RULE BASED ENTITY-RELATIONSHIP DIAGRAM MODELLING

OĞUZHAN ULUSOY
M.S., Computer Engineering, IŞIK UNIVERSITY, 2022
B.S., Computer Science and Engineering, IŞIK UNIVERSITY, 2018

Submitted to the School of Graduate Studies
in partial fulfillment of the requirements for the degree of
Master of Science
in
Computer Engineering

IŞIK UNIVERSITY
2022

IŞIK UNIVERSITY
SCHOOL OF GRADUATE STUDIES

RULE BASED ENTITY-RELATIONSHIP DIAGRAM MODELLING

OĞUZHAN ULUSOY

APPROVED BY:

Dr. Emine Ekin
Işık University
(Thesis Supervisor)

Prof. Dr. Olcay Taner Yıldız
Özyeğin University

Dr. Faik Boray Tek
Işık University

APPROVAL DATE: 07/02/2022

RULE BASED ENTITY-RELATIONSHIP DIAGRAM MODELLING

Abstract

Modern society needs to use database system since they involve many activities that are related to database interaction directly.

In this study, entity-relationship modeling using Natural Language Processing techniques is presented for the English language. Natural Language Processing refers to the capability of understanding human languages naturally, like Turkish and English, using computational power. To make this possible, combination of linguistics and current Machine Learning systems are used together. Entity-Relationship diagrams ensure to plan or trace relational databases in different fields.

In the beginning, all details of a standard database management and its components have been studied. Heuristic rules which indicate the relation between human language and database components have been defined. According to the defined heuristic rules previously, an event-based pipeline has been constructed. A full text has been analyzed and processed every word at this pipeline using Natural Language Processing techniques.

Keywords: Entity-relationship diagram/modelling, part of speech tagging (pos), database management system (dbms), relational database management system (rdbms), natural language processing (nlp), machine learning

KURALA DAYALI VARLIK-İLİŞKİ DİYAGRAMI MODELLEME

Özet

Modern topluluklar, direkt olarak veritabanı etkileşimi ile alakalı birçok aktivite günlük hayatlarında dahil olduklarından dolayı veritabanı sistemleri kullanmaya ihtiyaç duyarlar.

Bu çalışmada, İngilizce dili için Doğal Dil İşleme (NLP) tekniklerini kullanarak varlık-ilişki (ER) modellemesini temsil etmeye yönelik çalışmayı sunuyoruz. Doğal Dil İşleme, bilgisayarların hesaplama gücünü kullanarak Türkçe ve İngilizce gibi insan dillerini doğal olarak anlama yeteneği sağlar. Bunu mümkün kılmak için, dilbilim ve mevcut Makine Öğrenimi sistemlerinin birleşimi birlikte kullanılır. Varlık-İlişki diyagramları, yazılım mühendisliği, işletme bilgi sistemleri, eğitim ve araştırmada ilişkisel veritabanlarını planlamak veya izlemek için sıklıkla kullanılır.

Başlangıçta, standart bir veritabanı yönetim sistemi ve bileşenlerinin tüm ayrıntılarına çalışıldı. Doğal insan dili ve veritabanı semantiği arasındaki ilişkiler varsayımsal kurallar olarak tanımlandı. Daha önceden tanımlanan bu kurallara göre, etkinlik bazlı bir boru hattı inşa edildi. Komple bir metin analiz edilip, her bir kelime Doğal Dil İşleme yöntemleri ile işlendi.

Anahtar kelimeler: Varlık-ilişki diyagramı/modellemesi, konuşmanın bileşenleri, veritabanı yönetim sistemi, ilişkisel veritabanı yönetim sistemi, doğal dil işleme, makine öğrenmesi

To. . . I'm glad to finally sharing this study with my supervisor and family.

Table of Contents

Approval Page i

Abstract ii

Özet iii

List of Tables vii

List of Figures viii

List of Abbreviations ix

1 Introduction 1
- 1.1 **Application of entity relationship diagrams** 2

2 Literature Survey 5

3 Overview of DBMS & ERDs 8
- 3.1 Database Management Systems 9
- 3.2 Database Modeling 12
- 3.3 High-Level Conceptual Design 12
- 3.4 Database Components 14
  - 3.4.1 Entity 14
  - 3.4.2 Attribute 15
  - 3.4.3 Relationship 17
- 3.5 Proper Naming of Schema Constructs 18

4 Approach 19
- 4.1 Rule 1: Identify Entities 19
  - 4.1.1 A common noun may indicate an entity type. 19
  - 4.1.2 A proper noun may indicate an entity. 20
  - 4.1.3 In case of consecutive nouns existence, check the last noun. It may be an entity type, otherwise it may indicate an attribute. 20
  - 4.1.4 A gerund may indicate an entity. 21
  - 4.1.5 Ignore every proper noun. 21

4.2 Rule 2: Identify Attributes . . . 21
4.2.1 Noun phrase with genitive case may indicate an attribute. 21
4.2.2 The possessive case usually shows ownership it may indicate attribute type. . . . 22
4.2.3 A noun phrase such as has/have may indicate attribute. . 22
4.3 Rule 3: Identify Relationships . . . 22
4.3.1 A transitive verb can indicate relationship type. . . . 23
4.3.2 If a verb is in the current list: include, involve, comprises of, encompass, contain, split to, embrace, this suggests an aggregation or compositional relationship. . . . 23
4.3.3 Passive voice can be translated into active voice. . . . 24
4.4 Rule 4: Identify Primary Key . . . 24
4.4.1 Adverb indicates primary key of an entity. . . . 24

5 Design & Implementation Details 25
5.1 Proposed Design . . . 26
5.1.1 Segmentation . . . 27
5.1.2 Tokenization . . . 27
5.1.3 POS Tagger . . . 28
5.1.4 Chunking . . . 31
5.1.5 Parser . . . 33
5.1.6 ER Analysis . . . 34
5.2 Implementation Details . . . 34
5.2.1 System Architecture . . . 34
5.2.2 Logging & Exception Handling . . . 35
5.2.3 User-interface . . . 35
5.2.4 Trade-offs . . . 35

6 Discussion & Results 37
6.1 Discussion . . . 37
6.2 Results . . . 39
6.2.1 Case Studies . . . 39
6.2.2 Differences Between Actual And Experimental Results . . 43
6.2.3 More Experimental Results . . . 48

7 Conclusion 53

Reference 54

List of Tables

5.1 An example sentence for part-of-speech tagging. 29
5.2 An example for spacy tagging. 29
5.3 Dependency labels with descriptions between 1-23rd items 30
5.4 Dependency labels with descriptions between 24-45th items 31
5.5 An example sentence for chunking. 32
5.6 An example sentence for parser. 34

6.1 Entities are caught by rule-based entity-relationship diagram modelling algorithm 43
6.2 Attributes are caught by rule-based entity-relationship diagram modelling algorithm 44
6.3 Relationships are caught by rule-based entity-relationship diagram modelling algorithm 45
6.4 Actual entities from reference book 46
6.5 Actual attributes from reference book 46
6.6 Actual relationships from reference book 47
6.7 Numeric comparisons 47
6.8 Numeric analysis 47

List of Figures

3.1 Entity shape 15
3.2 Weak Entity shape 15
3.3 Attribute shape 15
3.4 Key Attribute shape 16
3.5 Multi-valued Attribute shape 16
3.6 Composite Attribute shape 16
3.7 Derived Attribute shape 17
3.8 Relationship shape 17
3.9 Identifying Relationship shape 18

5.1 Design of proposed system 26

6.1 Airplane database which is generated by algorithm 43
6.2 Airplane database which is referenced by book 45
6.3 1st entity-relationship diagram example 48
6.4 2nd entity-relationship diagram example 49
6.5 3rd entity-relationship diagram example 50
6.6 4th entity-relationship diagram example 50
6.7 5th entity-relationship diagram example 51
6.8 6th entity-relationship diagram example 51
6.9 7th entity-relationship diagram example 52
6.10 8th entity-relationship diagram example 52

List of Abbreviations

DB	Database
DBMS	Database Management System
ER	Entity-Relationship
ERD	Entity-Relationship Diagram
POS	Part of speech
ML	Machine Learning
NLP	Natural Language Processing
RDBMS	Relational Database Management System

Chapter 1

Introduction

Modern society needs to use database system since they involve many activities that are related to database interaction directly [1].

A database is a set of connected data which has been arranged. The term "data" is used to refer to valid truths that can be documented. A database includes the following characteristics:

1. A database is a representation of anything in the actual world. Database components are used to represent real-world items and relationships.

2. A database is a rational and logical gathering of meaningful data. A database will be unable to appropriately manage a randomly selected batch of data.

3. For a given aim, a database is occurred, developed, and growth of data.

An entity is something which can function alone and may be identified separately. For instance, certain aspects of the real world can be distinguished from the others [2].

In the area of software engineering, commercial information data systems, education, and inquest, entity-relationship charts are commonly applied to design or analyze database systems.

Entity forms (which classify the objects of interest) and connections (that may occur among entities) create a simple Entity-Relationship type (examples of those entity types). ERDs, also known as ER Models, illustrate the inter-connectedness of entities, connections, and their qualities by the usage of a predetermined system of symbols including like rectangles, diamonds, ovals, and connecting lines. They are formed likely to grammar and syntax, with beings acting like nouns and connections acting like verbs [2].

