University of the Philippines Manila
College of Arts and Sciences
Department of Physical Sciences and Mathematics

NERD: A Learning Tool for Database Normalization and ERD Development

A special problem in partial fulfillment
of the requirements for the degree of
Bachelor of Science in Computer Science

Submitted by:

Kim Isle G. Cortez

April 2014

Permission is given for the following people to have access to this SP:

Available to the general public	Yes
Available only after consultation with author/SP adviser	No
Available only to those bound by confidentiality agreement	No

ACCEPTANCE SHEET

The Special Problem entitled "NERD: A Learning Tool for Database Normalization and ERD Development" prepared and submitted by Kim Isle G. Cortez in partial fulfillment of the requirements for the degree of Bachelor of Science in Computer Science has been examined and is recommended for acceptance.

Gregorio B. Baes, Ph.D. (*candidate*)
Adviser

EXAMINERS:

	Approved	**Disapproved**
1. Avegail D. Carpio, M.Sc.	_____	_____
2. Richard Bryann L. Chua, Ph.D. (*candidate*)	_____	_____
3. Aldrich Colin K. Co, M.Sc. (*candidate*)	_____	_____
4. Ma. Sheila A. Magboo, M.Sc.	_____	_____
5. Vincent Peter C. Magboo, M.D., M.Sc.	_____	_____
6. Geoffrey A. Solano, M.Sc.	_____	_____
7. Bernie B. Terrado, M.Sc. (*candidate*)	_____	_____

Accepted and approved as partial fulfillment of the requirements for the degree of Bachelor of Science in Computer Science.

Ma. Sheila A. Magboo, M.Sc.
Unit Head
Mathematical and Computing Sciences Unit
Department of Physical Sciences
and Mathematics

Marcelina B. Lirazan, Ph.D.
Chair
Department of Physical Sciences
and Mathematics

Alex C. Gonzaga, Ph.D., Dr.Eng.
Dean
College of Arts and Sciences

Abstract

Data modelling has been a complicated task for students taking up a database course as well as for beginner developers. However, data modelling should not be ignored by people in a database related course or profession as it is required to develop a well-structured database. Although data modelling, ER modelling in particular, is a complicated task, it is not well provided with tool based support [7]. Several model development tools exist but learning tools for data modelling are very minimal in number.

This study intends to develop a computer-aided instruction (CAI) for the development of entity-relationship diagrams and normalization.

Keywords: CAI, database, modelling, ERD, normalization, NLP

Contents

Acceptance Sheet **i**

Abstract **ii**

List of Figures **v**

I. Introduction **1**

A. Background of the Study 1

B. Statement of the Problem 2

C. Objectives of the Study 3

D. Significance of the Project 4

E. Scope and Limitations 4

F. Assumptions 7

II. Review of Related Literature **8**

III. Theoretical Framework **11**

A. Entity-Relationship Diagram 11

B. Normalization 15

C. Computer-Aided Instruction 17

D. Data Modelling 18

E. Database Problem Specification 18

F. Natural Language Processing 19

G. Stanford Parser 19

H. Typed Dependencies of the Stanford Parser 23

I. Heuristics Rules 24

IV. Design and Implementation **27**

A. System Modules Overview 27

B. System Input Validation and Sentence Classification 29
C. Validation for Conflicting Requirements 30
D. Final ERD Representation 33
E. Component Extraction from Sentences using Heuristic Rules . . . 34

V. Architecture **36**
A. System Architecture 36
B. Technical Architecture 36

VI. Results **37**

VII. Discussions **55**

VIII. Conclusions **59**

IX. Recommendations **60**

X. Bibliography **61**

XI. Appendix **64**
A. Source Code 64

XII. Acknowledgement **127**

List of Figures

1 Examples of Entities 11

2 Example of Entities with Attributes 12

3 Example of Entities with Relationships 12

4 Types of Cardinalities 13

5 Example of Disjointness Constraint 14

6 Example of Completeness Constraint 15

7 POS Labels used by Stanford Parser 21

8 Typed Dependencies Generated From Example 22

9 Typed Dependency Structure of Stanford Parser 23

10 Graphical representation of rule format 24

11 System Flowchart 28

12 Regular Expressions for Sentence Validation 30

13 ERD Rendering and Representation 33

14 Home screen, NERD 37

15 Main editor, NERD 37

16 Generating ERD from text module menus, NERD 38

17 The user manual, NERD 38

18 Saving a text file, NERD 39

19 Loaded text file from file system, NERD 39

20 Add word to dictionary, NERD 40

21 Word added to dictionary, NERD 40

22 Word not added to dictionary, NERD 41

23 Sentence format error, NERD 42

24 Type 1 conflict found, NERD 42

25 Type 2 conflict found, NERD 43

26 Type 3 conflict found, NERD 43

27 Missing cardinalities in the requirement, NERD 44
28 Final ERD output, NERD 45
29 Save ERD functionality, NERD 45
30 Save as image functionality, NERD 46
31 Open ERD functionality, NERD 46
32 Parse string functionality, NERD 47
33 Clear functionality, NERD 47
34 Step by step module step 1, NERD 48
35 Step by step module step 2, NERD 48
36 Step by step module step 3, NERD 49
37 Step by step module step 4, NERD 49
38 Step by step module step 5, NERD 50
39 Normalization module, NERD 51
40 Normalization module with saved relation, NERD 51
41 Attribute not in functional dependency error, NERD 52
42 Normalization results, NERD 52
43 Normalization step by step process, NERD 53
44 Lesson module, NERD 53
45 Exam module, NERD 54

I. Introduction

A. Background of the Study

Databases today have become indispensable to any business function carried out by organizations [1]. Organizations deem it necessary to organize data and information through the use of database systems. Over time, business database systems have been proven useful and reliable in organizations in terms of storage and manipulation of bulk data [2].

One prerequisite to having a complete and working database system is the development of the database architecture. A critical consideration in crafting a robust database solution is the level of optimization which can be ensured through a well-defined design [3]. Having a good database structure means having a good way of organizing and managing data. A good database structure reduces anomalies or errors done when performing database operations.

An integral part of database design is the accurate modelling of the components of a large enterprise as a relational database, which can be accessed and updated efficiently by a large number of concurrently executing transactions which may include decision support queries [4]. Hence, data modelling and database normalization are required to develop a database design for a good database structure. Data modelling is the solution to the existing need for a unified file structure and database design among organizations [1].

Several data modelling techniques have been proposed and used over the years. These include network model, relational model, entity set model, and entity relationship model also known as the ER model, one of the more popular models proposed by Peter Chen in 1976 [5].

Along with the increased use of database systems and the necessity of data modelling is the emergence of several applications developed to render a more efficient

way of creating data models and normalizing databases. Among these are MS Visio, RDBNorma, and some other web-based applications. These tools, however, specialize in model creation only, and do not cover user instruction. The existing data modelling applications are indifferent to the users' knowledge in creating models and databases.

B. Statement of the Problem

Data modelling is the basis of a well-structured database [6]. Hence, the model would dictate how the database would perform. A well created model would result to a well-structured database.

Data modelling is a complicated task that involves identification of relevant facts from different information sources [7]. It has been a complex task for students taking database courses as well as for novice developers. Although ER modelling is a complicated task, it is not well provided with tool based support [7]. Tools exist like MS Visio and the like, but these tools aid in development and do not enhance learning of the user. However, having a tool for data modelling would be useless if one cannot create models in the first place. Given that there exists a lot of data modelling tools, a software intended for learning data modelling is much needed these days.

The existence of other learning materials regarding the topic is not enough for students to fully comprehend the topic. Even though there exist other learning tools like online tutorials, these tools have fixed lessons. Students or anyone who access the same tutorial would still see the same learning material instead of having new examples, so students tend to view solutions or explanations to the same problem they have already understood.

C. Objectives of the Study

General objective:

This study aims to create a computer-aided instruction (CAI) on the development and normalization of entity-relationship diagrams (ERD).

Specific Objectives:

This CAI on the creation and normalization of ERD has the following functionalities:

1. Allows the student to:
    (a) Input database problem specifications as paragraph to be solved by the system. Database problem specifications input must follow a certain structure which will be detailed out in section E.
    (b) View stored lessons. There are stored lessons in the system which is accessible and can be read by every student. These are how-to lessons, teaching students on how to create ERD.
    (c) Answer questions provided by the system. The system contains a set of questions or problems given to students in order to assess the student's skills and learning of the topic.
2. The application can:
    (a) Accept a database problem specification as a string or paragraph to be processed.
    (b) Parse the set of input strings or paragraph.
    (c) Extract ERD constructs from parsed input strings or paragraph.
    (d) Connect the ERD constructs extracted from parsed input into separate relationships.

(e) Generate a complete and connected ERD from the input provided by the student.

(f) Accept a set of functional dependencies for the normalization process.

(g) Generate a normalized ERD using the provided functional dependencies.

(h) Save and retrieve files containing the original input and the output for the given input.

(i) Edit or manipulate saved files.

(j) Read a saved ERD file and generate a set of sentences that descibes it.

D. Significance of the Project

Given that data modelling is a complex task, the outcome of this study would contribute to a better understanding of the topic and will make learning of data modelling and normalization easier.

This study will also improve the skills and knowledge of students undertaking a database course, beginner developers, or anyone engaged in a database related profession. The results of this study will be useful for them to build a well-structured database.

Moreover, the outcome of this study will aid instructors in their teaching of ERD construction and normalization. It will allow instructors to use a new way of teaching students instead of traditional lectures and face-to-face discussions.

E. Scope and Limitations

1. Users can input database problem specifications in paragraph form to be solved by the system.

2. Users can read lessons and answer exams provided by the system.

3. The system can only translate database problem specifications to an ERD normalized up to third normal form (3NF).

4. The system only includes lessons for learning ER modelling and normalization.

5. The system does not include instruction for other data models such as network model, UML, DFD, and the like.

6. The ERD constructs used in the system is limited to entities, relationships, cardinality, attributes, generalization/specialization, completeness constraints, and disjointness consstraints.

7. The system does not include ERD clustering.

8. The student must indicate relationships and cardinality constraints in two way.

9. The system draws the ERD in crow's feet notation. Other ERD notations like Chen notation is not included.

10. Words used in the input not found in the system dictionary (simple txt file or words) raises an error.

11. The relation input in the normalization module must be bar delimited.

12. The system only accepts the following sentence formats as input:

    (a) Type 1 Format.

        This format indicates a relationship with an entity and its attributes. The *subject* corresponds to the entity. The *objects* correspond to the attributes. The inclusion of the keyword **unique** or **distinct** indicates whether the attributes must be primary key or not.

        {*subject*} **has a** [**unique** | **distinct**] {*object*} [[**,**{*object*} ...] **and** {*object*}]**.**

Examples:

An instructor has a unique id.

An instructor has a name and a salary.

(b) Type 2 Format.

This format indicates a relationship between two entities, as well as the cardinality between them. The *subject* corresponds to the entity. The *objects* correspond to the other entities related to the *subject*. The *verb* corresponds to the relationship between entities. The cardinality indicates the cardinality from *subject* to *objects*.

{*subject*} {**must** | **may** | **might** | **can** | **could** | **should**} {*verb*} {**at most one** | **zero or one** | **only one** | **at least one** | **one or many** | **zero or many**} {*object*} [[,{*object*} ...] **or** {*object*}]**.**

Examples:

An employee must belong to at least one department.

A company must have at least one branch or department.

(c) Type 3 Format.

This format indicates a relationship with an entity and its subtypes. The *subject* corresponds to the entity. The *objects* correspond to the subtypes of this entity. The inclusion of the keyword **only** indicates whether the specialization is total or partial.

- {*subject*} **can be either** {*object*} [[,{*object*} ...] **or** {*object*}] [**only**]**.**
  This variation of type three format indicates that the specializations are disjoint. Because of the keyword **either**, the subtype must be just one of the *objects*, hence indicating disjointness.
  Examples:

  A patient can be either an inpatient or outpatient.

An art can be either a sculpture or a painiting.

- {*subject*} **can be** {*object*} [[**,**{*object*} ...] **or** {*object*}] [**only**]**.**

  This variation of type three format indicates that the specializations may overlap. Because of the connector **or**, the subtype must be just one or more or all of the *objects*, hence indicating overlap.

  Examples:

  An employee can be a project manager or team leader.

  A faculty can be a dean or a professor.

F. Assumptions

1. Sentences in database problem specifications:

   - contain words found in the dictionary,
   - do not use first, second, and third person pronouns,
   - use common nouns and not proper nouns,
   - are not in question form,
   - only have a single subject,
   - follow the sentence formats discussed in section E of chapter I

2. Every term in each sentence of database problem specifications must be used consistently throughout the whole specification. User must not use different terms referring to the same meaning especially on nouns.

3. The student would not delete the lessons in the lessons folder. If the student wish to add more lessons, the lesson must be added directly to the specified lessons folder.

II. Review of Related Literature

Several tools are implemented in the field of computer science which provide instruction and development assistance. Also, there have been exploits which are related to data modelling and database normalization.

Virvou et al. developed ASSET, which served as an adaptive training tool for learning UML. ASSET provides its users with an environment to construct UML diagrams. Also, ASSET provides users with adaptive support in order to assist them to achieve learning of UML modelling. ASSET has a daemon process which monitors every action performed by its student, and gives feedback depending on user expertise. ASSET only caters training for learning UML diagrams. [8]

Soler et al. developed ACME-DB, a web-based e-learning tool for UML diagrams. ACME-DB integrates a UML diagramming tool with an automated assessment for its students, automatic correction of exercises, and automated feedback from the system. However, similar to ASSET, ACME-DB only provides learning of UML diagrams. [9]

Suthers et al. developed COLER, a coached collaborative learning environment for entity relationship modelling. Users of COLER construct individual ER diagrams as a solution to a given database problem. Afterwards, users work in small groups to come up with a collaborated solution. However, the system does not provide solutions to problems. COLERs purpose is to help its users develop group collaboration and critical thinking skills. [10]

Suraweera developed KERMIT, a knowledge-based intelligent tutor for ER modelling. KERMIT provides a set of requirements, and then the students would construct ER models satisfying the problem. The system already has an ideal solution for each of its problems. Students' answers on the problems are evaluated by using the system's domain knowledge represented as a set of constraints. KERMIT is more of a problem solving environment wherein the system provides feedback and assistance to its users as they solve a problem. [6]

Kung et al. developed a web-based tool for teaching database normalization. This tool generates a normalized database given its set functional dependencies (FD) provided by the user. This tool provides automated database normalization up to the third normal form (3NF). The tool provides a step-by-step evaluation of the input, showing the users a partially normalized database at every step. [11]

Dongare et al. developed RDBNorma, a tool for relational database schema normalization. This tool performs database normalization up to 3NF. The same as other normalization tools, the user must input a list of FDs. This tool uses a single linked list to store relations and FDs. This tool is proven to use less space and to perform faster compared to other normalization tools. [2]

Shahbaz et al. developed a tool for automatic extraction of extended ER models using natural language processing. Users will input a set of database specifications in English sentences, and the system will generate the corresponding ER model. Shahbaz et al. extended the rules proposed by Chen in [12] to include rules for extended ER diagrams. [1] The tool only develops EERD and does not include the normalization process.

Omar et al. developed a tool for generating ER models through natural language processing called ER-Converter. Also, they proposed new heuristics for translating language constructs to ER models. Omar et al. generalized and improved the rules proposed by Chen in [12] to come up with different heuristics for translating language constructs to ER models. This tool was shown to have a higher recall rate and precision in translating problem requirements to ER models. [13]

Du proposed a new set of heuristics for generating conceptual models through natural language processing. Du related the proposed heuristics to the heuristics proposed in [12] and [13], on how these rules would process similar requirement inputs. Du came up with a different set of heuristics for extracting entities and relationships and classified each as link type, or typed dependency. Du addressed some of the issues

unanswered by other heuristics proposed in [12] and [13]. [14]

Many studies were conducted to provide tool-based support for computer science related topics particularly ER modelling and normalization. However, these existing tools only provide assistance to either data modelling or database normalization, even though these two tasks are very much related. A tool is needed to connect this gap between ER development and database normalization, and to help users understand how to perform these complicated tasks.

III. Theoretical Framework

A. Entity-Relationship Diagram

An Entity Relationship Diagram (ERD) is a major data model that helps organizing data into entities and defines relationships between entities. This data model has proved to enable the analyst to produce and build a well-structured database so that the data can be stored and retrieved in an efficient manner.

By using a graphical format, it may enhance communication of the design between the designer and the user, and the designer and the people who will implement it.

An ERD has the following fundamental components:

- Entity

  An entity is anything real or abstract about which we want to store data. These are principal data objects from which information is to be collected. A particular occurence of an entity is called an entity instance or entity occurence. Entity types can be roles, events, locations, tangible things or concepts. Specific examples of an entity are called instances. E.g. employee John, Smith's payment, etc.

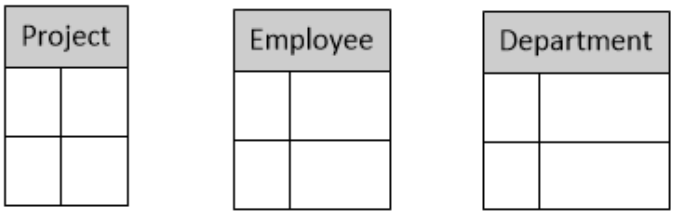

Figure 1: Examples of Entities

- Attribute

  A data attribute is a characteristic common to all or most instances of an entity

which provide descriptive details about them. Synonyms include property, data element or field. E.g. Name, address, Employee ID.

There are two types of attributes: identifiers and descriptors. An identifier(or key) is used to uniquely identify one and only one instance of an entity. E.g. Employee ID is a primary key for Employee, since Employee ID uniquely identifies an employee. A descriptor is used to specify a non-unique characteristic of a particular entity instance.

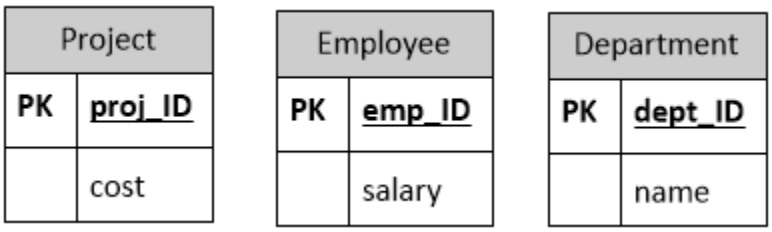

Figure 2: Example of Entities with Attributes

- Relationship

  A relationship is a natural or real-world association that exists between one or more entities, and as such, have no physical or conceptual existence other than that which depends upon their entity associations. E.g. Employees *work on* a project. *work on* is the relationship relating employees to a project.

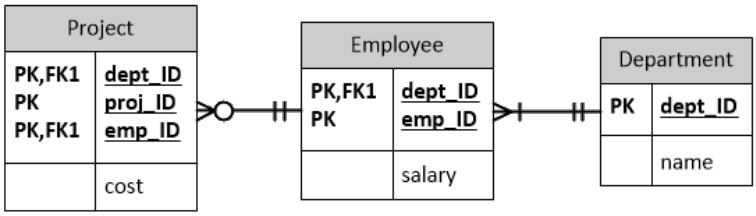

Figure 3: Example of Entities with Relationships

- Cardinality

  The cardinality defines the number of occurrences of one entity for a single occurrence of the related entity. E.g. an employee may or may not be assigned to a project.[15]

  Cardinality constraint indicates the minimum and maximum number of instances of one entity that can or must be associated with each instance of another entity. Cardinality constraint can be optional or mandatory. Optional constraint indicates that a single occurence of an entity may or may not have an association with an instance of another entity. Mandatory constracint indicates that a single occurence of an entity must have an association with at least one or more instances of another entity.

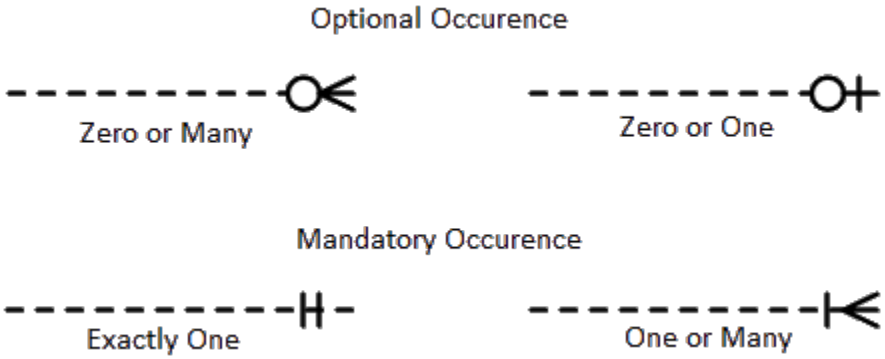

Figure 4: Types of Cardinalities

- Generalization

  The generalization relationship specifies that several types of entities with certain common attributes can be generalized into a higher entity type - a generic or ssuperclass entity, more commonly known as the *supertype*. The lower level of entities aare called *subtypes* of the generalization hierarchy.

  Generalization can further be classified by two important constraints on the subtype entities: *disjointness* and *completeness*.

Disjointness constraint indicates whether an instance of a supertype may simultaneously be a member of several subtypes. Disjointness constraint may either be *disjoint* or *overlap*. Disjoint constraint indicates that an instance of the supertype can only be one of the subtype. Overlap constraint indicates that an instance of the supertype can be more than one of the subtype.

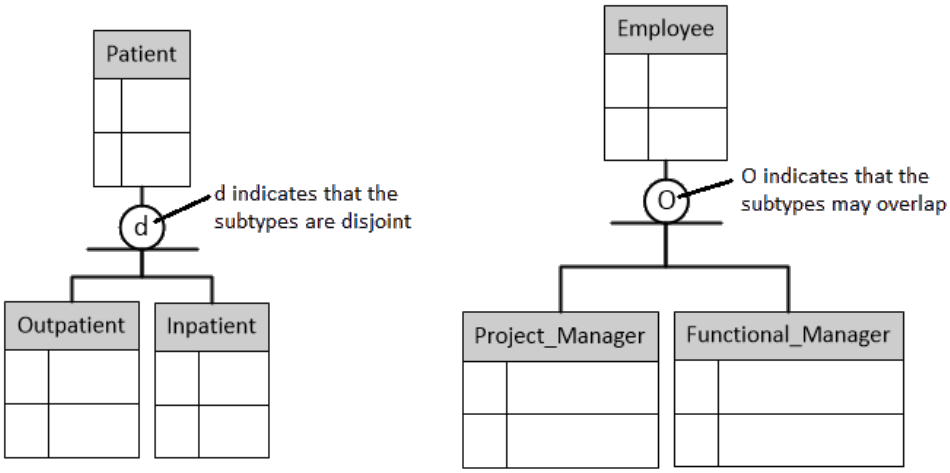

Figure 5: Example of Disjointness Constraint

Completeness constraint indicates whether an instance of the supertype must be a member of at least one subtype. Completeness constraint may either be *total* or *partial*. Total specialization indicates that an instance of the supertype must belong to at least one of the subtypes. Partial specialization indicates that an instance of the supertype may or may not belong to any subtype.

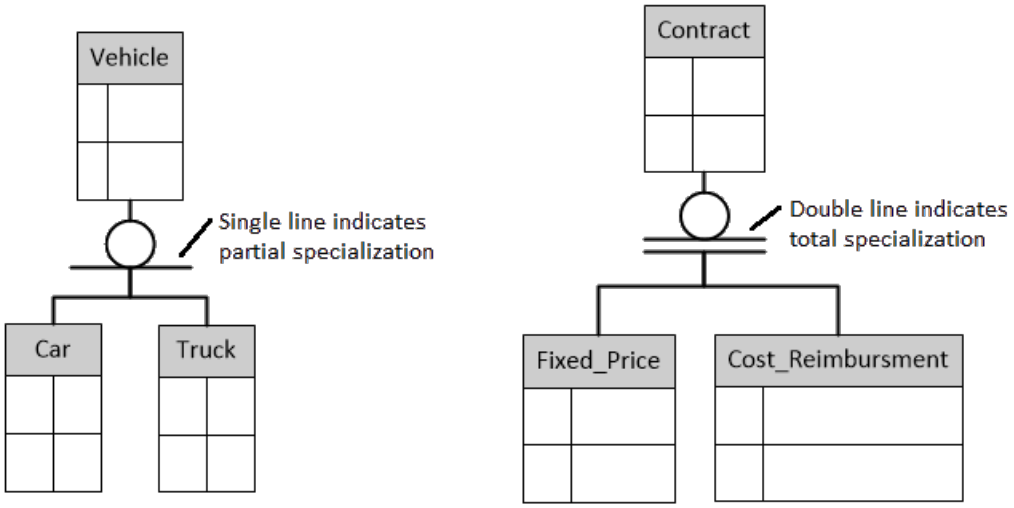

Figure 6: Example of Completeness Constraint

- Specialization

  The specialization relationship, the reverse of generalization, is the inversion of the same concept. It indicates that the subtypes specialize the supertype.

B. Normalization

In relational database design, normalization is the process of data organization to minimize redundancy. Usually, it involves dividing a database into two or more tables and defining relationships between the tables. This process aims to minimize modification problems that could arise after modifying a table field. [16]

Databases have forms established by the database community for ensuring that databases are normalized. These forms are referred to as normal forms, numbered from one, the first normal form, through five, the fifth normal form.

A table is in first normal form (1NF) if there are no duplicate rows in the table, each cell is single-valued, and column entities are of the same kind. A table is in second normal form (2NF) if it is in 1NF and it has no partial dependencies. The table is in third normal form if it is in 2NF and all the non-key attributes are fully

dependent on the primary key, the whole primary key, and nothing but the primary key. The fourth and fifth normal forms are more advanced forms of normalization, and it is commonly accepted that 3NF is adequate on most business needs. [17]

Functional dependency (FD) is a relationship that exists when one attribute uniquely determines another. For example, the FD: X $\longrightarrow$ Y means that X uniquely identifies Y. [18]

Given the universal relation T(A, B, C, D, E, F), and functional dependencies

FD1 : A $\longrightarrow$ B, C, D

FD2 : B $\longrightarrow$ C, D

FD3 : A, E $\longrightarrow$ B, C, D, F,

normalizing a database table to 3NF can be done through the following steps:

1. All functional dependencies on the left-hand side must keep their attributes intact.

2. Extraneous attributes on the right-hand side should be eliminated. This step will eliminate partial and transitive dependencies. Repeated right-hand side attributes should be identified in all the functional dependencies, one copy of the redundant attributes should be kept and the others should be deleted. The rules of thumb about which copy of attributes to keep are:

   - the functional dependencies that have fewer numbers of attributes on the left-hand side should be kept (this eliminates partial dependencies)

     Example: Attributes B, C, and D depend on part of the whole key (attribute A). Attributes B, C, and D in functional dependency FD3 will be deleted, since attributes B, C, and D appear in FD1 and FD1 has only one attribute on the left-hand side. A new functional dependency FD3' : A,E $\longrightarrow$ F is formed.

   - when two FDs have the same number of attributes on the left-hand side,

the attributes that have the fewer numbers of attributes on the right-hand side should be kept (this eliminates transitive dependencies).

Example: Attributes C and D appear in functional dependencies FD1 and FD2 . Attributes C and D are transitively dependent on Attribute A. Attributes C and D will be deleted from functional dependency FD1 , since FD1 has more right-hand side attributes than functional dependency FD2. A new functional dependency FD1': A $\longrightarrow$ B is formed.

3. Construct relations. Convert the functional dependencies without extraneous right-hand side attributes to relations.

   Example:

   The new functional dependencies are as follows:

   FD1': A $\longrightarrow$ B

   FD2 : B $\longrightarrow$ C, D

   FD3' : A, E $\longrightarrow$ F

These steps were stated by Kung in [11].

C. Computer-Aided Instruction

Computer-aided instruction (CAI) or computer-based instruction (CBI) is the presentation of instruction through a computer program to a passive student, or the computer is the platform for an interactive and personalized learning environment. [19]

It refers to any application of computers to teaching, which includes the restricted concept of a tutorial where the computer does all the work of a teacher. Also, it may refer to the concept of "Computer-Supplemented Instruction"(CSI), in which the computer acts as a supplement to regular teaching, to drill and possibly to test students on a material they have learned elsewhere. [20]

D. Data Modelling

Data modelling is the formalization and documentation of existing processes and events that occur during software design and development. Data modelling techniques and tools capture and translate complex system designs into easily understood representations of the data flows and processes, creating a blueprint for construction and/or re-engineering. [21]

Some data modelling tools include UML or the Unified Modelling Language. UML is a standard notation for modeling real-world objects as a first step in developing an object-oriented design methodology. [22] Another is the DFD or the Data Flow Diagram. It is a two-dimensional diagram that explains how data is processed and transferred within a system. The graphical representation identifies each source of data and how it interacts with other data sources to reach a common output. [23]

E. Database Problem Specification

Database problem specifications are a set of information which indicates how a database system must be implemented. These are information relevant in the development of the database design. These specifications indicate which data would be stored in the database. An example of a this specification is the business needs of an organization when developing a database architecture.

In the context of developing a database design through the use of ERD, the specifications would be the information from which ERD constructs will be extracted. In this case, the specifications must contain the information relevant in order to develop an ERD. Given that an ERD organizes data into entities and then relating these entities, the specifications must contain information about possible entities and relationship between these entities. The specification may also contain details on attributes and cardinalities of the entities.

F. Natural Language Processing

Natural language processing (NLP) is the ability of a computer program to understand human language. Development of NLP applications is challenging because traditionally, computers require humans to speak to them in a programming language that is precise, unambiguous and highly structured. However, human language is not always precise. It is often ambiguous and the linguistic structure can depend on many complex variables.

Common NLP activities in software these days include:

- Sentence segmentation, part-of-speech tagging and parsing.
- Deep analytics.
- Named entity extraction.
- Co-reference resolution. [24]

G. Stanford Parser

Stanford Parser is a Java implementation of probabilistic natural language parsers. The original version of this parser was mainly written by Dan Klein, with support code and linguistic grammar development by Christopher Manning. Additional work (internationalization and language-specific modeling, flexible input/output, grammar compaction, lattice parsing, k-best parsing, typed dependencies output, user support, etc.) has been done by Roger Levy, Christopher Manning, Teg Grenager, Galen Andrew, Marie-Catherine de Marneffe, Bill MacCartney, Anna Rafferty, Spence Green, Huihsin Tseng, Pi-Chuan Chang, Wolfgang Maier, and Jenny Finkel.

As well as providing an English parser, the parser can be and has been adapted to work with other languages. A Chinese parser based on the Chinese Treebank, a German parser based on the Negra corpus and Arabic parsers based on the Penn

Arabic Treebank are also included. The parser has also been used for other languages, such as Italian, Bulgarian, and Portuguese. [25]

Given the input "The strongest rain ever recorded in India shut down the financial hub of Mumbai, snapped communication lines, closed airports and forced thousands of people to sleep in their offices or walk home during the night, officials said today.", the parser can perform the following functionalities. Some of the functionalities not used in the study are not stated.

1. Part-of-speech Tagging

   Part-of-speech tagging is the process of assigning words its corresponding part of speech in a sentence. This function can be performed by a POS tagger, a software or program which tags or labels every word in a sentence to its corresponding part of speech.

