BSCS-F19-033

03-134162-065 Mohammad Abdullah

03-134162-084 Moiz Uddin

Automated Diagram Generator Using Natural Language Processing

In partial fulfilment of the requirements for the degree of

Bachelor of Science in Computer Science

Supervisor: Nadeem Sarwar

Department of Computer Sciences

Bahria University, Lahore Campus

July 2020

C e r t i f i c a t e

We accept the work contained in the report titled

"Automated Diagram Generator Using Natural Language Processing"

written by

Mohammad Abdullah

Moiz Uddin

as a confirmation to the required standard for the partial fulfilment of the degree of Bachelor of Science in Computer Science.

Approved by:

Supervisor: Nadeem Sarwar

_____ (Signature)

July 20, 2020

DECLARATION

We hereby declare that this project report is based on our original work except for citations and quotations which have been duly acknowledged. We also declare that it has not been previously and concurrently submitted for any other degree or award at Bahria University or other institutions.

Enrolment	Name	Signature
03-134162-065	Mohammad Abdullah	
03-134162-084	Moiz Uddin	

Date: July 20, 2020

Specially dedicated to
my beloved grandmother, mother, and father
(Mohammad Abdullah)
my beloved grandmother, mother, and father
(Moiz Uddin)

ACKNOWLEDGMENTS

We would like to thank everyone who had contributed to the successful completion of this project. We would like to express my/our gratitude to my research supervisor, Mr. Nadeem Sarwar for his invaluable advice, guidance, and his enormous patience throughout the development of the research.

Besides, We would also like to express my gratitude to our loving parents and friends who had helped and encouraged me.

Mohammad Abdullah

Moiz Uddin

Automated Diagram Generator using Natural Language Processing

ABSTRACT

In today's world, Natural Language Processing (NLP) plays an important role in fields such as business, education, sports, marketing, or anywhere that involves human activity. Diagrams are the most important process in information system design and software engineering. Designing is a part of five generic activities of software development. Software Engineer needs to generate software diagrams like Entity Relationship, use case, class diagrams. These diagrams help us to display the contents of the data. They also help us to visualize how the data of our system is connected and make relations among them. In this project, we develop an application by applying NLP techniques which is helpful to generate a software diagram. For this purpose, we are using a structural approach of parsing the sentences (Sentence Segmentation) and tag the chunk of words into different parts of speech (POS Tagger). It can also be used to map these words into functionalities, entities, attributes, and relationships. We are using different types of NLP toolkits. The most important advantage of this project is to reduce the human effort and saves a lot of time.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

1.1 Background

In this section, we discuss and review the previous work of implies with natural language on UML diagrams. Specifically on the Entity-Relationship Diagram. Most of the work has done on different software with the inclusion of relational database schemas like commercially used software like ERD plus, Microsoft Visio, Lucid chart, and many more. Different researchers used a different methodology to make a different tool that draws an ER Diagram. Entity-Relationship Diagram is an organized way to represent the relational data many researchers present many components based tools but draw an ER Diagram from natural language software requirement is a new way to draw ER Diagram model.

Many Researchers work on UML diagrams with different strategies. Eman Btoush presents a method to generate an ER diagram from requirement specification by using natural language processing tools like sentence segmentation, tokenization, chunking, parsing, and then apply a heuristic rule (means to solve a problem in a limited time by using some shortcuts) to identify (entities, attributes, and relationships) [1].

Peter Pin-Wan Chen develop 11 rules for the ER Diagram model which is built from English based software description. Peter Chen develop rules for ER Diagram elements like entities, attributes, and relationships. It used a noun as entities, adjectives as an attribute and for relationship used a transitive verb [2]. P. More and R. Phalnikar used the NLP and domain ontology techniques to generate a RAPID system to develop a UML diagram from textual user requirements. Later on, extended to specific class diagrams [3].

1.2 Problem Statements

Generating UML diagrams from software requirement specification without user intervention is a very difficult task because natural language text not easily cater when it is large. Most of the research work and laboratory base tool is to generate automated UML Diagrams from user requirements. But there are a lot of issues with these tools and methodologies which are based on conceptual models of UML Diagrams. Mainly focus on generating automated entity-relationship diagrams in UML Diagrams. The main problem in the generation of entity relation diagram from the software requirement specification is to handle the large size requirement text. Most of the irrelevant objects detect which creates noise in the generation of ER-Diagram which also affects the accuracy of the system.

The elements that help to construct the entity-relationship diagram is also very difficult to identify. Some of the identity by applying the heuristic rules but for key attributes and notations, there are a lot of problems faced due to sentence structure. In the Automated Diagram generator, we only focus on identifying entities, attributes, and relationships. Further, we have a verification problem rather the generated diagram or model of the relational database is correct or not. Many researchers faced the above-mentioned problems which are related to Natural Language Text.

1.3 Aims and Objectives

There are several aims & objectives of this project some of them are mentioned below:

i. Understanding of software requirements.
ii. Find entities and attributes with their relationships.
iii. Assign different colors to those entities according to the rules.
iv. Draw a software diagram.
v. To provide minimum human involvement.
vi. It is helpful for beginners in the field of software designing.

1.4 Scope of Project

There are some important features of this tool are segmentation of sentences, tokenizing, POS Tagging, chunking, parsing, and generating of the diagram. In the POS tagging process, we will assign different colors to the tags which identify words to their corresponding part of speech. This tool is very useful in the fields related to software development and designing. It reduces the human effort and saves time. It can also be used by non-technical people in small business. The main problem that occurs in the process of automation is how accurate the diagram you generated. The main audience of our system will be the one who wants to design some kind of software. We should not only target software development firms but also for anyone who wants to design a diagram for their system. Our system will be fault-tolerant because it will not be disturbed due to the failure of a single functionality.

CHAPTER 2

LITERATURE REVIEW

Natural language processing (NLP) is the automation processing of human natural language. Software requirements are regularly specified in Natural Language so its descriptions often need to be examined, transformed, and rationalized into a form of design representation during the development of software applications. Recent studies also focused on automating the extraction of information from natural language text using Natural Language Processing to build a UML Diagram [4].

There are some natural language processing (NLP) tools, which allow language analysis and provides automatic support to generate different Unified Modeling Language (UML) diagrams. Software diagrams are the most important process in the field of software development, especially in the designing phase. Unified Modeling Language offers numerous types of diagrams that are used to increase the understand-ability as well as the development of an application at the software construction phase. These software diagrams help to arrange the software information like Entities and attributes as well to make relations among them. It is also the graphical representation of the software. Entity Relationship and Class diagrams are the two most important UML diagrams which plays a vital role while designing a system database.

Though, obtain any type of entity-relationship model it must be required to identify and extract the entities and attributes from a system's requirements that needs a piece of expert knowledge which is a lengthy process or can take much time or money while designing manually. Lack of expertise creates chances of errors that can be difficult to resolve later. Therefore it is necessary to have a tool that can automatically generate an ERD diagram based on natural language requirement specifications. In this project, we are developing a desktop application using NLP techniques to generate the diagram. We are using NLP techniques for the extraction of UML diagrams from natural language requirements by implementing the structural approach. That approach begins with translating user requirements to words and applying with its specific Part Of Speech (POS).

The parsing process is proposed and a set of syntactic heuristics rules are applied to identifying entities, attributes, and relationships of the target system. We are using one of the most popular toolkits of python for Natural Language Processing i.e. NLTK. We will work on PyCharm which is an open-source IDE, especially for python. We will make sure to provide an efficient and fast way to produce a software diagram from the software requirements

2.1 Related Work

In this section, we discuss and review the previous work of implies with natural language on UML diagrams. Specifically on the Entity-Relationship Diagram. Most of the work has done on different software with the inclusion of relational database schemas like commercially used software like ERD plus, Microsoft Visio, Lucid chart, and many more. Different researchers used a different methodology to make a different tool that draws an ER Diagram. Entity-Relationship Diagram is an organized way to represent the relational data many researchers present many components based tools but draw an ER Diagram from natural language software requirement is a new way to draw ER Diagram model.

Many Researchers work on UML diagrams with different strategies. Eman Btoush and M. M. Hammad present a method to generate an ER diagram from requirement specification by using natural language processing tools like sentence segmentation, tokenization, chunking, parsing, and then apply a heuristic rule to identify (entities, attributes, and relationships) [1].

Peter Pin-Wan Chen develop 11 rules for the ER Diagram model which is built from English based software description. Peter Chen develop rules for ER Diagram elements like entities, attributes, and relationships. It used a noun as entities, adjectives as an attribute and for relationship used a transitive verb [2]. P. More and R. Phalnikar used the NLP and domain ontology techniques to generate a RAPID system to develop a UML diagram from textual user requirements. Later on, extended to specific class diagrams [3].

Sarita Gulia S. Gulia and T. Choudhury propose an automated UML Diagram generator from software requirement specification using NLP. It generates the activity and sequence diagram by using Stanford parser and Stanford POS tagger. It implements sequence and activity diagrams from user requirements. The sequence

diagram constructs with actor, object, and message. The author discovers some steps to identify the components of the sequence and activity diagram. For an actor it will use a subject, verb, and object will be detected by integrated words after the preposition. The message will be passed between the objects. The strategy for message detection used as a verb phrase in structure sentences. In the activity diagram, it will extract the different verbs from the requirement document. Also, activities derived from verbs [5].

P. G. T. H. Kashmira and S. Sumathipala used three-module to recognize and more efficient ways to detect irrelevant and incomplete information. It used a machine learning module to make a sure minimum intervention of the user. The machine learning module identifies entities, attributes, and relationships by supervised learning. It used the different algorithms for elements of ER-Diagram. For the accuracy of algorithms, it used to recall and precision calculations. Which helps to consider the most accurate features extraction of ER-Diagram [6].

Nazia Omar and et all-purpose heuristic-based ER-converter which reduces the human effort and helps the non-technical database designers [7]. It determines the elements of the ER diagram by using Memory Base Shallow Parser (MBSP) [8]. It determines the features include an entity, attribute, relationship, cardinalities, and weights. For entities, attributes, and the relationship we see most methodologies the same but with different changes. It develops heuristic for cardinalities as an adjective “many” or “any” shows a maximum cardinality. Weights are labeled as a true event at a high confidence level. Their tool refers to semi-automated [7].

D. K. Deeptimahanti and M. A. Babar develop a tool that generates UML Diagram models automatically build in java and NLP technologies to cater to the requirement specification. It used a tool that generates the models. The models generate through the NLP technique. The tool is a combination of a rational unified process and Iconix process. It develops the use-case diagram, conceptual model, collaboration diagram, and design class model [9].

I. Song and M. Evans used the notation that is used for modeling and designing the relational database. It mainly focuses on cardinality. It will analyze the ER-Diagram method that is mostly used in the case study classified as a binary model, n-ary model. It will compare 10 different notations of an entity-relationship diagram. It collectively used different models of different researchers [10].

N. Madnani describes the importance of Natural Language Diagram NLP. It discusses the problem related to NLP and solve through the NLP toolkit a few of are mentioned that will help a beginner to know about NLP from this paper. It also discusses the currently working flow that most of the researchers were written on. It gathered the resources of NLP which is helping to not spending time on taking resources from outside [11].

N. Sarwar and I. S. Bajwa present an object role model that graphically shows the framework for different areas. It full fills all the essential things that could be possible for input. It helps all the areas that could use to show his requirement as graphically. It involves a comprehensive step to generate the diagram. It used semantic business vocabulary rules SBVR that is utilized for the data framework. It used SBVR parser with NLP techniques then change the SBVR metamodel to ORM metamodel. The author includes the extracting elements of SBVR semantic business vocabulary rule that is related to our work which is the extraction of a noun [12].

S. Geetha and G. S. Anandha Mala use an approach to present the structured database from Software Requirement Specification (SRS). It generates the automated schema from requirement but mainly focuses on identifying the key attributes primary key, foreign key (PK, FK). The author uses the same previous methodologies for the generation of the class diagram but additionally uses a schema generator that extracts the attribute of class and then identifies the primary key and foreign key. After identifying the key attribute it will represent the diagram in XML and load the attributes in the relational database [13].

M. Ibrahim and R. Ahmad demonstrate the extraction of class diagram method and requirement analysis from Natural Language Text NLT and domain ontology. It develops a desktop software named. Requirement Analysis And Class Diagram (RACE). Domain ontology and NLP techniques used to generate an accurate automated class diagram. The author defines the identification rules of elements to help construct the automated class diagram. Wordnet and stemming algorithms used to validate and removing the distortion in text. The tool developed in C# that works on 100 words requirement [14].

F. Hogenboom talks about Natural Language Processing (NLP) based systems and applications. In all NLP system information extraction and information retrieval is the main focus to process the data to achieve a high accuracy level. In his research work, it used a statics-based approach, pattern-based approach, hybrid-

based approach. Static-based approaches require a large amount of data in text to develop models. It is used for a data-driven approach. Contrast with the statistic approach pattern-based approach required human intervention to process the text means a knowledge-driven approach. The hybrid approach is a mixture of both statistics and pattern-based approaches [15].

R. Sajjad and N. Sarwar propose a check and balance method for UML diagrams models. These models create to ease the programmer the follow the flow of graphically that will help a lot but the model they work on is correct or not they have no check and balance. It used three modules to verify the software model. It extracts the relations through morphological analysis or solving problem from all possible solutions, syntax analysis, and semantic analysis. It mainly focused on elements of the UML class model [16].