1.1 Application of entity relationship diagrams

- Database design: Creating an entity-relationship diagram is a common first step in defining the needs for an information systems project [3].
- Database diagnosing: ERDs are utilized to investigate current databases for the purpose of find and fix logic and deployment problems; creating the chart must disclose where the problem is occurring.
- Enterprise resource planning (ERP) systems: ERDs are utilized to create or analyze the use of relational databases in commercial processes. It can boost results by streamlining procedures, enabling it much easier to get data, and streamlining processes [2].
- Business procedure re-engineering (BPR): ERDs help in the examination of databases employed in BPR and the modeling of a fresh database structure [2].
- Education: ERDs can be useful in creating databases that store and retrieve relevant information for educational reasons [2].
- Research: ERD diagrams may be quite beneficial in creating useful databases for data analysis [2].

No-code development platforms (NCDPs) become more popular nowadays. Rather than programming, a no-code development platform allows anybody, engineer or

not, to create an application that comprises of graphical interfaces and settings. NCDPs aim to construct or design a project without implementing any code is this kind platform requires every little details.

Designing a database model, or rolling back already constructed database management system, or database troubleshooting may be very costly. Therefore, every little detail has to be processed accurately. At this point, a software manages and handles this precisely.

No-code development platforms may be used to design entity-relationship modeling to reduce higher costly operations that are defined above. Designing no-code development platform for entity-relationship modeling reduces missing or errors.

This study aims to construct no-code development platform for entity-relationship modeling and generate components of any entity-relationship diagram. These components have been defined as entity, attribute and relationship. They are going to be explained with own details at next sections.

The parts of this thesis are located as follows:

In Chapter 2, understanding the domain and state-of-art for ER modeling, more than fifty articles have been read and findings are similar works have been shared. Especially, the most important articles are preferred.

In Chapter 3, scope of database management systems has been studied within general concepts. This part is required to figure out how a database management system works. This chapter contains details of a DBM system.

In Chapter 4, heuristic rules which indicate relationship between human language and database management system have been defined with examples. In other say, this chapter includes detailed information how planned approach has been provided. The rules have been constructed for English language since this study's aim covers English text.

In Chapter 5, design and implementation details have been introduced. Design includes details of proposed system. Implementation also contains details of design.

In Chapter 6, results of experiment have been explained with discussions.

In Chapter 7, objectives of this thesis have been shared.

Chapter 2

Literature Survey

In today's world, mechanized software design and development technologies [4] are becoming increasingly widespread and popular. Computerized software design and development technologies [4] are research and development tools that aid in the construction of a project's foundation based on client needs and specifications. There are many types of this approach. This approach assists us when we are designing a schema or code base. However, they are commonly based on human text. The first goal of this approach is to design project base more accurately. Designing a schema or code base is required domain knowledge since every item in a human text has been parsed.

In this study, a study about representing entity-relationship modelling for English language has being presented. Automated software design and development tools are major study field of Natural Language Processing (NLP). Many resources for both mutual studies and NLP techniques for this approach have been researched, therefore our findings are presented in this section. There is now a rising interest in automating the eradication of data from innate language text, which forms a substantial portion of domain knowledge [5]. Domain knowledge and special rules have been studied on previous sections.

Geetha S. and Anandha Mala G. S. [5] who are from St. Joseph's College of Engineering in India, have aimed to information extraction from natural language text, and hereby they achieved to construct a database schema. Their work

focuses on rule-based approach. They make phrase chunking with using POS tags.

They finally make SVO structure. Therefore, they can understand attributes, objects and relations. They can also understand what attribute primary key is.

There is a similar study for generating UML models. Deva Kumar Deeptimahanti and Ratna Sanyal [6] achieved making semi-automatic generation of UML models from innate language requirements [6]. They specifically mentioned that one of the key causes of such possible issues is the declaration of software needs in Natural Language format [6]. This is the most common problem to extract information from human text. The architecture that has been developed by Deeptimanhanti and Sanyal is bigger than previous one. They use more than one framework at same time.

Sven Hartmann and Sebastian Link [7] explains basic rules that state correspondences between English sentence structures and [8] EER modeling features. They studied the impact of new ER features, such as specialization, generalization, higher-order [9] relationship types and collection types, on the heuristic guidelines.

Circe [10] is a system which is web-based for enabling innate language necessity gathering, elicitation, picking and confirmation.

For constructing an ER diagram from natural language specifications, Meziane [11] utilizes a semi-automatic technique. It takes normal English input and turns it to Logical Form Language (LFL). These logic forms serve as a foundation for recognising entities, properties, and their relationships. Appropriate degrees are assigned to recognized associations using heuristics. This method is largely reliant on quantifiers for determining the degree of relationships. The' and an are two existential quantifiers that are both definitive and indefinite.

N. Omar [12] did also research to collect semantic information from natural language issue statements in order to generate ER types. They demonstrate that

combining semantic lexical information with syntactic heuristics produces much more precise and accurate outcomes. Semantic research aids in the resolution of a broader variety of difficulties, including such anaphoric references and nominalization. Interpreting the outcomes of parsing allows you to add more expressions.

Ronak Dedhia, Atish Jain and Prof. Khushali Deulkar [13] try to find the several ways used by the current tools, for extracting the necessity specification from the explanation of the problem in the language of English.

Claes Wohlin and Aybuke Aurum made a study that to analyse checklist-based look-over for ER-charts [14]. Checklist based inspections are defined heuristically. Someone created the checklist for the research manually [14] before.

Chapter 3

Overview of DBMS & ERDs

The Entity Relationship Diagram (ERD) depicts the real world as a collection of entities, their interrelationships, and the qualities that define them. The entity where we wish to store data is referred to as an entity. The allowable interconnections among occurrences of entities are defined by a relation- ship. A trait shared by all or most occurrences of a certain entity is called an attribute. Because the ER method is simple to grasp, a designer can concentrate on conceptualizing an organization and deciding what entity sets, connection sets, and constraints to utilize.

An Entity-Relationship Model is an important player who has a main role in creating the data modeling structure of business systems. Entity Relation- ship (ER) diagrams have had a main role in systems specification, examination and development [15]. Due to abstraction and high level of conceptual data, ER modeling is an overwhelming task for database designers and system analysts. ER types are for controlling and monitoring system's databases [15]. In every organization for running a business at a large scale, databases are an integral element. Although database systems play critical role in design and development progress, obtaining and briefly explaining entity relationship diagram from requirements may be a lengthy and time consuming.

Newer studies have been targeted on automated extraction of data from human language using Natural Language Processing [15]. NLP is one of the purposes

of Artificial Intelligence. Generally, NLP is used to automatically change data stored in human language to a format which is machine understandable [15]. The core purpose of NLP is to get info from unstructured data [15] to be processed. Therefore, NLP is able to be used to create automation in order to generating ER chart.

In this section we introduce the fundamentals of database with advantages and basic components, overview of entity-relationship diagrams briefly.

3.1 Database Management Systems

Databases are a required elements of life in modern cultures: many of us involve many activities each day, some of them are relevant with database interaction.

A database is commonly an organized set of relevant info. By info, we mean common truths facts which can be recorded. A database has given properties:

1. A database is a representation of anything in the actual world. Database components are used to represent real-world objects and relationships [16].
2. A database is a rationally consistent set of info that has some meaning. A database cannot handle a randomized [17] type of batch of data correctly.
3. For a particular [17] function, a database is created, developed, and populated with data.

Traditional database systems store textual and numeric data. New media technology, on the other hand, has made it feasible to digitally save photographs, audio samples, and video streams. These kinds of files are critical element for only multi- media databases. Geographic information systems, in other say GIS, can store maps, data of weather and images of satellite [18]. It is a kind of multimedia database. There are many types of databases.

Database systems, kind is not important, are basically to store the data for supporting systems. Active database is a real-time data synchronization process, tech is utilized to manage processes of industrial and manufacturing. And database search ways, in other say information retrieval methodologies, are being used to the World Wide Web to enhance the search for data.

The expanding usage of computers has a significant influence on database technology. Databases are essential in practically every field where computers are employed [19]. Commerce, e - commerce, engineering, medical, genetics, law, education, library, and other fields are examples. Even in locations wherein laptops are not utilized, people must keep track of their data on paper.

A database might be of any volume and intricacy [18]. An instance of an important business database is Amazon.com. It includes information for over 20 million books, CDs, videos, DVDs, games, electronics, apparel, and other things. The database inhabits over 2 terabytes and is stored on 200 separately servers. Almost 15 million guests entry Amazon.com each day and use the database to make sells. The database is frequently informed as fresh books and other things are put to the stock and stock conditions are updated as sells are transacted.

A database might be created and sustained manually, or it may be handled electronically. A computerized database might be formed and continued either by a group of application programs written particularly. This study aims to develop a solution for this issue.

A database management system (DBMS) is a collection of data that enables users to create and maintain databases. The database management system (DBMS) is a general-purpose system software that simplifies the operations of creating, building, modifying, and sharing databases across multiple users and applications.

The database description or detailed description is also saved by the DBMS in the form of a database index or vocabulary and is referred to as meta-data.

A design requires requirements formulation and analysis (whether it is a fresh mechanism for a current database or the development of a brand-new database). These criteria are meticulously recorded and turned into a conceptual design known as an entity-relationship diagram. Some technologies may be used to represent and run a conceptual design, allowing it to be easily maintained, amended, and transformed into a database design. The conceptual design is subsequently transformed into a logical design that may be described in a database schema deployed in a commercialized database management system.

Database designers are responsible for deciding what data will be stored in the database and creating acceptable formats to represent and retain it. These activities are often completed prior to the database being deployed and filled with data. Database designers must interact with all potential database users in order to identify the needs and produce a design that satisfies them. Designers are frequently on the DBA's staff and may be allocated additional responsibilities when the database design is accomplished. Database designers often communicate with each conceivable set of users and create database views that match their data and processing needs. Every point of view is then examined and combined with the points of view of other user groups. The final database architecture should be able to meet the needs of all user of the groups.