   A part of speech is a term in traditional grammar for the eight categories into which words are classified according to their functions in sentences such as noun, verb, adjective, pronoun, etc. In contemporary linguistics, the term part of speech has generally been discarded in favor of the label word class or syntactic category. [26]

   The tags or labels following each word indicates its classification in the parts of speech. Figure 7 shows the labels used by the Stanford parser for tagging parts of speech.

CC Coordinating conjunction	CD Cardinal number
DT Determiner	EX Existential there
FW Foreign word	IN Preposition or subordinating conjunction
JJ Adjective	JJR Adjective, comparative
JJS Adjective, superlative	LS List item marker
MD Modal	NN Noun, singular or mass
NNS Noun, plural	NNP Proper noun, singular
NNPS Proper noun, plural	PDT Predeterminer
POS Possessive ending	PRP Personal pronoun
PRP\$ Possessive pronoun	RB Adverb
RBR Adverb, comparative	RBS Adverb, superlative
RP Particle	SYM Symbol
TO to	UH Interjection
VB Verb, base form	VBD Verb, past tense
VBG Verb, gerund or present participle	VBN Verb, past participle
VBP Verb, non-3rd person singular present	VBZ Verb, 3rd person singular present
WDT Wh-determiner	WP Wh-pronoun
WP\$ Possessive wh-pronoun	WRB Wh-adverb

Figure 7: POS Labels used by Stanford Parser

More details regarding POS tagging used by the Stanford Parser can be found in [27].

The output of the given input as generated by the Stanford Parser is as follows

```
The/DT strongest/JJS rain/NN ever/RB recorded/VBN in/IN
India/NNP shut/VBD down/RP the/DT financial/JJ  hub/NN
of/IN Mumbai/NNP ,/, snapped/VBD communication/NN lines/NNS
,/, closed/VBD airports/NNS and/CC forced/VBD thousands/NNS
of/IN people/NNS to/TO sleep/VB in/IN their/PRP\$ offices/NNS
or/CC walk/VB home/NN during/IN the/DT night/NN ,/,
officials/NNS said/VBD today/NN ./.
```

2. Generate Typed Dependency Representation

The Stanford typed dependencies representation was designed to provide a simple description of the grammatical relationships in a sentence that can easily be understood and effectively used by people without linguistic expertise who want to extract textual relations. In particular, rather than the phrase structure representations that have long dominated in the computational linguistic community, it represents all sentence relationships uniformly as typed dependency relations. [28].

Using the same input as used in the part-of-speech tagging, the Stanford would generate the following the following typed dependencies.

det(rain-3, The-1)
nsubj(snapped-16, rain-3)
advmod(recorded-5, ever-4)
ccomp(said-40, shut-8)
amod(hub-12, financial-11)
conj_and(shut-8, snapped-16)
dobj(snapped-16, lines-18)
dobj(closed-20, airports-21)
dobj(forced-23, thousands-24)
xcomp(forced-23, sleep-28)
xcomp(forced-23, walk-33)
det(night-37, the-36)
root(ROOT-0, said-40)
amod(rain-3, strongest-2)
nsubj(closed-20, rain-3)
partmod(rain-3, recorded-5)
prt(shut-8, down-9)
dobj(shut-8, hub-12)
ccomp(said-40, snapped-16)
conj_and(shut-8, closed-20)
conj_and(shut-8, forced-23)
prep_of(thousands-24, people-26)
poss(offices-31, their-30)
conj_or(sleep-28, walk-33)
prep_during(walk-33, night-37)
tmod(said-40, today-41
nsubj(shut-8, rain-3)
nsubj(forced-23, rain-3)
prep_in(recorded-5, India-7)
det(hub-12, the-10)
prep_of(hub-12, Mumbai-14)
nn(lines-18, communication-17)
ccomp(said-40, closed-20)
ccomp(said-40, forced-23)
aux(sleep-28, to-27)
prep_in(sleep-28, offices-31)
dobj(walk-33, home-34)
nsubj(said-40, officials-39)

Figure 8: Typed Dependencies Generated From Example

This simple and uniform representation is quite accessible to non-linguists thinking about tasks involving information extraction from text and is effective in relation extraction applications.

H. Typed Dependencies of the Stanford Parser

de Marneffe et al. proposed Typed Dependency grammatical relation in [29] based on the constituent tree output structure of the Stanford Parser. The typed dependencies extracted from the constituent tree structured parsing results are convenient for practical natural language applications such as NLP-based conceptual data modeling. [14]

The next figure shows the typed dependency hierarchy of the Stanford Parser. There are approximately 50 grammatical relations in the hierarchy. The most general relation is dependent which can be further divided into aux (auxiliary), arg (argument), mod (modier), etc. Parsing results are tagged with these tags. For example, the Typed Dependency parse result for sentence "An instructor teaches a course", is the following,

det(instructor , An) $nsubj$(teaches , instructor)

det(course , a) $dobj$(teaches , course)

The typed dependencies are all binary relations: a grammatical relation that holds between a governor (also known as a regent or a head) and a dependent. The governor is the first element in the relation and the dependent is the second word. The relation says that the dependent depends on the governor. These grammatical relations may be utilized to extract entity relations for automated data modeling. [14]

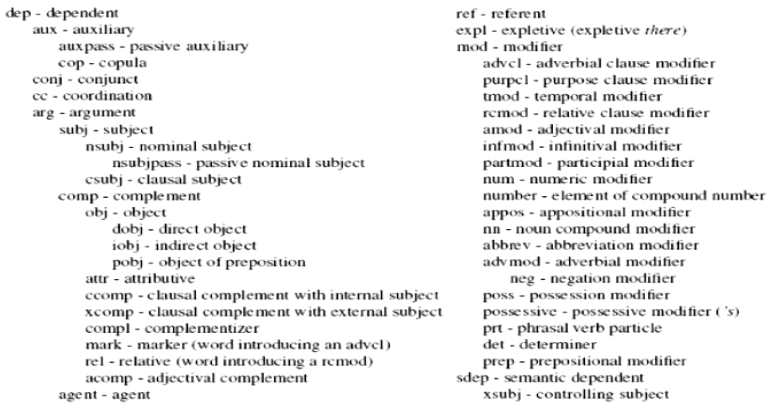

Figure 9: Typed Dependency Structure of Stanford Parser

I. Heuristics Rules

Heuristic rules have been utilized in previous research with the aim of capturing the relationships between different elements in natural languages and those in ERDs. [14] The rules proposed here are derived from [14] which are based on the idea that 'Nouns are corresponding to entities and attributes while verbs are likely relation candidates in ERDs'. These rules use the typed dependency parsing results of the Stanford parser to extract data such as entities and relationships.

Rules are used to extract entities and their relationships from a given database specification. Every rule has a condition and generates a tuple the rule is fired. The rules are of the following format,

$$Conditions \Rightarrow \langle relationship, entity1, entity2 \rangle \tag{1}$$

A tuple is always an ordered triple indicating the relationship and the two entities of the relation. An ordered triple as represented in the previous rule can be translated into graphical format as follows:

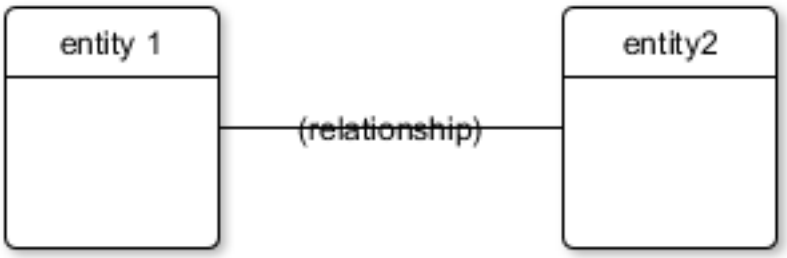

Figure 10: Graphical representation of rule format

The left hand side of the rules are the conditional typed dependencies required to fire the rule. $w1, w2, ..., wn$ are numbered English words from the typed dependency parsing results of the Stanford Parser. Word order sequence is required [14]. The

rules are described as follows.

$$nsubj(w2, w1) + dobj(w2, w3) \Rightarrow \langle w2, w1, w3 \rangle \tag{2}$$

nsubj and *dobj* type dependencies are both required to execute this rule. The binary relation *nsubj* holds the verb and the subject of the sentence, and the binary relation *dobj* holds the verb and the object of the sentence. The first attribute in the *nsubj* and *dobj* relation is a verb, which translates to a relationship, as stated in [12], between the two entities. For example:

An instructor has a name.

det(instructor , An) *nsubj*(has , instructor)

det(name , a) *dobj*(has , name)

By applying the rule given these conditions, the tuple ⟨teaches,instructor,course⟩ can be extracted from the example sentence.

$$nsubj(w2, w1) + prep(w2, w3) + pobj(w3, w4) \Rightarrow \langle w2\_w3, w1, w4 \rangle \tag{3}$$

nsubj, *prep* and *pobj* type dependencies are all required to execute this rule. *nsubj* would indicate a possible entity. *pobj*, the relation indicating the object of preposition would indicate the other entity. The *prep* relation is needed since it would indicate the relationship between the entities. For example:

An instructor must work on at least one department.

det(instructor , an) *nsubj*(work , instructor)

det(department , a) *prep*(work , on)

pobj(on , department)

By applying the rule given these conditions, the tuple ⟨work_on,instructor,department⟩

can be extracted from the example sentence.

$$nsubj(w2, w1) + aux(w2, w3) + dobj(w2, w4) \Rightarrow \langle w2, w1, w4 \rangle \quad (4)$$

nsubj, *aux* and *dobj* type dependencies are all required to execute this rule. For example:

An instructor must teach at least one course.

det(instructor , An)	*nsubj*(teach , instructor)	*mwe*(at , least)
aux(teach , must)	*quantmod*(one , at)	*pobj*(at , least)
num(course , one)	*dobj*(teach , course)	

By applying the rule given these conditions, the tuple ⟨teach,instructor,department⟩ can be extracted from this example.

Cardinalities between entities of a relationship can also be extracted from the Stanford Parser by using rules.

$$nsubj(w2, w1) + dobj(w2, w3) + amod(w3, w4) \Rightarrow \langle w4, w1, w3 \rangle \quad (5)$$

The relations *nsubj*, *amod*, and *dobj* are enough to fire the rule. The same as discussed in the previous rules, the relations *nsubj* and *dobj* indicates the entities for the relation. *amod*, an adjectival modifier, indicates the occurence of the entity in *dobj* for a single occurence of the entity in *nsubj*. For example:

An instructor can teach zero_or_many courses.

det(instructor , An)	*nsubj*(teach , instructor)	*aux*(teach , can)
amod(courses , zero_or_many)	*dobj*(teach , courses)	

By applying the rule given these conditions, the tuple ⟨zero_or_many,instructor,courses⟩ can be extracted from this example.

IV. Design and Implementation

A. System Modules Overview

The system has three main modules: input and validation module, parsing and extraction module, and the rendering module.

Given an input provided by the student, the system proceeds to validation, then to parsing and extraction, and finally to final ERD rendering.

The whole system is implemented in Java as a standalone application. There are several open-source resources used in the development of the system. The core resources used include the Stanford parser and the Graphstream graph package. The Stanford parser is the main module used for parsing and extracting data from the student input. The Graphstream graph package is used for the final ERD rendering and representation.

The student inputs a set of database requirements as a paragraph or as separated sentences. Before proceeding to parsing and extraction, the system performs word checking on the words of the sentence. User must correct spelling errors if there are any or add new words to a dictionary. If there are no spelling errors, system proceeds to checking the format of the input. The system passes the input to the Stanford parser to perform POS tagging. The tagged sentence will undergo the syntax check. Again, the student must correct any sentence that does not follow any of the six sentence formats discussed in the scope and limitations. More details about checking and validating the input are discussed on the next section. Conflicts between requirements are then resolved after passing the spelling and format validation. Details about conflict checking are discussed in the succeeding sections.

If the sentences passed the validation, the system proceeds to classifying each sentence into one of the six sentence formats. After classification, typed dependency parsing of the sentences is performed. Once the classification and typed dependencies

of each sentence are obtained, essential components and relationships are identified using the rules discussed in the previous chapter. This component identification module returns ordered triples containing components and relationships of the ERD to be generated.

After obtaining the ordered triples, the system renders the components into the canvas using the Graphstream package and outputs the final ERD to the student.

The next figure shows the flowchart of the whole system process.

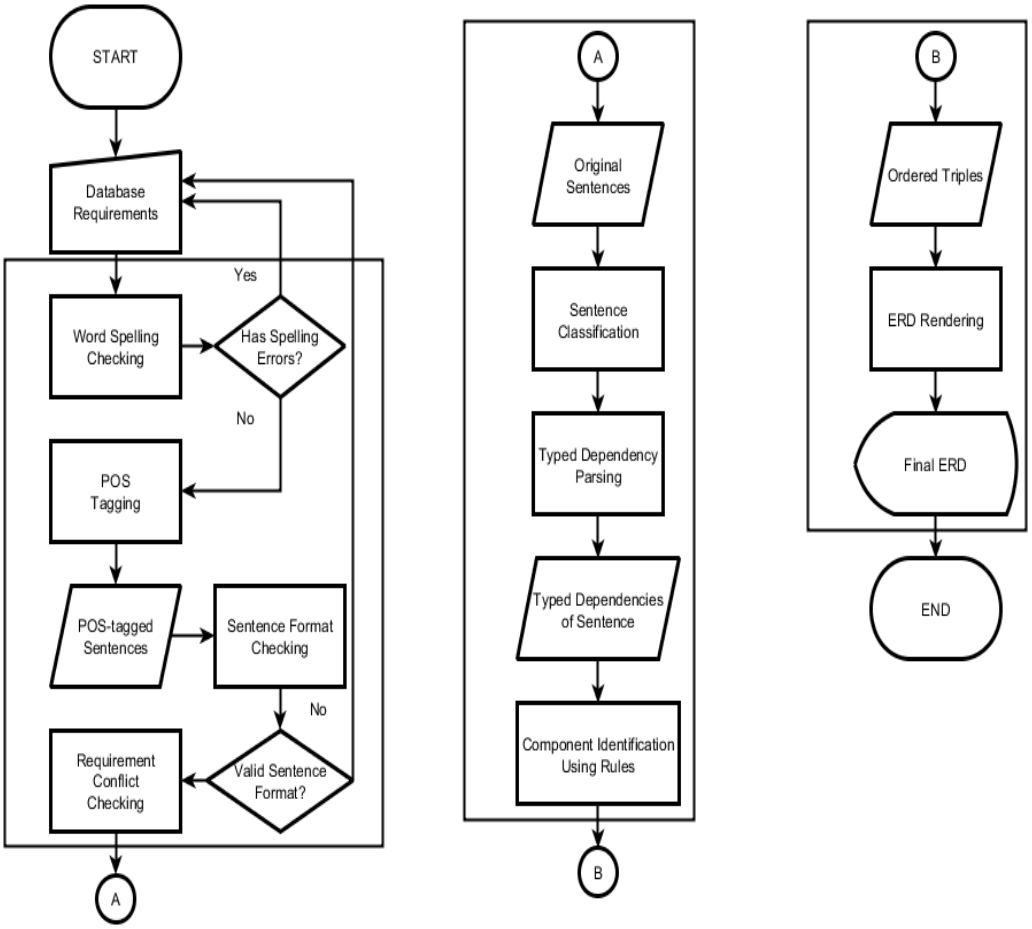

Figure 11: System Flowchart

B. System Input Validation and Sentence Classification

Before the system process and extract data from the input, the input must first be checked and validated by the system. The input of the system must satisfy the following criteria before proceeding to component extraction: it contains valid words, and it follows one of the sentence formats discussed in scope and limitations. Unless these criteria are satisfied, the system would keep on sending error messages to the student, and the student must perform necessary changes.

For valid word checking, the system has an included list of words (500,889 words from [30]) for checking words in the sentences. Words are valid if contained in the list, otherwise, invalid. In the case where the student is prompted for an invalid word, he is given the option to add the word to the system's dictionary, and it will be a valid word the next time the student uses the word again. However, there is no implemented functionality for deleting any word in the dictionary.

For sentence format validation, regular expressions are used. If the words are all valid, a sentence is accepted if it matches any of the six regular expressions. These regular expressions are applied to POS tagged sentences. Figure 12 shows the regular expressions used.

The first column indicates which sentence type format is validated by the regular expression in the second column of the corresponding row. The same regular expressions are used for sentence classification. The sentence's format is identified by the regular expression that it matches. However, in sentence classification, the system does not rely entirely on these regular expressions and has more classification criteria like the existence of certain keywords, like 'can be', 'at the same time', etc., in each of the sentences.

Type 1	(.*)/NNS? (HAS|HAVE)/VB(P|Z) (A|AN)/DT (.*)/NNS?((\\,/\\,| AND/CC) (.*)/NNS?)* \\./\\.
	(.*)/NNS? (HAS|HAVE)/VB(P|Z) A/DT (UNIQUE|DISTINCT)/JJ (.*)/NNS?((\\,/\\,| AND/CC) (.*)/NNS?)* \\./\\.
Type 2	(.*)/NNS? (MUST|MAY|MIGHT|CAN|COULD|SHOULD)/MD (.*)/VB(P|Z)?((.*)/(IN|TO))? (AT_MOST_ONE|ZERO_OR_ONE|ONLY_ONE|AT_LEAST_ONE|ONE_OR_MANY|ZERO_OR_MANY)/(JJ|VB) (.*)/NNS?((\\,/\\,| OR/CC) (.*)/NNS?)* \\./\\.
Type 3	(.*)/NNS? CAN/MD BE/VB EITHER/CC (A|AN)/DT (.*)/NNS?((\\,/\\,| OR/CC) (.*)/NNS?){1,}(ONLY/RB)? \\./\\.
	(.*)/NNS? CAN/MD BE/VB (A|AN)/DT (.*)/NNS?((\\,/\\,| OR/CC| AND/CC) (.*)/NNS?)*(ONLY/RB)? \\./\\.

Figure 12: Regular Expressions for Sentence Validation

C. Validation for Conflicting Requirements

Error checking for conflicting requirements in the specification is needed in order to produce the correct ERD. Without conflict checking, the resulting ERD is ambiguous and contains incorrect information.

Conflicts arise when there are requirements that contradict each other. An example of this would be having two entities in the requirement with different cardinality constraints indicated. For a more specific example of this constraint,

An employee must belong to only one department.

An employee must belong to at least one department.

The conflicts managed by the system can be classified into three based on the sentence type formats.

1. Type 1 Conflict

   Type 1 conflict arises when an attribute from one requirement is used as an entity in another requirement, as well as using an entity as an attribute of another entity.

Example:

> An employee has a unique id. An id must have at least one student.
>
> A system has a student.

From the example, the conflict arises when *id* is used as an entity in the second requirement. Also, the *student* is used as an attribute in the last requirement when it is an entity in the second.

Algorithm for checking conflict:

```
boolean hasType1Conflict(set of attributes, set of entities){
  Let S1 <− set of attributes
  Let S2 <− set of entities
  foreach(a in S1){
    if(S2 contains a)
      return TRUE
  }
  foreach(e in S2){
    if(S1 contains e)
      return TRUE
  }
  return FALSE
}
```

2. Type 2 Conflict

Type 2 conflict arises when the same relationship between two entities has multiple cardinality constraints definition in the requirements.

Example:

> An employee must belong to only one department. An employee must belong to at least one department.

From the example, the conflict arises when *employee* and *department* has the *only one* as the cardinality in the first requirement and then redefined to have *at least one* in the second requirement.

Algorithm for checking conflict:

```
boolean hasType2Conflict(set of relationships, requirement){
  Let S <− set of relationships
  foreach(r in S){
    if((r.cardinality != requirement.cardinality)
      AND (r.entity1 == requirement.subject)){
      if(requirement.objects contains r.entity2){
        return TRUE
      }
        }
  }
  return FALSE
}
```

3. Type 3 Conflict

Type 3 conflict arises when one or more subtype requirements contradict another. This may arise from requirements indicating total or partial specialization.

Example:

> An empoyee can be a manager or a supervisor only. An employee can be a leader.

From the example, the conflict arises in the requirement that indicates that an *employee* can be a *leader*. Since the first requirement indicates that the *employee* can be a *manager* or *supervisor* **only**, then an *employee* cannot be a *leader*.

Algorithm for checking conflict:

```
boolean hasType2Conflict(requirement1, requirement2){
  if(requirement1 has "only" OR requirement2 has "only"){
    total <− TRUE;
  }
  foreach(o in requirement1.objects){
    if(requirement2 not containts o AND total){
```

```
        return TRUE
      }
    }
    return FALSE
  }
```

D. Final ERD Representation

The Graphtstream package is a Java graph library. This library has powerful and customizable rendering options, and is easily extensible to render entity relationship diagrams. The drawback of this library, however, is that it has preconfigured layout algorithms which may not be the ideal layout for rendering and representing ERD.

The system has an ERD utility module that creates the ERD from the generated ordered triples. Since the system uses the crow's feet notation, each entity is represented as a node in the graph, and the relationships are simply edges connecting the nodes representing the entities. The same method can be applied in case the other notations are needed. As in the case of Chen notation used in [14], the everything is rendered as nodes (entities, attributes, relationships) with custom shapes to match the component and simply connecting related components.

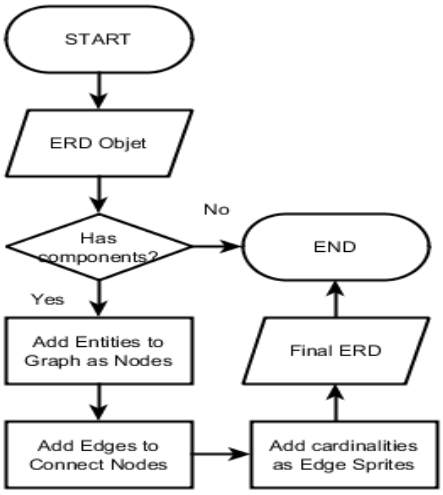

Figure 13: ERD Rendering and Representation

E. Component Extraction from Sentences using Heuristic Rules

As discussed in the previous chapter, there are several heuristic rules which can be used to extract components from sentences. These rules can be applied in order to extract components from sentences of the same format as presented in section E of chapter I.

For extracting details from sentence with type one format, the rules,

$$nsubj(w2, w1) + dobj(w2, w3) \Rightarrow \langle w2, w1, w3 \rangle$$
$$nsubj(w2, w1) + prep(w2, w3) + pobj(w3, w4) \Rightarrow \langle w2\_w3, w1, w4 \rangle$$

can be used. Typed dependency parsing performed by Stanford parser on sentences matching sentence format (a) are as follows:

$$nsubj(verb,\ subject) + dobj(verb,\ object) \Rightarrow \langle verb, subject, object \rangle$$
$$nsubj(verb, subject) + prep(verb, prep) + pobj(prep, object)$$
$$\Rightarrow \langle verb\_prep, subject, object \rangle$$

These rules may also be used to extract components from sentences matching sentence format type two. However, since only sentence format type is the one which indicates cardinality constraints, the following rule is also used along with the previous rules. The previos rules extract relationships, whereas this rule extracts cardinalities.

$$nsubj(w2, w1) + dobj(w2, w3) + amod(w3, w4) \Rightarrow \langle w4, w1, w3 \rangle$$

After processing sentences of sentence format type two, the rule would fire generate tuple as follows:

$$nsubj(verb, subject) + dobj(verb, object) +$$
$$amod(object, cardinality) \Rightarrow \langle \text{cardinality,subject,object} \rangle$$

The same rules used in type one sentences can be used in type three sentence. However, since the sentence formats exhibit multiple objects, the typed dependency parsing results to multiple *dobj*, and so multiple tuples are returned. The system then gets all combinations needed and generates the tuple from this combinations.

To follow the convention where the first element of the tuple is a relationship, the generated relationship in each tuples is replaced to give a more meaningful tuple.For type one sentences, the first elements is set to *has* to indicate relationship as being an attribute, and *unique* to indicate the attribute being unique. For type three sentences, *can_be_disjoint* and *can_be_overlap* are used respectively to indicate disjointness constraint. *_only* is added at the end of these relationships if the type three sentence includes **only** to indicate completeness constraint.

V. Architecture

A. System Architecture

NERD is a standalone application implemented in Java. There are no specific frameworks used like Spring, Hibernate, etc. However, there are several libraries used in order to simplify the development of the application. These libraries are the Stanford Parser, Graphstream and IcePDF.

The module for generating ERD from text relies on the Stanford Parser and the rendering of output relies on Graphstream. IcePDF library is used to display the lessons in PDF format.

No other special libraries and configurations are used in the development of the system.

B. Technical Architecture

The following are the minimun system requirements in order to run the application:

- Java Runtime Environment 1.5 or higher
- 1GB or higher free disk space for application storage
- 512MB of RAM or higher (1GB RAM recommended)
- Apache ANT 1.8 or higher (required by IcePDF)

VI. Results

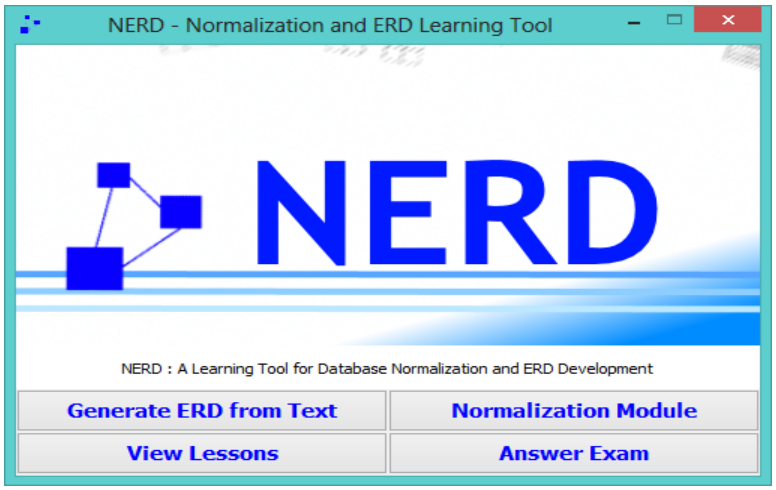

Figure 14: Home screen, NERD

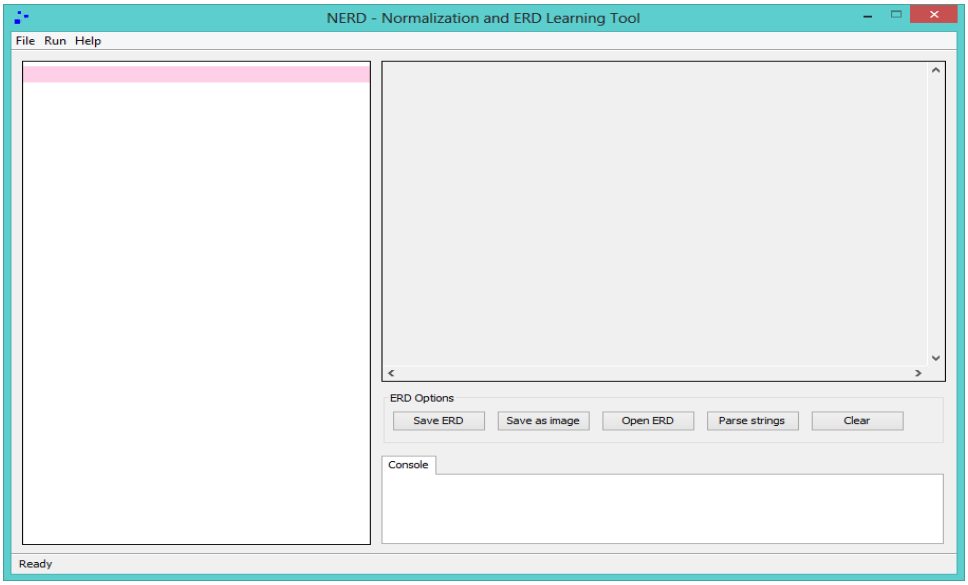

Figure 15: Main editor, NERD

Clicking Generate ERD from Text button from the home screen opens the window above. The window has three main areas: the editor (left side), the rendering canvas (right side), and the console (below canvas). The editor is for displaying and editing files, rendering canvas is for displaying output and console is for displaying error messages.

There are three menus: File, Run, and Help.

File menu includes options for saving the current file, opening a saved file, and creating new files.

Run menu includes options to run the current file, and step by step option. Clicking the run menu item starts the generation of ERD from text and outputs the result in the rendering canvas. Step by step option allows the student to view the step by step creation process of the ERD.

Help menu includes options for viewing user manual and details about the system. Clicking User Manual menu opens the system's user manual. About menu item displays details about the system.