All the systems and methodologies discussed above may use extraction techniques of Natural Language Text to build UML diagrams using NLP Tools and techniques. All the above used NLP techniques with some changes to automate the UML diagram from Software requirement Specification. Software Requirements are difficult to handle mostly a large amount of text. Many researchers propose the laboratory base tool to check on the requirements rather it is relevant or not but this requires more time to give worth accuracy in text. The generation of ER-Diagram through without intervention of the user is a difficult task but the researcher presents some limited tools and methodologies. Identify the elements from ambiguous and non-structured text is very difficult to produce high accuracy results. From the above research, we conclude that all the researchers proposed their methodologies but did not mention any live tool or system that generates the ER diagram automatically and the majority of them produce the results based on assumptions. But we present the automated diagram generator which specifically generates the Entity-Relationship Diagram from requirement specification using NLP Tools and techniques.

2.2 Limitation

Ambiguity and linguistic variation are the main problems in the Automated Diagram Generator (ADG). Incomplete or irrelevant software requirement is complex to handle in one system. A heuristic approach is far better to use in-text [1]. It is harder to find a primary key, foreign key, and cardinality. These elements are important to represent the relational database design through NLT [13]. Requirement analysis is the main step in the software development cycle [4].

‘

CHAPTER 3

DESIGN AND METHODOLOGY

3.1 Design

In the design part, some UML diagrams show a complete flow of ADG and how it can design the functionalities to fulfill the project. These UML diagrams are as follow:

i. Use Case Diagram
ii. Sequence Diagram
iii. Domain Model
iv. Class Model
v. Relational Data Model

3.1.1 Use Case Diagram

A use case diagram shows the functionality that occurs in a system. The invocation of methods is shown including the actors which can perform these functionalities directly or indirectly but in our scenario, there will be users. The use case Diagram of ADG is as follows.

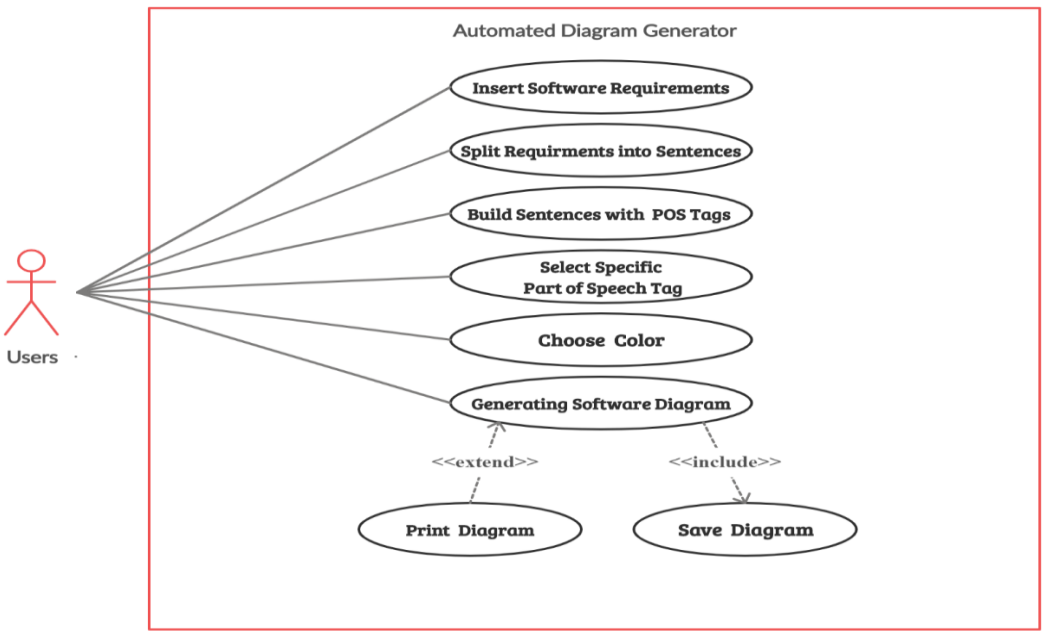

Figure 3.1: Use Case

Below there are some tables that describe the use case diagram completely step by step of Automated Diagram Generator (ADG).

Table 3.1: Insert Software Requirements

Name and ID	Insert Software Requirements (U1)
Brief description	Insert the software requirements in the provided workspace area. In which users enter the valid software requirements. Morphological analysis is applied to software requirement text.
Preconditions	ADG must be open or running.
Basic flow or Happy path	Users copy the software requirements (case study) paste in the text area or type the software requirements.
Trigger	The user has a software requirement.
Post-conditions	Software requirements are inserted in the text area.

Table 3.2: Split Requirements into Sentences

Name and ID	Split Requirements into Sentences (U2)
Brief description	The users split the requirements into sentences to remove the ambiguity.
Preconditions	U1 or Software requirements are already inserted
Basic flow or Happy path	The user selects the Sentence Segmentation option. The system validates the natural language text. The system eliminates the non-word tokens. Split the software requirements into sentences.
Alternate flows	If the system does not recognize the natural language text (English) an error pop up to indicate user to enter English text.
Post-conditions	Text is in sentences and ready to be tokenized.

Table 3.3: Build Sentences with POS Tags

Name and ID	Build Sentences with POS tags (U3)
Brief description	The user operates to transform sentences with appropriate tags and colors.
Preconditions	Software requirements are in the form of sentences and these are converted into tokens.
Basic flow or Happy path	The user selects the POS tag option. The system converts sentences into words or tokens. The system assigns the relevant tags to the tokens. The system assigns different colors to each tagged token. The system converts the tagged tokens into sentences
Post-conditions	Sentences with the colors represent the part of speech tag of each word.

Table 3.4: Select Specific Part of Speech Tag

Name and ID	Select Specific Part of Speech Tag (U4)
Brief description	The user can select a specific part of speech tag to understand the color patterns of tags.
Preconditions	Colors and tags are assigned to the sentences.
Basic flow or Happy path	Default all the POS tags are selected. Users can choose the specific POS tag option. The system displays only the selected options words. The system converts the tagged tokens into sentences
Post-conditions	User can see the desired category of Part of Speech.

Table 3.5: Choose Colors

Name and ID	Choose Colors (U5)
Brief description	The user can choose the colors for entities and attributes.
Preconditions	Colors and tags are assigned to the sentences
Basic flow or Happy path	Default colors are selected. Users can choose specific colors among different options. The system saves the user selection
Post-conditions	The user saves the colors for the entities and attributes to the system.

Table 3.6: Generating Software Diagram

Name and ID	Generating Software Diagram (U6)
Brief description	The user gets an ER diagram from the software requirements.
Preconditions	Colors and tags are assigned to the text and also colors for entities and attributes are selected already
Basic flow or Happy path	The user selects the generated diagram. The system parses the tagged text by applying rules. The system extracts the entities and attributes. The system assigns the user colors to entities and attributes. The system makes relationships for entities and attributes. Displays an ER diagram.
Post-conditions	An ER diagram is generated successfully.

Table 3.7: Save Diagram

Name and ID	Save Diagram (U7)
Brief description	The user can save the ER diagram.
Preconditions	The diagram must be generated.
Basic flow or Happy path	User set file name. User select save for saving diagram. Users select no for without saving.
Post-conditions	Diagram save successfully or not.

Table 3.8: Print Diagram

Name and ID	Print Diagram (U8)
Brief description	The user can print the ER diagram
Preconditions	The diagram must be generated.
Basic flow or Happy path	The generated diagram must be printed
Alternate flows	While printing the ER diagram printer does not work properly. The user connects the printer.
Post-conditions	Diagram successfully printed.

3.1.2 Sequence Diagram

A Sequence diagram depicts the sequence of actions that occur in a system. The invocation of methods in each object and the order in which the invocation occurs is captured in a Sequence diagram. The Sequence Diagram of ADG is as follows:

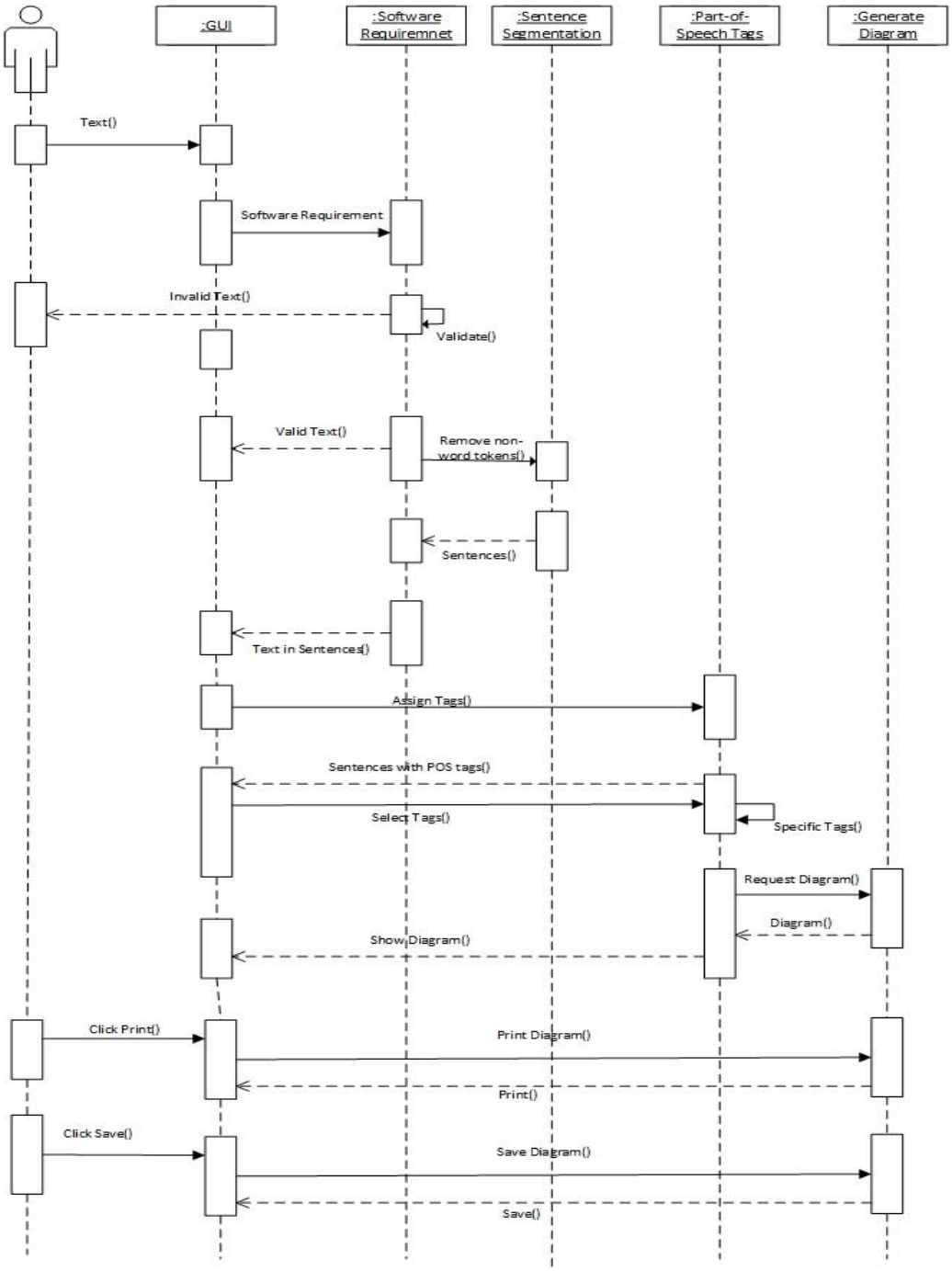

Figure 3.2: Sequence Diagram

3.1.3 Domain Model

The domain model shows concepts of domain related to their relationships. It is a package that contains all the workflows between different classes with their relations. The domain model of ADG is as follow:

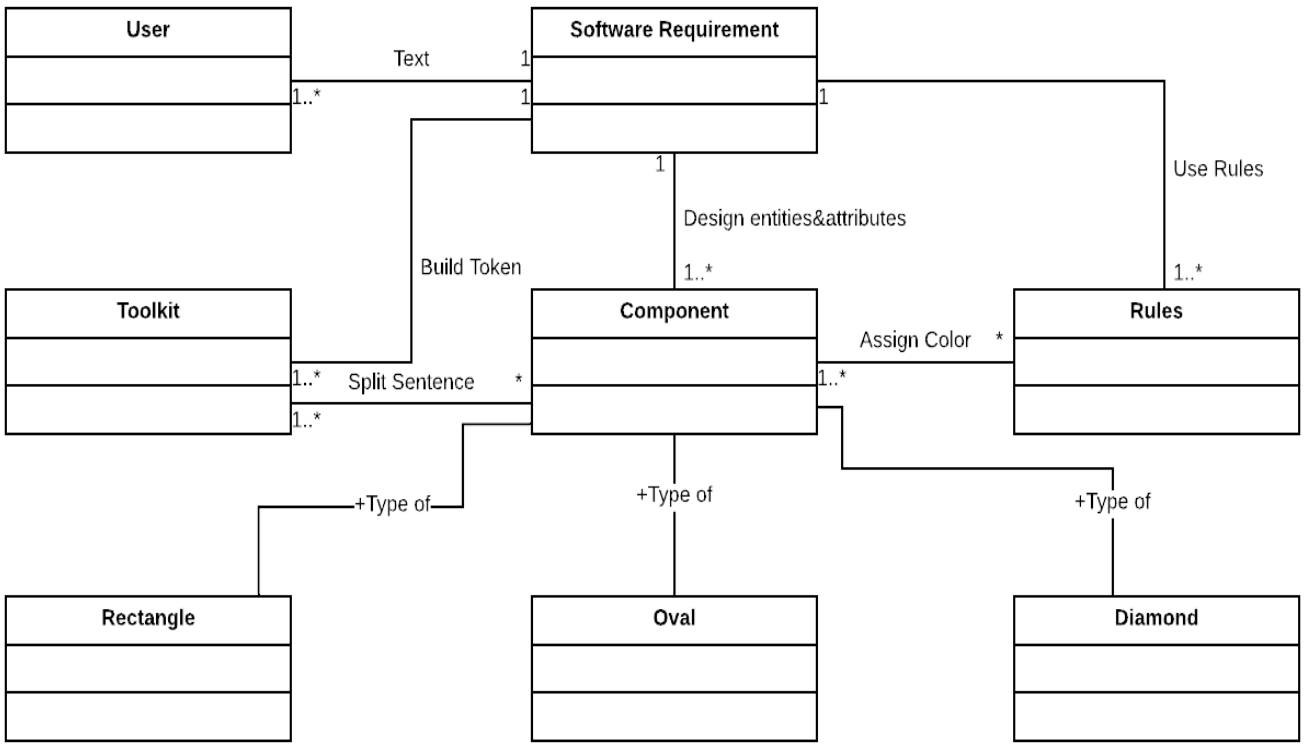

Figure 3.3: Domain Model

3.1.4 Class Diagram

The class diagram shows the pattern of how we proceeds to the implementation phase and accomplish the final goal. It describes the logical and physical representation of variables with their methods or functionalities of each class. The class diagram of ADG is given below:

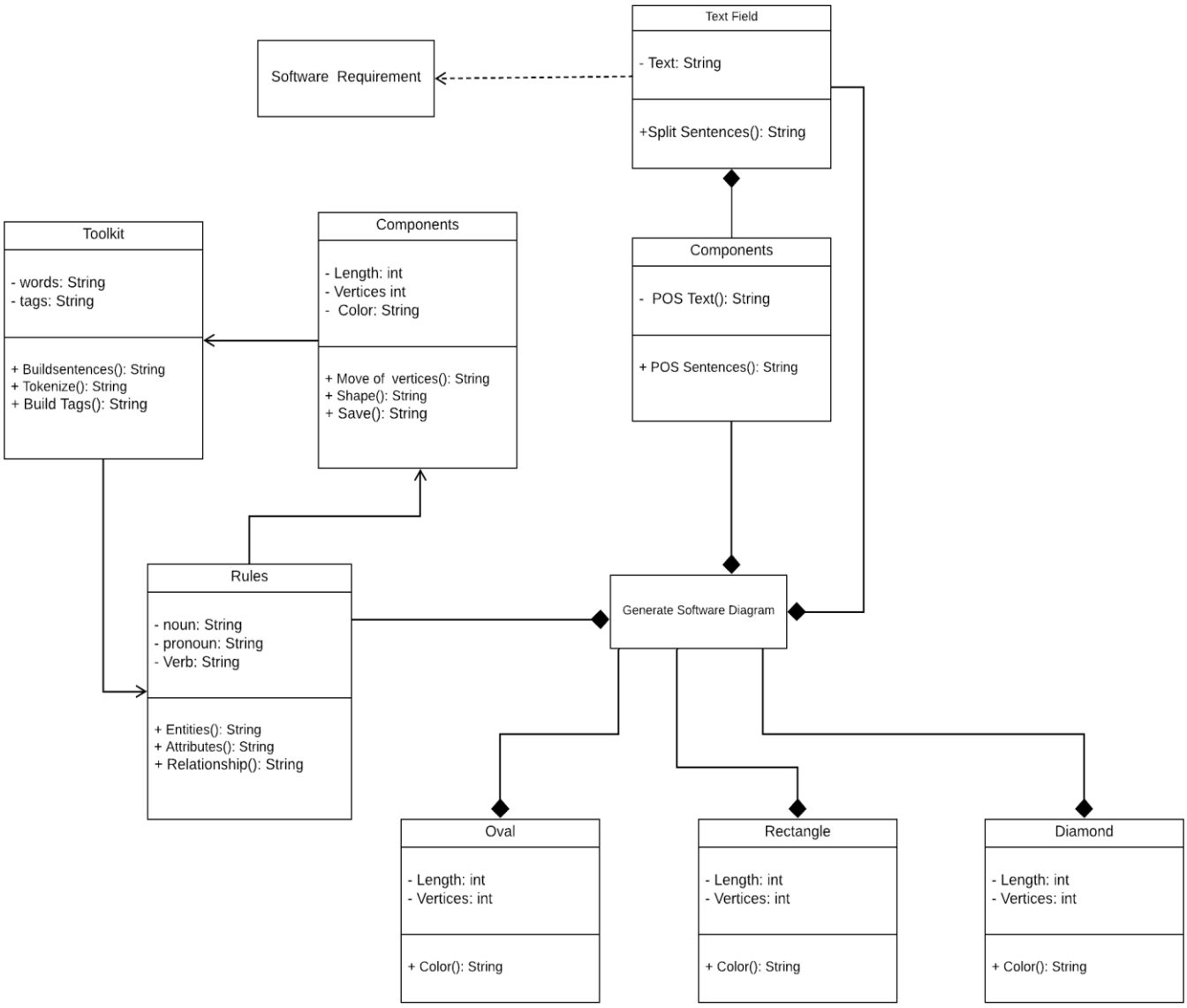

Figure 3.4: Class Diagram

3.1.5 Relational Data Model

The data model is a subset of the implementation model, which describes the logical and physical representation of persistent data in the system. The data model of ADG is given below:

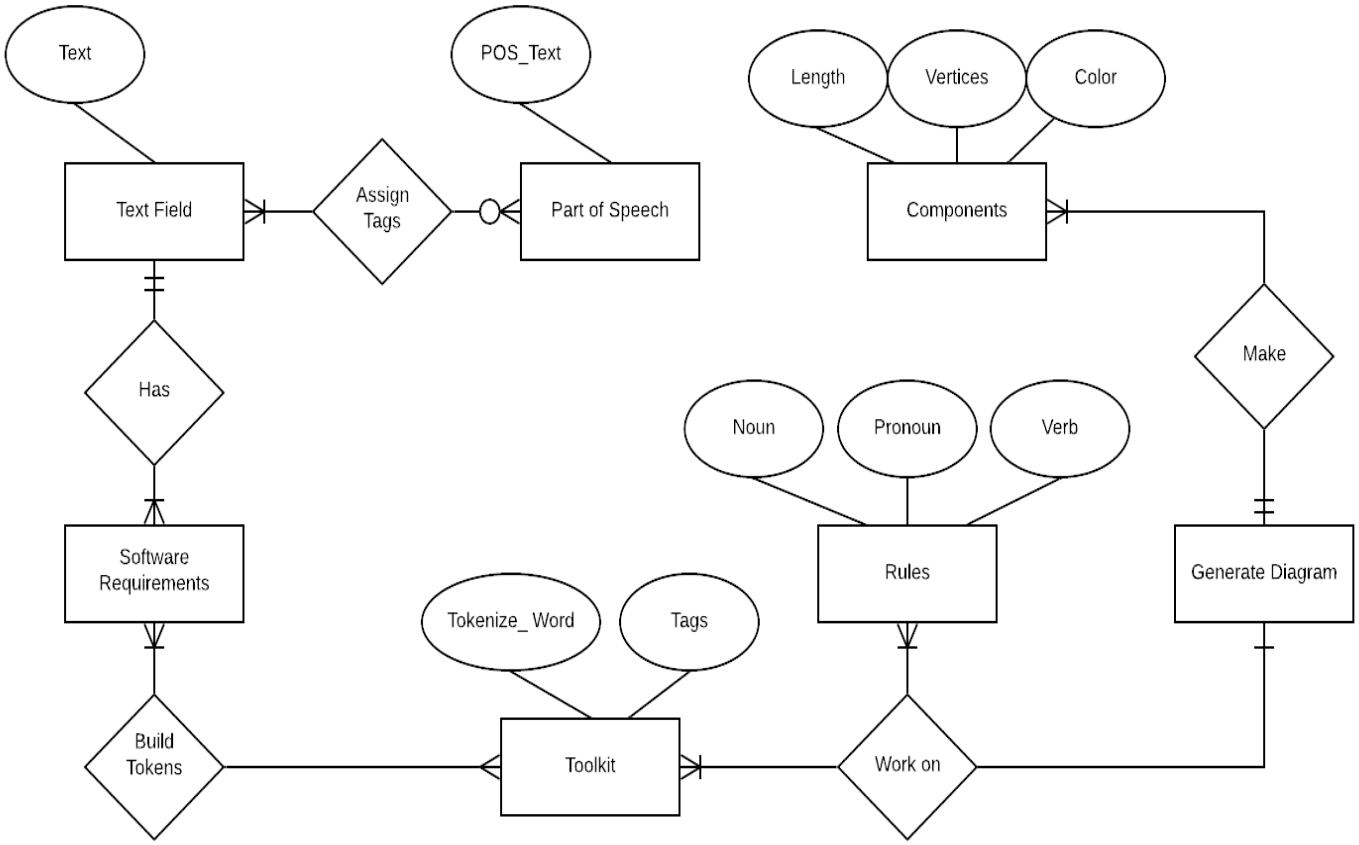

Figure 3.5: Relational Data Model

3.2 Methodology

The approach (as shown in Figure 1) of our project automated diagram generator that generates the ER Diagram from user software requirement specification. Software requirement in a textual syntactic structure which processed with natural language techniques.

1. Software requirements split into sentences with no ambiguity by sentence segmentation technique.
2. Split sentences are shaped on tokens. Every word of a sentence are made up as a token
3. Token assigns with tags to ensure more clean sky to pick up entities, attributes, and relationships.
4. Parsing technique and sematic analysis on POS tags to get the fine results.
5. Semantic analysis and morphological analysis also apply on tags that help to heuristic rules to choose valid entities, attributes, and relationships.
6. Common and proper noun candidates for entity type which identifies by event detection[1]. Object with informative attributes makes an entity [10].
7. Attributes can be identifying by event detector which may include a noun phrase or adjective and certain conditions[1][2].
8. The relationship can be identified by an event detector which includes transitive verb [1][2]

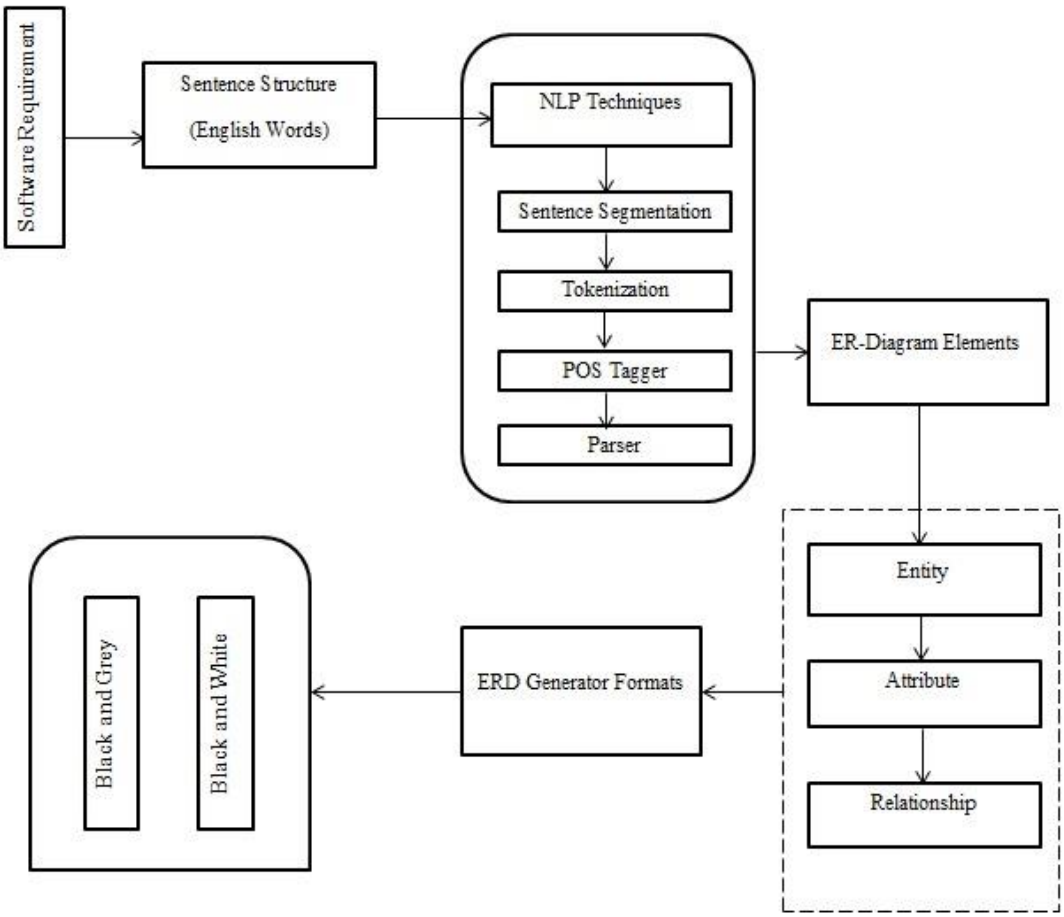

Figure 3.6: Proposed Methodology

3.2.1 Tools & Techniques

Software and tools are used to develop the application (ADG, automated diagram generator) which are as follow:

1) Sentence Segmentation
2) Tokenization
3) POS Tagger
4) Parser
5) Rules

3.2.1.1 Sentence Segmentation (Tool)

Sentence Segmentation is an analysis tool in natural language. Which is used to split the sentences, detect sentence boundaries, and make a noun (plural) into singular. Remove all ambiguity from the text. To make it easy to build tokens of

all non-words [1]. This sentence segmentation tool is a part of the ADG automated Diagram generator. The user fills the text area with software requirement then sentence segmentation removes all the punctuation and split the all text into sentences which further move to make tokens.