The DBMS modules and interfaces are designed and implemented as a software package by DBMS system designers and implementers. A DBMS is an extremely complex software system that comprises of numerous components, or modules, such as modules for catalog implementation, query language processing, interface processing, data access and buffering, concurrency control, and data recovery and security. The database management system (DBMS) must communicate with other system software like the operating system and translators for different languages of programming.

3.2 Database Modeling

Database modeling is to construct conceptual design. Conceptual design is required for designing a successful database system. Conceptual design has different ways like entity-relationship diagram or object modeling. Accordingly, conceptual design is to design entity-relationship chart.

This study aims to focus on database modeling, in other say conceptual design, such as entity-relationship diagram.

Entity-relationship (ER) chart may be called entity-relationship modeling. An ER diagram consists of entity, attribute and relationship. They are described as figures on a diagram. Since there are various shapes, they have their own meaning. Entity, attribute and relationship might be represented in various ways such as object modeling.

Object modeling might be represented in various ways such class diagram. It is mostly known that is used in data object classes in software. Object modeling is applied through Unified Modeling Language, in other say UML.

There are many ventures to describe conceptual design. But, this study aims to mostly focus on entity-relationship diagram.

3.3 High-Level Conceptual Design

Conceptual design may consist of data flow, diagram and scenario. Requirements are commonly described in these various schema. ER modeling is a diagram.

The conceptual design path is introduced as follows:

1. Designer or analyst gathers the requirements over written text. Understanding the basics of requirement text is the initial step. All needs are described such as property or object. In ER modeling, a property is called attribute and object is called entity.

2. Designer or analyst starts to understand the limitation of requirement text such as relationship among objects.

3. Designer or analyst makes comparisons between requirement and analysis. Unless it meets, sustaining effort should be. This is a milestone to be sure.

4. Once the requirements have been collected and analyzed, the next step is to create a conceptual design, in other say picking up a appropriate schema. Accordingly, this study focus on entity-relationship diagram.

Collecting requirements is required in initial step. During this step, the person who is going to design the database, gathers overview of requirement text. Requirement text should be as possible as including details. All basics or needs should be understood by any person who reads. Therefore, requirement text should be well defined. All needs are described such as property and object. In ER modeling, an object is described as entity. An entity represents a real world thing [19]. A property is described as attribute. An attribute explains a real world thing, in other say object.

After requirements have been collected and analyzed, the limitations should be determined. A limitation may be a primary key, or multiplicity, or nullable, or data object type. In ER modeling, some limitations such as data object type may be skipped. But, some limitations such as a primary key or multiplicity cannot be bounced. These kinds of constraints have critical role in ER modeling since they include descriptive purpose. A conceptual design in high-level should cover this feature.

A high-level conceptual design is able to be used as a reference to ensure whether requirements are met, or not. Accordingly, database designer can change her focus. Because a high-level conceptual design enables database designers to concentrate on specifying the all pieces. Unless conceptual design meets the requirements, all steps may be repeated so far. This is a milestone to confirmation. Hereby, the conceptual design provides to make accurate database specifications.

When all analysis and confirmations have been completed, a proper conceptual design or schema should be pick up. It varies through preferences, like this study has aimed to focus on the entity-relationship diagram. After high-level conceptual design is transformed from requirement to schema, this phase may ended up. Eventually the constructed schema is called logical design or data model mapping. Therefore, when an entity-relationship diagram is constructed, it refers logical design or data model mapping.

After the database designer gets logical design or data model mapping [20], implementation begins within structured query language. This step is actually the last step that completes physical design and development. It includes internal storage structures, organizations of files, indexes, and access ways and so on.

3.4 Database Components

In this section, basic components of a database are described. A database is made of entities, attributes, and relationships fundamentally.

An entity is a thing that has the ability to exist independently and can be individually recognized. For instance, there are certain characteristics of the actual world that may be identified from others. An attribute is a term used to describe anything. A relationship arises when two or more things interact and defines their relationship.

3.4.1 Entity

The core item that ER modeling portrays is an entity, that is a real-world thing with its own existence. An entity can be a tangible item, such as an individual, vehicle, property, or worker, or it can be an abstract object, such as an instance, a corporation, a job, or a university class. An entity is presented as rectangle in a ER diagram as following:

Figure 3.1: Entity shape

Unless an entity includes unique identifier, in other say primary key, it is called weak entity [21]. A weak entity is presented as nested rectangle in a ER diagram as following:

Figure 3.2: Weak Entity shape

3.4.2 Attribute

A real world object, it is called an entity, may be described with properties. A property is called an attribute among database components.

An entity is made up of at least one attribute. An entity has aspects that define it particularly. For instance, a course entity includes several attributes that define the entity such as title, abstract, prefix, code, department and so on.

An entity is represented through means of its attributes. All aspects should have own attitudes. For instance, when "pupil" abstract object is an entity, it should contain name, class, age and so on.

An entity is presented as oval shape in a ER diagram as following:

Figure 3.3: Attribute shape

An attribute may various named through its role in the context. Here the list:

1. Key attribute

2. Multi-valued attribute

3. Composite attribute

4. Derived attribute

Key attribute is an attribute, but its role indicates to identify the entity uniquely. It is also called primary key. A key attribute is presented as the underlined oval shape in an ER diagram as follows:

Figure 3.4: Key Attribute shape

Multi-valued attribute is an attribute that addresses a feature of real world object. A multi-valued attribute can have many values. A individual, for instance, can have many phone numbers, email addresses, and so on. A list is used to represent a multi-valued attribute. In an ER diagram, a multi-valued attribute is represented by a nested oval shape like follows:

Figure 3.5: Multi-valued Attribute shape

Composite attribute is an attribute that is divided into sub-attributes in conceptual design. It is needed to determine requirements clearly at conceptual design. Composite attributes are formed of many basic traits. For instance, a student's full name may include both his or her first and last names. A composite attribute is presented as interconnected oval shapes in an ER diagram as follows:

Figure 3.6: Composite Attribute shape

A derived attribute is one that does not appear in the actual database but derives its value from other characteristics in the database. As an instance; The average

pay in a sector must not be stored straight in the database, but rather calculated. Another example is that age may be calculated from birth data. In an ER chart, a derived attribute is represented by a dotted oval shape, as seen below:

Figure 3.7: Derived Attribute shape

3.4.3 Relationship

A collection of associations —or a relationship set— among entities from these entity types is defined by a relationship type R among n entity types $E1$, $E2$,..., En. A relationship type and its matching relationship set are commonly referred to by the same term, as is the case with entity types and entity sets R. The relationship set R is mathematically defined as a collection of relationship instances ri, each of which associates n unique entities $(E1, E2, ..., En)$, and each entity Ej in ri is a member of entity set Ej, $1jn$. As a result, a relationship set is a mathematical relationship on $E1, E2, ..., En$; it may also be described as a subset of the Cartesian product of the entity sets $E1, E2, ..., En$. Every one of the entity types $E1, E2, ..., En$ is said to be a member of the relationship type R; likewise, each of the individual entities $E1, E2, ..., En$ is said to be a member of the relationship instance $ri = (E1, E2, ..., En)$.

A relationship is presented as diamond shape in an ER diagram as follows:

Figure 3.8: Relationship shape

A relationship may include own attribute. It is called identifying relationship. A identifying relationship is presented as nested diamond shape in an ER diagram as follows:

Figure 3.9: Identifying Relationship shape

3.5 Proper Naming of Schema Constructs

The selection of labels for entity types, characteristics, relation types, and (especially) roles while building a database structure is not necessarily clear. Names should be chosen in such a way that they express as much as feasible the meanings associated with the various components in the database. We prefer solitary names for entity types over plural names since the entity name pertains to each unique entity that belongs to that entity type. In our ER charts, entity type and type of relationship identities will be capital letters, attribute names will have their initial letter uppercase, and position names will be lowercase letters.

In principle, provided a narrative description of the database needs, nouns in the narrative usually give birth to entity type names, while verbs tend to imply type of relationship names. Attribute names are often derived from nouns which characterize the nouns that belong to entity kinds.

Chapter 4

Approach

In this study, a traditional natural language application approach that generates entity-relationship diagram from human languages is presented. Database fundamentals have been studied and entity-relationship diagram has been investigated detailed so far.

There is a connection between human natural languages and conceptual database design [22] [23]. Before starting design, heuristic rules that explain the connection have been determined clearly. A heuristic is a rule that allows you to solve complex issues faster. When you just have a limited amount of time and/or data to make a choice, heuristics come in handy. The majority of the time, heuristics will guide you to a decent conclusion.

Some heuristic rules have been defined. Heuristic rules are not certain cases, they are just well-defined assumptions. Our heuristic rules are giving in below.

4.1 Rule 1: Identify Entities

4.1.1 A common noun may indicate an entity type.

A noun like "record," "database," "company," "system," "information," or "organization" may not be a good choice for an entity set. As an instance,

The term "company" may refer to the business setting and must not be listed among the entity categories [24].

For example:

- "An insurance company wishes to create a database to keep track of its operations."
- "An organization purchases items from a number of suppliers."

4.1.2 A proper noun may indicate an entity.

A proper noun such as "course", "student", "employee" and "customer" may indicate an entity. They are applicable for both real world objects and object-oriented programming [15].

For example:

- "Student takes course."
- "Employees work for a company."

4.1.3 In case of consecutive nouns existence, check the last noun. It may be an entity type, otherwise it may indicate an attribute.

It may be an entity type, if in case of consecutive nouns exists. Otherwise it may indicate an attribute.

For example:

- Incoming salary should be recorded for each employee.

4.1.4 A gerund may indicate an entity.

A gerund may indicate an entity.