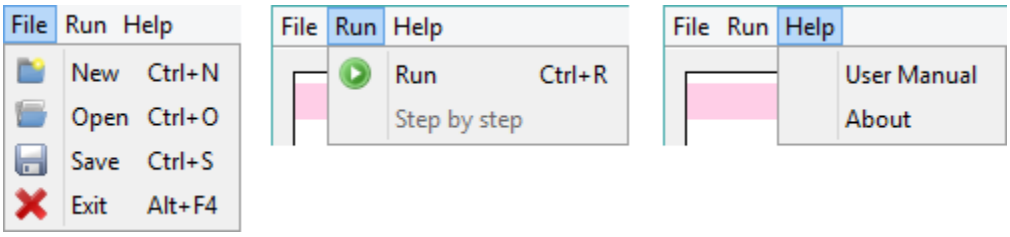

Figure 16: Generating ERD from text module menus, NERD

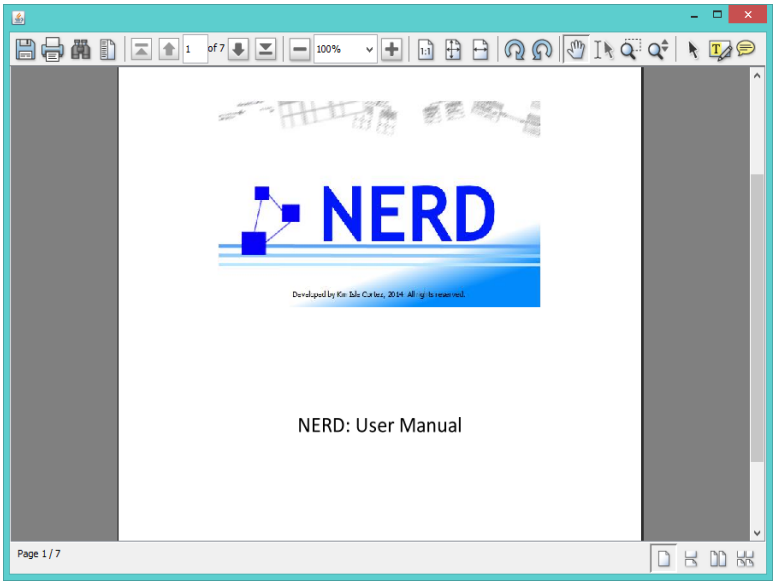

Figure 17: The user manual, NERD

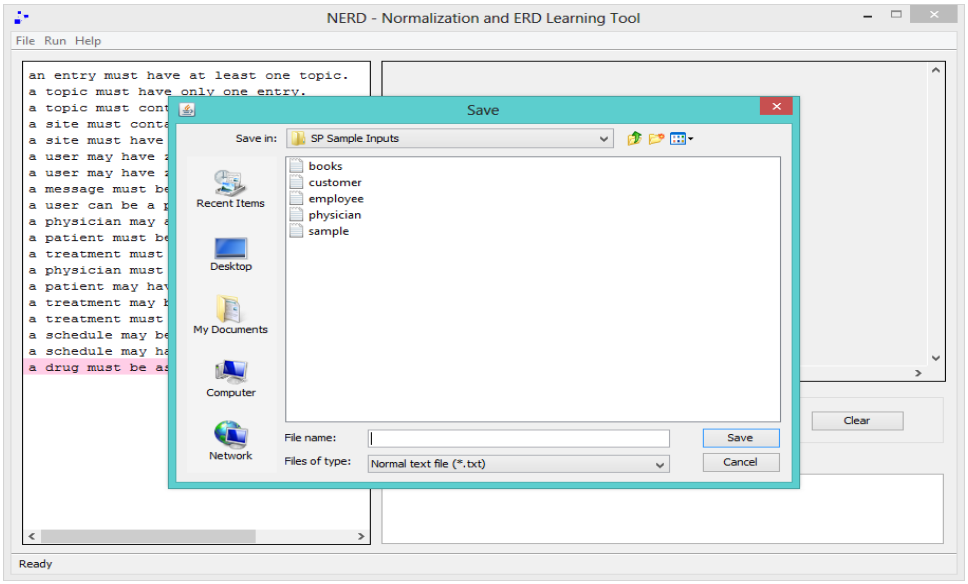

Figure 18: Saving a text file, NERD

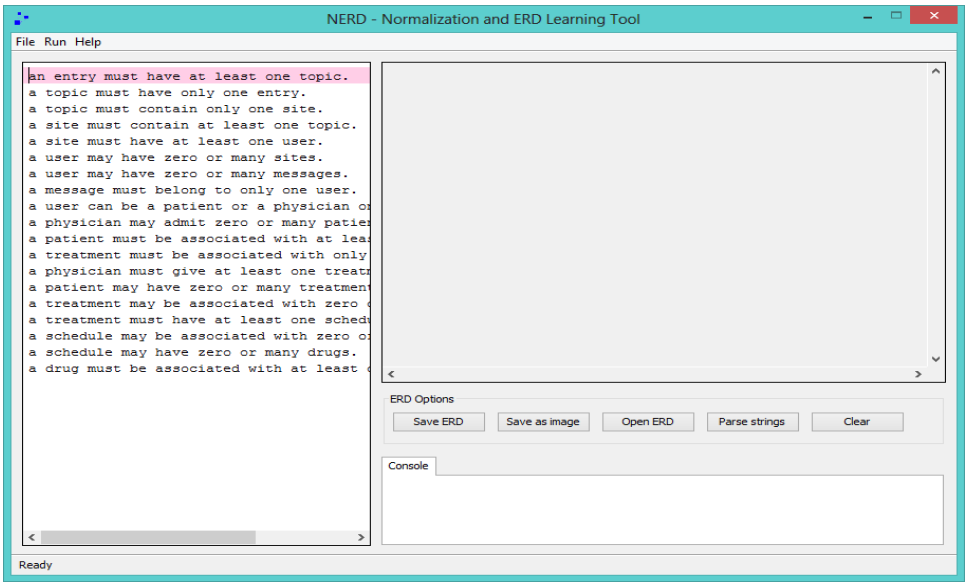

Figure 19: Loaded text file from file system, NERD

Files containing the input can be saved (Figure 18), opened and manipulated (Figure 19) in the main editor. New files are saved in .txt format. Also, only .txt files are opened in the editor.

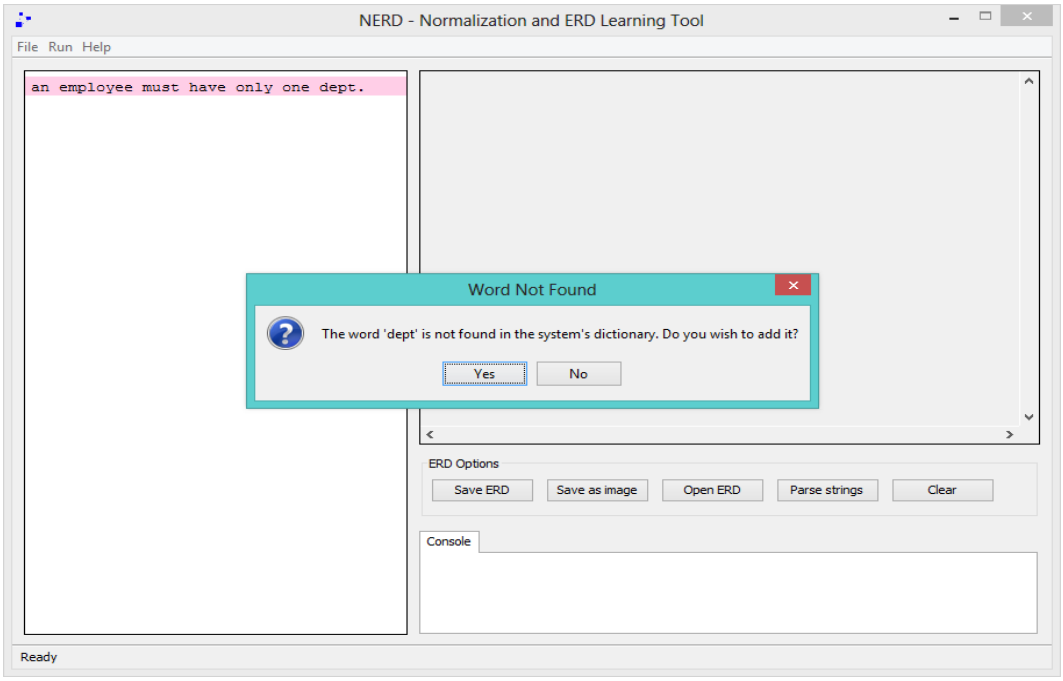

Figure 20: Add word to dictionary, NERD

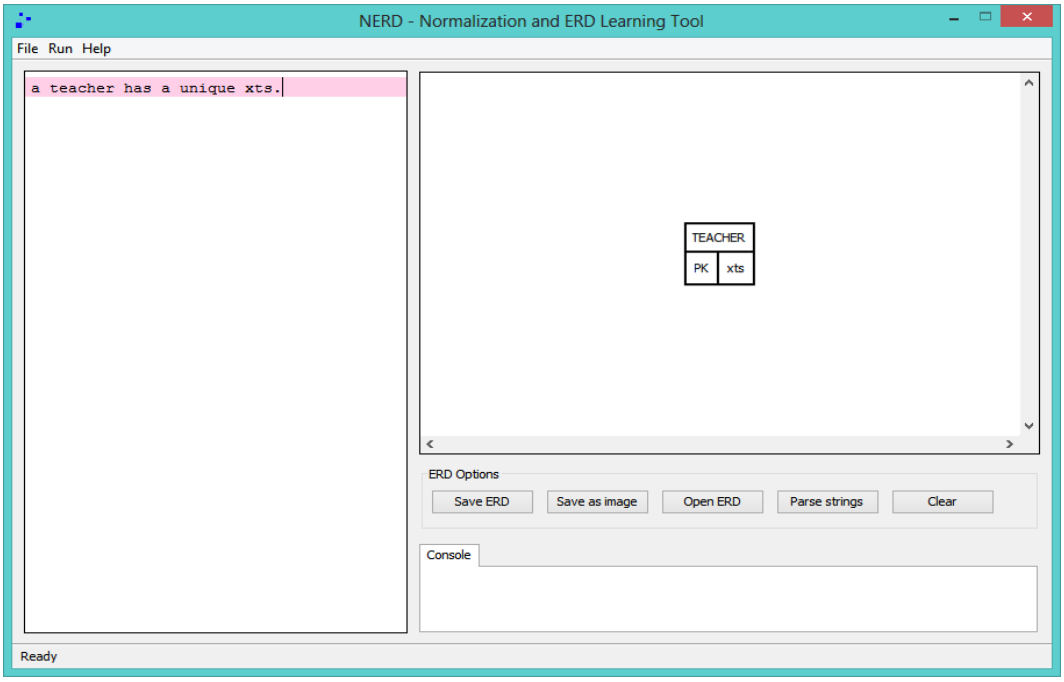

Figure 21: Word added to dictionary, NERD

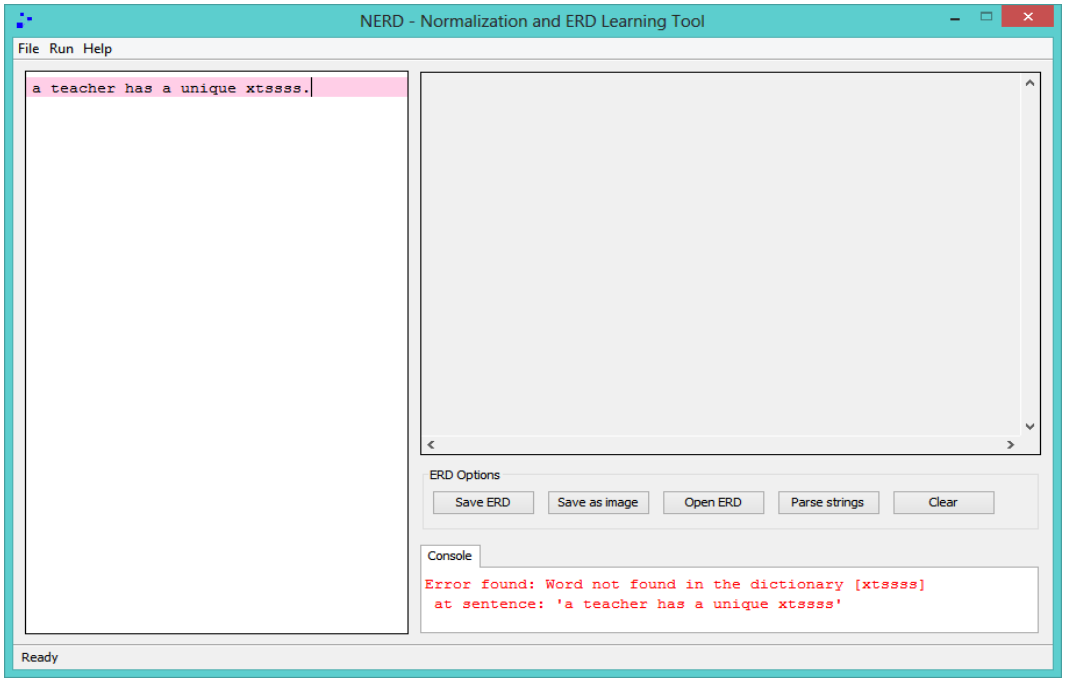

Figure 22: Word not added to dictionary, NERD

The module performs spelling checking before it continues to component extraction as a part of its validation. If a word in the input is not in the dictionary, it prompts the student for an option to add the word to the dictionary (Figure 20). If the student chooses to add the word, the word is added in the dictionary and proceeds without error messages, and then proceeds to the processing procedure (Figure 21). If the student chooses not to add the word, an error message indicating that a word is not in the dictionary is displayed in the console (Figure 22).

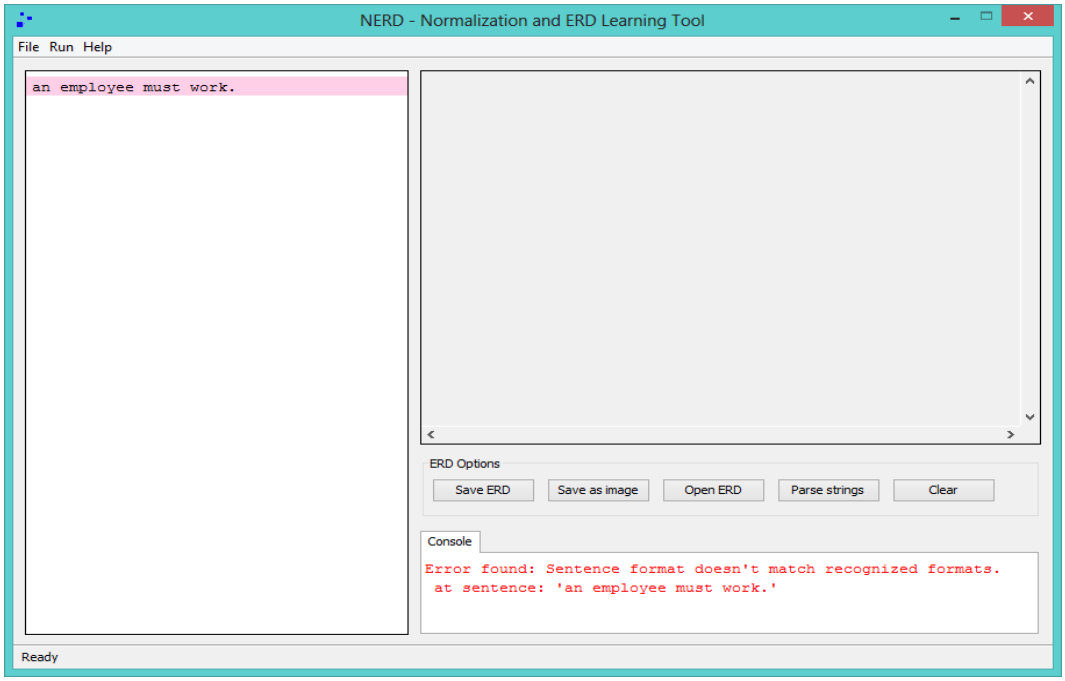

Figure 23: Sentence format error, NERD

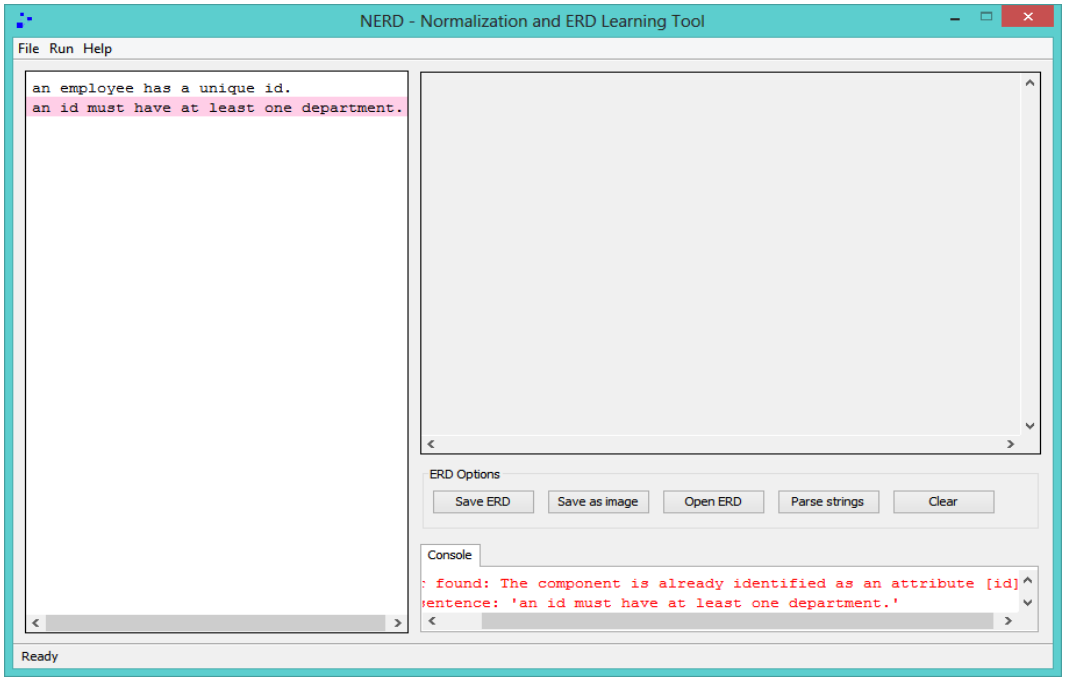

Figure 24: Type 1 conflict found, NERD

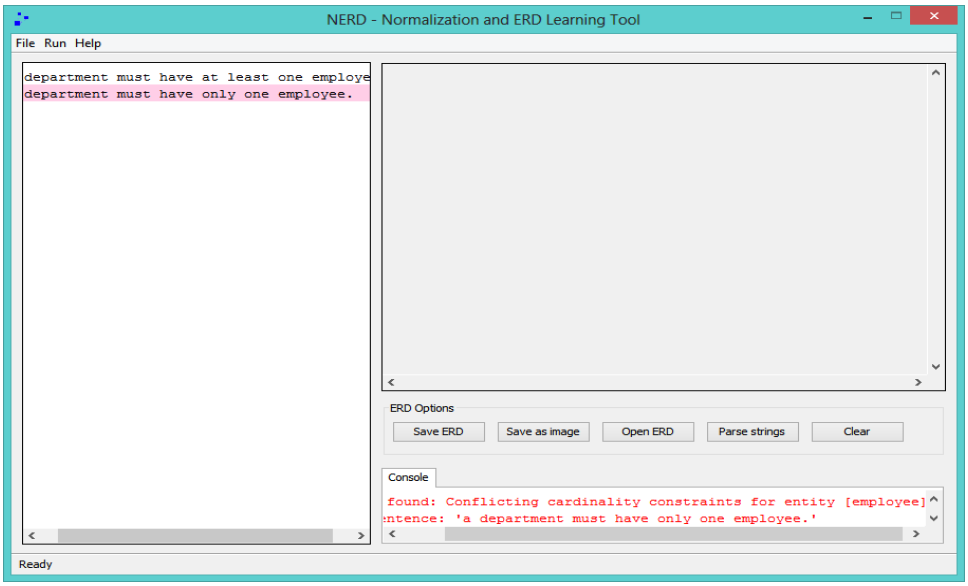

Figure 25: Type 2 conflict found, NERD

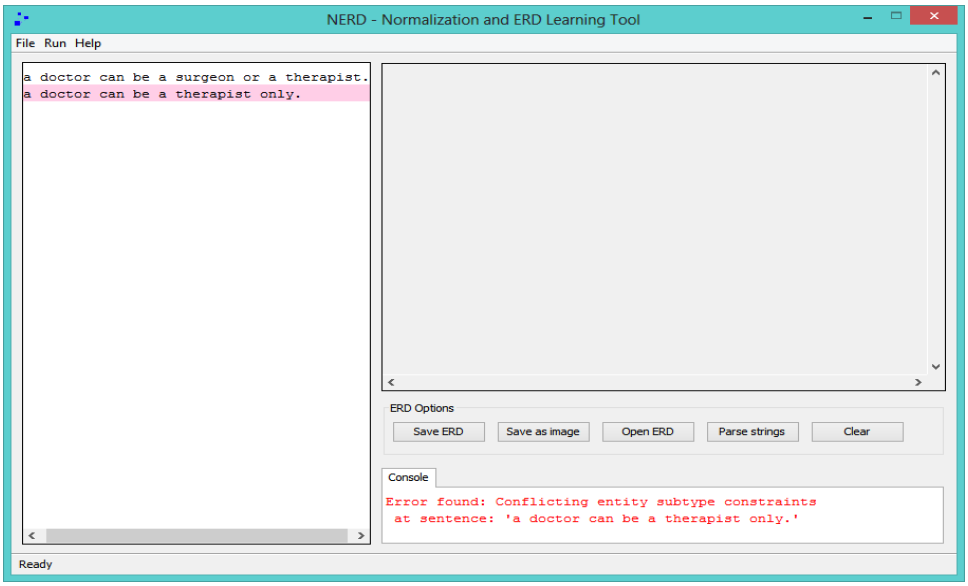

Figure 26: Type 3 conflict found, NERD

Figures 23 to 26 show the other validation capabilities of the system: requirement conflict checking and sentence format checking.

Figure 23 shows the sentence format validation. The sentence *an employee belongs to a department* does not belong to any of the sentence type format discussed in section E chapter I, hence an error.

Figures 24, 25 and 26 show the validation for conflicting requirements.

Figure 24 shows a type one conflict error. The requirements *an employee has an id* and *an id must have at least one department* has type one error since *id* is used as entity even though it is already used as an attribute.

Figure 25 shows type two conflict error. The requirements *a department must have at least one employee* and *a department must have only one employee* returns conflict error because of multiple cardinality defined between *employee* and *department*.

Figure 26 shows a type three conflict error. The requirements *a doctor can be a surgeon or a therapist* and *a doctor can be a therapist only* raised the conflict since a *doctor* cannot be a *surgeon* if he can be a *therapist* only.

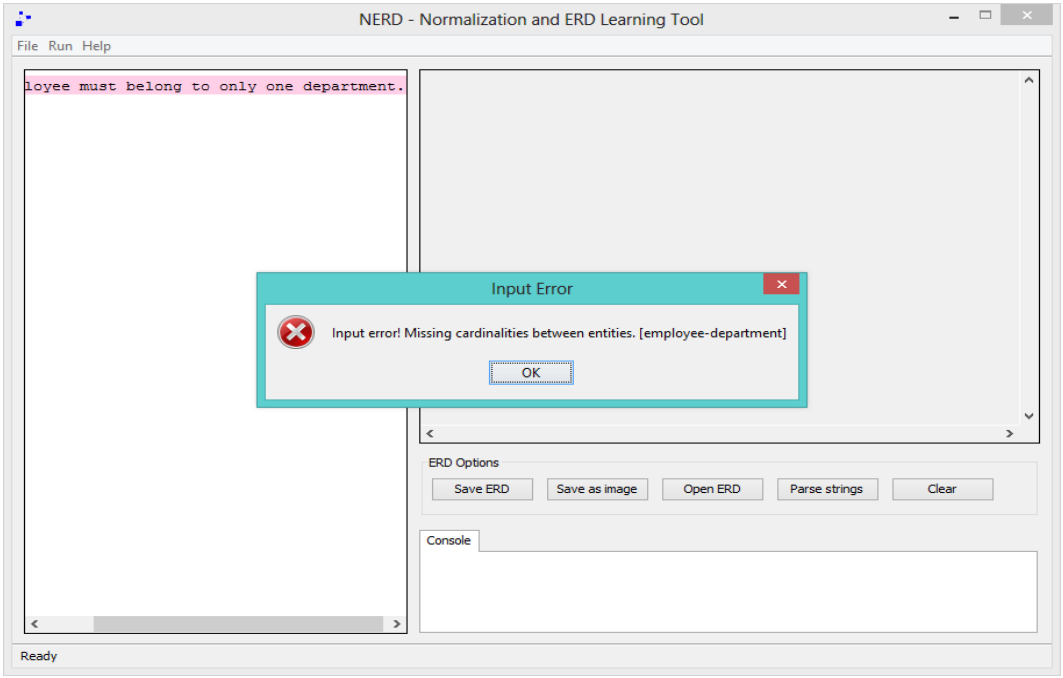

Figure 27: Missing cardinalities in the requirement, NERD

Figure 27 shows a missing cardinality error after processing *a department must have only employee* since the requirements do not contain the cardinality from *employee* to *department*. Recall from section E of chapter I that cardinality constraints must be indicated in two way.

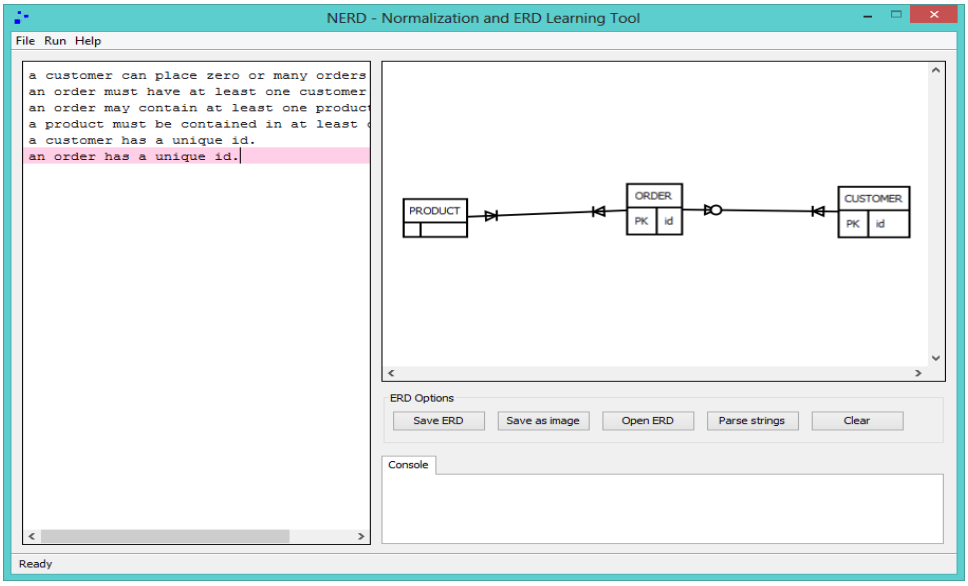

Figure 28: Final ERD output, NERD

Figure 28 shows the final output in the rendering canvas. Below the rendering canvas are five buttons which are options for the ERD in the rendering canvas: Save ERD, Save as image, Open ERD, Parse strings, and Clear.

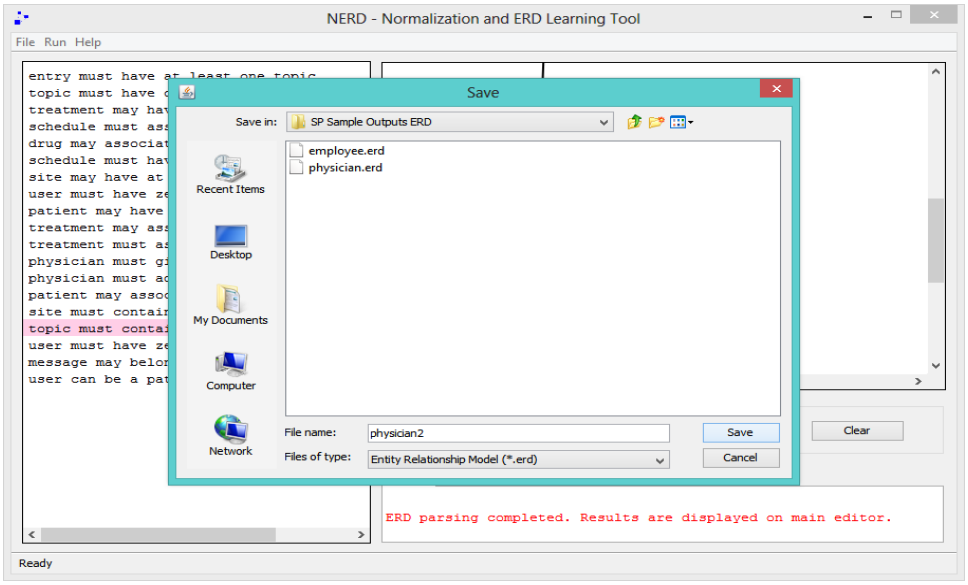

Figure 29: Save ERD functionality, NERD

The Save ERD button saves the currently rendered ERD in a serialized file with .erd format which is readable by the system.

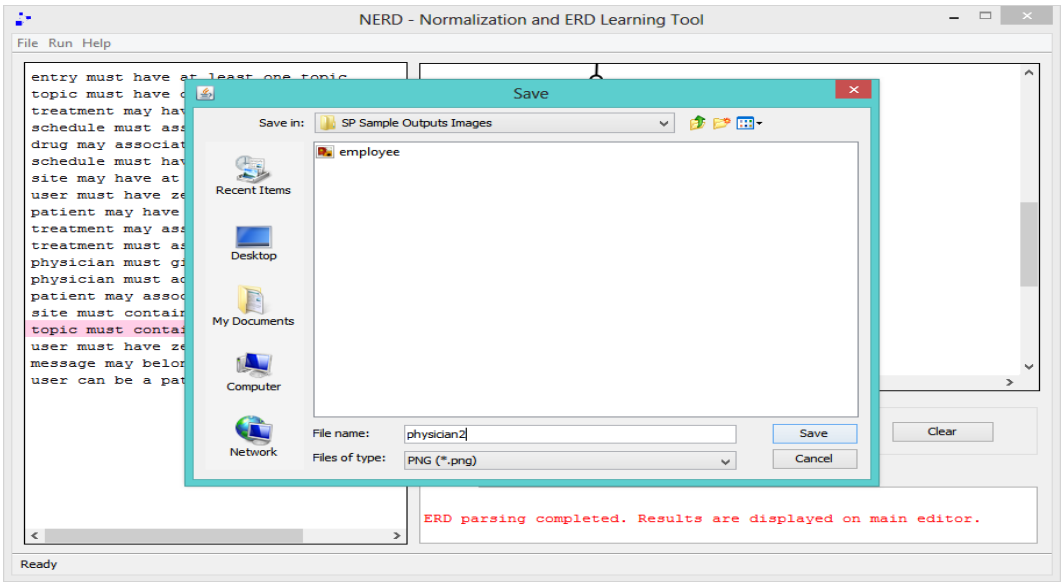

Figure 30: Save as image functionality, NERD

The Save as image button saves the currently rendered ERD as an image file in PNG format. The whole ERD is saved as image, and not just the visible part.

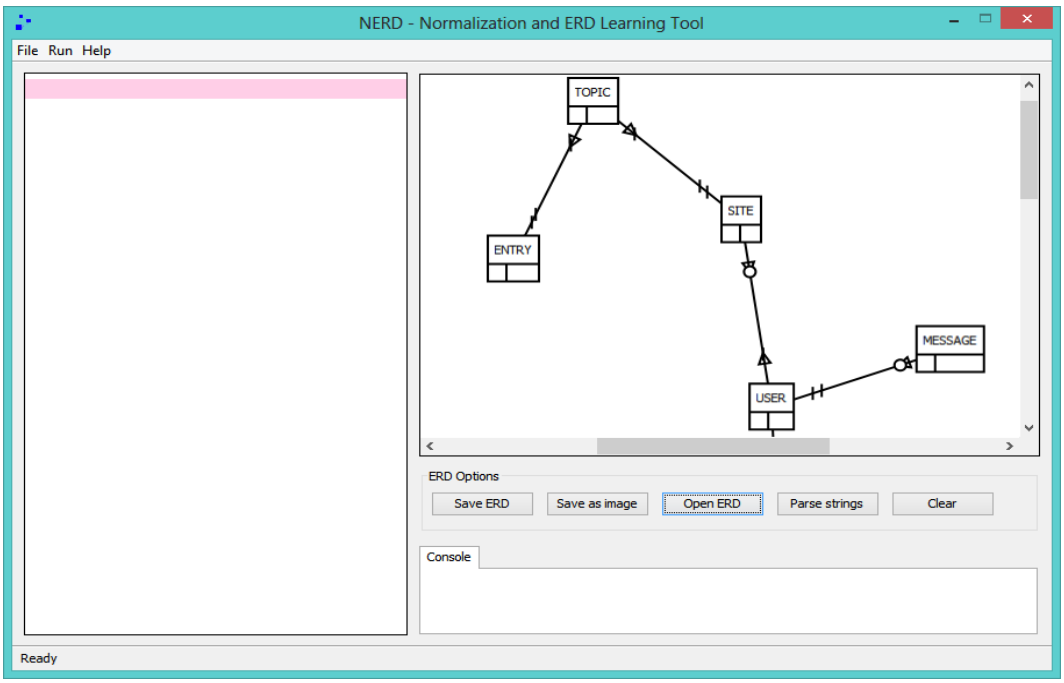

Figure 31: Open ERD functionality, NERD

The Open ERD button opens an ERD file (.erd file) saved in the file system, and then displays it to the canvas.

