

All the systems and methodologies reviewed utilize various natural language processing techniques in designing databases. Table 2.4 presents a summary of the systems reviewed, their aims, target users and the techniques used. Approaches like ANNAPURNA (Eick and Lockemann, 1985), Tseng et al. (1992) and FORSEN (Meziane, 1994) are based on simplification that the input language is formalized. One of the disadvantages is the limited expressiveness of formal representations (Tjoa and Berger, 1993). The user is restricted in putting forward his views of the UoD as he is unable to convey the knowledge using natural language. It may also be time consuming to adhere to the strict rules and sophisticated data abstraction considerations (Tjoa and Berger, 1993). However, these approaches may help to eliminate problems like ambiguities in natural language.

Other systems like E-R Generator (Gomez et al., 1999) and CM-Builder (Harmain and Gaizauskas, 2003) produce conceptual models directly from natural language requirements' specifications. Although this approach may suffer from ambiguities, fuzziness and redundancy of natural language, the advancement in NLP techniques could improve performance. VCS (Storey, 1988) and RADD (Buchholz et al., 1995) hold dialogue sessions with experts or database designers in eliciting user views. One disadvantage of the approach is the amount of the interaction time needed between the user and the system. As

System	*Aim*	*Type of user*	*Techniques used*
ANNAPURNA (Eick and Lockemann, 1985)	To provide a computerized environment for semi-automatic database design	Database Designer Expert of Universe of Discourse (UoD)	▪ S-Diagrams ▪ Heuristics
View Creation System (VCS) (Storey, 1988)	To provide an interactive system for eliciting user views	Database designer End user	▪ Procedural and production rules ▪ Heuristics
Tseng et al. (1992)	To map natural language contructs into relational algebra through ER representation	Database Designer	▪ Logical forms
DMG (Tjoa and Berger, 1993)	To support designer in extracting knowledge from requirements' specifications	Database Designer	▪ Rules ▪ Heuristics ▪ Dialogue
FORSEN (Meziane, 1994)	To obtain ER models from language specification	Database designer	▪ Logical forms
Dialogue Tool (RADD) (Buchholz et al., 1995)	To obtain a skeleton design of EER model from designer	Database Designer	▪ Dialogue ▪ Syntactic analysis – ID/LP format ▪ Semantic analysis – using Jackendoff's hypothesis ▪ Heuristics ▪ Attribute Grammar ▪ Pragmatic interpretation
COLOR-X (Burg and van de Riet, 1996)	To facilitate the process of generating conceptual modelling	Database designer	▪ CEMs ▪ CSOMs ▪ paraphrasing
E-R Generator (Gomez et al., 1999)	To generate ER models from natural language specifications	Database Designer	▪ Rules ▪ Semantic interpretation
CM-Builder (Harmain, 2000; Harmain and Gaizauskas, 2003)	To build object-oriented conceptual models	Systems Analyst	▪ Frequency analysis ▪ Discourse interpretation

Table 2.4 Systems that apply NLP to database design

the problem becomes more complex, the user may have to spend quite a considerable time answering various questions posed by the system. In comparison, a computerized system has 'patience' though bombarded with many problems to be solved. However, as noted from all the systems reviewed in Table 2.4, full automation in designing databases is almost impossible due to the abstract nature of the problems and ambiguities in natural language.

Heuristics, based on linguistic rules, are reported to be utilized in many of the systems like ANNAPURNA (Eick and Lockemann, 1985), VCS (Storey, 1988), DMG (Tjoa and Berger, 1993) and RADD (Buchholz et al., 1995). However, only DMG (Tjoa and Berger, 1993) presents a precise set of heuristics used in deriving an EER model. What appears to be lacking in most of the systems reviewed is formal evaluation or testing to verify their usefulness in the context of a real world application. CM-Builder (Harmain and Gaizauskas, 2003) performs a formal evaluation in terms of *recall* and *precision*, to validate the results of the output.

2.3 Heuristics in database design

The word "heuristic" is derived from the Greek "heuriskein", meaning to "discover" (Zanakis and Evans, 1981; Groner et al., 1983). This suggests that a heuristic may be applied to something requiring exploration or investigation or to a chance encounter. To practitioners, heuristics are simple procedures, often guided by common sense, that are meant to provide good but not necessarily optimal solutions to difficult problems, easily and quickly (Zanakis and Evans, 1981). A slight distinction needs to be made between *heuristics* and *rules* as both are reportedly applied in database design tools (e.g. Eick and Lockemann, 1985; Tjoa and Berger, 1993; Buchholz et al., 1995; Gomez et al., 1999). *Rules* represent a definite assumption (Tjoa and Berger, 1993) or can be theoretically derived under certain assumptions (Batra and Zanakis, 1994). *Heuristics* are largely "rules-of-thumb" based mainly on observations, common sense, intuition and experience (Batra and Zanakis, 1994) or serve as an aid to learning, discovery, or problem-solving by experimental and especially trial-and-error methods (Merriam-Webster Dictionary, 1997).

Zanakis and Evans (1981) gave several instances where the use of heuristics is advantageous. One of these instances is that heuristics are *simple* and easily understood by users and therefore likely to be implemented. Another example is that a heuristic solution is "good enough" if it produces results better than those currently realized.

In the context of database design, heuristics are applied mostly in conceptual design (Batra and Zanakis, 1994), extracting knowledge from requirements' specifications (Eick and Lockemann, 1985; Tjoa and Berger, 1993) or through dialogue sessions with designers (Buchholz et al., 1995) to model Entity-Relationships and in the refinement of pre-physical database design or schemas (Cerpa, 1995; Rosenthal and Reiner, 1994). More recently, a heuristics-based methodology has been proposed for creating and managing ontologies for the development of database designs (Sugumaran and Storey, 2002). It is evident that heuristics-based approaches are gaining popularity as a means of solving problems in database design.

Research on the formation and use of heuristics to aid the construction of logical databases structures from natural language has been scarce. DMG (Tjoa and Berger, 1993) proposes a large number of heuristics to be used in the transformation from natural language to ER models. However the work has not yet been developed into a practical tool. Tjoa and Berger (1993) proposed both syntactic and semantic heuristics to be applied in extracting knowledge from requirements' specifications. Although E-R Generator (Gomez et al., 1999) and RADD (Buchholz et al., 1995) utilized heuristics in their work, they do not detail the precise set of heuristics used in their approach. Chen (1983) suggested that the basic constructs of English sentences can be mapped into ER schemas in a natural way and presented a set of rules. Though the set are referred as "rules", Chen mentioned that they are better viewed as "guidelines" as it is possible to find counter examples to them. Chen's "rules" are therefore regarded as heuristics as they are largely "rules-of-thumb" based on observations rather than theoretically derived.

When dealing with heuristics, many decisions are based on beliefs concerning the likelihood of a certain event happening (Kahneman et al., 1982; Griffin and Tversky, 2002). These beliefs are commonly expressed in statements such as "I think that…", "Chances are that …" and so forth. Some beliefs are expressed in numerical form as odds or subjective probabilities (Kahneman et al., 1982). For example, MYCIN (Buchanan and Shortliffe, 1984) uses certainty factors to indicate the strength of a fact. MYCIN uses certainty factors as an alternative to probabilistic reasoning. Certainty factors range from –1 (definitely false) to +1 (completely true).

2.4 WordNet

WordNet (Fellbaum, 1998) is an on-line lexical reference system which differs from the standard dictionary in which English nouns, verbs, adjectives and adverbs are organized into synonym sets, each representing one underlying lexical concept. It groups English words into sets of synonyms called *synsets*, provides brief definitions, and maintains the various semantic relations between these synonym sets. WordNet aims to produce a combination of a dictionary and a thesaurus that is more intuitively accessible, and to support automatic text analysis and artificial intelligence applications.

WordNet is comprised of four parts: the lexicographers' source files, software known as Grinder to convert these files into the WordNet lexical database, the WordNet lexical database and a suite of software tools to access the database. All word forms are arranged into synsets. These are further organized into a set of lexicographers' source files by syntactic category like noun, verb, adjective and adverb. Each word form is known by its orthographic representation, syntactic representation, semantic field and sense number. Relational pointers, which can be *lexical* or *semantic* are created to represent the relations between the word forms. *Lexical* relations can exist between relational adjectives and the nouns they relate to and between adverbs and adjectives from which they are derived. *Semantic* relations that represent a relation between meanings are appended to the list of word forms in the synset.

In conceptual modelling, WordNet has been utilized, for example in Color-X (Burg and van de Riet, 1998) and CM-Builder (Harmain and Gaizauskas, 2003) to assist the user in determining the meaning and context of a word. In Color-X, for instance, WordNet can be used to disambiguate the meaning of a verb by examining synonyms. For example, the verb 'sell' would result in a few senses and the user needs to select the suitable meaning in its context. In CM-Builder, WordNet is used to identify hidden attributes that may arise from adjectives.

2.5 Intelligent tutoring systems (ITSs)

One context where ER-Converter can be applied is in the area of Intelligent Tutoring System (ITS) (Sleeman and Brown, 1982), as part of the domain model. This section provides a brief overview of ITSs and its components namely the domain model, the tutor model, the student model and the user interface.

Computer-based instruction systems that display some degree of "intelligence" have been used in education for over 20 years. Computer-Based training (CBT) (Dean and Whitlock, 1992) and Computer-Aided Instruction (CAI) (Self, 1988) were among the first such systems that were introduced to teach students using computers. While both CBT and CAI may seem to be effective in helping learners, they are incapable of providing individualized attention and feedback as a human tutor could have given to the students. Thus, a new field of research has emerged known as intelligent tutoring systems (ITSs).

An intelligent tutoring system (ITS) is a software system that uses artificial intelligence (AI) techniques to tutor people in a given domain. The goal of intelligent tutoring systems is to provide a learning experience for each student that approaches the standard of learning that he/she would receive from a human tutor. To achieve its goal, intelligent tutoring system software monitors each student's interactions and builds a 'student model' for each individual. This model comprises the student's performance on training/problem-solving and remediation exercises; knowledge of all information and remediation received; the knowledge mastered, failed and misunderstood by the students; and the student's learning style. Apart from the student model, two other important models in an ITS include the domain model and the tutor model. The domain model represents the knowledge of the subject area while the tutoring model contains methods on how to select, sequence and present materials to the students. A more detailed discussion of these three models is presented in the following section.

ITS systems are also intended to facilitate *learning-by-doing*: transforming factual knowledge into experiential knowledge. They attempt to combine the problem-solving experience and motivation of 'discovery' learning with the effective guidance of tutorial interactions. To enable this, the system must have its *own* problem-solving expertise, its own diagnostic or student modelling capabilities and its own explanatory capabilities. In order to orchestrate these reasoning capabilities, it must also have explicit control or tutorial strategies specifying *when* to interrupt a student's problem-solving activity, *what* to say and *how* best to say it; all in order to provide the student with instructionally effective advice (Sleeman and Brown, 1982).

2.5.1 Components of an ITS

An intelligent tutoring system should comprise the following four components: the domain model, the tutor model, the student model and the user interface (Burns and Capps, 1988). Figure 2.6 shows the relationships between the main components of an ITS.

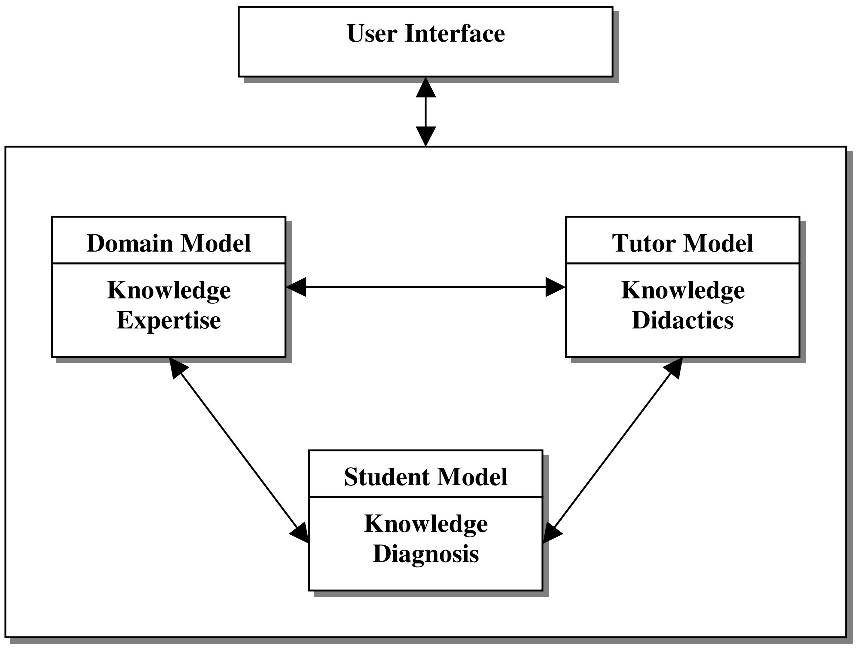


Figure 2.6: The Components of an ITS

The domain model

The domain model (sometimes referred as the *expert model*) contains knowledge of the specific domain to be taught. It forms the backbone of any intelligent tutoring system

(Anderson, 1988) as it provides the domain intelligence. The intelligent tutoring system uses its domain knowledge to reason about and solve a problem posed by a student or set by the system. The knowledge or expertise has to be encoded and represented in such a way that it supports reasoning that resembles the human problem-solving process within the teaching domain (Siemer and Angelides, 1998).

Three approaches have been used to encode knowledge in the domain model. These are known as black box models, glass-box models and cognitive models.

Black-box models

A black box model is a way of reasoning about the domain without requiring any codification of the knowledge that underlies human intelligence. It can be used as a judge of correctness as it generates correct input-output behaviour over a range of tasks in the domain. The classical example of a black box model is the unique work on SOPHIE (Brown et al., 1982). The underlying circuit simulator, SPICE, a black box expert, was used to determine the reasonableness of various measurements that a student would make in troubleshooting faulty electronic circuits. This expert is used only to check the consistency of a student's hypotheses and answer some of his questions. However, its mechanisms are concealed from the student since they are not the mechanisms the student is expected to learn (Burton and Brown, 1982).

Glass-box models

The second approach to encoding knowledge in the domain model is using the glass-box model. This approach uses knowledge-engineering techniques to control the tutorial mechanisms of the system. A rule-based formalism is usually used to represent the knowledge. The implementation does not necessarily correspond to the way a human expert reasons. It allows only for explanations of the information process inherent in the rules of its knowledge base. An example of this glass-box model is GUIDON (Clancey, 1982), an ITS which teaches physician consultancy. GUIDON uses MYCIN, an expert system for diagnosing bacterial infections as the domain model within an ITS. Some difficulties arose from this project. For example, the actual reasoning process used by MYCIN to deploy its knowledge, an in-depth backward search, is not the way the knowledge is deployed by humans. In addition, the highly compiled rules of MYCIN were difficult for GUIDON to understand and too complex to be directly taught to novices.

Cognitive models

A lesson learned from the GUIDON project is that for tutoring systems to be effective, the way the knowledge is deployed is equally important. The domain model must deploy its knowledge according to the way a human does. This principle leads to the cognitive modelling approach. The goal of the cognitive modelling approach is to develop a simulation of problem solving in a domain in which the knowledge is composed into meaningful, human-like components and deployed in a human-like manner. In this way, the system can communicate the domain knowledge clearly to the student. Cognitive domain models distinguish between three types of domain knowledge: procedural, declarative and qualitative. Procedural knowledge is concerned about *how* to perform a task such as mathematical problem solving. Meanwhile, declarative knowledge conveys knowledge in the form of a set of *organized facts* to enable human reasoning. For example, there are domains like geography where the tutorial goal is to convey declarative knowledge in the form of a set of facts appropriately ordered so that one can reason with them. Qualitative models allow one to reason about behaviour using mental models of systems such as when troubleshooting an electronic device.

Procedural knowledge in a cognitive domain model usually takes the form of a rule-based production system. BUGGY (Burton, 1982) and the LEEDS modelling system (Sleeman, 1982) are among the systems that use this rule-based approach. These systems involve a set of if-then rules matched to a working memory of facts. This working memory resembles the short-term memory of a human. Declarative knowledge representations are useful when there is a need for the student to understand the principles and facts of a domain and applying them. However, this does not mean that the aims of procedural and declarative tutoring are mutually incompatible. Sometimes the nature of the subject matter requires the student to be facile about the rules of a problem domain but clear about the justifications for the rules. This can be the case in the domain of medical diagnosis (Clancey, 1982).

The student model

An intelligent tutoring system is a computer program that instructs the student in an intelligent way. It infers a model of the student's current understanding of the subject matter and uses this model to adapt the instruction to the student's needs (VanLehn, 1988). This model is referred to as the *student model*. Student modelling is the most important part of an ITS since the student has the central role in the teaching process (Stankov, 1996). The behaviour of an ITS depends largely on the student model, which in turn depends on the domain model for the diagnosis of the student's knowledge. An ideal student model should contain the aspects of the student's knowledge, feeling and behaviour that might affect the student's learning (Tong, 1997).

The tutoring model

An intelligent tutoring system should display various tutoring characteristics. These are contained in the tutoring model. The characteristics of the tutoring model include (Halff, 1988):

a) controls over the tutorial discourse, i.e. the representation of the instructional knowledge for sequencing and selecting the appropriate materials of the subject matter,
b) the ability to respond to the student's queries about instructional goals and content,
c) strategies to offer help when needed and ways of delivering it.

The goal of this model is to circumscribe the nature of tutoring and to implement it as a solution to the educational problem. The central issues in the tutoring model are the problems of developing methods for selecting and sequencing material and methods for presenting it.

The user interface

The user interface acts as a front-end to an ITS and it provides a means of interaction between the student and the system. The aim of the user interface is to enhance '*conversation*' between the system and the student to facilitate the communication of knowledge between both parties (Tong, 1997). Miller (1988) emphasizes the need to make appropriate tradeoffs in the design of ITSs due to several issues. First, the learner working with an ITS must learn some subject matter that he or she may not understand. An ITS should not complicate the matter by having a complex interface to deal with. If the user interface is poorly designed, the tutoring session will probably be ineffective. The goal of interface design is therefore to make the interface transparent.

2.6 ITSs for database design

There are several existing intelligent tutoring systems for the database domain. However, not many of them focus specifically on the topic of Data Modelling. The following sub-sections review each of the ITS systems.

2.6.1 DB-Tutor

DB-Tutor (Raguphati and Schkade, 1992), an intelligent tutoring system for database design, was developed using hypertext. It is designed to assist users in database design by providing examples and graphics to illustrate database design techniques. Here ‘database design’ refers to the ability of the database designer to apply a number of guidelines and rules-of-thumb in designing a database. This sometimes involves creativity and use of heuristics in arriving at a good design. The domain was restricted to conceptual database design using the relational model. The information on databases was sorted into related topics and presented in a nonlinear manner using hypertext in the form of nodes and links. The idea of hypertext is to link related information together, irrespective of its location. A node is a continuous flow of text. All terms within a topic that referenced another topic were represented as links. For example the two topics ‘Normalization’ and ‘Relational Models’ were represented as a link from one topic to the other.

DB-Tutor was implemented using a hypertext software tool. Only three of the primary components of ITS were present in the system; the domain model, which contains the information on database design, the user interface and the tutoring model, which provides facilities for the presentation of the information. With the absence of the student model, an important component of an ITS, this system is incapable of monitoring the student’s progress and current understanding of the subject matter. As noted previously, student modelling is the most important part of an intelligent tutoring system since the student has the central role in the teaching process (Stankov, 1996).

2.6.2 Canavan (1996)

Canavan (1996) developed a prototype for an intelligent tutoring system in database design, in the topic of Normalization for advanced UK GNVQ (General National Vocational Qualification) level students. The prototype was partially built as part of her investigation into how far intelligent computer aided instruction (ICAI) and intelligent tutoring system (ITS) can be brought to bear on the problems of education. The prototype was limited to the test and tutorial element of the ITS due to time constraints and limited resources.