For example:

- Exchanging students should be registered by their passports.

4.1.5 Ignore every proper noun.

Every proper noun must be ignored since they are data.

For example:

- Computer science department has a curriculum, number of students.

4.2 Rule 2: Identify Attributes

Attributes are nouns which are stated with their entity; it may be followed by verbs including such "have, contains, or includes" to imply that an entity is ascribed with a feature. This study will make use of a corpus.

For instance, in "course has prefix, code, title and description", course is detected as an entity and prefix, code, title and description are detected as attributes.

Here there are some rules that detect attributes in specifications.

4.2.1 Noun phrase with genitive case may indicate an attribute.

A word or phrase in the genitive form may refer to an attribute. A noun phrase is made up of a noun or pronoun as the subject and any dependaet terms preceding or following the subject. Dependent words provide data about the head.

For example:

- Each student should register into departmental elective courses.
- Each student should register into complementary elective courses.

4.2.2 The possessive case usually shows ownership it may indicate attribute type.

The possessive form normally indicates possession, although it can also signify attribute type. To express that something pertains to something or someone, we employ apostrophe s ('s), commonly known as possessive's

4.2.3 A noun phrase such as has/have may indicate attribute.

A noun phrase such as has/have may indicate attribute. "Has/have" show ownership.

For example:

- A student has first name, last name and e-mail address.

If a noun is preceded by some other noun, and the latter comes to a set, it is possible that the both nouns are attributes; otherwise, it is possible that the both nouns are entities. A noun such as "course code", "department name" "belonging faculty" and "semester type" refer to an attribute.

4.3 Rule 3: Identify Relationships

A connection is more probable to be the major verb that happens among two things. Two entities can be differentiated by the main verb alone, by the main verb and an auxiliary verb, or by the main verb and a modal verb. For instance, if a bank has several branches, the branching is identified as a connection.

4.3.1 A transitive verb can indicate relationship type.

A transitive verb can indicate relationship. A transitive verb is a verb that accepts one or more objects.

For example.

- Student enrolls at least one course per a semester.
- Department announces at least three courses per a semester.

4.3.2 If a verb is in the current list: include, involve, comprises of, encompass, contain, split to, embrace, this suggests an aggregation or compositional relationship.

If a verb appears in the preceding list, it indicates an aggregate or composition connection. We are creating a corpus using this feature. Include, involve, consist of, include, contain, split to, embrace are all words that can be used to describe a corpus. This heuristic rule is akin to the "a noun phrase so that has/have may indicate attribute.

For example:

- A student object consists of first name, last name and e-mail address.

There are also two assumptions as following:

- An adverb can denote a connection trait.
- A verb preceded by a preposition including such "by", "to", "on", "or", "in" might indicate the sort of relationship.

4.3.3 Passive voice can be translated into active voice.

If a sentence includes *am/is/are* and third form verb, it is a passive voice sentence. If a sentence contains just verb, it is an active voice sentence. A passive voice sentence can be translated into active voice.

For example:

- A course is taken by a student (passive voice).
- A student takes a course (active voice).

4.4 Rule 4: Identify Primary Key

4.4.1 Adverb indicates primary key of an entity.

An entity includes at least one attribute. Attributes are to used to determine the entity. Primary key is an attribute. Primary key explains the entity. A uniqueness attribute is a primary key. Uniquely adverb indicates primary key of an entity.

Chapter 5

Design & Implementation Details

In this section, design and implementation have been introduced in detail. Design is proposed system and implementation is a pipeline. Heuristic rules have been defined before in Chapter 4.

It is known that natural language processing, in shortly NLP, includes all functionality about human language. There are various functions in NLP field.

Software is based on defined rules. These rules may be very small pieces for a little process. Within scope of this study, rules have been defined. Since they do not cover all cases, they are called heuristic rules. But, these heuristic rules are just text. They require computational statements. Therefore, a text should be investigated at a deep level. To make possible this investigation for each word, NLP functions should be used.

The proposed system, in other say design, represents what function is going to be in use. Following figure represents it.

Accordingly, it is shown there are seven features in the design. These seven features are applicable own self. But, to achieve to objectives of this thesis, they should be used at same pipeline. Since this idea, implementation should be end-to-end system.

The seven features are described with own detail at next section.

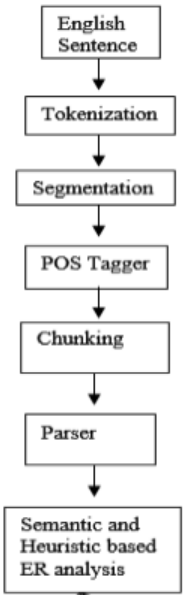


Figure 5.1: Design of proposed system

5.1 Proposed Design

Heuristic rules have been introduced in Chapter 4. They lead development to how the implementation is going to do. Since they have critical roles, heuristic rules should be well defined. The design should be constructed as end-to-end system, and provide to process any item.

The design includes seven features. But, one is an input layer and the others are NLP functions. Here the list:

1. Segmentation
2. Tokenization
3. POS tagging
4. Chunking
5. Parser
6. ER analysis

They are mostly implemented in NLP tools such as NLTK and Spacy. They are used directly and indirectly for common tasks.

A sample text is going to be used describing any phases.

Sample text is following:

- Musicians take many courses. Each musician has unique number, a name, multiple addresses. Musician may have phone numbers. Each song recorded at Music Company has a unique title and an author.

5.1.1 Segmentation

A function that splits an entire text into single sentences is known as segmentation. A sentence is a full set of words which has a subject and predicate, conveys a declaration, inquiry, exclamation, or order, and consists of a major clause and possibly one or more subordinate clauses.

Third party tool is used, Spacy, since it enables a capability introduced above.

Sample text is divided into smaller items as following:

- Musicians take many courses.
- Each musician has unique number, a name, multiple addresses.
- Musician may have phone numbers.
- Each song recorded at Music Company has a unique title and an author.

5.1.2 Tokenization

Tokenization is a manner of differentiating a piece, or a sentence, of text into fewer units that are called tokens [25]. Tokens can be all words, characters, sub words even whitespaces and punctuations. Hence, tokenization can be classified

into 3 types – word, character, and sub word tokenization [25]. Third party tool has been used since it enables tokenization feature by itself.

While Spacy is performing this capability, it uses BERT language model. BERT, in other say Bidirectional Encoder Representations from Transformers, is a transformer-relied machine learning technique for NLP pre-training developed by Google [26].

Smaller pieces of sample text are tokenized as following:

- "Musicians", "take", "many", "courses", "."
- "Each", "musician", "has", "unique", "number", ",", "a", "name", ",", "multiple", "addresses", "."
- "Musician", "may", "have", "phone", "numbers", "."
- "Each", "song", "recorded", "at", "Music", "Company", "has", "a", "unique", "title", "and", "an", "author", "."

5.1.3 POS Tagger

Definition of Part of Speech Tagging

Part-of-speech tagging (POS tagging or PoS tagging or POST), also known as grammatical tagging in corpus linguistics, is the technique of labeling a word in a text (corpus) as relating to a certain part of speech based on both its definition and its context. A simplified version of this is typically given to school-age children, in which words are identified as nouns, verbs, adjectives, adverbs, and so on.

Tagging is a kind of classification that may be defined as the automatic assignment of description to the tokens.

Following figure is a example of POS tagging.

Word	Part-of Speech Tag	Dept
Musicians	noun	nsubj
take	verb	root
many	adj	amod
course	noun	dobj
.	punct	punct

Table 5.1: An example sentence for part-of-speech tagging.

In this phase, third party tool Spacy has been preferred since it provides more attributes than NLTK. Eight attributes are provided by Spacy and given in following table [27].

Text	**Lemma**	**Pos**	**Tag**	**Dep**	**Shape**	**Alpha**	**Stop**
Apple	apple	PROPN	NNP	nsubj	Xxxxx	True	False
is	be	AUX	VBZ	aux	xx	True	True
looking	look	VERB	VBG	ROOT	xxxx	True	False
at	at	ADP	IN	prep	xx	True	True
buying	buy	VERB	VBG	pcomp	xxxx	True	False
U.K.	u.k.	PROPN	NNP	compound	X.X.	False	False
startup	startup	NOUN	NN	dobj	xxxx	True	False
for	for	ADP	IN	prep	xxx	True	True
$	$	SYM	$	quantmod	$	False	False
1	1	NUM	CD	compound	d	False	False
billion	billion	NUM	CD	pobj	xxxx	True	False

Table 5.2: An example for spacy tagging.

These eight attributes are described in spacy web site as following:

1. **Text:** The original word text [27].
2. **Lemma:** The base form of the word [27].
3. **POS:** The simple UPOS part-of-speech tag [27].
4. **Tag:** The detailed part-of-speech tag [27].
5. **Dep:** Syntactic dependency, i.e. the relation between tokens [27].
6. **Shape:** The word shape – capitalization, punctuation, digits [27].
7. **is alpha:** Is the token an alpha character [27]?

8. **is stop:** Is the token part of a stop list, i.e. the most common words of the language [27]?

Dependency labels explain relation between token and following table includes dependency labels with descriptions. We mostly use *dep* in chunking feature.