3.2.1.2 Tokenization

Tokenization is a natural language technique to divide the text into a sequence of words and characters. Tokenizer based on regular-expression. It breaks the sentences into words of tokens and then parser recognizes the root of the words and sentences [8]. In the ADG tokenization tool used after sentence segmentation to process its constituent tokens of split sentences.

3.2.1.3 POS Tagger (Tool)

Part-of-speech is a natural language processing tool that is used to differentiate in a text with the noun, adjective, etc. in the ADG automated diagram it is used to identify the noun, adjective, and verb to help in identifying elements of an ER diagram. This is used in the tokenization process [1]. It read the text and assigns a POS tag to each word [1].

3.2.1.4 Parser (Tool)

A parsing technique in natural language is used to gather the grammatical structure of the text in the form of sentences [1]. Natural language parser joined the words, sentences, etc after analysis. In the ADG parsing tool used to gather the pos-tags with morphological analysis. In ADG automated diagram software parsing used to select more competitive tags to choose for elements of the ER diagram.

3.2.1.5 Rules

For Entities: In Automated Diagram Generator (ADG) entities are automatically identified by applying heuristic rules. Heuristic rules include a proper noun that may indicate the entity. a verb that acts like a noun ending withing a “gerund” satisfactory for the entity type. A noun such as “database”, “record”, “system”, “information”, “organization” and “detail” not suitable for the entity type. Proper noun like “Place”, “Name” not eligible for the entity type.

For Attributes: Attributes are along with their entities that are also extracted from rules to automate the system. Attributes are nouns such as “employee id”, “identity No.”, “date”,” address”, “name”, etc.

For Relationship: A relationship identification is also done by automating by applying rules. The connection between the two entities is likely called a relationship. In other words, a transitive verb indicates the relationship. A verb goes along with by a preposition such as "by", "to", "on" and "in" can indicate a relationship type.

3.2.2 Elements of ER-Diagram

By using different toolkits of natural language processing technique we will generate a diagram from user requirements. MBSP is a text analysis system that offers tools for chunking, splitting of sentences, part of speech tagging, tokenizing, and relation finding. We present a particular structure for generating a diagram from natural language in Figure 1. It begins by splitting the natural language text into sentences in the sentence segmentation process. Each sentence terminates with a period. It also eliminates the non-word tokens. These sentences then break into tokens in the tokenizing process. NLTK is a popular library that plays an important role in the completion of major parts of our project. It has a function of word tokenizer which is very beneficial for the process of tokenization. This process separates each word of all sentences with spaces. Part of Speech (POS) Tagging is a process of labelling the word-tokens with its acronyms. We will also assign colors to these tokens. The chunking process organizes these tag tokens into sentence units. This process is also used to indicate the type of tag token by selecting small chunks. The parsing process determines a parsing tree based on Syntactic rules or methodologies that help us to

relate sentence units with each other and generate a diagram. The most common elements required to generate a diagram are as follow:

- Entities
- Attributes
- Relationships

3.2.2.1 Entities

An entity is an object that contains more than one attribute. An Entity type is derived from a noun which is discussed above that heuristic rule applies to figure out the proper object for the entity type. In ADG (Automated Diagram Generator) we presently work in entities. Later we have work on weak entities that are dependent upon another entity called parent entity.

3.2.2.2 Attribute

An attribute is a property that describes the characteristics of an entity. In ADG (automated Diagram Generator) attribute can be defined by heuristic rules which may choose adjective or noun phrase as an attribute as we discussed above. An entity must have a primary key attribute that is uniquely defined in the relational database. An attribute that is the primary key of one entity and relates to another entity is being defined as a foreign key of that entity. In ADG (Automated Diagram Generator) we only work in the identification of finding attributes by heuristic rules.

3.2.2.3 Relationship

A relationship is an association between entities and their attributes. In ADG (Automated Diagram Generator) we work on finding a relationship by applying heuristic rules which are discussed above. A transitive verb from software requirement might be a relationship between two entities. Adding more in a relationship described as cardinality. They are in the form of 1:1, 1:M, M: M. it will help in modelling of relational database design.

CHAPTER 4

IMPLEMENTATION

The implementation part or development phase of a project is a process to shaped designs into productive products. ADG is a software system or a desktop application so there are some technical constraints used in the development phase. These constraints are:

1. Hardware
2. Programming Language (Frontend and Backend)

4.1 Hardware

For Developing there is need of PC or personal Laptop with minimum 250 to 500 GB Hard drive with 4 GB RAM for smooth running of libraries or toolkits of natural language processing and for the tools on which ADG developed.

4.2 Programming Language (Python)

We are using Python as a programming language for our project ADG. Python is a vast programming language. It is secure, speedy, and easy for developers because of its standard libraries which help the developers in developing. It is a fundamental technology that good in scratch programs including games, entertainment, business applications. it supports all operating systems. ADG software develops in python. In python, many GUI frameworks used to develop GUI for applications but for ADG we are using PyQt5 for frontend and PyCharm 2019.2.4 for the backend.

4.2.1 Frontend

PyQt5 is a python toolkit used for GUI frameworks. It has both the designer tool (from which you can drag and drop the objects and saves time) as well as programming surface (typically code for objects. We used different designer tools of PyQt5 for frontend.

4.2.2 Backend

In our project, we integrate PyQt5 with PyCharm community 2019.2.4 which is the best programming environment for python. PyQt5 gives a lot of libraries for designing like for shapes that are used in the building of the ER diagram. Also some kind of functionality for applying colors for the POS Tagger tool. Basic python libraries including nltk etc are also used for backend.

CHAPTER 5

RESULTS AND DISCUSSIONS

5.1 Home Screen Layout

There is a welcome screen with the message of "Welcome to Automated Diagram Generator". It also contains the menu bar which contains a "File menu". The file menu contains two action bars "New Project" or "Exit". Exit action bar terminates the systems or application whereas the other action bar take user to the next screen where the main components of ADG are present.

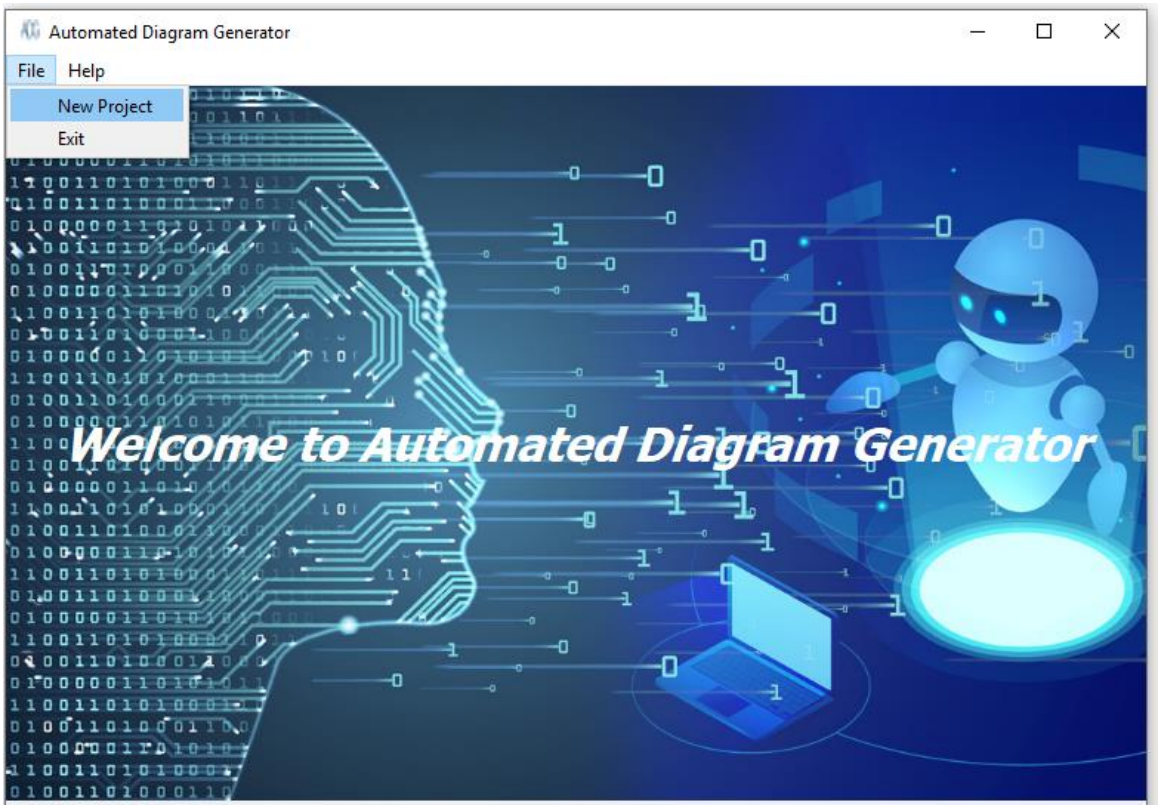

Figure 5.1: Home Screen Layout

5.2 Main Screen Layout

There are several objects in this screen consisting of three columns. In the first column, there are three buttons (for sentence segmentation, POS Tagger, and Generate ERD) with a combo box (to select colors for ERD). In the second column, there are two text fields or areas, one is for input (valid software requirement) and the other is for results of sentence segmentation as well as for POS tagger. The third and the final column of the screen have some color bars reflecting the assignment of a specific color to the text concerning their category of the tag.

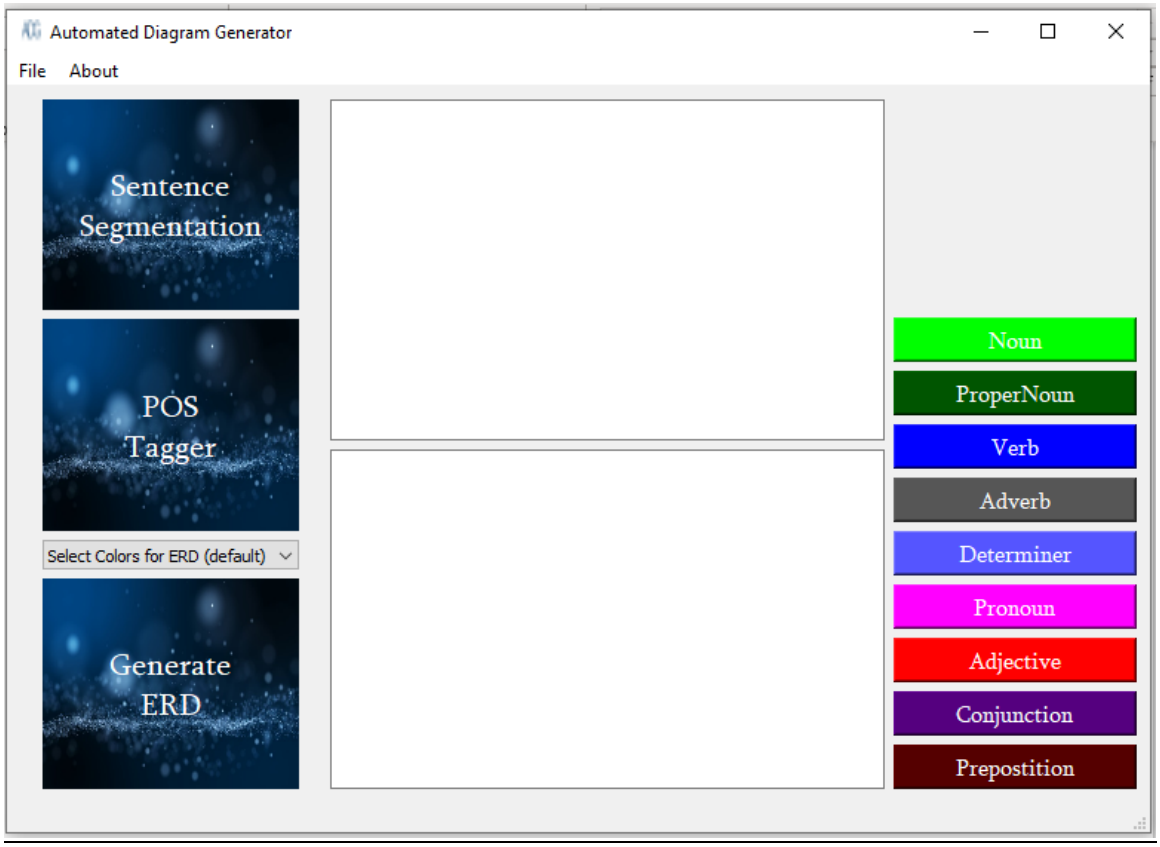

Figure 5.2: Main Screen Layout

5.3 Case Study 1 (Valid Software Requirements)

In this case study, we are using simple requirements which are based on entities and relationships. This example shows how the entities are extracted from requirements and make relationships with each other.