Figure 2.7 shows the outline of the initial prototype. It is a menu-based system where users are presented with the appropriate screens during the tutoring session. To start with, the user is presented with an options menu from which he may choose any of the nine tutorials or exercises. For the tutorial option, the user will be presented with some background information on the topic. He is then given the option of choosing an example or a question. If the question is answered incorrectly, the user is given the option to proceed to a tutorial before moving to a next topic. Ideally, the system should be able to give some feedback or hint during the exercise, depending on the attempt the student makes. In addition, the system should direct the user to the appropriate material depending on his capability, rather than allowing the user to make the decision.

2.6.3 SQL-Tutor

Another intelligent tutoring system in the database domain, SQL-Tutor (Mitrovic, 1998; Mitrovic and Ohlsson, 1999), tutors students in the dominant database language, SQL (Structured Query Language). It is designed as a guided discovery-learning environment and supports problem solving, conceptual and meta-learning. It is based on Constraint-Based Modelling, a student modelling approach proposed by Ohlsson (1992). A constraint-based model represents knowledge about a domain as a set of constraints on correct solutions. The constraints partition the universe of all possible solutions into correct and incorrect ones. This approach concentrates on the violations of the basic principles in the domain of instruction. Constraint violations are identified by inspecting the student's solution and comparing it to the stored ideal solution. If a constraint is violated, this outcome will be recorded in the student model and appropriate action is taken by the system. For example, when a student makes a mistake in an SQL statement, the system will generate the correct solution alongside the student's solution to point out the error. Feedback is generated from SQL-Tutor to explain each of the mistakes in the student's solution.

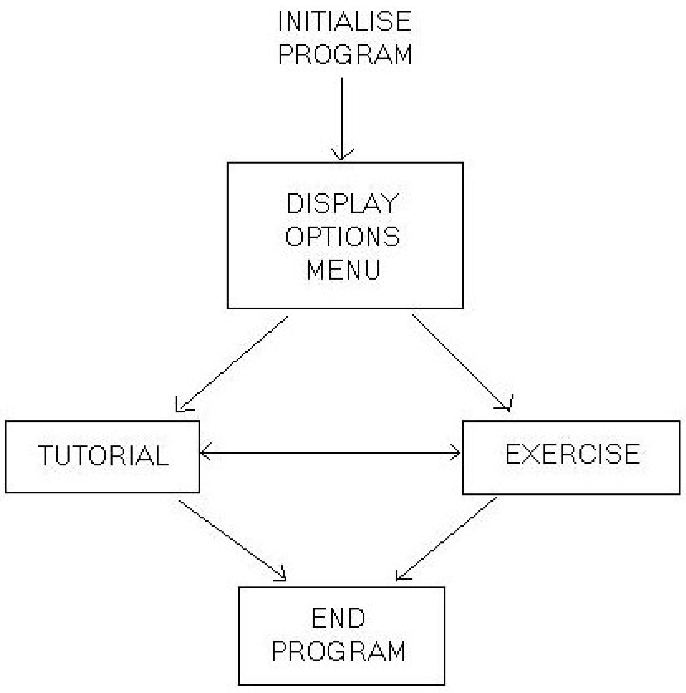


Figure 2.7: Outline of the initial prototype structure (adapted from Canavan, 1996)

The components of the system include the interface, a pedagogical module and a student modeller. Figure 2.8 shows the architecture of SQL-Tutor. The pedagogical module generates feedback messages and selects practice problems. The instruction is individualized in the sense that both types of actions are based on the student model. The student modeller records the history of each constraint. This record contains information about how often the constraint was relevant for the ideal solution to the practice problems the student attempted, how often it was relevant for the student's solution and how often it was satisfied or violated. This record is used by the pedagogical module. There is no domain model for SQL-Tutor. The domain knowledge is instead represented in the form of constraints in the student modeller. Currently, the published system only deals with the SELECT statement in SQL.

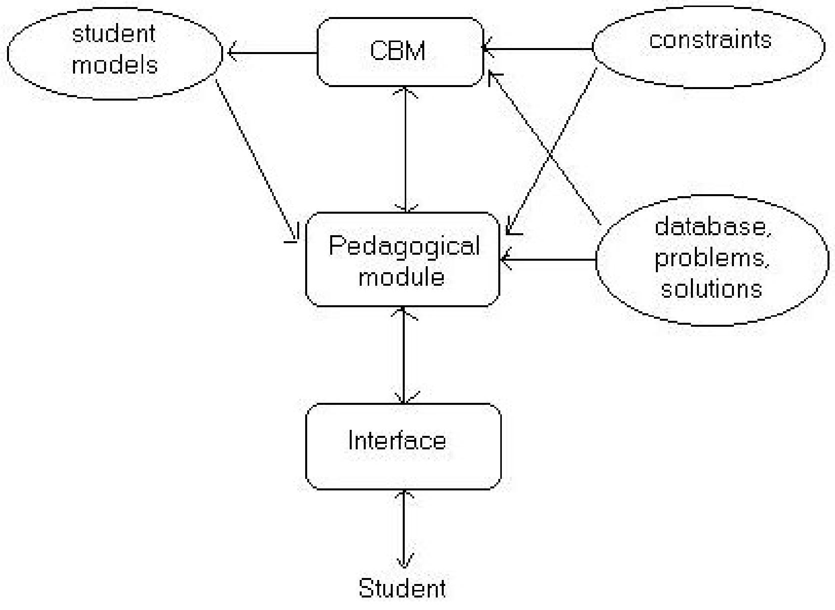


Figure 2.8: The architecture of SQL-Tutor (Mitrovic and Ohlsson, 1999)

2.6.4 COLER

COLER (Constantino-Gonzalez and Suthers, 2000) is a World Wide Web (WWW)-based computer-mediated collaborative learning environment for entity-relationship modelling. An agent is designed for coaching the students in a collaborative learning environment. Students begin by constructing individual entity-relationship diagrams and then work in small groups to agree upon a group solution.

COLER's implementation is based on architecture for intelligent collaborative learning systems from other software, Belvedere. The system was implemented in Java. The implementation concentrated on the coach module, which was built to monitor participation and to identify and evaluate differences between diagrams to encourage students to collaborate. COLER provides four different modes of operation according to the type of user (student/professor) and the selected type of session (individual/group). The interface consists of a *problem description window*, a *private workspace*, a *shared workspace* and a *chat window*. The *problem description window* presents an ER modelling problem. Students construct their individual solutions in the *private workspace*. The *shared workspace* is used to construct a collaborative ER diagram. The *chat window* is used by the students to communicate among themselves. Each student's clients contain a private coach, which monitors the private workspace of its students called CMS (currently monitored student). The coach also monitors the shared workspace and records the students' opinions in the workspace and in chat discussions. However, no natural language interpretation is attempted.

There are no apparent domain or tutoring modules present in the system. A form of student monitoring is conducted in the coach module which consists of four sub-modules. Table 2.5 describes the function of each of sub-module.

Sub-module	*Function*
Differences Recognizer	This module either finds differences specifically related to the currently added object or finds all "extra work" that the student can contribute to the group.
Diagram Analyzer	This module detects ER diagram anomalies. Currently, it is only syntax-based.
Participation Monitor	This monitor attends to the activity in the group diagram. If a session is left idle for a period of time, it will report the event. It also monitors whether each student is participating (too much / too little). It also tracks each student's contribution.
Personal Coach	This module receives feedback from other modules and generates potentially applicable advice and selects the advice to give, if any.

Table 2.5: Function of COLER's coach sub-modules

The current published version of COLER only has access to the student's private workspace and the shared workspace. A student is only able to compare his solution with the group's solution, not with a particular colleague. Constantino-Gonzalez and Suthers (2000) reported that a future version would incorporate this feature to create opportunities for proper collaborative learning.

2.6.5 Kermit

Kermit (Suraweera and Mitrovic, 2002), an ITS for ER modelling was developed as a practice environment for students to model databases for a given problem, with the aim of individualised instructions. Kermit is based on constraint modelling, a student modelling approach that focuses on errors.

The system consists of a user interface, a student modeller, a pedagogical module and a knowledge base. It evaluates the student's answers by comparing them with the correct answers in the knowledge base. The knowledge base is represented in descriptive form and consists of constraints used for testing the student's solution against the system's ideal solution. The constraints deal with both syntactic and semantic errors. The syntactic constraints concentrate mainly on syntactic errors and these are independent of the system's ideal solution. An example of a syntactic error would be a simple constraint like "an entity name should be in upper case". Semantic constraints operate on the relations between the student solution and the system's solution. "The student's solution should consist of all the entities present in the ideal solution" is an example of a semantic constraint.

An animated pedagogical agent, implemented using Microsoft Agent, is used in the learning environment to facilitate learning. The agent offers instructional messages and displays a strong visual presence. However, once the agent presents feedback messages like pointing out an error, the student does not have the opportunity to refer to the feedback once the agent has completed his speech. The student needs to correct all of his errors before referring back to the feedback.

In contrast to a typical ITS, Kermit does not contain a domain module that is capable of solving the problems given to students. The authors of Kermit mentioned that developing a problem solver for database modelling would be extremely difficult, if not entirely impossible. However, the research work presented in this thesis shows that this challenge is possible in a semi-automated environment.

2.7 Summary

This chapter has reviewed a range of systems, from those that apply natural language processing to databases to intelligent tutoring systems (ITSs) for Databases. The different techniques adopted in the NLP systems that attempt to transform natural language to conceptual models were reviewed. A number of the systems reviewed reported on the utilization of heuristics to aid database design. However, only DMG (Tjoa and Berger, 1993) presented a set of detailed heuristics for the transformation from natural language to EER models. A summary table of the review of the systems that apply NLP in Database design was presented and discussed. A few of the systems like E-R Generator (Gomez et al., 1999) and CM-Builder (Harmain and Gaizauskas, 2003) reported the results on their systems's evaluation. Other relevant issues such as heuristics in database design and WordNet were also reviewed. ITSs and their components were also discussed. ITS is one context where ER-Converter can be applied. The next chapter discusses the issues of NLP in database design in more detail.

Chapter 3

Natural language processing in database design

This chapter discusses natural language processing and its application in database design. The use of natural language requirements' specifications in eliciting the knowledge to be modelled and the potential presence of ambiguities in the document are also discussed. This chapter also elaborates on Memory-Based Shallow Parser (MBSP), the parser used to process natural language requirements' specifications. The techniques used in each of the modules of the parser and their applications in natural language processing are discussed.

3.1 Brief overview of database systems analysis

Database problems can be expressed with natural language descriptions of the application which are known as *requirements' specifications*. It is a formal document in natural language that describes the *universe of discourse* (UoD) or the world to be modelled. Users or so-called *domain experts* usually provide the description in the specification. Figure 3.1 shows the stages of database systems analysis involved from producing the requirements' specifications up to the physical database design.

The process of obtaining an initial specification is known as *elicitation*. Systems analysts who are responsible for this analysis in the early stage of the information system development usually carry out this task. During this stage, the primary task of the systems analyst is to map the initial specification on to concepts of a particular conceptual modelling technique.

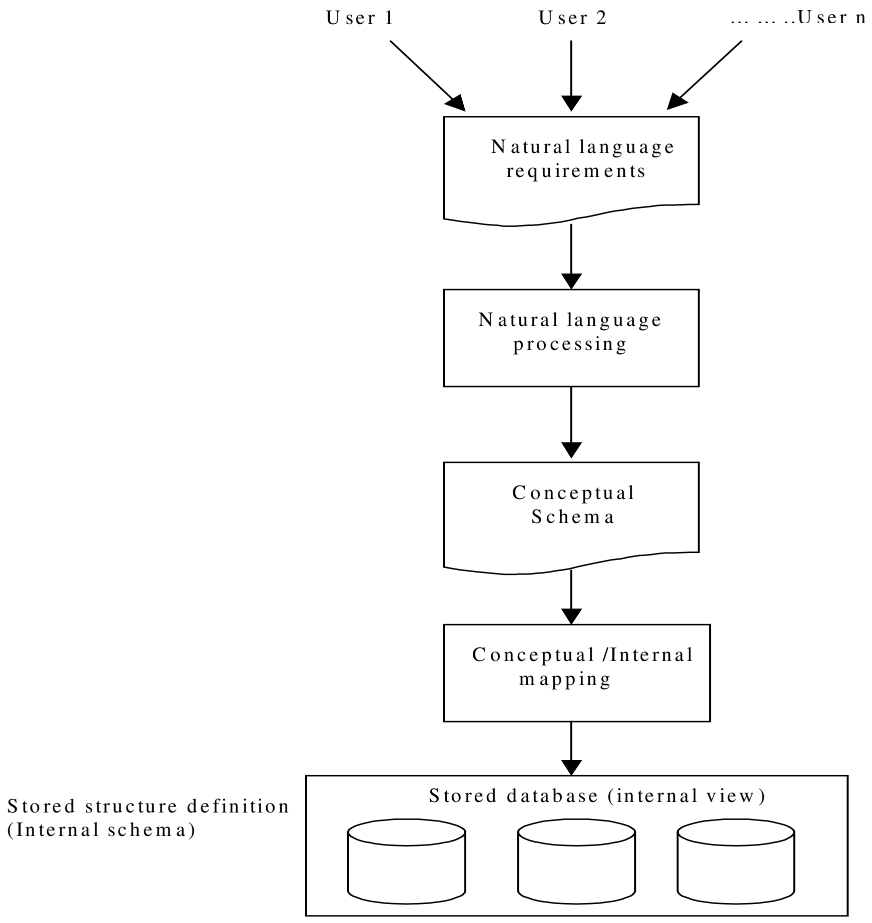

User 1
User 2
… … ..User n
Natural language requirements
Natural language processing
Conceptual Schema
Conceptual /Internal mapping
Stored database (internal view)
Stored structure definition (Internal schema)

Figure 3.1: The stages in database systems analysis

An example of a requirements' specification based on a university environment, taken from Willits (1992) is as follows:

The university wishes to maintain a student database. The student will be identified by a student registration number. Other details include the student's programme, the name of the study advisor, together with the code of each module which a student studies, its title, its lecturer and room number. For each module completed, the student obtains a grade and a number of credits.

The next stage involves natural language processing of the specification before the natural language/conceptual mapping is completed. In the natural language processing (NLP) stage, the syntactic and semantic knowledge captured (extracted/deduced) during the natural language specification analysis is processed. Natural language processing is performed based on the application of heuristics to the specification. The result would be processed further to determine the salient elements of Entity-Relationship models in the conceptual phase.

Once the conceptual model or view is completed, the next stage involves the conceptual/internal mapping. In this stage, the physical structure of the database is determined. This usually involves the data design and determination of the storage structures which result in the creation of the necessary physical tables and their implementation.

3.2 Natural language processing in database design

Natural language processing enables the computer to ‘understand’ human language through analysis, manipulation and generation. This can refer to anything from morphological analysis to higher-level Artificial Intelligence-like tasks such as processing user queries in natural language. Morphological analysis is concerned with the study of the construction of words from more basic components or meaning units called morphemes (Allen, 1995). Manipulation may involve tasks like stemming. Stemming determines the morphological root of a given word form. For example, the stemmer, a program which accomplishes the task of stemming, should identify the word “swimming” based on the word “swim”. In terms of generation, the natural language processing tasks may involve applications such as a text-to-speech system that synthesizes natural sounding speech from ordinary text and a report generator which produces a report from texts.

Generally, natural language processing follows three stages: parsing, semantic interpretation and contextual/world knowledge interpretation (Luger and Stubblefield, 1997; Allen, 1995). The first stage, parsing, analyses the syntactic structure of sentences. Parsing not only verifies that sentences are syntactically well formed, but also determines their linguistic structure. This ensures that the sentence is a legitimate sequence of words in a language. This phase identifies major linguistic relations such as subject-verb, verb-object and modifier. This is often represented as a parse tree. Most parsers employ knowledge of language syntax, morphology and some semantics. An example of a parse tree for the sentence, "The man likes the car" is shown in Figure 3.2.

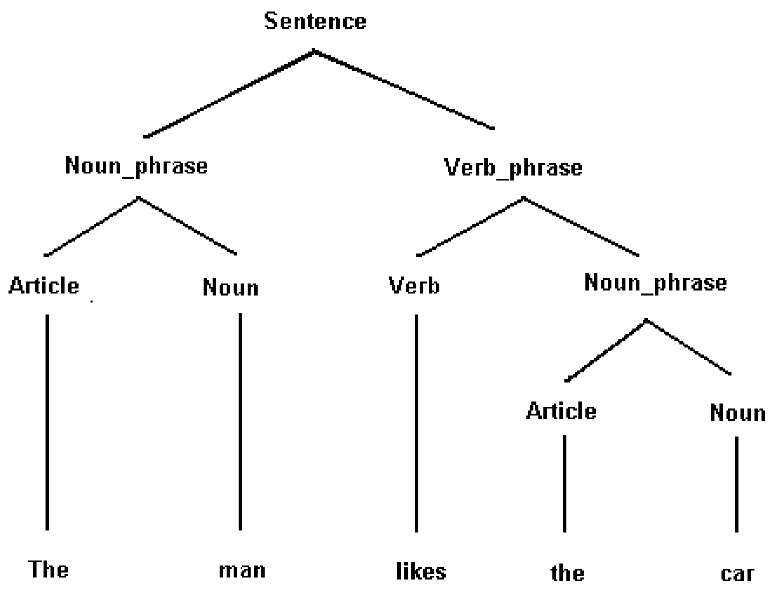


Figure 3.2: Parse tree for the sentence "The man likes the car"

Research at the syntactic level of analysis is primarily concerned with the construction of wide-coverage grammars, efficient parsing strategies and grammar formalisms (Neri and Saitta, 1997). These have led to the development of grammars like structure grammars, context-free grammars and context-sensitive grammars.

The second stage is semantic interpretation, which produces a representation of the meaning of the text. This stage focuses on issues such as what type of knowledge representation formalism is used for determining meaning and how to interpret utterances like:

"I saw her painting. "

which could mean any of the following:

a) “I saw her painting the wall.”

b) “I saw her painting (artwork) at the gallery.”

This type of ambiguity may be resolved by determining the context of the word. Further evidence may come from the immediate textual context, or from a general understanding of the real world. In the context of the given sentence, for example, there may be some evidence from a preceding sentence which denotes that the subject is in a museum. This additional world knowledge can be used to filter the inappropriate parses in order to resolve the ambiguity.

The final stage involves adding structures from a knowledge base to the internal representation of the sentence to produce an expanded representation of the sentence’s meaning. This adds further world knowledge required for complete understanding. Some examples of world knowledge are facts in a given context such as “the man likes the Jaguar”, “Jaguar is not an animal” and “Jaguar is a car”. These facts may be represented using *type hierarchies*, a method of expressing knowledge about the structure of the world. As Jaguar may represent both a living object, i.e. an animal or a non-living object, for instance, a vehicle, these facts need to be ascertained before the meaning is deduced. The resulting structure represents the meaning of the natural language text and is used by the system for enhanced understanding.

In database design, natural language processing usually involves analyzing the requirements’ specifications and looking for linguistic structures that can be mapped to the conceptual schema. The emphasis is on the extraction of relevant information and correct interpretation of the knowledge extracted in order to produce a sound conceptual model. Most of the efforts require studying the relationship between sentence structure and the model to be mapped. However, due to limitations in natural language specifications such as ambiguities in natural language, this presents numerous challenges in processing the documents. These issues will be discussed further in the following section.

3.2.1 Ambiguities in natural language specifications

The main disadvantage of using natural language requirements' specifications is the potential presence of ambiguities. This leads to two issues. The first is the detection of ambiguities and the second is the resolution of the ambiguities. In natural language processing, it is essential to remove any ambiguities before proceeding to further analysis to minimize any errors during the modelling. According to Somers (2000), ambiguities can be categorized as:

- Lexical ambiguity
- Structural ambiguity

In database design, these ambiguities are sometimes present in the requirements' specifications.