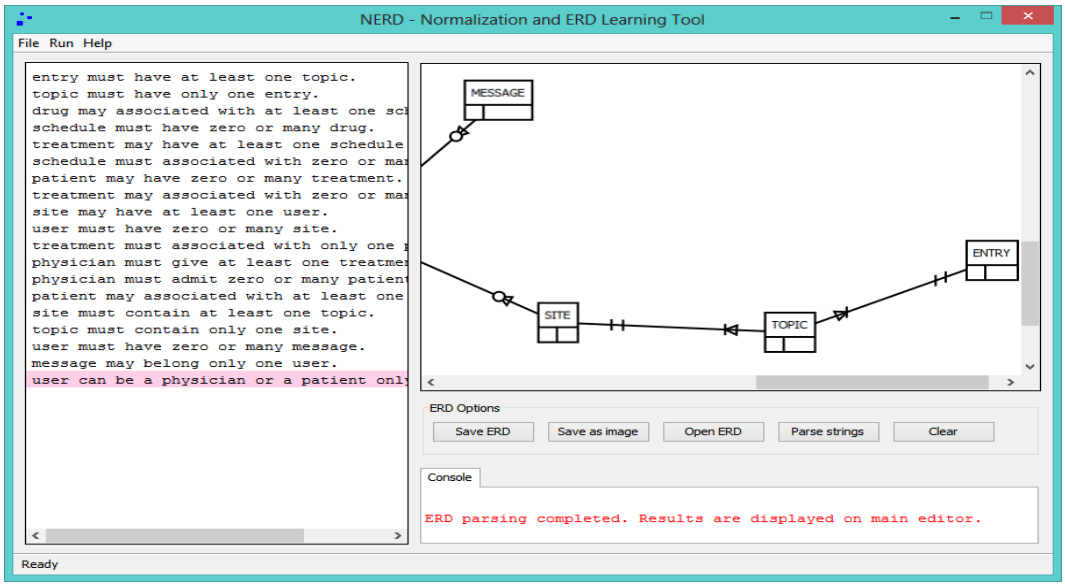

Figure 32: Parse string functionality, NERD

The Parse strings button generates the requirements if the ERD file is the only file available. After parsing the ERD, requirements are displayed back to the main editor.

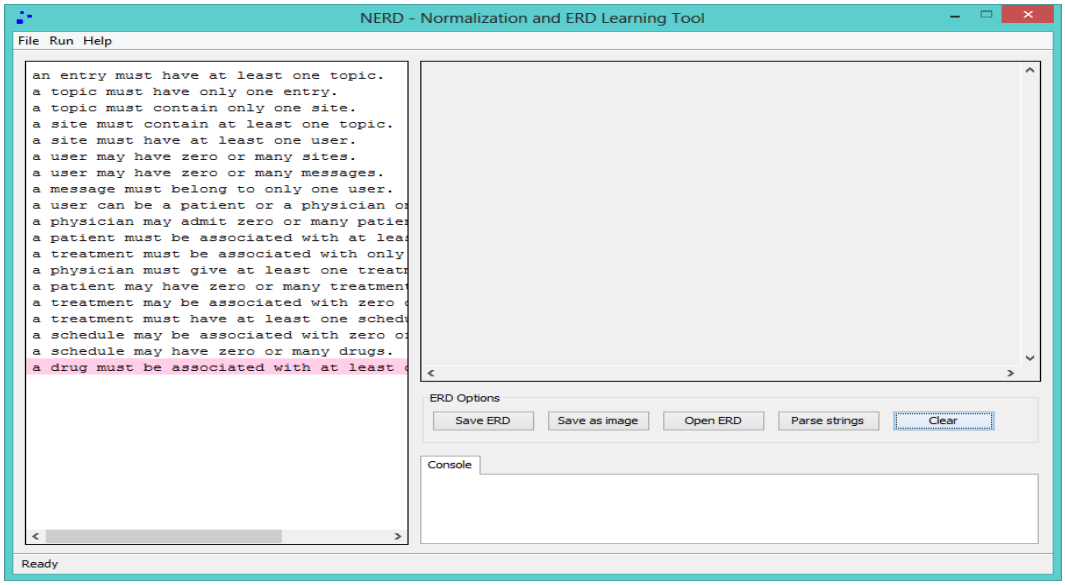

Figure 33: Clear functionality, NERD

The Clear button simply clears the rendering canvas.

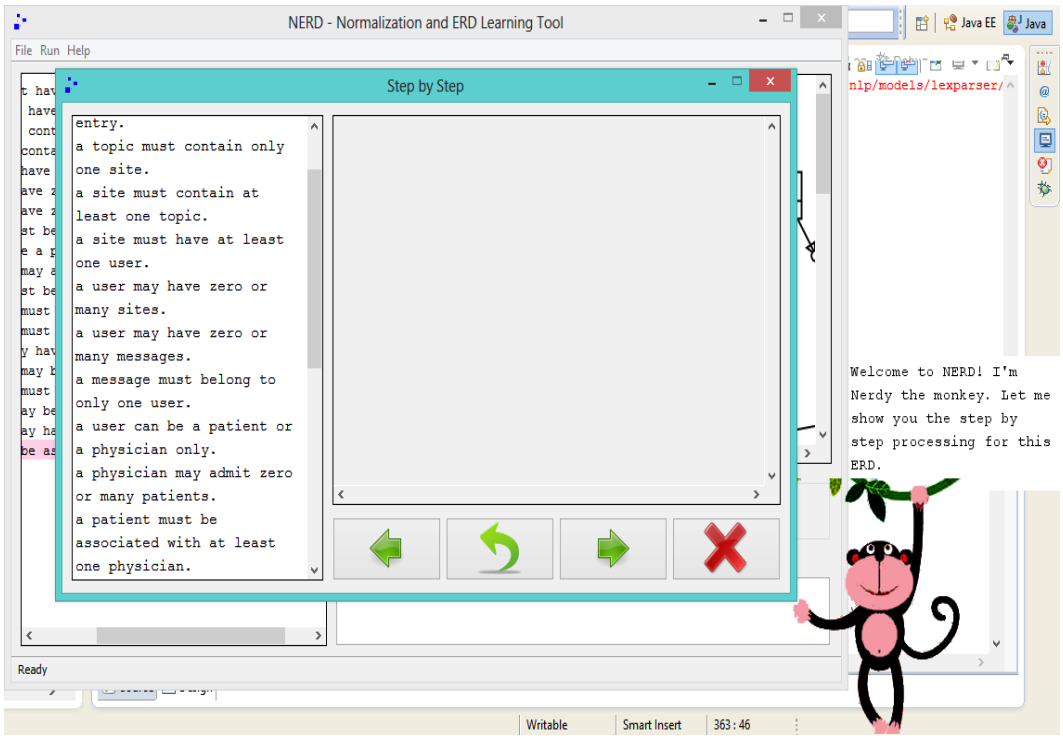

Figure 34: Step by step module step 1, NERD

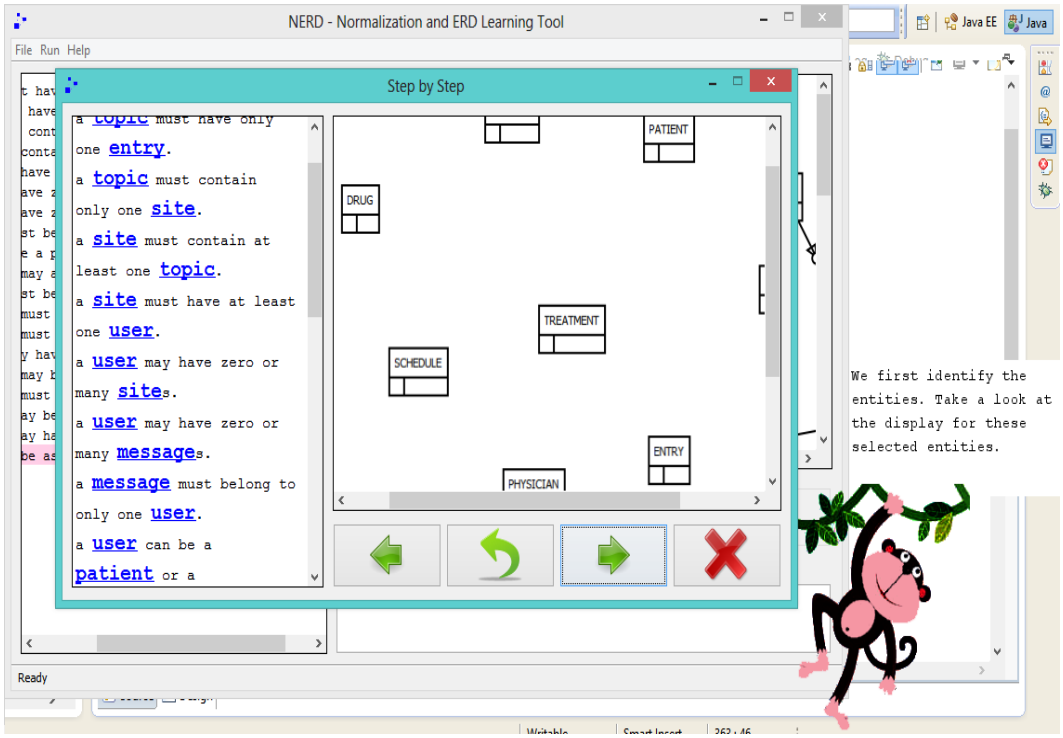

Figure 35: Step by step module step 2, NERD

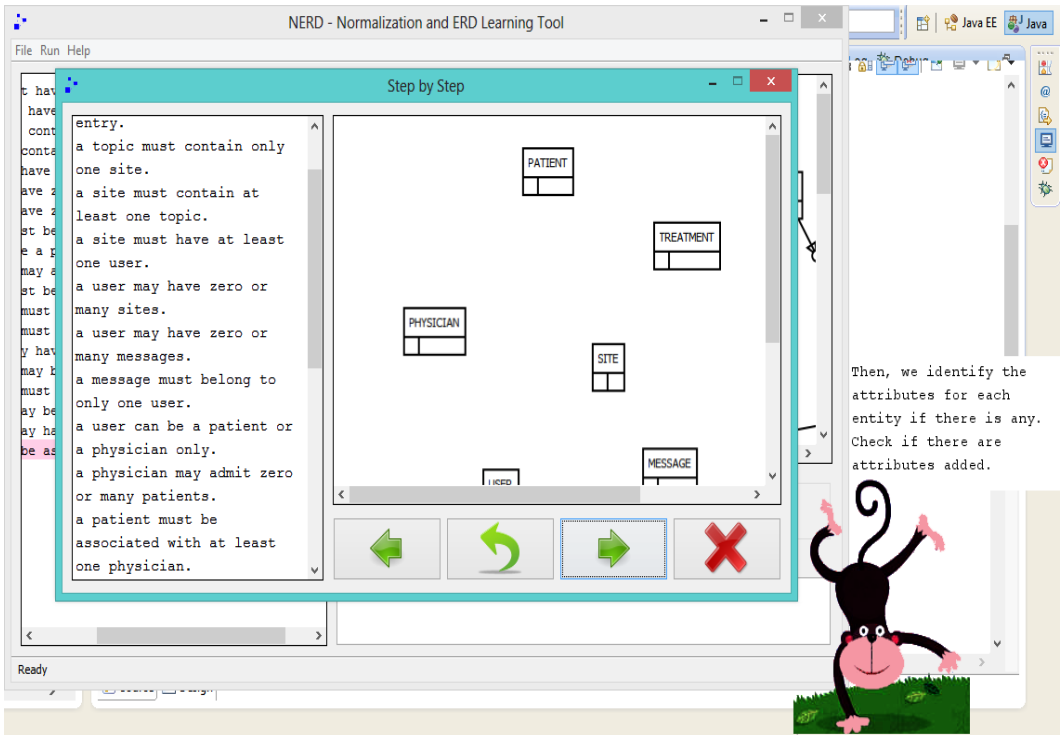

Figure 36: Step by step module step 3, NERD

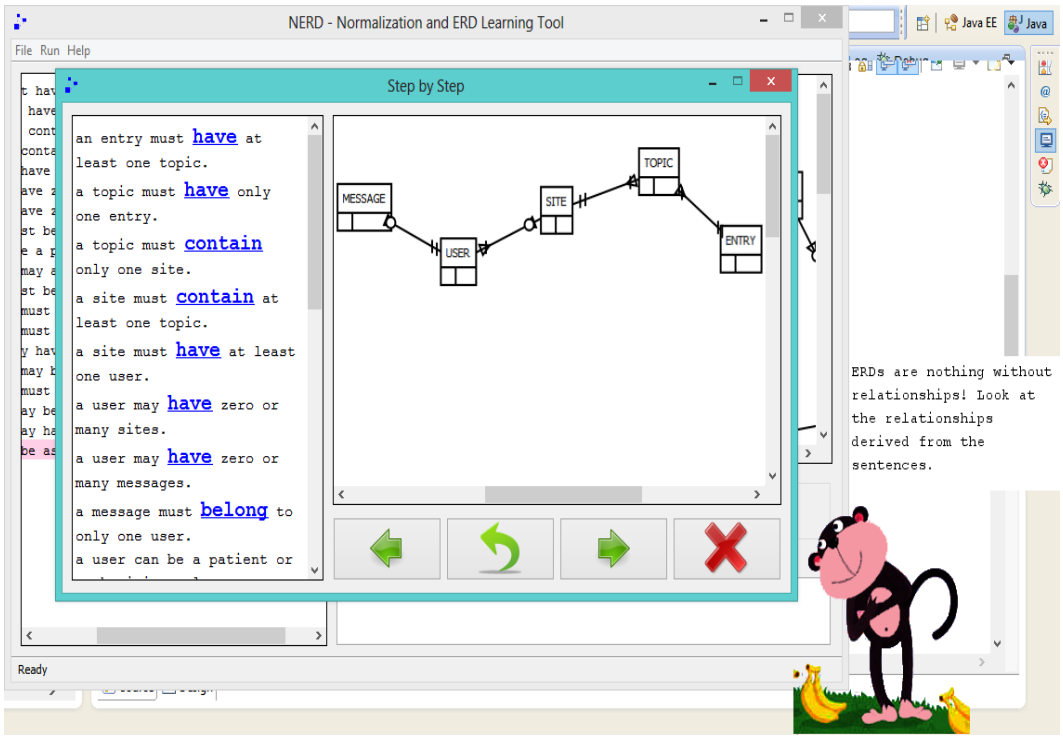

Figure 37: Step by step module step 4, NERD

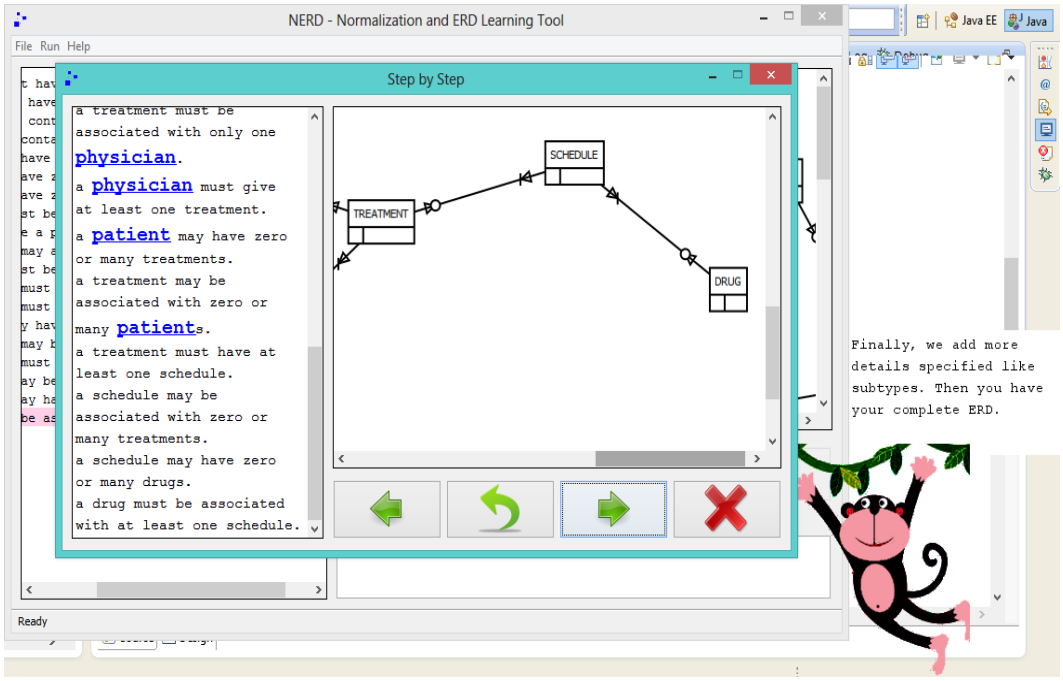

Figure 38: Step by step module step 5, NERD

Figures 34 to 38 displays the step by step module. In the step by step module, the student sees how the ERD is generated from the requirements. There are four buttons for next step, previous step, repeat, and close.

The first step (Figure 35) highlights all the entities which can be extracted from the requirements and displays them in the rendering canvas.

The second step (Figure 36) highlights all attributes which can be extracted from the requirements and displays them in the canvas contained in the corresponding entity.

The third step (Figure 37) highlights all the relations between entities and displays connected entities in the rendering canvas.

The last step (Figure 38) highlights all the subtypes which can be extracted and displays the final ERD in the rendering canvas.

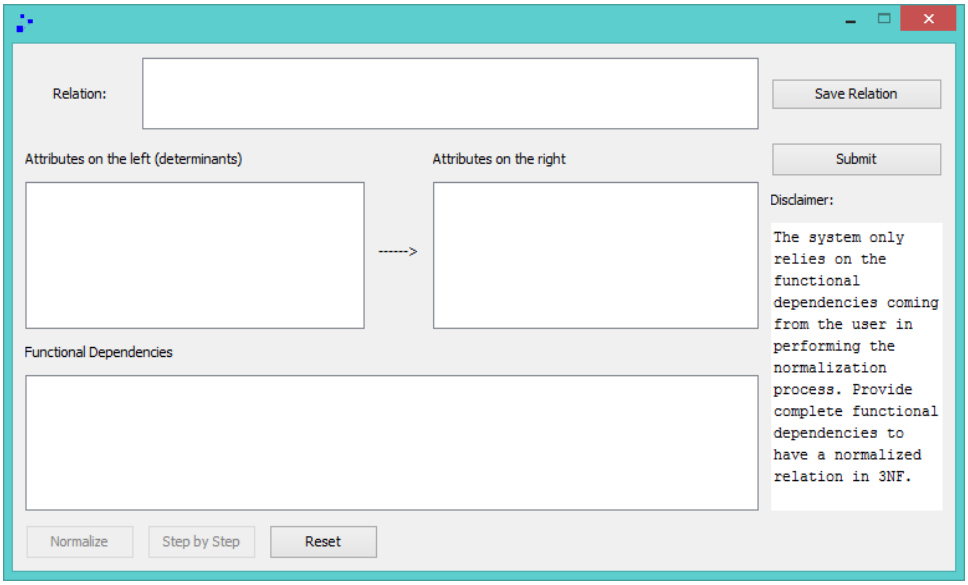

Figure 39: Normalization module, NERD

The relation is provided by the student in the relation textfield above. The relation must be bar delimited, delimiting each attribute of the relation. The student must click the Save Relation button and then the system would store the atributes in the list boxes below (Figure 40).

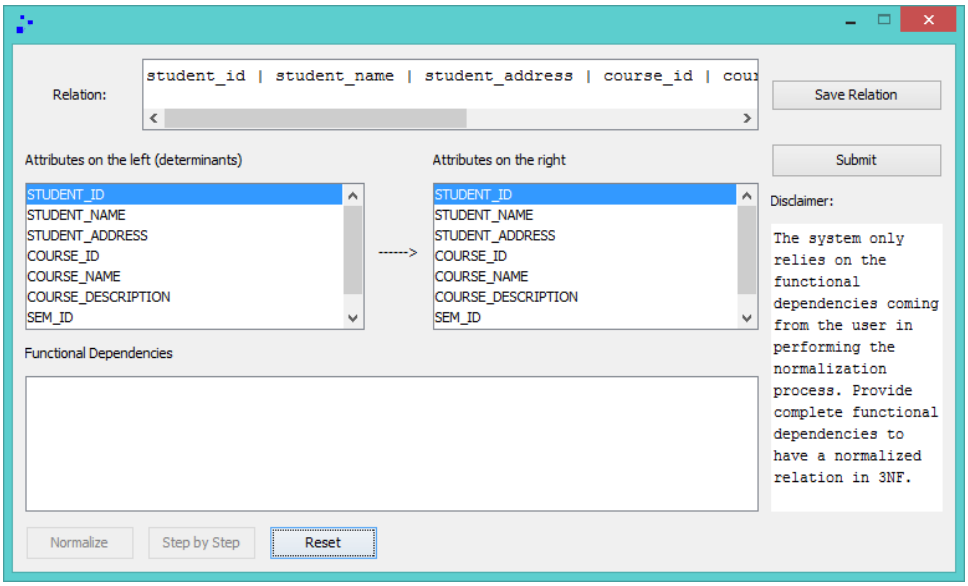

Figure 40: Normalization module with saved relation, NERD

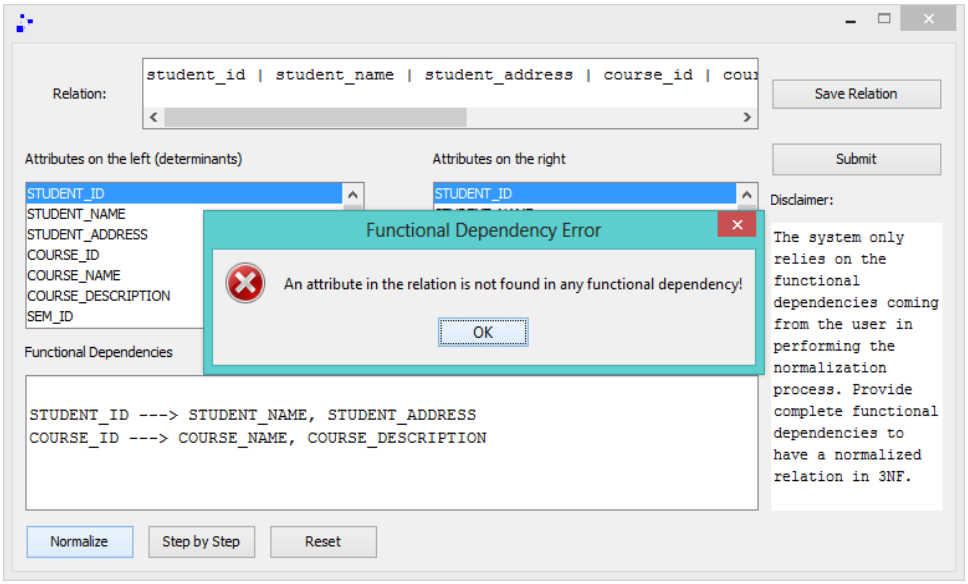

Figure 41: Attribute not in functional dependency error, NERD

All the attributes in the relation must be included in the functional dependencies. If one attribute is not included, an error message is displayed and the system does not proceed to normalization.

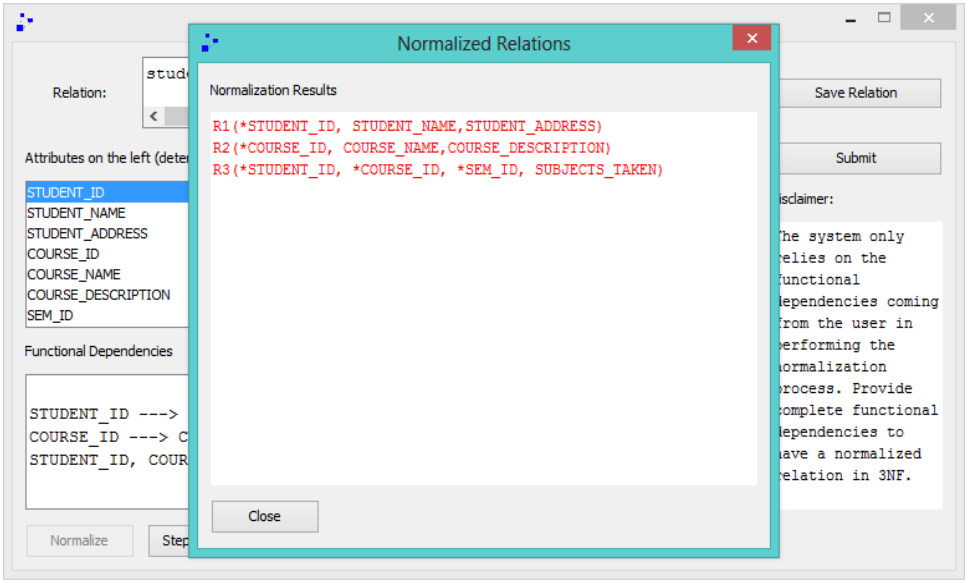

Figure 42: Normalization results, NERD

The normalization results are only based on the functional dependencies provided by the student. The relations are in 3NF.

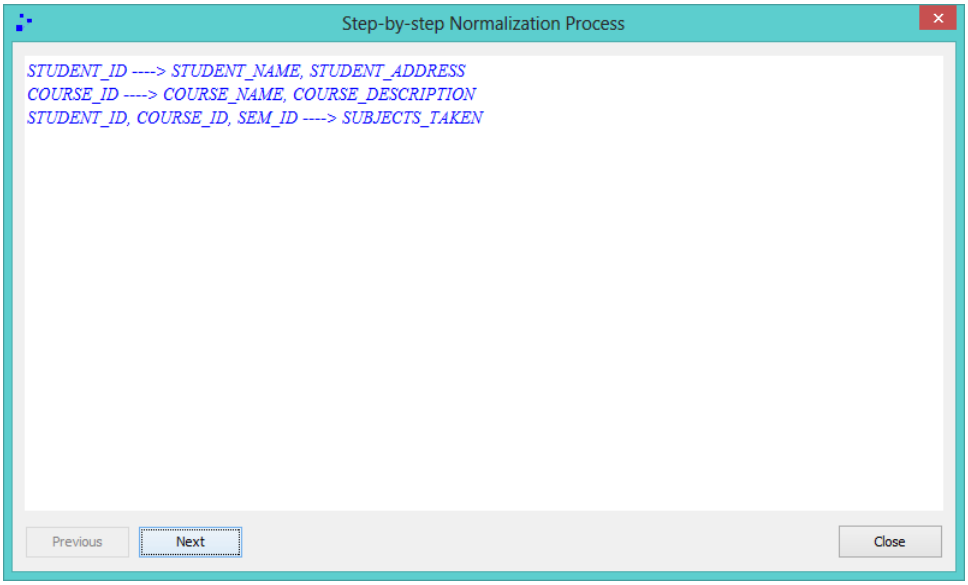

Figure 43: Normalization step by step process, NERD

The student can click the Step by step button and view the step by step nrmalization process.

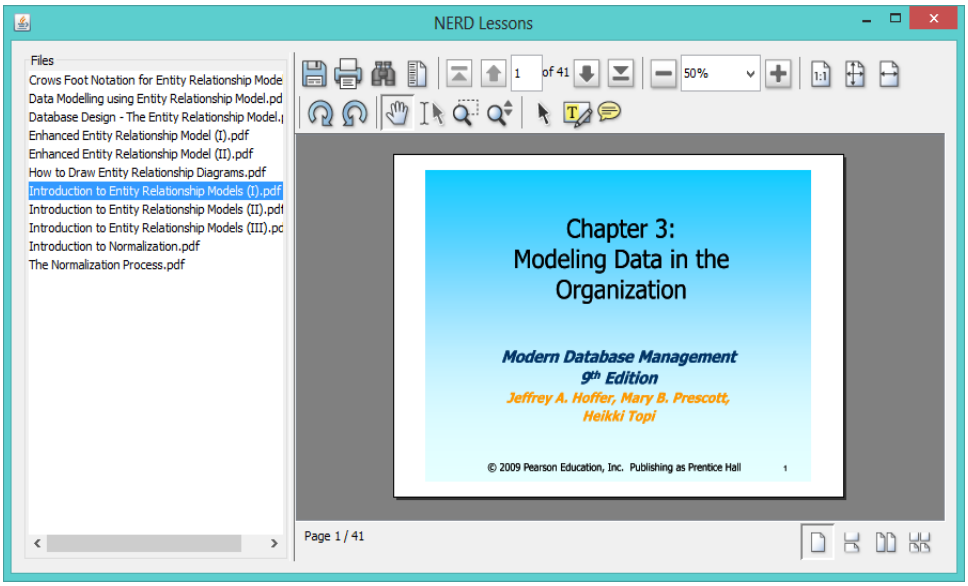

Figure 44: Lesson module, NERD

The lesson module loads the files from the lessons folder. The student may only add lessons by adding PDF files in the lessons folder.

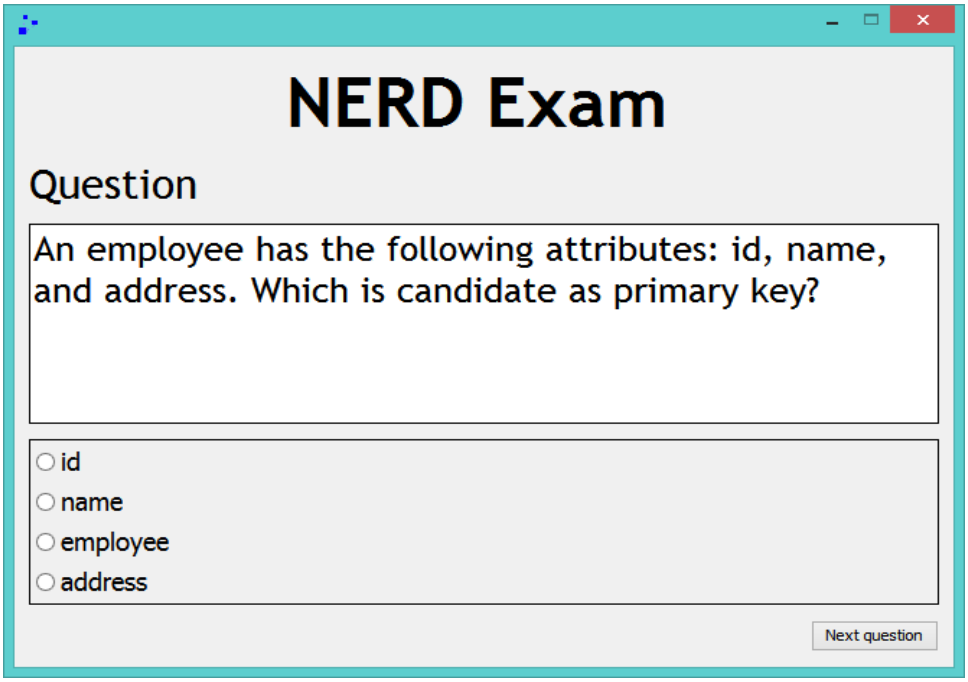

Figure 45: Exam module, NERD

The exam module provides a set of questions to the student to test his/her knowledge of the topic. There are 15 and displayed in random order.