5.3.1 Insert Software Requirements

First, we add the valid software requirements in our input text field as shown in figure 5.3. Before inserting these requirements all the buttons are disabled and inserting the requirements only sentence segmentation is enabled.

"Assume that an employee may work in up to two departments or may not be assigned to any department. Each department contains phone numbers and employees have these phone numbers."

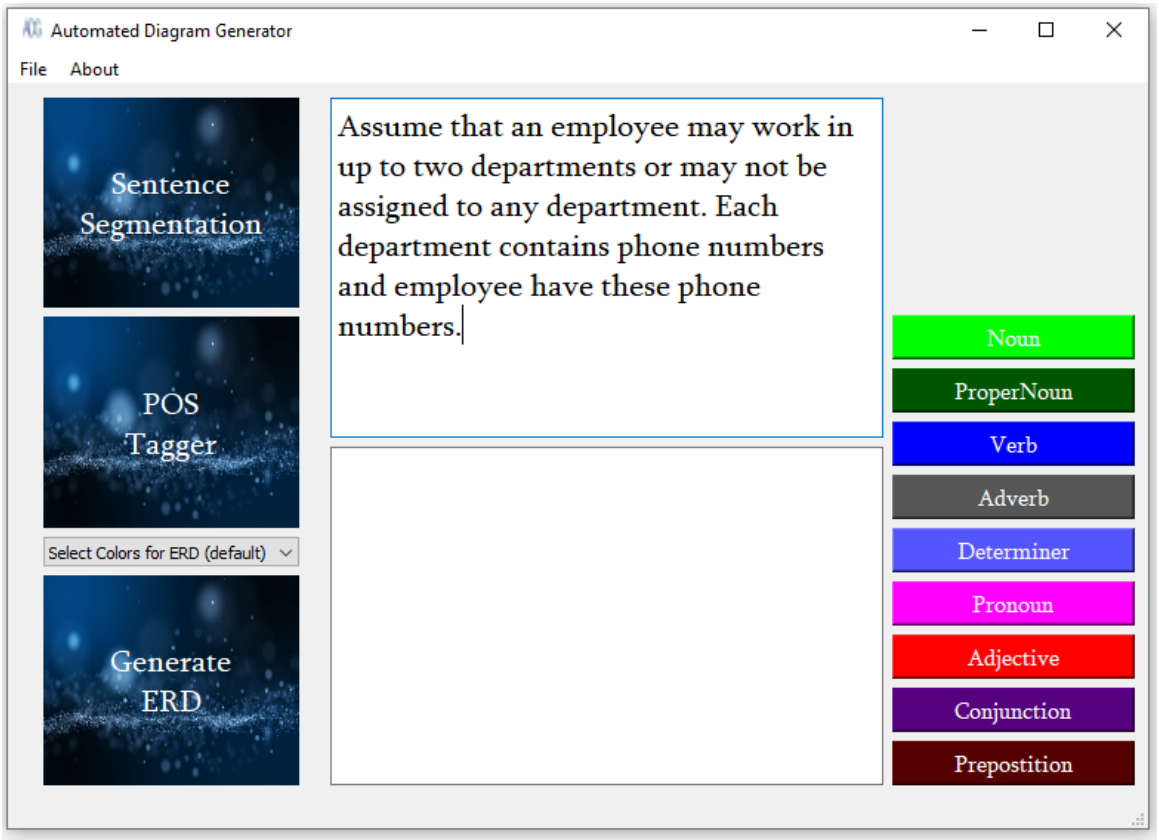

Figure 5.3: Insert Software Requirements

5.3.2 Sentence Segmentation (Tool)

This tool converts the input requirements into sentences and removes the ambiguity by singularizing the plural words (e.g. in this example it will convert “departments” to department) because it will help in the process of extraction especially for entities. By clicking on the “Sentence Segmentation” button we perform the tool and it shows the results in the resultant text field. By doing segmentation, the POS tagger is enabled.

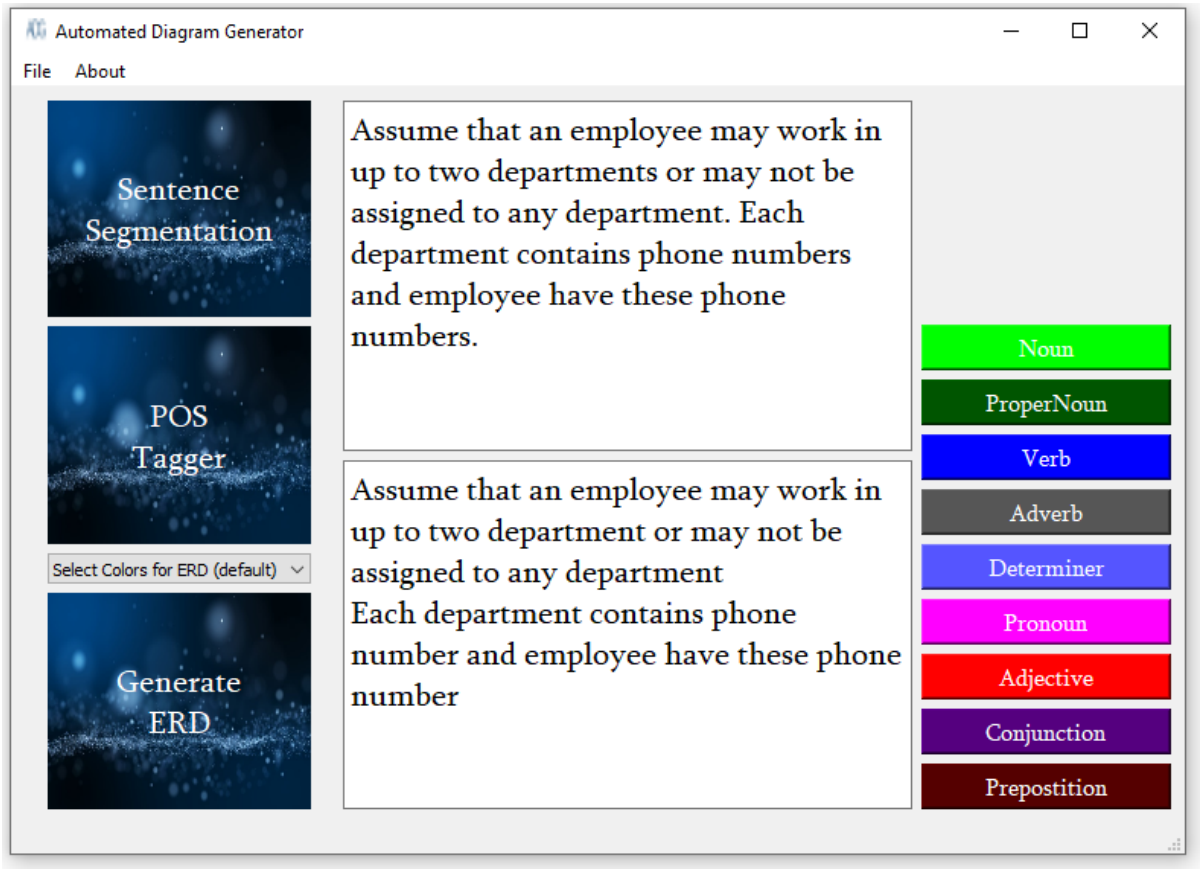

Figure 5.4: Sentence Segmentation

5.3.3 POS Tagger (Tool)

This tool uses the tokenization of the sentences and gives them POS tags with the help of the built-in library. Then assign specific colors to these tags. These colors concerning their part of speech tag can be shown in the third column. All these color bars represent their POS tag. Now the choose color and/or select a specific part of speech and/or even generate an ER diagram.

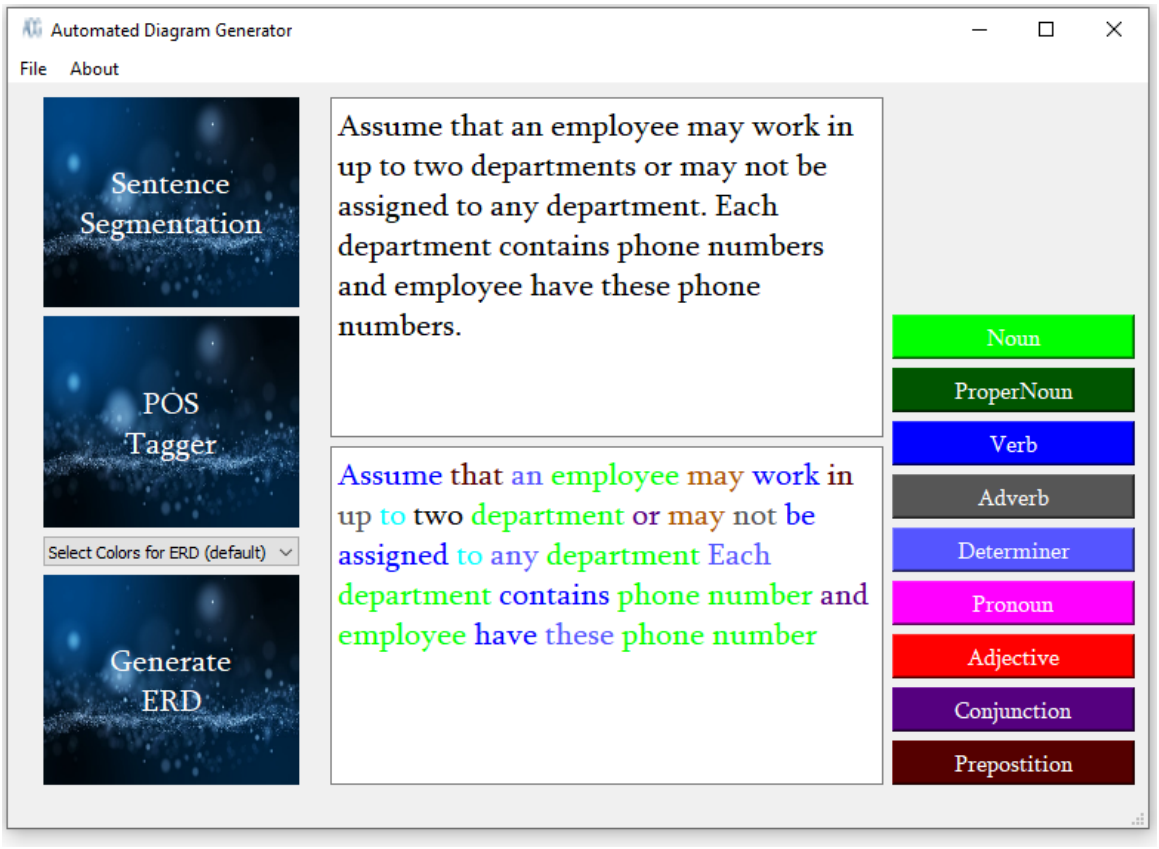

Figure 5.5: POS Tagger

5.3.4 Choose Specific Part of Speech

Select a specific POS tag by selecting the desired bar. In this case, we select “Noun” so it displays the result in the resultant text field.

