

**ENTITY-RELATIONSHIP DIAGRAM GENERATION WITH  
NATURAL LANGUAGE PROCESSING AND MACHINE  
LEARNING APPROACH**

**MERTALİ KÖPRÜLÜ**

**IŞIK UNIVERSITY  
AUGUST, 2023**

ENTITY-RELATIONSHIP DIAGRAM GENERATION WITH  
NATURAL LANGUAGE PROCESSING AND MACHINE  
LEARNING APPROACH

MERTALİ KÖPRÜLÜ

Işık University, School of Graduate Studies, M.S. Program in Computer Engineering,  
2023

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  
AUGUST, 2023

IŞIK UNIVERSITY  
SCHOOL OF GRADUATE STUDIES  
MASTER OF SCIENCE IN COMPUTER ENGINEERING

ENTITY-RELATIONSHIP DIAGRAM GENERATION WITH NATURAL  
LANGUAGE PROCESSING AND MACHINE LEARNING APPROACH

MERTALİ KÖPRÜLÜ

APPROVED BY:

Assist. Prof. Emine Ekin  
(Thesis Supervisor)

Işık University

Assist. Prof. Rahim Dehkharghani

Işık University

Assist. Prof. Faik Boray Tek

Istanbul Technical University

APPROVAL DATE:

.... /..../....

# ENTITY-RELATIONSHIP DIAGRAM GENERATION WITH NATURAL LANGUAGE PROCESSING AND MACHINE LEARNING APPROACH

## ABSTRACT

As software systems continue to grow in complexity, the need for efficient and accurate design methodologies becomes increasingly critical. Entity-Relationship Diagrams (ERDs) provide a powerful visual representation of system structures and dependencies, serving as a foundation for software engineering and database design. However, manually creating ERDs from textual requirements is time-consuming and manual. To address this challenge, this research explores the application of natural language processing (NLP) techniques to automatically extract relevant information from unstructured text and generate ERDs. The proposed approach leverages the strengths of rule-based techniques, semantic analysis, and machine learning algorithms to automatically identify entities, attributes, relationships, and cardinalities from natural language input. Our study offers practical insights into the utilization of linguistic and semantic analysis, and machine learning for efficient information extraction. The proposed system aims to streamline the ERD creation process and improve the accuracy and quality of the resulting diagrams. While the proposed approach shows promising results, the limitations in heuristic rule coverage and data dependencies are acknowledged. Furthermore, the evaluation results demonstrate in detecting entities, attributes, and relations, with f1-scores of 0.96, 0.93, and 0.92, and resolving the components specifications achieved accuracy of 0.87, 0.84, 0.91, respectively. The findings contribute to advancing ERD extraction from text and suggest future research directions for improving the robustness and usability of the solution. The fusion of NLP techniques with ERD creation highlights the potential for enhancing the software development lifecycle and opens new avenues for research in the realm of information extraction from natural language text.

**Keywords:** Entity-Relationship Diagram, Natural Language Processing, Named Entity Recognition, Information Extraction.

# DOĞAL DİL İŞLEME VE MAKİNE ÖĞRENMESİ YAKLAŞIMIYLA VARLIK-İLİŞKİ DİYAGRAM ÜRETİMİ

## ÖZET

Yazılım sistemleri giderek karmaşıklık kazandıkça, verimli ve doğru tasarım yöntemlerine olan ihtiyaç artan bir şekilde kritik hale gelmektedir. Varlık İlişki Diyagramları (ERD), sistem yapılarını ve bağımlılıklarını güçlü bir görsel diyagram ile sunarak yazılım mühendisliği ve dahi veri tabanı tasarımının temelini oluştururlar. Ancak, metinsel gereksinimlerden ERD'lerin el ile oluşturulması zaman alıcı ve zahmet gerektirir iken, tasarım yapan kişinin öznel eleştirisine bağlıdır. Bu zorluğun üstesinden gelmek için bu tez, doğal dil işleme (NLP) tekniklerinin kullanımını ve metinden diyagram ile ilgili gerekli olan bilgileri otomatik olarak çıkarmak ve ERD'ler oluşturmak için incelemektedir. Önerilen bu yaklaşım, doğal dil girdilerinden varlık, varlıkların özniteliklerini ve ilişkilerini ve kardinalitelerini otomatik olarak belirlemek için kural tabanlı tekniklerin, anlamsal analizin ve makine öğrenimi algoritmalarının birleşimini kullanır. Bu çalışma, dilbilimsel ve anlamsal analiz ile makine öğreniminin verimli bilgi çıkarımı için kullanılmasına ilişkin araştırmaları sunarak deneyler yapar ve bu deney sonuçlarını karşılaştırması sonucu önerilen yöntemin eksikliklerini ve güçlü yönlerini bildirir. Önerilen bu sistem, ERD oluşturma sürecini basitleştirmeyi ve bilgi çıkarımı ile ERD'lerin doğru ve kaliteli üretimini amaçlar. Ek olarak, bu değerlendirme, varlık, öznitelik ve ilişkilerin tespitinde sırasıyla 0.96, 0.93 ve 0.92 f1 puanı almış, bileşen özelliklerinin çözümlenmesinde ise doğru diyagram varlıklarının özelliklerini bulmada sırasıyla 0.87, 0.84 ve 0.91 doğruluk oranını elde etmiştir. Elde edilen bu bulgular, metinden ERD çıkarma konusunda ilerlemeye katkı sağlayıp ve dahi çözümün sağlamlığını ve kullanılabilirliğini artırmak için gelecekteki araştırmalar için yönergeler ve çözümler önerir. NLP tekniklerinin ERD oluşturma ile birleştirilmesi ve yazılım geliştirme yaşam döngüsünü geliştirmenin potansiyelini vurgulayarak metinden bilgi çıkarma alanına da yeni araştırma olanakları sunar.

**Anahtar Kelimeler:** Varlık-İlişki Diyagramı, Doğal Dil İşleme, Adlandırılmış Varlık Tanıma, Bilgi Çıkarımı.

## **ACKNOWLEDGEMENTS**

I would like to express my deepest appreciation to my consultant/supervisor, Dr. Emine Ekin, who has the attitude and essence of a genius in my eye for providing guidance and feedback throughout the project. Dr. Ekin constantly helped me to investigate the awareness of the points I could not see most of the time and she gave precious feedback, comments that made me feel encouraged. That was an honor to work with Dr. Ekin.

Mertali KÖPRÜLÜ

## TABLE OF CONTENTS

<b>APPROVAL PAGE.....</b>	<b>i</b>
<b>ABSTRACT .....</b>	<b>ii</b>
<b>ÖZET .....</b>	<b>iii</b>
<b>ACKNOWLEDGEMENTS.....</b>	<b>iv</b>
<b>TABLE OF CONTENTS.....</b>	<b>v</b>
<b>LIST OF TABLES .....</b>	<b>viii</b>
<b>LIST OF FIGURES.....</b>	<b>ix</b>
<b>LIST OF ABBREVIATIONS.....</b>	<b>xi</b>
<b>CHAPTER 1 .....</b>	<b>1</b>
1. INTRODUCTION .....	1
1.1 Problem Definition.....	3
1.2 Purpose of the System.....	4
<b>CHAPTER 2 .....</b>	<b>5</b>
2. OVERVIEW OF ENTITY RELATIONSHIP DIAGRAMS .....	5
2.1 Database Modelling .....	6
2.2 Entity-Relationship Diagrams.....	6
2.3 Components of Entity Relationship Diagram.....	7
2.3.1 Entity.....	7
2.3.2 Weak Entity .....	7
2.3.3 Attribute .....	8
2.3.4 Key Attribute.....	8
2.3.5 Derived Attribute.....	9
2.3.6 Multi-Valued Attribute.....	10
2.3.7 Composite Attribute .....	10
2.3.8 Relationship .....	11
2.3.9 Identifying Relationship.....	11
2.3.10 Cardinalities .....	12

<b>CHAPTER 3 .....</b>	<b>13</b>
3. LITERATURE REVIEW .....	13
3.1 Rule-Based Approaches on Diagram Generation .....	14
3.2 Semantic-Based Approaches .....	18
3.3 Machine-Learning Approach .....	21
<b>CHAPTER 4 .....</b>	<b>25</b>
4. APPROACH.....	25
4.1 Proposed Design .....	25
4.1.1 Dataset Collection.....	26
4.2 Pre-Processing Module .....	28
4.2.1 Sentence Segmentation.....	29
4.2.2 Word Correction (Optional).....	29
4.2.3 Tokenization.....	29
4.2.4 Chunking.....	30
4.2.5 Part-Of-Speech Tagging.....	30
4.2.6 Wordnet Synonym Extraction.....	31
4.2.7 Word Dependency.....	31
4.3 Custom Named Entity Extraction Module.....	34
4.3.1 Dependency Extraction.....	35
4.3.2 Candidate Component Extraction.....	36
4.3.3 Vectorization.....	37
4.3.4 Custom Named Entity Recognizer.....	37
4.4 Component Feature Extraction Module.....	43
4.4.1 Specification Resolver.....	43
4.4.2 Component Tagging.....	46
4.4.3 Reduction of Redundant Information.....	47
4.5 Graph Pre – Processing.....	48
<b>CHAPTER 5 .....</b>	<b>50</b>
5. RESULTS.....	50
5.1 Experimental Results of Custom Named Entity Recognition Module.....	50
5.2 Model Case Outputs.....	52
5.2.1 Case 1: "Attribute" of "Entity" Extraction.....	52
5.2.2 Case 2: Relation Extraction.....	53
5.2.3 Case 3: "Attribute" of "Relation" Extraction.....	57
5.2.4 Case 4: Complex Relations.....	59



5.3 Evaluation .....	64
5.4 Confusion Matrices.....	64
<b>CHAPTER 6 .....</b>	<b>68</b>
6. CONCLUSION AND DISCUSSION .....	68
<b>REFERENCES .....</b>	<b>71</b>
<b>RESUME.....</b>	<b>73</b>

## LIST OF TABLES

Table 3.1 Semantic Roles and Their Definitions .....	21
Table 3.2 Approaches of Relation Extraction from Text .....	24
Table 4.1 Universal Part-of-Speech Tags .....	30
Table 4.2 Spacy Dependency Labels .....	32
Table 4.3 Spacy Token Features .....	35
Table 4.4 Sample of Component Relation Structure.....	47
Table 5.1 Bi-Directional Long Short-Term Memory Network Feature Comparison for Custom Named Entity Recognition.....	51
Table 5.2 Confusion Matrix of Entity Extraction.....	64
Table 5.3 Confusion Matrix of Attribute Extraction.....	65
Table 5.4 Confusion Matrix of Key Attribute Extraction .....	66
Table 5.5 Confusion Matrix of Relation Extraction.....	66

## LIST OF FIGURES

Figure 2.1 Entity Shape .....	7
Figure 2.2 Weak Entity Shape.....	8
Figure 2.3 Attribute Shape .....	8
Figure 2.4 Key Attribute Shape.....	9
Figure 2.5 Derived Attribute Shape .....	9
Figure 2.6 Multi-Valued Attribute Shape.....	10
Figure 2.7 Composite Attribute Shape .....	10
Figure 2.8 Relationship Shape .....	11
Figure 2.9 Identifying Relationship Shape.....	11
Figure 2.10 Cardinality Ratio one-to-many, E1: E2 on Relation R .....	12
Figure 3.1 Proposed Model of Habib .....	15
Figure 3.2 Parser tree of the sentence “X hit the ball.” .....	16
Figure 3.3 Approach of S. Btoush.....	17
Figure 3.4 ERD Modeling Generation Framework.....	17
Figure 3.5 Block diagram of large-scale Object-Based Language Interactor .....	18
Figure 3.6 Illustration of Semantic Net .....	19
Figure 3.7 Model of ER-Converter Tool.....	20
Figure 3.8 Machine Learning Model of Kashmira.....	22
Figure 3.9 Annotated Data output of Kashmira .....	22
Figure 3.10 Proposed Model .....	23
Figure 4.1 Proposed System Architecture .....	26
Figure 4.2 Pre-processing Module .....	28
Figure 4.3 Dependency Output of Given Sentence .....	32
Figure 4.4 Example Dependency Tree .....	33
Figure 4.5 Custom Named Entity Extraction Module.....	34
Figure 4.6 Default Named Entity Recognizer output of SpaCy.....	38
Figure 4.7 Recurrent Neural Network Cell Structure.....	39

Figure 4.8 Example Usage of RNN in NER .....	40
Figure 4.9 Long-Short Term Memory Cell Structure .....	40
Figure 4.10 Bi-directional LSTM Architecture.....	41
Figure 4.11 Component Feature Extraction Module.....	43
Figure 4.12 Illustration of dependency tree of 5 <sup>th</sup> sentence .....	44
Figure 4.13 Graph Pre-Processing Module .....	48
Figure 5.1 Generated Entity Relationship Diagram of Scenario 1.....	53
Figure 5.2 Illustration of Elmasri on Scenario 1 .....	53
Figure 5.3 Generated Entity Relationship Diagram of Scenario 2 (a), ERD illustration (b) .....	54
Figure 5.4 Generated Entity Relationship Diagram of Scenario 3.....	55
Figure 5.5 Elmasri Illustration on Scenario 3 .....	55
Figure 5.6 Generated Entity Relationship Diagram of Scenario 4.....	56
Figure 5.7 Generated Entity Relationship Diagram of Scenario 5.....	57
Figure 5.8 Generated Entity Relationship Diagram of Scenario 6.....	58
Figure 5.9 Dependency Tree Illustration of sentence: “Suppliers, Parts, and Projects have ternary relation called ‘supply’ which has quantity attribute.”.....	59
Figure 5.10 Solution Entity Relationship Diagram of Scenario 7.....	60
Figure 5.11 Generated Entity Relationship Diagram of Scenario 7.....	61
Figure 5.12 Generated Entity Relationship Diagram of Scenario 8.....	62
Figure 5.13 Generated Entity Relationship Diagram of Scenario 9.....	63

## **LIST OF ABBREVIATIONS**

DBMS: Database Management System  
EERD: Enhanced Entity Relationship Diagram  
ERD: Entity Relationship Diagram  
LOLITA: Large Scale Object-Based Language Interactor  
LSI: Latent Semantic Indexing  
LSTM: Long-Short-Term Memory  
MBSP: Memory-Based Shallow Parser  
ML: Machine Learning  
NER: Named Entity Recognition  
NLP: Natural Language Processing  
POS: Part of Speech  
SVD: Singular Value Decomposition  
SVM: Support Vector Machine  
TBP: Transition-Based Parser  
TF-IDF: Term Frequency – Inverse Document Frequency  
UML: Unified Modelling Language

# CHAPTER 1

## 1. INTRODUCTION

Modeling plays a significant role in developing software systems at various stages of the development life cycle with different aims. In requirements engineering, the analysis models aim to represent the captured requirements given in natural language form from different perspectives, i.e., structural, behavioral. Analysis models use only the existing information, requirements, where the audiences who are the stakeholders are expected to communicate via the models to validate the requirements in terms of consistency, completeness, correctness, unambiguity. On the contrary, in architectural design activity where the target is the software development professionals, to satisfy the requirements the solution domain objects are modeled as interacting components.

With a thorough analysis of the different points of view regarding the requirements, creating analysis models plays a crucial role. To effectively represent the overall structure, it is beneficial to employ a class diagram, which captures the relationships and attributes of different classes within the system. Additionally, dynamic models are essential for modeling system behaviors, highlighting the flow of information and interactions between various components. However, when it comes to representing the data requirements, the Enhanced Entity Relationship (EER) model stands out as the most suitable choice as a high-level conceptual data model. While different modeling methods exist in system modeling, it is important to note that in the absence of an object-oriented structures and models, the EER model remains consistently relevant for accurately representing the data components and their relationships. Therefore, considering the diverse modeling options available, the EER model consistently emerges as the foundation for a robust conceptual data model.

Despite the importance of EERD, the process of construction from text is still largely manual, which can be time-consuming. Existing approaches for generating EERD from text typically rely on either syntactic or semantic analysis. Although the manual based approach is restricted to specific understanding of input and hand-crafted rules, the solution method is based on natural language processing (NLP) to automate this task.

The use of NLP techniques to extract information from unstructured text has gained significant attention in recent years. One of the key challenges in NLP is to retrieve information from text. The subject of our study is focusing on extracting relevant information to find out the relationship between entities and their attributes. Thus, this essential information leads to understanding and modeling complex systems' data requirements to generate various diagrams. The actual problem comes from the need of domain specific knowledge when it comes to understanding related information. There are such NLP methods in information extraction, such common examples are text summarization, named entity recognition (NER), sentiment analysis, and topic modelling. However, those methods need to be utilized and build problem domain of our study.

When it comes to extracting information in relation extraction for requirement analysis, there are several methods provided by NLP. Although the methods are focusing on generalized extracting relevant information from the text, that information could be used to utilize generation of different conceptual diagrams. The common approach when it comes to generation of diagram is the rule-based diagram generation. The information to be extracted is set on rules which have been defined and according to that information the diagrams have been created. The rule-based diagram generation approach can be categorized into two main categories: manual rules and automated techniques. In the manual approach, human experts define a set of explicit rules and guidelines to guide the diagram generation process. These rules are typically based on domain knowledge and expertise, allowing for customization, and fine-tuning of the generated diagrams. However, this approach can be time-consuming and subjective, as it heavily relies on human judgment and may lack scalability, yet the most common.

On the other hand, automated techniques aim to generate diagrams automatically by leveraging algorithms, like machine learning (ML). These methods analyze the underlying data or input specifications and use predefined algorithms to infer the relationships and structures necessary for generating the diagram. Automated

approaches offer the advantage of scalability and consistency, as they can manage large datasets efficiently and consistently apply the generation rules. However, they may require extensive preprocessing and datasets to achieve accurate and reliable results.

While automated based tool of (S. Btoush & M. Hammad, 2015) have been successful in experimental cases. However, this tool is often limited by lack of contextual understanding. Approach also requires a significant amount of domain-specific knowledge. As such, with the development of NLP, recent researchers have focused on improving information extraction to overcome this issue.

Moreover, ML techniques require annotated data, which there are no available labeled data on the field for this purpose. Therefore, in our proposed work have created its own dataset and its labeling system according to the syntactical features of the input text. NLP methods used throughout our study, such features like part of speech (POS) taggers, sentence segmentation, tokenizers, and NER to label the input data.