VII. Discussions

NERD, a learning tool for database normalization and ERD development, is an application for learning ERD development and normalization.

NERD is a standalone application which is developed using Java. The system has four functionalities: generating ERD from text provided by the student, performing normalization process given relation and functional dependencies, displaying lessons regarding the topic to the student, and giving the student a set of questions which serves as an exam.

The first functionality, generating ERD from text, has three core modules in it: input validation, input parsing, which also includes data extraction, and output representation.

The first module, input validation, is a crucial part in the development of the system. Since the system allows the student to input free text, it has to be sure that the input is correct or valid before passing it to the next module.

The input passes validation if it has correct words, it follows a specific sentence structure, and it has no conflicting requirements.

Spelling checking for words is done through the use of a dictionary, and words are valid if they are contained in it. WordNet is first used as the dictionary for spelling checking. However, this library, does not contain some words essential to the system like the word 'has'. Also, this library provides so much more functionalities to simply be used for spelling checking. Instead of WordNet, a simple text file containing a list of words is used for spelling checking. The text file contains 500,889 words, with both singular and plural forms of a word present. The student may add words to this list if the system returns a spelling error.

For validating sentence structures, regular expressions are used. It is hard to validate whether a string is a valid sentence, moreover, to validate if a string is a valid sentence with a specific structure. To do this, the system used the POS tagging

module of the Stanford parser. This way, it is easier to check for sentence structure. Regular expressions are also backed up with keyword checking to avoid relying too much on regular expressions. The system checks for keywords which must be present in some sentence formats specified, along with checking with the regular expressions.

Conflicting requirements are checked using the algorithms presented in the design and implementation section C. For each requirement, if it passes the first two validation process, it proceeds to extraction. Newly read requirements are checked in the parsed components. This method is ideal if there would be no conflicts since it has already finished extraction when conflict checking returns no error. However, if there are errors, waiting time is increased since extracting is repeatedly done. Worst case would be having a conflict at the last requirement of the input. The reason behind this is that since the system has extracted from almost all the requirements, it has to repeat all over again the same process bbecause of conflict, and also increases waiting time for output.

Once the input passes the validation process, it is passed on to the next module.

The second module, parsing and data extraction, relies mostly on the third party library, Stanford parser. This module performs the large part of the system. Most of the functionalities used in this module are already implemented in the said library. These functionalities include the POS tagging used in validation and the typed dependency parsing which is important in this module.

This module processes the input, and extracts ERD components for final output. This is the longest process done by the system. The system tokenizes the paragraph into separate sentences then extracts data from each of them. The system identifies the sentence format, then applies the specific rule to extract the important components. However, method of the Stanford parser takes long processing time, which affected the overall processing time of the system.

The third module, output representation, generates the final ERD and displays it

to the student. A third party library is also used, Graphstream. Graphstream is a graph API library. This contains graph generating methods, graph algorithms, and graph layout modules.

A graph library is used instead of a graphics library since an ERD can be seen as a graph. An ERD can be seen as a graph, where nodes are entities and edges between nodes are present if the entities represented as nodes have a direct relationship with each other. The notion of using graphs to display ERD is used in [14]. Although a graphics package is also suitable for rendering, it would consume time compared to using an already available material which performs similar task.

Graphstream is powerful in generating graphs and it is also easy to customize to make a graph look like an ERD. It simply uses a CSS stylesheet to style nodes and edges. However, the system uses the layout provided by Graphstream which is not optimal layout for ERDs. Also, ERD are best rendered with orthogonal edges, which is not supported by graphstream. There are several graph libraries which offer the same level of customization but are not for free. Also, Others are implemented in Javascript and CSS3.

The second functionality, performing normalization process from relation and functional dependencies, has two core modules in it: direct normalization of the relation and step by step normalization of the relation. The user must provide the relation first with bar delimiter between the attributes, and save the relation.

The first module, direct normalization of the relation, directly performs the normalization of the relation up to third normal form. The student must provide all the functional dependencies for the relation in order to correctly normalize the relation. The student must input at least two functional dependencies before he can click the normalization process. Also, the student must include all the attributes in the functional dependencies, or the system will return an error. The step by step module can be performed if there are no errors found.

The third functionality, displaying lessons, displays a set of lessons regarding ERD and normalization. Some of the lessons are from the CMSC 127 slides, while others are PDF chapters of books from the internet. The lessons are stored as PDF files in a lessons directory. The student should not delete these files or the module will not display any lesson. If the student wishes to add more lessons, he/she must add the lesson directly to the lessons folder.

The last functionality, providing exams, gives the student a set of questions which he/she can answer to test his/her knowledge. The questions are more on learning concepts and does not have questions asking the user to draw ERD. The questions are loaded in random order. The exam functionality, however, loads only a fixed number of only 15 questions from a text file.

VIII. Conclusions

This study is able to create an application for learning database normalization and creation of entity relationship diagrams, NERD.

NERD allows the student to input database problem specifications and then it outputs the ERD which can be generated from the specifications. The student can read lessons which are included in the system. Users are also allowed to take a simple examination which is also part of the system.

NERD accepts a set of database problem specifications which can be in the form of a paragraph. NERD parses each sentence in the paragraph and extract ERD components. These ERD components are then connected in order to create the final ERD and display it to the student. These steps allow the student to quickly view the diagram that is generated from a set of specifications, and inspect where components came from and how components are related. NERD also has a step-by-step module which guides the student on how the ERD is generated from the specifications. The student may create new files or edit existing files. NERD also provides a module for generating the specifications or description of an ERD if the student only has an ERD file.

NERD also has a module for learning database normalization. The student provides a set of functional dependencies of a table to the system, and then the system performs a step-by-step process on how the current table transforms to its normalized form. The system, however, only generates normalized form up to 3NF only.

Data modelling is a hard task to learn and NERD provides a basic introduction and learning of the topic. NERD provides learning to its students through varried examples provided by the student. NERD gives its students a new learning experience where the student learns and tries to explore new examples instead of relearning the same examples repeatedly.

IX. Recommendations

After evaluating NERD, a normalization and ERD learning tool, there are certainly a lot of room for improvement.

One limitation of the application is that it accepts only specific sentence formats for input. The application would be better if there would be more sentence formats, or probably no more sentence formats at all. Adding more formats would allow the student to learn creation of ERD from any sentence format.

A disambiaguation feature may be added to the system. Disambiaguation is important to remove the notion in this study that a sentence extracts specific components only. Disambiguation module has an artificial intelligence part since the system must decide whether or not to use a certain word as entity or attribute or any other component. For example, through disambiguation, using *id* as an entity would raise an error since an *id* is more likely an attribute.

It is also good to have a better ERD representation and rendering like those on modelling applications like MS Visio, DIA, etc. Having other representation is ideal for this system since graph layouts used in this application is not optimal in ERD, and the students might have a hard time viewing the whole diagram.

Creating a more interactive environment which engages the student more into the activity can also be done to the system. One activity is allowing the student to manipulate the ERD via GUI interface like adding new entities and creating new relations through buttons, without typing any requirement.

A lot more improvements can be done on this application. Simply adding the things discussed above would greatly improve the application.

X. Bibliography

[1] D. M. Shahbaz, D. S. Ahsan, M. Shaheen, R. M. A. Nawab, and S. A. Masood, "Automatic generation of extended er diagram using natural language processing," *Journal of American Science*, 2011.

[2] Y. V. Dongare, P. S. Dhabe, and S. V. Deshmukh, "Rdbnorma: - a semi-automated tool for relational database schema normalization up to third normal form," *International Journal of Database Management Systems*, 2011.

[3] L. Ahmedi and E. Jajaga, "A database normalization tool using semantic web technologies," *International Journal of Systems Applications, Engineering and Development*, 2011.

[4] *Databases and Transaction Processing*, ch. 5. AW Higher Education, 2002.

[5] P. P.-S. Chen, "The entity-relationship model - toward a unified view of data," *ACM Transactions on Database Systems*, 1976.

[6] P. Suraweera, "Kermit: A knowledge-based entity relationship modelling intelligent tutor," in *New Zealand Computer Science Research Students' Conference (NZCSRSC), 2011.*

[7] A. Gordon and L. Hall, "A collaborative learning environment for data modelling," in *American Association of Artificial Intelligence.*

[8] M. Virvou and K. Tourtoglou, "An adaptive training environment for uml," in *Sixth International Conference on Advanced Learning Technologies.*

[9] J. Soler, I. Boada, F. Prados, J. Poch, and R. Fabregat, "A web-based e-learning tool for uml class diagrams," *IEEE EDUCON Education Engineering 2010 - The Future of Global Learning Engineering Education*, 2010.

[10] G. Gauthier, C. Frasson, and K. VanLehn, "A coached collaborative learning environment for entity-relationship modeling," in *5th International Conference In Intelligent Tutoring Systems, 2000.*

[11] H.-J. Kung and H.-L. Tung, "A web-based tool for teaching data modeling," *Journal of Computing Sciences in Colleges*, 2010.

[12] P. P.-S. Chen, "English sentence structure and entity-relationship models," *Information Systems*, 1983.

[13] N. Omar, J. Hanna, and P. McKevitt, "Heuristics-based entity-relationship modelling through natural language processing," in *Proc. of the 15th Artificial Intelligence and Cognitive Science Conference, 2004.*

[14] S. Du, *On the Use of Natural Language Processing For Automated Conceptual Data Modelling.* PhD thesis, University of Pittsburgh, 2008.

[15] Developing Entity-Relationship Diagrams (ERDs).

[16] M. Demba, "An algorithmic approach to database normalization," *International Journal of Digital Information and Wireless Communications*, 2013.

[17] *Relational Database Modeling and Database Design.*

[18] Retrieved from http://www.techopedia.com/definition/19504/functional-dependency.

[19] "What is computer-assisted instruction." Retrieved from: http://www.wisegeek.org/what-is-computer-assisted-instruction.htm.

[20] J. R. Allen, "Current trends in computer-assisted instruction," *Computers and the Humanities*, 1972.

[21] Retrieved from: http://searchdatamanagement.techtarget.com/definition/data-modeling.

[22] http://searchsoftwarequality.techtarget.com/definition/Unified-Modeling-Language.

[23] http://www.businessdictionary.com/definition/data-flow-diagram.html.

[24] Retrieved from: http://searchcontentmanagement.techtarget.com/definition/natural-language-processing-NLP.

[25] Retrieved from http://nlp.stanford.edu/software/lex-parser.shtml.

[26] http://grammar.about.com/od/pq/g/partsspeechterm.htm.

[27] http://www.cis.upenn.edu/~treebank/.

[28] *Stanford typed dependencies manual*.

[29] B. M. Marie-Catherine de Marneffe and C. D. Manning, "Generating typed dependency parses from phrase structure parses," in *LREC 2006*.

[30] http://downloads.sourceforge.net/wordlist/scowl-7.1.zip.