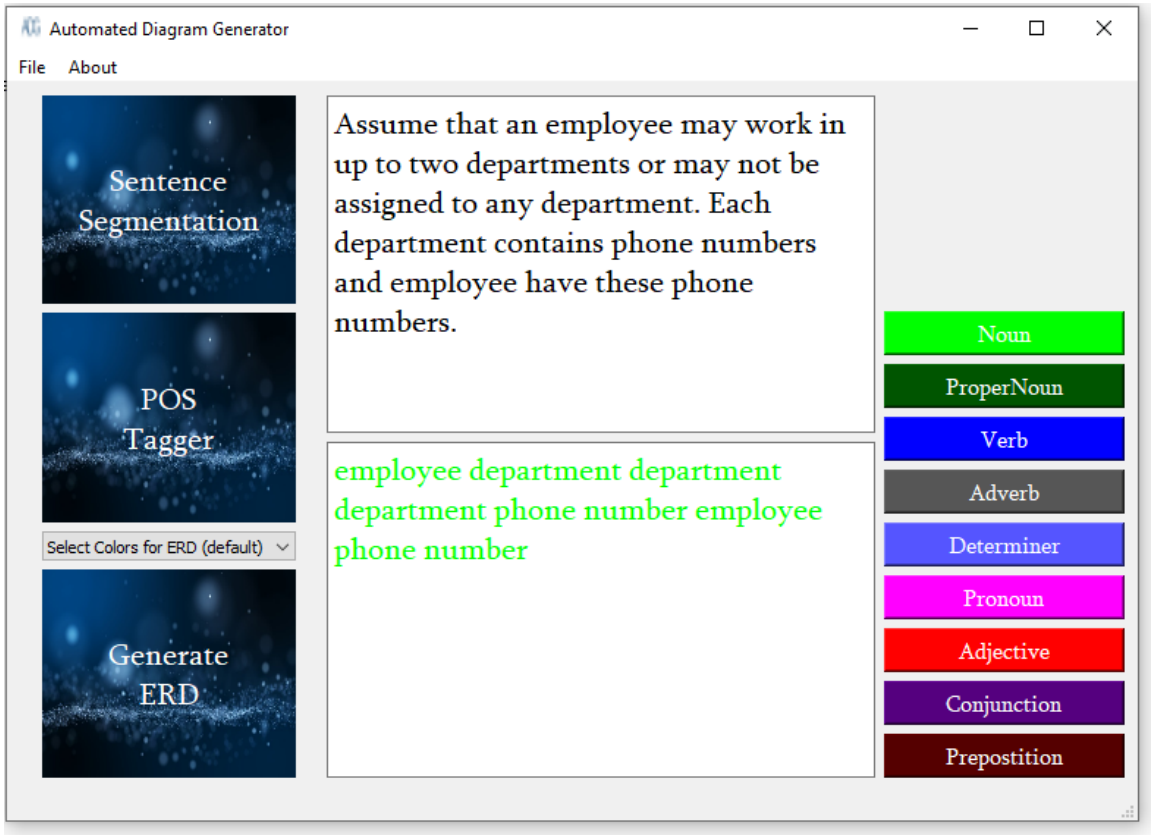

Figure 5.6: Choose Specific POS Tag (Noun)

5.3.5 Choose a color pattern for ERD

Select a color pattern for ERD in this case we select the “Black and White” which is also a standardized color pattern.

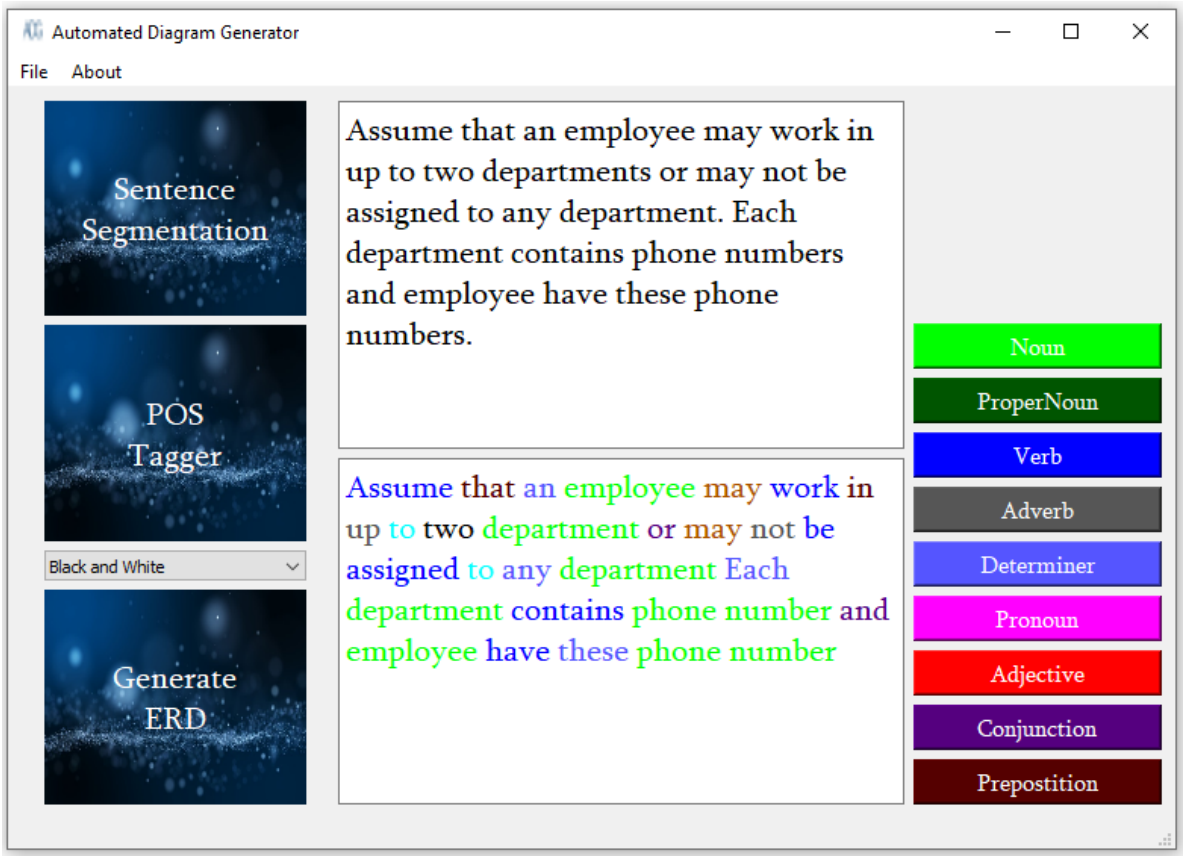

Figure 5.7: Choose Black and White Style

5.3.6 Generated ERD

Generated the ER diagram in a standardized black and white pattern. This case study takes one to two seconds to extract the entities, attributes, and relationships and generate the ERD.

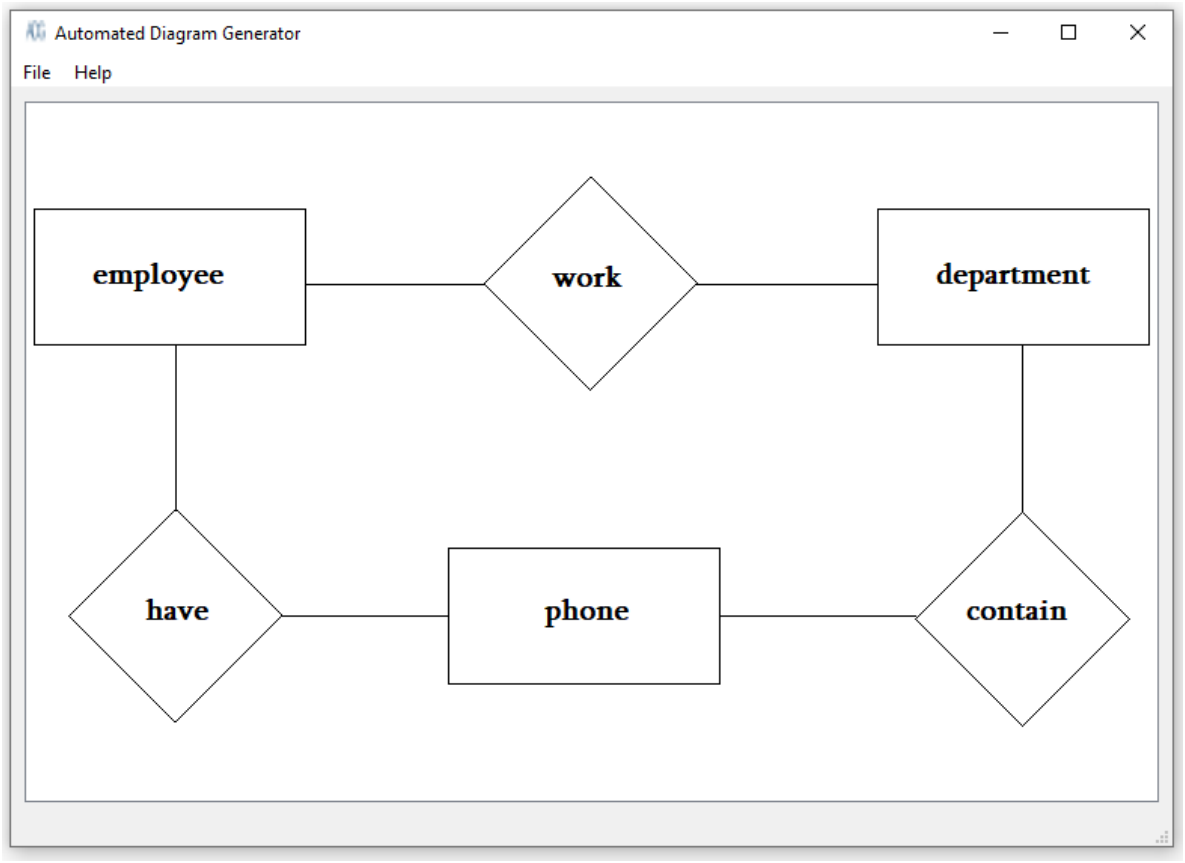

Figure 5.8: Generated ERD

5.4 Case Study 2 (Valid Software Requirements)

In this case study, we are using software requirements for UPS which is an information system for having up-to-date processing of data for the current location of each shipped item. which are based on entities and relationships? This example shows how the entities are extracted from requirements and make the relationships with each other and also it shows the attributes as well. For that, we repeat a similar kind of steps that are used in a previous example.

5.4.1 Insert Software Requirements

Same as before we insert the valid software requirements. These are:

"UPS prides itself on having up-to-date information data on the processing and current location of each shipped item. To do this, UPS relies on a company-wide information system. Shipped items are the heart of the UPS product tracking information system. Shipped items can be characterized by item number, weight, dimensions, insurance amount, destination, and final delivery date. Shipped items are received into the UPS system at a single retail center. Retail centers are characterized by their type, ID, and address. Shipped items are sent to their destination via standard UPS transportation events. These transportation events are characterized by a schedule number. a type and a delivery route."

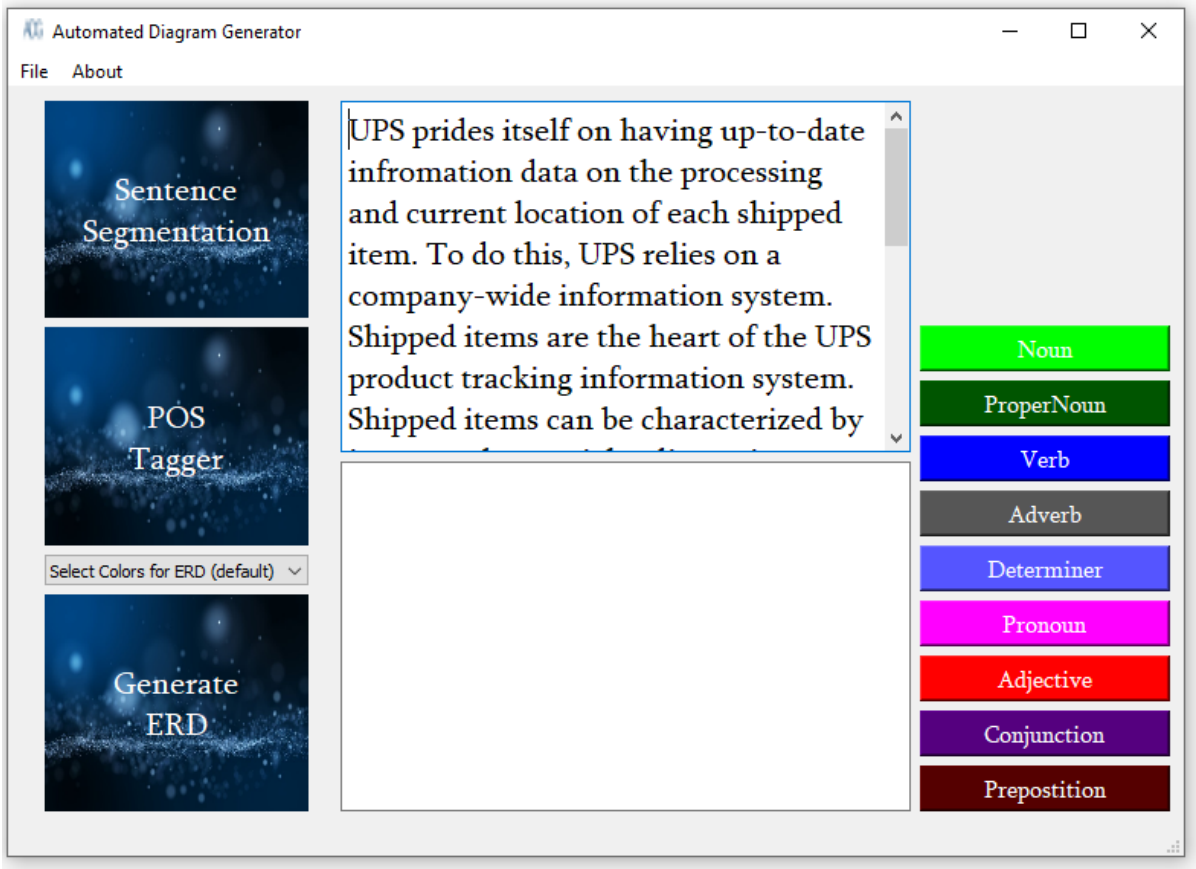

Figure 5.9: Insert Software Requirements

5.4.2 Sentence Segmentation (Tool)

This tool converts the input requirements into sentences and removes the ambiguity similar to in section 5.3.2.