Our study gathered its own dataset from textbooks and web sources. One of the modules has been responsible for generating its set by labeling entities, attributes, and relations with their synonyms. Another module is to prepare numerical vectors with each word's syntactic attributes to feed custom NER. ML model used to create custom NER which provide with labeled tokens as if a token is an entity, attribute, or relation.

In our study, we propose a novel approach to extract relations and automatically generate Entity Relationship Diagrams (ERD) from text that combines custom NER with syntactic features of sentences. Our approach leverages the strengths of both methods to overcome the limitations of existing approaches and produce more accurate and comprehensive ERDs.

## **1.1 Problem Definition**

The problem addressed in our study is the extraction of corresponding relationships to build relational diagrams. Our study used that extracted information to generate EERD from data requirements given as natural language text in English. EERD are commonly used in producing software systems and database design to represent the relationships between entities and attributes in a system. The specific challenge addressed in this proposed thesis is how to accurately identify and label entities, attributes, and relationships from natural language text, and then use this information to generate a complete and accurate EERD.



To describe the system's domain problem, a problem statement is composed in the requirement statements which explain entities/tables, relations, and attributes with their specifications. Requirement statements do not have a standard format, thus any plain text file which describes requirement of components can be entered into the model. Dataset have been collected from textbooks, scholarly articles, and websites which are simply descriptive texts without annotations.

## **1.2 Purpose of the System**

The purpose of this system is to automatically find relations between entities and extract EERD components from natural language text, with the goal of improving the systematic EERD creation in software engineering and database design. By leveraging techniques from NLP, including NER, part-of-speech (POS) tagging, and dependency parsing, the system aims to accurately identify and label entities, attributes, and relationships from text, and use extracted features to generate an EERD. The system will be designed to be scalable and adaptable to new domains. Ultimately, the system aims to streamline the EERD creation process and improve the accuracy and quality of the resulting diagrams.

Several theories discussed various approaches and techniques used for this task, including rules and ML based approaches, and will highlight the potential benefits and challenges associated with each approach. The chapters of this thesis are organized to provide a comprehensive overview of the problem domain. Chapter 2 an overview of ERD is presented, establishing a solid foundation for the subsequent chapters. In Chapter 3, digs into a literature review, examining the merits and demerits of rule, semantic and ML based approaches in modeling EERD. The approach and implementation details of the proposed approach are discussed in Chapter 4. Chapter 5 is dedicated to presenting the experimental results and engaging in an in-depth discussion of the findings. Finally, Chapter 6 concludes the thesis by summarizing the research findings, discussing limitations, and identifying potential areas for future research. In essence, this thesis offers an exploration of generating EERD from unstructured input text, aiming to contribute to the ongoing research in this field.

## **CHAPTER 2**

### **2. OVERVIEW OF ENTITY RELATIONSHIP DIAGRAMS**

A relational database is a type of DBMS that organizes and stores data in a structured manner using tables, where each table represents an entity or a concept in the real world. The foundation of a relational database is the concept of a relation, which refers to a set of related data organized into rows (also known as tuples) and columns (also known as attributes). These tables are interconnected through relationships, which define how the data in one table is related to the data in another. Relational databases have become a widely adopted solution for storing and managing data in various domains, ranging from business applications to scientific research. Their inherent structure, flexibility, and ability to manage complex relationships make them a valuable tool for efficiently organizing and accessing data in a structured manner. (Elmasri and Navathe 2016)

EERD helps in designing the schema for a relational database and aids in the maintenance by providing a graphical representation of the data and its relationships. EERD is used by people such as database designers and developers to create a visual representation of the data model for a particular system. This representation helps them to identify the relationships between entities and to ensure that the database is designed to meet the specific needs of the organization.

Components, ERD is a visual representation of entities and the relationships between them in a database. An entity in an ERD represents a real-world object or concept that has data attributes that can be stored in a database. For example, in a university database, entities may include students, courses, and professors. Relationships between entities in an ERD are depicted using lines, symbols, and

cardinality indicators that specify the number of instances of one entity that can be associated with another entity. (Elmasri and Navathe 2016)

Overall, ERDs provide an effective tool for modeling complex relationships and designing effective database systems, helping designers identify data redundancies and inconsistencies and ensuring that databases are designed to meet specific organizational needs. This section introduces the fundamental concepts of databases, including their advantages and basic components. It also briefly presents an overview of ERD.

## **2.1 Database Modelling**

The process of database modeling involves creating a conceptual design, which is crucial for the success of a database system. This conceptual design can be achieved through different methods such as ERDs or object modeling. ERD consists of entities, attributes, and relationships which are represented on a diagram using various shapes with different meanings. Object modeling, on the other hand, uses class diagrams and is commonly applied in software which is implemented in Unified Modeling Language (UML). Although there are different approaches to conceptual design, our study primarily emphasizes the use of ERDs.

## **2.2 Entity-Relationship Diagrams**

The conceptual design of an ERD involves creating a high-level, abstract representation of the data entities and their relationships in a database system. It focuses on the overall structure of the database and the relationships between the data entities, without worrying about implementation details such as data types or storage mechanisms.

The process of constructing an ERD is to define the requirements of the system. This involves identifying the entities and their attributes, as well as the relationships between them. The requirement analyst and system designers must collaborate closely with the stakeholders and end-users to gather information about the system, its goals, and its requirements. During this process, the analyst must consider factors such as data redundancy, data consistency, and data integrity and ensure that the system meets the specific needs of the organization. This step is time consuming and requires

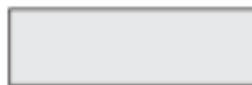
high domain knowledge. Once the conceptual design is complete, the programmers can move on to the logical and physical design phases, which involve translating the conceptual design into a detailed data model and implementing the database system. (Elmasri and Navathe 2016)

### **2.3 Components of Entity Relationship Diagram**

The ERD typically consists of three main components: entities, attributes, and relationships. Entities are represented as rectangles, with their attributes linked within the rectangle. Attributes are the specific characteristics or properties of an entity, and they are represented as ovals. Relationships are represented as lines connecting the entities via relations, and they indicate the nature of the connection between the entities. By using these components, designers can model the structure of a database and ensure that it accurately reflects the relationships and attributes of the entities involved. This section outlines the fundamental elements of ERDs. (Elmasri and Navathe 2016)

#### **2.3.1 Entity**

The ER model represents entities as the primary objects, which are real-world things with an independent existence. These entities can either be physical objects, such as a car or a house, or conceptual objects, such as a job or a course.

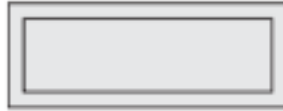


**Figure 2.1** Entity Shape

Each entity has its own unique characteristics and can be distinguished from other entities. An entity is represented by a rectangle in an ERD as Figure 2.1.

#### **2.3.2 Weak Entity**

A weak entity is an entity in a database that cannot be uniquely identified by its own attributes alone. It depends on a related entity called a strong entity to give it meaning and context. A weak entity is always identified by a combination of its own attributes and the attributes of the related strong entity.



**Figure 2.2** Weak Entity Shape

For example, consider a database that stores information about bank accounts. A weak entity in this database might be a transaction, which depends on the related strong entity of an account to give it context. A transaction entity would have attributes such as date, time, and amount, but it would also have a foreign key that references the primary key of the account with which it is associated. This means that a transaction entity cannot exist without an associated account entity, and its primary key is a combination of its own attributes and the primary key of the associated account entity. Weak entities are often represented as Figure 2.2 by using a double rectangle.

### 2.3.3 Attribute

Attributes are characteristics or properties of an entity. An attribute is represented by an oval or ellipse in an ERD as Figure 2.3.



**Figure 2.3** Attribute Shape

Examples of attributes for a customer entity could include customer ID, name, address, and phone number. Categorization of attributes in ERD as follows:

- Key Attribute
- Derived Attribute
- Multi-valued Attribute
- Composite Attribute

### 2.3.4 Key Attribute

A key attribute, also known as a candidate key, is an attribute or a set of attributes in a database table that can uniquely identify each record or row in the table. A key

attribute must be unique, meaning that no two records in the table can have the same value for the key attribute(s).



**Figure 2.4** Key Attribute Shape

For example, in a table that stores information about customers, a unique identifier such as a customer identification number could be a key attribute. This would ensure that each record in the table can be uniquely identified by its customer identification number. Key attributes are important in database design as they are used to enforce data integrity and to ensure that each record in the table is unique.

### 2.3.5 Derived Attributes

In a relational database, a derived attribute is an attribute which has derived or calculated from using other attributes in database. A weak attribute is an attribute that cannot be uniquely identified using its own attributes alone. A weak attribute is always part of a composite key and depends on the presence of another entity to identify it uniquely.



**Figure 2.5** Derived Attribute Shape

For example, consider a table that stores information about rooms in a hotel. The room number alone may not be sufficient to identify a unique room, as there could be multiple rooms with the same number in different buildings or on different floors. In this case, the floor number or building name would also be needed to uniquely identify the room. Therefore, the room number would be considered a weak attribute, as it depends on the presence of the building and floor number to identify it uniquely. For instance, of derived attribute, the entity “Customer” in this hotel has attributes like date of birth and age. The attribute “age” could be derived attribute which is calculated from

the attribute called “date of birth.” Derived attributes are often represented in an ERD by using a dashed oval or ellipse as Figure 2.5.

### 2.3.6 Multi-Valued Attributes

A multi-valued attribute is an attribute that can have multiple values for a single instance of an entity in a database. This means that a single entity can have multiple values for a specific attribute.

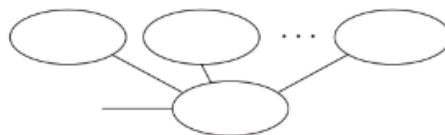


**Figure 2.6** Multi-Valued Attribute Shape

For example, a person entity may have multiple phone numbers or email addresses associated with it. Multi-valued attributes are represented in an ERD by using a double oval or ellipse as Figure 2.6.

### 2.3.7 Composite Attribute

A composite attribute is an attribute in a database that is made up of multiple sub-attributes. A composite attribute can be broken down into smaller parts, each of which represents a separate piece of information about the entity being modeled.



**Figure 2.7** Composite Attribute Shape

For example, consider a table that stores information about a person, where one of the attributes is "Full name." Full name can be broken down into smaller sub-attributes such as first, middle, and last name. Each of these sub-attributes provides additional information about the person's address and can be used to query or manipulate the data more effectively.

### 2.3.8 Relationship

In an ERD, a relation refers to the association or connection between two or more entities. It is represented as a line connecting the entities, and it indicates the relationship between them. The relationship cardinality can be one-to-one, one-to-many, or many-to-many.



**Figure 2.8** Relationship Shape

For example, consider a database that stores information about a school. The ERD for this database might include entities such as "students," "teachers," and "courses." A relation in this context could be the association between students and courses. Since a student can enroll in courses, and a course can have students, the relationship would be named "enroll." A relationship is represented by a diamond shape in an ERD as Figure 2.8.

### 2.3.9 Identifying Relationship

An identifier relationship is a type of relationship in a database where a child entity's primary key is also part of the parent entity's primary key. Identifier relationships are commonly used in database design when there is a one-to-many relationship between two tables.



**Figure 2.9** Identifying Relationship Shape

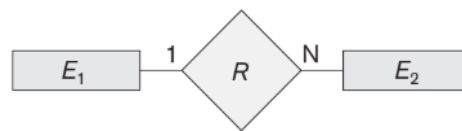
For example, consider a database with two entities: a "Customer" and an "Order." Each order in the order table belongs to a single customer in the customer table. The customer's ID is used as the primary key in the customer table, and the order



table has its own primary key as well as a foreign key that references the customer's ID. In this case, the relationship between the two tables is an identifier relationship because the customer's ID is part of the order's primary key, represented as Figure 2.9.

### 2.3.10 Cardinalities

Cardinality defines the number of occurrences of one entity that are related to the number of occurrences of another entity. Cardinality is represented by notation on the relationship line as seen in Figure 2.10, such as one-to-one, one-to-many, or many-to-many.



**Figure 2.10** Cardinality Ratio one-to-many,  $E_1: E_2$  on Relation  $R$

## **CHAPTER 3**

### **3. LITERATURE REVIEW**

Software systems are becoming increasingly complex and designing them is becoming more challenging than ever. Among the many techniques that have been developed to address this challenge is the use of EERD, which provides a visual representation of a system's structure and dependencies. To address this challenge, researchers have explored the use of NLP techniques to extract relationships from the text and generate varying diagrams. This literature review provides an overview of the approaches that have been proposed for this problem, including rule-based and ML-based approaches, as well as the potential applications and challenges of relation extraction from text to generate diagrams.

The studies which are focusing on generating several diagrams (e.g., Class, Use-Case), are all in the field of information extraction from the text input. A variety of articles differ on how to extract the information. While some studies put assumptions and gives direction to the user of the system with specific input format, other researchers decided to use rule-based approach to automatically extract information by defining overall possible input patterns as rules.

These input patterns might be quite complex and challenging by the nature of language. Studies are highlighting their way of information extraction techniques with varying NLP approaches; hence the studies are trying to extract information from text.

For example, the sentence "There are lecturers who teach courses.". This seemingly simple sentence contains a wealth of information that can be automatically inferred by the reader. Through their knowledge of language and the world, the reader can deduce the existence of a relationship between the lecturer and the course, as well as the fact that the course is taught to students. However, in information extraction,

capturing such complex information requires a variety of techniques, including linguistic analysis for syntax and semantics rules for extraction, as well as heuristics for handling world knowledge. (Kashmira & Sumathipala, 2018)

Data modeling involves identifying specific elements within user requirements, such as entities, attributes, and relationships, which are entered in textual form. NLP plays a crucial role in this process by identifying and extracting the nouns and other parts of speech (POS) tags necessary to determine these elements, including their attributes and cardinalities. NLP involves several steps, including morphological analysis, tokenization, POS tagging, chunking, and parsing, which are used to process user requirements written in natural language.

In information extraction, identifying entities and their relationships from unstructured text is a key challenge. This task involves two sub-tasks: recognizing named entities and extracting relations between them. Early work used a sequential approach, with separate models for entity extraction and relation classification. However, in recent years, end-to-end systems have become more prevalent in evaluation. Current NERs are not fully compatible with extracting the entities that point this problem domain. According to the experiments of (Zhong & Chen, 2021), it is found that cross-sentence information is useful to improve identifying named entities in sentences.

Some researchers have used NLP to generate diverse types of diagrams. These researchers have employed rule based, semantic based and ML-based approaches to achieve the desired outcomes. However, much of the research mostly focuses on the class diagram or the use case diagram. The following section provides a brief overview of previous research in this area.

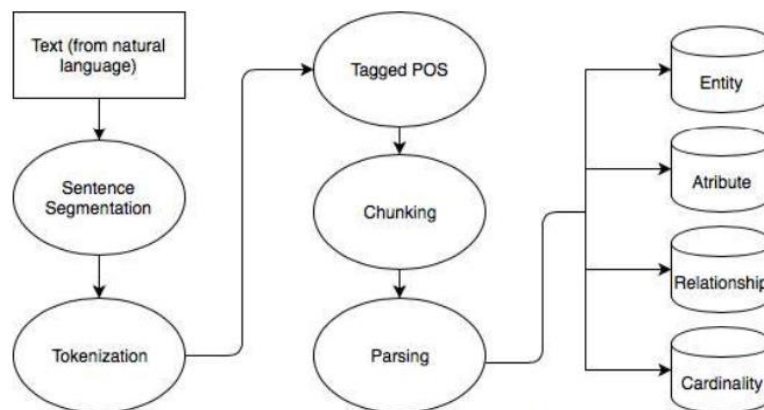
### **3.1 Rule-Based Approaches on Diagram Generation**

Numerous studies have explored the structural analysis of language for generating ERDs. Additionally, with the aid of NLP, there are two approaches for mapping natural language to conceptual design: rule-based and probability-based mapping. Each approach has its own advantages and disadvantages.

Rule-based design store rules and heuristics in multiple knowledge bases. These tools utilize a parsing algorithm that extracts information from a grammar and lexicon that fulfills the tool's needs. During parsing, the sentence is analyzed based on the

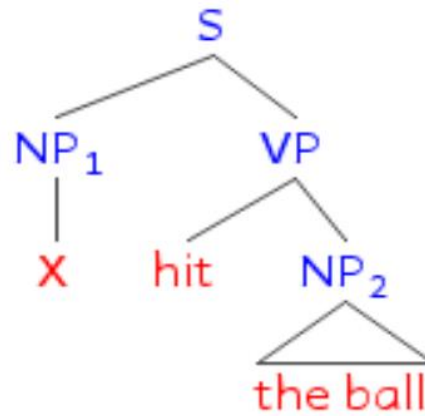
syntactic rules and the lexicon. The parsing results are then passed on to the rules and heuristics, which establish a connection between linguistic and design knowledge. The rule-based approach for translating verbs and nouns into entities and relationships has limitations because it may not always be accurate to convert all verbs into relationships or all nouns into entities. To address this issue, researchers have turned to using the n-gram model, which is a probabilistic language model that relies on the probability of a word in a document based on its previous (e.g., n-1) words. N-grams are commonly used in language modeling and can help to overcome the limitations of rule-based approaches. (Habib 2019)

Study of (Habib 2019), is focused on the process flow with given Figure 3.1. below. The process starts with segmenting sentences. This is followed by tokenization, which produces words as outputs. POS tag had been assigned to each token to aid parsing step which is a morphological analysis and applied heuristic rules.



**Figure 3.1** Proposed Model of Habib Reference: Habib, M., Kasra. (2019). On the Automated Entity-Relationship and Schema Design by Natural Language Processing. *The International Journal of Engineering and Science (IJES)*, 8(11), 42–48. doi:10.9790/1813-0811034248

Chunking and parsing are applied to the input words, where multiple possible analyses are conducted. Parsing is the process of assigning a syntactic analysis to a string of words, using grammar to create a parsing tree. Finally, extracted information from the parsing tree which is used to create an ERD. The process flow of Habib is illustrated in Figure 3.1.