Lexical Ambiguity

Ambiguity may arise at a lexical level where a word may have more than one interpretation (sometimes referred to as word *senses*) (Allen, 1995). This type of ambiguity may happen as a word can represent different *parts of speech* and have different meanings. *Parts of speech* refers to the classification of words depending on how they are used. For example the word ‘store’ may mean a business establishment or it may indicate something that is kept for future use. It may represent a noun, an adjective or a transitive verb depending on the context.

This type of ambiguity may be resolved when the syntactic category of the word is identified. This is possible through natural language processing techniques such as *parsing* as this can often identify the part of speech of the ambiguous word. This understanding may come from the more or less immediate textual context, or from a general understanding of the real world (Somers, 2000). For example, consider sentences (1) and (2):

“The store is located in Ballymena.” (1)

"The system must store information about the patient." (2)

In sentence (1), the word 'store' is a noun that acts as a subject. Due to the juxtaposition of the word, it is easier to deduce that the word 'store' is a noun. In (2), the word 'store' is a verb which requires a subject prior to its use. Due to the different parts of speech in different contexts, the correct interpretation of the sentence is easier to obtain automatically when the context is determined.

Structural Ambiguity

Ambiguities may also arise from how a sentence is structured. Structural ambiguity results when the combined lexical ambiguity of the words making up a sentence means that it has several interpretations. Two common forms of structural ambiguity are *attachment ambiguit*y and *coordination ambiguity*. *Attachment ambiguity* as illustrated in sentence (3), can pose a problem as some knowledge is needed as there may be several interpretations leading to different meanings.

"The boy saw his dad with a pair of glasses." (3)

Structural ambiguity leads to the following two interpretations:

a) "The boy saw (his dad with a pair of glasses)."
The boy saw his dad and his dad was wearing glasses at that point of time.

b) "The boy saw (his dad) with a pair of glasses."
The boy saw his dad whilst wearing a pair of glasses.

In general, a natural language understanding system may not be able to decide which interpretation is intended. However, the ambiguity can be highlighted and user intervention requested to choose the correct interpretation.

Coordination ambiguity occurs when there are different sets of phrases that can be conjoined by a conjunction like *and* (Jurafsky and Martin, 2000). For example, the phrase "young kids and adults" can be bracketed as [young [kids and adults]] meaning [young kids and young adults] or [[young kids] and [adults]] which refers to young, juvenile kids but not necessarily young adults. Due to the fact that there are many unreasonable parses for a sentence affected by this type of ambiguity, *disambiguation* or choosing the correct parse may be necessary (Jurafsky and Martin, 2000). Disambiguation algorithms generally require both statistical and semantic knowledge (Jurafsky and Martin, 2000).

Structural ambiguities can be further classified according to the 'range' or scope of the ambiguity they represent. These ambiguities can be categorised as:

- local ambiguities
- global ambiguities
- anaphora resolution

Local ambiguities can occur when some part of the sentence is ambiguous, having more than one parse, even if the whole sentence is not ambiguous (Jurafsky and Martin, 2000). Sentence (4) is an example of such ambiguity:

"The students taught by the direct method failed." (4)

In the process of parsing the sentence, the word "taught" would be processed as a transitive verb, with the reading that the student taught someone or something. But further analysis may reveal that the students are the ones being taught i.e. "taught" is not a main verb but instead a participle.

Global ambiguities, on the other hand, are caused by combinations of ambiguities: different analysis involves different category choices (Somers, 2000). An example is shown in sentence (5):

"John saw her playing cards." (5)

This sentence either means John saw someone playing with cards or John actually spotted the cards. The word “playing” can either be an adjective or a verb in the sentence resulting in different interpretations.

Anaphora resolution is the identification of antecedents of pronouns. *Anaphora* refers to an entity that has been mentioned previously in the discourse or text. Sentence (6) shows an example of anaphora resolution:

“Each module is attached to one course where it has a name and a code.” (6)

In sentence (5), the word ‘it’ may be referring to either the module or the course. This is ambiguous and may need some background knowledge to determine the correct interpretation. In a natural language specification which will be analysed automatically, it is important to resolve this sort of ambiguity and replace the anaphoric terms with their actual reference or antecedent.

Another particular case of substitution apart from anaphoric resolution that may be ambiguous is *ellipsis*. It occurs when certain words or phrases, which have been mentioned earlier in the text, are omitted. For example, the noun phrase ‘a team of employees’ may be replaced with just ‘employees’ in subsequent sentences. This occurs quite frequently in natural language requirements’ specifications and should be resolved either prior to the automated processing or in the program itself.

3.2.2 Solutions to ambiguity

Presented with many types of ambiguities, natural language requirements’ specifications need to be pre-processed, either manually or automatically, to minimize any errors in the interpretation of the text. In terms of manual processing, among the solutions adopted are:

- Controlled language
- Pre-editing

Controlled language or sometimes referred as *restricted input* is a controlled language developed to limit the vocabulary, syntax and semantics of the input language (Harmain and Gaizauskas, 2003). It is sometimes used to write software specifications and used by tools to analyse these specifications to produce useful results. An example of such a system is Attempto Controlled English (ACE) (Fuchs and Schwitter, 1996; Fuchs and Schwertel, 2003). *Pre-editing* usually involves some checking done of the specifications to adhere to certain restrictions of the system. Meziane (1994) reported on pre-editing the specification text to resolve the problems of conjunctions and pronoun references, which are not handled by the implemented system, before proceeding to the automatic analysis of the text.

A potential solution to these problems is the incorporation of more sophisticated techniques of analyzing the text to be processed. This usually involves the use of a parser, tagger and a semantic interpreter. In this research, a parser known as Memory-based Shallow Parser (MBSP) has been selected for the purpose mentioned above, that is, to analyse the natural language requirements with the utilization of the natural language techniques that the parser has to offer. MBSP was selected due to the accuracy of its tagger, as will be discussed in Section 3.3.2, and its availability on the Internet. Other parsers like Snow-Based Shallow Parser (Munoz et al., 1999) and Connexor (Tapanainen, 1996) were also considered. MBSP and its approach used in the processing of the natural language input are discussed in the next section.

3.3 Memory-based Shallow Parser (MBSP)

Shallow parsing is an essential component in text analysis systems in text mining applications such as information extraction and question answering (Zavrel and Daelemans, 2003). Shallow parsing performs only partial analysis of the syntactic structure of sentences as opposed to full-sentence parsing. The parsing includes detecting the main constituents of sentences (for example noun phrases (NPs) and verb phrases (VPs)) and their head nouns, and determining syntactic relationships like subject, object and adjunct relations between verbs. This early step uncovers basic information like *who*, *what* and *where* in sentences.

In the MBSP approach, the syntactic analysis process is split up into a number of classification tasks. These classification tasks can be segmentation tasks (for example in deciding whether a focus word or tag is the start or end of NP) or disambiguation tasks (for example whether a chunk is a subject NP or object NP). Output of some memory-based modules is used as input to other memory-based modules.

The MBSP for English consists of the following modules:

- A tokenizer
- A tagger
- A chunker
- A subject/object detector

Each of these modules yields results in separate output files. The first three modules will be explained in more detail in the following sections.

3.3.1 Tokenizer

A tokenizer basically breaks up the sequence of characters in a text by locating the word boundaries, the point where one ends and another begins (Daelemans et al., 2000). The goal is to break up the text into smaller units called tokens. Each token corresponds to a word form, a number, a punctuation mark or other kind of unit to be passed on to subsequent processing. In MBSP, the tokenizer splits punctuation marks like period, comma, question mark and colon from words. For example the genitive clause 'the client's name' will result in 4 tokens. These tokens are 'the', 'client', ''s' and 'name'. These elements are later tagged separately according to their part-of-speech (POS).

3.3.2 Memory-Based tagger

A POS tagger assigns the words in a text to their morphosyntactic categories based on the characteristics of the words and the context in which they occur. POS tagging is the first level of abstraction in text analysis and plays an important role in many language technology applications such as information retrieval, speech recognition and text mining

(Jurafsky and Martin, 2000). Memory-based tagging is based on the idea that words occurring in similar contexts will have the same POS tag. The tagging technique used in Memory-Based tagger (MBT) is based on Memory-Based Learning (MBL). MBL is a supervised classification-based learning method. It consists of storing the instances seen during learning in memory along with the corresponding categories. A new instance can be classified in a category by computing the distance between the new instance and the stored instances in memory. The distance is computed using *similarity metric*, a feature of IB1-IG (Daelemans and Van de Bosch, 1992), a memory-based learning algorithm that builds a database of instances during learning.

MBT is a tagger generator (Daelemans et al., 1996). This means that it can be applied to any annotated training corpus, and yields a working tagger that can accurately annotate previously unseen text in the same manner as in the training corpus. For this purpose, a lexicon and a disambiguator for known and unknown words are derived fully automatically from the tagged example corpus.

The construction of a POS tagger is described in Daelemans et al. (1998). Given an annotated corpus, three data structures are automatically extracted: a *lexicon*, a case base for *known words* (words occurring in the lexicon), and a case base for *unknown words*. For known words, cases comprise of information about a focus word to be tagged, and an associated category (tag) valid for the focus word in that context. For unknown words, this tag can only be guessed on the basis of the *form* or *context* of the word. During tagging, each word in the text that is to be tagged is looked up in the lexicon. If the word is found, its lexical representation is retrieved and the word's context is determined. The resulting pattern is disambiguated using extrapolation from the most similar words in the known words base case. If a word is not found in the lexicon, its lexical representation is computed on the basis of its form, its context determined and the resulting pattern is disambiguated using extrapolation from the most similar cases from the unknown words case base. In each case, the output is based on the best guess of the category for the word in its current context.

Table 3.1 compares the accuracy of MBT with a number of alternative tagging methods. These alternatives are Rule-Based (Brill, 1994), Trigram (Steeskamp, 1995) and

Maximum-Entropy (Ratnaparkhi, 1996). Based on the same corpus, each of these taggers uses different features of the text to be tagged. Each of them has a completely different representation of the language model.

Tagger	*Accuracy (%)*
Trigram	96.1
Rule Based	96.5
MBT	97.0
Maximum Entropy	97.4

Table 3.1: Accuracy of different taggers (Zavrel and Daelemans, 1999)

The results, shown in Table 3.1, show that MBT scores marginally better generalization accuracy than two widely used methods, i.e. trigram tagging and rule-based tagging (Zavrel and Daelemans, 1999). This may be due to the fact that in MBT, all information is stored in memory, compared to probabilistic and other machine learning approaches adopted by the other two taggers. The Maximum Entropy tagger performs better, which is due to the fact that the tagger's weighting is better able to deal with the dependencies in the rich feature-set (Zavrel and Daelemans, 1999). Due to its comparable accuracy in tagging and ease of access, MBT was selected for this research work for the tagging of natural language requirements' specifications. An example output from sentence (7) using the MBT tagger is shown as follows:

Example sentence:
"A payment may settle the invoice in full or by instalments i.e. an invoice may be associated with many payments." (7)

Output:
```
A/DT payment/NN may/MD settle/VB the/DT invoice/NN in/IN full/JJ or/CC
by/IN instalments/NNS i.e./FW an/FW invoice/NNP may/MD be/VB
associated/VBN with/IN many/JJ payments/NNS./.
```

The output shows the generic format of the tagger. The full list of abbreviations of the Penn Treebank II part of speech tags can be found in Appendix B.

3.3.3 Chunker

Phrase chunking involves the process of detecting the boundaries between phrases (for example noun phrases) in sentences (Daelemans et al., 1998). Chunking can be regarded as light parsing. In MBSP, *NP chunking* and *bracket prediction* is applied for the chunking purposes.

In *NP chunking*, sentences are divided into chunks and labels are assigned to these chunks. The process of chunking and labelling is carried out from left-to-right in a sentence and a tag is assigned to each word. The types of chunks determined in this chunker include NP (noun phrase), VP (verb phrase), ADJP (adjective phrase) and ADVP (adverb phrase). For a noun chunk, the chunking starts from the beginning of a noun phrase up to the head noun, thus excluding any complements or adjuncts following the head. Using the previous example, the chunking process would provide the following output:

```
[NP A/DT payment/NN NP] [VP may/MD settle/VB VP] [NP the/DT invoice/NN
NP] {PNP [PP in/IN PP] [NP full/JJ NP] PNP} or/CC {PNP [PP by/IN PP] [NP
instalments/NNS NP] PNP} i.e./FW [NP an/FW invoice/NN NP] [VP may/MD
be/VB associated/VBN VP] {PNP [PP with/IN PP] [NP many/JJ payments/NNS
NP] PNP}./.
```

[NP…NP] brackets denote the noun phrases. An example of an NP chunk from the sample for the phrase "A payment" is [NP A/DT payment/NN NP]. The structure {PNP…PNP} represents a preposition and one or more NPs that together form a prepositional chunk. Prepositional chunks that do not contain any NPs are not marked as PNPs but simply as [PP...PP] to represent a preposition. Adjectival chunks [ADJP…ADJP] start from the beginning of an adjectival phrase up to the head adjective, excluding any complements or adjuncts following the head. Similar rules apply to the adverbial chunk, [ADVP…ADVP]. The verbal chunk [VP…VP] comprises a main verb, its entire modal and auxiliary verbs, any intervening verbs and any directly following verbal complements of the main verb.

For adverbial functions, the classes for the adverbial function labels used are LOC (locative), TMP (temporal), DIR (directional), PRP (purpose and reason), MNR (manner) and "-" for none of the former. This task is performed by Subject Object Detector, a tool which involves grammatical relation assignment. This tool attempts to resolve the attachment between labelled phrases.

In *bracket prediction*, the task is to predict the sequence of closing and opening brackets preceding a word. Closing all pending open brackets at the end of the sentence suffices to construct an unlabeled parse tree of the sentence out of sentence predictions (Daelemans et al., 1998). The input of the bracket predictor is a tagged sentence. For example, given a tagged sentence "The/DT cat/NN liked/VBD it/PRP ./.", the output is an unlabeled, bracketed sentence as follows:

[[[The] [cat]] [[liked][[it]]]].

3.4 Summary

This chapter has discussed the area of natural language processing in database design. The main emphasis was on the use of natural language specifications in extracting knowledge of conceptual modelling and potential ambiguities during the processing. This chapter has also discussed the parser to be used in the implementation stage, Memory-based Shallow Parser (MBSP). Each of the modules of MBSP and the techniques used were described. Memory-based Tagger (MBT), the tagger used in MBSP has an accuracy of 97.0%. MBSP has been selected due to its comparable accuracy and ease of access. The next chapter discusses the heuristics-based approach to support the transformation of natural language requirements' specifications to Entity-Relationship (ER) models.

Chapter 4

Heuristics in Database Design

This chapter introduces heuristics in general and their application to database design. Heuristics for database design from existing literature are presented and discussed. A set of proposed new heuristics that may improve the generation of ER models from natural language specifications is presented in Section 4.2. Heuristics' weights and their application are discussed. Results on the training dataset, the heuristics' selections and the justification for these selections are presented and discussed.

4.1 Existing heuristics

Previously published heuristics to aid the construction of ER models from natural language requirements' specifications will now be presented. The scope of the language of the specifications and heuristics used in this research is English. The list in Figure 4.1 shows the abbreviations used in identifying the categories of each of the heuristics.

Abbreviation	*Category of heuristic*
HE	Entity type
HA	Attribute type
HR	Relationship type
HC	Cardinality type

Figure 4.1: Abbreviations used for categories of heuristics

4.1.1 Heuristics to determine entity types

The following presents a set of previously published heuristics to determine entity types. They are largely based on those given in Chen (1983) and DMG (Tjoa and Berger, 1993).

Heuristic HE1: All nouns are converted to entity types (Tjoa and Berger, 1993)

This heuristic assumes that all nouns can be directly mapped to entity types. This includes all type of nouns such as collective nouns, common nouns, count nouns, mass nouns and proper nouns. Examples of these nouns are 'people', 'Washington' and 'peace'.

Heuristic HE2: A common noun may indicate an entity type (Chen, 1983; Tjoa and Berger, 1993)

A common noun is a type of noun that names any person, place, thing, or idea. An example to illustrate the use of common nouns is as follows:

"The school has a principal and many teachers".

Note that "school", "principal" and "teachers" are common nouns and therefore this heuristic implies that they correspond to entity types.

Heuristic HE3: A proper noun may indicate an entity (Chen, 1983; Tjoa and Berger, 1993)

A proper noun is a noun that names a *specific* person, place, thing or idea. An example is shown below:

"Mr. Arnold Johnson works in the Human Resource department".

In this example, "Mr. Arnold Johnson" and "Human Resource department" refer to a specific person and place and therefore HE3 indicates that they indicate entities.

Heuristic HE4: A gerund may indicate an entity type which is converted from a relationship type (Chen, 1983)

A gerund is a type of noun converted from a verb, also known as a verbal noun, usually ending in *–ing* (for English). An example is as follows:

“Customers may hire many cars and the hiring is processed by a clerk”.

From the example given, the verb “hire” is converted to a gerund “hiring” as the subject of the second clause. The verb “hire” can be said to correspond to a relationship type and this has been converted into the entity type “hiring”. When a relationship type is converted into an entity type, it usually inherits some form of attributes. In this example, “hiring” may have attributes like “hiring number”, “clerk id” and “hiring date”.

Heuristic HE5: A clause may indicate a high-level entity type which hides a detailed Entity Relationship Diagram (ERD) (Chen, 1983)

The clause is the main building block in English which includes a subject and predicate. A clause or sub clause can be built upon another clause. The following example illustrates this heuristic:

“The lecturer decides which project to assign to each student”.

In the example, the clause “which project to assign to each student” is sub-clause of the verb “decides”. In this particular clause, “project” and “student” are entity types while “assign to” denotes a relationship type. The entire clause could be viewed as an equivalent to a high-level entity called *assignment*.

4.1.2 Heuristics to determine attribute types

This section presents a set of previously published heuristics to determine attribute types.

Heuristic HA1: A noun which takes the general form of TERM_SUFFIX such as noun_id, noun_no, noun_type or noun_number may indicate an attribute type (Storey, 1988)

A noun such as “person_id”, “group_no”, “room_type” and “vehicle_number” may indicate an attribute type. The TERM_SUFFIX representation is often used in database problems’ specifications. An example is shown as follows:

“Each textbook has a book_id and a title”.

The noun “book_id” in the above example may indicate that it is an attribute type.

Heuristic HA2: A noun phrase which follows the phrase “identified by” may indicate the presence of attribute types (Gomez et al., 1999)

Examples are:

a) “A person, *identified by* person_id and a surname, can own any number of vehicles”.
b) “Suppliers are *identified by* supplier_id”.

Heuristic HA3: A noun phrase succeeding the “has/have” verb phrase may indicate the presence of attribute types (Storey, 1988)

“Have/has” verb phrases may indicate a relationship between an entity and its attributes. Four types of interpretations of “has/have” are possible (Storey, 1988):

1. A possesses B

An example that shows possession is:

“The car hire company has many branches”.

In this type of “has/have” phrase, the noun that occurs after the phrase does not usually denote an attribute. The “possession” would normally show a relationship between two entities. From the above example, "car hire company" and "branches" indicate that both are entities. Therefore, the heuristic HA3 may not apply in these cases.

2. B component-of A

An example of this component-of interpretation is:

"The machine has a thermostat".

In the example, "thermostat" can be interpreted as a component of "machine". Therefore, it represents an attribute of "machine".

3. B instance-of A/ example-of A

An example of instance-of interpretation of have/has could be:

"The books have many volumes ".

In this example, "volumes" is an instance of "books". Both can be regarded as entity types in the example.

4. B associated-with A in some other way

This type of interpretation commonly shows an association between two entity types or an entity with its attributes. The following shows examples of this association:

"The library has many book suppliers".

In this relationship, "book suppliers" may be regarded as an entity associated to "library".

"The book has a publisher".