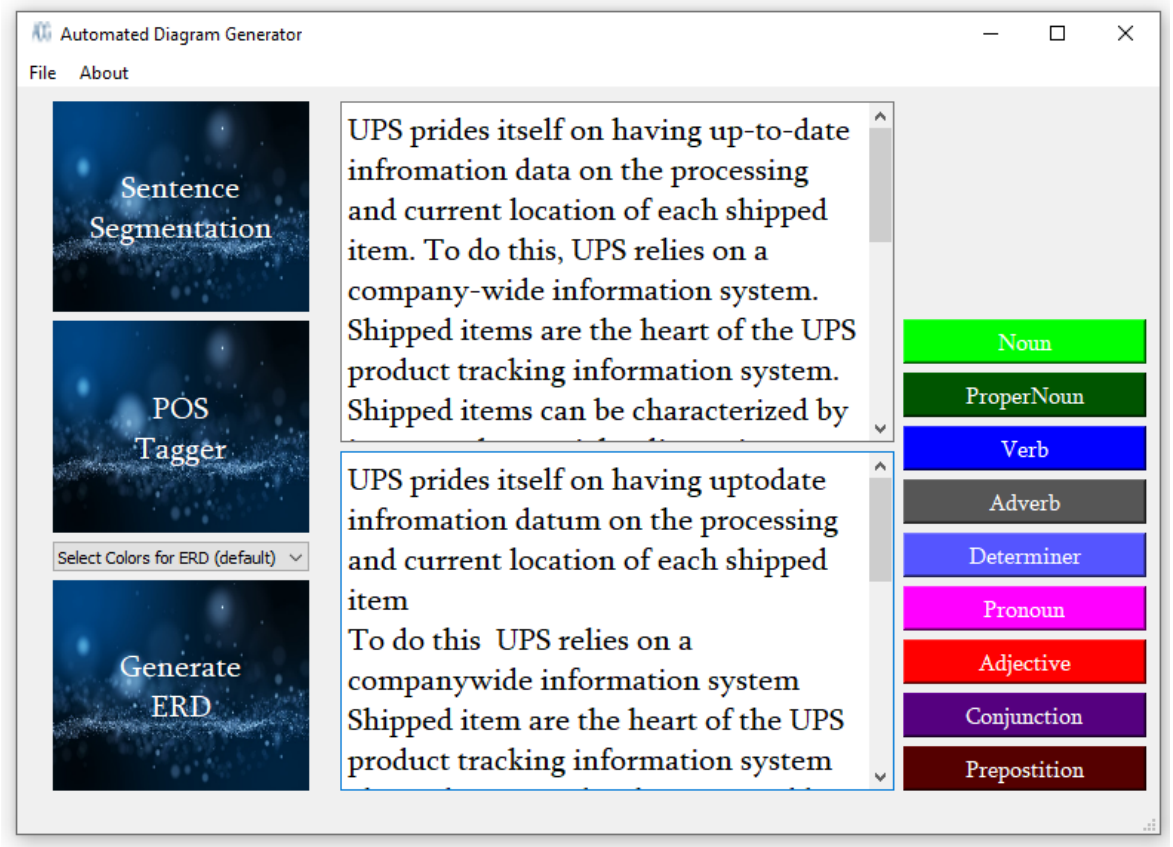

Figure 5.10: Sentence Segmentation

5.4.3 POS Tagger (Tool)

POS tagger the tokenize of the sentences and give them POS tags as well as specific colors just as in section 5.3.3.

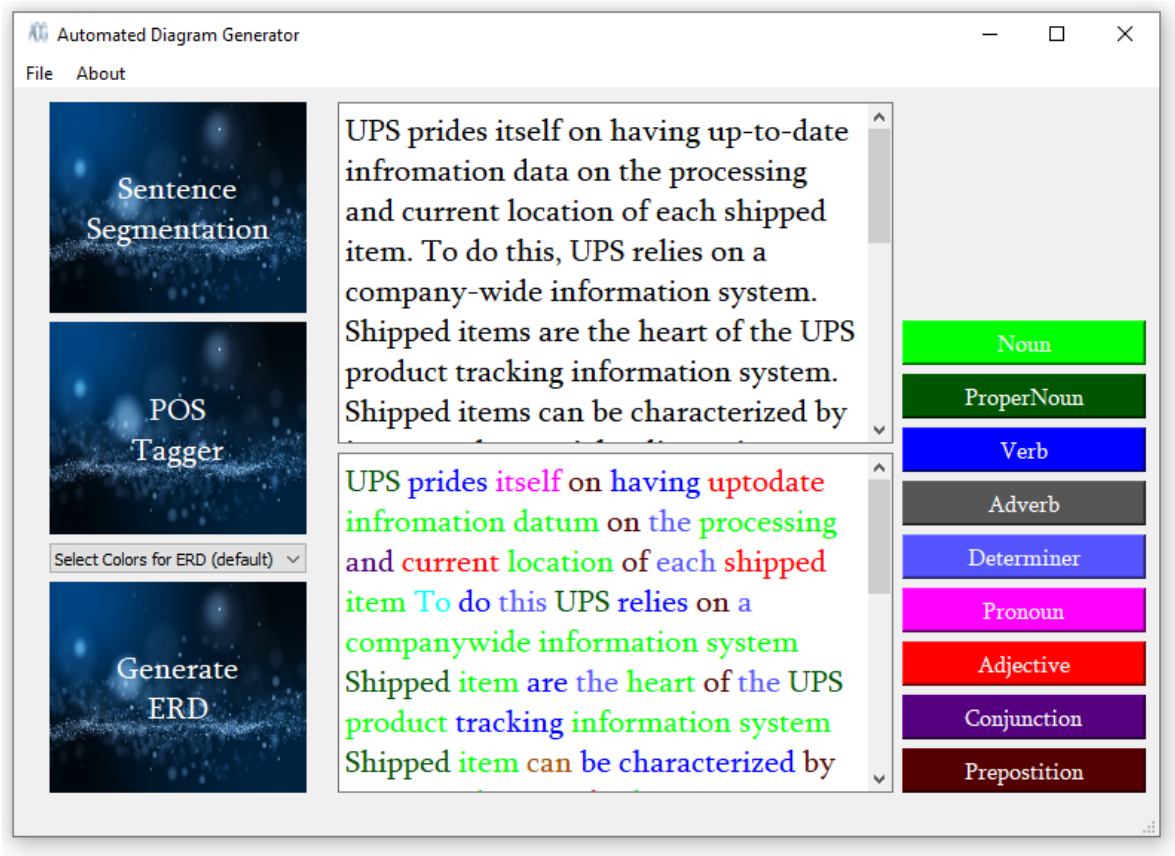

Figure 5.11: POS Tagger

5.4.4 Choose Specific Part of Speech

Select a specific POS tag by selecting the desired bar. In this case, we select “Verb” so it displays the result in the resultant text field with blue color.

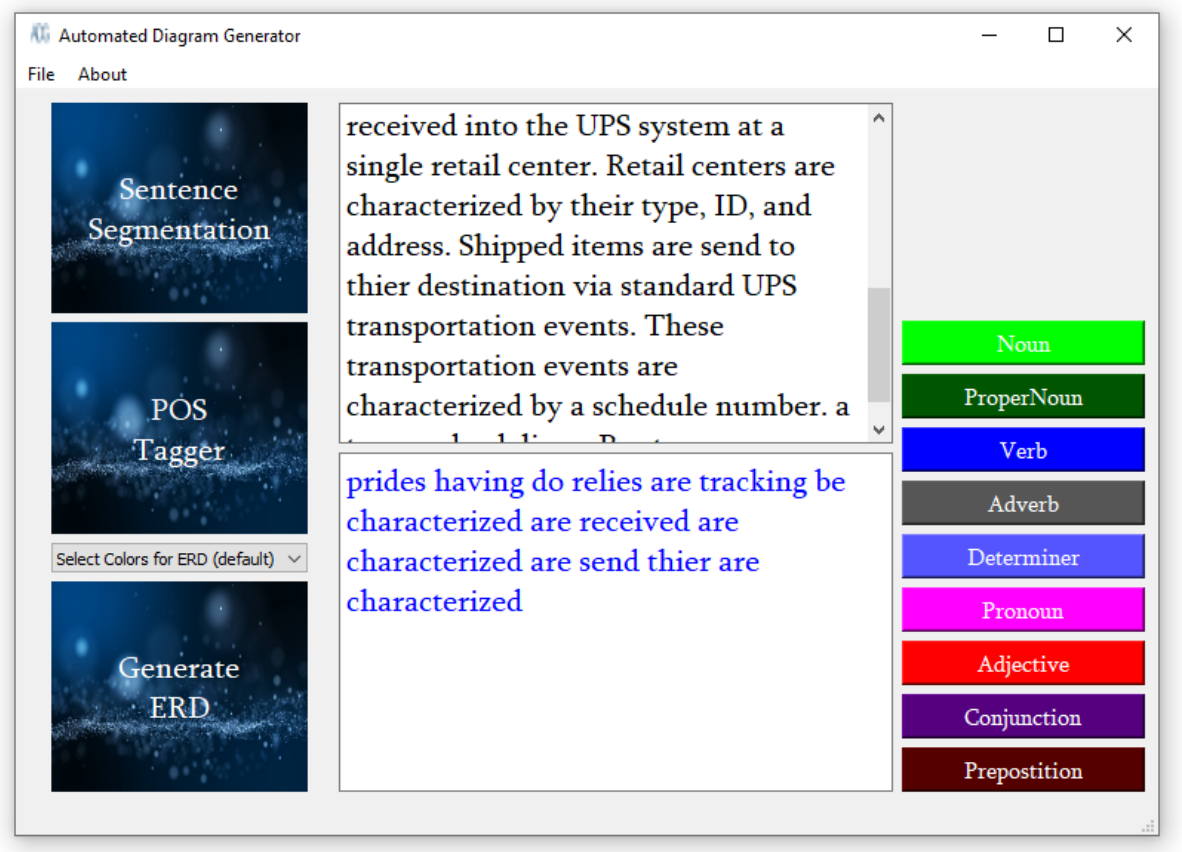

Figure 5.12: Choose Specific POS Tag (Verb)

5.4.5 Choose a color pattern for ERD

Select a color pattern for ERD in this case we select the “Black and White with Grey” in which entities are filled with grey color.

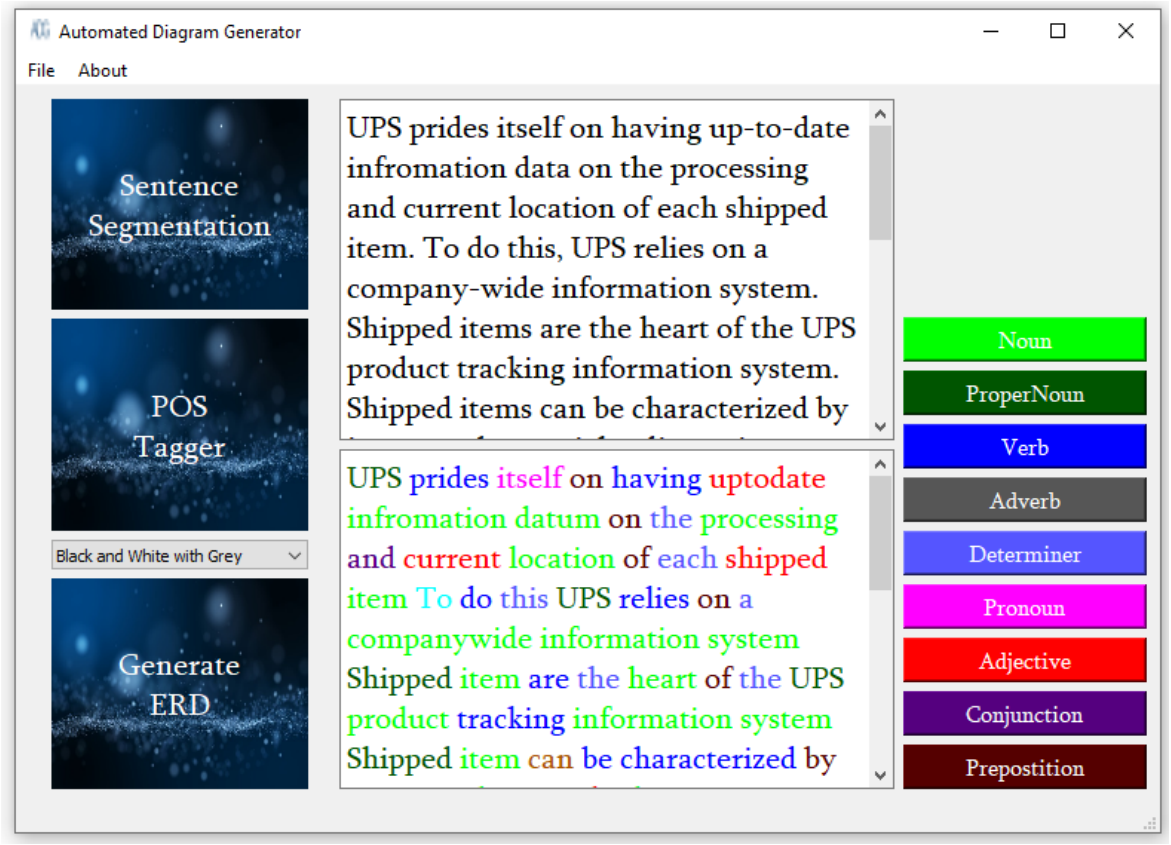

Figure 5.13: Select Black and White with Grey Style

5.4.6 Generated ERD

Generated the ER diagram in black and white with a grey pattern. This case study takes 3 to 4 seconds to generate an ERD.