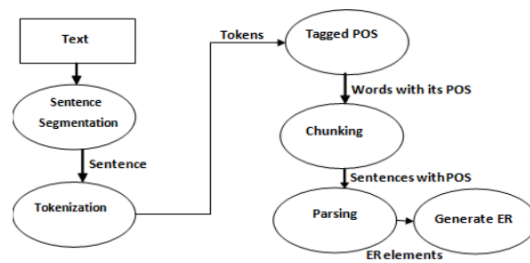


**Figure 3.2** Parser tree of the sentence “X hit the ball.” Reference: Habib, M., Kasra. (2019). On the Automated Entity-Relationship and Schema Design by Natural Language Processing. *The International Journal of Engineering and Science (IJES)*, 8(11), 42–48. doi:10.9790/1813-0811034248

The author defined the rules from parse tree of given unstructured-text input. As a conclusion of approaches, probabilistic based, and rule-based approach achieved 91.3, 86.9 precision, and 89.0, 86.3 recall, respectively. The author claims that the rule-based approaches are failing to map entities if the specific patterns have not been given to the system. (Habib 2019)

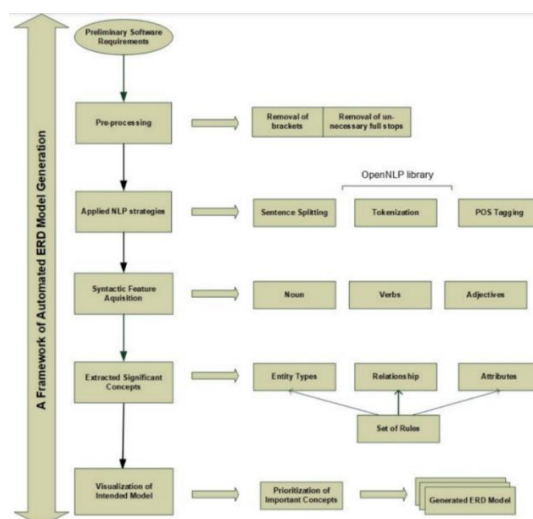
The study of (Karaa et al. 2015) points out this problem, and states that those diversity of patterns is hard to deal with. They have produced the idea that using word dependencies from Stanford’s NLP library to increase the number of patterns that they can cover against the unexpected patterns with using semantic analysis. Study claims to have huge size of set of patterns against other studies and one patterns can discover more information like class names and attributes. However, Karaa’s system is lacking from redundant information problem. For instance, if the input contains synonyms of two same entity (e.g., Client and Customer), the system will not be able to deal with it and creates 2 separate entities. The authors are aware of this problem and claim that to be expanded into semantic information might solve this problem and improve output of the overall system.

Another study (S. Btoush & M. Hammad, 2015), had focused on generation of ERD from text, had define specific heuristic rules for extracting entity, attribute, and relation with using structural analysis. However, study has not covered weak, derived attributes, and cardinalities of relations. As a conclusion, the article states that, the word dependencies need to be considered to extract components of ERD.



**Figure 3.3** Approach of S. Btoush Reference: S. Btoush, E., & M. Hammad, M. (2015). Generating ER diagrams from requirement specifications based on Natural Language Processing. *International Journal of Database Theory and Application*, 8(2), 61–70. doi:10.14257/ijdta.2015.8.2.07

One of the latest studies has been done by (Vidya Sagar & Abirami, 2014), by clarifying problem statements with defining criteria, to minimize unexpected inputs from users in the application. The study achieves extracting information including preprocess steps like extracting grammatical occurrences and extracting elements from sentence (defining rules for subject and object extraction, identifying rules to detect entities, attributes, and relationships). It also focused on syntactic feature extraction with the use of POS tags, design of elements by dependency analyzer as seen in Figure 3.4. In addition to that focused on relation types like aggregation, composition, and generalization. As a conclusion of this work states that this approach is also needed for semantic information in conceptual model.

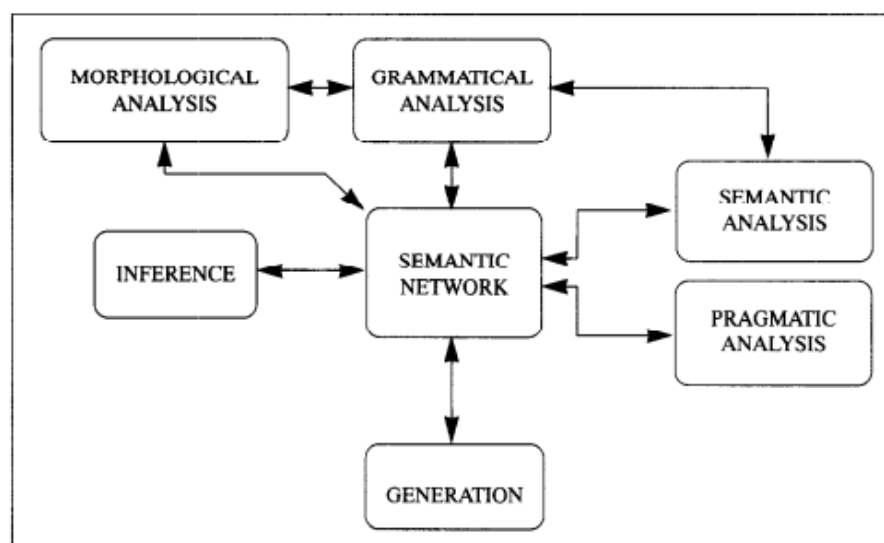


**Figure 3.4** ERD Modeling Generation Framework Reference: Sagar, V. B. R. V., & Abirami, S. (2014). Conceptual modeling of natural language functional requirements. *Journal of Systems and Software*, 88, 25–41. doi:10.1016/j.jss.2013.08.036

According to the recent study (Ahmed, Ahsan, Qamar, & Butt, 2021), has observed that there is a lack of research aimed at automating the generation of a Class diagram from an initial set of requirements in various research repositories. The article created pre-defined rules to extract entities using sentence segmentation, tokenization, and POS tagging. Study removed redundant classes by defining dictionary which includes irrelevant glossary words. The algorithm structure described as  $C: \in \{C, A, O, R\}$  where C is the candidate class, A is the attribute, o is the operation and r are the relationship. Relationships are considered and extracted with description as  $R: R: \in \{rT, Cr, Rc\}$ , where rT is type of relationship, Cr is cardinality and Rc is related Class.

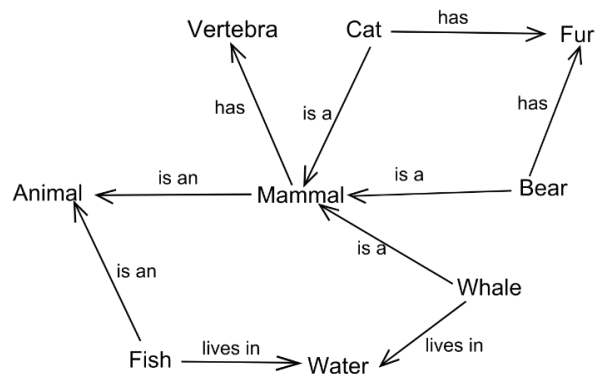
### 3.2 Semantic-Based Approaches

Recent development in NLP, in 2019 March, the authors (Utama & Jang 2019), have decided to use natural language tools to construct class diagram. The study has been aided by the NLP library called Spacy. Article achieved the solution by using preprocessing methods to normalize input data then combined semantic analysis with dependency relation. Approach is to define rules by using more relevant parameters (word dependencies) which previous studies have missed. Articles claim to achieve higher precision in their system than the other rule-based approaches.



**Figure 3.5** Block diagram of large-scale Object-Based Language Interactor Reference: Morgan, R., & Garigl, R. (1995). *Natural Language Processing with LOLITA. Endeavour, 19(1)*, 11–15. doi:10.1016/0160-9327(95)98888-m

The system which has been proposed by (Morgan & Garigl, 1995), -Large scale Object-based Language Interactor, Translator and Analyzer (LOLITA)- is able to generate object model from natural language specification as seen the Figure 3.6. Unfortunately, the approach of the author is domain specific and input language is assumed in assertive sentences with the form of Subject, Verb and Object. Moreover, the study uses domain-specific databases for labeling. In addition to grammatical analysis, the study also focuses on the meaning of the text and creating semantic network representation to create semantic networks that allow study to construct graph. According to the author of this article, the system lacks uniformity of population density, therefore it is not robust and needs more domain specific network information.



**Figure 3.6** Illustration of Semantic Net Reference: Morgan, R., & Garigl, R. (1995). Natural Language Processing with LOLITA. *Endeavour*, 19(1), 11–15. doi:10.1016/0160-9327(95)98888-m

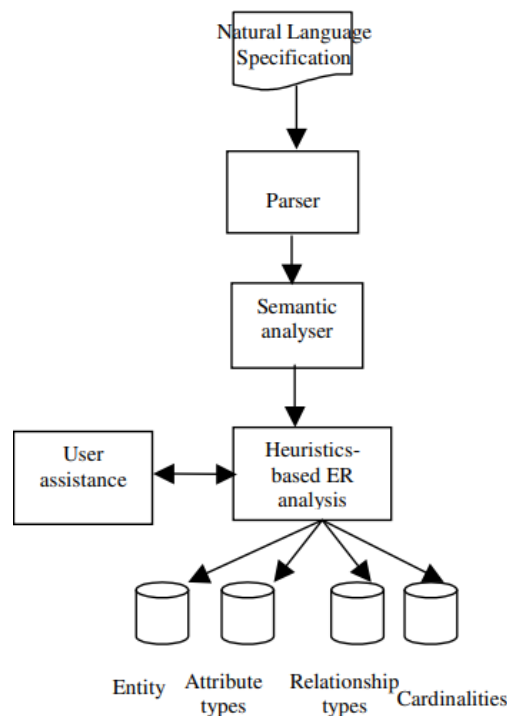
The article (Kchaou, Bouassida, & Ben-Abdallah, 2017) claims to define patterns using text similarity technique. The authors use term frequency-inverse document frequency (TF-IDF) and latent semantic indexing (LSI). TF-IDF is a numerical statistic that reflects how important a word is to a document in a collection or corpus. The higher the TF-IDF score for a word in a document, the more relevant that word is to the document. In other words, a high TF-IDF score indicates that a word is both frequent in the document and rare in the corpus, which means that it is a suitable candidate for identifying the main topics or themes of the document.

LSI was founded to identify patterns in the relationships between the terms and concepts from collection of text. If the words are used in the same context, they are highly probable to be in similar meaning. It uses singular value decomposition (SVD) technique to identify patterns. Overall, the approach is to find similarity of input



documentations with overall the datasets and produce output. However, they lack large evaluation and datasets and perform bad on unseen data. The study defined a set of input rules to eliminate redundancy. As a result, the usage of LSI out-performs the TF-IDF technique with following F-measure scores, 0.77, 0.40.

One of the articles (Omar, Hanna, & Mc Kevitt, 2006), had proposed use of semantic analysis to automate generation of ER models from natural language. Author states processes as follows: First, parsing the English sentences and obtain their POS tags, a parser such as Memory-Based Shallow Parser (MBSP) has used. The output produced by the parser is then fed into a semantic analyzer for semantic analysis. Afterward, the parsed text is passed through ER-Converter to identify appropriate data modeling elements.



**Figure 3.7** Model of ER-Converter Tool Reference: Omar, N., Hanna, P., & Mc Kevitt, P. (2006). Semantic Analysis in the Automation of ER Modelling Through Natural Language Processing. *2006 International Conference on Computing & Informatics*. doi:10.1109/icoci.2006.5276559

There are several steps involved in achieving the desired ER model from the natural language input, including POS tagging, semantic role assignment, syntactic and semantic heuristics application, and human intervention in (Omar, Hanna, & Mc Kevitt, 2006). For example, the output of semantic analyzer for the following sentence:

The *purchaser*(AGENT)sends an *order*(THEME) to the *supplier*.(GOAL). The author decided to use semantic tags to write corresponding heuristic rules. After assigning these semantic and syntactic labels with weights, the candidate entities and attributes are served to the user to investigate and approve.

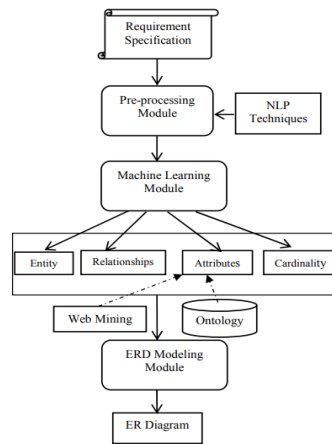
**Table 3.1** Semantic Roles and Their Definitions Reference: Omar, N., Hanna, P., & Mc Kevitt, P. (2006). Semantic Analysis in the Automation of ER Modelling Through Natural Language Processing. *2006 International Conference on Computing & Informatics*. doi:10.1109/icoci.2006.5276559

Semantic Role	Definition
AGENT	The volitional causer of an event
THEME	The participant most directly affected by event
...	
RESULT	The product of an event
GOAL	The destination of an object of a transfer event

Finally, the attributes are attached to their corresponding entities, entities are attached to their corresponding relationships, and entities are attached to their corresponding cardinality, resulting in the final ER model. (Omar, Hanna, & Mc Kevitt, 2006)

### 3.3 Machine-Learning Approach

The study (Kashmira & Sumathipala, 2018), which has been published on 2018, proposed approach includes a model that automatically generates an ERD while minimizing user intervention. This model addresses issues such as incomplete information and redundancies in requirement specifications. It consists of three main modules: the Preprocessing Module, ML Module, and ERD Modeling Module.



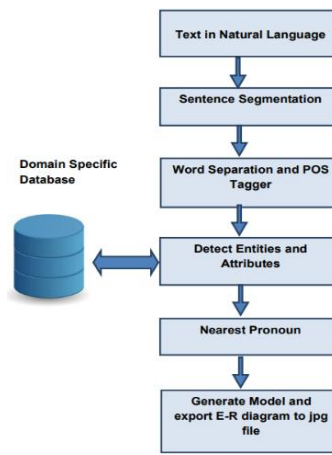
**Figure 3.8** Machine Learning Model of Kashmira Reference: Kashmira, P. G., & Sumathipala, S. (2018). Generating Entity Relationship Diagram from Requirement Specification Based on NLP. *2018 3rd International Conference on Information Technology Research (ICITR)*. doi:10.1109/icitr.2018.8736146

Pre-processing module is responsible to segment the sentences and convert plural nouns to singular to satisfy standard general rule to identify entity. Module of machine-learning is responsible to find entities, relationships, and attributes with supervised-learning approach. When designing an ERD, the designer typically uses world knowledge to decide which attributes are related to specific entities. To tackle this challenge, the proposed model employs ontology and web mining techniques to identify and filter out the relevant attributes from extracted entities. Figure 3.9 illustrates the output of study, words with green background are labeled as “entity,” pinks are “attribute” and yellows are “sub-classes.” The study tried several ML algorithms like random forest, naïve bayes, decision table, and sequential minimal optimization. However, the study has not fully finished, it cannot find relationship types, cardinalities, and other component specifications for ERD.

there are **lecturers** who teach **courses** **instructors** who assist in doing tutorials and practical and projects under taken by lecturers a **lecturer** can be employed as either **professor** **senior lecturer** or **probationary lecturer** each **lecturer** has an **id** **name** **gender** **year of birth** **salary** **educational qualifications** a **professor** has a **rank** a **senior lecturer** has a **grade** and **probationary lecturer** has a **period** a **course** has a **course code** **title** and **credit** value a **course** can be either undergraduate or post graduate an **under graduate course** can have a **type** while **post graduate course** has a **level** an **instructor** has an **id** and **name** each **course** is assigned to **lecturer** and **instructor** a **course** is assigned to one or many **lecturers** and one or many **instructors** an **instructor** is assigned for one or many **course** and one or many lecturers a **lecturer** can undertake zero or many research **projects** for a **project** one or many **lecturers** can be involved each **project** has a **title** **grant** **name** **duration** and **budget**

**Figure 3.9** Annotated Data output of Kashmira Reference: Kashmira, P. G., & Sumathipala, S. (2018). Generating Entity Relationship Diagram from Requirement Specification Based on NLP. *2018 3rd International Conference on Information Technology Research (ICITR)*. doi:10.1109/icitr.2018.8736146

The proposed model of the (Ghosh, Mukherjee, Chakraborty, & Bashar, 2018), has focused on generating ERD, with usage of pre-defined database and support vector machine (SVM) classifier to identify ERD components. It compares found “NOUN” phrases with its database, then states to use SVM classifier to identify components and expand its database with synonym of found entities. After that fetches all adjectives and denotes that token as attributes of entity within same sentence. However, the system works on assertive sentences (i.e., subject + verb + object) but cannot deal with complex sentences.



**Figure 3.10** Proposed Model Reference: Ghosh, S., & Bashar, R. (2018). Automated Generation of E-R Diagram from a Given Text in Natural Language. *International Conference on Machine Learning and Data Engineering (iCMLDE)*. doi:10.1109/icmlde.2018.00026

Current NER taggers label words in a text that are the names of specific things such as people or company names. However, they do not correctly label words as entities in the ERD. To address this issue, a custom NER module has been implemented to identify certain features of the ERD, such as entities and attributes. Study has annotated as entity, sub-entity, attributes of dataset which contain 50 documents. Overall, the result of the study only labels entity, sub-entity-and attributes by using ML algorithm and states ERD modelling module to be implemented as future works. However, the author does not provide accuracy and F1-scores.