In this case, “publisher” may become an attribute of “book” or another entity type linked to it. This depends on whether the existence of “publisher” is highly significant in the business environment where other attributes of “publisher” like publisher’s address, id and telephone number are kept. It may also exist simply as an attribute to “book” when only the publisher’s name, for example, is recorded in the particular environment. The heuristic may apply in the latter case.

Heuristic HA4: An intransitive verb may indicate an attribute type (Chen, 1983)

If a main verb does not require another element to complete it, the verb is intransitive. Chen (1983) has proposed that an intransitive verb may indicate an attribute type. The following shows such an example:

“The train arrives every morning at approximately 8:15 am”.

In this example, the verb “arrives” does not require a direct object to complete the sentence and thus is referred to as an *intransitive verb*. A hidden attribute type, “arrival time” may be deduced from the verb “arrives” to denote the arrival time of the train. This is deduced from the sentence where the arrival time is mentioned.

Heuristic HA5: An adjective can be an attribute type (Chen, 1983; Tjoa and Berger, 1993)

An adjective is a word or phrase naming an attribute, added to or grammatically related to a noun to modify it or to describe it. An adjective is said to correspond to an attribute of an entity. For example:

“The large photo album has extra charges on delivery”.

The adjective “large” may indicate an attribute of type “size” for the entity “photo album”.

Heuristic HA6: Genitive case in the noun phrase may indicate an attributive function (Tjoa and Berger, 1993)

Case genitive indicates possession, or a relationship to another noun similar to that expressed by “of” in English. As stated under Heuristic HE6, a case genitive may suggest an attributive function. This can be shown in the following:

“The employee’s name is stored”.

The noun “name” may suggest an attribute type to the entity type “employee”.

4.1.3 Heuristics to determine relationship types

This section presents previously published heuristics to determine relationship types. These are as follows:

Heuristic HR1: An adverb can indicate an attribute for relationship (Chen, 1983)

An adverb is a word or phrase that modifies or qualifies another word, expressing a relation of place, time, circumstance, manner, cause or degree. The following shows an example of an adverbial phrase:

“The employee visits the site a maximum of twice a week”.

In this example, both “employee” and “site” are nouns and can be considered as entity types. The verb “visits” corresponds to the relationship type. The adverbial phrase “twice a week” describes the frequency of “visits”. Therefore, an attribute called “frequency of visit” can be linked to the relationship “visit”.

Heuristic HR2: A transitive verb can be a candidate for relationship type (Chen, 1983)

A transitive verb is where a main verb requires a direct object to complete the sentence. It may be a candidate for a relationship type. An example is shown below:

“A borrower may borrow many books”.

Note from the above example that “borrower” and “books” are both entity types. “Borrower” is the subject followed by “books” as the direct object. The verb “borrow” is a transitive verb and therefore corresponds to a relationship type.

4.1.4 Heuristics to determine cardinality types

Tjoa and Berger (1993) presented only one of the many heuristics they claimed to determine cardinality types:

Heuristic HC1: A noun or a prepositional phrase whose noun is singular gets a minimal and maximum cardinality of 1 (Tjoa and Berger, 1993)

Consider this example:
“The single room is meant for only one guest”.

In this example, “room” and “guest” are both singular nouns and these may suggest that the cardinality is of type one-to-one.

4.2 Proposed new heuristics

Due to the desire to improve the accuracy of results, an additional set of new heuristics is proposed here. These may be viewed as additional syntactic heuristics which may help to improve the results in the determination of ER elements from English natural language requirements’ specifications. Only heuristics based on syntax are considered and proposed. These heuristics are presented in the following subsections.

4.2.1 Heuristics to determine entity types

The following are a set of proposed heuristics to determine entity types:

Heuristic HE6: If a noun occurs before a genitive, it may indicate an entity type

A genitive case indicates that the noun is dependent on the noun that follows it. The genitive case always describes a property situation (possession) or has an attributive function. For example:

"The student's address is recorded".

In this example "student" may indicate an entity type.

Heuristic HE7: If compound nouns are present, check the last noun. If it is not one of the words in set S where S={number, no, code, date, type, volume, birth, id, address, name}, most likely it is an entity type. Else it may indicate an attribute type.

This heuristic places emphasis on the last word of a compound noun to determine whether a word is either an entity or attribute. An example is shown below:

"The registration office also keeps a record of the registration number and registration date".

"Registration office", "registration number" and "registration date" are all compound nouns. In this example, "registration office" suggests that the noun is an entity type as its last noun does not belongs to set S. Both "registration number" and "registration date" suggest that they are attribute types.

Heuristic HE8: If a noun occurs before the verb 'has'/ 'have', it may indicate an entity type

For relationships of the form A have/has B where A and B are both nouns, the occurrence of A may indicate that it is an entity. This is true in cases where the relationship between A and B implies B instance-of A or B component-of A. This is illustrated in the following example:

"Each piece of equipment has an equipment number and a description".

In this example, "equipment" may suggest that it is an entity type due to its occurrence prior to the "has" verb phrase.

Heuristic HE9: If a noun occurs before the verb 'identified by', it may indicate an entity type

The verb phrase "identified by" is commonly used in database problem specifications to indicate the instances of an entity type. Therefore, the noun that occurs before the verb phrase may imply that it is an entity. For example:

"Suppliers are identified by supplier id".

"Supplier", in this case, may indicate an entity type.

4.2.2 Heuristic to determine non-entity types

A proposed heuristic to determine non-entity types is presented below:

Heuristic HEX: If a noun belongs to any of the set X where X= {record, database, company, system, information, organization, detail, interest, number, track} exclude it as a potential entity type candidate

A noun such as *record, database, company, system, information* and *organization* may not be a suitable candidate for an entity type. For example, *company* may indicate the business environment and should not be included as part of the entity types. For example:

a) "An insurance company wishes to create a database to keep track of its operations".
b) "An organization purchases items from a number of suppliers".
c) "A database on diets is to be maintained that can store any number of diets".
d) "A record is kept of customer payments and each customer is allocated a status depending on their history".
e) "A hospital wishes to computerize its information about staff, wards, patients and operations".

4.2.3 Heuristics to determine attribute types

Proposed heuristics to determine attribute types are:

Heuristic HA7: A noun phrase which precedes the verb phrase "is/are stored", "is/are recorded" or "is/are kept" or the phrase "is/are of interest" may indicate the presence of attribute types

All the verb phrases indicated above usually indicate storage of data. This commonly refers to storage of attributes. The noun phrase "is/are kept" is mainly used in American English. Examples are as follows:

a) "The time and source of each document are stored".

The attribute types that can be derived from this sentence are "time" and "source".

b) "Each patient has a unique number and information such as his or her date of birth, address and occupation is stored".

In this sentence, the nouns that may indicate attribute types are "number", "date of birth", "address" and "occupation".

c) "The hour and length_of_use are recorded".

In this example, the nouns "hour" and "length_of_use" may indicate attribute types.

d) "The booking number and room number are kept".

"Booking number" and "room number" may indicate attribute types in the given example.

Heuristic HA8: If a noun is followed directly by another noun and the latter belongs to set S where S={number, no, code, date, type, volume, birth, id, address, name}, this may indicate that both words are an attribute. Otherwise, it is most likely an entity (HE7).

In this case, the last word of a compound noun is checked to determine whether the noun belongs to set S as described above. If it does, there may be a possibility that the compound noun is an attribute type. An example is as follows:

"Every project has a project number and completion date".

"Project number" and "completion date" both may indicate attribute types of "project".

4.2.4 Heuristics to determine relationship types

A set of proposed heuristics to determine relationship types are presented here.

Heuristic HR3: The preposition "for" can indicate a relationship type

A preposition relates nouns, pronouns and phrases to other words in a sentence. The preposition 'for' may indicate a relationship type where the subject and the object are entity types. This is shown in the following example:

"Rooms are available for hire".

In this case, both "rooms" and "hire" are entity types related through the preposition "for".

Heuristic HR4: A verb followed by a preposition such as "on", "in", "by" and "to" may indicate a relationship type

A verb may typically indicate a relationship type. However, this may not be true in all cases. Consider this example:

"All employees work six days a week".

In this example, the verb "work" may not represent a direct relationship type. On the other hand, a verb followed by a preposition may indicate a relationship type. For example:

"People work on projects".

In this example, "work on" represents a relationship type between the entity types "people" and "projects". Other examples may include "*assigned to*" and "*managed by*".

Heuristic HR5: A verb that appears before an adjective "many" or "any" may indicate a relationship

The adjective "many" or "any" may show the cardinality of an entity in connection with another entity. A verb that appears prior to these adjectives may indicate a relationship type. This can be shown in the following examples:

"A surgeon may perform many operations".

In these examples, the verb "perform" that appears prior to the adjective "many" may indicate a relationship type.

4.2.5 Heuristics to determine cardinality types

The cardinality or degree of a relationship is concerned with the maximum participation of entity types. Adjectives, cardinal numbers and nouns may determine the cardinality of relationship types. Proposed heuristics to determine cardinalities are presented below:

Heuristic HC2: The adjective "many" or "any" may suggest a maximum cardinality

A maximum cardinality usually signifies more than one occurrence of an entity type. Thus, the adjectives "many" or "any" are often used in requirements' specifications to describe this. The following examples illustrate the use of "many" to suggest maximum cardinality:

a) "A surgeon can perform *many* operations".
b) "A department may manage *many* projects".
c) "Each diet may be made of *any* number of servings".
d) "Each serving is made up of *any* number of food elements."

Heuristic HC3: A comparative adjective "more" followed by the preposition "than" and a cardinal number may indicate the degree of the cardinality between two entities

The phrase "more than" is a synonym for the adjective "many" to indicate a maximum cardinality. An example is as follows:

"Each patient could have *more than one* operation".

However, the phrase may also indicate an occurrence of multivalued attributes within an entity such as the example below:

"There may be *more than one* such address for each supplier".

In these cases, heuristic HC3 may not apply as cardinalities are only drawn between two entity types and not between an entity type and its attributes.

Heuristic HC4: Cardinal number "one" or the adjective "single" may indicate cardinality of one

The cardinality of one usually shows that for any single occurrence of an entity type, there can possibly be at most one occurrence. Therefore, the cardinal number "one" may indicate this type of cardinality. An example is as follows:

"Each project is managed by a single department."

Heuristic HC5: The noun "none" or the cardinal number "zero" may indicate the lower bound of a cardinality

The lower bound of a cardinality may indicate that the relationship is optional. The noun "none" may be used in this type of cardinality. An example is as below:

"A person may work on *none* of the projects in his department".

Heuristic HC6: The phrase "one or more" or the adjective "multiple" may indicate a maximum cardinality.

The phrase "one or more" is another synonym to indicate maximum cardinality of a relationship type. An example is as shown:

"Each area has *one or more* representatives".

4.3 Development of the heuristics

In order to produce a set of functional heuristics, a number of stages, as shown in Figure 4.2, were involved. Early in the research work, a collection of existing heuristics from the literature were gathered and analysed. The focus of the study at this stage was on previously published heuristics, based on natural language syntax, to determine ER models from natural language specifications.

The second stage involved the formation of new heuristics, which it was anticipated would improve the accuracy of the results of the desired ER models. This involved analysing natural language requirements' specifications of database problems to determine the potential new heuristics.

Once both the existing and new heuristics were compiled, a training dataset consisting of 10 database problems in English were used to test these heuristics. The training dataset can be found in Appendix C. At this stage, a manual test was conducted to investigate the practicality of each of the heuristics.

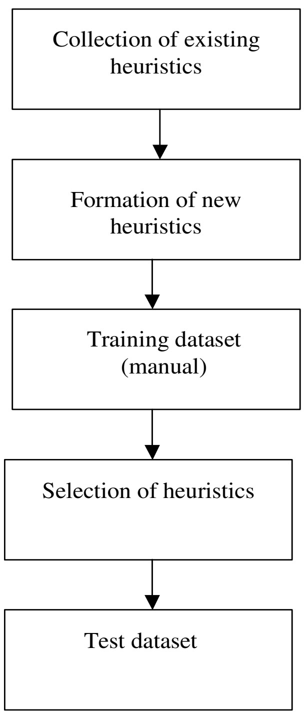

Collection of existing heuristics
Formation of new heuristics
Training dataset (manual)
Selection of heuristics
Test dataset

Figure 4.2 Processes involved in the development of heuristics

The next stage involved the selection of the heuristics to be implemented. The selection of each of the heuristics was based on a number of criteria as presented in section 4.6. The selected heuristics were then implemented in the transformation tool, ER-Converter. The test dataset, which consists of 30 database problems, was used in the final evaluation of ER-Converter. The test dataset can be found in Appendix D.

4.4 Heuristic weights

In this research work, similar concepts to certainty factors as used in MYCIN, as given in Section 2.3, are adopted to represent uncertainty when dealing with the heuristics. The difference in comparison to MYCIN though, is that the negative weights do not mean that the confidence level of the evidence is low. Each of the heuristics is given a positive or negative weight depending on the level of confidence. Positive weights are assigned to entity type, relationship type and cardinality type heuristics. Negative weights are assigned to attribute type heuristics. This is due to the fact that a noun may indicate an entity type or an attribute type. As relationships and cardinalities deal with different parts-of-speech (POS), problems of multiple evidence occurring between the two categories is impossible. Relationships are commonly denoted by verbs whereas cardinalities can be identified through adjectives or cardinal numbers. Therefore, their weights are assigned with positive values only.

The negative weights are assigned such that if more than one heuristic from either the entity or attribute type categories are applied to a word, this would reduce the sum of the total weights. The sum of weights can be outside of –1 and +1 range. Values approaching zero are treated as "low confidence". Two or more "weak" pieces of evidence are combined to give the weight an acceptable level of confidence. When these "low confidence" values fall within the threshold of –0.2 and 0.4, the user will be prompted to intervene in deciding the word's ER category. These threshold values resulted from the observations of the training dataset, where the occurrences of some multiple contradicting heuristics to a word may identify the wrong ER elements if their total weight lies within certain thresholds values. Therefore, the aim of the threshold values is to minimize these errors and instead, request human intervention in deciding to determine a word's ER element category. In doing so, this task also aims to achieve optimal threshold values where the incorrect results and user interventions can be minimised. Early in the investigation, initial threshold values of –0.4 and 0.4 were used based on intuition. However, as more database problems were processed, it was observed that, from the occurrence of two multiple contradicting heuristics on a word, HE7 and HA3, where the total weight is –0.3, the word is always an attribute. This is due to the fact that the list of suffixes in the set S, as described in heuristic HE7 in section 4.2.1, is not exhaustive and may be extended to suit any application domains in future work. In order to optimise the results, the negative value of the thresholds, -0.4 has been changed to –0.2. Table 4.1 shows each heuristic selected for use in the tool, *ER-Converter*, with their assigned weights. The method and justification for their selection is explained in Section 4.6.

The heuristics' weight values are assigned according to the confidence level that the event is true. For example, HE1 (one of the heuristics to determine entity types) states that a common noun may indicate an entity type. It has been given a weight of 0.5.

Though the assignment of the weights is based on intuition, these weights were also compared and reflected against the results obtained from the training set. For each of the heuristics, the average correctness of the heuristics applied were produced and compared against the intuition-based values. As the sample of the training set is small with a total of 10 examples, it is difficult to place weights simply by judging the results on this basis. In addition, the granularity of the weights is not critical at this point. For example, a choice of weight with the value of either 0.7 or 0.68 is not likely to make much difference to the predicted result. However, when the weight is placed with a 0.1 difference between each heuristic (for example 0.8 and 0.9), the total weight would provide a marked indication of the ER category for a word, depending on the threshold values.

Heuristic	*Weight*	*Status*
HEX	100	New
HE1	0.5	Old
HE7	0.6	New
HE8	0.7	New
HE9	0.7	New
HA1	-0.9	Old
HA2	-0.7	Old
HA3	-0.9	Old
HA7	-0.5	New
HA8	-0.8	New
HR4	0.8	New
HR5	0.8	New
HC2	0.9	New
HC3	0.6	New
HC4	0.5	New
HC5	0.5	New
HC6	0.5	New

Table 4.1: Heuristics' weights

Most of the values assigned lie between –1 and 1 with the exception of HEX which is assigned a value of 100. This value acts as a safe border that differentiates between an entity type and a non-entity type. For example, there may be much evidence occurring for a word indicating it is an entity type. This is reflected in the total sum of the weights of all evidence found. As both entity types and non-entity types have positive values, a value of 100 and over indicates that a word is a non-entity type.

Consider this example:

“A company undertakes projects for clients. The client’s name and address are stored”.

In the first sentence, the noun “company” is identified as a non-entity through the application of HEX. It has been assigned a weight value of 100. There is evidence that suggests “client” is an entity type from the application of heuristic HE1. HE1 states that a noun may indicate an entity type. However, in the second sentence, “client” may be an attribute type due to the occurrence of the phrase “are stored” as described in heuristic HA7. The total weight of the two heuristics is the summation of both weights ((-0.5) +0.5) which results in 0. This value lies between thresholds –0.2 and 0.4 inclusive which triggers prompting for user assistance in determining the category of the word “client” (either an entity type or attribute type).

4.5 Results on training dataset

In order to test the newly developed heuristics, a manual test was completed prior to the implementation of an automated tool, ER-Converter. This stage is seen as an important phase as the heuristics’ contributions need to be ascertained before proceeding to the implementation phase. Ten examples, as given in Appendix C, were selected as the training dataset. These examples, which are natural language requirements’ specifications, were gathered mainly from database text books.

Table 4.2 presents the training dataset results. The evaluation measures considered at this stage are *Correct, Incorrect* and *Ask User*. *Correct* refers to an entity relationship (ER) element in the solution that matches an element in the ‘answer key’ which were solutions produced by a human analyst. An ER element is regarded *incorrect* if there is a contradicting match with the answer key. An ER element is categorised under *Ask_User* when the total weight lies between the threshold of –0.2 and 0.4 and user assistance is requested in obtaining its ER category.

Example	*Correct*	*Incorrect*	*Ask user*	*% correct*
Dept. Proj	25	0	1	96.2
Instr_Course	29	1	0	96.7
Supplier	22	1	0	95.7
Hospital	44	7	2	83.0
Vehicle Registration	14	3	1	77.8
Articles	29	4	2	82.9
Buildings	25	4	2	80.7
Documents	19	1	0	95.0
Vehicle Driver	17	2	0	89.5
Training Course	34	10	0	77.3
Average				87.5

Table 4.2 Training dataset results

In Table 4.2, both *Vehicle Registration* and *Training Course* have a low percentage correctness. Most of these errors resulted from the use of the conjunction ‘and’ in the sentences, whereby the nouns occurring after this conjunction are treated as entity types. As this does not hold true for all cases as the nouns may also imply attribute types, this contributes most of the errors in both examples. The overall percentage of correctness of the training dataset is 87.5%. This indicates that the heuristics-based approach is viable. However, the contribution of the new heuristics needs to be ascertained in order to determine their utility.

Table 4.3 shows contribution of the new heuristics for each example in the training dataset. The following formulas (1) and (2) are used to calculate the contribution of the heuristics:

$$Contribution\_new = \frac{Frequency\_correct\_new}{Frequency\_correct\_new + Frequency\_correct\_old} \qquad (1)$$

$$Contribution\_old = \frac{Frequency\_correct\_old}{Frequency\_correct\_new + Frequency\_correct\_old} \qquad (2)$$

The previously published heuristics contribute most in the training dataset results with an average of 62.3%. However, based on these training dataset results, it can be concluded that the new heuristics show adequate contribution that merits further steps to be taken in terms of implementation to confirm this on a larger dataset.