Number	**DEP**	**Description**
1.	ACL	Clausal modifier of noun [28]
2.	ACOMP	Adjectival complement [28]
3.	ADVCL	Adverbial clause modifier [29]
4.	ADVMOD	Adverbial modifier [29]
5.	AGENT	Agent [29]
6.	AMOD	Adjectival modifier [29]
7.	APPOS	Appositional modifier [29]
8.	ATTR	Attribute
9.	AUX	Auxiliary
10.	AUXPASS	Auxiliary (passive)
11.	CASE	Case marker
12.	CC	Coordinating conjunction
13.	CCOMP	Clausal complement
14.	COMPOUND	Compound modifier
15.	CONJ	Conjunct
16.	CSUBJ	Clausal subject
17.	CSUBJPASS	Clausal subject (passive)
18.	DATIVE	Dative
19.	DEP	Unclassified dependent
20.	DET	Determiner
21.	DOBJ	Direct Object
22.	EXPL	Expletive
23.	INTJ	Interjection

Table 5.3: Dependency labels with descriptions between 1-23rd items

Number	DEP	Description
24.	MARK	Marker
25.	META	Meta modifier
26.	NEG	Negation modifier
27.	NOUNMOD	Modifier of nominal
28.	NPMOD	Noun phrase as adverbial modifier
29.	NSUBJ	Nominal subject
30.	NSUBJPASS	Nominal subject (passive)
31.	NUMMOD	Number modifier
32.	OPRD	Object predicate
33.	PARATAXIS	Parataxis
34.	PCOMP	Complement of preposition
35.	POBJ	Object of preposition
36.	POSS	Possession modifier
37.	PRECONJ	Pre-correlative conjunction
38.	PREDET	Pre-determiner
39.	PREP	Prepositional modifier
40.	PRT	Particle
41.	PUNCT	Punctuation
42.	QUANTMOD	Modifier of quantifier
43.	RELCL	Relative clause modifier
44.	ROOT	Root
45.	XCOMP	Open clausal complement

Table 5.4: Dependency labels with descriptions between 24-45th items

5.1.4 Chunking

A full-text has been taken as an input, it has been divided into separate sentences in segmentation feature, then each individual sentence is tokenized in tokenization feature, then POS tagging feature run for each separate sentence. So far, three features of six has been launched. Chunking is the forth feature in queue. Chunking is a method that makes decisions.

In this feature, a sentence object has been created. Sentence object includes subject, verb, object, (possible) primary keys and (possible) multiplicities. Subject, verb and object (SVO) attributes are required for each sentence, but primary keys and multiplicities are not essential.

Subject

Dependency labels for subject is pre-defined as a list. This list includes "nsubj " and "nsubjpass" labels. They were introduced previous table. This list is used to compare to each token in a sentence. If this comparison is applicable, subject item is obtained.

Verb

While verb item in a sentence is obtained, each token in a sentence is controlled. If part of speech is verb and dependency label is root in currently, this token is obtained as verb item.

Object

While object item in a sentence is obtained, part of speech tag is looked for. POS tag has to be noun. Some nouns are consists of two or three words. We use dependency label, while we are making these words is a group. If dependency label is compound, these can become a group.

Primary Key

If part of speech tag is adjective and text is unique currently, it is defined as primary key item.

Multiplicity

If part of speech tag is adjective and text is many currently, it is defined as multiplicity item.

Following table presents an example for chunking.

Subject	Verb	Object	Primary Key	Multiplicity
Musicians	take	courses	n/a	musicians, courses

Table 5.5: An example sentence for chunking.

5.1.5 Parser

Before entity-relationship diagram is created, parser runs. Parser is a actually final-state machine which makes rule-based decisions. Parser feature uses our heuristics we defined before in Chapter 3. NLP has multiple possible analysis due to its ambiguous grammar. Parsing determines parse tree of a given sentence where in a group of words is transformed into structures. This may be inapplicable for our heuristics.

Understanding verb

Parser is focused on verb first. It behaves differently according to verb is special, or not. Some verbs are predefined as a list before. This list includes "have/has, contain, include, consists of" and so on.

Incoming verb is a special verb, in other say if it is in the predefined list, object is going to be attribute case. Otherwise, object is going to be relation case.

Verb indicates object is attribute

If verb is in the predefined list, object is attribute. In this behaviour, we have to focus on: Subject might be created before as an entity, or not.

If subject is created before as an entity, new attribute(s) must be created and added to existing entity. Unless subject is created before as an entity, entity must be created first, then attributes are added to this entity.

While new attributes are creating, multi-valued case are controlled. Subject is important in this case since it might be singular or plural.

Verb indicates a relationship

Unless verb is in the predefined list, verb indicates a relationship between subject and object. Subject and object are called entity in this behaviour.

Form of each item in SVO structure should be investigated. In this behaviour, third party tool nltk are utilized. nltk has two operations to convert a word

into singular form: SnowballStemmer and WordNetLemmatizer networks. These methods are called helper methods.

After helper methods return singular form, before-after states are controlled for both subject and object items. Hereby, multiplicity can be determined. For example, if subject changes and object does not change at method calls, multiplicity is captured N-to-1.

Following table presents an example for chunking.

Subject	Object	Verb	Singularity	Primary Key	Multiplicity
Musician	take	course	false	not exists	n-n

Table 5.6: An example sentence for parser.

5.1.6 ER Analysis

ER analysis the last feature in the queue, and it is a method which creates entity-relationship diagram. It uses captured lists for entities, attributes and relations. System also exports these lists as xml output. An entity-relationship diagram is drawn as traditional ways with different shapes. But, it can be also drawn in modern ways with a rectangle that includes entity name and attributes.

5.2 Implementation Details

5.2.1 System Architecture

We implement the proposed design according to the object-oriented programming approach. Four packages are involved to the project. Object package is consists of four classes. These classes are customized as our data structure. Others are settings, app and ERD respectively.

Settings stores all configurations and base information. App presents a console application to use. ERD is the most important part, since system architecture is

implemented here. run method manages all run time. setup, analyzer and fsm methods are called respectively. analyzer method is to used for chunking and fsm method is to used for parser.

5.2.2 Logging & Exception Handling

Logging

Logging is a important piece of any system. We use logging mechanism with on/off attribute. When logging is required, this attribute can be enabled. Otherwise, it can be disabled. We use logging tool in Python. This module offers methods and classes for a configurable incident logging system [30] for programs and libraries. This module also includes different log levels such as debug, info, error and so on. We logs both trace and failure cases. app file with log extension is created at first run, logs are saved to this file. This capability provides easy debug.

Exception Handling

Exception Handling is an important piece of any system. We use try-except-finally blocks in implementation. So that, we try to prevent crashes. While except case occurs, it is logged.

5.2.3 User-interface

Our implementation contains a console application. It is used by entering a choice, then it switches. Finally entities, attributes and relations are captured, they are exported as xml. Another user interface can be designed.

5.2.4 Trade-offs

In this section, we are going to explain trade-offs in the project.

- At the beginning, we used nltk for part-of-speech tagging. Some nouns are consists of two or three words. nltk does not provide to understand these nouns. Because of this, we preferred to use spacy instead of nltk. Spacy enables to understand these nouns.
- Since this approach is based on heuristic rules, it does not always process complex sentences. Hereby, we prefer more easier sentences.
- Since SVO structure is required, every sentence has to have this structure.
- Passive sentence is not processed straight-forward, so that while sentences are typing, these must be active sentence.

Chapter 6

Discussion & Results

In this chapter, the basic reasons of unexpected results of this study which does not cover our expectations have been discussed. Why unexpected results has occurred has been explained. Also, we give example that cause bad results for entity-relationship analysis from human text.

6.1 Discussion

After research has been completed, two popular Natural Language Processing tools that are NLTK and Spacy when we are developing have been compared. As a result, NLTK is more popular than Spacy. However, Spacy provides more feature in machine learning field and using Spacy instead of NLTK was obligation. Because some nouns are not one word. (For example, "phone number" or "computer science department".) To understand this kind nouns, a connection between two or three words is required. NLTK provides one attribute while part-of-speech tag feature is performing. Whereas, Spacy provides eight attributes while part-of-speech tag feature is performing. One attribute, is named dependency label, provides connection between two or three words.

Relationships may have own attributes. For example, a student takes three courses per a year. In this example, it is seen that "student is able to take course" as relationship. Also, "taking three course per a year" describes relation-

ship attribute. It is not possible to store "per a year" information in this heuristic approach since human language is more complex than a computer is able to understand. Defining new heuristics may not be a solution for this problem. For example, NLTK has approximately 40 thousands structure in itself. Therefore, implementing a rule for all sentences is impossible.

Since same reasons are introduced above, taking more accurate results is impossible in every time. Because sentences change through itself. Fixed patterns really does not exist. So, understanding human language is not accurate clearly. It is actually very common to extract information from sentences partly. But, we try to extract all information from entire sentence. This becomes to make our job hard. So, we should prefer more basic sentences, not complex ones. For example, "a department has unique identification number, short name, long name, prefix, department head and phone number." In this example, we see a basic sentence, and heuristic can process every information. These are captured as attributes. For example, "a student takes many courses". In this example, subject and object are captured as entities and this is example for relation representation. Another example is, "Library Service is like I¸sık University Library Service. This service has two libraries, Maslak Library, and Sile Library, where Sile Library is indicated as main." This is very difficult sentence for a computer is able to understand.

Understanding passive voice sentences is harder than active voice sentences. Because words are located in different places. In this way, passive sentence is not processed straight-forward. Before the software is run, sentences are converted from passive voice to active voice. Dependency label attribute of Spacy has passive elements for subject. But, objects may not described in passive. Similar studies do not have any information converting sentences from passive to active voice. Just using active sentences may be a strict rule. There are some methods to understand which sentences are passive, or not. It is also studied to understand active and passive sentences.

When it divides sentences into active and passive voices, dependency labels are

needed. If there are root and aux dependency labels, sentence is passive voice. Otherwise, sentence is active. For example, "a course is taken by a student". In this example, "is" word has "aux" dependency label and "taken" word has "root" dependency label. Therefore, this is a passive voice sentence. For example, "a student takes courses". In this example, "takes" has just "root" dependency label. Therefore, this is an active voice sentence. It is possible to understand whether sentence is active or not, since the correlation we explained above. After which sentence is passive has been understood, it is converted previous and next parts of verb. However, this is not appropriate to planned.