**Table 3.2** Approaches of Relation Extraction from Text

Study	Approach	Entity		Attribute				Relation		
		Normal	Weak	Normal	Key	Derived	Multi-Valued	Normal	Identifying	Cardinality
(Karaa et al. 2015)	Rule	✓		✓	✓			✓	✓	✓
(Ahmed, Ahsan, Qamar, & Butt, 2021)	Rule	✓		✓				✓	✓	✓
(Habib 2019)	Rule	✓		✓	✓			✓	✓	
(S. Btoush & M. Hammad, 2015)	Rule	✓		✓	✓			✓		
(Vidya Sagar & Abirami, 2014)	Rule	✓		✓	✓		✓	✓	✓	✓
(Utama & Jang 2019)	Semantic	✓	✓	✓				✓	✓	
(Morgan & Garigl, 1995)	Semantic	✓						✓		
(Omar, Hanna, & Mc Kevitt, 2006)	Semantic	✓	✓	✓				✓		✓
(Kchaou, Bouassida, & Ben-Abdallah, 2017)	Semantic	✓		✓	✓			✓		✓
(Kashmira & Sumathipala, 2018)	Machine Learning	✓		✓						
(Ghosh, Mukherjee, Chakraborty, & Bashar, 2018)	Machine Learning	✓		✓				✓		
Our Study	Machine Learning	✓	✓	✓	✓	✓	✓	✓	✓	✓

## **CHAPTER 4**

### **4. APPROACH**

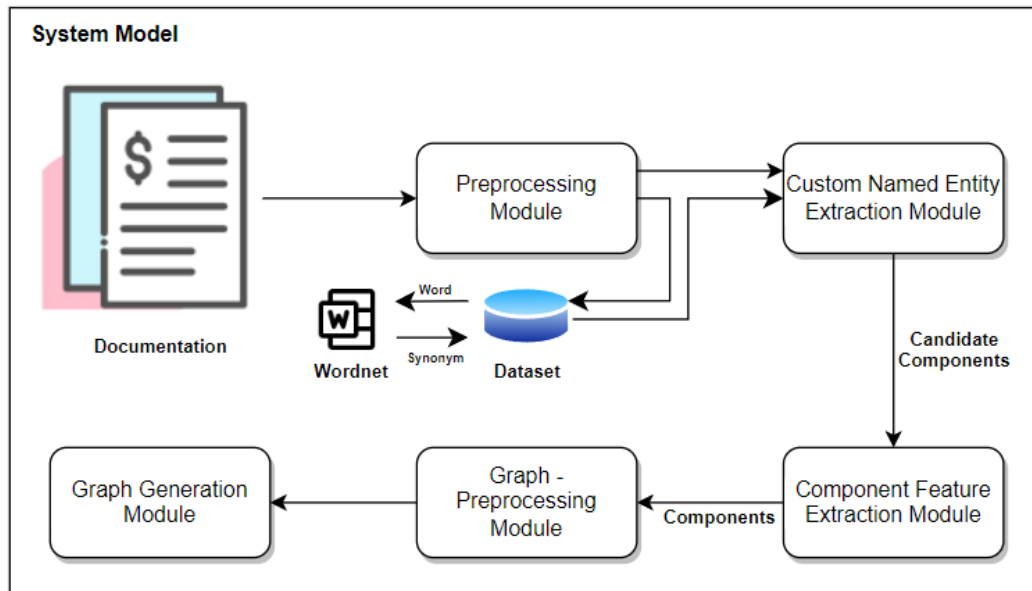
This section provides a detailed introduction to the design and implementation of the proposed system. The design of the system refers to the plan for the system, while the implementation refers to the steps taken to execute that plan.

NLP encompasses a wide range of functionalities related to human language, including the identification of semantic and grammatical patterns in text using a combination of general rules and ML models. Since our study lacks an annotated dataset, it relies on self-labeling of document input using a set of general rules established by previous articles. However, these rules do not cover all grammar combinations and patterns, and thus, the custom NER extraction module is utilized to identify hidden features and patterns in the input text.

To achieve this, NLP techniques from widely used libraries like Spacy and NLTK are employed to analyze the text and link it to its linguistic information. The proposed system, i.e., the design, outlines the process to be followed, which will be presented in the following part.

#### **4.1 Proposed Design**

The proposed system consists of five different modules. Figure 4.1 illustrates the process line of the system module. The first module is responsible for receiving the input document and utilizing pre-processing techniques to extract linguistic features from each sentence in the document. These linguistic features are then linked to each token/word, also forwarded to wordnet to obtain synonyms with same POS tags to create its own dataset.



**Figure 4.1** Proposed System Architecture

The second module is responsible for creating a custom NER. Our study used NLP framework called Spacy. Spacy provides a base environment to build custom models. According to (Jiang, Banchs, & Li, 2016) , Spacy outperforms the common other frameworks like “LingPipe” and “NLTK” in NER. This module creates vectors by using word dependencies and POS tags. The ML model has been fed by the information extracted from the vector created in the pre-processing module. In other words, this module generates custom NER; retrieves candidate components and links them with corresponding labels which are “Entity,” “Attribute,” and “Relation,” alongside the vectorized sentences.

The third module aims to find and extract specifications of each component, by assigning component features. This module takes the output labels to determine specific attribute information, such as whether an attribute is a primary key, has multi-valued information, or whether an entity is weak. Model tries to address the issue of word redundancy, which was a challenge in the (Karaa et al. 2015). By eliminating redundancy to detect components, the proposed system will provide a more accurate to extract entities.

The fourth module of the proposed system is responsible for assigning component features to GraphViz nodes. This module takes the output labels which are tagged by component feature extraction module, to assign component shapes and styles to

generate ERD. In the following subsections, this study will provide a detailed explanation of the modules and features used in the proposed system.

#### 4.1.1 Dataset Collection

The dataset used in this study serves as a foundational resource for training and evaluating the Named Entity Recognition (NER) model focused on extracting components of ERDs from unstructured text. The dataset, consisting of 105 documents with absence of explicit labeling. The dataset was meticulously compiled from a diverse range of sources, primarily ERD texts of authoritative database textbooks and reputable online resources. Given the nature of the dataset, our approach involves transforming this unlabeled text into a labeled format. This process entails assigning approximate labels to each instance of an ERD component within the text by using general heuristic rules. The evaluation set consists exclusively of extracted from authoritative database textbooks and obtained ground-truth labels for this evaluation set, we meticulously extracted information from solution figures provided within the textbooks. The components recognized by the NER model are categorized into distinct features, each capturing important attributes of the components. Entities are divided into “Normal” and “Weak”. Attributes are categorized into “Normal”, ”Key”, ”Derived” and ”Multi-valued”. Relation component has divided into “Normal”, “Identifier” and “Cardinality”. In conclusion, the dataset collection process involves combination of information from authoritative database textbooks and online resources to construct a diverse and extensive dataset of unlabeled requirement descriptions.

The evaluation set had been extracted from solution figures to have a robust dataset that serves as the foundation for evaluation of system.

An example of collected dataset documentation is “ *Employee contains name, address, salary, sex, B\_date, ssn. Ssn is a primary attribute. Each Department has a name, unique number, location, and Number\_Of\_Employees and particular one Employee who manages the department. Department may have several locations. Number\_of\_employees is a weak attribute. The start\_date is recorded in manages relation. Many Employees works for one Department. An Employee can supervise many Employees. A Department controls many numbers of Projects which many employees can work on. Work on relation stores number\_of\_hours attribute. Both*



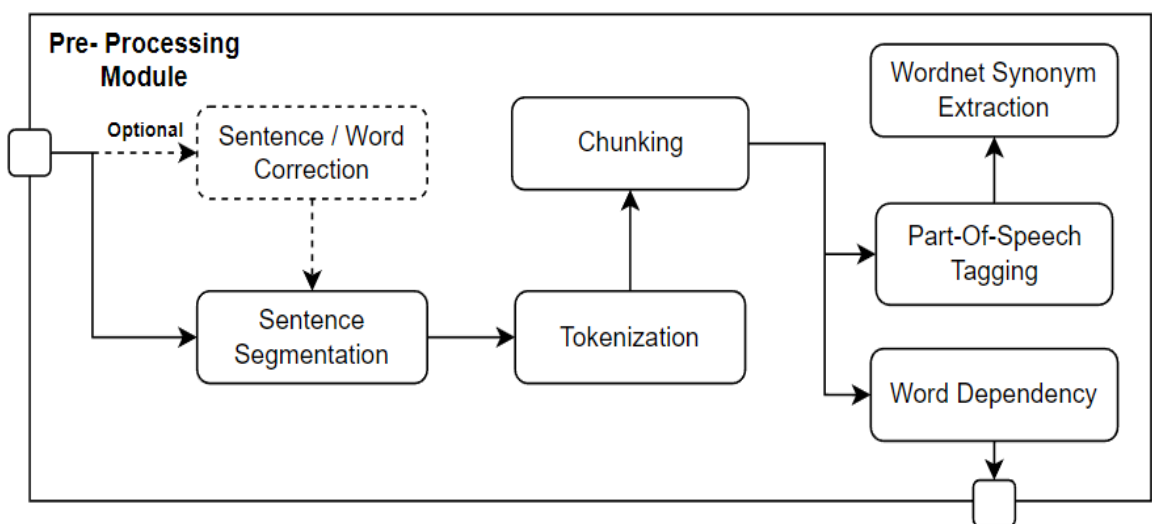
*participants are determined to be total. An Employee depends on many Dependent entity. Dependents of is identifier relation for the Dependent entity which is a weak entity. Dependent entity contains sex, birth\_date, relationship, and name.”*

## 4.2 Pre-Processing Module

The pre-processing module is a crucial part of the proposed system and serves to extract various linguistic features from text input. The features used in this module illustrated on Figure 4.2, are derived from the Spacy and NLTK library as follows:

1. Sentence segmentation,
2. Word correction (optional),
3. Tokenization
4. Chunking
  - a. POS Tagging
5. Wordnet Synonym Extraction
6. Word Dependency

Example documentation is going to be used while explaining each process as follows: “Student has name, surname, unique id, and multiple addresses. Students take many courses. Each course has a unique number, name, and instructor name. A course can be given by an instructor. Each instructor has a unique id, name and given course information. An instructor can give many Courses.”



**Figure 4.2** Pre-processing Module

### **4.2.1 Sentence Segmentation**

Sentence segmentation is the process of splitting a text document into individual sentences. This is important because most NLP tasks require sentence-level input. The most common way to perform sentence segmentation is to split the text by punctuation marks. However, this can be tricky when dealing with abbreviations, numbers, and other exceptional cases. Spacy has been employed to manage such cases. The input document is segmented into individual sentences. For example, the given documentation in proposed design can be segmented into the following sentences:

- Student has name, surname, unique id, and multiple addresses.
- Students take many courses.
- Each course has unique number, name, and instructor name.
- A course can be given by an instructor.
- Each instructor has unique id, name and given course information.
- An instructor can give many Courses.

### **4.2.2 Word Correction (Optional)**

Word correction is an optional step that involves correcting common spelling and grammatical errors in the text. This step can improve the accuracy of downstream NLP tasks. Word correction can be done using various techniques like dictionary-based correction, rule-based correction, or ML-based correction.

### **4.2.3 Tokenization**

Tokenization is the process of separating a phrase into words, also known as tokens. This is done by splitting the sentence based on spaces, punctuation, and other delimiters. Tokenization is important because most NLP tasks operate on individual words, and not on entire sentences. Example of application on first sentence could be tokenized into the following words:

- Student
- has
- name
- surname
- unique

- id,
- and
- multiple
- addresses.

Spacy had struggled to tokenize and label the word which indicates identification number (*id*). Those inputs might be like: “student\_id, course-id, couseID, ...,” which the spacy could not understand those tokens. Therefore, the system looked for prefixes and postfix of words which contain the term in them. However, spacy also stuck to tokenize the term “id,” it assumed the term is 2 individual tokens and give output as “i” and “d”. System investigated these tokens and merged them as “id.”

#### 4.2.4 Chunking

Chunking is the process of grouping individual words together into meaningful phrases or "chunks". This is done based on grammatical rules and syntactic structures of the language as “NOUN,” “VERB,” etc. ...

#### 4.2.5 Part-Of-Speech Tagging

POS tagging is the process of assigning a grammatical label to tokenized sentences, such as noun, adjective, or verb. POS tagging is important for many NLP tasks, like NER, sentiment analysis, and text classification. After tokenization, spaCy can label and parse a given document by universal POS tag set as seen on Table 4.1.

**Table 4.1** Universal Part-of-Speech Tags Reference: <http://spacy.io/linguistic-features>

Name	Abbreviation	Example
Adposition	ADP	In, to, during...
Adverb	ADV	Very, well, exactly, up, down, when, where, how, never...
Auxiliary	AUX	Has, was, should, must, ...
Coord. Conjunction	CCONJ	And, or but
Determiner	DET	A, an, the, this...
...		
Noun	NOUN	Girl, tree, beauty, decision...
Numeral	NUM	0,1,2,3,4, ...
Pronoun	PRON	I, you, he, myself, mine, everybody, ...
Proper Noun	PROPN	Mary, John, London...
Verb	VERB	Run, eat, runs, ate, running, eating...

The trained components include binary data that is generated from showing the system an adequate number of examples, allowing the software to make generalized predictions across the language. For example, the first sentence can be POS tagged as follows:

- Student (NOUN)
- has (VERB)
- name (NOUN)
- surname (NOUN)
- unique (ADJ)
- id (NOUN)
- and (CCONJ)
- multiple (ADJ)
- addresses (NOUN)

#### **4.2.6 Wordnet Synonym Extraction**

WordNet is a lexical database that groups English words into sets of synonyms, also known as synsets. This step involves using WordNet to extract synonyms of words found in the text. This can help to reduce word redundancy and improve the accuracy of downstream NLP tasks. This module obtains entities, attributes, and relations then stores the synonyms which are with same POS Tags from wordnet. For example, synonyms for the entity "Student" can include "learner", "pupil", and "scholar".

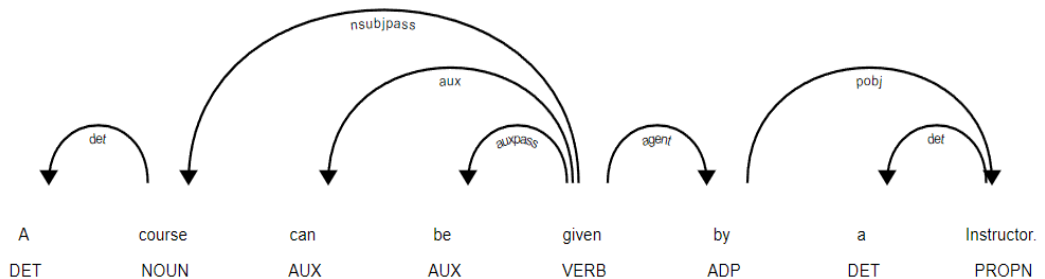
#### **4.2.7 Word Dependency**

Word dependency is the relationship between words in a sentence based on their syntactic and semantic roles. This step involves analyzing the dependency relationships among the words in the text to extract more information about the structure and meaning of the tokens. The table of dependency tags which has been published by the spacy is as follow in Table 4.2.

**Table 4.2** Spacy Dependency Labels Reference: <https://spacy.io/linguistic-features>

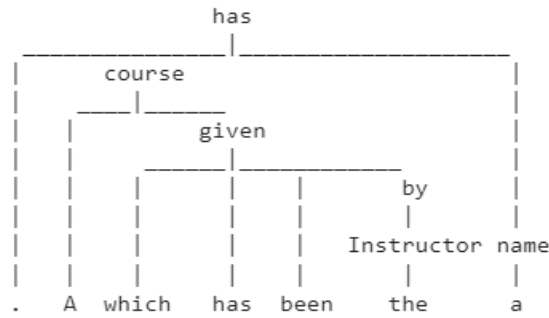
Label	Description
ROOT	Root
ADVMOD	Adverbial Modifier
AMOD	Adjectival Modifier
AUX	Auxiliary
AUXPASS	Auxiliary (passive)
...	
CSUBJ	Clausal Subject
DOBJ	Direct Object
NSUBJPASS	Nominal Subject (Passive)
NUMMOD	Numeric Modifier
POBJ	Object of Preposition

The dependencies between the words in the text are identified according to the table above. For example, in Figure 4.3, "A course can be given by an instructor.", the word "course" is dependent on the word "given", and the word "Instructor" is dependent on the word "by". Therefore, it is possible to find out the tokens which are related to subjects and objects of the sentence according to dependencies between root word which is "given."



**Figure 4.3** Dependency Output of Given Sentence

In addition to that, Spacy provides illustration of word dependency hierarchy in tree representation. Structure is finds child of root word of the sentence and creates tree as shown in Figure 4.4. This would help us to observe relations in tokens. The dependency tree illustration of the sentence "A course which has been given by the instructor, has a name." as the follows:



**Figure 4.4** Example Dependency Tree

The dependency of the sentence is the word “has” and leads to extract relation and attribute of entity “Course.” If the root verb is stating subject ownership of the word (has, have, contain, include, ...), the right sub-tree is responsible for finding attributes. If the containing component is an entity, it basically an indication of ownership relation. For example, “Department has Employees.” The entities are “Department” and “Employee” while the root dependency is “has.” Therefore, “has-a” relation identifies entity. The module called specification resolver will investigate individual found entities, to find whether that has-a relation identifies the entity. The root of the left sub-tree is the word “Course” which indicates entity with dependency subject, so if its child node represents action so that tree is indicating relation.

In this example Course is “entity,” and the token “name” is attribute of “Course.” On the left sub-tree with the root “Course” will be investigated and the relation verb token is found “Given” as relation. In addition to that the right sub-tree contains another “NOUN” token “Instructor” which has the dependency label as “object,” indicates the relation refers to its root noun phrase which is “Course.” Therefore, each sub-tree will be investigated separately and considered as a separate tree whether to find components like “Entity,” “Attribute,” or “Relation.” Wordnet had been used to extract synonyms of found candidate components to reduce redundant information as much as possible which to be used in custom NER and specification resolver module.

As a conclusion of this module, the sentences could be corrected to reduce errors that can be caused from user inputs. Then the splitting sentences in each document would help to investigate and extract tokens more accurately. The Spacy provides sentence segmentation, tokenization, chunking, POS tagging and word dependencies. In addition to that, proposed approach to use WordNet-Synonyms to detect possible entities or components.