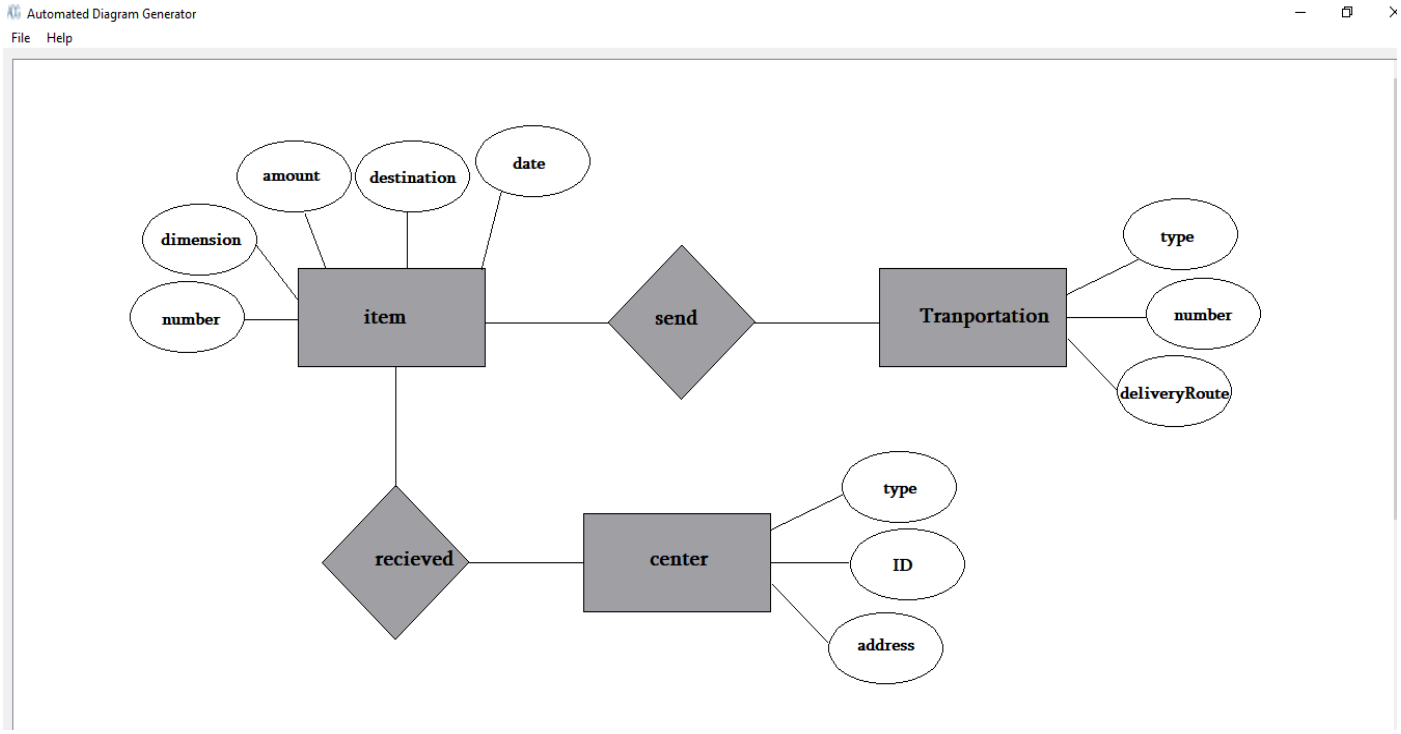

Figure 5.14: Generated ERD

From the above layouts, we show the results of our software system that generates an ERD by integrating the POS Tagger and Sentence Segmentation. It also showed some other functionality like choosing a pattern and choosing a specific part of speech tags. Processing time for generating a diagram depends on the length of the case study. A case study with words up to 50 will take a fraction of a second. Similarly case study with words in the range of 100 to 200 will take 2 to 3 seconds of processing time.

5.5 Evaluation

Generating Entity-Relationship Diagram with software requirements quite easy but to check and evaluate the results as output either correct or not is the main problem or deficiency where most of the automated system failed. To evaluate the results we have used precision and recall formulas to check the accuracy level or by other means to check accuracy by taking expert feedback on consecutive extracted or generated ER Diagrams from ADG.

Table 5.1: Case Studies Confusion Matrix

Case Studies	Features	True Positive	False Positive	False Negative	True Negative
Sample 1	Entities	3	0	0	3
	Attributes	11	0	1	12
	Relationships	3	0	0	3
Sample 2	Entities	3	0	0	3
	Attributes	8	0	1	9
	Relationships	3	0	0	3
Sample 3	Entities	3	0	1	4
	Attributes	10	1	2	13
	Relationships	3	0	0	3
Sample 4	Entities	3	0	1	4
	Attributes	13	1	2	16
	Relationships	3	0	1	4
Sample 5	Entities	3	1	1	5
	Attributes	12	1	3	16
	Relationships	3	1	1	5

To determine the accuracy we use the formulas of precision (TP/TP+FP), recall (TP/TP+FN), and accuracy (TP+TN/TP+TN+FN+FP). The accuracy level checked basis on elements of ER Diagram which are now part of data to be processed by the algorithms.

Table 5.2: Accuracy of Case Studies

Case Studies	Recall	Precision	Accuracy
Sample 1	94.44	100	97.22
Sample 2	93.33	100	96.66
Sample 3	84.21	94.11	90.00
Sample 4	82.60	95.00	89.58
Sample 5	78.26	85.71	84.61

In the above table 5.2, we have samples of five different case studies and their complexity will be increase with their numbers. As in the first, we have 97% accuracy just because of their text (software requirement) is short. The issue arises when we have a large English text valid software requirements then we probably fail to maintain accuracy factor because extraction from a large scale or complex software requirements can be difficult as there is still a lot of work that is needed to handle the extraction of such case studies. In the future, we maintain our accuracy for large text software requirements also to fully reduce the user intervention.

CHAPTER 6

CONCLUSION AND RECOMMENDATIONS

6.1 Future Work

In ADG automated diagram generator future work will be extended to work on cardinality [14]. More accurate our first step is software requirement which is to be suitable for the text area. The most important factor is accuracy in the generation of ER-Diagram it is not possible without a verification system that encounters the software requirement [17].

This could be possible from heuristic rules but for a short project or short software specification. For this Artificial intelligence (AI) and Machine learning is used for a better understanding of software requirements. For nouns, we used SBVR (Semantic Business Vocabulary Rules) the extraction method of elements of ER Diagram [12][2].

6.2 Conclusion

Design ER Diagram from software requirement specification. Entity-Relationship Diagram is a high-level technique to represent a relational database as visually. Extract the meaningful information from software requirements without user intervention is complex but the heuristic rule may help for short projects. In Automated Diagram Generator ADG, we generate the ER-Diagram automatically without user involvement.

The methodology used in Automated Diagram Generator (ADG), software requirements splits into sentences then tokenization applies to transformed sentences which are further assigned tags by POS-tags. User can pick their desire POS-Tags which are available side-by-side to this above Software tool. POS-Tags preliminary go to its constituent parser. Parsing is the main module to generate the ER diagram which is a combination of POS-tags, Tokens, and heuristic rules. in the generation of ER-Diagram, two formats will be used “Black and White” or “Black and Grey”. The generation of ERD according to time varies as per the length of software requirements.

REFERENCES

[1] E. S. Btoush and M. M. Hammad, “Generating ER Diagrams from Requirement Specifications Based On Natural Language Processing,” *Int. J. Database Theory Appl.*, vol. 8, no. 2, pp. 61–70, 2015, doi: 10.14257/ijdta.2015.8.2.07.

[2] L. Angeles, “PETER PIN-SHAN CIIEN The ER diagram was formally proposed in [4]. Figure l (a) is an example of a simple ER diagram. The rectangular-shaped boxes represent entity types, and the diamond-shaped boxes represent relationship types. For example, in Fi,” vol. 149, pp. 127–149, 1983.

[3] P. More and R. Phalnikar, “Generating UML Diagrams from Natural Language Specifications,” *Int. J. Appl. Inf. Syst.*, vol. 1, no. 8, pp. 19–23, 2012, doi: 10.5120/ijais12-450222.

[4] S. D.Joshi and D. Deshpande, “Textual Requirement Analysis for UML Diagram Extraction by using NLP,” *Int. J. Comput. Appl.*, vol. 50, no. 8, pp. 42–46, 2012, doi: 10.5120/7795-0906.

[5] S. Gulia and T. Choudhury, “An efficient automated design to generate UML diagram from Natural Language Specifications,” *Proc. 2016 6th Int. Conf. - Cloud Syst. Big Data Eng. Conflu. 2016*, pp. 641–648, 2016, doi: 10.1109/CONFLUENCE.2016.7508197.

[6] P. G. T. H. Kashmira and S. Sumathipala, “Generating Entity Relationship Diagram from Requirement Specification based on NLP,” *2018 3rd Int. Conf. Inf. Technol. Res. ICITR 2018*, pp. 1–4, 2018, doi: 10.1109/ICITR.2018.8736146.

[7] N. Omar, P. Hanna, and P. McKevitt, “Heuristics-based entity-relationship modeling through natural language processing,” *Proc. 15th Artif. Intell. Cogn. Sci. Conf.*, pp. 1–12, 2004.

[8] T. De Smedt, V. Van Asch, and W. Daelemans, *Memory-based Shallow Parser for Python*. 2010.

[9] D. K. Deeptimahanti and M. A. Babar, “An automated tool for generating UML models from natural language requirements,” *ASE2009 - 24th IEEE/ACM Int. Conf. Autom. Softw. Eng.*, pp. 680–682, 2009, doi: 10.1109/ASE.2009.48.

[10] I. Song and M. Evans, “A Comparative Analysis of Entity-Relationship Diagrams 1 1 INTRODUCTION,” *Computer (Long. Beach. Calif).*, vol. 3, no. 4, pp. 427–459, 1995.

[11] N. Madnani, “Getting started on natural language processing with Python,” *XRDS Crossroads, ACM Mag. Students*, vol. 13, no. 4, pp. 5–5, 2007, doi: 10.1145/1315325.1315330.

[12] N. Sarwar and I. S. Bajwa, “Automated Generation of Express-G Models Using Nlp.,” *Sindh Univ. Res. Journal-SURJ (Science Ser.*, no. March 2016, pp. 4–12, 2016.

[13] S. Geetha and G. S. Anandha Mala, “Automatic database construction from natural language requirements specification text,” *ARPN J. Eng. Appl. Sci.*, vol. 9, no. 8, pp. 1260–1266, 2014.

[14] M. Ibrahim and R. Ahmad, “Class diagram extraction from textual requirements using natural language processing (NLP) techniques,” *2nd Int. Conf. Comput. Res. Dev. ICCRD 2010*, pp. 200–204, 2010, doi: 10.1109/ICCRD.2010.71.

[15] F. Hogenboom, “An Overview of Approaches to Extract Information from Natural Language Corpora,” *Language (Baltim).*, 2010.

[16] R. Sajjad and N. Sarwar, “NLP based verification of a UML class model,” *2016 6th Int. Conf. Innov. Comput. Technol. INTECH 2016*, no. August 2016, pp. 30–35, 2017, doi: 10.1109/INTECH.2016.7845070.

[17] M. Elallaoui, K. Nafil, and R. Touahni, “Automatic Transformation of User Stories into UML Use Case Diagrams using NLP Techniques,” *Procedia Comput. Sci.*, vol. 130, pp. 42–49, 2018, doi: 10.1016/j.procs.2018.04.010.

APPENDICES

APPENDIX A: Design Phase List

Actor: Represent role, outside of system interdepend with the system.

Use Cases: Represent interaction between system and actor.

Use Case Diagram: Represent the use of the functionality of the system by the actor.

Sequential Diagram: It is used to show a conversation between the objects (like button and text field) to carry out the functionality.

Domain Model: It is a model that integrates behaviour and data and shows the workflow of classes with relations.

Class: Describe the group related object having the same attributes, operations, relationships, etc.

Class Diagram: Describe the generalized modelling process for the building of the application or system.

Entity: It is considered as an object or a data component.

Attribute: These are assets of an entity, characterized by ellipses.

Relationship: It relates to the interaction between the entities.

Relational Data Model: It the combination of entity and attribute among there relationship. Also include the cardinalities.

APPENDIX B: Development Phase List

Python: Used as a development language

PyCharm: It is an IDE used to perform backend work in python

PyQt5: It is a toolkit used for design UI with its designer tool as well as designing the ERD with the help of different shapes.

NLTK: Famous toolkit for natural language processing used to perform for its techniques.

Button: It is used to perform the functionality by clicking on it.

Text Field: It is used to insert or type of software requirement.

Combo box: It is used for selecting one option from various choices.

Shapes: Shapes like rectangle, ellipse, and diamond are used to represent ERD.

Rectangle: It is used to represent an entity in the generated ERD.

Ellipse: It is used for attributes in the generated ERD.

Diamond: It is used for relationships in the generated ERD.