6.2 Results

6.2.1 Case Studies

As a result, actual and expected results have been shared in this section. Actual results are made by this heuristic approach for two data sets and expected results are made by a human. Hereby, differences between actual and expected results have been examined. Cases include human text, heuristic approach (actual result), human investigation (expected result) and comparison (differences between actual result and expected result).

Case 1

Musicians take many courses. Musician has phone numbers. Each song recorded at Music Company has a unique title and an author.

Heuristic approach:

- "Musician" and "song" are captured as entity.
- "Musician" entity has one attribute which is called "phone number".
- "Phone number" attribute is captured as multi-valued.

- "Song" entity has three attributes which are called "music company", "title" and "author" respectively.
- "Title" attribute is captured as primary key.
- There is only one relation which is between "musician" and "course".
- Multiplicity is captured as "N" for both "musician" and "course".
- Action of this relation is to "take".

Human investigation:

- There three entities: "musician", "song" and "course".
- There is an action-based relation which is to "take".
- "Musician" entity has only one attribute which is "phone number" and it is defined as multi-valued.
- "Song" entity has two attributes unique "title" and "author".
- "Title" is defined as primary key.
- Some database designer may define "music company" is an attribute. It is open to comment.
- "Course" should be defined as an entity that do not include any attribute.

Comparison:

- "Course" is not defined as an entity in heuristic approach.
- "Music company" is defined as an attribute in heuristic approach. It might be an attribute, or not.

Case 2

A university contains many faculties. Faculty has unique identification number and name. Each department belongs to a faculty. A department has identification number, name, head, phone numbers. Many students register into program. A course includes prefix, unique identification number, title and description. Each department opens courses in a semester. And students take courses.

Heuristic approach:

- "University", "faculty", "department" and "course" are captured as entity.
- "University" entity includes multi-valued attribute "faculty".
- "Faculty" entity includes primary key attribute "identification number" and "name".
- "Department" entity includes "identification number", "name", "head" and multi-valued attribute "phone number".
- "Course" entity includes "prefix", primary key attribute "identification number", "title" and "description".
- There are four captured relations.
- There is 1-1 relation between "department" and "faculty" based on "belong" action.
- There is N-N relation between "student" and "program" based on "regist" action. It should have defined as "register", but stemming method captures it as "regist".
- There is 1-N relation between "department" and "course" based on "open" action.
- There is N-N relation between "student" and "course" based on "take" action.

Human investigation:

- There are certain four entities: “university”, “faculty”, “department” and “course”.
- “Program” and “student” should be defined as entity that do not include any attribute.
- “University” entity includes multi-valued attribute “faculty”.
- “Faculty” entity includes primary key as “identification number” and “name”.
- “Department” entity includes primary key as “identification number”, “name”, “head” and multi-valued attribute “phone number”.
- “Course” entity includes primary key as “identification number”, “prefix”, “title” and “description”.
- “Student” and “program” should be defined as entity. According to the text, they are empty.
- There are four relations. These are 1-1 between “department” and “faculty”; and 1-N “department” and “course; and N-N between “student” and “program”; and N-N between “student” and “course”.
- Actions to be performed are “belong”, “open”, “register” and “take” respectively.

Comparison:

- “Student” and “program” are not defined as entity.
- “Department” has “identification number”. It should be captured as primary key. However, heuristic approach could not figure out since there is no unique adjective.
- “Regist” is not correct word, it should have defined as “register”. Stemming method could not catch at right form.

6.2.2 Differences Between Actual And Experimental Results

In this section, actual results which are referenced from book (Fundamentals of Database Systems: 8th edition by Elmasri) at university level and experimental results (rule-based entity-relationship diagram modelling algorithm) have been shared.

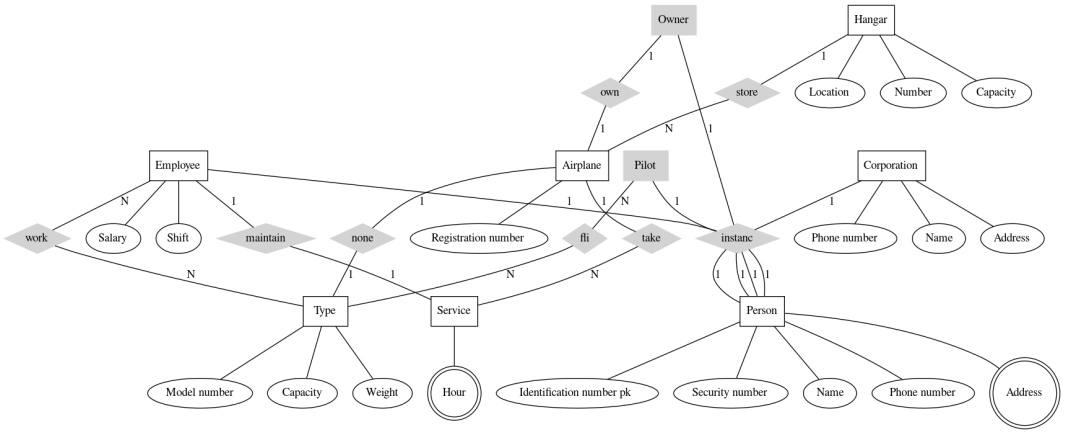


Figure 6.1: Airplane database which is generated by algorithm

Rule-based entity-relationship diagram modelling algorithm catches given entities as below:

Entity
Employee
Type
Airplane
Hangar
Service
Owner
Corporation
Person
Pilot

Table 6.1: Entities are caught by rule-based entity-relationship diagram modelling algorithm

Rule-based entity-relationship diagram modelling algorithm catches given attributes as below:

Entity	**Attribute**
Employee	Salary
Employee	Shift
Type	Model number
Type	Capacity
Type	Weight
Service	Hour
Person	Identification number
Person	Security number
Person	Name
Person	Phone number
Person	Address
Airplane	Registration number
Hangar	Location
Hangar	Number
Hangar	Capacity
Corporation	Phone number
Corporation	Name
Corporation	Address
Owner	No attribute
Pilot	No attribute

Table 6.2: Attributes are caught by rule-based entity-relationship diagram modelling algorithm

Rule-based entity-relationship diagram modelling algorithm catches given relationships as below:

Entity-1	**Relation**	**Entity-2**
Employee	work	Type
Employee	maintain	Service
Owner	own	Airplane
Pilot	fly	Type
Corporation	instance	Person
Pilot	instance	Person
Owner	instance	Person
Employee	instance	Person
Hangar	store	Airplane

Table 6.3: Relationships are caught by rule-based entity-relationship diagram modelling algorithm

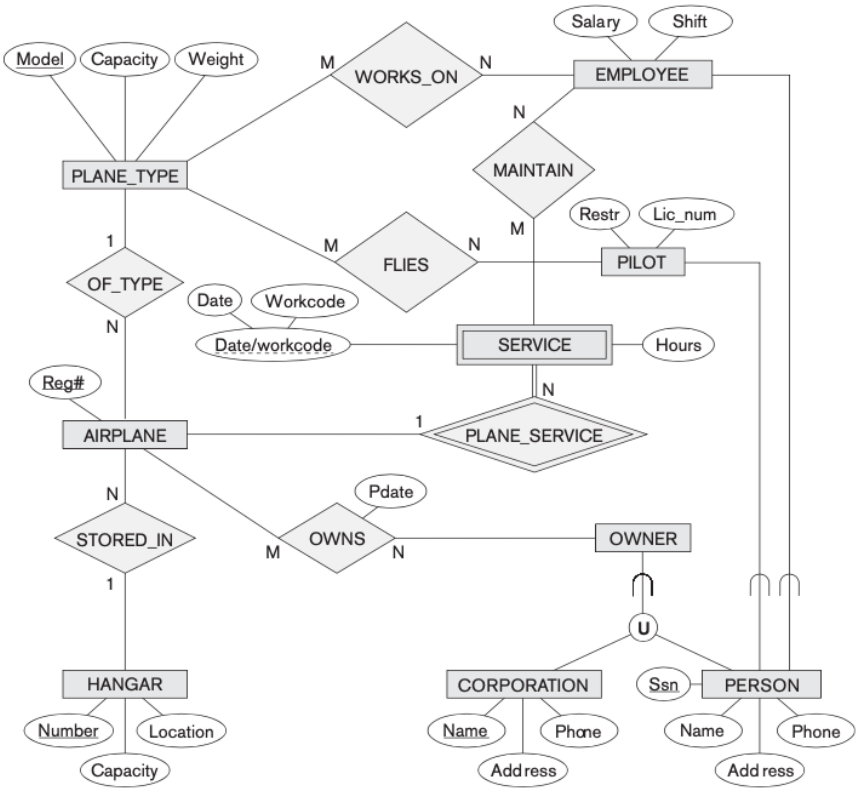


Figure 6.2: Airplane database which is referenced by book

Actual entities from reference book are given as below:

Entity
Employee
Planet type
Airplane
Hangar
Service
Owner
Corporation
Person
Pilot

Table 6.4: Actual entities from reference book

Actual attributes from reference book are given as below:

Entity	**Attribute**
Employee	Salary
Employee	Shift
Planet type	Model
Planet type	Capacity
Planet type	Weight
Pilot	Restr
Pilot	Lic num
Service	Hours
Service	Date (Date + Workcode)
Airplane	Reg
Hangar	Number
Hangar	Capacity
Hangar	Location
Corporation	Name
Corporation	Address
Corporation	Phone
Person	Ssn
Person	Name
Person	Address
Person	Phone