Example	*Frequency correct (new)*	*Frequency correct (old)*	*% contribution (new)*	*% contribution (old)*
Dept. Proj	6	19	24.0	76.0
Instr_Course	6	23	20.7	79.3
Supplier	9	13	40.9	59.1
Hospital	17	27	38.6	61.4
Vehicle Registration	7	7	50.0	50.0
Articles	10	19	34.5	65.5
Buildings	8	17	32.0	68.0
Documents	4	15	21.1	78.9
Vehicle Driver	6	11	35.3	64.7
Training Course	27	7	79.4	20.6
Average			37.7	62.3

Table 4.3 Contribution of new and old heuristics

4.6 Justification on heuristics' selection

Presented with a collection of existing and newly proposed heuristics, a decision needs to be made on selecting a set of heuristics for the implementation. In the context of this research, ideally the three defining criteria that can be used for heuristic's selection includes:

a) application can be automated
b) frequency of use and percentage of correctness

For the manual investigation using the training dataset, criterion (b) has been chosen to select the heuristics. An analysis on the frequency of count, percentage of use and percentage of correctness for each of the heuristics (both existing and new) is carried out and the result is presented in Table 4.4.

Heuristic	*Weight*	*ER-element*	*Old/ New*	*Frequency count of heuristics applied*	*% use*	*% correct*	*Selection (Yes/No)*	*Difficulty of automation (High/Low)*
HEX	100	Non entity	New	31	9.9	93.5	√	Low
HE1	0.5	Entity	Old	135	43.3	82.2	√	Low
HE3	0.5	“	Old	0	0.0	-	X	Low
HE4	0.5	“	Old	0	0.0	-	X	Low
HE5	0.5	“	Old	0	0.0	-	X	High
HE6	0.5	“	New	1	0.3	100.0	X	Low
HE7	0.6	“	New	18	5.8	77.8	√	Low
HA1	-0.9	Attribute	Old	32	10.3	93.8	√	Low
HA2	-0.7	“	Old	16	5.1	100.0	√	Low
HA3	-0.9	“	Old	12	3.8	100.0	√	Low
HA4	-0.4	“	Old	0	0.0	-	X	High
HA5	-0.3	“	Old	0	0.0	-	X	Low
HA6	-0.3	“	Old	1	0.3	0.0	X	Low
HA7	-0.5	“	New	24	7.7	79.2	√	Low
HA8	-0.8	“	New	13	4.2	100.0	√	Low
HR1	0.5	Relationship	Old	0	0.0	-	X	Low
HR2	0.5	“	Old	0	0.0	-	X	High
HR3	0.3	“	New	1	0.3	0.0	X	Low
HR4	0.8	“	New	19	6.1	89.5	√	Low
HC1	0.5	Cardinality	Old	0	0.0	-	X	Low
HC2	0.9	“	New	4	1.3	100.0	√	Low
HC3	0.6	“	New	4	1.3	100.0	√	Low
HC4	0.5	“	New	2	0.6	50.0	√	Low
Total				313	100			

Table 4.4: Frequency count of heuristics applied in the training sets

The selection of the final heuristics is based on the percentage of use and correctness of each of the heuristics when applied to the training set. The following formula is used to calculate each of the heuristics' use in the training set:

$$Use = \frac{Frequency\_of\_heuristics\_applied}{Total\_frequency\_of\_heuristics\_applied} \times 100$$

A heuristic was selected if it was applied in more than one case. In addition to the selected heuristics in Table 4.4, five new heuristics were added to the final list. These are HE8, HE9 HC5, HC6 and HR5. These additional heuristics resulted from the refinement of existing heuristics and new evidence found prior to the selection of the test dataset. The refinement of the existing heuristics resulted after the manual test was carried out. For example, heuristic HE8 states that if a noun occurs before the verb 'has'/ 'have', it may indicate an entity type. Previously, only nouns that occur after the verb phrase were considered to denote attribute types, as defined by heuristic HA3.

One of the existing heuristics, HE2, is omitted from the selection. The main reason for the omission is due to the fact that HE2 is a subset of HE1. Since common nouns occur regularly in the specifications, the application of both heuristics at the same time would lead to biases. This is due to the fact that the chances that a noun is an entity are always higher due to the total weight of both heuristics. Thus, a decision was made to omit HE2 from the selection.

4.7 Summary

This chapter has discussed heuristics in general and their application to database design. The main focus of this chapter is on the previously published heuristics found in the literature and the proposal of a set of new heuristics to improve the results from the semi-automatic generation of ER models based on natural language requirements' specifications. A manual test shows 87.5% correctness in determining the ER elements using the combination of both the new and existing heuristics. The new heuristics contribute 37.7% towards the correctness of the results. This provides a solid ground for the proposed heuristics to be implemented. A total of 12 previously published and newly proposed heuristics out of 21 in total were selected for the implementation, as listed in Appendix E. The next chapter discusses the implementation of the previous and newly proposed heuristics in an automated tool, *ER-Converter*.

Chapter 5

ER-Converter

In the previous chapter, a heuristics-based methodology was proposed to transform a natural language requirements' specification in English into an Entity-Relationship (ER) model. This chapter discusses how this approach has been implemented. The tool is called *ER-Converter* and is implemented in Perl. Section 5.1 discusses how the input text is syntactically analysed by ER-Converter and how the output of this analysis is used to derive the elements of the ER model based on the application of heuristics. A worked example is used to illustrate the application of the heuristics.

5.1.The Entity-Relationship Converter tool

Figure 5.1 shows the architecture of the *ER-Converter* tool. Natural language requirements' specifications are to be parsed by the Memory Based Shallow Parser (MBSP) before any further processing is performed. The parsed and tagged file is then analysed by ER-Converter to produce the ER elements from the specification.

The size of the ER-Converter program is approximately 1200 lines of code. On average, ER-Converter takes 1 second to completely process a specification and generate the ER elements, but this excludes the parsing. The parsing takes approximately 1.5 seconds to parse a sentence and produce the part of speech tags for each word in the sentence. The size of the specifications range from 24 to 100 words each. For the testing, a Pentium II computer system, which runs Windows ME, was used. The hard disk was 20GB with 248MB RAM.

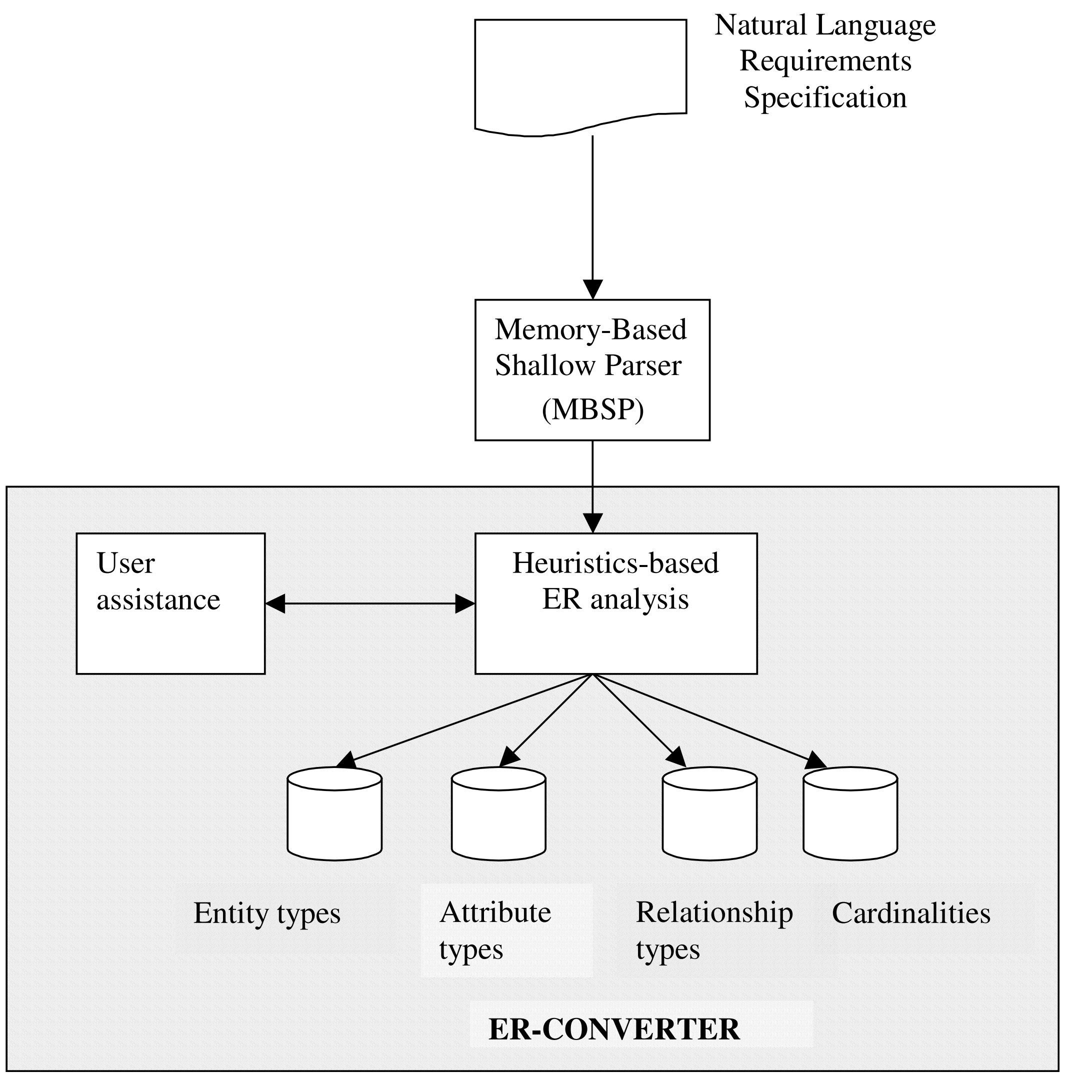

Natural Language Requirements Specification
Memory-Based Shallow Parser (MBSP)
User assistance
Heuristics-based ER analysis
Entity types
Attribute types
Relationship types
Cardinalities
ER-CONVERTER

Figure 5.1: Architecture of ER-Converter tool

The processing task requires several steps to be carried out in order to achieve the desired ER model from the natural language input, each of which is listed below:

- Step 1: Part of speech tagging using Memory-Based Shallow Parser (MBSP)
- Step 2: Identification of attributes and entities from tagged text file
- Step 3: Human intervention
- Step 4: Attachment of attributes to their corresponding entity
- Step 5: Attachment of entities to their corresponding relationship

- Step 6: Attachment of entities to their corresponding cardinality
- Step 7: Output of final result

The following sections detail each of the steps involved. An example scenario called 'Company' is used throughout these steps to illustrate the tasks carried out:

The company is organized into departments. Each department has a unique department name and a unique number. A department may manage many employees but an employee can only be managed by one department. The employee name and employee id are recorded. A department may have several locations.

Figure 5.2 shows the completed ER model, a diagrammatic form or the 'answer key' for the given scenario.

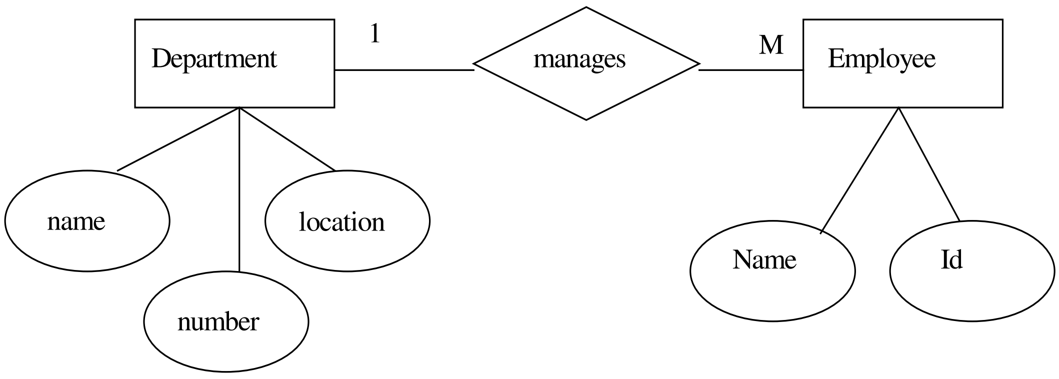


Figure 5.2: An Entity-Relationship model of the 'Company' scenario

5.1.1 Part of speech tagging using Memory-Based Shallow Parser (MBSP)

The process begins by reading a plain text file containing a requirements' specification of a database problem written in English. Examples of such specifications can be found in Appendix C. In order to obtain the corresponding syntactic category for each word, the sentences must be parsed. For this purpose, a parser is required to obtain each word's part-of-speech (POS) tag. Part of speech tagging means assigning each word in an input sentence its proper part of speech such as noun, verb and determiner to reflect the word's syntactic category (Brill, 1992). The parser used here is Memory-Based Shallow Parser (MBSP) (Zavrel and Daelemans, 1999), which includes a Memory-Based Tagger (MBT) (Daelemans et al., 1996) to produce part-of-speech tags for each word in a sentence. MBT takes an annotated corpus as input, and produces a lexicon and memory-based POS tags as output. The POS tags provide a useful abstraction of words whereby words can be classified according to their POS classes. For example, a noun may indicate a potential entity or attribute depending on its context. The result obtained from the last sentence from the 'Company' scenario using MBSP is shown in Table 5.1:

Words tagged	*Result*	*Meaning (Penn Treebank POS)*
A	DT	Determiner
department	NN	Noun, singular mass
may	MD	Modal
have	VB	Verb, base form
several	JJ	Adjective
locations	NNS	Noun, plural
.	.	.

Table 5.1: Example using MBSP

5.1.2 Identification of attributes and entities from tagged text file

The parsed and tagged text is then fed into ER-Converter to identify suitable data modelling elements from the specification. The aim at this stage is to extract the nouns, verbs and adjectives which are the indicators of the entities, attributes, relationships and cardinalities of an ER model from the tagged input. Normally, a noun may indicate an entity or an attribute. A verb may indicate a relationship. A cardinal number or an adjective may indicate cardinality. The input text is read and processed sentence by sentence by ER-Converter.

The set of heuristics identified in Chapter 4 are applied to further determine which ER element (entity, attribute, relationship or cardinality) a word belongs to. This is the core part of text analysis in ER-Converter. Heuristics are applied to any relevant words within certain word categories that meet the heuristics' criteria. As there are exceptions and ambiguities when dealing with the ER constructs, a human instructor may overrule these heuristics at a later stage.

Part of the algorithm to *apply* a selection of heuristics to determine entities and attributes is presented in Figure 5.3. In this context, 'apply' refers firstly to the identification of a match, which meets the heuristic's condition. When a match is found, the applied heuristic's name, weight value, sentence number and the ER element it indicates are stored in a record. The ordering of the heuristics under the same category (for example, heuristics to determine entities) is not significant.

```
1.   FOR each sentence in parsed and tagged text file, DO:
2.   FOR each noun, DO:
     2.1. Check for compound nouns.
          IF the noun is followed consecutively by other nouns, check the last noun.
              IF last noun NOT in the set SUFFIX where SUFFIX = {number, no, id, code,
              address, name, date, type, volume}, apply 'HE7' else apply 'HA8'.
     2.2. Check for TERM_SUFFIX noun.
          IF the noun takes the general form of TERM_SUFFIX, apply HA1 to it.
     2.3. Check_storage_indication:
                IF the verb phrase 'is stored', 'are stored' OR 'to be stored' is present, check the
                preceding noun phrase. Apply 'HA7' to the noun in the noun phrase.
                ELSE
                IF the verb phrase 'is recorded', 'are recorded' OR 'to be recorded' is present, check the
                preceding noun phrase. Apply 'HA7' to the noun in the noun phrase.
                ELSE
                IF the verb phrase 'is of interest' or 'are of interest' is present, check the preceding
                noun phrase. Apply 'HA7' to the noun in the noun phrase.
                ELSE
                IF the verb phrase 'is kept' or 'are kept' is present, check the preceding noun phrase.
                Apply 'HA7' to the noun in the noun phrase.
     2.4. IF the verb phrase 'have' or 'has' is present before the noun, apply 'HA3' to the noun
          ELSE
     2.5. IF the noun phrase is followed by the phrase 'identified by', apply 'HA2' to the following
          noun. ELSE
     2.6. IF the noun belongs to one of these set EXCLUDE where EXCLUDE ={record,
          database, company, system, information, organization, details}, apply 'HEX' ELSE
     2.7. Apply 'HE1' to noun
```

Figure 5.3: Extract from the algorithm for heuristics to determine entities and attributes

In ER-Converter, the heuristics for determining entities and attributes are applied first followed by the heuristics for determining relationships and cardinalities. This is performed in such a way that entities and attributes are already determined once the relationships and cardinalities are identified to enable the necessary attachments of the corresponding entities to these ER elements. Figure 5.4 shows an extract from the algorithm to identify relationships.

```
FOR each sentence, DO:
        FOR each word, DO:
                IF word is of type 'verb' DO:
                    IF next word is of type 'preposition', check for noun_before and
                       noun_after
                        IF noun_before is an 'entity' AND noun_after is an 'entity'
                                Attach_relationship to both 'entities and' apply HR4
```

Figure 5.4 Extract from algorithm for heuristics to determine relationships

Before an attachment is completed, the roles of the nouns to be attached to the relationships are checked to ensure that they have been identified as 'entities'. A similar procedure is used to determine cardinalities from the parsed and tagged text file. Figure 5.5 shows an extract from the algorithm to determine cardinalities:

To illustrate the processes used, consider the following sentence from the 'Company' scenario:

"A department may have several locations."

```
FOR each sentence, DO:
    FOR each word, DO:
        IF word is 'many' or 'any', apply heuristic HC2
        ELSE
        IF word is 'more than', apply heuristic HC3
        ELSE
        IF word is 'one' or 'single', apply heuristic HC4
        ELSE
        IF word is 'none' or 'zero', apply heuristic HC5
        ….
         ….
        Check for noun_before and noun_after
            IF noun_before is an 'entity' AND noun_after is an 'entity'
                Attach cardinality to both 'entities'
```

Figure 5.5 Extract from algorithm to determine cardinalities

From this sentence, the noun 'department' appears before the 'have' phrase. ER-Converter will deal with this by applying heuristic HE8 (as described in Section 4.2.1) that states that a noun that occurs before the verb 'has/have' may indicate an entity type. The corresponding information regarding the application of the heuristic is stored and updated if there is further evidence that suggests 'department' is an entity type or otherwise. Figure 5.6 shows the structure of the record where information on each relevant word is stored. WORD stores the relevant word to which a heuristic is applied. WORD_SENTENCE_NO stores the position of the sentence where the word is found. This information may also be needed for attachment purposes, like the attachment of an attribute to its entity. HEURISTIC_APPLIED keeps an array of heuristics applied to the particular word being stored. TOTAL_WEIGHT is the total weight of all the heuristics applied to the word. VALUE assigns the ER element of the word, i.e. whether it is an entity, attribute, a relationship or a cardinality. All this information is needed for the analysis and the determination of the ER element value a word indicates.

Weights are assigned to each heuristic to determine the confidence of the ER element a word is assigned to. Each heuristic carries different weights (as discussed in Section 4.4). All weights for entity types, relationship types and cardinality types carry positive values. For attribute types, all of the weights are assigned negative values, i.e. if a situation occurs

```
$myrec = {
         WORD              => $string,
         WORD_SENTENCE_NO  => [@word_sentence_no],
         HEURISTIC_APPLIED => [@applied],
         TOTAL_WEIGHT      => $weight_value,
         VALUE             => $er_element,
};
```

Figure 5.6: Record structure to store words' information

where a noun may represent an attribute type and at the same time it may also indicate as an entity type, the negative weight will reduce the strength of the noun as a strong entity type. These are the only two categories that may have conflicting evidence as a noun may represent an entity type or an attribute type. Other heuristics for the relationship types and cardinality types will not contradict each other as they deal with different POS tags. Relationship types are mostly based on verbs and therefore their weights are assigned with positive values.

Each word may have single, multiple or contradicting heuristics applied to it. The construction of the sentences often influences this decision. An example where multiple heuristics are applied is as follows:

"Each department has a unique department name and a unique department number."