XI. Appendix

A. Source Code

```
/**
 * Main.java
 * Class containing the main method
 */
package com.kimcortez.sp.gui;

import java.awt.EventQueue;
import java.io.IOException;

import javax.swing.JFrame;
import javax.swing.UIManager;
import javax.swing.UnsupportedLookAndFeelException;

import com.kimcortez.sp.utils.Dictionary;

import edu.stanford.nlp.parser.lexparser.LexicalizedParser;
import java.awt.Toolkit;
import javax.swing.JLabel;
import java.awt.BorderLayout;
import javax.swing.ImageIcon;
import javax.swing.JPanel;
import java.awt.GridLayout;
import javax.swing.JButton;
import java.awt.Font;
import java.awt.Color;
import java.awt.Cursor;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import javax.swing.SwingConstants;

/**
 *
 * @author Kim Isle Cortez
 *
 */
public class Main implements ActionListener {
    private JFrame frmNerdNormalization;
    private LexicalizedParser lp;
    private Dictionary dict;
    private JButton btnInputProblems, btnViewLessons, btnAnswerExam;
    private JButton btnNormalizer;
    private JLabel lblNerdA;

    /**
     * Launch the application.
     */
    public static void main(String[] args) {
        try {
            UIManager.setLookAndFeel(UIManager
                    .getSystemLookAndFeelClassName());
        } catch (ClassNotFoundException | InstantiationException
                | IllegalAccessException
                | UnsupportedLookAndFeelException e1) {
            e1.printStackTrace();
        }
        final Splash splash = new Splash();
        EventQueue.invokeLater(new Runnable() {
            public void run() {
                try {
                    Main window = new Main();
```

```
            try {
                window.lp =
                        LexicalizedParser
                                .loadModel("edu/stanford/nlp/models/lexparser/englishPCFG.ser.gz");
                window.dict = new Dictionary();
                window.dict.load();
                System.err.println("Dictionary has been opened.");
                System.setProperty("org.graphstream.ui.renderer",
                        "org.graphstream.ui.j2dviewer.J2DGraphRenderer");
            } catch (IOException e) {
                e.printStackTrace();
            }
            splash.setVisible(false);
            window.frmNerdNormalization.setVisible(true);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
  });
}

/**
 * Create the application.
 */
public Main() {
  initialize();
}

/**
 * Initialize the contents of the frame.
 */
private void initialize() {
  frmNerdNormalization = new JFrame();
  frmNerdNormalization
      .setTitle("NERD - Normalization and ERD Learning Tool");
  frmNerdNormalization.setResizable(false);
  frmNerdNormalization
      .setIconImage(Toolkit
          .getDefaultToolkit()
          .getImage(
              Main.class
                  .getResource("/com/kimcortez/sp/gui/icons/icon.png")));
  frmNerdNormalization.setBounds(100, 100, 502, 363);
  frmNerdNormalization
      .setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
  frmNerdNormalization.setLocationRelativeTo(null);
  JPanel panel = new JPanel();
  panel.setBackground(Color.WHITE);
  frmNerdNormalization.getContentPane().add(panel,
      BorderLayout.CENTER);
  panel.setLayout(null);
  JLabel label = new JLabel("");
  label.setHorizontalTextPosition(SwingConstants.CENTER);
  label.setHorizontalAlignment(SwingConstants.CENTER);
  label.setIcon(new ImageIcon(Main.class
      .getResource("/com/kimcortez/sp/gui/icons/NERD.png")));
  label.setBounds(0, 0, 497, 233);
  panel.add(label);
  JPanel panel_1 = new JPanel();
  panel_1.setBounds(0, 267, 497, 67);
  panel.add(panel_1);
  panel_1.setLayout(new GridLayout(2, 2, 0, 0));
  btnInputProblems = new JButton("Generate ERD from Text");
  btnInputProblems.setCursor(Cursor
      .getPredefinedCursor(Cursor.HAND_CURSOR));
  btnInputProblems.setBorderPainted(false);
  btnInputProblems.setForeground(Color.BLUE);
  btnInputProblems.setBorder(null);
```

```
        btnInputProblems.setFont(new Font("Tahoma", Font.BOLD, 15));
        btnInputProblems.setFocusable(false);
        btnInputProblems.addActionListener(this);
        panel_1.add(btnInputProblems);
        btnViewLessons = new JButton("View Lessons");
        btnViewLessons.setCursor(Cursor
                .getPredefinedCursor(Cursor.HAND_CURSOR));
        btnViewLessons.setBorderPainted(false);
        btnViewLessons.setForeground(Color.BLUE);
        btnViewLessons.setBorder(null);
        btnViewLessons.setFont(new Font("Tahoma", Font.BOLD, 15));
        btnViewLessons.setFocusable(false);
        btnViewLessons.addActionListener(this);
        btnNormalizer = new JButton("Normalization Module");
        btnNormalizer.setFocusable(false);
        btnNormalizer.setFont(new Font("Tahoma", Font.BOLD, 15));
        btnNormalizer.setForeground(Color.BLUE);
        btnNormalizer.addActionListener(this);
        panel_1.add(btnNormalizer);
        panel_1.add(btnViewLessons);
        btnAnswerExam = new JButton("Answer Exam");
        btnAnswerExam.setCursor(Cursor
                .getPredefinedCursor(Cursor.HAND_CURSOR));
        btnAnswerExam.setBorderPainted(false);
        btnAnswerExam.setForeground(Color.BLUE);
        btnAnswerExam.setBorder(null);
        btnAnswerExam.setFont(new Font("Tahoma", Font.BOLD, 15));
        btnAnswerExam.setFocusable(false);
        btnAnswerExam.addActionListener(this);
        panel_1.add(btnAnswerExam);
        lblNerdA =
                new JLabel(
                        "NERD : A Learning Tool for Database Normalization and ERD Development");
        lblNerdA.setHorizontalTextPosition(SwingConstants.CENTER);
        lblNerdA.setHorizontalAlignment(SwingConstants.CENTER);
        lblNerdA.setBounds(0, 244, 497, 14);
        panel.add(lblNerdA);
    }

    @Override
    public void actionPerformed(ActionEvent e) {
        if (e.getSource() == btnAnswerExam) {
            ExamGui exam = new ExamGui();
            exam.setVisible(true);
        } else if (e.getSource() == btnViewLessons) {
            LessonGui lessongui = new LessonGui();
            lessongui.setVisible(true);
            lessongui.setLocationRelativeTo(null);
        } else if (e.getSource() == btnInputProblems) {
            InputGui inputGui = new InputGui();
            inputGui.setLp(lp);
            inputGui.setDict(dict);
            inputGui.setVisible(true);
        } else if (e.getSource() == btnNormalizer) {
            new NormalizationGui();
        }
    }
}

/**
 * InputGui.java
 * GUI for inputting problems
 */
package com.kimcortez.sp.gui;

import java.awt.Color;
import java.awt.Component;
import java.awt.Insets;
```

```
import java.awt.Toolkit;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.awt.event.InputEvent;
import java.awt.event.KeyEvent;
import java.awt.event.WindowAdapter;
import java.awt.event.WindowEvent;
import java.io.BufferedReader;
import java.io.BufferedWriter;
import java.io.File;
import java.io.FileReader;
import java.io.FileWriter;
import java.io.IOException;
import java.util.List;

import javax.swing.ImageIcon;
import javax.swing.JButton;
import javax.swing.JFileChooser;
import javax.swing.JFrame;
import javax.swing.JLabel;
import javax.swing.JMenu;
import javax.swing.JMenuBar;
import javax.swing.JMenuItem;
import javax.swing.JOptionPane;
import javax.swing.JPanel;
import javax.swing.JScrollPane;
import javax.swing.JSeparator;
import javax.swing.JTabbedPane;
import javax.swing.JTextArea;
import javax.swing.KeyStroke;
import javax.swing.ScrollPaneConstants;
import javax.swing.SwingConstants;
import javax.swing.border.LineBorder;
import javax.swing.border.TitledBorder;
import javax.swing.filechooser.FileFilter;
import javax.swing.filechooser.FileNameExtensionFilter;

import org.icepdf.ri.common.ComponentKeyBinding;
import org.icepdf.ri.common.SwingController;
import org.icepdf.ri.common.SwingViewBuilder;

import com.kimcortez.sp.erd.models.ERD;
import com.kimcortez.sp.erd.utils.ERDUtils;
import com.kimcortez.sp.parser.FileProcessor;
import com.kimcortez.sp.parser.models.CustomError;
import com.kimcortez.sp.utils.Dictionary;

import edu.stanford.nlp.parser.lexparser.LexicalizedParser;

/**
 * @author Kim Isle Cortez
 */
public class InputGui extends JFrame implements ActionListener {

    private static final long serialVersionUID = -46260477999190750L;
    private JLabel status;
    private JMenuItem mntmOpen, mntmNew, mntmSave, mntmExit, mntmRun,
            mntmNormalizer, mntmAbout,
            mntmStepByStep, mntmUserManual;
    private JButton saveErdBtn, saveImageBtn, openErdBtn,
            parseStringsBtn, clearBtn;;
    private JScrollPane erdPanel, editorPanel;
    private JFileChooser fileChooser;
    private JTextArea editor, console;
    private File currentFile = null;
    private LexicalizedParser lp;
    private Dictionary dict;
    private FileProcessor fileProcessor;
```

```
private ERD erd;
private ERDUtils erdUtils;
private JPanel erdOptionsPanel;

/**
 * Create the application.
 */
public InputGui() {
    initialize();
}

public void setLp(LexicalizedParser lp) {
    this.lp = lp;
}

public void setDict(Dictionary dict) {
    this.dict = dict;
}

/**
 * Initialize the contents of the
 */
private void initialize() {
    setTitle("NERD - Normalization and ERD Learning Tool");
    setIconImage(Toolkit
            .getDefaultToolkit()
            .getImage(
                    InputGui.class
                            .getResource("/com/kimcortez/sp/gui/icons/icon.png")));
    setResizable(false);
    setBounds(100, 100, 900, 630);
    setLocationRelativeTo(null);
    JMenuBar menuBar = new JMenuBar();
    setJMenuBar(menuBar);
    JMenu mnFile = new JMenu("File");
    menuBar.add(mnFile);
    mntmNew = new JMenuItem("New");
    mntmNew.setIcon(new ImageIcon(getClass().getResource(
            "icons/new.png")));
    mntmNew.setAccelerator(KeyStroke.getKeyStroke(KeyEvent.VK_N,
            InputEvent.CTRL_MASK));
    mntmNew.addActionListener(this);
    mnFile.add(mntmNew);
    mntmOpen = new JMenuItem("Open");
    mntmOpen.setIcon(new ImageIcon(getClass().getResource(
            "icons/open.png")));
    mntmOpen.setAccelerator(KeyStroke.getKeyStroke(KeyEvent.VK_O,
            InputEvent.CTRL_MASK));
    mntmOpen.addActionListener(this);
    mnFile.add(mntmOpen);
    mntmSave = new JMenuItem("Save");
    mntmSave.setIcon(new ImageIcon(getClass().getResource(
            "icons/save.png")));
    mntmSave.setAccelerator(KeyStroke.getKeyStroke(KeyEvent.VK_S,
            InputEvent.CTRL_MASK));
    mntmSave.addActionListener(this);
    mnFile.add(mntmSave);
    mntmExit = new JMenuItem("Exit");
    mntmExit.setIcon(new ImageIcon(getClass().getResource(
            "icons/exit.png")));
    mntmExit.setAccelerator(KeyStroke.getKeyStroke(KeyEvent.VK_F4,
            InputEvent.ALT_MASK));
    mnFile.add(mntmExit);
    JMenu mnRun = new JMenu("Run");
    menuBar.add(mnRun);
    mntmRun = new JMenuItem("Run");
    mntmRun.setIcon(new ImageIcon(getClass().getResource(
            "icons/play.png")));
```

```
mntmRun.setAccelerator(KeyStroke.getKeyStroke(KeyEvent.VK_R,
        InputEvent.CTRL_MASK));
mntmRun.addActionListener(this);
mnRun.add(mntmRun);
mntmNormalizer = new JMenuItem("Normalizer");
mntmNormalizer.addActionListener(this);
mntmStepByStep = new JMenuItem("Step_by_step");
mntmStepByStep.setEnabled(false);
mntmStepByStep.addActionListener(this);
mnRun.add(mntmStepByStep);
mnRun.add(mntmNormalizer);
JMenu mnHelp = new JMenu("Help");
menuBar.add(mnHelp);
mntmAbout = new JMenuItem("About");
mntmAbout.addActionListener(this);
mntmUserManual = new JMenuItem("User_Manual");
mntmUserManual.addActionListener(this);
mnHelp.add(mntmUserManual);
mnHelp.add(mntmAbout);
getContentPane().setLayout(null);
JPanel panel = new JPanel();
panel.setBounds(0, 559, 894, 21);
getContentPane().add(panel);
panel.setLayout(null);
status = new JLabel("Ready");
status.setHorizontalTextPosition(SwingConstants.CENTER);
status.setHorizontalAlignment(SwingConstants.CENTER);
status.setAlignmentX(Component.CENTER_ALIGNMENT);
status.setBounds(0, 0, 46, 21);
panel.add(status);
JSeparator separator_1 = new JSeparator();
separator_1.setBounds(0, 557, 894, 2);
getContentPane().add(separator_1);
erdPanel = new JScrollPane();
erdPanel
        .setHorizontalScrollBarPolicy(ScrollPaneConstants.HORIZONTAL_SCROLLBAR_ALWAYS);
erdPanel
        .setVerticalScrollBarPolicy(ScrollPaneConstants.VERTICAL_SCROLLBAR_ALWAYS);
erdPanel.setBorder(new LineBorder(new Color(0, 0, 0)));
erdPanel.setBounds(350, 11, 534, 356);
getContentPane().add(erdPanel);
editorPanel = new JScrollPane();
editorPanel.setBorder(new LineBorder(new Color(0, 0, 0)));
editorPanel.setBounds(10, 11, 330, 537);
getContentPane().add(editorPanel);
editor = new JTextArea();
editor.setMargin(new Insets(5, 5, 5, 5));
float[] color = new float[3];
Color.RGBtoHSB(255, 206, 231, color);
new LinePainter(editor, Color.getHSBColor(color[0], color[1],
        color[2]));
editorPanel.setViewportView(editor);
JTabbedPane tabbedPane = new JTabbedPane(JTabbedPane.TOP);
tabbedPane.setBorder(null);
tabbedPane.setBounds(350, 448, 534, 100);
getContentPane().add(tabbedPane);
JScrollPane scrollPane_1 = new JScrollPane();
scrollPane_1.setBorder(null);
tabbedPane.addTab("Console", null, scrollPane_1, null);
console = new JTextArea();
console.setForeground(Color.RED);
console.setEditable(false);
scrollPane_1.setViewportView(console);
erdOptionsPanel = new JPanel();
erdOptionsPanel.setBorder(new TitledBorder(null, "ERD_Options",
        TitledBorder.LEADING, TitledBorder.TOP, null, null));
erdOptionsPanel.setBounds(350, 378, 534, 59);
getContentPane().add(erdOptionsPanel);
```

```
    erdOptionsPanel.setLayout(null);
    saveErdBtn = new JButton("Save_ERD");
    saveErdBtn.setBounds(10, 21, 89, 23);
    erdOptionsPanel.add(saveErdBtn);
    saveErdBtn.setBorder(null);
    saveImageBtn = new JButton("Save_as_image");
    saveImageBtn.setBounds(109, 21, 89, 23);
    erdOptionsPanel.add(saveImageBtn);
    saveImageBtn.setBorder(null);
    openErdBtn = new JButton("Open_ERD");
    openErdBtn.setBounds(208, 21, 89, 23);
    erdOptionsPanel.add(openErdBtn);
    openErdBtn.setBorder(null);
    parseStringsBtn = new JButton("Parse_strings");
    parseStringsBtn.setBounds(307, 21, 89, 23);
    erdOptionsPanel.add(parseStringsBtn);
    parseStringsBtn.setBorder(null);
    clearBtn = new JButton("Clear");
    clearBtn.setBounds(406, 21, 89, 23);
    erdOptionsPanel.add(clearBtn);
    clearBtn.setBorder(null);
    clearBtn.addActionListener(this);
    parseStringsBtn.addActionListener(this);
    openErdBtn.addActionListener(this);
    saveImageBtn.addActionListener(this);
    saveErdBtn.addActionListener(this);
    fileChooser = new JFileChooser();
    FileFilter fileFilter =
            new FileNameExtensionFilter("Normal_text_file_(*.txt)",
                    "txt");
    fileChooser.setFileFilter(fileFilter);
    fileChooser.addChoosableFileFilter(fileFilter);
    fileChooser.setAcceptAllFileFilterUsed(false);
    erdUtils = new ERDUtils();
    addWindowListener(new WindowAdapter() {
        public void windowClosing(WindowEvent e) {
            File dir = new File("temp");
            for (File file : dir.listFiles()) {
                file.delete();
            }
            try {
                dict.saveWords();
            } catch (IOException e1) {
                e1.printStackTrace();
            }

        }
    });
    setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE);
}

@Override
public void actionPerformed(ActionEvent e) {
    if (e.getSource() == mntmExit) {
        System.exit(0);
    } else if (e.getSource() == mntmOpen) {
        if (fileChooser.showOpenDialog(null) == JFileChooser.APPROVE_OPTION) {
            currentFile = fileChooser.getSelectedFile();
            displayNewFile(currentFile);
            erdPanel.setViewportView(null);
        }
    } else if (e.getSource() == mntmSave) {
        saveFile();
    } else if (e.getSource() == mntmNew) {
        addNewProject();
    } else if (e.getSource() == mntmRun) {
        console.setText("");
        erdUtils.clear();
```

```
        erdPanel.setViewportView(null);
        editor.setText(editor.getText().trim());
        if (currentFile != null
                || editor.getText().trim().length() > 0) {
            startTutorial();
        }
        mntmStepByStep.setEnabled(true);
    } else if (e.getSource() == mntmNormalizer) {
        new NormalizationGui();
    } else if (e.getSource() == saveErdBtn) {
        if (erdUtils.getErd() == null) {
            JOptionPane.showMessageDialog(null,
                    "Error. There is no valid ERD content to save.",
                    "Save Error", JOptionPane.ERROR_MESSAGE);
        } else {
            erdUtils.save();
        }
    } else if (e.getSource() == saveImageBtn) {
        if (erdUtils.getErd() == null) {
            JOptionPane.showMessageDialog(null,
                    "Error. There is no valid ERD content to save.",
                    "Save Error", JOptionPane.ERROR_MESSAGE);
        } else {
            erdUtils.saveImage();
        }
    } else if (e.getSource() == openErdBtn) {
        ERD erd = erdUtils.open();
        if (erd != null) {
            erdUtils.setErd(erd);
            erdPanel.setViewportView(erdUtils.render());
        }
    } else if (e.getSource() == clearBtn) {
        erdUtils.clear();
        erdPanel.setViewportView(null);
        mntmStepByStep.setEnabled(false);
    } else if (e.getSource() == parseStringsBtn) {
        List<String> strings = erdUtils.parseStrings();
        if (strings != null) {
            editor.setText("");
            for (String sentence : strings) {
                editor.append(sentence.replaceAll("_", " ") + ".\n");
            }
            editor.setText(editor.getText().trim());
            console
                    .append("\nERD parsing completed. Results are displayed on main editor.");
        } else {
            JOptionPane.showMessageDialog(null,
                    "Error. Unable to parse incorrect ERD.",
                    "Parse Error", JOptionPane.ERROR_MESSAGE);
        }
        editor.requestFocus();
    } else if (e.getSource() == mntmUserManual) {
        SwingController controller = new SwingController();
        SwingViewBuilder factory =
                new SwingViewBuilder(controller);
        JFrame frame = new JFrame();
        JPanel panel = factory.buildViewerPanel();
        ComponentKeyBinding.install(controller, panel);
        controller.getDocumentViewController()
                .setAnnotationCallback(
                        new org.icepdf.ri.common.MyAnnotationCallback(
                                controller.getDocumentViewController()));
        controller.openDocument(getClass().getResource("resources/NERD_User_Manual.pdf"));
        frame.setContentPane(panel);
        frame.setSize(this.getWidth(), this.getHeight());
        frame.setVisible(true);
    } else if (e.getSource() == mntmAbout) {
        AboutGui about = new AboutGui();
```

```
            about.setModal(true);
            about.setVisible(true);
        } else if (e.getSource() == mntmStepByStep) {
            LearningGui learningGui = new LearningGui();
            learningGui.setContent(editor.getText());
            fileProcessor.setContent(editor.getText());
            fileProcessor.process();
            erd =
                    new ERD(fileProcessor.getEntities(),
                            fileProcessor.getRelations());
            erdUtils.clear();
            erdUtils.setErd(erd);
            learningGui.setErdUtils(erdUtils);
            learningGui.setVisible(true);
            learningGui.setLocationRelativeTo(null);
        }
        revalidate();
        repaint();
    }

    /**
     * Validates and then starts content processing
     */
    private void startTutorial() {
        fileProcessor = new FileProcessor(lp, dict);
        CustomError error =
                fileProcessor.isValidContent(editor.getText());
        if (error == null) {
            fileProcessor.setContent(editor.getText());
            CustomError conflictError = fileProcessor.process();
            if (conflictError == null) {
                erd =
                        new ERD(fileProcessor.getEntities(),
                                fileProcessor.getRelations());
                erdUtils.clear();
                erdUtils.setErd(erd);
                erdPanel.setViewportView(erdUtils.render());
            } else {
                console.append(conflictError.toString());
            }
        } else {
            console.append(error.toString());
        }
        revalidate();
        repaint();
    }

    /**
     * Add new project tab on the display
     */
    private void addNewProject() {
        saveFile();
        editor.setText("");
        currentFile = null;
    }

    /**
     * Display new file on the editor
     *
     * @param file
     * - the file to be displayed
     */
    private void displayNewFile(File file) {
        BufferedReader br;
        try {
            br = new BufferedReader(new FileReader(file));
            editor.read(br, null);
            br.close();
```

```
        } catch (IOException e) {
            e.printStackTrace();
        }
        editor.requestFocus();
    }

    /**
     * Writes content of the editor to a file
     *
     * @param file
     * - the file to write the content on
     */
    private void writeToFile(File file) {
        BufferedWriter bw;
        try {
            bw = new BufferedWriter(new FileWriter(file));
            editor.write(bw);
            bw.close();
        } catch (IOException e) {
            e.printStackTrace();
        }
        currentFile = file;
        editor.requestFocus();
    }

    /**
     * Saves the current file, or the newly created file
     */
    private void saveFile() {
        if (currentFile != null) {
            writeToFile(currentFile);
            displayNewFile(currentFile);
        } else {
            if (fileChooser.showSaveDialog(null) == JFileChooser.APPROVE_OPTION) {
                File newFile = fileChooser.getSelectedFile();
                String filename = newFile + "";
                if (!filename.substring(filename.lastIndexOf(".") + 1,
                        filename.length()).equals("txt")) {
                    filename += ".txt";
                }
                writeToFile(new File(filename));
                displayNewFile(new File(filename));
            }
        }
        editor.requestFocus();
    }
}

/**
 * LearningGui.java
 * The GUI for step by step function
 */
package com.kimcortez.sp.gui;

import java.awt.Color;
import java.awt.Font;
import java.awt.GridLayout;
import java.awt.Toolkit;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.awt.event.WindowAdapter;
import java.awt.event.WindowEvent;
import java.util.ArrayList;
import java.util.HashSet;
import java.util.Set;

import javax.swing.ImageIcon;
import javax.swing.JButton;
```

```
import javax.swing.JFrame;
import javax.swing.JPanel;
import javax.swing.JScrollPane;
import javax.swing.JTextPane;
import javax.swing.ScrollPaneConstants;
import javax.swing.border.LineBorder;
import javax.swing.text.Style;
import javax.swing.text.StyleConstants;

import org.apache.commons.lang.SerializationUtils;

import com.kimcortez.sp.erd.models.Attribute;
import com.kimcortez.sp.erd.models.ERD;
import com.kimcortez.sp.erd.models.Entity;
import com.kimcortez.sp.erd.models.Relationship;
import com.kimcortez.sp.erd.utils.ERDUtils;

/**
 *
 * @author Kim Isle Cortez
 *
 */
public class LearningGui extends JFrame implements ActionListener {
    private static final long serialVersionUID = 3989727100385731639L;
    private int step = 0;
    private JButton btnPrev, btnRepeat, btnNext, btnClose;
    private JScrollPane graphPanel;
    private ERDUtils erdUtils;
    private ERD originalERD;
    private TutorGui tutorGui;
    private JTextPane textArea;
    private Style plain, formatted;

    public ERDUtils getErdUtils() {
        return erdUtils;
    }

    public void setErdUtils(ERDUtils erdUtils) {
        this.erdUtils = erdUtils;
        originalERD = this.erdUtils.getErd();
    }

    public void setContent(String content) {
        textArea.setText(content);
    }

    public LearningGui() {
        initialize();
        initializeStyles();
        tutorGui = new TutorGui();
        tutorGui.setVisible(true);
        addWindowListener(new WindowAdapter() {
            public void windowClosing(WindowEvent e) {
                step = 0;
                tutorGui.dispose();
            }
        });
    }

    /**
     * Create the frame.
     */
    public void initialize() {
        setSize(790, 470);
        setResizable(false);
        setTitle("Step_by_Step");
        setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE);
        setIconImage(Toolkit.getDefaultToolkit().getImage(
```

```
            getClass().getResource("icons/icon.png")));
    getContentPane().setLayout(null);
    JScrollPane editorPanel = new JScrollPane();
    editorPanel.setBorder(new LineBorder(new Color(0, 0, 0)));
    editorPanel.setBounds(10, 11, 271, 419);
    getContentPane().add(editorPanel);
    textArea = new JTextPane();
    textArea.setFont(new Font("Monospaced", Font.PLAIN, 15));
    textArea.setEditable(false);
    editorPanel.setViewportView(textArea);
    graphPanel = new JScrollPane();
    graphPanel
            .setVerticalScrollBarPolicy(ScrollPaneConstants.VERTICAL_SCROLLBAR_ALWAYS);
    graphPanel
            .setHorizontalScrollBarPolicy(ScrollPaneConstants.HORIZONTAL_SCROLLBAR_ALWAYS);
    graphPanel.setBorder(new LineBorder(new Color(0, 0, 0)));
    graphPanel.setBounds(291, 11, 483, 351);
    getContentPane().add(graphPanel);
    JPanel panel = new JPanel();
    panel.setBounds(291, 373, 483, 57);
    getContentPane().add(panel);
    panel.setLayout(new GridLayout(1, 4, 5, 5));
    btnPrev = new JButton("");
    btnPrev.setBorder(null);
    btnPrev.setIcon(new ImageIcon(getClass().getResource(
            "icons/prev.png")));
    btnPrev.addActionListener(this);
    panel.add(btnPrev);
    btnRepeat = new JButton("");
    btnRepeat.setBorder(null);
    btnRepeat.setIcon(new ImageIcon(getClass().getResource(
            "icons/back.png")));
    btnRepeat.addActionListener(this);
    panel.add(btnRepeat);
    btnNext = new JButton("");
    btnNext.setBorder(null);
    btnNext.setIcon(new ImageIcon(getClass().getResource(
            "icons/next.png")));
    btnNext.addActionListener(this);
    panel.add(btnNext);
    btnClose = new JButton("");
    btnClose.setBorder(null);
    btnClose.setIcon(new ImageIcon(getClass().getResource(
            "icons/deleteBig.png")));
    btnClose.addActionListener(this);
    panel.add(btnClose);
}

@Override
public void actionPerformed(ActionEvent e) {
    if (e.getSource() == btnClose) {
        this.dispose();
        tutorGui.dispose();
    } else if (e.getSource() == btnRepeat) {
        step = 0;
        erdUtils.clear();
        graphPanel.setViewportView(null);
        textArea.getStyledDocument().setCharacterAttributes(0,
                textArea.getText().length(), plain, true);
    } else if (e.getSource() == btnNext && step < 4) {
        step++;
        displayStep();
    } else if (e.getSource() == btnPrev && step > 0) {
        step--;
        displayStep();
    }
}
```

```
private void initializeStyles() {
    plain = textArea.addStyle("plain", null);
    formatted = textArea.addStyle("red", null);
    StyleConstants.setFontSize(formatted, 20);
    StyleConstants.setUnderline(formatted, true);
    StyleConstants.setBold(formatted, true);
    StyleConstants.setForeground(formatted, Color.BLUE);
    StyleConstants.setBold(plain, false);
}

private void displayStep() {
    erdUtils.setErd(getStepERD());
    graphPanel.setViewportView(erdUtils.render());
    revalidate();
    repaint();
}

private ERD getStepERD() {
    textArea.getStyledDocument().setCharacterAttributes(0,
            textArea.getText().length(), plain, true);
    ERD erd = (ERD) SerializationUtils.clone(originalERD);
    switch (step) {
    case 0:
        erd =
                new ERD(new HashSet<Entity>(),
                        new HashSet<Relationship>());
        tutorGui.setIcon(new ImageIcon(getClass().getResource(
                "images/3.png")));
        tutorGui.setText("Welcome to NERD! I'm Nerdy the monkey. "
                + "Let me show you the step by step processing "
                + "for this ERD.");
        break;
    case 1:
        removeAttributes(erd);
        removeRelationships(erd);
        removeSubtypes(erd);
        addWordStyles(erd.getEntities(), false, false);
        tutorGui.setIcon(new ImageIcon(getClass().getResource(
                "images/5.png")));
        tutorGui.setText("We first identify the entities. Take a "
                + "look at the display for these selected entities.");
        break;
    case 2:
        removeRelationships(erd);
        removeSubtypes(erd);
        addWordStyles(erd.getEntities(), true, false);
        tutorGui.setIcon(new ImageIcon(getClass().getResource(
                "images/4.png")));
        tutorGui
                .setText("Then, we identify the attributes for each "
                        + "entity if there is any. Check if there "
                        + "are attributes added.");
        break;
    case 3:
        removeSubtypes(erd);
        addWordStyles(erd.getRelationships());
        tutorGui.setIcon(new ImageIcon(getClass().getResource(
                "images/1.png")));
        tutorGui
                .setText("ERDs are nothing without relationships! "
                        + "Look at the relationships derived from the sentences.");
        break;
    case 4:
        addWordStyles(erd.getEntities(), false, true);
        tutorGui.setIcon(new ImageIcon(getClass().getResource(
                "images/2.png")));
        tutorGui.setText("Finally, we add more details specified "
                + "like subtypes. Then you have your complete ERD.");
```

```
            break;
        }
        return erd;
    }

    private void removeSubtypes(ERD erd) {
        Set<Entity> entities = erd.getEntities();
        for (Entity e : entities) {
            e.setSubtypes(new HashSet<Entity>());
        }
    }

    private void removeRelationships(ERD erd) {
        erd.setRelationships(new HashSet<Relationship>());
    }

    private void removeAttributes(ERD erd) {
        Set<Entity> entities = erd.getEntities();
        for (Entity e : entities) {
            e.setAttributes(new ArrayList<Attribute>());
        }
    }

    private void addWordStyles(Set<Entity> entities, boolean attrOnly,
            boolean subtypeOnly) {
        if (attrOnly) {
            for (Entity e : entities) {
                for (Attribute a : e.getAttributes()) {
                    addStyle(a.getName());
                }
            }
        } else if (subtypeOnly) {
            for (Entity e : entities) {
                for (Entity s : e.getSubtypes()) {
                    addStyle(s.getName());
                }
            }
        } else {
            for (Entity e : entities) {
                addStyle(e.getName());
            }
        }
    }

    private void addWordStyles(Set<Relationship> relationships) {
        for (Relationship r : relationships) {
            addStyle(r.getFirstLabel().replaceAll("[␣]", "␣"));
            addStyle(r.getSecondLabel().replaceAll("[␣]", "␣"));
        }
    }

    private void addStyle(String word) {
        String text = textArea.getText();
        for (int i = -1; (i = text.indexOf(word, i + 1)) != -1;) {
            textArea.getStyledDocument().setCharacterAttributes(i,
                    word.length(), formatted, true);
        }
    }
}

/**
 * ExamGui.java
 * GUI for exam function
 */
package com.kimcortez.sp.gui;

import java.awt.Color;
import java.awt.Component;
```

```
import java.awt.Font;
import java.awt.GridLayout;
import java.awt.Toolkit;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.io.BufferedReader;
import java.io.File;
import java.io.FileReader;
import java.io.IOException;
import java.util.ArrayList;
import java.util.List;
import java.util.Random;
import java.util.StringTokenizer;

import javax.swing.ButtonGroup;
import javax.swing.JButton;
import javax.swing.JFrame;
import javax.swing.JLabel;
import javax.swing.JOptionPane;
import javax.swing.JPanel;
import javax.swing.JRadioButton;
import javax.swing.JTextArea;
import javax.swing.SwingConstants;
import javax.swing.UIManager;
import javax.swing.border.CompoundBorder;
import javax.swing.border.EmptyBorder;
import javax.swing.border.LineBorder;

/**
 *
 * @author Kim Isle Cortez
 *
 */
public class ExamGui extends JFrame implements ActionListener {
    private static final long serialVersionUID = -4183094980699258793L;
    private JPanel contentPane;
    private JTextArea questionArea;
    private JButton btnNext;
    private JLabel lblQuestionNo;
    private JRadioButton rdBtnA, rdBtnB, rdBtnC, rdBtnD;
    private ButtonGroup btnGroup;
    private int score = 0;
    private List<String> questions = new ArrayList<String>();
    private List<String> choices = new ArrayList<String>();
    private String answer;

    public ExamGui() {
        loadQuestions();
        init();
    }

    private void init() {
        setIconImage(Toolkit
                .getDefaultToolkit()
                .getImage(
                        ExamGui.class
                                .getResource("/com/kimcortez/sp/gui/icons/icon.png")));
        setDefaultCloseOperation(JFrame.HIDE_ON_CLOSE);
        setBounds(100, 100, 657, 478);
        setResizable(false);
        contentPane = new JPanel();
        contentPane.setBorder(new EmptyBorder(5, 5, 5, 5));
        setContentPane(contentPane);
        contentPane.setLayout(null);
        btnNext = new JButton("Next_question");
        btnNext.setBorder(null);
        btnNext.setAlignmentX(Component.CENTER_ALIGNMENT);
        btnNext.setBounds(552, 415, 89, 23);
```

```
    btnNext.addActionListener(this);
    contentPane.add(btnNext);
    JLabel lblNerdExam = new JLabel("NERD Exam");
    lblNerdExam.setFont(new Font("Trebuchet MS", Font.BOLD, 50));
    lblNerdExam.setHorizontalTextPosition(SwingConstants.CENTER);
    lblNerdExam.setHorizontalAlignment(SwingConstants.CENTER);
    lblNerdExam.setAlignmentX(Component.CENTER_ALIGNMENT);
    lblNerdExam.setBounds(10, 11, 621, 59);
    contentPane.add(lblNerdExam);
    lblQuestionNo = new JLabel("Question");
    lblQuestionNo.setFont(new Font("Trebuchet MS", Font.PLAIN, 30));
    lblQuestionNo.setBounds(10, 81, 631, 36);
    contentPane.add(lblQuestionNo);
    JPanel panel = new JPanel();
    panel.setBorder(new LineBorder(new Color(0, 0, 0)));
    panel.setBounds(10, 284, 631, 120);
    contentPane.add(panel);
    panel.setLayout(new GridLayout(4, 1, 0, 0));
    btnGroup = new ButtonGroup();
    rdBtnA = new JRadioButton("");
    rdBtnA.setFont(new Font("Tahoma", Font.PLAIN, 18));
    panel.add(rdBtnA);
    rdBtnB = new JRadioButton("");
    rdBtnB.setFont(new Font("Tahoma", Font.PLAIN, 18));
    panel.add(rdBtnB);
    rdBtnC = new JRadioButton("");
    rdBtnC.setFont(new Font("Tahoma", Font.PLAIN, 18));
    panel.add(rdBtnC);
    rdBtnD = new JRadioButton("");
    rdBtnD.setFont(new Font("Tahoma", Font.PLAIN, 18));
    panel.add(rdBtnD);
    btnGroup.add(rdBtnA);
    btnGroup.add(rdBtnB);
    btnGroup.add(rdBtnC);
    btnGroup.add(rdBtnD);
    questionArea = new JTextArea();
    questionArea.setWrapStyleWord(true);
    questionArea.setLineWrap(true);
    questionArea.setBorder(new CompoundBorder(new LineBorder(
            new Color(0, 0, 0)), UIManager
            .getBorder("CheckBoxMenuItem.border")));
    questionArea.setEditable(false);
    questionArea.setFont(new Font("Trebuchet MS", Font.PLAIN, 25));
    questionArea.setBounds(10, 128, 631, 145);
    contentPane.add(questionArea);
    setLocationRelativeTo(null);
    getQuestion();
}

@Override
public void actionPerformed(ActionEvent e) {
    if (e.getSource() == btnNext) {
        JOptionPane.showMessageDialog(null, "You answered : "
                + getAnswer() + "\nThe correct answer is : " + answer,
                "Answer", JOptionPane.PLAIN_MESSAGE);
        if (getAnswer().equalsIgnoreCase(answer)) {
            score++;
        }
        getQuestion();
    }
}

private void getQuestion() {
    if (!questions.isEmpty()) {
        int index = new Random().nextInt(questions.size());
        String question = questions.remove(index);
        String choices2 = choices.remove(index);
        String[] parts = choices2.split("[=]");
```

```
            answer = parts[1];
            StringTokenizer tok = new StringTokenizer(parts[0], ",");
            rdBtnA.setText(tok.nextToken());
            rdBtnB.setText(tok.nextToken());
            rdBtnC.setText(tok.nextToken());
            rdBtnD.setText(tok.nextToken());
            btnGroup.clearSelection();
            questionArea.setText(question);
        } else {
            JOptionPane.showMessageDialog(this,
                    "Exam completed! You scored " + score + ".", "Done",
                    JOptionPane.INFORMATION_MESSAGE);
            this.dispose();
        }
    }

    private String getAnswer() {
        if (rdBtnA.isSelected()) {
            return rdBtnA.getText();
        } else if (rdBtnB.isSelected()) {
            return rdBtnB.getText();
        } else if (rdBtnC.isSelected()) {
            return rdBtnC.getText();
        } else if (rdBtnD.isSelected()) {
            return rdBtnD.getText();
        } else {
            return "";
        }
    }

    private void loadQuestions() {
        try {
            File questions =
                    new File(getClass().getResource(
                            "questions/questions.txt").getFile());
            BufferedReader br =
                    new BufferedReader(new FileReader(questions));
            String line = br.readLine();
            while (line != null) {
                storeToMap(line);
                line = br.readLine();
            }
            br.close();
        } catch (IOException e) {
            e.printStackTrace();
        }
    }

    private void storeToMap(String line) {
        StringTokenizer tok = new StringTokenizer(line, "#");
        questions.add(tok.nextToken());
        choices.add(tok.nextToken());
    }
}

/**
 * LessonGui.java
 * The GUI for lessons
 */
package com.kimcortez.sp.gui;

import java.awt.Dimension;
import java.awt.GridLayout;
import java.io.File;
import java.io.IOException;
import java.nio.file.Files;

import javax.swing.DefaultListModel;
```

```
import javax.swing.JFrame;
import javax.swing.JList;
import javax.swing.JPanel;
import javax.swing.JScrollPane;
import javax.swing.JSplitPane;
import javax.swing.border.EmptyBorder;
import javax.swing.border.TitledBorder;
import javax.swing.event.ListSelectionEvent;
import javax.swing.event.ListSelectionListener;

import org.icepdf.ri.common.ComponentKeyBinding;
import org.icepdf.ri.common.SwingController;
import org.icepdf.ri.common.SwingViewBuilder;

/**
 *
 * @author Kim Isle Cortez
 *
 */
public class LessonGui extends JFrame implements
        ListSelectionListener {
    private static final long serialVersionUID = -5316638372903106403L;