Table 6.5: Actual attributes from reference book

Actual relationships from reference book are given as below:

Entity-1	Relation	Entity-2
Employee	work on	Plane Type
Employee	maintain	Service
Owner	own	Airplane
Pilot	fly	Type
Hangar	store in	Airplane
Airplane	plane service	Service
Plane	type of	Airplane

Table 6.6: Actual relationships from reference book

Rule-based entity-relationship diagram modelling algorithm does not cover union cases. In actual diagram, there exists several union of structure. These are given below:

- Owner union of Corporation and Person
- Person union of Employee and Pilot

Numeric comparisons are given as following:

Actual	Experimental
9 entities	9 entities
20 attributes	18 attributes
7 relations	9 relations

Table 6.7: Numeric comparisons

Numeric analysis is given as following (i.e entity (attribute count) format):

Actual	Experimental
Employee (2)	Employee (2)
Plane Type (3)	Type (3)
Service (2)	Service (1)
Airplane (1)	Airplane (1)
Hangar (3)	Hangar (3)
Corporation (3)	Corporation (3)
Person (4)	Person (5)

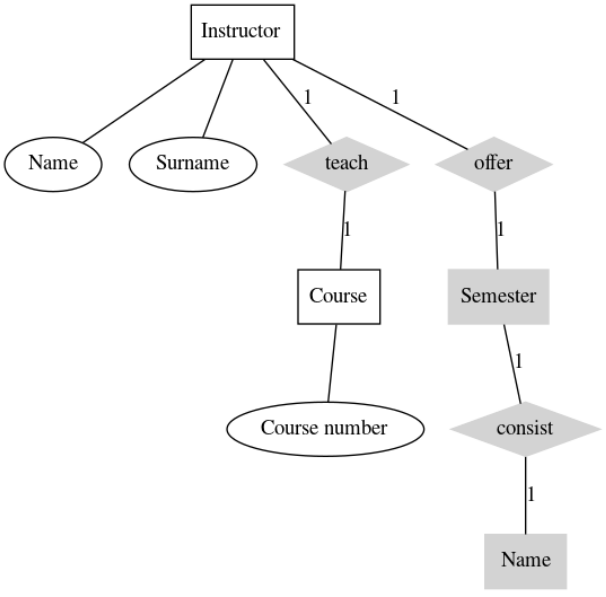
Table 6.8: Numeric analysis

Notes:

- Planet type is caught as type by rule-based entity-relationship diagram modelling algorithm.
- Date attribute is missed in rule-based entity-relationship diagram modelling algorithm.
- Reg from actual result is caught as registration number by rule-based entity-relationship diagram modelling algorithm.
- One additional field is used at experiment.

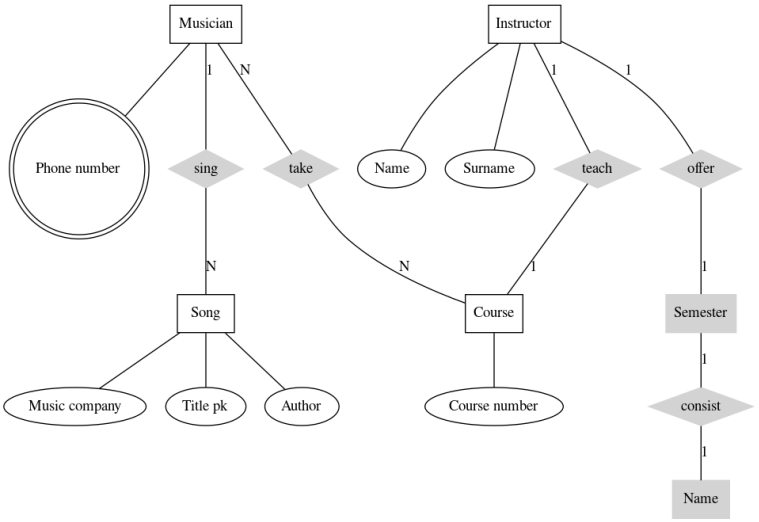
6.2.3 More Experimental Results

In this section, more experimental results have been shared. After subject-verb-object parser algorithm has been executed and database components have been caught up, entity-relationship diagram is created.



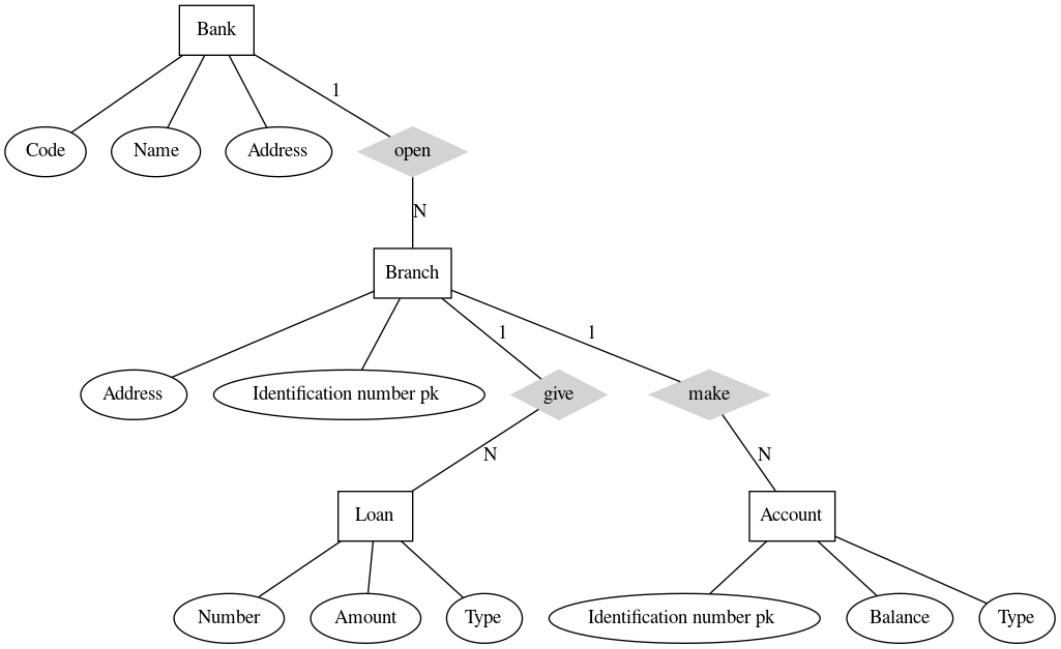
Instructor has name, surname.
An instructor teaches course.
A course has a course number.
Instructor offers a semester.
A semester consists of name, year.

Figure 6.3: 1st entity-relationship diagram example



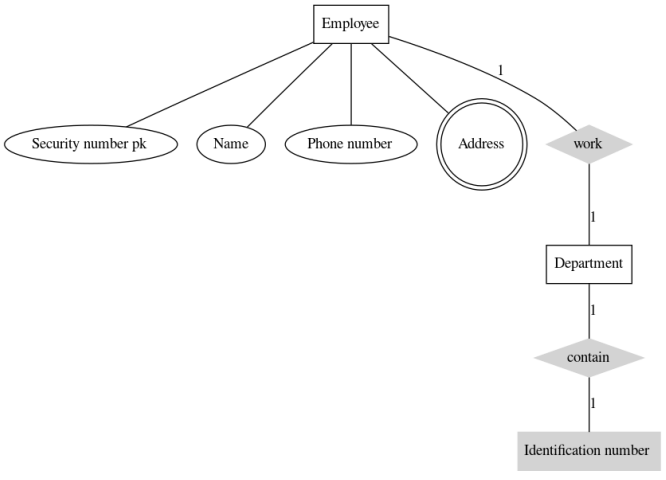
Instructor has name, surname.
An instructor teaches course.
A course has a course number.
Instructor offers a semester.
A semester consists of name, year.

Figure 6.4: 2nd entity-relationship diagram example



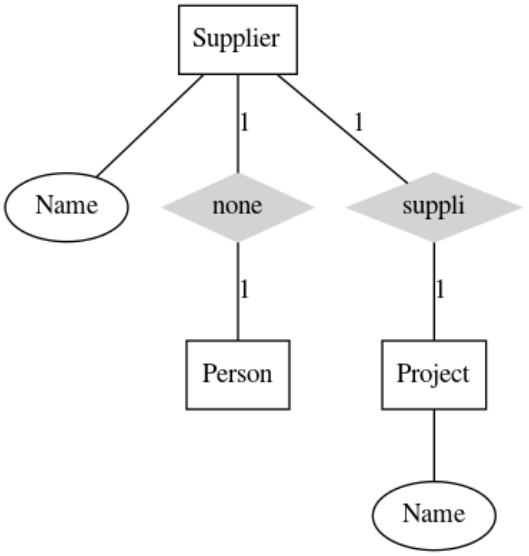
A bank includes code, name and address.
A bank opens branches.
A branch has address and unique identification number.
A loan includes number, amount, type.
A branch gives many loans.
An account has unique identification number, balance, type.
A branch makes many accounts.

Figure 6.5: 3rd entity-relationship diagram example



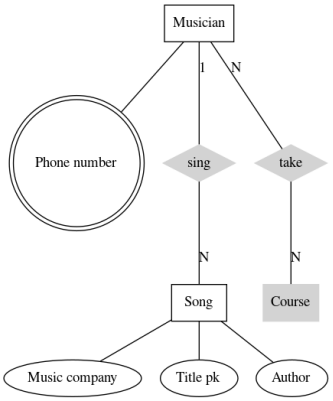
An employee has unique social security number, name, phone number, many addresses.
An employee works in a department.
Department contains unique identification number, code, name, phone number, head and e-mail address.