The compound noun "department name" consists of two compound nouns. As its second noun "name" belongs to the set S (as described in Section 4.2.3), "department name" may indicate an attribute type and therefore HA8 is applied. As it also appears after the verb 'has', HA3 is also applied. Therefore, the extracted output from ER-Converter regarding this sentence is as below:

```
department/NN name/NN has been applied with HA8, HA3
at sentence 2,2.[1]
```

[1] The number 2,2 refers to the sentence where each of the heuristic, i.e. HA8 and HA3 are applied. In this example both heuristics are applied at sentence 2.

In this case, the multiple evidence falls under the same category and would therefore strengthen a word's certainty factor in being correctly categorized as an attribute. On the other hand, if there was contradicting evidence occurring for the word, this may suggest otherwise. This conflicting evidence needs to be resolved. Therefore, each heuristic is assigned a weight depending on the confidence level. An example shows the output of ER-Converter on a word from the sample sentence and its corresponding total weight:

```
department/NN name/NN has been applied with HA8, HA3.
at sentence  2, 2.
It has the total weight of -1.7
The value is Attribute
```

In the output, details regarding the heuristics applied, the sentence number where they are applied and the total weight is shown. In the example given, two heuristics, HA8 and HA3 are applied, both at sentence number 2 through the indication 'at sentence 2, 2'. The combined weight of all heuristics applied is used to determine the if a word is an attribute or an entity, with negative values implying an attribute and positive values implying an entity.

5.1.3 Human intervention

At this stage, a human instructor may have been requested to intervene if a conflict arises in determining the ER category of a word. In cases where multiple heuristics are applied to a word that may indicate it being an entity and an attribute and the total weight lies between the threshold of –0.2 and 0.4 as explained in section 4.4, human intervention is requested. Once a response is received from the user, the value will be updated to reflect the changes. User intervention is mostly requested to solve contradicting evidence resulting from entity types and attribute types.

5.1.4 Attachment of attributes to their corresponding entities

The next step requires each attribute type identified to be attached to its parent entity type. There are two situations in which ER-Converter considers the attachment. This is elaborated as follows:

a) When the association is clear

The attachment of attributes to their corresponding entities are possible when the relationship is visibly mentioned in the specification text. For example, consider the sentence below from the ‘Company’ scenario:

“Each department has a unique department name and a unique number.”

From the example, the verb ‘has’ indicates an association between the entity ‘department’ and attributes ‘department name’ and ‘number’. In order to attach the attributes to their parent entity, heuristics applied to the words are used to establish these relationships. The link between the applied heuristics to the words in the same sentence may help to establish the attachment. Table 5.2 shows the relation between heuristics in the attachment process:

Entity type heuristic	*Attribute type heuristic*
HE8	HA3
HE9	HA2

Table 5.2: Relationship between heuristics for determining attribute attachment

HE8 states that if a noun occurs before the verb ‘has/ have’, it may indicate an entity type. HA3 states that if a noun occurs after the verb ‘has/have’, it may indicate an attribute type. These two heuristics are directly related to each other and can be used to attach the attributes with their corresponding entity. HE9 and HA2 both concern the identification of entity type and attribute type from the verb 'identified by'. A sample output from the example given shows the attachment of attributes to the entity 'department ':

```
The entity is department/NN
The attributes are
department/NN number/NN,locations/NNS
```

b) When no direct relation is mentioned in the text

In some cases, the direct relationship between the attributes and their parent entity is not explicitly given in the text. Consider this example:

"The employee name and employee id are recorded."

'Employee name' and 'employee id' are both attribute types. As there is no indication in this particular sentence to show that 'employee' and these attributes are related, the attachment between these ER elements may be overlooked due to lack of world knowledge. However, the entity 'employee' and the attribute 'employee address', for instance, may indicate that there is a relationship of entity-attribute between these two items based on the similarity of their names. Assuming that 'employee' is mentioned somewhere else in the text which indicates that it is an entity type, ER-Converter attempts to attach all the attributes with the suffix from set S where S ={number, no, code, date, type, volume, birth, id, address, name} to that entity. The output from this given example is shown as follows:

```
The entity is employee/NN
The attributes are
employee/NN name/NN,employee/NN id/NN
```

c) When no relation is stated

In cases where there is no relation mentioned between entities and their attributes explicitly in the text and the attributes fall outside the set S, both the entities and attributes remain unattached.

5.1.5 Attachment of entities to their corresponding relationships

The relationship types are identified through the application of heuristics HR4 and HR5, as discussed in Section 4.2.4. Both deal with a verb that is followed either by a preposition or an adjective to determine the relationship type. Entity types to be attached to the relationship type are determined prior to the attachment of the relationship to them. Nouns that occur prior and post to the verb, which have been identified as entity types, are the candidates for the entities to be attached. Consider the following example from the 'Company' scenario:

"A department may be managed by an employee."

In the given example, HR4 is applied to the verb and preposition 'managed by' indicating it as a relationship type. The entities attached to this relationship are 'department' and 'employee' as both nouns occur on the same sentence and were identified previously as entity types. The output is shown as follows:

```
From the relationship record:
The relationship is managed/VBN by/IN
1st entity is department/NN
2nd entity is employee/NN
at sentence  3
```

5.1.6 Attachment of entities to their corresponding cardinality

The cardinality, sometimes referred to as a *degree* of a relationship, concerns the maximum number in the participation of entity types. Basically, the cardinality can be represented as follows:

a) one-to-one (1:1)
b) one-to-many (1:M)
c) many-to-many (M:N)

Cardinality types are identified through the use of adjectives like ‘many’ or cardinal number ‘one’ in the input text. The process of attaching the corresponding entities to its cardinality is similar to the attachment in relationship types. Nouns that occur prior and post to the cardinalities, which have been identified as entity types, are the candidates for the entities to be attached. There is a constraint in the ‘Company’ scenario where a department may manage *many* employees but an employee can only be managed by *one* department. In this case, ER-Converter will apply the appropriate heuristics, i.e. HC2 for the adjective “many” to indicate a many-sided cardinality and HC4 for the cardinal number “one” to show a one-sided cardinality. ER-Converter will represent this as two separate cardinalities.

```
From the cardinality record:
The cardinality is many (department/NN, employee/NN)
at sentence  5
The cardinality is one (employee/NN, department/NN)
at sentence  5
```

The first cardinality represents the many-sided from department to employees whereas the second cardinality represents the one-sided from employee to department.

5.1.7 Output of final result

ER-Converter produces five types of output (in text format):

- A list of candidate entity types and their corresponding attributes
- A list of relationship types and their corresponding entities
- A list of cardinality types and their corresponding entities
- A list of individual ER elements (entities, attributes, relationships or cardinalities) with the heuristics applied, the sentence number where these are applied, the total weight and the ER element assigned to the word.

An extract of the overall output from the ‘Company’ scenario is presented in Figure 5.7.

```
The entity is employee/NN
The attributes are
employee/NN name/NN,employee/NN id/NN,

The entity is department/NN
The attributes are
locations/NNS,department/NN name/NN,department/NN number/NN,

From the relationship record:
The relationship is managed/VBN by/IN
1st entity is employee/NN
2nd entity is department/NN
at sentence  2

From the cardinality record:
The cardinality is  many (department/NN,employees/NNS)
at sentence  2
The cardinality is  one (employee/NN,department/NN)
at sentence  2
```

Figure 5.7: Extract of output from ‘Company’ scenario

5.2 Summary

This chapter has presented the implementation of ER-Converter, which transforms natural language input database specifications text into ER-models using a heuristics-based methodology. The architecture of ER-Converter was presented and explained. The seven steps involved in the processing of ER-Converter were described. A sample scenario, ‘Company’ was used throughout the Chapter to illustrate the various steps involved in translating natural language specifications into ER models. The final output from ER-Converter is presented in the final section. The aim is to achieve the closest match as possible to the ER model in the ‘answer key’. The following chapter will discuss a more detailed quantitative evaluation of ER-Converter’s results.

Chapter 6

Experimental results

One of the main objectives of this research is to develop heuristics as a technique to transform natural language requirements' specifications to Entity-Relationship (ER) models. This chapter will present and discuss the results of experiments carried out using the heuristics discussed in Chapter 4 and the ER-Converter tool discussed in Chapter 5. An overview of evaluation types and approaches adopted are presented whilst the evaluation metrics are introduced and the evaluation results are discussed.

6.1 Evaluation types

Evaluation plays an essential role in natural language processing for both system developers and users. It also plays a crucial role in guiding and focussing research. Hirschman and Thompson (1995) broadly distinguish three kinds of evaluation, appropriate to three different goals:

6.1.1 Adequacy evaluation

This type of evaluation determines the fitness of a system for a given task. Basically it evaluates whether the system does what is required and how satisfactorily the task is carried out.

6.1.2 Diagnostic evaluation

This refers to the production of a system performance profile with respect to some classification of the possible inputs or *test suites*. It usually involves a large amount of data as this is needed to determine the coverage of the system and fix any faults if found.

6.1.3 Performance evaluation

This type of evaluation measures the performance of a system in one or more specific areas. It is useful in cases where a comparison is made between two different implementations of a technology or successive generations of the same system. Performance evaluation has long been used in information retrieval and many of its concepts have now been suitably adopted to the area of natural language processing. Generally, there are three levels of specificity that are considered when performance evaluation is carried. The concepts that must be taken into account are:

- *Criterion*: what are the main interests of the evaluation? (e.g. precision, error rate and speed)
- *Measure*: the relevant property of system performance which attempts to determine the chosen criterion (for example ratio of hits plus misses, seconds to process and percentage incorrect)
- *Method*: how the appropriate value for a given measure and a given system is determined: some form of concurrent or post-analytic measurement of system behaviour over some benchmark task.

The most relevant evaluation in this research is *performance evaluation*. The main interest is the measurement of correctness the system produced in comparison with the correct solutions produced manually by human analysts. This will be described in more detail in the next section.

6.2 Evaluation metrics

The following metrics have been applied in accordance with the performance evaluation's level of specificity. Harmain (2000) and Harmain and Gaizauskas (2003) have also adopted these metrics in their work. In addition to the existing metrics, new ones are defined to suit the need of the quantitative evaluation of this research.

6.2.1 Criterion

The criterion applied is how closely the models produced by the system compare to those produced by human analysts (*system responses* versus *answer key*) (Harmain, 2000). In ER modelling, there is no single solution to a problem as different human analysts can usually derive different solutions due to the abstract nature of the problem. Thus, these models cannot be categorised strictly as *correct* or *incorrect* but rather *good* or *bad* (Harmain and Gaizauskas, 2003). In the evaluation of this research, the solutions produced by human analysts and those provided in database textbooks are considered as *good* models and treated as the standard models or *answer key*.

6.2.2 Measure

The approach in this research uses methods for evaluating Information Extraction systems, primarily Message Understanding Conferences (MUC) evaluations (Grishman and Sundheim, 1996), i.e. *recall* and *precision.* These are the basic measures used in evaluating search strategies. In any system, both precision and recall should be as close to 100% as possible. However, in general, an increase in precision tends to decrease recall and vice versa.

In the context of this research, the definition of *recall* and *precision* below are adopted as used by Harmain and Gaizauskas (2003) and Grishman and Sundheim (1996) and new measures are defined. The new measures are *undergenerated*, *ask_user*, *unattached* and *wrongly_attached*. In contrast to both precision and recall, all the new measures introduced below should be as close to 0% as possible. The measures employed are as follows:

Recall

Recall is the percentage of information available that is actually found. In this research context, it refers to the percentage of the correct information returned by the system. The correct information is then compared with those produced by human analysts or *answer key*. The following formula is used to calculate recall:

$$Recall = \frac{N_{correct}}{N_{key}} \qquad [1]$$

The number of the correct ER elements generated by ER-Converter is represented by $N_{correct}$. The number of the ER elements as present in the answer key or N_{key} is actually the number of correct ER elements ($N_{correct}$) plus the number of ER elements that are part of the answer key but wrongly identified in their ER element's category ($N_{part\_correct}$), the number of missing elements ($N_{undergenerated}$) and the number of solutions provided by ER-Converter through user intervention (N_{ask}). Thus, the formula is refined as follows:

$$Recall = \frac{N_{correct}}{N_{correct} + N_{part\_correct} + N_{undergenerated} + N_{ask}} \qquad [2]$$

Similarly, for the following measures, $N_{key} = N_{correct} + N_{part\_correct} + N_{undergenerated} + N_{ask}$

Overgenerated

Overgenerated measures how much extra information is found in ER-Converter output that is not found in the answer key. Harmain and Gaizauskas (2003) term this *overspecification*. This may arise from the use of synonyms in the requirements' specifications. The following formula is used to measure overgenerated:

$$Overgenerated = \frac{N_{overgenerated}}{N_{correct} + N_{part\_correct} + N_{undergenerated} + N_{ask}} \qquad [3]$$

where $N_{overgenerated}$ refers to the number of overgenerated items given by ER-Converter which are not specified in the answer key.

Undergenerated

Undergenerated is a measure to represent missing ER elements that are found in the answer key but not in the ER-Converter's output. $N_{undergenerated}$ represents the number of missing items. The following formula [4] is used to measure undergenerated:

$$Undergenerated = \frac{N_{undergenerated}}{N_{correct} + N_{part\_correct} + N_{undergenerated} + N_{ask}} \quad [4]$$

Ask_user

Ask_user measures the degree of user assistance requested of ER-Converter. This user intervention is requested when an ER-element has a low absolute value in its weight and falls between two thresholds. N_{ask} is the number of the requests for user assistance and the formula is as follows:

$$Ask\_user = \frac{N_{ask}}{N_{correct} + N_{part\_correct} + N_{undergenerated} + N_{ask}} \quad [5]$$

Unattached

Unattached is a measure of ER elements correctly identified by ER-Converter that are not attached to their corresponding items. This may refer to attributes that are not attached to their corresponding entities and also entities that are not attached to their corresponding relationships or cardinalities. This inaccuracy needs to be taken into account as the error will be reflected in the output of the system. However, unattached is measured at a second level of the evaluation. This is due to the fact that the ER elements that are unattached are in fact correct ER elements ($N_{correct}$), i.e. they are part of the answer key and correctly indentified in their corresponding ER element's category. $N_{unattach}$ represents the number of unattached ER elements. The following formula [6] is used to measure *unattached*:

$$Unattached = \frac{N_{unattach}}{N_{correct} + N_{part\_correct} + N_{undergenerated} + N_{ask}} \quad [6]$$

Wrongly_attached

Wrongly_attached measures the degree of the correctly identified ER elements that are wrongly attached to other items. Similar to *unattached*, *wrongly_attached* is measured on the second level of the evaluation. The number of the wrongly attached items is represented by $N_{wrongattach}$. The following formula [7] is used to calculate this measure:

$$Wrongly\ \ attached = \frac{N_{wrongattach}}{N_{correct} + N_{part\_correct} + N_{undergenerated} + N_{ask}} \quad [7]$$

Precision

Precision is a measure of percentage of correctness of the information produced. It reflects the accuracy of the system in obtaining the correct result. The standard precision formula is as follows:

$$Precision = \frac{N_{correct}}{N_{correct} + N_{incorrect}} \qquad [8]$$

In this research, a more detailed formula is used to evaluate the accuracy of the results produced. Apart from $N_{incorrect}$, other additional figures such as N_{ask} and $N_{overgenerated}$ need to be taken into account. The following formula is thus defined to calculate precision:

$$Precision = \frac{N_{correct}}{N_{correct} + N_{part\_correct} + N_{ask} + N_{incorrect} + N_{overgenerated}} \qquad [9]$$

Figure 6.1 shows a Venn diagram to illustrate the concept of evaluation measures used in the evaluation of this research. Each set and their meaning is explained in Table 6.1. The letters referred to the coloured regions.

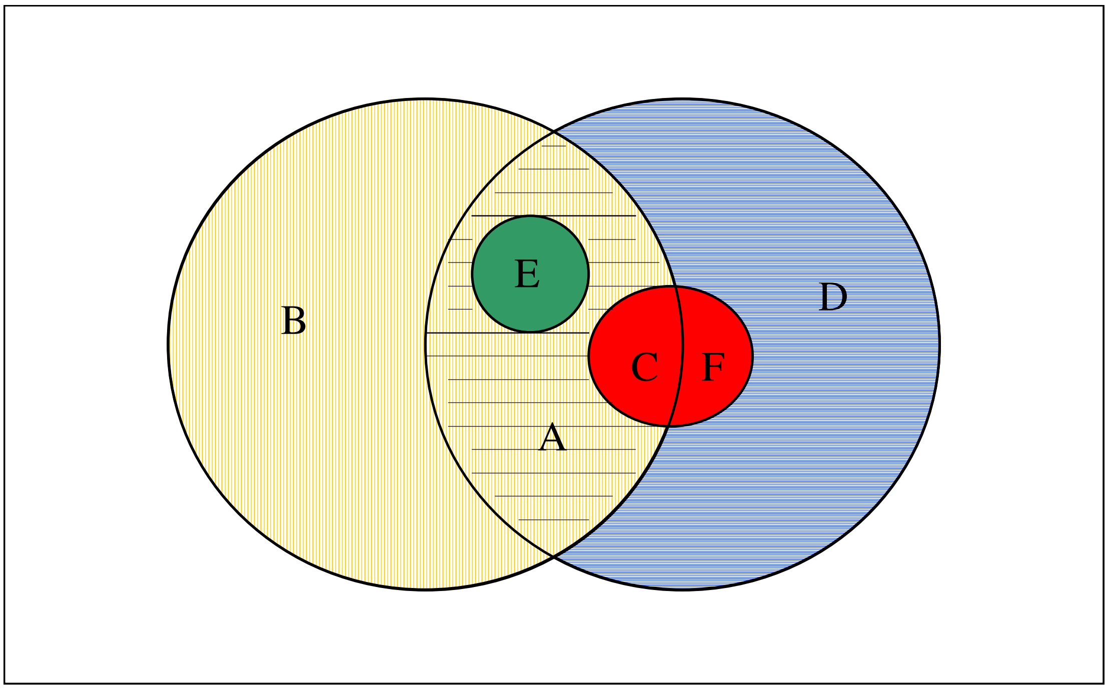


Figure 6.1: Venn Diagram to illustrate evaluation measures

Set	*Meaning*
A∪B∪C∪E	The set of correct elements produced by human analysts or known as *answer key*.
A∪C∪D∪E∪F	The set of elements retrieved by ER-Converter.
A	The set of *correct* elements retrieved by ER-Converter.
B	The set of correct elements not retrieved by ER-Converter or *undergenerated*.
C	The set of *partially correct* elements retrieved by ER-Converter whereby the elements are part of the answer key but wrongly identified in their ER element categories.
D	The set of *incorrect* elements retrieved by ER-Converter.
E	The set of elements whereby ER-Converter requests assistance from the user.
F	The set of *overgenerated* elements retrieved by ER-Converter.

Table 6.1. Definition of each of the sets in Venn Diagram (Figure 6.1)

6.2.3 Method

A manual method has been employed where the results produced by the system are compared with the answer key. The ER elements evaluated under this study include entities, attributes, relationships and cardinalities. Each of the correct answers that matches the answer key is given one point. For every error or *mismatch* that occurs, each of these is also given a point, each depending on its category. These errors can either be classified as *incorrect, overgenerated, undergenerated, ask_ user, unattached* and *wrongly_attached*.

The evaluation is split into two levels. The first level evaluates the results using the measures *recall, precision, overgenerated, undergenerated* and *ask_ user*. These measures are calculated using the formulas defined in Section 6.2.2.

The second level evaluates the attachment errors that occur in the results. The categories involved under this evaluation are *Unattached* and *Wrongly_attached*. For these categories, each of the ER elements below is given a point where they are either unattached or wrongly attached:

a) entity
b) attributes attached to corresponding entity
c) relationship
d) entities attached to their corresponding relationship
e) cardinality
f) entities attached to their corresponding cardinality

6.3 Evaluation of results

The evaluation results from applying the ER-Converter to the test dataset are presented here. The output from ER-Converter is compared against the *answer key* produced by a human analyst. The evaluation methodology as defined in Sections 6.1 and 6.2 is applied and the previously existing and new measures are utilized for evaluation purposes. The test dataset can be found in Appendix D.