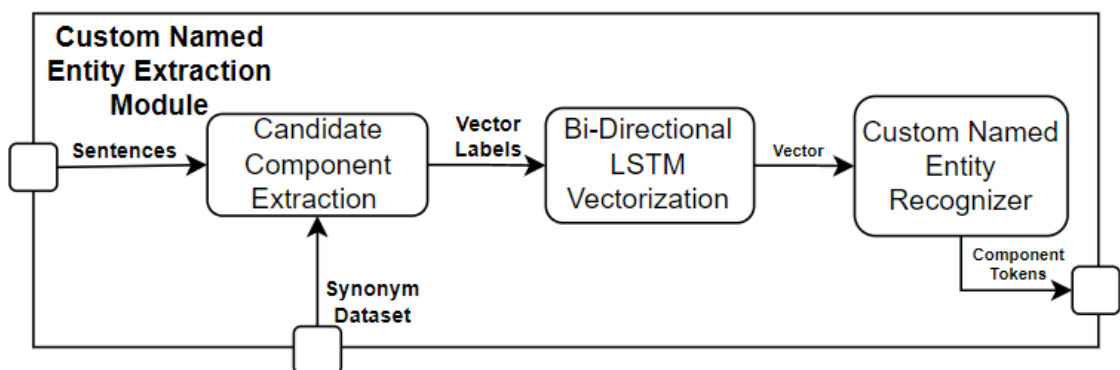
### 4.3 Custom Named Entity Extraction Module

In the custom NER module of our proposed system, applies advanced techniques to enhance the extraction and labeling of key information from the input text. This module plays a crucial role in leveraging the power of ML algorithms to improve the accuracy and efficiency of the overall system. The section will provide a detailed explanation of each process. Figure 4.5 illustrates the processes as follows:

1. Candidate Component Extraction
2. Dependency Extraction
3. Candidate Extraction (Entity, Attribute, Relation)
4. Vectorization
5. Custom Named Entity Recognizer

The module consists of several interconnected processes, starting with the extraction of dependency information, followed by the identification of candidate labels based on predefined rules. Subsequently, the label data is vectorized to transform it into labels for ML.

Finally, the processed data is fed into a custom NER module, which further refines the output by recognizing and categorizing specific entities within the text as “Entity,” “Attribute,” and “Relation.” Through the integration of these processes, Custom Named Entity Extraction Module empowers the system to capture related components from the complex linguistic structures present in the input data.



**Figure 4.5** Custom Named Entity Extraction Module

### 4.3.1 Dependency Extraction

To achieve the desired labeling, our study utilized the capabilities of the Spacy library. With integration of Spacy, the input document can be parsed and tagged with relevant information. This is where the trained pipeline and its statistical models become instrumental. In addition to that, enabling Spacy to make accurate predictions regarding the appropriate tags or labels within the given context.

The trained components within Spacy comprise binary data that is generated through examples and enabling the system to make generalized predictions across the language. For instance, in English, it is highly probable that a word following the token "the" would be classified as a noun. As an illustration of the output generated by the Spacy pipeline, consider the following example provided in documentation of spacy as examined in Table 4.3.

**Table 4.3** Spacy Token Features

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

The input sentence is considered in 8 different topics:

- **TEXT:** The original word text.
- **LEMMA:** The base form of the word.
- **POS:** The simple part-of-speech tag.
- **TAG:** The detailed part-of-speech tag.
- **DEP:** Syntactic dependency, i.e., the relation between tokens.
- **SHAPE:** The word shape – capitalization, punctuation, digits.



- **ISALPHA:** Does the token consist of alphabetic characters.
- **STOP:** Is the token part of a stop list, i.e., the most frequently used words of the language (“Linguistic Features · spaCy Usage Documentation,” n.d.)

#### 4.3.2 Candidate Component Extraction

This step involves applying general rules to extract candidate components, like entities and attributes, from the text. For example, in the sentence "Student has name, surname, unique id, and multiple addresses", the candidate entity would be "Student" and the attributes would be "name", "surname", "unique id", and "multiple addresses". These rules can be based on linguistic patterns, regular expressions, or domain-specific knowledge. The idea of manual rules had been applied to extract candidates with generalized terms.

Root word had been focused and categorized into 2 different topics to determine if corresponding dependency tree is defining a specification or relation. If it is representing relation then find the subject and object by looking at left and right subtree of root word, with help of dependency tags like “nsubj”, “csubj”, “iobj”, “obj.” Tokens examined on the left and right subtree if root is standing for any other identifier/determining subtrees or relations.

If root verb with lemma is in “have, contain, include, store”, algorithm finds for tokens with “NOUN”, “PROPN”, or “ADJ” based pos tags, to determine object and subject in addition to the dependency tags. Some noun phrases could consist of more than one token, which is why compound and modifier tokens also though and noun phrases captured and merged for instance the noun “Care Centre.” Those tokens have to be merged into 1 token and may represent entity, and attribute.

The found subjects tokens examined as whether they have been in other documents as creating relationships while iterating to have further look to identify that noun and object token is representing “Attribute” or “Entity.” If they are a sentence that specifies attribute information of found token, it is highly probable that token is an entity. Otherwise, it is highly probable that component is attribute. In addition to that, dependency also provides object, subject relation, therefore validating to make sure to have correct objects/nouns, subject/nouns, adjectives on right and left sub-tree. Tokens found shall has the dependency as subject, object. The information of candidate component had been expanded by look-up set from generated synonym dataset to

determine possible candidate entities, attributes of relation with the same POS tag. (Chen, 1983)

NLTK library is employed to find corresponding tokens, start, and end indexes. After that the components found labeled as “Entity,” “Attribute,” and “Relation,” with next to their indexes. Let  $\hat{C}$  denote list of sequence with position features:

$$\hat{C} = (w_i, \langle \text{entity}, c_{start}, c_{end} \rangle), \dots, (w_j, \langle \text{relation}, c_{start}, c_{end} \rangle), \dots, (w_z, \langle \text{attribute}, c_{start}, c_{end} \rangle), \dots \quad (4,1)$$

Specifically, given an input sentence  $X$  and found components  $C$  -structured list-, as:  $C = (w_i, S)$ , where  $w_i$  is token to be tagged,  $S$  is the component spans, where  $S = (tag, c_i, c_f)$ , where tag is “entity”, “attribute”, or “relation” and  $c_i, c_f$  is position markers, which has inserted into the input sentence after extraction of component.

### 4.3.3 Vectorization

Vectorization is the process of representing words or phrases as numerical vectors that can be used as input to ML algorithms. This step involves converting the tokens, entities, attributes, and relations extracted from the text into numerical vectors. This is important because most ML algorithms require numerical input. POS, and dependency tags have merged into each individual token before converting sentence into vector. Vectorization has assigned one of the components of Spacy called Tok2Vec.

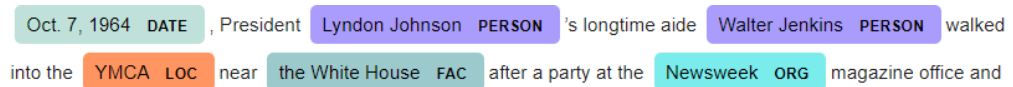
Approach is to feed generated vectors into custom NER to specialize domain of our study. Moreover, ML model in custom NER also considers surrounding context of given token by using Bi-directional Long-Short-Term-Memory Network (LSTM), therefore it may give an insight about semantic relationships even between tokens in different sentences. This allows the model to capture semantic and syntactic relationship between words/tokens. Custom NER requires vectorized sentences which to be explained in the following section.

### 4.3.4 Custom Named Entity Recognizer

Although convolutional neural networks can deal with the varying input sizes, fixing the input size in natural language might cause to lose essential information. The

tokens are passing information to further ones that creates repeated action. Therefore, Recurrent Neural Network (RNN) approaches shall be applied in custom NER.

Custom NER, tailored to the requirements of our problem definition. Default NER by Spacy and other libraries offer a range of predefined labels as seen in Figure 4.6, tags like “PERSON,” “LOCATION,” “TIME,” ... but these labels do not cover required components for the problem domain.



Oct. 7, 1964 DATE, President Lyndon Johnson PERSON's longtime aide Walter Jenkins PERSON walked into the YMCA LOC near the White House FAC after a party at the Newsweek ORG magazine office and

**Figure 4.6** Default Named Entity Recognizer output of SpaCy

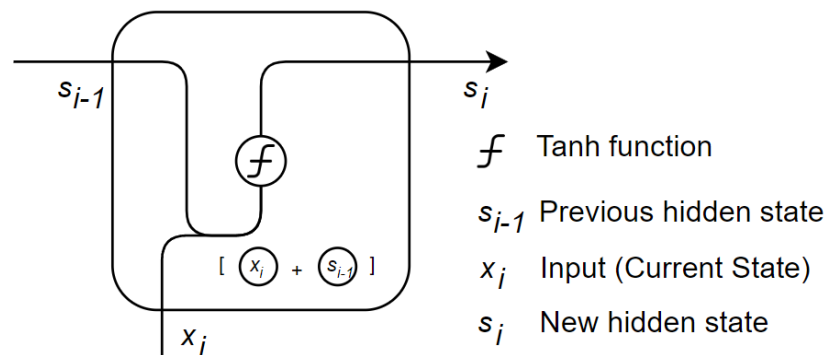
On the other hand, this approach enables us to precisely capture the specific elements that are integral to constructing an ERD, allowing us to gain deeper insights into the underlying structure of the given text. Custom NER module allows tailing the tagging process to align with the distinctive requirements of problem domain with semantic and structural aspects.

To feed the custom NER model converting tokens into flattened vectors is required. This study used Multi Hash Embedded Vector which was generated by Tok2Vec. Embedding table of multi hash embedded vector contains orthographic features like normalized form, prefix, suffix, and shape of each token. In order to avoid collusion in hash table, the spacy uses multi hash tables. According to research using dependency of tokens, POS tags, contributes NER systems. Therefore, rather than using basic NER structure in Spacy which uses “NORM,” “PREFIX,” “SUFFIX,” and “SHAPE. Our study also experimented several other features: “POS,” ”TAG”, “DEP”, “ENT\_TYPE”, “ ENT\_IOB”, where “POS” is simple POS tags, “TAG” is extended POS tags, “DEP” is dependency labels, “ENT\_TYPE” is token’s entity label, “ENT\_IOB” is inside-outside-begging (IOB) tags to detect name phrases. According to experimental results, the best model achieved when the model features are “NORM,” “PREFIX,” “SUFFIX,” “SHAPE,” “POS,” and “DEP.”

The NER components are a transition-based parser (TBP) model. TBP is an approach used in structured prediction, what the process of predicting the structure is transformed into a sequence of state transitions as the type of NER.

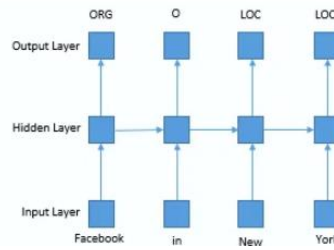
- Toc2vec: Responsible to map each token into a vector for each batch.
- Lower Network: Construct token feature pair as a vector for each batch. (Token, Feature)
- Upper Network: State representation | Prediction Scores (“Model Architectures · spaCy API Documentation,” n.d.)

Custom NER computes representation of each token in one row and creates sentence matrix. Then gets sequence of word vectors which has derived from embedding step. After that, module reduces number of vectors and classifies the token label to decide whether the token shall classified as “Entity,” “Attribute,” or “Relation.” The problem is to solve the input size problem because each size of sentence might be different than each other. The previous states must be observed to decide annotations. Therefore, to deal with sequential data, rather than using feed-forward neural network, RNN cells shall be used in our study domain.



**Figure 4.7** Recurrent Neural Network Cell Structure

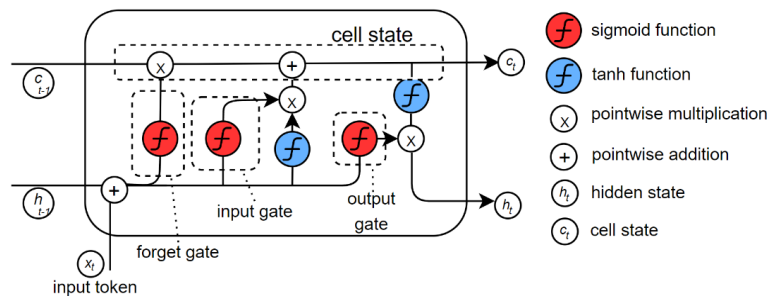
Figure 4.7 illustrates the basic structure of RNN cells. Initial step which has been taken is the transforming words to machine readable vectors, then processed a sequence of vectors one by one while processing it passes the previous head state to the next step of the sequence in hidden state access to neural networks of memory. Figure 4.8 illustrates the RNN structure in NER. Cells hold information on previous data that the network has seen before.



**Figure 4.8** Example Usage of RNN in NER Reference: <https://zhoubeiqi.medium.com/named-entity-recognition-ner-using-keras-lstm-spacy-da3ea63d24c5>

Although traditional RNN models in NLP tasks achieve promising results, are not fully capable of identifying all the named entities in long-term dependencies. RNN cells stores the information for current features as well neighboring features for prediction. It is particularly useful to find and detect short distance features, but when it comes to predicting features with long distance it fails because of the conducting irrelevant information from previous states. Architecture needs to retain information from distant past and get rid of unnecessary information when it is needed.

According to the study (Bajpai 2021), Bi-directional LSTM leads researchers to effectively address the challenges posed by long-term dependencies and achieve improved performance in text classification problems with higher performance and accuracy than LSTM. That is the reason why the system uses Bi-Directional LSTM Network applied to retrieve information in the further tokens next to past ones.

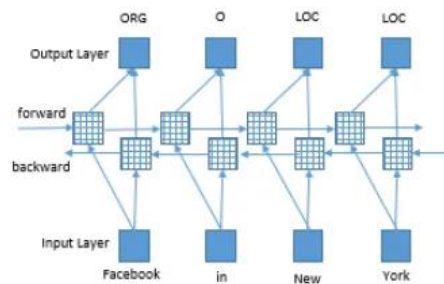


**Figure 4.9** Long-Short Term Memory Cell Structure

In LSTM networks, the conventional hidden layers of RNNs are replaced with memory cells and gates. Each individual chunk of layer conducts information, that the forget gate decides what information is essential to keep or forget. Passing the value

through the sigmoid function to increase or decrease importance of the information, enhancing their ability to capture and retain long-range dependencies. As seen in Figure 4.9, The input gate decides which information the cell state is going to hold by passing the previous hidden state and current input into sigmoid and tanh function that enhances the values in between -1 and 1. Sigmoid output will decide if current information is to keep or forget from the tanh output. The cell state gets multiplied with output of forget gate vector. This process even creates a possibility of converging values in cell state to 0. The cell adds the input gate output and alters cell state vector to new values that network finds relevant to create new cell state. The last gate is the output gate which is responsible to create new hidden state shall carry, for next cell by using filtered information from the cell state and previous hidden states information. This attribute is particularly crucial in the context of sentence structures, where understanding the relationships between distant words is essential.

The LSTM network alters hidden layers of RNN with updated by memory cells that ensures gradient can pass across many time steps without altered by next token. The problem here is that the token information is passed to the further tokens, but the initial tokens are lacking from further token information. When the model deals with passive sentences, there is information loss in LSTM. To annotate entities, attributes and relations, the information in the further tokens might be crucial for initial tokens. That is the reason bi-directional LSTM developed. Figure 4.10 shows bi-directional LSTM structure which is a combination of two LSTMs, while one of them sequences from start to end, second LSTM runs end to start.



**Figure 4.10** Bi-directional LSTM Architecture Reference: <https://zhoubeiqi.medium.com/named-entity-recognition-ner-using-keras-lstm-spacy-da3ea63d24c5>

Reduction in number of vectors into the single vector for each hidden layer, individual vectors are summarized and derived hidden parameters according to the given label. Hence the last vector contains relevant information about the given labels, with two networks output the prediction has been done with minimized information loss. This might be class label, a real value, or a vector, but system investigates if a given token is an “entity,” “attribute,” or “relation.” The model has been fed by the vector which has been created by the vectorization step, and the labels are derived from candidate extraction component. For example, the sentence “Apple is something that...” may related to company “Apple” or the fruit. However, the application of LSTM does not know about what “Apple” means, since it has lack of information from the future, but the bi-directional LSTM captures this entity.

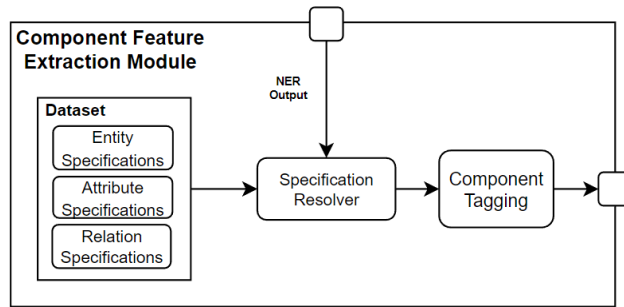
Spacy provides trainable components that help to build custom components in the network. To build a pipeline for defining a custom NER in spaCy, several steps have been taken. The initialization of a blank spaCy model by merging NER component to the pipeline. This component serves as the foundation for training custom NER model. With the pipeline set up, proceeding to train the NER model using the embedded vectors and annotated vectors. This involves running multiple iterations of the training process, during which the model learns to recognize and label the target entities based on the provided examples. The training involves optimizing the model's parameters to minimize the loss and improve its performance. According to the experimental results, enhancing base features of Spacy with POS tags, and dependency labels performed best among the other models.

The proposed pipeline only consists of NER therefore upper state is NER Model. The model has bi-directional LSTM encoder with relative length according to the input size and the layer of TBP's length is set to 16. The batch size is 8, and epoch is set to 10. Pieces to use in state prediction is set to 3. The Training data consists of 105 individual documents with approximately 3200 words. As an output, the model predicts if a token is an entity, attribute, or relation. Example output of this module is structured from input “*Student has name and address.*”. The vectorization done after the feature set covered with tags which are ”NORM”, “PREFIX”, “SUFFIX”, “SHAPE”, “POS”, “DEP”. The output of this module gives structure as: {`entities`: [(0, 7, `entity`), (8, 11, `relation`), (12, 16, `attribute`), (21, 28, `attribute`)]}. The output tags are assigned to corresponding token.

## 4.4 Component Feature Extraction Module

Extracting various specifications that explain component features from the text input is important when it comes to extracting information between entities. This module focuses on finding relevant information about found components which have been generated by ML module. The process line in this module seen in Figure 4.11 as:

1. Specification Resolver
2. Component Tagging



**Figure 4.11** Component Feature Extraction Module

Hence, the proposed work assumes that the found tags are correct, therefore focusing on individual tokens highlights the features of corresponding component.