    private JPanel contentPane;
    private JList<String> list;
    private DefaultListModel<String> userFileListModel;
    private JPanel lessonPanel;
    private JSplitPane splitPane;
    private SwingController controller = new SwingController();
    private SwingViewBuilder factory =
            new SwingViewBuilder(controller);

    /**
     * Create the frame.
     */
    public LessonGui() {
        initializeModels();
        setTitle("NERD_Lessons");
        setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE);
        setBounds(100, 100, 900, 500);
        contentPane = new JPanel();
        contentPane.setBorder(new EmptyBorder(5, 5, 5, 5));
        setContentPane(contentPane);
        contentPane.setLayout(new GridLayout(1, 2, 0, 0));
        splitPane = new JSplitPane();
        splitPane.setBorder(new EmptyBorder(5, 5, 5, 5));
        contentPane.add(splitPane);
        JScrollPane scrollPane = new JScrollPane();
        scrollPane.setMinimumSize(new Dimension(200, 23));
        scrollPane.setBorder(new TitledBorder(null, "Files",
                TitledBorder.LEADING, TitledBorder.TOP, null, null));
        splitPane.setLeftComponent(scrollPane);
        list = new JList<String>(userFileListModel);
        list.addListSelectionListener(this);
        scrollPane.setViewportView(list);
        lessonPanel = factory.buildViewerPanel();
        splitPane.setRightComponent(lessonPanel);
    }

    private void initializeModels() {
        userFileListModel = new DefaultListModel<String>();
        File dir = new File("files");
        for (File file : dir.listFiles()) {
            try {
                if (Files.probeContentType(file.toPath())
                        .equalsIgnoreCase("application/pdf")) {
                    userFileListModel.addElement(file.getName());
                }
            } catch (IOException e) {
```

```
                e.printStackTrace();
            }
        }
    }

    @Override
    public void valueChanged(ListSelectionEvent e) {
        if (e.getSource() == list) {
            ComponentKeyBinding.install(controller, lessonPanel);
            controller.getDocumentViewController()
                    .setAnnotationCallback(
                            new org.icepdf.ri.common.MyAnnotationCallback(
                                    controller.getDocumentViewController()));
            controller.openDocument("files/" + list.getSelectedValue());
        }
        revalidate();
        repaint();
    }
}

/**
 * AboutGui.java
 * GUI for about
 */
package com.kimcortez.sp.gui;

import java.awt.Font;

import javax.swing.ImageIcon;
import javax.swing.JDialog;
import javax.swing.JLabel;
import javax.swing.SwingConstants;
import java.awt.Toolkit;

/**
 *
 * @author Kim Isle Cortez
 *
 */
public class AboutGui extends JDialog {
    private static final long serialVersionUID = -4660679921708957637L;

    public AboutGui() {
        setIconImage(Toolkit
                .getDefaultToolkit()
                .getImage(
                        AboutGui.class
                                .getResource("/com/kimcortez/sp/gui/icons/icon.png")));
        setBounds(100, 100, 450, 251);
        getContentPane().setLayout(null);
        JLabel label = new JLabel("");
        label.setIcon(new ImageIcon(AboutGui.class
                .getResource("/com/kimcortez/sp/gui/icons/icon.png")));
        label.setHorizontalAlignment(SwingConstants.LEFT);
        label.setBounds(10, 11, 90, 99);
        getContentPane().add(label);
        JLabel lblNerd = new JLabel("NERD");
        lblNerd.setFont(new Font("Tahoma", Font.BOLD, 60));
        lblNerd.setBounds(110, 11, 314, 73);
        getContentPane().add(lblNerd);
        JLabel lblANormalizationAnd =
                new JLabel("A Normalization and ERD Learning Tool");
        lblANormalizationAnd
                .setFont(new Font("Tahoma", Font.PLAIN, 15));
        lblANormalizationAnd.setBounds(110, 85, 314, 25);
        getContentPane().add(lblANormalizationAnd);
        JLabel lblDevelopedByKim =
                new JLabel("Developed by Kim Isle Cortez");
```

```
        lblDevelopedByKim.setBounds(10, 137, 414, 14);
        getContentPane().add(lblDevelopedByKim);
        JLabel lblBsComputerScience = new JLabel("BS_Computer_Science");
        lblBsComputerScience.setBounds(10, 162, 414, 14);
        getContentPane().add(lblBsComputerScience);
        JLabel lblAllRights = new JLabel("2014._All_Rights_Reserved.");
        lblAllRights.setBounds(10, 187, 414, 14);
        getContentPane().add(lblAllRights);
    }
}

/**
 * Splash.java
 * Splash screen for the application
 */
package com.kimcortez.sp.gui;

import java.awt.Component;

import javax.swing.ImageIcon;
import javax.swing.JFrame;
import javax.swing.JLabel;
import javax.swing.JPanel;
import javax.swing.SwingConstants;
import java.awt.Toolkit;

/**
 *
 * @author Kim Isle Cortez
 *
 */
public class Splash extends JFrame {
    private static final long serialVersionUID = 5842271160773827353L;
    private JPanel contentPane;

    public Splash() {
        setIconImage(Toolkit.getDefaultToolkit().getImage(
                Splash.class.getResource("/com/kimcortez/sp/"
                        + "gui/icons/icon.png")));
        setSize(500, 300);
        contentPane = new JPanel();
        contentPane.setBorder(null);
        setContentPane(contentPane);
        contentPane.setLayout(null);
        JLabel label =
                new JLabel(
                        "Developed_by_Kim_Isle_Cortez,_2014._All_rights_reserved.");
        label.setAlignmentX(Component.CENTER_ALIGNMENT);
        label.setHorizontalTextPosition(SwingConstants.CENTER);
        label.setHorizontalAlignment(SwingConstants.CENTER);
        label.setBounds(0, 275, 500, 14);
        contentPane.add(label);
        JLabel lblNewLabel = new JLabel("");
        lblNewLabel.setIcon(new ImageIcon(getClass().getResource(
                "icons/NERD.png")));
        lblNewLabel.setBounds(0, 0, 500, 300);
        contentPane.add(lblNewLabel);
        setUndecorated(true);
        setLocationRelativeTo(null);
        setVisible(true);
    }
}

/**
 * TutorGui.java
 * GUI for the monkey tutor
 */
package com.kimcortez.sp.gui;
```

```
import java.awt.Color;
import java.awt.Component;
import java.awt.Dimension;
import java.awt.GraphicsDevice;
import java.awt.GraphicsEnvironment;
import java.awt.Rectangle;

import javax.swing.ImageIcon;
import javax.swing.JFrame;
import javax.swing.JLabel;
import javax.swing.JTextArea;
import java.awt.Font;

/**
 *
 * @author Kim Isle Cortez
 *
 */
public class TutorGui extends JFrame {
    private static final long serialVersionUID = -8681229474795683943L;
    private JLabel label;
    private JTextArea textArea;

    public TutorGui() {
        setUndecorated(true);
        setSize(new Dimension(288, 340));
        setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE);
        setBackground(new Color(1.0f, 1.0f, 1.0f, 0f));
        getContentPane().setLayout(null);
        label = new JLabel("");
        label.setAlignmentX(Component.CENTER_ALIGNMENT);
        label.setBounds(0, 110, 190, 230);
        label.setIcon(new ImageIcon(getClass().getResource(
                "images/3.png")));
        getContentPane().add(label);
        textArea = new JTextArea();
        textArea.setFont(new Font("Monospaced", Font.PLAIN, 15));
        textArea
                .setText("Welcome to NERD! I'm Nerdy the monkey. Let me"
                        + " show you the step by step processing for this ERD.");
        textArea.setEditable(false);
        textArea.setWrapStyleWord(true);
        textArea.setLineWrap(true);
        textArea.setBounds(60, 0, 228, 121);
        getContentPane().add(textArea);
        GraphicsEnvironment ge =
                GraphicsEnvironment.getLocalGraphicsEnvironment();
        GraphicsDevice defaultScreen = ge.getDefaultScreenDevice();
        Rectangle rect =
                defaultScreen.getDefaultConfiguration().getBounds();
        int x = (int) rect.getMaxX() - getWidth();
        int y = (int) rect.getMaxY() - getHeight();
        setLocation(x, y - 45);
    }

    public void setIcon(ImageIcon icon) {
        label.setIcon(icon);
    }

    public void setText(String text) {
        textArea.setText(text);
    }
}

/**
 * Attribute.java
 * Represents attribute of an entity in an ERD
```

```
 */
package com.kimcortez.sp.erd.models;

import java.io.Serializable;

/**
 *
 * @author Kim Isle Cortez
 *
 */
public class Attribute implements Serializable {
    private static final long serialVersionUID = 3507614896119025595L;
    private String name;
    private boolean primaryKey;

    public Attribute(String name, boolean pk) {
        this.name = name;
        this.primaryKey = pk;
    }

    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }

    public boolean isPrimaryKey() {
        return primaryKey;
    }

    public void setPrimaryKey(boolean primaryKey) {
        this.primaryKey = primaryKey;
    }

    @Override
    public String toString() {
        return (primaryKey ? "unique " : "") + name;
    }

    @Override
    public int hashCode() {
        final int prime = 31;
        int result = 1;
        result =
                prime * result + ((name == null) ? 0 : name.hashCode());
        return result;
    }

    @Override
    public boolean equals(Object obj) {
        if (this == obj)
            return true;
        if (obj == null)
            return false;
        if (getClass() != obj.getClass())
            return false;
        Attribute other = (Attribute) obj;
        if (name == null) {
            if (other.name != null)
                return false;
        } else if (!name.equals(other.name))
            return false;
        return true;
    }
}
```

```
/**
 * Cardinality.java
 * Represents a cardinality in an ERD
 */
package com.kimcortez.sp.erd.models;

/**
 *
 * @author Kim Isle Cortez
 *
 */
public enum Cardinality {
    MANDATORY_ONE("11"), OPTIONAL_ONE("01"), MANDATORY_MANY("1N"),
    OPTIONAL_MANY("0N");
    private String code;

    private Cardinality(String code) {
        this.code = code;
    }

    public String getCode() {
        return code;
    }
}

/**
 * Entity.java
 * Represents an entity in an ERD
 */
package com.kimcortez.sp.erd.models;

import java.io.Serializable;
import java.util.ArrayList;
import java.util.HashSet;
import java.util.List;
import java.util.Set;

/**
 *
 * @author Kim Isle Cortez
 *
 */
public class Entity implements Serializable {
    private static final long serialVersionUID = -6202132456201057770L;
    private String name;
    private List<Attribute> attributes = new ArrayList<Attribute>();
    private Entity supertype;
    private Set<Entity> subtypes = new HashSet<Entity>();
    private String disjoint = "";
    private String total = "partial";

    public Entity(String name) {
        this.name = name;
    }

    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }

    public List<Attribute> getAttributes() {
        return attributes;
    }

    public void setAttributes(List<Attribute> attributes) {
```

```
    this.attributes = attributes;
}

public Entity getSupertype() {
    return supertype;
}

public void setSupertype(Entity supertype) {
    this.supertype = supertype;
}

public Set<Entity> getSubtypes() {
    return subtypes;
}

public void setSubtypes(Set<Entity> subtypes) {
    this.subtypes = subtypes;
}

public String getDisjoint() {
    return disjoint;
}

public void setDisjoint(String disjoint) {
    this.disjoint = disjoint;
}

public String getTotal() {
    return total;
}

public void setTotal(String total) {
    this.total = total;
}

public void addAttribute(Attribute attr) {
    boolean add = true;
    Attribute duplicate = null;
    for (Attribute a : attributes) {
        if (a.getName().equalsIgnoreCase(attr.getName())) {
            add = !a.isPrimaryKey();
            duplicate = a;
        }
    }
    if (add) {
        attributes.remove(duplicate);
        attributes.add(attr);
    }
    arrangeAttributes();
}

private void arrangeAttributes() {
    Attribute pk = null;
    List<Attribute> arranged = new ArrayList<Attribute>();
    for (Attribute a : attributes) {
        pk = a.isPrimaryKey() ? a : null;
    }
    if (pk != null) {
        arranged.add(pk);
        attributes.remove(pk);
        arranged.addAll(attributes);
        attributes = new ArrayList<Attribute>(arranged);
    }
}

public boolean addSubtype(Entity subtype) {
    return subtypes.add(subtype);
}
```

```
    @Override
    public String toString() {
        return "Entity [name=" + name + ", attributes=" + attributes
                + ", supertype=" + "" + ", subtypes=" + "" + "]";
    }

    @Override
    public int hashCode() {
        final int prime = 31;
        int result = 1;
        result =
                prime * result + ((name == null) ? 0 : name.hashCode());
        return result;
    }

    @Override
    public boolean equals(Object obj) {
        if (this == obj)
            return true;
        if (obj == null)
            return false;
        if (getClass() != obj.getClass())
            return false;
        Entity other = (Entity) obj;
        if (name == null) {
            if (other.name != null)
                return false;
        } else if (!name.equals(other.name))
            return false;
        return true;
    }
}

/**
 * ERD.java
 * Represents an ERD
 */
package com.kimcortez.sp.erd.models;

import java.io.Serializable;
import java.util.HashSet;
import java.util.Set;

/**
 *
 * @author Kim Isle Cortez
 *
 */
public class ERD implements Serializable {
    private static final long serialVersionUID = 1L;
    private Set<Entity> entities = new HashSet<Entity>();
    private Set<Relationship> relationships =
            new HashSet<Relationship>();

    public ERD(Set<Entity> entities, Set<Relationship> relationships) {
        this.entities = entities;
        this.relationships = relationships;
    }

    public Set<Entity> getEntities() {
        return entities;
    }

    public void setEntities(Set<Entity> entities) {
        this.entities = entities;
    }
```

```
    public Set<Relationship> getRelationships() {
        return relationships;
    }

    public void setRelationships(Set<Relationship> relationships) {
        this.relationships = relationships;
    }

    @Override
    public String toString() {
        return "ERD [entities=" + entities + ", relationships="
                + relationships + "]";
    }

    @Override
    public int hashCode() {
        final int prime = 31;
        int result = 1;
        result =
                prime * result
                        + ((entities == null) ? 0 : entities.hashCode());
        result =
                prime
                        * result
                        + ((relationships == null) ? 0 : relationships
                                .hashCode());
        return result;
    }

    @Override
    public boolean equals(Object obj) {
        if (this == obj)
            return true;
        if (obj == null)
            return false;
        if (getClass() != obj.getClass())
            return false;
        ERD other = (ERD) obj;
        if (entities == null) {
            if (other.entities != null)
                return false;
        } else if (!entities.equals(other.entities))
            return false;
        if (relationships == null) {
            if (other.relationships != null)
                return false;
        } else if (!relationships.equals(other.relationships))
            return false;
        return true;
    }
}

/**
 * Relationship.java
 * Represents relationship component in ERD
 */
package com.kimcortez.sp.erd.models;

import java.io.Serializable;

/**
 *
 * @author Kim Isle Cortez
 *
 */
public class Relationship implements Serializable {
    private static final long serialVersionUID = -1061604850739713851L;
    private String firstLabel, secondLabel;
```

```
private Entity firstEntity;
private Entity secondEntity;
private Cardinality firstEntityCardinality;
private Cardinality secondEntityCardinality;

public Relationship(Entity firstEntity, Entity secondEntity) {
    this.firstEntity = firstEntity;
    this.secondEntity = secondEntity;
}

public String getFirstLabel() {
    return firstLabel;
}

public void setFirstLabel(String firstLabel) {
    this.firstLabel = firstLabel;
}

public String getSecondLabel() {
    return secondLabel;
}

public void setSecondLabel(String secondLabel) {
    this.secondLabel = secondLabel;
}

public Entity getFirstEntity() {
    return firstEntity;
}

public void setFirstEntity(Entity firstEntity) {
    this.firstEntity = firstEntity;
}

public Entity getSecondEntity() {
    return secondEntity;
}

public void setSecondEntity(Entity secondEntity) {
    this.secondEntity = secondEntity;
}

public Cardinality getFirstEntityCardinality() {
    return firstEntityCardinality;
}

public void setFirstEntityCardinality(
        Cardinality firstEntityCardinality) {
    this.firstEntityCardinality = firstEntityCardinality;
}

public Cardinality getSecondEntityCardinality() {
    return secondEntityCardinality;
}

public void setSecondEntityCardinality(
        Cardinality secondEntityCardinality) {
    this.secondEntityCardinality = secondEntityCardinality;
}

@Override
public String toString() {
    return "Relationship [firstLabel=" + firstLabel
            + ", secondLabel=" + secondLabel + ", firstEntity="
            + firstEntity + ", secondEntity=" + secondEntity
            + ", firstEntityCardinality=" + firstEntityCardinality
            + ", secondEntityCardinality=" + secondEntityCardinality
            + "]";
```

```
    }

    @Override
    public int hashCode() {
        final int prime = 31;
        int result = 1;
        result =
                prime
                        * result
                        + ((firstEntity == null) ? 0 : firstEntity
                                .hashCode());
        result =
                prime
                        * result
                        + ((secondEntity == null) ? 0 : secondEntity
                                .hashCode());
        return result;
    }

    @Override
    public boolean equals(Object obj) {
        if (this == obj)
            return true;
        if (obj == null)
            return false;
        if (getClass() != obj.getClass())
            return false;
        Relationship other = (Relationship) obj;
        if (firstEntity == null) {
            if (other.firstEntity != null)
                return false;
        } else if (!firstEntity.equals(other.firstEntity))
            return false;
        if (secondEntity == null) {
            if (other.secondEntity != null)
                return false;
        } else if (!secondEntity.equals(other.secondEntity))
            return false;
        return true;
    }
}

/**
 * ERDUtils.java
 * Utilities for manipulating the ERD object
 */
package com.kimcortez.sp.erd.utils;

import java.awt.Dimension;
import java.awt.Image;
import java.io.File;
import java.io.FileInputStream;
import java.io.FileOutputStream;
import java.io.IOException;
import java.io.ObjectInputStream;
import java.io.ObjectOutputStream;
import java.util.ArrayList;
import java.util.Collections;
import java.util.HashSet;
import java.util.List;
import java.util.Set;

import javax.swing.ImageIcon;
import javax.swing.JFileChooser;
import javax.swing.JOptionPane;
import javax.swing.JPanel;
import javax.swing.filechooser.FileFilter;
import javax.swing.filechooser.FileNameExtensionFilter;
```

```
import org.graphstream.graph.Graph;
import org.graphstream.graph.Node;
import org.graphstream.graph.implementations.MultiGraph;
import org.graphstream.ui.spriteManager.Sprite;
import org.graphstream.ui.spriteManager.SpriteManager;
import org.graphstream.ui.swingViewer.View;
import org.graphstream.ui.swingViewer.Viewer;

import com.kimcortez.sp.erd.models.Attribute;
import com.kimcortez.sp.erd.models.Cardinality;
import com.kimcortez.sp.erd.models.ERD;
import com.kimcortez.sp.erd.models.Entity;
import com.kimcortez.sp.erd.models.Relationship;
import com.kimcortez.sp.gui.ScreenImage;

/**
 * @author Kim Isle Cortez
 */
public class ERDUtils {
    private ERD erd;
    private Graph graph;
    private SpriteManager spriteManager;
    private JFileChooser fileChooser;
    private View view;
    private int width, height;
    private List<Integer> id = new ArrayList<Integer>();

    public ERDUtils() {
        fileChooser = new JFileChooser();
        FileFilter fileFilter =
                new FileNameExtensionFilter(
                        "Entity Relationship Model (*.erd)", "erd");
        fileChooser.setFileFilter(fileFilter);
        fileChooser.addChoosableFileFilter(fileFilter);
        fileChooser.setAcceptAllFileFilterUsed(false);
        for (int i = 0; i < 1000000; i++) {
            id.add(new Integer(i));
        }
    }

    public ERDUtils(ERD erd) {
        this();
        this.erd = erd;
    }

    public ERD getErd() {
        return erd;
    }

    public void setErd(ERD erd) {
        this.erd = erd;
    }

    /**
     * Renders the ERD
     * @return the component that holds the ERD
     */
    public View render() {
        File dir = new File("temp");
        for (File file : dir.listFiles()) {
            file.delete();
        }
        graph = null;
        graph = new MultiGraph("ERD");
        graph.addAttribute("ui.antialias");
        graph.addAttribute("ui.quality");
        graph.addAttribute("ui.stylesheet", getStylesheet());
```

```
    graph.addAttribute("layout.gravity", 0.005);
    spriteManager = new SpriteManager(graph);
    populateGraph(graph);
    Viewer viewer =
            new Viewer(graph,
                    Viewer.ThreadingModel.GRAPH_IN_ANOTHER_THREAD);
    viewer.enableAutoLayout();
    view = viewer.addDefaultView(false);
    view.setMinimumSize(new Dimension((int) (width), (int) (height)));
    view.setPreferredSize(new Dimension((int) (1.2 * width),
            (int) (1.2 * height)));
    view.setMaximumSize(new Dimension(3 * width, 3 * height));
    return view;
}

/**
 * The stylesheet for graphstream
 * @return the stylesheet to use
 */
private String getStylesheet() {
    return "node {"
            + " fill-mode: image-scaled-ratio-max;"
            + " shape: box;"
            + " size-mode: fit;"
            + "}"
            + "edge {"
            + " size: 2px;"
            + "}"
            + "sprite.many {"
            + " shape: arrow; fill-mode:none; stroke-mode: " +
            "plain; stroke-width: 2px;"
            + "}"
            + "sprite.zero {"
            + " shape: circle; fill-color: white; stroke-mode: " +
            "plain; stroke-width: 2px; z-index:5;"
            + "}" + "sprite.one {" + "shape: box; z-index: 4;"
            + "} graph {padding:50px;}";
}

/**
 * Populates the given graph with nodes and edges
 * @param graph
 */
private void populateGraph(Graph graph) {
    Collections.shuffle(id);
    Set<Entity> entities = new HashSet<>(erd.getEntities());
    Set<Relationship> relations =
            new HashSet<>(erd.getRelationships());
    width = height = 0;
    JPanel panel = null;
    for (Entity entity : entities) {
        graph.addNode(entity.getName());
        Node n = graph.getNode(entity.getName());
        panel = new NodeImage(entity);
        try {
            ScreenImage.writeImage(ScreenImage.createImage(panel),
                    "temp/" + entity.getName() + id.get(0) + ".png");
        } catch (IOException e) {
            e.printStackTrace();
        }
        Image image =
                new ImageIcon("temp/" + entity.getName() + id.get(0)
                        + ".png").getImage();
        n.changeAttribute(
                "ui.style",
                "fill-image: url('temp/" + entity.getName()
                        + id.get(0) + ".png'); size:"
                        + image.getWidth(null) + "px, "
```

```
                    + image.getHeight(null) + "px;");
            width += image.getWidth(null);
            height += image.getHeight(null);
        }
        for (Entity entity : entities) {
            if (entity.getSubtypes().size() > 0) {
                graph.addEdge(entity.getName() + "_toSubtypes",
                        graph.getNode(entity.getName()),
                        addSubtypes(entity), true);
            }
        }
        for (Relationship relation : relations) {
            String source = null, target = null;
            for (Entity entity : entities) {
                if (entity.getName().equals(
                        relation.getFirstEntity().getName())) {
                    source = entity.getName();
                }
                if (entity.getName().equals(
                        relation.getSecondEntity().getName())) {
                    target = entity.getName();
                }
            }
            String edgeId = source + "-" + target;
            graph.addEdge(edgeId, source, target);
            manageCardinalitySprites(edgeId, relation);
            width += 50;
            height += 50;
        }
        entities = null;
        relations = null;
    }

    /**
     * Adds cardinality sprites on graph edges
     * @param edgeId
     * @param relation
     */
    private void manageCardinalitySprites(String edgeId,
            Relationship relation) {
        String c1 = "";
        String c2 = "";
        try {
            c1 = relation.getFirstEntityCardinality().getCode();
            c2 = relation.getSecondEntityCardinality().getCode();
        } catch (NullPointerException e) {
            JOptionPane.showMessageDialog(null,
                    "Input_error!_Missing_cardinalities_between_entities._["
                            + relation.getFirstEntity().getName() + "-"
                            + relation.getSecondEntity().getName() + "]",
                    "Input_Error", JOptionPane.ERROR_MESSAGE);
            return;
        }
        Sprite s1 = spriteManager.addSprite(relation.toString() + 1);
        Sprite s2 = spriteManager.addSprite(relation.toString() + 2);
        Sprite s3 = spriteManager.addSprite(relation.toString() + 3);
        Sprite s4 = spriteManager.addSprite(relation.toString() + 4);
        s1.attachToEdge(edgeId);
        s2.attachToEdge(edgeId);
        s3.attachToEdge(edgeId);
        s4.attachToEdge(edgeId);
        s1.setPosition(.23);
        s2.setPosition(.77);
        s3.setPosition(.28);
        s4.setPosition(.72);
        s1.addAttribute("ui.style", "sprite-orientation:to;");
        s2.addAttribute("ui.style", "sprite-orientation:from;");
        s3.addAttribute("ui.style", "sprite-orientation:to;");
```

```
    s4.addAttribute("ui.style", "sprite-orientation:from;");
    // s1,s3
    switch (c1) {
    case "01":
        s1.addAttribute("ui.class", "one");
        s1.addAttribute("ui.style", "size: 2px,12px;");
        s3.addAttribute("ui.class", "zero");
        break;
    case "11":
        s1.addAttribute("ui.class", "one");
        s1.addAttribute("ui.style", "size: 2px,12px;");
        s3.addAttribute("ui.class", "one");
        s3.addAttribute("ui.style", "size: 2px,12px;");
        break;
    case "0N":
        s1.addAttribute("ui.class", "many");
        s3.addAttribute("ui.class", "zero");
        break;
    case "1N":
        s1.addAttribute("ui.class", "many");
        s3.addAttribute("ui.class", "one");
        s3.addAttribute("ui.style", "size: 2px,12px;");
        break;
    }
    // s2,s4
    switch (c2) {
    case "01":
        s2.addAttribute("ui.class", "one");
        s2.addAttribute("ui.style", "size: 2px,12px;");
        s4.addAttribute("ui.class", "zero");
        break;
    case "11":
        s2.addAttribute("ui.class", "one");
        s2.addAttribute("ui.style", "size: 2px,12px;");
        s4.addAttribute("ui.class", "one");
        s4.addAttribute("ui.style", "size: 2px,12px;");
        break;
    case "0N":
        s2.addAttribute("ui.class", "many");
        s4.addAttribute("ui.class", "zero");
        break;
    case "1N":
        s2.addAttribute("ui.class", "many");
        s4.addAttribute("ui.class", "one");
        s4.addAttribute("ui.style", "size: 2px,12px;");
        break;
    }
}

/**
 * Adds subtypes to entity
 * @param entity
 * @return node with entity and subtype connected
 */
private Node addSubtypes(Entity entity) {
    Node n = graph.addNode(entity.getName() + "_subtypes");
    String path =
            "src/com/kimcortez/sp/erd/models/images/"
                    + getSubtypeImage(entity) + ".png";
    Image image = new ImageIcon(path).getImage();
    n.addAttribute(
            "ui.style",
            "fill-image: url('" + path + "'); size:"
                    + image.getWidth(null) + "px, "
                    + image.getHeight(null) + "px;");
    width += image.getWidth(null);
    height += image.getHeight(null);
    for (Node node : graph.getNodeSet()) {
```

```
        for (Entity e : entity.getSubtypes()) {
            if (node.getId().equals(e.getName())) {
                graph.addEdge(entity.getName() + "_" + e.getName(), n,
                        node, true);
            }
        }
    }
    return n;
}

/**
 * Saves the ERD into (*.erd)
 */
public void save() {
    if (fileChooser.showSaveDialog(null) == JFileChooser.APPROVE_OPTION) {
        File file = fileChooser.getSelectedFile();
        ObjectOutputStream oos;
        try {
            oos =
                    new ObjectOutputStream(new FileOutputStream(file
                            + ".erd"));
            oos.writeObject(erd);
            oos.close();
        } catch (IOException e) {
            e.printStackTrace();
        }
        JOptionPane.showMessageDialog(null, "ERD_file_saved!",
                "Save", JOptionPane.INFORMATION_MESSAGE);
    }
}

/**
 * Opens an ERD
 * @return the ERD file opened
 */
public ERD open() {
    ERD erd = null;
    if (fileChooser.showOpenDialog(null) == JFileChooser.APPROVE_OPTION) {
        File newFile = fileChooser.getSelectedFile();
        ObjectInputStream ois;
        try {
            ois = new ObjectInputStream(new FileInputStream(newFile));
            erd = (ERD) ois.readObject();
            ois.close();
        } catch (IOException | ClassNotFoundException e) {
            e.printStackTrace();
        }
    }
    return erd;
}

/**
 * Saves ERD as an image (*.png)
 */
public void saveImage() {
    JFileChooser imageSaver = new JFileChooser();
    FileFilter fileFilter =
            new FileNameExtensionFilter("PNG (*.png)", "png");
    imageSaver.setFileFilter(fileFilter);
    imageSaver.addChoosableFileFilter(fileFilter);
    imageSaver.setAcceptAllFileFilterUsed(false);
    if (imageSaver.showSaveDialog(null) == JFileChooser.APPROVE_OPTION) {
        File file = imageSaver.getSelectedFile();
        try {
            ScreenImage.writeImage(ScreenImage.createImage(view),
                    file + ".png");
        } catch (IOException e) {
            e.printStackTrace();
```

```
        }
        JOptionPane.showMessageDialog(null, "ERD_image_saved!",
                "Save", JOptionPane.INFORMATION_MESSAGE);
    }
}

/**
 * Parses the ERD to generate strings describing it
 * @return
 */
public List<String> parseStrings() {
    List<String> sentences = null;
    if (erd != null) {
        sentences = new ArrayList<String>();
        Set<Entity> entities = erd.getEntities();
        Set<Relationship> relations = erd.getRelationships();
        for (Relationship r : relations) {
            List<String> strings = translateToSentence(r);
            if (strings == null) {
                return null;
            }
            sentences.addAll(strings);
        }
        for (Entity e : entities) {
            if (e.getAttributes().size() > 0) {
                sentences.addAll(translateToSentence(e,
                        e.getAttributes()));
            }
            if (e.getSubtypes().size() > 0) {
                sentences.add(translateToSentence(e, e.getSubtypes()));
            }
        }
    }
    return sentences;
}

/**
 * Clears the erd, graph and view
 */
public void clear() {
    erd = null;
    graph = null;
    view = null;
}

private String getSubtypeImage(Entity entity) {
    return entity.getTotal() + entity.getDisjoint();
}

/**
 * Translates cardinality to string
 * @param cardinality
 * @return
 */
private String translateCardinality(Cardinality cardinality) {
    switch (cardinality.getCode()) {
    case "11":
        return "only_one";
    case "01":
        return "at_most_one";
    case "1N":
        return "at_least_one";
    case "0N":
        return "zero_or_many";
    default:
        return "zero_or_many";
    }
}
```

```
/**
 * Translates relationship to sentence
 * @param r
 * @return
 */
private List<String> translateToSentence(Relationship r) {
    List<String> translated = new ArrayList<String>();
    String s1 = "", s2 = "";
    Entity e1 = r.getFirstEntity(), e2 = r.getSecondEntity();
    try {
        Cardinality c1 = r.getFirstEntityCardinality(), c2 =
                r.getSecondEntityCardinality();
        String rln1 = r.getFirstLabel(), rln2 = r.getSecondLabel();
        String c1s = c1.getCode().charAt(0) == '1' ? "must" : "may";
        String c2s = c2.getCode().charAt(0) == '1' ? "must" : "may";
        s1 =
                e1.getName() + "␣" + c1s + "␣" + rln1 + "␣"
                        + translateCardinality(c2) + "␣" + e2.getName();
        s2 =
                e2.getName() + "␣" + c2s + "␣" + rln2 + "␣"
                        + translateCardinality(c1) + "␣" + e1.getName();
    } catch (NullPointerException e) {
        return null;
    }
    translated.add(s1);
    translated.add(s2);
    return translated;
}

/**
 * Translates entity with subtype to sentence
 * @param entity
 * @param subtypes
 * @return
 */
private String translateToSentence(Entity entity,
        Set<Entity> subtypes) {
    String sentence =
            entity.getName()
                    + (entity.getDisjoint().equals("disjoint")
                            ? "␣can␣be␣either␣a␣" : "␣can␣be␣a␣");
    List<String> subtypeList = new ArrayList<String>();
    for (Entity e : subtypes) {
        subtypeList.add(e.getName());
    }
    sentence += getSentence(subtypeList, "or");
    sentence += (entity.getTotal().length() > 0) ? "␣only" : "";
    return sentence;
}

/**
 * Translates entity with attributes to sentence
 * @param entity
 * @param attributes
 * @return
 */
private List<String> translateToSentence(Entity entity,
        List<Attribute> attributes) {
    List<String> sentences = new ArrayList<String>();
    List<String> unique = new ArrayList<String>();
    List<String> notUnique = new ArrayList<String>();
    String sentence1 = entity.getName() + "␣has␣a␣unique␣";
    String sentence2 = entity.getName() + "␣has␣a␣";
    for (Attribute attr : attributes) {
        if (attr.toString().contains("unique")) {
            unique.add(attr.getName());
        } else {
```

```
                notUnique.add(attr.getName());
            }
        }
        sentence1 += getSentence(unique, "and");
        sentence2 += getSentence(notUnique, "and");
        sentences.add(sentence1);
        sentences.add(sentence2);
        return sentences;
    }

    private String getSentence(List<String> list, String conj) {
        StringBuilder builder = new StringBuilder();
        builder.append(list.remove(0));
        for (String s : list) {
            if (list.get(list.size() - 1).equalsIgnoreCase(s)) {
                builder.append(" " + conj + " a ");
            } else {
                builder.append(", a ");
            }
            builder.append(s);
        }
        return builder.toString();
    }
}

/**
 * NodeImage.java
 * Used so that the node in graphstream looks like an entity
 */
package com.kimcortez.sp.erd.utils;

import java.awt.BorderLayout;
import java.awt.Color;
import java.awt.GridLayout;

import javax.swing.JLabel;
import javax.swing.JPanel;
import javax.swing.border.CompoundBorder;
import javax.swing.border.LineBorder;

import com.kimcortez.sp.erd.models.Attribute;
import com.kimcortez.sp.erd.models.Entity;

/**
 *
 * @author Kim Isle Cortez
 *
 */
public class NodeImage extends JPanel {
    private static final long serialVersionUID = -1644074004999342999L;

    public NodeImage(Entity entity) {
        setBorder(new LineBorder(new Color(0, 0, 0)));
        setBackground(Color.WHITE);
        setLayout(new BorderLayout(0, 0));
        JPanel panel = new JPanel();
        panel.setBackground(Color.WHITE);
        panel.setBorder(new LineBorder(new Color(0, 0, 0)));
        add(panel, BorderLayout.NORTH);
        JLabel lblEntityName =
                new JLabel(entity.getName().toUpperCase());
        panel.add(lblEntityName);
        JPanel panel_2 = new JPanel();
        panel_2.setBackground(Color.WHITE);
        panel_2.setBorder(new CompoundBorder(new LineBorder(new Color(
                0, 0, 0)), new LineBorder(new Color(255, 255, 255), 7)));
        add(panel_2, BorderLayout.WEST);
        panel_2.setLayout(new GridLayout(entity.getAttributes().size(),
```

```
                1, 0, 0));
        JPanel panel_1 = new JPanel();
        panel_1.setBackground(Color.WHITE);
        panel_1.setBorder(new CompoundBorder(new LineBorder(new Color(
                0, 0, 0)), new LineBorder(new Color(255, 255, 255), 7)));
        add(panel_1, BorderLayout.CENTER);
        panel_1.setLayout(new GridLayout(0, 1, 0, 0));
        for (Attribute attr : entity.getAttributes()) {
            panel_1.add(new JLabel(attr.getName()));
            if (attr.isPrimaryKey()) {
                panel_2.add(new JLabel("PK"));
            } else {
                panel_2.add(new JLabel("␣"));
            }
        }
    }
}

/**
 * ComponentIdentifier.java
 * Class that identifies components from tuples
 */
package com.kimcortez.sp.parser;

import java.util.HashSet;
import java.util.Set;

import com.kimcortez.sp.erd.models.Attribute;
import com.kimcortez.sp.erd.models.Cardinality;
import com.kimcortez.sp.erd.models.Entity;
import com.kimcortez.sp.erd.models.Relationship;
import com.kimcortez.sp.parser.models.Tuple;
import com.kimcortez.sp.utils.Dictionary;
import com.kimcortez.sp.utils.EnglishNoun;

/**
 *
 * @author Kim Isle Cortez
 *
 */
public class ComponentIdentifier {
    private Set<Tuple> entityTuples = new HashSet<Tuple>();
    private Set<Tuple> attributeTuples = new HashSet<Tuple>();
    private Set<Tuple> cardinalityTuples = new HashSet<Tuple>();
    private Set<Tuple> subtypeTuples = new HashSet<Tuple>();
    private Set<Entity> entities = new HashSet<Entity>();
    private Set<Relationship> relations = new HashSet<Relationship>();
    private Dictionary dictionary;

    public ComponentIdentifier(Set<Tuple> entityTuples,
            Set<Tuple> attributeTuples, Set<Tuple> cardinalityTuples,
            Set<Tuple> subtypeTuples) {
        this.entityTuples = entityTuples;
        this.attributeTuples = attributeTuples;
        this.cardinalityTuples = cardinalityTuples;
        this.subtypeTuples = subtypeTuples;
    }

    public Set<Tuple> getEntityTuples() {
        return entityTuples;
    }

    public void setEntityTuples(Set<Tuple> entityTuples) {
        this.entityTuples = entityTuples;
    }

    public Set<Tuple> getAttributeTuples() {
        return attributeTuples;
```

```
}