6.3.1 Overall result

Table 6.2 shows the summary of the results for each sample in the test dataset.

Dataset	$N_{correct}$	$N_{part\_correct}$	$N_{incorrect}$	$N_{overgenerated}$	$N_{undergenerated}$	N_{ask}
Airplane	9	0	1	0	2	0
Bank	14	1	2	1	0	1
Boat hire	8	0	2	0	0	0
Bus	7	0	2	0	2	0
Cars	13	0	6	1	0	0
Clientnew	9	0	0	0	0	1
Company	16	1	0	0	0	1
Computernew	11	0	1	1	1	0
Doctor	8	0	1	0	1	0
Dreamhome	11	0	3	0	0	0
Elect_supp	14	0	3	0	0	0
Employee	12	0	0	0	1	0
Fault	16	3	2	1	2	1
Hospitalnew	10	0	3	0	1	0
Invoice	14	0	0	0	2	0
Library	14	0	3	0	1	0
Librarybook	20	0	1	0	2	0
Machine	11	0	0	1	0	0
Musician	18	0	0	0	1	0
Order	18	0	2	2	0	0
Painter	6	0	0	1	1	0
Photograph	10	1	1	0	1	0
Professor	21	0	0	0	1	0
Project	14	3	0	1	0	0
Reliablerentals	6	0	1	0	1	1
Salesrep	9	0	1	0	0	0
Stud_hall	13	0	0	1	1	0
Student	12	1	0	0	0	0
Travel	10	2	1	0	0	1
Univ_d'base	9	0	3	0	0	0

Table 6.2: Results from ER-Converter applied to test dataset

The raw data from Table 6.2 is then mapped to the formulas defined in Section 6.2. The results are given in Table 6.3.

Dataset	*Recall (%)*	*Precision (%)*	*Over Generated (%)*	*Under Generated (%)*	*Ask_ user (%)*
Airplane	81.8	90.0	0.0	18.2	0.0
Bank	87.5	73.7	6.3	0.0	6.3
Boat hire	100.0	80.0	0.0	0.0	0.0
Bus	77.8	77.8	0.0	22.2	0.0
Cars	100.0	65.0	7.7	0.0	0.0
Clientnew	90.0	90.0	0.0	0.0	10.0
Company	88.9	88.9	0.0	0.0	5.6
Computernew	91.7	84.6	8.3	8.3	0.0
Doctor	88.9	88.9	0.0	11.1	0.0
Dreamhome	100.0	78.6	0.0	0.0	0.0
Elect_supp	100.0	82.4	0.0	0.0	0.0
Employee	92.3	100.0	0.0	7.7	0.0
Fault	72.7	69.6	4.5	9.1	4.5
Hospitalnew	90.9	76.9	0.0	9.1	0.0
Invoice	87.5	100.0	0.0	12.5	0.0
Library	93.3	82.4	0.0	6.7	0.0
Librarybook	90.9	95.2	4.5	9.1	0.0
Machine	100.0	91.7	9.1	0.0	0.0
Musician	94.7	100.0	0.0	5.3	0.0
Order	100.0	81.8	11.1	0.0	0.0
Painter	85.7	85.7	14.3	14.3	0.0
Photograph	83.3	83.3	0.0	8.3	0.0
Professor	95.5	100.0	0.0	4.5	0.0
Project	82.4	77.8	5.9	0.0	0.0
Reliablerentals	75.0	75.0	0.0	12.5	12.5
Salesrep	100.0	90.0	0.0	0.0	0.0
Stud_hall	92.9	92.9	7.1	7.1	0.0
Student	92.3	92.3	0.0	0.0	0.0
Travel	76.9	71.4	0.0	0.0	7.7
Univ_d'base	100.0	75.0	0.0	0.0	0.0
Average	90.4	84.7	2.5	5.5	1.6

Table 6.3 Evaluation results

From the results in Table 6.3, it is observed that ER-Converter achieved a high average *recall* of 90.4%. ER-Converter has successfully produced relevant Entity-Relationship (ER) elements, i.e. elements that matched 100% of the answer key, in 23% of the problems. With a high degree of recall, the heuristics-based ER-Converter is capable of applying the corresponding heuristics to the relevant items. An investigation revealed that all of the missing or *undergenerated* items are either relationships or cardinalities. The undergenerated relationships may be due to the fact that verbs are not translated directly as relationships. However, it may not be appropriate to translate all verbs into relationships, as this does not hold true for all cases. With respect to the cardinalities, these are mainly due to synonyms and implicit phrases that imply cardinalities. For example, from the phrase "each bus is allocated a particular route", the adjective 'particular' may imply a one-sided cardinality. To overcome this, additional adjectives may be incorporated within the existing heuristics.

In terms of *precision* of the result produced, ER-Converter scored an average of 85% on the test dataset. The results support the hypotheses that a syntax-only heuristics-based approach to transform a natural language requirements' specification to an ER model can be utilized to aid conceptual modelling in the early stages of database systems development.

An investigation on the test dataset that have the highest and lowest results for precision was undertaken. The dataset *Employee, Invoice, Musician* and *Professor* have the highest percentage of *precision*, i.e. 100%. This means that the all the ER elements produced by ER-Converter are correct. However, the second level evaluation reveals that *Employee, Invoice* and *Professor* have unattached elements in the results produced whilst in *Invoice*, two of the ER elements are wrongly attached. The unattached ER elements in *Employee* are due to the fact that the sentence contains anaphoric references. This problem is explored further towards the end of this section. The unattached elements in *Invoice* and *Professor* are due to unattached entities to their corresponding relationships. *Musician* has no unattached or wrongly attached elements in its output. The 100% precision in Musician is highly influenced by the frequent use of the *have/has* verb phrases. For example, the fourth sentence is structured as follows "Each song has a title and an author". Four out of five sentences of the dataset have this structure and they were all correctly identified in terms of possession (A possess B). The respective heuristic applied in these instances is HA3.

The dataset *Cars* has the lowest precision of 65%. This is partly due to the incorrect identification of '*manufacture*' as an entity in a noun phrase. The respective sentence is as follows: "Each model is made up from many parts and each part may be used in the manufacture of more than one model". An analysis of the output produced by ER-Converter revealed that the error resulted from the interpretation of *manufacture* as an entity as it is tagged as a noun by MBSP. Heuristic HE1 has been applied. As no evidence suggests otherwise throughout the problem, *manufacture* remains as an entity due to application of HE1. This sort of error also explains why HE1 contributes most in the incorrectly applied heuristics in the applications as shown in Table 6.3. In the actual solution of the problem *manufacture* is not an ER element. This incorrect entity is then attached to a relationship and a cardinality, which adds further errors.

An interesting result to note is on the degree of the user assistance, referred to as the *Ask_User* measure in the evaluation. A user's response is sought when ER-Converter is unsure as to whether an ER element is an attribute or an entity, depending on their weights. From the evaluation results, it is evident that human intervention in the ER-Converter is minimal with only 1.6% of *Ask_User* on average. Despite the low amount of interaction with the user, the overall results from ER-Converter are good. Although full automation is seen as impossible due to incomplete presentation of knowledge, ambiguities and redundancies (Eick and Lockemann, 1985), this research has shown that it is still possible to provide an almost complete automation with limited user assistance on the solutions produced. The strength lies in the use of present and newly formed heuristics.

Table 6.4 presents the results of the unattached and wrongly attached ER elements. These ER elements are correctly identified but either unattached or wrongly attached to their corresponding items. These attachment problems may result from structural ambiguities of the sentences in the specification text, lack of world knowledge and some limitations of ER-Converter. These limitations are elaborated further in Section 6.3.5.

From Table 6.4, the average percentage of unattached elements for the test dataset is 7.9%.

Dataset	$N_{unattached}$	$N_{wronglyattached}$	*Unattached (%)*	*Wrongly_ attached (%)*
Airplane	3	1	27.3	9.1
Bank	0	0	0.0	0.0
Boat hire	0	1	0.0	12.5
Bus	0	0	0.0	0.0
Cars	0	1	0.0	7.7
Clientnew	4	0	40.0	0.0
Company	2	0	11.1	0.0
Computernew	0	0	0.0	0.0
Doctor	0	1	0.0	11.1
Dreamhome	0	2	0.0	18.2
Elect_supp	4	2	28.6	14.3
Employee	6	0	46.2	0.0
Fault	1	3	4.5	13.6
Hospitalnew	0	0	0.0	0.0
Invoice	2	0	12.5	0.0
Library	2	3	13.3	20.0
Librarybook	1	0	4.5	0.0
Machine	0	0	0.0	0.0
Musician	0	0	0.0	0.0
Order	0	1	0.0	5.6
Painter	0	0	0.0	0.0
Photograph	0	0	0.0	0.0
Professor	2	0	9.1	0.0
Project	2	0	11.8	0.0
Reliablerentals	1	0	12.5	0.0
Salesrep	0	6	0.0	66.7
Stud_hall	0	1	0.0	7.1
Student	0	4	0.0	30.8
Travel	2	1	15.4	7.7
Univ_d'base	0	0	0.0	0.0
Average			**7.9**	**7.5**

Table 6.4: Unattached and wrongly_attached results

Employee has the highest unattached elements with 46.2%. An investigation showed that the sentence below from Employee contributes a significant proportion of the error:

"His name, address, telephone number, job title, date of joining and salary are to be kept."

From this sentence, ER-Converter is unable to determine whom 'his' is referring to. This type of ambiguity is referred to as 'anaphoric reference'. Due to this lack of knowledge and limitations of the tool, this results in the unattached attributes of the entity 'employee'. Semantic interpretation of the sentences may help in resolving the problem of anaphoric references. A suggestion on how the sentences could be structured to minimize this type of error is described in Section 6.3.5.

The wrongly attached ER elements have an average of 7.5% in the final results. The dataset *Salesrep* generates the most wrongly attached ER elements, 66.7%. The errors resulted mostly from the wrongly attached nouns that appear in the same sentence as the actual ER elements to be attached. This is one of the limitations of the ER-Converter in handling the juxtaposition of the nouns identified as entity types, where the leftmost noun is always selected to be the parent entity, though this may not be true in all cases. However, the result can be improved if the sentence wording is re-structured accordingly. In the test dataset, most of the database problems, which were taken from database textbooks, were not modified or pre-edited before they were parsed and processed by ER-Converter.

6.3.2 Contribution of individual heuristics

Table 6.5 shows the percentage of correctness of each selected heuristic in the test dataset. Each heuristic's individual contribution is also presented. HA1 and HC5 do not contribute at all due to nil application in the test dataset. The new heuristics contribute 55% of the total frequency of correctly applied heuristics in the test dataset. In terms of individual contribution of the new heuristics, HA7 has the highest contribution of 8.3%. HA7 deals with specific verb phrases, as described in Section 4.2.3, which may indicate attribute types. As these phrases occur quite frequently in requirements' specifications, this is the main factor in the contribution of HA7.

Heuristic	*Status*	*Frequency correct*	*Frequency incorrect*	*% correct*	*% contribution*
HEX	New	34	1	97	6.0
HE1	Old	114	32	78	20.0
HE7	New	26	6	81	4.6
HE8	New	44	3	94	7.7
HE9	New	15	3	83	2.6
HA1	Old	0	0	0	0
HA2	Old	30	1	97	5.3
HA3	Old	113	12	90	19.9
HA7	New	47	5	90	8.3
HA8	New	45	1	98	7.9
HR4	New	33	6	85	5.8
HR5	New	32	9	78	5.6
HC2	New	14	2	88	2.5
HC3	New	3	0	100	0.5
HC4	New	17	3	85	3.0
HC5	New	0	0	0	0
HC6	New	2	1	67	0.4
Totals		569	85		100
Average (new)				55	

Table 6.5: Frequency of heuristics applied correctly and incorrectly

In Table 6.5, HA3 and HE1 are the two most frequently applied heuristics in the test dataset. HA3 deals with 'have/has' verb phrases. This verb phrase occurs very frequently in almost every requirements' specification to show possession or attributes of entities. Hence, the high frequency of its application is expected. HE1 deals with nouns. As the number of nouns far outweighs other parts of speech (POS), HE1 is frequently applied.

6.3.3 Weight applications

Weights are assigned to each of the heuristics to determine their reliability in determining an ER category. This approach has been realised in the implementation and proves useful in determining an ER element. The following categorization describes how an ER element can be determined by utilizing the weights applied to it:

a) Single heuristic

The application of single heuristics accounts for 53% of the cases in the test dataset. In these cases, the weight of each of the individual heuristics decided which category an element fell into. An example from the dataset *Airplane* is given as follows:

```
airplane/NN has been applied with HE8.
at sentence  1.
It has the total weight of 0.7
The value is Entity
```

b) Multiple concurring heuristics

The example below shows one of the many cases where multiple heuristics are applied in the test dataset. This accounts for 39% of the cases in the test datasets. When each of the heuristics belongs to the same category (for example HA2 and HA3 both indicate attributes), this adds to the total value of the element's weight. Hence, the more evidence or heuristics are applied, the stronger or higher the level of confidence in an item.

```
capacity/NN has been applied with HA2, HA3.
at sentence  3, 3.
It has the total weight of -1.6
The value is Attribute
```

c) Multiple contradicting heuristics

Another category of multiple heuristics is contradicting heuristics. These heuristics account for around 8% of the cases in the test datasets. User intervention will be requested when the weights of these heuristics lie between identified threshold values. There are cases where two contradicting heuristics may not need user intervention where one of the heuristics has a significantly stronger weight than the other. An example below shows an example of multiple contradicting heuristics.

```
vehicle/NN has been applied with HE2, HA7.
at sentence  2, 2.
It has the total weight of 0
The value is Ask User
```

6.3.4 Rejected heuristics' results

In chapter 4 (Section 4.5), a manual investigation using a training dataset was carried out to select the heuristics to be implemented based on certain criteria. This section looks again at those heuristics that were rejected in the process. The main reason why these heuristics were rejected was due to nil contribution or poor frequencies of their application in the training set. However, it remains possible that they could contribute to a different dataset and hence must be considered at this point.

As shown in Table 6.6, there is evidence that the rejected heuristics are not making any significant contributions in the test dataset. Two measures were taken into consideration:

i) frequency of use

ii) percentage of correctness of the applied heuristics.

Though HE6 and HA6 shows an encouraging percentage on correctness, their low application across the 30 samples in the test dataset is not convincing enough to warrant them in terms of implementation. In addition, the overall aim of the heuristics' selection is to find a small set of manageable heuristics to be implemented.

Heuristics	*Frequency applied*	*Frequency correct*	*% correct*
HE3	0	0	-
HE4	3	1	33.3
HE5	-	-	-
HE6	3	2	66.7
HA4	2	0	0.0
HA5	24	0	0.0
HA6	3	2	66.7
HR1	10	0	0
HR2	69	26	37.7
HR3	0	0	-
HC1	0	0	-

Table 6.6: Rejected heuristics' frequencies in test dataset

6.3.5 Problems identified as result of evaluation

Ambiguities that may be present in requirements' specifications may lead to inaccuracy or errors in ER-Converter's output. The different types of ambiguities that may arise have been discussed in Chapter 3. In this section, sentence constructs are analysed and discussed. Most of the problems are due to comma splices and coordinating conjunctions like *and*, *but* and *or*. Pre-processing rules to be adhered to when preparing natural language requirements' specifications, which will improve further the accuracy of the output are presented below.

Attachment problem

In entity-attribute attachment, ER-Converter utilizes some of the applied heuristics in determining which elements are to be attached. As ER-Converter attaches the attributes of an entity to its parent entity based on the leftmost identified entity in the sentence, some sentence structures may result in wrongly attached ER attributes to their entity. For example:

“An employee belongs to one or more libraries, each with a name and location”.

In the example, *name* and *location* are attached to employee instead of the actual entity i.e. *library*. The sentence may be broken down further to eliminate this problem. In such circumstances, an ideal solution is to break the sentence into simpler sentences. Pre-editing may be necessary to reduce the ambiguities and hence improve the results. The sentence from can be simplified further for clarity as shown below:

“An employee belongs to one or more libraries. Each library has a name and location.”

This gives rise to rule 1 as follows:

Rule 1: Break down the sentence if there are comma splices or a coordinating conjunction that may result in wrongly attached ER elements

Active/passive voice

Sentences are said to be in active voice if the subject does or "acts upon" the verb in them. The order of the active sentences can be changed in such a way that the subject is no longer *active*, but is, instead, being *acted upon* by the verb - or *passive.* As passive voice sentences sometimes add words and change the normal ‘*doer-action-receiver of action’* direction, this may result in some inaccuracy in the result of the ER-Converter. This is due to the fact that some words’ POS tags are different when the sentence is in active or passive form. An example is:

“A patient is treated by many doctors.”

This passive voice sentence can be changed to an active one as follows:

“Many doctors treat a patient.”

The active voice sentence is more direct and hence helps ER-Converter in establishing a relationship between ‘doctors’ and ‘patients’. This rule can be stated as below:

Rule 2: Convert all passive voice sentences to active voice form.

The limitation of the verb ‘has’

The verb ‘has’ is the basis of one of the heuristics to determine attribute types, i.e. HA3 and it is commonly used in requirements’ specifications mainly to indicate a relationship between an entity and its attributes. However, in certain circumstances, the use of ‘has’ may mean otherwise. Consider the following example:

“Each department has a set of employees, a set of projects and a set of offices.”

In this example, all the nouns after the verb ‘has’ do not indicate attributes of the entity ‘department’. ‘Employees’, ‘projects’ and ‘offices’ are also entity types that exist in the business environment. However, from the application of the heuristic HA3, ER-Converter may suggest that they are all attribute types based on this sentence alone. Therefore, for such a sentence, this may be better replaced with other synonyms like ‘consists of’, ‘comprises’ or ‘contains’. The sentence can be rewritten such as follows:

“Each department consists of a set of employees, a set of projects and a set of offices.”

A rule on this limitation can be defined as follows:

Rule 3: For every sentence that contains the verb ‘has’ and where the nouns following this verb may not indicate attribute types, replace the verb with other suitable synonyms.

6.4 Summary

This chapter introduced an evaluation methodology commonly used for Information Extraction systems which has been adopted in this research. The selected methodology is based on two main evaluation metrics, namely *recall* and *precision*. Other metrics, *undergenerated, ask_user, unattached* and *wrongly_attached* were introduced to measure the accuracy of results produced by ER-Converter.

This chapter also presented results from applying ER-Converter to the test dataset which comprises 30 database problems in the form of natural language specifications. ER-Converter has an average of 90% recall, 85% precision, 3% overgenerated, 6% undergenerated and 2% ask_user for the test dataset. With regard to attachments, ER-Converter has an average of 7.9% *unattached elements* and 7.5% *wrongly attached* elements. The new heuristics contribute 55% of the total frequency of correctly applied heuristics in the test dataset. HA7, a heuristic that deals with 'has/have' verb phrases, has the highest percentage of the individual contribution of the new heuristics. The rejected heuristics are considered in the test dataset and the results show that they are not making any significant contribution. This chapter also identified some problems as a result of the evaluation and suggested some rules to be adhered in processing natural language requirements' specifications which will improve the accuracy of output generated by ER-Converter. The results support the hypothesis that the newly formed heuristics do contribute in generating ER models in an automated environment.

Chapter 7

Conclusion and Future Work

This final chapter begins by summarising the thesis. This is followed by comparison of the results produced by ER-Converter with other related work. Potential avenues of research for future work are also explored.

Chapter 1 established that conceptual modelling is one of the most important and difficult stages in the software lifecycle of an information system (Connolly and Begg, 1999). It was noted that it is common for designers to use ER models as a representation of the conceptual design. However, due to its abstract nature, ER modelling can be a daunting task to designers and students alike (Storey and Goldstein, 1988; Batra and Antony, 1994; Moody, 1996; Marsden and Staniforth, 1996; Antony and Batra, 2002). Much research has attempted to apply natural language processing in extracting knowledge from requirements' specifications with the aim to design databases. However, research on the formation and use of heuristics to aid the construction of logical databases from natural language has been scarce. This thesis has developed new heuristics to assist the transformation from natural language requirements' specifications to ER models.