Figure 6.6: 4th entity-relationship diagram example



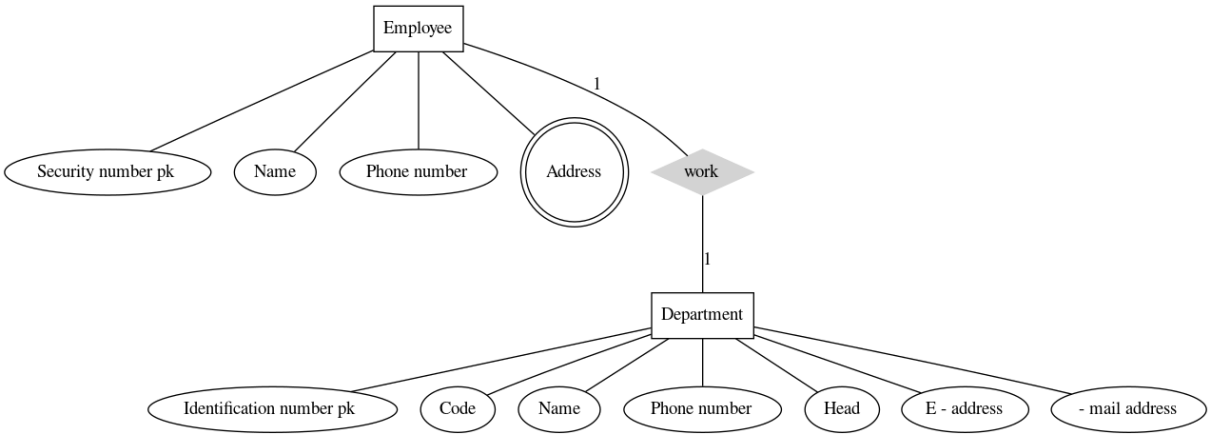
Supplier has full name.
Supplier is a person.
Supplier supplies a project.
A project must have a name.

Figure 6.7: 5th entity-relationship diagram example



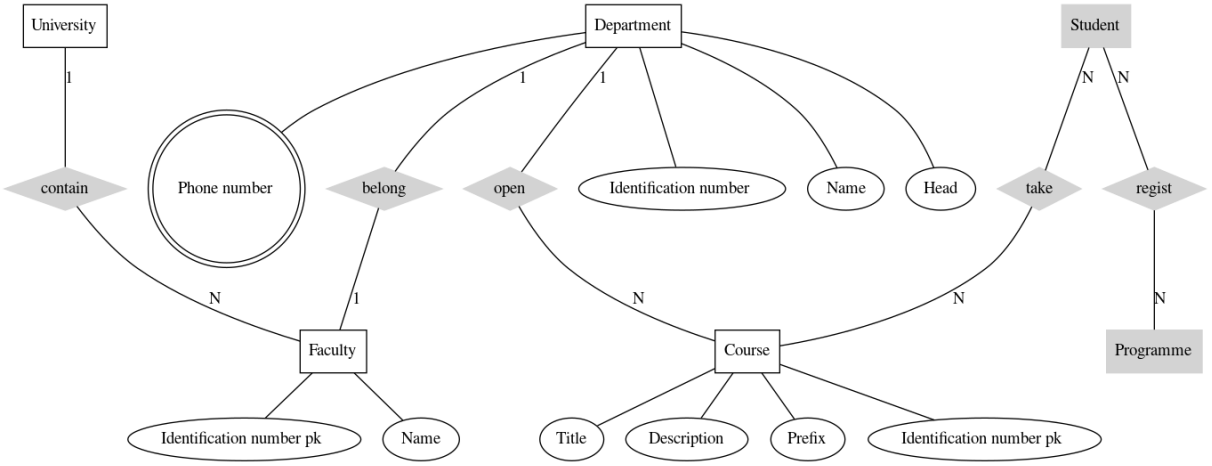
A musician sings many songs.
Musicians take many course.
Each red musician has unique number, a name, an addresses.
Musician has phone numbers.
Each song recorded at Music Company has a unique title and an author.

Figure 6.8: 6th entity-relationship diagram example



An employee has unique social security number, name, phone number, many addresses.
An employee works in a department.
A department has unique identification number, code, name, phone number, head and e-mail address.

Figure 6.9: 7th entity-relationship diagram example



A university contains many faculty.
Faculty has unique identification number and name.
Each department belongs to a faculty.
A department has identification number, name, head, phone numbers.
Many students register into programme.
A course includes prefix, unique identification number, title and description.
Each department opens courses in a semester.
And, students take courses.

Figure 6.10: 8th entity-relationship diagram example

Chapter 7

Conclusion

This work eventually aims to make an entity-relationship diagram from human text. Development path is divided into three parts: understanding relationship between human language and database management systems (chapter 3), defining heuristic rules (chapter 4) and implementing final state machine (chapter 5).

The purpose of these heuristics is to extract SVO structure from a sentence. Parsing sentences for both active and passive voices is a big part of defining heuristic rules.

This heuristic approach may not be applicable for database conceptual design since human language is a huge domain. However, limited heuristic rules have been learned. Deep learning techniques or neural networks may be used in conceptual database design as future works.

References

[1] "Fish species," http://http://fish-species.org.uk/, accessed: 2014.

[2] "What is an entity diagram (erd)?" https://medium.com/@soni.dumitru/what-is-an-entity-relationship-diagram-erd-13daee5b2a/, accessed: 2020.

[3] a. Btoush, EmanS., "Generating er diagrams from requirement specifications based on natural language processing," *International Journal of Database Theory Application*, 2015.

[4] S. Teasley, L. Covi, M. Krishnan, and J. Olson, "Rapid software development through team collocation," 2002.

[5] G. S. and A. M. G.S., "Automatic database construction from natural language requirements specification text," *ARPN Journal of Engineering and Applied Sciences*, vol. 9, no. 8, pp. 1260–1266, 2014.

[6] D. K. Deeptimahanti and R. Sanyal, "Semi-automatic generation of uml models from natural language requirements," pp. 165–174, 2011.

[7] S. Hartmann and S. Link, "English sentence structures and eer modeling," pp. 27–35, 2007.

[8] M. Elbendak, "Parsed use case descriptions as a basis for object-oriented class model generation," 2017.

[9] P. R. Krishna, A. Khandekar, and K. Karlapalem, "Modeling dynamic relationship types for subsets of entity type instances and across entity types," 2016.

[10] A. Vincenzo and V. Gervasi, "Processing natural language requirements," 1997.

[11] F. Meziane and S. Vadera, "Obtaining er dia- grams semi-automatically from natural language specifications," pp. 638–642, 2004.

[12] N. Omar, J. Hanna, and P. McKevitt, "Heuristic-based entity-relationship modelling through natural language processing," pp. 302–313, 2004.

[13] R. Dedhia, A. Jain, and P. K. Deulkar, "Techniques to automatically generate entity relationship diagram," *International Journal of Innovations Advancement in Computer Science*, vol. 4, no. 10, pp. 68–73, 2015.

[14] C. Wohlin and A. Aurum, "An evaluation of checklist-based reading for entity-relationship diagrams," 2004.

[15] F. Meziane and S. Vadera, "Obtaining e-r diagrams semi-automatically from natural language specifications," 2004.

[16] "Database design for business applications," https://www.taxmann.com/bookstore/bookshop/bookfiles/SKSharmacontentchapter7.pdf, accessed: 2021-10-10.

[17] "Database properties," https://www.educative.io/courses/database-design-fundamentals/B6VQBZ6NnnW, accessed: 2021-10-10.

[18] "An introduction to gis," https://www.slideshare.net/sumantagargibhattacharyadas/geographic-information-system-29590419, accessed: 2013-12-31.

[19] "Database management system course (canara engineering college)," https://santoshhiremath.weebly.com/uploads/6/7/0/5/67052617/module_1_dbms-18cs53_.pdf?cv=1, accessed: 2026.

[20] "Database system concepts and architecture," https://studyres.com/doc/4088995/, accessed: 2021.

[21] "Dbms notes," https://www.learnpick.in/prime/documents/notes/details/3918/dbms-notes, accessed: 2014.

[22] A. M. Tjoa and L. Berger., "Transformation of requirement specifications expressed in natural language into an eer model." pp. 206–217, 1994.

[23] R. J. Abbott, "Program design by informal english descriptions." pp. 882–894, 1983.

[24] P. H. Omar, N. and P. M. Kevitt, "Semantic analysis in the automation of er modelling through natural language processing," 2006.

[25] S. S. Chettiar, B. P, H. P, and B. N. M, "Talkie text: The image reader," 2021.

[26] A. Bennetot, I. Donadellod, A. E. Qadic, M. Dragoni, T. Frossard, B. Wagner, A. Saranti, S. Tulli, M. Trocan, R. Chatila, A. Holzinger, A. d'Avila Garcez, and N. D. Rodriguez, "A practical tutorial on explainable ai techniques," 2021.

[27] "Linguistic features," https://spacy.io/usage/linguistic-features, note = Accessed: 2021.

[28] S. K. Kang, L. Patil, A. Rangarajan, A. Moitra, T. Jia, D. Robinson, F. Ameri, and D. Dutta, "Extraction of formal manufacturing rules from unstructured english text," 2021.

[29] N. B. Hemmati, O. Tabibzadeh, and M. Mansoorizadeh, "Adaptation of universal dependencies to specific grammars: The case of persian dependency grammar," 2016.

[30] "Python documentation (logging facility for python)," https://cgi.cse.unsw.edu.au/~cs2041/doc/python-2.7.2-docs-html/library/logging.html?cv=1, accessed: 2011.