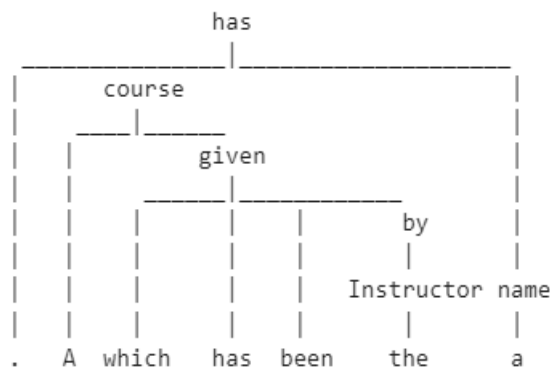
### 4.4.1 Specification Resolver

Train of custom NER shall find components much better than initial component extraction patterns from dependencies. This step is responsible to detect component specifications by iterating in dependency sub-tree and looking for definitive tokens, then links them in meaningful way. The first thing what it does is to sense sentence structure if a sentence is explaining about a relation or explanation of a component. For example:

1. *The student has name, id, email as attributes.*
2. *Email attribute is unique attribute.*
3. *A Student can enroll many Courses.*
4. *The relation “enroll” is storing enrolled\_date.*
5. *A Course which has been given by Instructor has a name.*
6. *Enroll is identifier relation.*



The first sentence explains attributes of the entity “Student.” On the other hand, the second sentence gives a clue about the type of attribute which “Email” is unique attribute. The third sentence is identifying relation and its specifications next to possible entity which is “Course.” The fourth sentence highlights the existing attribute of a relation “enroll.” Figure 4.12 illustrates dependency tree of sentence 5. Dependency Tree and NER tags help over here to sense which type of sentence occurs to be investigated. Investigation of each subtree highlights the sensing feature of tagged component. Every subtree of this step is considered separately.



**Figure 4.12** Illustration of dependency tree of 5<sup>th</sup> sentence

The output of custom NER contains if a token is “entity,” “attribute,” or “relation.” If the root token labeled as “relation” contains the words like "has, contain, include, store", it highlights the relation between entity and attribute or another entity. Next tokens which have found and labeled with “Attribute” or “Entity” to investigate the sentence. Those tokens in subtree shall present their specifications; therefore, searching within dependencies that modifies labeled tokens from custom NER output gives idea about that annotated token.

If a token type is “attribute,” look for modifiers like “identifier, unique, different, primary, key, only one” to tag as primary key candidate. The modifiers like “calculated, derived” highlights attribute is derived attribute. If there are no indicators, the system looks-up the indicated entities from generated dataset from documents. Moreover, if there are no indicated ones in dataset, the attribute component considered as normal type. If “has-a” relation contains the entity component on the right sub-tree, indication of the token “from” had been noticed whether that attribute contains by the entity which has dependency with. By looking for these modifiers, attribute specs had

been revealed. In addition to that, if labeled tag is “entity,” therefore the relation is a “has-a” with another entity. The words that have dependency which modify the token with “attribute” label, is one of the terms like “multiple,” “more than one,” or “multivalued,” that attribute token had been validated as multivalued attribute. If the attribute does not have any indicator mentioned in document, look-up table provided by pre-defined dataset had been taken into consideration to define tag of attribute.

On the relation side, the root verb of each sub-tree is the one who is in charge. If the root word is an active word that represents action, the case considered for dependency of tokens labeled as “Entity.” In this case, investigation of subject and object entity conducted to find cardinality. By using POS tags, plurality and singularity of a noun token had help here to identify cardinality. In addition to that, the corresponding sub-tree might contain the kind of modifiers that might represent cardinalities. Therefore, modifiers of “relation” token had been investigated and looked for the words like “many”, ”multiple”, ”more than <num>,” where <num> is a numerical text, next to the POS tag plurality. Moreover, even if there are no adjectives, the plurality form of entity represents the cardinality. If an entity is in plural form, it may indicate that the relationship is many. By looking at those words, pluralities have been revealed the cardinality like “one-to-many,” “many-to-many” or “one-to-one.” Moreover, the sentence might define relation specifications directly as well, like the sentence tells a feature about relation as if the relation is an identifier.

After finding out entities, attributes, and relation specification, look for any update that might occur during the second iteration of document. Some other features explored during the iteration process in sentence. For example, the 6th sentence gives a specification about the relation. The pre-data generated by Specification Resolver step, had been investigated and compared with overall document. In this case the function looks for if a sentence is describing about an entity, relation, or attribute specification to satisfy consistency between found components.

After generating component data for graph, algorithm traversed sentences once more to update component features if necessary. In this case the function looked for if a sentence is describing an entity, relation, or attribute feature. If a sentence indicates relation, by the descriptive sentence, this process updates component tag. If a sentence is indicating relation, system looked for as if its description is the words like “identifier,” “descriptive.”

The subject of descriptive sentence is investigated on NER output to link with component which has been found by custom NER and specification resolver. In addition to that, the subject of descriptive sentence may be indicating attribute or even entity. Attribute components found in part of the entity, is updated if there are any specific indication about their specification like “derived,” “foreign,” “primary,” or “multivalued.” Updating those attributes might even cause the identifier relation, along with weak entity. In entity side, the key to weak entity is considered as foreign key to the relation or entity which has linked to. Moreover, after the update of attributes, the entity specification might be change, like weak or normal.

#### 4.4.2 Component Tagging

This process is responsible to tag components according to its specification which is generated by specification resolver process. Hence, the main goal of our study is to extract information about relations between components, the illustration of components is up to application preferences like the type of ERD generation as Chen, or Foot style ERDs. The enumeration has been applied to identify component shapes that satisfies GraphViz functions. Entity representations have tagged with “0” for rectangular and “1” for double rectangular entity components as explained in overview part. Attributes has enumerated from 1 to 4; 1 for key/unique attributes as underlined oval, 2 for multivalued attributes as double oval, 3 for normal attributes as oval, and 4 for derived attributes as dashed oval shape. The tagging for relations is enumerated as “0” for diamond, and “1” for double diamond shape to indicate whether the relation is identifier relationship. Cardinalities also have an enumeration, 1 for one to one, 2 for one to many, 3 for many to one, and 4 for many to many cardinalities.

After tagging each component the data has become 2 different structures, the first one which represents relationships between entities and for relations of entity and attributes. Processed list data structure is constructed as Equation 4,2 and 4,3:

$$\begin{aligned}
 \textit{Entity - Entity}: [ < \textit{source entity name} >, & (< \textit{relation name} >, \\
 & \textit{enum}(\textit{relation type}): \textit{int}, \textit{enum}(\textit{cardinality}) : \textit{int}), \quad (4,2) \\
 & < \textit{target entity name} >],
 \end{aligned}$$

$$\begin{aligned}
 \textit{Entity - Attribute}: (< \textit{entity name} >, \textit{enum}(\textit{entity type})) , [( \\
 & < \textit{attribute name} >, \textit{enum}(\textit{attribute type})] \quad (4,3)
 \end{aligned}$$

**Table 4.4** Sample of Component Relation Structure

Document	Customer has name, passport_id, address, and age. Country has Airport entity. Address is a multivalued attribute. Country has name, and c_id. ... .. Flight_id is derived attribute. Purchase contains purch_date.
Entity - Entity	[('Country', (has, 0, 1), 'Airport'), ('Airport', (departs, 0, 2), 'Flight'), ... .. ('Airport', (stores, 0, 2), 'Airplane'), ('Airplane', (reserves, 0, 2), 'Ticket'),]
Entity - Attribute	[((Customer, 0), [(name, 3), (passport_id, 1), ('address_drv', 4), (age, 3), (address,2)]), ... .. ((Country, 0), [(name, 3), (c_id, 1)]), ((Airport, 0), [(airportName, 3), (countryId, 1), (ProvinceID, 1), (airportId, 1)],)]

The term “enum” stands for enumeration which was described above. The instance of processed data is illustrated in Table 4.4.

#### 4.4.3 Reduction of Redundant Information

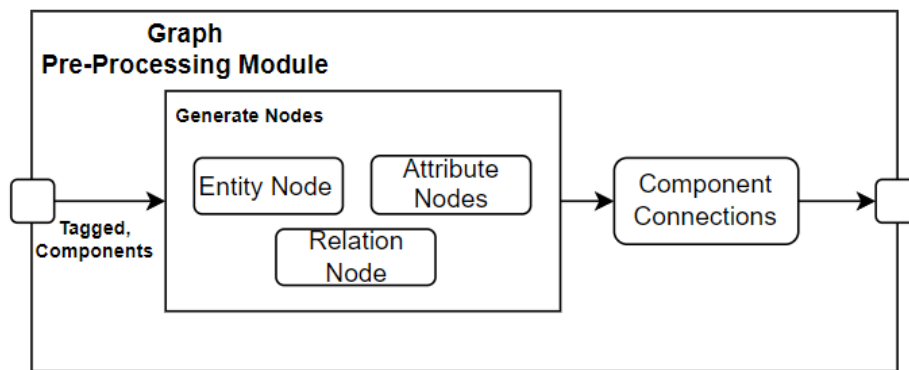
Reduction of redundant components is a challenging task to obtain. Utilization of word net and word similarity had been investigated which helps to reduce duplication of annotated data. If there are synonymous attributes, entities present, with similar POS tags which are related to the same entity shall be eliminated to reduce redundancy. For example, the sentence:

"*Lecturer* has name, surname. *Teacher* contains the address and number." It might be known that the “*Lecturer*” and “*Teacher*” refer to the same entity. To reduce redundancy, this module basically looks for synonyms for each labeled token. If annotated token uses same POS tag and, in the word-net synonym list, it is high probable that those nouns lemma refer to same component. However, sometimes merging those components leads to information loss, therefore the overall context of documentation is important to consider about. Looking for the component cases which use the same POS tags, and labeled components, creates information loss. For example,

one of synonyms of the word “car” is “truck.” Unfortunately, the usage of synonyms might cause information loss when the description contains both terms together because they are indicating different entities. Therefore, morphosemantic analysis is required to reduce redundancy. The proposed work also tried to use word similarity within sentences, but this also created information loss in the documents which contain similar meaning words. For example, the word “student” brings “grad student” as highly similar entity, which is correct but in context, they might be actor of different entities as well. As a conclusion, our study could not fully reduce the redundant information across the documentation.

#### 4.5 Graph Pre – Processing

The final generation of ERDs are tailored by the library called GraphViz. This library provides nodes to create graphs with several types of shapes and connections. To accomplish this task. The output of component feature extraction module had been used to separately generate graph components. The overall structure of this module is illustrated in Figure 4.13.



**Figure 4.13** Graph Pre-Processing Module

Each component is generated according to the structure of entity-entity and entity-attribute data structure. The first step is to generate entities, and attributes. The entity-attribute structure had been traveled and each entity node has been created according to their enumerated shape type. After those attributes had been created with corresponding shape, each entity had been linked to its attributes. As same method had been applied to entity-entity structure to create relationship node with its shape, as well

as introduced in overview of ERD. After the generation of the node the entities are linked to each other with their cardinalities. The last step is to render pictures from graph information which GraphViz framework had generated by constructed node and connections.

## **CHAPTER 5**

### **5. RESULTS**

The following section presents a thorough analysis of the obtained results from a series of features used in experiments and case examples of best network, along with an in-depth discussion of the identified limitations and issues encountered during the implementation of the system. This examination aims to provide a comprehensive evaluation of the system's performance and offer insights into areas where the generated outputs may deviate from the desired outcomes. By critically examining these outputs, valuable insights will be gained into the weaknesses of the system and potential gaps for improvement. Through a detailed exploration of the case examples, this section aims to dig into the underlying factors contributing to the identified issues with meaningful discussions on their implications and proposing recommendations for future enhancements.

#### **5.1 Experimental Results of Custom Named Entity Recognition Module**

To develop an efficient NER system, our study explored various attribute features to find the optimal model. The results of our study are experimented with multiple attributes, including POS tags, dependency labels, entity labels, and more. Each attribute provides a unique perspective on the sentence structure. By incorporating these diverse attributes, our study aimed to capture a comprehensive representation of the text, facilitating the extraction of components such as entities, attributes, and relations. Through testing and evaluation, has analyzed the performance of different attribute combinations, seeking to identify the attributes that contribute most effectively to the NER task.

Considering that collected 105 documents with approximately 3200 words for training is relatively small. In such cases, smaller batch sizes are commonly used to ensure that the model can learn from a diverse set of examples within each batch. A typical batch size for a small dataset could range from 8 to 32. Our study experimented with batch sizes with 8 and observed the performance and utilization during training.

Regarding the number of epochs, it represents the number of times the custom NER model will iterate over the entire training dataset. Hence, there are no fixed rules, but general practice is to start with smaller number of epochs, such proposed model has been tried and monitored model’s learning progress between 10 epochs with the 0.001 learning rate. The learning curve starts to converge, and model gets overfit after 10 epochs. The pipeline consists of vectorization, multi-hash embedding with related features, bi-directional LSTM encoder, transition-based parser, Tok2Vec Listener, and named entity recognizer. Table 5.1 provides experimental results of several feature sets with evaluation metrics. The Tok2Vec technique helps in transforming the discrete tokens of a sentence into continuous vector representation, which are then fed into the NER model for prediction. Therefore, by analyzing the evaluation metrics would give a preview about which feature is feasible to meaningful representation.

**Table 5.1** Bi-Directional Long Short-Term Memory Network Feature Comparison for Custom Named Entity Recognition

Model	Features	Loss Tok2Vec	Loss NER	F1 Score - NER	Precision - NER	Recall - NER
1	{SPCY} <sup>1</sup>	474.84	349.70	71.85	70.27	73.51
2	{SPCY}, "POS"	180.67	611.14	80.74	76.45	85.54
3	{SPCY}, "DEP"	796.38	190.62	89.63	90.58	88.69
4	{SPCY}, "POS", "DEP"	<b>389.28</b>	<b>189.98</b>	<b>93.87</b>	<b>92.96</b>	<b>94.81</b>
5	{SPCY}, "POS", "TAG", "DEP", "ENT_TYPE", "ENT_IOB"	423.58	257.57	92.17	91.53	92.82

<sup>1</sup> {SPCY}: Consist of base Spacy attributes as follows: "NORM", "PREFIX", "SUFFIX", "SHAPE"



Spacy base attributes are annotated as {SPCY}, which has features like "NORM," "PREFIX," "SUFFIX," "SHAPE," where "NORM" stands for normalized form of the token.

Model 2 incorporates additional POS tags alongside the default attributes of Spacy. This integration resulted in improved performance compared to the base Spacy custom NER module. However, Model 2 encountered challenges in detecting attributes that contained special characters. Furthermore, there were instances where attributes were incorrectly labeled as "Entity." This issue primarily arose from attributes with a "NOUN" tag, which exhibited similar behavior to entity properties. Moreover, descriptive sentences which are related to attributes are sometimes considered as entity.

The inclusion of dependency labels proved to be more effective in comparison to model 1 and 2. By incorporating dependency labels into the base Spacy features, the projected outcome was an improved identification of entities and attributes, when compared to the model obtained POS tags. Model 4, which integrated both POS tags, and dependency labels alongside the base Spacy features achieved the best overall results among the assessment procedure. Model 5 comprised all of the characteristic features of model 4, as well as Spacy's pre-trained NER module output, which included standard NER tags like "MONEY", "LOCATION, as well as tags with IOB tags. Unfortunately, the addition of these features resulted in higher losses and worse F1 score than Model 4. As a result, model 4 was chosen to be used in the evaluation process. In the following section, the case model outputs will be examined Model 4.

## **5.2 Model Case Outputs**

The following subsection delves into a series of cases that examine diverse types of features within data relationships. Each case presents a unique scenario, shedding light on various aspects of data organization and connectivity to be examined and have further investigate output.

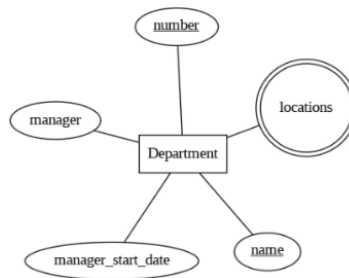
### **5.2.1 Case 1: "Attribute" of "Entity" Extraction**

In Case 1, our study focused on understanding the relationship between entities and attributes. Generated custom NER delve into how attributes are associated with specific entities and explore the implications of these connections. By examining this

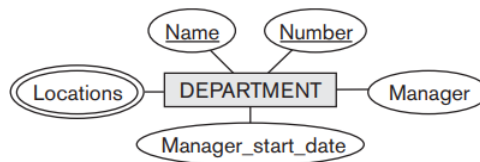
fundamental aspect of data relationships, it gains insights into how attributes define and characterize entities.

**Scenario 1:**

*Department contains name, number, locations, manager, and manager\_start\_date. Locations is the only multivalued attribute. Name and number are key attributes. (Elmasri and Navathe 2016)*



**Figure 5.1** Generated Entity Relationship Diagram of Scenario 1



**Figure 5.2** Illustration of Elmasri on Scenario 1 Reference: Elmasri, R., & Navathe, S. B. (2016). *Fundamentals of Database Systems*.

Attribute and Entity extraction scenarios which describe entity, attribute specifications are usually captured correctly. The aid of custom NER is not tagging the attribute “Locations” as” Entity” but the “Attribute.” Since it checks further tokens and sentences it is capable of tagging correctly of each subject token of sentence. Unfortunately, the specification resolver module is lacking in detecting compound attributes.

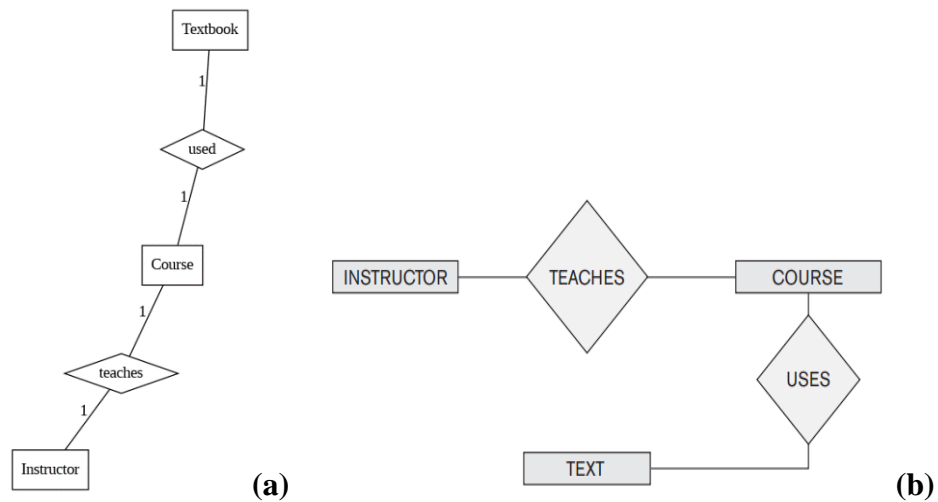
**5.2.2 Case 2: Relation Extraction**

Case 2 shifts attention to the relationships between different entities. Our study explores how entities interact with each other, the dependencies that exist, and the significance of these connections. By analyzing the connections between entities, the

results of this study indicate and gain a deeper understanding of how data is linked across different entities.