In order to accomplish the generation of ER models from natural language requirements' specifications, the following objectives were achieved:

- review of previous work on heuristics in database design
- review systems that apply natural language in database design
- examine NLP tools and techniques and identify the most suitable ones to be utilised in this research

- define new heuristics to assist the transformation from natural language requirements' specifications to ER models
- implement the heuristics proposed in a tool called *ER-Converter*
- test ER-Converter using datasets that comprise domain independent database problems
- evaluate the approach against human performance and compare it with similar previous work in the area

The importance of ER Modelling is addressed in Chapter 1. The difficulties associated with ER modelling were also presented. The results of a survey regarding the Databases' subject's difficulties were also discussed. Previous work that applies NLP to Databases are reviewed. Various techniques like the use of logical forms and rules were used in the work reviewed to translate natural language requirements' specifications to conceptual models. Other literature studied included ITS and ITS for Databases. Currently published ITSs for Databases were reviewed. So far, none of the ITSs have a domain model that is capable of solving ER problems. ITS is one of the contexts where this research work could be applied. Natural language processing for database design was also investigated. Problems that may be encountered during processing such as the different types of ambiguities were discussed. The parser used for this work, i.e. Memory-Based Shallow Parser (MBSP) is also reviewed.

Having examined the literature, the syntactic heuristics are then proposed. Previously published heuristics are also presented and discussed. The combination of both the new and pre-existing heuristics forms the basis for the semi-automated transformation from natural language requirements' specifications to ER models. A manual test prior to the implementation shows 87.5% correctness in determining the ER elements using the heuristics. Following this, suitable heuristics, based on their computability, frequency of use and percentage of correctness were selected for the implementation. The aim of the selection is to obtain a manageable set of contributing heuristics. The next stage involved the implementation of the heuristics. The tool, ER-Converter, is implemented using Perl. A scenario was used to illustrate how the relevant heuristics are applied in the process of generating the ER model. The output from ER-Converter was then matched against the 'answer key', where the aim is to achieve the closest match possible. Following this, the results produced by ER-Converter are evaluated and discussed. New measures, in addition to the standard measures *recall* and *precision*, were introduced in the evaluation. From the test dataset used, ER-Converter has 90% recall and 85% precision. It can be concluded that these results support the hypothesis that the newly formed heuristics do contribute in generating ER models in a semi-automated environment.

7.1 Comparison with Related Work

A variety of different approaches to automate or more appropriately, semi-automate the process of ER modelling have been proposed in the past. A summary of the analysis of existing systems that apply natural language processing in database design has been presented in Chapter 2. This section compares the different approaches and test results of the systems reviewed with ER-Converter.

The most relevant work in connection with our research is E-R Generator (Gomez et al., 1999). However, no direct comparison can be carried out for two reasons. Firstly, no figures were available in terms of *recall* and *precision* on E-R Generator's overall result. In addition, both systems do not use the same datasets in the final result though efforts were made to obtain the actual test sets used by Gomez and his colleagues. Gomez's test dataset comprised 30 natural language specifications where 75% of these are mainly gathered from database textbooks and the rest are entered interactively by users. Gomez et al. (1999) reported that their E-R Generator was able to identify the relevant ER relationships and entities with 75% correctness in average. However, the result was based on only 25% of the total test dataset which were entered interactively by users. The program overgenerated or undergenerated ER entities and relationships in 50% of the cases. No overall results were revealed on the complete test dataset. With ER-Converter, the precision or the accuracy of the system in obtaining the correct result is 85%, which indicates better performance.

The goal of this research is identical to E-R Generator, i.e. to generate ER models from natural language specifications through natural language processing. A major difference between ER-Converter and E-R Generator is the high utilization of heuristics and their weights in ER-Converter to derive ER elements from natural language specifications. E-R

Generator relies more on semantic interpretation and final knowledge representation of sentences. Though it is arguable that syntactic linguistic knowledge alone may be insufficient to derive an ER model, the results from this work have demonstrated that such an approach is still viable. The strength in the approach of ER-Converter lies in the utilization of heuristics that are targeted at specific categories of words or phrases that reflects the ER modelling elements. Some of the tasks in processing the natural language specifications like the identification of attributes with certain suffixes and their entities can be solved by simple syntactic rules, as demonstrated by the application of the new and previously published heuristics. Tjoa and Berger (1993) also question whether the effort of determining semantic properties is justified compared to the achieved results.

E-R Generator identifies ER elements through the application of specific and generic rules. Specific rules use semantic cues that are relevant to database design to construct these elements. Generic rules identify the ER elements on the basis of the logical form and on the basis of the ER elements under construction. The generic rules can be classified into three categories: unary, binary or *n*-ary rules. In general, unary rules result in the identification of attributes; binary rules may define attributes, entities and relationships and *n*-ary rules result in the definition of relationships. In contrast, ER-Converter identifies ER elements on the basis of their total weight for evidences found in the input text. This is employed using heuristics rather than rules, in terms of what was discussed in Section 2.3.

In terms of user intervention, E-R Generator requires user help in resolving ambiguities like intersentential anaphora. Other user involvement includes interacting with E-R Generator when some background knowledge about a word to describe the database application is needed, as the system is based on semantic interpretation. In addition, E-R Generator also requests user help in the attachment of attributes in some cases. In ER-Converter, the attachment process is done automatically without any user intervention. To summarise, the difference between E-R Generator and ER-Converter lies mostly on the utilization of rules in E-R Generator to identify the ER-elements whilst ER-Converter works on the application of heuristics to achieve the desired ER models.

CM-Builder (Harmain, 2000; Harmain and Gaizauskas, 2003) concentrates on building object-oriented conceptual models to be represented in the Unified Modelling Language

(UML). Though it is not comparable in terms of the end results as CM-Builder produces object-oriented models and not an ER model, the techniques used in the natural language processing and evaluation are similar. CM-Builder has two versions. The first version, CM-Builder 1 performs surface analysis, i.e. to generate a list of candidates, attributes and relationships using frequency analysis. In this analysis, CM-Builder generates a list of candidate classes and attributes by calculating their frequency in the text. The resulting candidate lists need to be filtered manually by the user and in associating candidate attributes, classes and relations with each other. In ER-Converter, user intervention is only needed when a heuristic's weight is low and lies between two threshold values. This also means that not all of the specifications processed require user intervention.

CM-Builder 2 performs semantic analysis in an object-oriented analysis module (OOA) and produces three kinds of output: a list of candidate classes, relationships and a conceptual model. In this version, there is no user interaction but users are likely to be needed to further refine and extend the model produced. CM-Builder 2 utilizes WordNet, an external lexical database, to find appropriate attribute names from adjectives. Simple heuristics are also employed to find attributes. The overall performance of CM-Builder was 73% recall, 66% precision and 62% of overspecification or overgenerated items. Using the same measures, ER-Converter's results are 90% recall, 85% precision and 3% of overgenerated items. Comparing the result with CM-Builder, ER-Converter's performance is well beyond these figures, especially with the significantly low results of overgenerated items.

Other relevant work like FORSEN (Meziane, 1994; Meziane and Vadera, 2004) and Dialogue Tool (RADD) (Buchholz et al., 1995) have not published formal evaluations or tests, hence no figures are available for direct comparison on their performance. The aim of FORSEN is similar to ER-Converter and E-R Generator in which all three aim to produce ER models from natural language specifications, semi-automatically. FORSEN uses a representation language called logical forms as a technique to transform the natural language sentences to ER models. The logical forms of sentences are used as a basis for identifying the entities and relationships. Heuristics are then used to suggest suitable degrees for the identified relationships. In contrast, ER-Converter relies heavily on heuristics to determine the ER elements.

Dialgoue Tool (RADD) (Buchholz et al., 1995) uses dialogues with users in order to obtain the structure of an application in database design. The main aim is to produce an EER model from the dialogues. The transformation from the designer input to EER model is completed using world knowledge and heuristics. The heuristics used are defined and formalised using context-free and context-sensitive rules. Another system that elicits knowledge through dialogues with users is VCS (Storey, 1988). Though the aim is similar to ER-Converter, VCS employs procedural and production rules, stored in a knowledge base, in order to produce the ER models. The task relies heavily on the user to provide information regarding the requirements of the database system. Though VCS has performed some system testing, this is not a quantitative evaluation but rather a test on the usability of the system.

DMG (Tjoa and Berger, 1993) also aims to transform natural language specifications into EER models. However, it is only a proposed tool where the implementation has not been published. DMG processes the parsing results of the input language using rules and heuristics, which set up a relationship between linguistic and design knowledge. DMG gave detailed accounts of the heuristics applied in their system. The heuristics provided some basis for this thesis, in which the selected heuristics are implemented in ER-Converter. User intervention is required in DMG when the requirements are incomplete.

ANNAPURNA (Eick and Lockemann, 1985) aims to provide a computerised environment for semi-automatic database design. Their approach utilizes S-diagrams, a formalism for the description of the terminological knowledge acquired from the experts. Tseng et al. (1992) proposed a methodology to map natural language constructs into relational algebra through ER representation. Their methodology employs a logical form to represent the natural language queries. In COLOR-X (Burg and van de Riet, 1996), CEM and CSOM models are used, as discussed in Chapter 2, to facilitate the process of generating conceptual modelling. In comparison to ANNAPURNA, Tseng et al.'s (1992) approach and COLOR-X, ER-Converter processes natural language requirements directly from the input text without any intermediate representation to produce the intended ER models, through the utilization of heuristics.

Table 7.1 summarizes the evaluation results from the other systems and ER-Converter. Similarly to E-R Generator, a direct comparison to CM-Builder is not possible due to different datasets and modelling domains. However, the figures presented show some indication of the performance of ER-Converter. The *Other* column refers to the percentage of overgenerated items for each of the systems.

System	*Evaluation results*		
	Recall	*Precision*	*Other*
E-R Generator	75%	-	50%
CM-Builder	73%	66%	62%
ER-Converter	90%	85%	3%

Table 7.1. Comparison of results with related work

In comparison with other systems, ER-Converter requires minimal user intervention in generating the ER-models. This is evident from the evaluation results presented in Chapter 6, i.e. with only 1.6% user intervention. Though complete automation is extremely difficult due to the nature of ER modelling and ambiguities in natural language, limited human intervention is still possible, as shown in this thesis.

As noted from the discussion, most of the systems have accepted natural language semantics as being vitally important to the understanding process and have put time and effort into acquiring such information. Though the importance of semantics is undeniable in enhancing the results, it is interesting to investigate whether the effort is justified with the achieved results. ER-Converter concentrates mainly on the syntactic knowledge and the results are better than previously published systems. On the basis of the comparison, it can be concluded that ER-Converter has a potential in the automation of ER modelling and can be applied to other areas such as ITS for Databases.

7.2 Future Work

This section highlights suggestions which may provide grounds for future work.

7.2.1 Semantic analysis

In order to resolve a wider range of problems related to ambiguities in requirements' specifications such as anaphoric references or nominalization, without pre-processing text or using restricted language, semantic analysis of the sentences may be necessary to handle such issues. Semantic analysis involves a process whereby meaning representations are created and assigned to linguistic inputs (Jurafsky and Martin, 2000). The 'understanding' of the results of the parsing, lexical information, context and common sense reasoning is referred to as the *semantic interpretation* of the text. More expressive power can be added when semantic interpretation is used.

Semantic roles in objects like *agent*, *instrument*, *source* and *location* (Fillmore, 1971) may be helpful in interpreting possible elements of the ER model. *Semantic roles* or sometimes known as *thematic roles* are conceptual notions which provide a shallow semantic language for characterizing certain arguments of verbs (Jurafsky and Martin, 2000). Table 7.2 shows some commonly used semantic roles and their definitions.

The following example illustrates the concept of semantic roles:

$$\text{"The } \underbrace{\text{purchaser}}_{\text{AGENT}} \text{ sends an } \underbrace{\text{order form}}_{\text{THEME}} \text{ to the } \underbrace{\text{supplier}}_{\text{GOAL}}.\text{"}$$

Semantic role	*Definition*
AGENT	The volitional causer of an event
EXPERIENCER	The experiencer of an event
FORCE	The non-volitional causer of the event
THEME	The participant most directly affected by an event
RESULT	The end product of an event
CONTENT	The proposition or content of a prepositional event
INSTRUMENT	An instrument used in an event
BENEFICIARY	The beneficiary of an event
SOURCE	The origin of the object of a transfer event
GOAL	The destination of an object of a transfer event

Table 7.2: Semantic roles and their definitions (Jurafsky and Martin, 2000)

From the example, the subject, i.e. ‘the purchaser’ acts as an agent as the causer of the event. The object, ‘order form’, has the semantic role ‘THEME’ as it is directly affected by the event. ‘Supplier’ represents the GOAL where it is the destination of the transfer event. These semantic roles may assist in the pragmatic interpretation of the natural language input to an ER model. For example, the semantic roles *agent*, *goal* and *beneficiary* may indicate entity types, depending on the context. The utilization of semantic heuristics may be useful at this stage. DMG (Tjoa and Berger, 1993) and Martinez and Garcia-Serrano (2001) may provide a basis for such heuristics. For example, in DMG, one of the semantic heuristics states that if a sentence includes a comparative, then both the nouns belong to the semantic roles *subject* and *subject complement*. The adjective in the sentence represents an attribute which describes both nouns and may indicate that it may be related to the supertype of both nouns (in an enhanced-entity relationship modelling). However, Tjoa and Berger (1993) questioned whether such an effort for determining semantic properties of a sentence is justified compared to the results achieved.

In order to resolve intersentential anaphora as mentioned in Chapter 6, the maintenance of a record of all objects or nouns mentioned in the preceding sentences, known as *history list*, may be useful (Allen, 1995). Using the same sentences from the Employee database specification as given in Chapter 6, the following history list in Table 7.3 is produced.

“An employee is identified by an id. His name, address, telephone number, job title, date of joining and salary are to be kept.”

Constituent	*Object name*	*Value*
NP_1	employee	Entity
NP_2	id	Attribute

Table 7.3: History list of the first sentence in Employee

Given the history list, the antecedent of the pronoun ‘his’ in the second sentence can be determined, i.e. ‘employee’ which occurs as the first constituent of the preceding sentence and has a value of an ‘entity’. This enables the attachment problem as mentioned in Section

6.3 to be resolved. *Selectional restrictions*, the restrictions as to which constituent should be selected, imposed during semantic interpretation, may provide the necessary information to handle problems like anaphoric references (Allen, 1995). With the provision of automatic semantic role labelling tools, for example by Gildea and Jurafsky (2002), the incorporation of semantic analysis may further aid the process of transforming natural language specifications into ER models and improve the accuracy of the results.

7.2.2 WordNet

As discussed in Chapter 2, WordNet is an external lexical database for English that may be applied in natural language processing systems to incorporate linguistic knowledge. It organizes lexical knowledge in terms of word senses, whereby all the words are organized into an inheritance hierarchy. Due to ambiguities in natural language, words may have several meanings (homonyms and polysemes) and many concepts can be represented by two or more words (synonyms) (Burg and van de Riet, 1998). With the help of WordNet, the right meaning of a word in each specific universe of discourse (UoD) can be determined.

As for conceptual modelling, WordNet can be viewed as a source of reusable knowledge, which can be used to ensure that the resulting models are correct (Burg and Van de Riet, 1998). A possible extension in this research work is to integrate WordNet with ER-Converter to improve the results of the system. Among the possible uses of WordNet are:

- to disambiguate the meaning of verb or noun by examining synonyms. For instance, a user needs to know the meaning of a word in a specific UoD like ‘book’ to ensure the correct interpretation of the word in the right context. Consider this example:

  “The customer can book a place for the fishing trip.”

  In this example, the word ‘book’ is categorised as a verb. WordNet has 10 senses for the word ‘book’ as a noun and four senses as a verb. The output from WordNet for the verb category of the word ‘book’ is as follows:

1. book -- (record a charge in a police register; "The policeman booked her when she tried to solicit a man")
2. reserve, hold, book -- (arrange for and reserve (something for someone else) in advance; "reserve me a seat on a flight"; "The agent booked tickets to the show for the whole family"; "please hold a table at Maxim's")
3. book -- (engage for a performance; "Her agent had booked her for several concerts in Tokyo")
4. book -- (register in a hotel booker)

The users can then select the intended meaning from the options given. In this case, option (2) suits best the meaning of ‘book’ in the given example. The disambiguation process can help to verify consistency and correctness in the resulting ER model produced by ER-Converter.

- to discover some hidden relationships through WordNet. For instance, the relationship like ‘employee’ is a ‘person’ may be helpful in the conceptual modelling especially in aggregation and inheritance concepts. The output from the *hypernym* search category, which defines concepts to super-ordinates, has 1 sense on the word ‘employee’. This is shown in Figure 7.1. The information relating to the hierarchical concepts is useful especially when the relationships between two ER elements are not explicitly mentioned in the natural language specifications. This additional feature may enhance the accuracy of the results produced by ER-Converter.

- to identify hidden attributes like the use of adjectives (Harmain, 2000). This can be sought by using WordNet through the senses provided. For example, given a sentence, “The large branch has many departments”, the adjective ‘large’ may indicate size of the branch and hence it may be an attribute of the entity ‘branch’. This feature may help in reducing the number of undergenerated attributes in ER-Converter.

```
1 sense of employee

Sense 1
employee -- (a worker who is hired to perform a job)
       => worker -- (a person who works at a specific occupation;
          "he is a good worker")
           => person, individual, someone, somebody, mortal, human, soul --
                 (a human being; "there was too much for one person to do")
               => organism, being -- (a living thing that has (or can
                   develop) the ability to act or function independently)
                   => living thing, animate thing -- (a living (or once
                       living) entity)
                       => object, physical object -- (a tangible and
                           visible entity; an entity that can cast a
                            shadow; "it was full of rackets, balls and
                             other objects")
                           => entity -- (that which is perceived or known
                               or inferred to have its own distinct
                                existence (living or nonliving))
               => causal agent, cause, causal agency -- (any entity that
                   causes events to happen)
                   => entity -- (that which is perceived or known or
                       inferred to have its own distinct existence (living
                        or nonliving))
```

Figure 7.1: Output from WordNet for the *hypernym* search category of the word 'Employee'

7.2.3 Heuristics' weights

The heuristics' weights might have not proved optimal and the level of granularity is still debatable. A larger dataset is needed to ascertain the optimal weights. The weights could also be investigated with different granularities. The optimisation of the weights can be investigated in future work.

7.2.4 Part of a domain model in an ITS

ER-Converter could be incorporated as part of the domain model in an ITS environment. To date, no ITS for Databases that incorporates a dynamic domain model is capable of solving database problems. Most existing ITSs store the solutions prior to tutoring. ER-Converter could be embedded in the domain model and may provide improvement by allowing students to enter their own examples.

7.3 Summary

In conclusion, this work has achieved favourable results on the performance of ER-Converter, a heuristics-based tool to generate ER models automatically from natural language requirements' specifications. Though ER-Converter only performs syntactic analysis, based on the combination of previously published and newly formed heuristics to produce the ER models, the result is comparable to other systems that utilize semantics knowledge. In addition, the degree of user intervention is low, with 1.6% in the test dataset. In addition to developing ER-Converter, this work has defined new measures to be employed in the quantitative evaluation against human performance. The measures can be used as a benchmark for other similar systems to be evaluated. The evaluation results support the hypothesis that the newly formed heuristics do contribute in generating ER models in a semi-automated environment. Suggestions are made for future work to improve the accuracy of the results produced by ER-Converter. These include semantic analysis, the use of WordNet and the optimisation of the weights. The resultant software system, ER-Converter, could be applied to areas such as ITSs for teaching Databases.