**Scenario 2:**

*Instructor teaches Course. Textbook which is used by Course.*

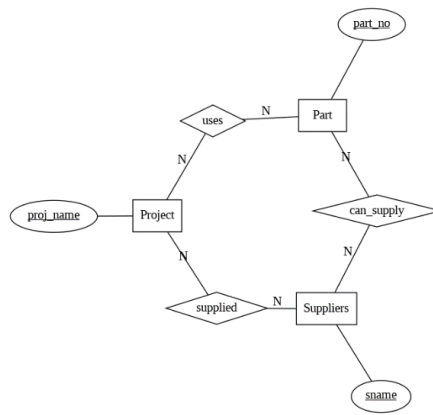


**Figure 5.3** Generated Entity Relationship Diagram of Scenario 2 (a), ERD Illustration (b) Reference: Elmasri, R., & Navathe, S. B. (2016). *Fundamentals of Database Systems*.

Figure 5.3 illustrates output of scenario 2 which is solution and captured components. The relationship between entities generally is captured correctly. Hence the relationship is represented by each branch of dependency tree generally a verb phrase and indicates root word corresponding sub-tree.

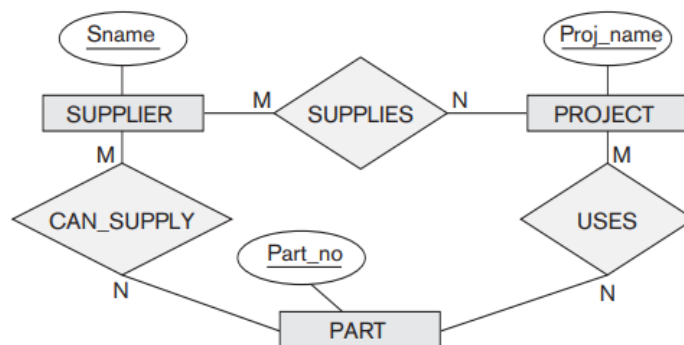
**Scenario 3:**

*Suppliers have sname. Project has proj\_name. Part has part\_no. Many Projects can supplied by many Suppliers. Many Projects uses many Parts. Many Suppliers can\_supply many Parts. sname is primary attribute. proj\_name is primary attribute. part\_no is primary attribute.*



**Figure 5.4** Generated Entity Relationship Diagram of Scenario 3

Figure 5.5 is an illustration of scenario 3 which is generated by the author. The generated ERD (Figure 5.4) is quite similar and same as the generated output from the model which has been proposed. Based on the experimental result, it was found that the information extraction module successfully managed and accurately labeled the components in scenario 3. This means that the module was able to identify and assign correct labels to the various elements such as “Entity,” “Attribute,” and “Relation” on simple structured sentences.



**Figure 5.5** Elmasri Illustration on Scenario 3 Reference: Elmasri, R., & Navathe, S. B. (2016). *Fundamentals of Database Systems*.

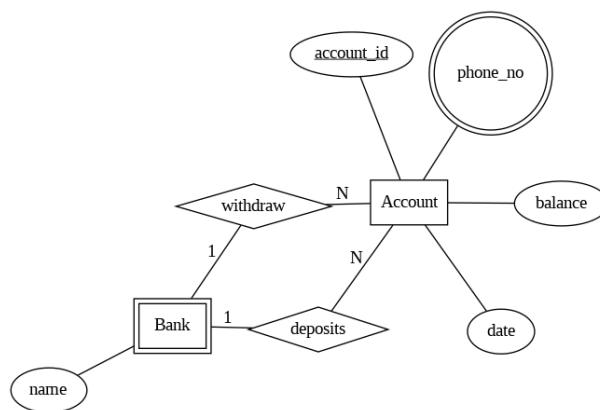
Moreover, these structural sentences could be extracted using heuristic approaches which is accepted as current literature. This suggests that by implementing certain rules or patterns, the system was able to extract meaningful sentences that describe the relationships and connections between different items in the diagram with the aid of custom NER.

The dependency tree of the input sentence played a crucial role in analysis. It provided valuable insights and aided in identifying the dependencies and associations between the different components. By examining the dependency tree, our study was able to uncover the relationships and connections between items, allowing for a more comprehensive understanding of the diagram's structure.

Overall, the combination of an effective information extraction module and the utilization of heuristic approaches and dependency trees greatly contributed to accurately describing the diagram and generating meaningful outputs.

**Scenario 4:**

*Many Accounts deposits from the Bank. Many Accounts withdraw from the Bank. Account has balance, phone\_no, date and account\_id. Bank has name. Bank is a weak entity. Phone\_no is a multivalued attribute.*

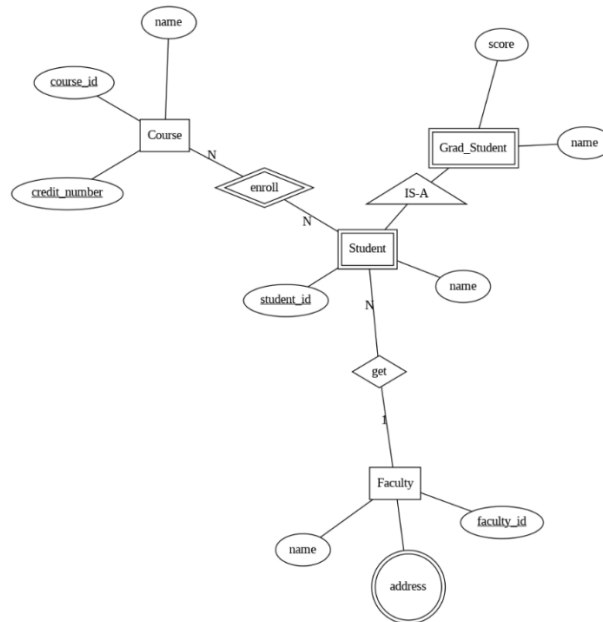


**Figure 5.6** Generated Entity Relationship Diagram of Scenario 4

The system also captures the weak entities if it has been declared in input sentences. If the existence of an entity is related to another entity, those entities are considered as weak entities. The system looks for specific identifying specification for attributes. If there are no specifications of key attributes, the system investigates his dataset which has been created while tagging the previous documents and tags attributes. For example, in Scenario 4, account\_id has not been declared as primary key. Although there are no indicators, the system proposes and labels the attribute “account\_id” as primary key (Figure 5.6).

## Scenario 5:

*Student has name, and student\_id. The course has name, course\_id, and credit\_number. The students enroll in many courses. Student\_id is a unique attribute. Students are weak entity. Enroll is identifier. Faculty has name, faculty\_id, and multivalued address. A Faculty get many Students. Grad\_Student has name, and score. Grad\_Student is a student.*



**Figure 5.7** Generated Entity Relationship Diagram of Scenario 5

Figure 5.7 describes the structure of basic course enrollment -i.e., Scenario 5-. As you can see the “IS-A” relationship had been captured. In addition to that, the feature of parent entity which is “Weak Entity” also embedded into child “Entity.” Moreover, if there are no key attributes of “entity,” it does consider that entity as weak because the entity does not have uniquely identified by its own attributes alone. Therefore, the output of the system considers and labels “entity” without any key attribute as “weak entity” as well.

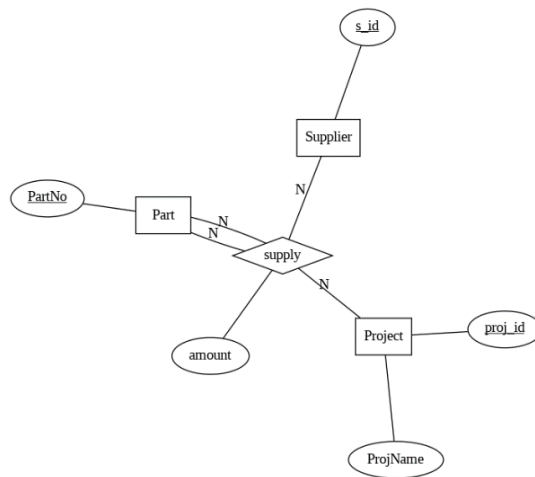
### 5.2.3 Case 3: "Attribute" of "Relation" Extraction

Case 3 investigates the relationship between relations and attributes. System explores how attributes contribute to the definition and organization of relations. By

examining this connection, the proposed model gains insights into how attributes shape the structure and interpretation of relations within a dataset.

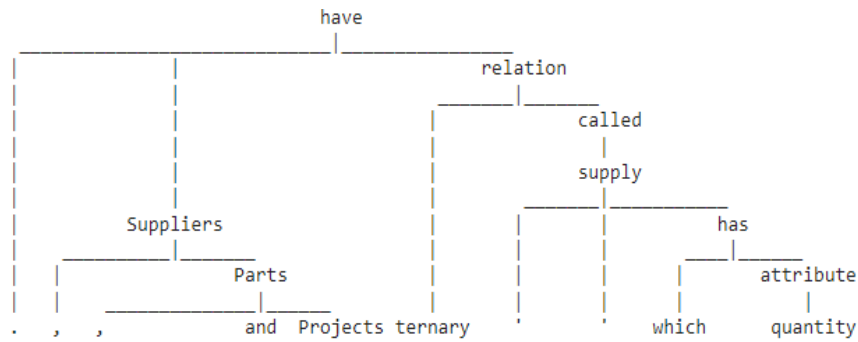
**Scenario 6:**

*Project has proj\_id, and ProjName. s\_id is unique attribute which has been stored by Supplier. Part entity stores PartNo as attribute. Suppliers, Parts, and Projects have ternary relation called 'supply' which has quantity attribute. Many Suppliers supply many Parts to many Projects.*



**Figure 5.8** Generated Entity Relationship Diagram of Scenario 6

As we can observe from Figure 5.8, the attribute of relation “amount” had been captured by the specification resolver module. The sentence: “Suppliers, Parts, and Projects have ternary relation called 'supply' which has quantity attribute.,” contains significant information about relation. The custom NER finds and tags tokens as entity, attribute, and relation. Although it finds relation and cardinalities, readers might get confused on relationship. The entity “Part” uses the relation “supply” component to have relationship between other entities “Supplier” and “Project.” However, there are no indications that which association is linked to which entity. It creates problems with understanding cardinalities. Possible solution to this problem is to change color of associations which is referring to or, create another relationship called “supply” but declaring another relationship means that another relationship needs to be built on physical design of database which is incorrect.



**Figure 5.9** Dependency Tree Illustration of sentence: “Suppliers, Parts, and Projects have ternary relation called ‘supply’ which has quantity attribute.”

In this scenario, entities are “Part,” “Suppliers,” “Projects,” attribute tagged by NER is “amount,” and relation is “supply.” Figure 5.9 is an illustration of dependency tree of the sentence. In dependency tree, the subjects of sentence are “Suppliers,” “Parts,” “Projects” which had been labeled as “entity,” on right hand side the relationship between those entities are described with the word “supply” which had been labeled as “Relation” by NER. Therefore, the direct relationship between those entities had been captured and created with one-to-one cardinality. Then specification resolver is considered plurality of tokens then alters the cardinality from “one-to-one” to “many-to-many.”

The token labeled as “relation” which is “supply” also had its own sub-tree. Thus, these sub-trees are investigated separately, and the token labeled as “attribute” is linked to its root token which is “supply.” The same algorithm runs for this sub-tree therefore, it is possible to capture attribute of relation on top of the overall sentence.

Custom NER brings out the component information that helps to get rid of to define different heuristic rules to detect entities, attributes, and relationships from the sentences, by having information about overall documentation. While some of the sentences are directly indicating corresponding components specification of entity, attribute, or relation, heuristic approaches have to be tailored specifically to identify described specification of component.

#### 5.2.4 Case 4: Complex Relations

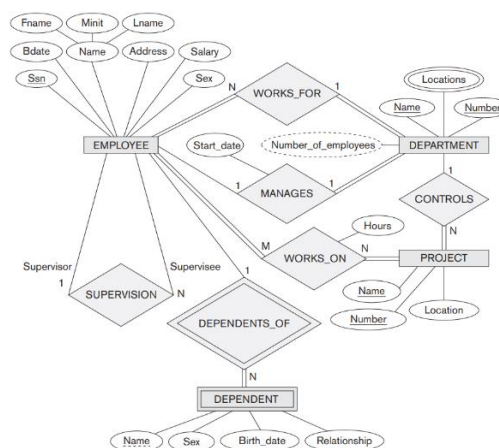
Case 4 focuses on complex relations, which involve intricate data relationships with multiple entities, attributes, and possibly nested structures. The proposed study



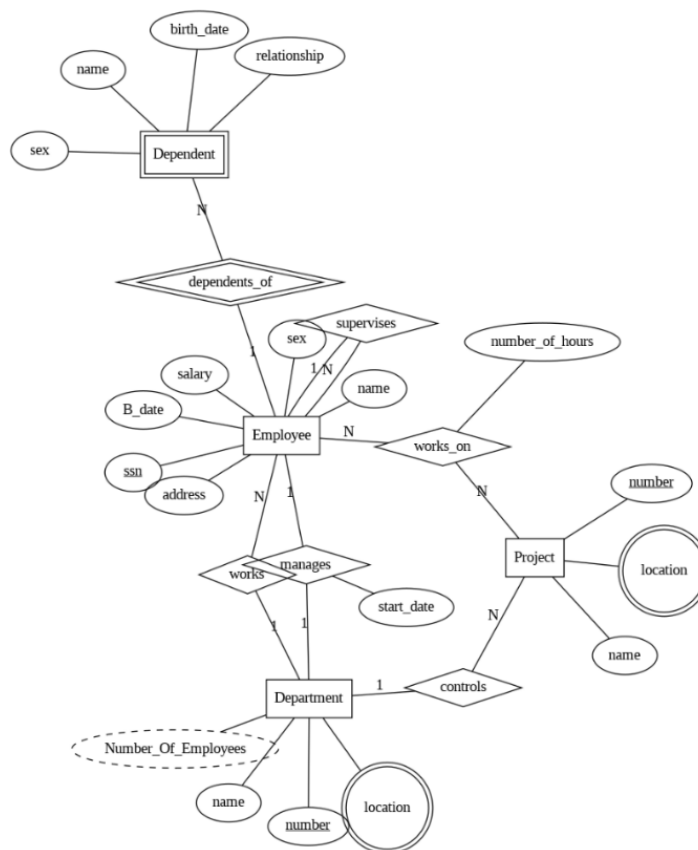
explored the challenges and opportunities presented by these complex connections. By investigating complex relations, we gained a comprehensive understanding of how data can be organized and analyzed in more intricate scenarios. By exploring these different cases, the system gains valuable insights into the diverse types of features that exist within data relationships.

### Scenario 7:

*Employee* contains name, address, salary, sex, B\_date, ssn. Ssn is a primary attribute. The name attribute is creation of 3 different components, such as Fname, Minit, and Lname. Each *Department* has a name, unique number, location, and Number\_Of\_Employees and particular one *Employee* who manages the department. *Department* may have several locations. Number\_of\_employees is a weak attribute. Location is a multivalued attribute. The start\_date is recorded in manages relation. Many *Employees* works one *Department*. Both participants are total. An *Employee* can supervises many *Employees*. A *Department* controls many numbers of *Projects* which many employees can work on. *Project* has name, number, and location. *Projects* has unique number. Work on relation stores number\_of\_hours attribute. Both participants are determined to be total. An *Employee* depends of many *Dependent* entity. *Dependents* of is identifier relation for the *Dependent* entity which is a weak entity *Dependent*. The participation of *Employee* is partial, whereas that of *dependent* is total. *Dependent* entity contains sex, birth\_date, relationship, and name. (Elmasri and Navathe 2016)



**Figure 5.10** Solution Entity Relationship Diagram of Scenario 7 Reference: Elmasri, R., & Navathe, S. B. (2016). *Fundamentals of Database Systems*.



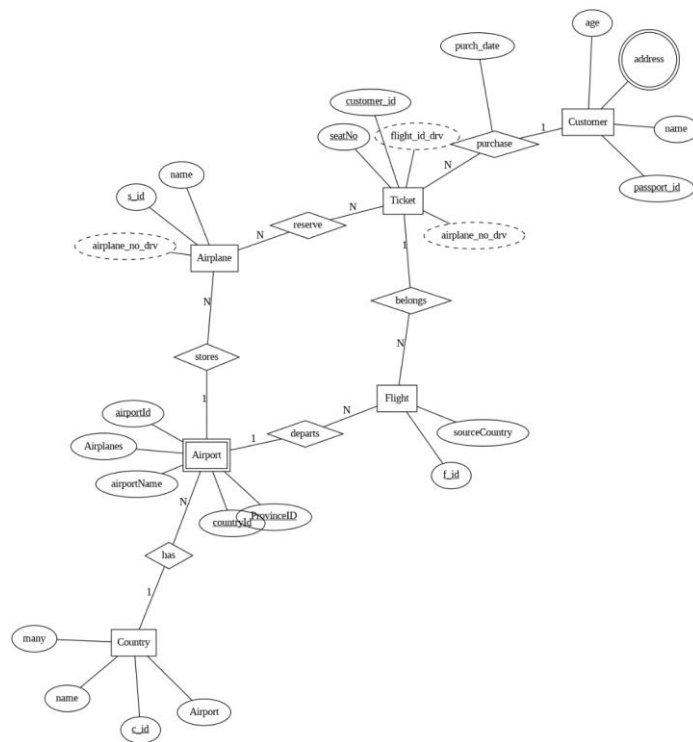
**Figure 5.11** Generated Entity Relationship Diagram of Scenario 7

Investigation of this scenario 7 as the author (Elmasri and Navathe 2016) says participation of entity “Department” is not clearly understandable from the requirement. Scenario questions the users, who say that a “department” must always have a manager, which implies total participation. Therefore, domain-knowledge is required to assign these kinds of relations.

Another problem in the output is that the pronouns which had been used in scenario shall be linked to the “Department” and “Employee.” (Figure 5.11) Hence there are no proper entity in that sentence. There could be way to utilize this issue, one is to replace the phrase “both participants” with corresponding entities shall be linked to phrase with nearest pronoun method. Even though, extracting information like entities, attributes, and relations are correct. However, the attribute “Location” is annotated as multivalued attribute for both entities which is “Department” and “Project.” The problem over here is caused by not linking corresponding component at nearest entity.

**Scenario 8:**

*Customer has name, passport\_id, address, and age. Country has many Airports. Address is a multivalued attribute. Country has name, and c\_id. Airport has airportName, countryId, ProvinceID, airportId. Flight is an entity which stores f\_id, sourceCountry, destCountry, numberOfCustomer. Each Airport departs many Flights. Airplane has name, airplane\_no, and s\_id. Ticket contains customer\_id, seatNo, airplane\_no, flight\_id. An airport stores many Airplanes. Each Airplane may have many Tickets. Airplanes reserve seats for many Tickets. A Ticket belongs to many Flights. Flight\_id is derived attribute. Purch\_date has been stored in purchase relation.*



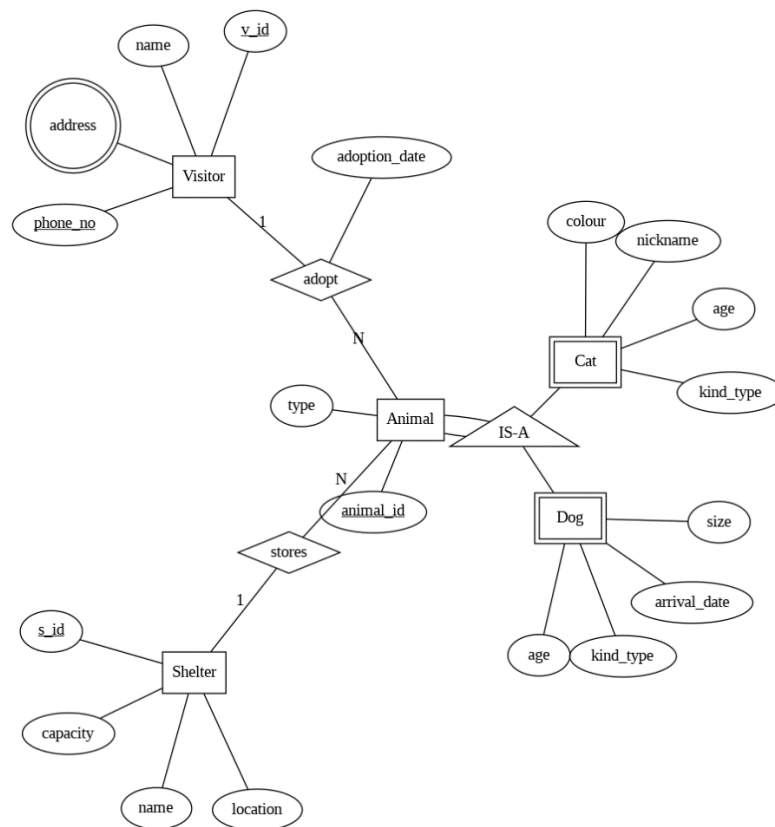
**Figure 5.12** Generated Entity Relationship Diagram of Scenario 8

In scenario 8, the problem occurred because of wrong labeling caused by custom NER. The problem that arises from the input text is related to incorrect or abbreviated terminology. For example, instead of using "destination country," the text uses an abbreviated form "destCountry" (Figure 5.12). This presents a challenge to the system because the abbreviated or incorrect terminology can lead to confusion or misinterpretation of the intended meaning and syntactic analysis. It may cause

difficulties in understanding the context or correctly identifying the entities or relationships being referred to. Moreover, it affects the POS tagging and dependency tree so the incorrect annotation from the custom NER.

**Scenario 9:**

*Shelter stores Animals. Shelter has location, capacity, s\_id and name. Animal contains type, and animal\_id. Dog has arrival\_date, kind\_type, age, and size. Cat includes nickname, kind\_type, age, and colour. The Dog and Cat is Animal. Visitors contain the name, phone\_no, multiple address, and v\_id. V\_id and phone\_no are key attributes. A Visitor can adopt many Animals. Adopt relation contains adoption\_date.*



**Figure 5.13** Generated Entity Relationship Diagram of Scenario 9

Overall, in this scenario, there is no inconsistency in output but the relation between animal and its related sub-entities. The same problem occurs on entities which use the same relation. Relationship token might be declared separately to increase readability. Since the relationship indicates “IS-A,” it is still understandable but the relationships which require cardinality might create confusion in the reader’s mind.

### 5.3 Evaluation

In this section, our study assesses the performance and reliability of a model designed to extract entity-relationship (ER) diagram components from unstructured text. Through the analysis of 25 diverse text and component features collected from database books with their problem text and ground-truth labels are collected from solution of ERD figure from the book. Overall, total of 1329 words, evaluated with the 4<sup>th</sup> from the Table 5.1 accuracy, precision, recall. The results validate the model's effectiveness, providing insights into its strengths and limitations of ER diagram extraction from text, enabling improved efficiency and automation in database design and analysis.

### 5.4 Confusion Matrices

To evaluate in the context of extracting ERD components from text, True Negatives (TN) represents cases where the model correctly predicts the absence of ERD component. However, in the task of extracting specific components (entities, attributes, and relations) from the text, the absence of these components is generally assumed unless they are explicitly mentioned. Therefore, the focus of evaluation shall be on True Positives (TP) which are correctly extracted components, False Positives (FP) which are incorrectly identified components.

**Table 5.2** Confusion Matrix of Entity Extraction

		Gold Standard	
		TRUE	FALSE
Prediction	TRUE	99	6
	FALSE	2	X

Table 5.2 provides an overview of entity identification results within the evaluation dataset. The evaluation achieved a recall score of approximately 0.98 and a

precision score of around 0.94, yielding an F1 score of 0.96. Our study conducted a thorough investigation into the specifications of the identified components, distinguishing between normal entities and weak entities. The inaccurate predictions primarily stemmed from the Custom NER output, responsible for component labelling. Conversely, the specification resolver module received this output and assigned specification tags to annotated tokens. Among the entities correctly identified by the Custom NER module, the specification resolver module accurately labeled 76 entities as normal and 17 entities as weak, while six of the normal entity were mislabeled as weak entity. Since the investigation of across the correctly found entity, FP is 0 for normal entity, and TN is 0 for weak entities. In terms of evaluation, the recall for tagging normal entities was approximately 0.92, and the precision for labeling weak entities was 0.73.

**Table 5.3** Confusion Matrix of Attribute Extraction

		Gold Standard	
		TRUE	FALSE
Prediction	TRUE	289	11
	FALSE	26	X

In terms of annotating attributes, the confusion matrix, presented in Table 5.3, provides an overview. The model accurately predicts 289 attribute components within the evaluation set. However, certain outputs such as abbreviations and words containing special characters (#, &, \$, -, ...) posed challenges for the custom NER module to resolve. Moreover, there were instances where entities were incorrectly labeled as attributes within certain contexts. The model achieved a recall of approximately 0.91 and a precision of 0.96, resulting in an F1 score of 0.93 for attribute extraction. To evaluate the specification resolver's performance in identifying attribute specifications from correctly classified attributes, the following analysis focuses on finding key attributes.

**Table 5.4** Confusion Matrix of Key Attribute Extraction

		<b>Gold Standard</b>	
		TRUE	FALSE
<b>Prediction</b>	TRUE	83	3
	FALSE	10	X

According to the data presented in Table 5.4, the specification resolver module successfully extracted 83 key attributes correctly, but 10 components were identified as false positives. However, there were 3 attributes wrongly predicted as key attributes. The recall for resolving key attributes was measured at 0.89, while the precision achieved was 0.96. The resolver module identified 6 attributes as derived and 4 as multivalued, out of a total of 8 derived and 4 multivalued attributes, with 2 incorrect identifications of derived attributes. Unfortunately, there was limited evaluation for specifications such as multivalued in the evaluation set. Overall, most of the incorrect resolving occurred due to undefined key attributes and need for domain specific knowledge. As the description lacks information on primary attributes, the module resorts to using its own dataset to label attributes for entities without specified key attributes.

**Table 5.5** Confusion Matrix of Relation Extraction

		<b>Gold Standard</b>	
		TRUE	FALSE
<b>Prediction</b>	TRUE	77	4
	FALSE	8	X

Table 5.5 presents the annotations of relation components across the evaluation dataset. The precision and recall scores for relation prediction are 0.90 and 0.95, respectively, resulting in an F1 score of 0.92. The investigation of relation extraction was prompted by issues with the incorrect extraction of dependency trees. These problems arose from relation words containing special characters and phrases, which led the model to make incorrect label predictions. Within the specification resolver, 4 relations were correctly identified as identifiers, although 1 of them was not actually an identifier, out of a total of 5 identifier relationships. Overall, the model accurately identified 69 cardinalities, while 8 of them were incorrect. As a result, the module achieved a recall of 0.89 for extracting cardinalities among the correctly defined relations.



## CHAPTER 6

### 6. CONCLUSION AND DISCUSSION

The aim of our study was to develop a solution for extracting information and revealing relation between entity to generate ERDs from text input. The proposed solution demonstrated promising results in detecting entities, attributes, and relations. The f1-scores of detecting entities is 0.96, while for attribute and relation are 0.93 and 0.92, respectively. Unlike many previous studies that often overlooked crucial information such as entity type, attribute type, and identifying relations, the current solution aimed to retrieve these essential components. However, we encountered several limitations, which are discussed below along with proposed solutions for future improvement.

The first limitation observed in our study was the issue of redundant information. Although the proposed solution showed effectiveness in some cases, there were still limitations in handling certain instances. For example, when presented with the statement "Teacher has name. Teacher gives lecture. A Lecturer can give many Courses," the solution recognize that "Lecturer" refers to the same entity as "Teacher." The proposed solution attempted to resolve this problem by considering synonyms of found entities with the same POS tags. However, there were cases, such as "Game has Player and Bat (Animal). Players can have a bat as a weapon. Bat has name," where the model struggled to determine which instance of "Bat" the attribute "name" referred to. To overcome this limitation, we proposed looking for contextual information and incorporating word sense similarity between identified entities. However, this approach resulted in other challenges, as highly similar entities like "student" and "grad\_student" caused information loss when one of them was removed. Consequently, there is still a need to further address the issue of redundant information. As a future

direction, the “SynoExtractor” pipeline might be applied to reduce redundant information which uses synonyms, word embeddings, lemmas, and POS tags to find actual synonyms with same sense in document.

The second limitation identified was the challenge of linking entities across different sentences. When component references were scattered across multiple sentences, it became difficult to establish the correspondence between tokens and accurately resolve the connections. For instance, the statement "Actress has name and age information. Those pieces of information are all multivalued. The entity contains phone numbers." posed difficulties in linking the attributes and entities mentioned in the first sentence. As a future direction linking entities to the nearest pronoun as a possible solution, but this approach was not consistently accurate. To mitigate this challenge, future research should explore more advanced techniques, such as leveraging contextual analysis or syntactic and discourse analysis, to establish reliable connections between entities across sentences. Moreover, the difficulty in identifying and linking tokens that refer and mean to the same component. For instance, the statement "Teacher gives Courses. The person who gives a lecture has a name." indicates that "Teacher" and "Person" referred to the same entity, but the proposed solution treated them as separate entities. To address this challenge, we tried leveraging semantic similarity measures to identify tokens with equivalent or similar meanings but failed to manage the cases described above. Future research could explore advanced NLP techniques, including deep learning models or transformer-based architectures, to improve the identification and linking of tokens with similar meanings.

The third limitation pertained to managing abbreviations of different components. For example, the statement "Each Flight has Flight\_id. F\_id is a unique attribute" posed challenges in correctly identifying and linking attributes. To resolve this problem, utilizing an ontology to map abbreviations to their corresponding full forms could be used. This approach would allow for accurate recognition and linking of attributes, even when different abbreviations were used. Future research could focus on automatically generating or updating domain-specific ontologies based on the textual input, along with incorporating ML techniques for abbreviation resolution and NER.

The fourth limitation identified was the potential generation of unrealistic ERDs that do not align with the logical interpretation. Model tries to satisfy input text even though the logic of the input text is not correct. As a suggestion, by analyzing patterns

and inconsistencies in the dataset, the recommendation system could be built and provide suggestions to the user when generating new diagrams. Incorporating web ontology and mining techniques can further enhance the suggestions by extracting relevant domain knowledge, might improve the logical coherence of the generated ERDs and minimize the occurrence of unrealistic representations.

In conclusion, the results of the proposed work demonstrate the effectiveness of the proposed solution in extracting information and generating ERDs from text input. However, several limitations were encountered, including handling redundant information, linking entities across sentences, dealing with abbreviations. Future directions for improvement include addressing these limitations by incorporating techniques such as contextual analysis, semantic similarity measures, total participation, ontology-based mapping, and recommendation systems. By refining the solution and exploring these avenues, finding relations between entities might lead the systems like generation of ERDs, can be enhanced to provide more accurate representations of the input text.

Furthermore, future research could also focus on evaluating the proposed solution on a larger and more diverse annotated dataset to validate its effectiveness and generalizability. The inclusion of user studies and feedback can provide valuable insights into the usability and practicality of the solution, aiding in its refinement and real-world applicability.

Overall, our study contributes to the field of ERD generation by identifying limitations and proposing potential solutions for further improvement. By addressing these challenges, diagram generation systems can become more robust, reliable, and user-friendly, supporting various software applications.

## REFERENCES

- Ahmed, M. A., Ahsan, I., Qamar, U., & Butt, W. H. (2021). A Novel Natural Language Processing Approach to Automatically Visualize Entity-Relationship Model from Initial Software Requirements. *2021 International Conference on Communication Technologies(ComTech)*. doi:10.1109/comtech52583.2021.9616949
- Bajpai, A. (2021). Recurrent Neural Network & LSTM: Deep Learning for NLP, Towards Data Science. *Medium*. Retrieved from <https://towardsdatascience.com/recurrent-neural-networks-and-natural-language-processing-73af640c2aa1>
- Chen, P. P. (1983). English Sentence Structure and Entity-Relationship Diagrams. *Information Sciences*, 29(2–3), 127–149. doi:10.1016/0020-0255(83)90014-2
- Elmasri, R., & Navathe, S. B. (2016). *Fundamentals of Database Systems*.
- Ghosh, S., & Bashar, R. (2018). Automated Generation of E-R Diagram from a Given Text in Natural Language. *International Conference on Machine Learning and Data Engineering (iCMLDE)*. doi:10.1109/icmlde.2018.00026
- Habib, M., Kasra. (2019). On the Automated Entity-Relationship and Schema Design by Natural Language Processing. *The International Journal of Engineering and Science (IJES)*, 8(11), 42–48. doi:10.9790/1813-0811034248
- Jiang, R., Banchs, R. E., & Li, H. (2016). Evaluating and Combining Name Entity Recognition Systems. *Proceedings of the Sixth Named Entity Workshop*. doi:10.18653/v1/w16-2703
- Karaa, W. B. A., Azzouz, Z. B., Singh, A., Dey, N., Ashour, A. S., & Ghazala, H. B. (2015). Automatic Builder of Class Diagram (ABCD): an Application of UML Generation from Functional Requirements. *Software - Practice and Experience*, 46(11), 1443–1458. doi:10.1002/spe.2384
- Kashmira, P. G., & Sumathipala, S. (2018). Generating Entity Relationship Diagram from Requirement Specification Based on NLP. *2018 3rd International Conference on Information Technology Research (ICITR)*. doi:10.1109/icitr.2018.8736146

- Kchaou, D., Bouassida, N., & Ben-Abdallah, H. (2017). UML Models Change Impact Analysis Using a Text Similarity Technique. *IET Software*, 11(1), 27–37. doi:10.1049/iet-sen.2015.0113
- Morgan, R., & Garigl, R. (1995). Natural Language Processing with LOLITA. *Endeavour*, 19(1), 11–15. doi:10.1016/0160-9327(95)98888-m
- Omar, N., Hanna, P., & Mc Kevitt, P. (2006). Semantic Analysis in the Automation of ER Modelling Through Natural Language Processing. *2006 International Conference on Computing & Informatics*. doi:10.1109/icoci.2006.5276559
- Spacy. (n.d.). *Spacy Usage Documentation*. Retrieved August 7, 2023, from <https://spacy.io/usage/linguistic-features>
- Spacy. (n.d.). *Spacy API Documentation*. Retrieved August 7, 2023, from <https://spacy.io/api/architectures#parser>
- S. Btoush, E., & M. Hammad, M. (2015). Generating ER diagrams from requirement specifications based on Natural Language Processing. *International Journal of Database Theory and Application*, 8(2), 61–70. doi:10.14257/ijtda.2015.8.2.07
- Utama, A. Z., & Jang, D. (2019). An Automatic Construction for Class Diagram from Problem Statement using Natural Language Processing. *Journal of Korea Multimedia Society*, 22(3), 386–394. doi:10.9717/kmms.2019.22.3.386
- Sagar, V. B. R. V., & Abirami, S. (2014). Conceptual modeling of natural language functional requirements. *Journal of Systems and Software*, 88, 25–41. doi:10.1016/j.jss.2013.08.036
- Zhong, Z., & Chen, D. (2021). A frustratingly easy approach for entity and relation extraction. *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. doi:10.18653/v1/2021.naacl-main.5

## RESUME

### Publications

Köprülü, Mertali, and M. Taner Eşkil. (2022). Analysis of Single Image Super Resolution Models. *International Conference on Electrical, Computer, Communications and Mechatronics Engineering*.  
doi:0.1109/iceccme55909.2022.9988599.