public void setAttributeTuples(Set<Tuple> attributeTuples) {
    this.attributeTuples = attributeTuples;
}

public Set<Tuple> getCardinalityTuples() {
    return cardinalityTuples;
}

public void setCardinalityTuples(Set<Tuple> cardinalityTuples) {
    this.cardinalityTuples = cardinalityTuples;
}

public Set<Entity> getEntities() {
    return entities;
}

public void setEntities(Set<Entity> entities) {
    this.entities = entities;
}

public Set<Relationship> getRelations() {
    return relations;
}

public void setRelations(Set<Relationship> relations) {
    this.relations = relations;
}

public Dictionary getDictionary() {
    return dictionary;
}

public void setDictionary(Dictionary dictionary) {
    this.dictionary = dictionary;
}

/**
 * Stores component into entity and relationship sets
 */
public void identify() {
    // adding entities from normal entity relationship tuples
    for (Tuple t : entityTuples) {
        Entity e1 = getEntity(getSingular(t.getSecondElt()));
        Entity e2 = getEntity(getSingular(t.getThirdElt()));
        addNewRelation(new Relationship(e1, e2), t.getFirstElt());
    }
    // adding subtypes/supertypes from subtype tuples
    for (Tuple t : subtypeTuples) {
        Entity e1 = getEntity(getSingular(t.getSecondElt()));
        Entity e2 = getEntity(getSingular(t.getThirdElt()));
        e1.setDisjoint(t.getFirstElt().contains("disjoint")
                ? "disjoint" : e1.getDisjoint());
        e1.setDisjoint(t.getFirstElt().contains("overlap")
                ? "overlap" : e1.getDisjoint());
        e1.setTotal(t.getFirstElt().contains("only") ? "total" : e1
                .getTotal());
        e1.addSubtype(e2);
        e2.setSupertype(e1);
    }
    // adding cardinalities to relationships
    for (Tuple t : cardinalityTuples) {
        for (Relationship r : relations) {
            if (r.getFirstEntity().getName()
                    .equals(getSingular(t.getSecondElt()))
                    && r.getSecondEntity().getName()
                            .equals(getSingular(t.getThirdElt()))) {
```

```
                r.setSecondEntityCardinality(getCardinality(t
                        .getFirstElt()));
            } else if (r.getSecondEntity().getName()
                    .equals(getSingular(t.getSecondElt()))
                    && r.getFirstEntity().getName()
                            .equals(getSingular(t.getThirdElt()))) {
                r.setFirstEntityCardinality(getCardinality(t
                        .getFirstElt()));
            }
        }
    }
    // adding attributes from attribute tuples
    for (Tuple t : attributeTuples) {
        Entity e = getEntity(getSingular(t.getSecondElt()));
        String attr = t.getThirdElt();
        boolean pk =
                "unique".equalsIgnoreCase(t.getFirstElt())
                        || "distinct".equalsIgnoreCase(t.getFirstElt());
        e.addAttribute(new Attribute(attr, pk));
    }
}

private String getSingular(String word) {
    String firstWord = "";
    if (word.contains("_")) {
        firstWord = word.substring(0, word.indexOf('_') + 1);
        word = word.substring(word.indexOf('_') + 1);
    }
    return firstWord + EnglishNoun.singularOf(word);
}

private Entity getEntity(String name) {
    Entity entity = null;
    for (Entity e : entities) {
        if (e.getName().equals(getSingular(name))) {
            entity = e;
            break;
        }
    }
    if (entity == null) {
        entity = new Entity(getSingular(name));
        entities.add(entity);
    }
    return entity;
}

private Cardinality getCardinality(String cardinality) {
    switch (cardinality) {
    case "at_most_one":
    case "zero_or_one":
        return Cardinality.OPTIONAL_ONE;
    case "only_one":
        return Cardinality.MANDATORY_ONE;
    case "at_least_one":
    case "one_or_many":
        return Cardinality.MANDATORY_MANY;
    case "zero_or_many":
        return Cardinality.OPTIONAL_MANY;
    default:
        return Cardinality.OPTIONAL_MANY;
    }
}

private void addNewRelation(Relationship r, String label) {
    if (!relations.contains(new Relationship(r.getSecondEntity(), r
            .getFirstEntity()))) {
        r.setFirstLabel(label);
        relations.add(r);
```

```
        } else {
            for (Relationship relation : relations) {
                if (relation.equals(new Relationship(r.getSecondEntity(),
                        r.getFirstEntity()))) {
                    relation.setSecondLabel(label);
                }
            }
        }
    }
}

/**
 * ConflictChecker.java
 * Class for checking requirement conflicts
 */
package com.kimcortez.sp.parser;

import java.util.ArrayList;
import java.util.HashMap;
import java.util.List;
import java.util.Map;
import java.util.Map.Entry;
import java.util.Set;

import com.kimcortez.sp.parser.models.CustomError;
import com.kimcortez.sp.parser.models.Tuple;

import edu.stanford.nlp.trees.TypedDependency;

/**
 *
 * @author Kim Isle Cortez
 *
 */
public class ConflictChecker {
    private static final String cardinalityErr =
            "Conflicting cardinality constraints for entity [";
    private static final String subtypeErr =
            "Conflicting entity subtype constraints";
    private static final String entityErr =
            "The component is already identified as an entity [";
    private static final String attrErr =
            "The component is already identified as an attribute [";

    /**
     * Checks for subtype conflicts
     * @param subtypeTuples
     * @param attributeTuples
     * @param selected
     * @param sentence
     * @return
     */
    public static CustomError getSubtypeConflict(
            Set<Tuple> subtypeTuples, Set<Tuple> attributeTuples,
            List<TypedDependency> selected, String sentence) {
        Map<String, List<String>> totalMap =
                generateSubtypeMap(subtypeTuples, "only");
        Map<String, List<String>> disjointMap =
                generateSubtypeMap(subtypeTuples, "disjoint");
        Map<String, List<String>> overlapMap =
                generateSubtypeMap(subtypeTuples, "overlap");
        Map<String, List<String>> attributeMap =
                generateHashMap(attributeTuples);
        String relation =
                sentence.contains("can be either ") ? "can_be_disjoint"
                        : "can_be_overlap";
        String relation2 =
                sentence.contains("only.") ? relation + " only"
```

```
                : relation;
TypedDependency nsubj = null;
TypedDependency conj = null;
String supertype = "";
for (TypedDependency typedD : selected) {
    if (typedD.reln().toString().contains("nsubj")) {
        nsubj = typedD;
        supertype = nsubj.dep().nodeString();
    }
    if (typedD.reln().toString().contains("conj_")) {
        conj = typedD;
    }
    if (nsubj != null && conj != null) {
        String w1 = conj.gov().nodeString();
        String w2 = conj.dep().nodeString();
        for (Entry<String, List<String>> e : totalMap.entrySet()) {
            String key = (String) e.getKey();
            List<String> set = (List<String>) e.getValue();
            if (supertype.equalsIgnoreCase(key)) {
                if (!set.contains(w1) || !set.contains(w2)) {
                    return new CustomError(sentence, subtypeErr);
                }
            }
        }
        for (Entry<String, List<String>> e : disjointMap
                .entrySet()) {
            String key = (String) e.getKey();
            List<String> set = (List<String>) e.getValue();
            if (supertype.equalsIgnoreCase(key)) {
                if ((set.contains(w1) && set.contains(w2))
                        && relation2.contains("overlap")) {
                    return new CustomError(sentence, subtypeErr);
                }
            }
        }
        for (Entry<String, List<String>> e : overlapMap
                .entrySet()) {
            String key = (String) e.getKey();
            List<String> set = (List<String>) e.getValue();
            if (supertype.equalsIgnoreCase(key)) {
                if ((set.contains(w1) && set.contains(w2))
                        && relation2.contains("disjoint")) {
                    return new CustomError(sentence, subtypeErr);
                }
            }
        }
    }
    if (nsubj != null && selected.size() == 1) {
        String w1 = nsubj.gov().nodeString();
        for (Entry<String, List<String>> e : totalMap.entrySet()) {
            String key = (String) e.getKey();
            List<String> set = (List<String>) e.getValue();
            if (supertype.equalsIgnoreCase(key)) {
                if (!set.contains(w1)) {
                    return new CustomError(sentence, subtypeErr);
                }
            }
        }
        for (Entry<String, List<String>> e : disjointMap
                .entrySet()) {
            String key = (String) e.getKey();
            List<String> set = (List<String>) e.getValue();
            if (supertype.equalsIgnoreCase(key)) {
                if (set.contains(w1)
                        && relation.contains("overlap")) {
                    return new CustomError(sentence, subtypeErr);
                }
                if (relation2.contains("only") && set.size() > 1) {
```

```
                    return new CustomError(sentence, subtypeErr);
                }
            }
        }
        for (Entry<String, List<String>> e : overlapMap
                .entrySet()) {
            String key = (String) e.getKey();
            List<String> set = (List<String>) e.getValue();
            if (supertype.equalsIgnoreCase(key)) {
                if (set.contains(w1)
                        && relation.contains("disjoint")) {
                    return new CustomError(sentence, subtypeErr);
                }
                if (relation2.contains("only") && set.size() > 1) {
                    return new CustomError(sentence, subtypeErr);
                }
            }
        }
    }
}
nsubj = null;
conj = null;
supertype = "";
for (TypedDependency typedD : selected) {
    if (typedD.reln().toString().contains("nsubj")) {
        nsubj = typedD;
        supertype = nsubj.dep().nodeString();
    }
    if (typedD.reln().toString().contains("conj_")) {
        conj = typedD;
    }
    if (nsubj != null && conj != null) {
        String w1 = conj.gov().nodeString();
        String w2 = conj.dep().nodeString();
        for (Entry<String, List<String>> e : attributeMap
                .entrySet()) {
            List<String> set = (List<String>) e.getValue();
            if (set.contains(w1)) {
                return new CustomError(sentence, attrErr + w1 + "]");
            }
            if (set.contains(w2)) {
                return new CustomError(sentence, attrErr + w2 + "]");
            }
        }
    }
    if (nsubj != null && selected.size() == 1) {
        String w1 = nsubj.gov().nodeString();
        for (Entry<String, List<String>> e : attributeMap
                .entrySet()) {
            List<String> set = (List<String>) e.getValue();
            if (set.contains(w1)) {
                return new CustomError(sentence, attrErr + w1 + "]");
            }
        }
    }
}
return null;
}

/**
 * Check for cardinality conflicts
 * @param cardinalityTuples
 * @param selected
 * @param sentence
 * @return
 */
public static CustomError getCardinalityConflict(
        Set<Tuple> cardinalityTuples,
```

```
        List<TypedDependency> selected, String sentence) {
    Map<String, List<String>> entityMap =
            new HashMap<String, List<String>>();
    for (Tuple t : cardinalityTuples) {
        List<String> currentSet = entityMap.get(t.getSecondElt());
        if (currentSet == null) {
            entityMap.put(t.getSecondElt(), new ArrayList<String>());
            currentSet = entityMap.get(t.getSecondElt());
        }
        currentSet.add(t.getFirstElt() + "␣" + t.getThirdElt());
    }
    for (Entry<String, List<String>> e : entityMap.entrySet()) {
        String key = (String) e.getKey();
        List<String> set = (List<String>) e.getValue();
        TypedDependency nsubj = null;
        TypedDependency dobj = null;
        TypedDependency amod = null;
        for (TypedDependency typedD : selected) {
            if (typedD.reln().toString().contains("nsubj")) {
                nsubj = typedD;
            }
            if ("dobj".equals(typedD.reln().toString())
                    || typedD.reln().toString().contains("prep")) {
                dobj = typedD;
            }
            if ("amod".equals(typedD.reln().toString())) {
                amod = typedD;
            }
            if (nsubj != null && dobj != null && amod != null) {
                String w1 = nsubj.dep().nodeString();
                String w3 =
                        dobj.dep().nodeString()
                                .equals(amod.gov().nodeString()) ? amod
                                .gov().nodeString() : "";
                String w4 = amod.dep().nodeString();
                if (key.equalsIgnoreCase(w1)) {
                    for (String cardinality : set) {
                        String[] parts = cardinality.split("␣");
                        if (!parts[0].equalsIgnoreCase(w4)
                                && parts[1].equalsIgnoreCase(w3)) {
                            return new CustomError(sentence,
                                    cardinalityErr + w3 + "]");
                        }
                    }
                }
            }
            if (nsubj != null && dobj != null && amod == null) {
                String w1 = dobj.gov().nodeString();
                String w2 = nsubj.dep().nodeString();
                String w3 = dobj.dep().nodeString();
                if (key.equalsIgnoreCase(w2)) {
                    for (String cardinality : set) {
                        String[] parts = cardinality.split("␣");
                        if (!parts[0].equalsIgnoreCase(w1)
                                && parts[1].equalsIgnoreCase(w3)) {
                            return new CustomError(sentence,
                                    cardinalityErr + w3 + "]");
                        }
                    }
                }
            }
        }
    }
    return null;
}

/**
 * Check for entity conflicts
```

```
 * @param entityTuples
 * @param attributeTuples
 * @param subtypeTuples
 * @param cardinalityTuples
 * @param selected
 * @param sentence
 * @return
 */
public static CustomError getEntityConflict(
        Set<Tuple> entityTuples, Set<Tuple> attributeTuples,
        Set<Tuple> subtypeTuples, Set<Tuple> cardinalityTuples,
        List<TypedDependency> selected, String sentence) {
    Map<String, List<String>> attributeMap =
            generateHashMap(attributeTuples);
    Map<String, List<String>> entityMap =
            generateHashMap(entityTuples);
    Map<String, List<String>> subtypeMap =
            generateHashMap(subtypeTuples);
    Map<String, List<String>> cardinalityMap =
            generateHashMap(cardinalityTuples);
    TypedDependency nsubj = null;
    TypedDependency conj = null;
    TypedDependency dobj = null;
    for (TypedDependency typedD : selected) {
        if (typedD.reln().toString().contains("nsubj")) {
            nsubj = typedD;
        }
        if (typedD.reln().toString().contains("conj_")) {
            conj = typedD;
        }
        if ("dobj".equals(typedD.reln().toString())) {
            dobj = typedD;
        }
        if (nsubj != null && dobj != null) {
            String w3 = dobj.dep().nodeString();
            String w1 = nsubj.dep().nodeString();
            for (Entry<String, List<String>> e : entityMap.entrySet()) {
                String key = (String) e.getKey();
                List<String> set = (List<String>) e.getValue();
                if (key.equalsIgnoreCase(w3) || set.contains(w3)) {
                    return new CustomError(sentence, entityErr + w3
                            + "]");
                }
            }
            for (Entry<String, List<String>> e : subtypeMap
                    .entrySet()) {
                String key = (String) e.getKey();
                List<String> set = (List<String>) e.getValue();
                if (key.equalsIgnoreCase(w3) || set.contains(w3)) {
                    return new CustomError(sentence, entityErr + w3
                            + "]");
                }
            }
            for (Entry<String, List<String>> e : cardinalityMap
                    .entrySet()) {
                String key = (String) e.getKey();
                List<String> set = (List<String>) e.getValue();
                if (key.equalsIgnoreCase(w3) || set.contains(w3)) {
                    return new CustomError(sentence, entityErr + w3
                            + "]");
                }
            }
            for (Entry<String, List<String>> e : attributeMap
                    .entrySet()) {
                String key = (String) e.getKey();
                List<String> set = (List<String>) e.getValue();
                if (set.contains(w1)) {
                    return new CustomError(sentence, attrErr + w1 + "]");
```

```
            }
            if (key.equalsIgnoreCase(w3)) {
                return new CustomError(sentence, entityErr + w3
                        + "]");
            }
        }
    }
    if (nsubj != null && conj != null) {
        String w1 = nsubj.dep().nodeString();
        String w3 = conj.dep().nodeString();
        String w4 = conj.gov().nodeString();
        for (Entry<String, List<String>> e : entityMap.entrySet()) {
            String key = (String) e.getKey();
            List<String> set = (List<String>) e.getValue();
            if (key.equals(w3) || set.contains(w3)) {
                return new CustomError(sentence, entityErr + w3
                        + "]");
            }
            if (key.equals(w4) || set.contains(w4)) {
                return new CustomError(sentence, entityErr + w4
                        + "]");
            }
        }
        for (Entry<String, List<String>> e : subtypeMap
                .entrySet()) {
            String key = (String) e.getKey();
            List<String> set = (List<String>) e.getValue();
            if (key.equals(w3) || set.contains(w3)) {
                return new CustomError(sentence, entityErr + w3
                        + "]");
            }
            if (key.equals(w4) || set.contains(w4)) {
                return new CustomError(sentence, entityErr + w4
                        + "]");
            }
        }
        for (Entry<String, List<String>> e : cardinalityMap
                .entrySet()) {
            String key = (String) e.getKey();
            List<String> set = (List<String>) e.getValue();
            if (key.equals(w3) || set.contains(w3)) {
                return new CustomError(sentence, entityErr + w3
                        + "]");
            }
            if (key.equals(w4) || set.contains(w4)) {
                return new CustomError(sentence, entityErr + w4
                        + "]");
            }
        }
        for (Entry<String, List<String>> e : attributeMap
                .entrySet()) {
            String key = (String) e.getKey();
            List<String> set = (List<String>) e.getValue();
            if (set.contains(w1)) {
                return new CustomError(sentence, attrErr + w1 + "]");
            }
            if (key.equalsIgnoreCase(w3)) {
                return new CustomError(sentence, entityErr + w3
                        + "]");
            }
            if (key.equalsIgnoreCase(w4)) {
                return new CustomError(sentence, entityErr + w4
                        + "]");
            }
        }
    }
}
return null;
```

```
}

/**
 * Check for attribute conflicts
 * @param attributeTuples
 * @param selected
 * @param sentence
 * @return
 */
public static CustomError getAttributeConflict(
        Set<Tuple> attributeTuples, List<TypedDependency> selected,
        String sentence) {
    Map<String, List<String>> attributeMap =
            generateHashMap(attributeTuples);
    CustomError error = null;
    error = usingAgentConflict(selected, sentence, attributeMap);
    if (error != null) {
        return error;
    }
    error = usingRule1Conflict(selected, sentence, attributeMap);
    if (error != null) {
        return error;
    }
    error = usingRule2Conflict(selected, sentence, attributeMap);
    if (error != null) {
        return error;
    }
    return null;
}

private static CustomError usingRule2Conflict(
        List<TypedDependency> selected, String sentence,
        Map<String, List<String>> map) {
    TypedDependency nsubj = null;
    TypedDependency prep = null;
    for (TypedDependency typedD : selected) {
        if (typedD.reln().toString().contains("nsubj")) {
            nsubj = typedD;
        }
        if (typedD.reln().toString().contains("prep")) {
            prep = typedD;
        }
        if (nsubj != null && prep != null) {
            String w3 = prep.dep().nodeString();
            String w1 = nsubj.dep().nodeString();
            for (Entry<String, List<String>> e : map.entrySet()) {
                List<String> set = (List<String>) e.getValue();
                if (set.contains(w3)) {
                    return new CustomError(sentence, attrErr + w3 + "]");
                }
                if (set.contains(w1)) {
                    return new CustomError(sentence, attrErr + w1 + "]");
                }
            }
        }
    }
    return null;
}

private static CustomError usingAgentConflict(
        List<TypedDependency> selected, String sentence,
        Map<String, List<String>> map) {
    TypedDependency nsubj = null;
    TypedDependency agent = null;
    for (TypedDependency typedD : selected) {
        if (typedD.reln().toString().contains("nsubj")) {
            nsubj = typedD;
        }
```

```
        if ("agent".equals(typedD.reln().toString())) {
            agent = typedD;
        }
        if (nsubj != null && agent != null) {
            String w3 = agent.dep().nodeString();
            String w1 = nsubj.dep().nodeString();
            for (Entry<String, List<String>> e : map.entrySet()) {
                List<String> set = (List<String>) e.getValue();
                if (set.contains(w3)) {
                    return new CustomError(sentence, attrErr + w3 + "]");
                }
                if (set.contains(w1)) {
                    return new CustomError(sentence, attrErr + w1 + "]");
                }
            }
        }
    }
    return null;
}

private static CustomError usingRule1Conflict(
        List<TypedDependency> selected, String sentence,
        Map<String, List<String>> map) {
    TypedDependency nsubj = null;
    TypedDependency dobj = null;
    for (TypedDependency typedD : selected) {
        if (typedD.reln().toString().contains("nsubj")) {
            nsubj = typedD;
        }
        if ("dobj".equals(typedD.reln().toString())) {
            dobj = typedD;
        }
        if (nsubj != null && dobj != null) {
            String w3 = dobj.dep().nodeString();
            String w1 = nsubj.dep().nodeString();
            for (Entry<String, List<String>> e : map.entrySet()) {
                List<String> set = (List<String>) e.getValue();
                if (set.contains(w3)) {
                    return new CustomError(sentence, attrErr + w3 + "]");
                }
                if (set.contains(w1)) {
                    return new CustomError(sentence, attrErr + w1 + "]");
                }
            }
        }
    }
    return null;
}

/**
 * Generates hashmap for tuples
 * @param tuples
 * @return
 */
private static Map<String, List<String>> generateHashMap(
        Set<Tuple> tuples) {
    Map<String, List<String>> map =
            new HashMap<String, List<String>>();
    for (Tuple t : tuples) {
        List<String> currentSet = map.get(t.getSecondElt());
        if (currentSet == null) {
            map.put(t.getSecondElt(), new ArrayList<String>());
            currentSet = map.get(t.getSecondElt());
        }
        currentSet.add(t.getThirdElt());
    }
    return map;
}
```

```
    private static Map<String, List<String>> generateSubtypeMap(
            Set<Tuple> tuples, String keyword) {
        Map<String, List<String>> map =
                new HashMap<String, List<String>>();
        for (Tuple t : tuples) {
            List<String> currentSet = map.get(t.getSecondElt());
            if (currentSet == null) {
                map.put(t.getSecondElt(), new ArrayList<String>());
                currentSet = map.get(t.getSecondElt());
            }
            if (t.getFirstElt().contains(keyword)) {
                currentSet.add(t.getThirdElt());
            } else {
                map.remove(t.getSecondElt());
            }
        }
        return map;
    }
}

/**
 * FileProcessor.java
 * The main class for parsing
 */
package com.kimcortez.sp.parser;

import java.io.StringReader;
import java.util.ArrayList;
import java.util.HashSet;
import java.util.List;
import java.util.Set;
import java.util.StringTokenizer;

import com.kimcortez.sp.erd.models.Entity;
import com.kimcortez.sp.erd.models.Relationship;
import com.kimcortez.sp.parser.models.CustomError;
import com.kimcortez.sp.parser.models.Tuple;
import com.kimcortez.sp.utils.Dictionary;

import edu.stanford.nlp.ling.CoreLabel;
import edu.stanford.nlp.parser.lexparser.LexicalizedParser;
import edu.stanford.nlp.process.CoreLabelTokenFactory;
import edu.stanford.nlp.process.PTBTokenizer;
import edu.stanford.nlp.process.TokenizerFactory;
import edu.stanford.nlp.trees.GrammaticalStructure;
import edu.stanford.nlp.trees.GrammaticalStructureFactory;
import edu.stanford.nlp.trees.PennTreebankLanguagePack;
import edu.stanford.nlp.trees.Tree;
import edu.stanford.nlp.trees.TreebankLanguagePack;
import edu.stanford.nlp.trees.TypedDependency;

/**
 *
 * @author Kim Isle Cortez
 *
 */
public class FileProcessor {
    private LexicalizedParser lp;
    private Dictionary dict;
    private SentenceChecker sentenceChecker;
    private String content;
    private List<String> sentences = new ArrayList<String>();
    private Set<Tuple> entityTuples = new HashSet<Tuple>();
    private Set<Tuple> attributeTuples = new HashSet<Tuple>();
    private Set<Tuple> cardinalityTuples = new HashSet<Tuple>();
    private Set<Tuple> subtypeTuples = new HashSet<Tuple>();
    private Set<Entity> entities = new HashSet<Entity>();
```

```
private Set<Relationship> relations = new HashSet<Relationship>();

public FileProcessor(LexicalizedParser lp, Dictionary dict) {
    sentences = new ArrayList<String>();
    entityTuples = new HashSet<Tuple>();
    attributeTuples = new HashSet<Tuple>();
    cardinalityTuples = new HashSet<Tuple>();
    subtypeTuples = new HashSet<Tuple>();
    entities = new HashSet<Entity>();
    relations = new HashSet<Relationship>();
    this.lp = lp;
    this.dict = dict;
    sentenceChecker = new SentenceChecker(lp, dict);
}

public LexicalizedParser getLp() {
    return lp;
}

public void setLp(LexicalizedParser lp) {
    this.lp = lp;
}

public String getContent() {
    return content;
}

public void setContent(String content) {
    this.content = content;
}

public Set<Tuple> getTuples() {
    return entityTuples;
}

public void setTuples(Set<Tuple> tuples) {
    this.entityTuples = tuples;
}

public Dictionary getDict() {
    return dict;
}

public void setDict(Dictionary dict) {
    this.dict = dict;
}

public List<String> getSentences() {
    return sentences;
}

public void setSentences(List<String> sentences) {
    this.sentences = sentences;
}

public Set<Tuple> getEntityTuples() {
    return entityTuples;
}

public void setEntityTuples(Set<Tuple> entityTuples) {
    this.entityTuples = entityTuples;
}

public Set<Tuple> getAttributeTuples() {
    return attributeTuples;
}

public void setAttributeTuples(Set<Tuple> attributeTuples) {
```

```
    this.attributeTuples = attributeTuples;
}

public Set<Tuple> getCardinalityTuples() {
    return cardinalityTuples;
}

public void setCardinalityTuples(Set<Tuple> cardinalityTuples) {
    this.cardinalityTuples = cardinalityTuples;
}

public Set<Entity> getEntities() {
    return entities;
}

public void setEntities(Set<Entity> entities) {
    this.entities = entities;
}

public Set<Relationship> getRelations() {
    return relations;
}

public void setRelations(Set<Relationship> relations) {
    this.relations = relations;
}

/**
 * Checks if all sentences in the content are valid
 *
 * @param conten
 * - content to check
 * @return error - the error that occured
 */
public CustomError isValidContent(String content) {
    StringTokenizer stringTokenizer =
            new StringTokenizer(content, ".");
    while (stringTokenizer.hasMoreTokens()) {
        String sentence = stringTokenizer.nextToken();
        sentence += ".";
        CustomError error = sentenceChecker.isValid(sentence);
        if (error != null) {
            return error;
        }
    }
    return null;
}

/**
 * Parses the content into typed dependencies
 *
 * @param content
 * - the content to be parsed
 * @return list of lists of typed dependencies for every sentence in the
 * content
 */
private List<List<TypedDependency>> parseContent(String content) {
    StringTokenizer stringTokenizer =
            new StringTokenizer(content, ".");
    List<List<TypedDependency>> tdl =
            new ArrayList<List<TypedDependency>>();
    while (stringTokenizer.hasMoreTokens()) {
        String token = stringTokenizer.nextToken() + ".";
        token = sentenceChecker.reconstruct(token);
        sentences.add(token);
        tdl.add(parseTypedDependencies(token));
    }
    return tdl;
```

```
}

/**
 * Parses the typed dependencies of a sentence
 *
 * @param sentence
 * - the sentence to be parsed
 * @return list of typed dependencies of the sentence
 */
private List<TypedDependency> parseTypedDependencies(
        String sentence) {
    TokenizerFactory<CoreLabel> tokenizerFactory =
            PTBTokenizer.factory(new CoreLabelTokenFactory(), "");
    List<CoreLabel> rawWords2 =
            tokenizerFactory.getTokenizer(new StringReader(sentence))
                    .tokenize();
    Tree parse = lp.apply(rawWords2);
    TreebankLanguagePack tlp = new PennTreebankLanguagePack();
    GrammaticalStructureFactory gsf =
            tlp.grammaticalStructureFactory();
    GrammaticalStructure gs = gsf.newGrammaticalStructure(parse);
    List<TypedDependency> tdl = gs.typedDependenciesCCprocessed();
    return tdl;
}

/**
 * Main processing method
 * @return
 */
public CustomError process() {
    content = content.toLowerCase();
    List<List<TypedDependency>> tdl = parseContent(content);
    return getComponents(tdl);
}

/**
 * Gets the ERD components using typed dependencies of the sentence
 *
 * @param tdl
 * - typed dependency list of the content
 */
private CustomError getComponents(List<List<TypedDependency>> tdl) {
    for (int i = 0; i < tdl.size(); i++) {
        List<TypedDependency> td = tdl.get(i);
        String ruleToUse =
                sentenceChecker.getStructure(sentences.get(i));
        CustomError error = extract(td, sentences.get(i), ruleToUse);
        if (error != null) {
            return error;
        }
    }
    ComponentIdentifier ce =
            new ComponentIdentifier(entityTuples, attributeTuples,
                    cardinalityTuples, subtypeTuples);
    ce.setDictionary(dict);
    ce.identify();
    entities = ce.getEntities();
    relations = ce.getRelations();
    return null;
}

/**
 * Extract contents from typed dependencies
 *
 * @param td
 * @param ruleToUse
 * @param ruleToUse2
 */
```

```
private CustomError extract(List<TypedDependency> td,
        String sentence, String ruleToUse) {
    switch (ruleToUse) {
    case "ssa":
    case "ssb":
        return extractSSB(td, sentence);
    case "ssc":
        return extractSSC(td, sentence);
    case "sse":
        return extractSSE(td, sentence);
    case "ssf":
        return extractSSF(td, sentence);
    default:
        break;
    }
    return null;
}

private CustomError extractSSA(List<TypedDependency> td,
        String sentence) {
    boolean useRule1 = true;
    boolean useAgentRule = false;
    List<TypedDependency> selected =
            new ArrayList<TypedDependency>();
    for (TypedDependency typedD : td) {
        if (typedD.reln().toString().contains("nsubj")
                || "dobj".equals(typedD.reln().toString())
                || typedD.reln().toString().contains("prep")
                || "agent".equals(typedD.reln().toString())) {
            selected.add(typedD);
        }
        if (typedD.reln().toString().contains("prep")) {
            useRule1 = false;
        } else if ("agent".equals(typedD.reln().toString())) {
            useAgentRule = true;
        }
    }
    CustomError error =
            ConflictChecker.getAttributeConflict(attributeTuples,
                    selected, sentence);
    if (error != null) {
        return error;
    }
    if (useAgentRule) {
        extractUsingAgent(selected, sentence);
    } else if (useRule1) {
        extractUsingRule1(selected, sentence);
    } else {
        extractUsingRule2(selected, sentence);
    }
    selected.clear();
    return null;
}

private CustomError extractSSB(List<TypedDependency> td,
        String sentence) {
    List<TypedDependency> selected =
            new ArrayList<TypedDependency>();
    boolean forGeneralAttr = true;
    for (TypedDependency typedD : td) {
        if (typedD.reln().toString().contains("nsubj")
                || "amod".equals(typedD.reln().toString())
                || "dobj".equals(typedD.reln().toString())
                || typedD.reln().toString().contains("conj_")) {
            selected.add(typedD);
        }
        if ("amod".equals(typedD.reln().toString())) {
            forGeneralAttr = false;
```

```
        }
    }
    CustomError error =
            ConflictChecker.getEntityConflict(entityTuples,
                    attributeTuples, subtypeTuples, cardinalityTuples,
                    selected, sentence);
    if (error != null) {
        return error;
    }
    extractAttributes(selected, forGeneralAttr);
    selected.clear();
    return null;
}

private CustomError extractSSC(List<TypedDependency> td,
        String sentence) {
    CustomError error = null;
    error = extractSSA(td, sentence);
    if (error != null) {
        return error;
    }
    error = extractCardinalities(td, sentence);
    if (error != null) {
        return error;
    }
    return null;
}

private CustomError extractSSE(List<TypedDependency> td,
        String sentence) {
    String relation =
            sentence.contains("_can_be_either_") ? "can_be_disjoint"
                    : "can_be";
    String relation2 =
            sentence.contains("only.") ? relation + "_only"
                    : relation;
    List<TypedDependency> selected =
            new ArrayList<TypedDependency>();
    for (TypedDependency typedD : td) {
        if (typedD.reln().toString().contains("nsubj")
                || typedD.reln().toString().contains("conj_")) {
            selected.add(typedD);
        }
    }
    CustomError error =
            ConflictChecker.getSubtypeConflict(subtypeTuples,
                    attributeTuples, selected, sentence);
    if (error != null) {
        return error;
    }
    extractSubtypes(selected, relation2);
    selected.clear();
    return null;
}

private CustomError extractSSF(List<TypedDependency> td,
        String sentence) {
    String relation =
            !sentence.contains("_can_be_either_") ? "can_be_overlap"
                    : "can_be";
    String relation2 =
            sentence.contains("only.") ? relation + "_only"
                    : relation;
    List<TypedDependency> selected =
            new ArrayList<TypedDependency>();
    for (TypedDependency typedD : td) {
        if (typedD.reln().toString().contains("nsubj")
                || typedD.reln().toString().contains("conj_")) {
```

```
            selected.add(typedD);
        }
    }
    CustomError error =
            ConflictChecker.getSubtypeConflict(subtypeTuples,
                    attributeTuples, selected, sentence);
    if (error != null) {
        return error;
    }
    extractSubtypes(selected, relation2);
    selected.clear();
    return null;
}

/**
 * Extracts component via rule 1
 * @param selected
 * @param sentence
 */
private void extractUsingRule1(List<TypedDependency> selected,
        String sentence) {
    TypedDependency nsubj = null;
    TypedDependency dobj = null;
    for (TypedDependency typedD : selected) {
        if (typedD.reln().toString().contains("nsubj")) {
            nsubj = typedD;
        }
        if ("dobj".equals(typedD.reln().toString())) {
            dobj = typedD;
        }
        if (nsubj != null && dobj != null) {
            String w2n = nsubj.gov().nodeString();
            String w3 = dobj.dep().nodeString();
            String w1 = nsubj.dep().nodeString();
            entityTuples.add(new Tuple(w2n, w1, w3));
        }
    }
}

/**
 * Extracts components via rule2
 * @param selected
 * @param sentence
 */
private void extractUsingRule2(List<TypedDependency> selected,
        String sentence) {
    TypedDependency nsubj = null;
    TypedDependency prep = null;
    for (TypedDependency typedD : selected) {
        if (typedD.reln().toString().contains("nsubj")) {
            nsubj = typedD;
        }
        if (typedD.reln().toString().contains("prep")) {
            prep = typedD;
        }
        if (nsubj != null && prep != null) {
            String w2n = nsubj.gov().nodeString(), w2p =
                    prep.gov().nodeString();
            String w3 = prep.dep().nodeString(), w1 =
                    nsubj.dep().nodeString();
            String relation =
                    prep.reln().toString().replace("prep", w2p);
            if (w2n.equals(w2p)) {
                entityTuples.add(new Tuple(relation, w1, w3));
            }
        }
    }
}
```

```
/**
 * Extracts components containing agent
 * @param selected
 * @param sentence
 */
private void extractUsingAgent(List<TypedDependency> selected,
        String sentence) {
    TypedDependency nsubj = null;
    TypedDependency agent = null;
    for (TypedDependency typedD : selected) {
        if (typedD.reln().toString().contains("nsubj")) {
            nsubj = typedD;
        }
        if ("agent".equals(typedD.reln().toString())) {
            agent = typedD;
        }
        if (nsubj != null && agent != null) {
            String w2n = nsubj.gov().nodeString();
            String w3 = agent.dep().nodeString(), w1 =
                    nsubj.dep().nodeString();
            entityTuples.add(new Tuple(w2n, w1, w3));
        }
    }
}

/**
 * Extracts attribute components
 * @param selected
 * @param forGeneralAttr
 */
private void extractAttributes(List<TypedDependency> selected,
        boolean forGeneralAttr) {
    TypedDependency nsubj = null;
    TypedDependency conj = null;
    TypedDependency dobj = null;
    for (TypedDependency typedD : selected) {
        if (typedD.reln().toString().contains("nsubj")) {
            nsubj = typedD;
        }
        if (typedD.reln().toString().contains("conj_")) {
            conj = typedD;
        }
        if ("dobj".equals(typedD.reln().toString())) {
            dobj = typedD;
        }
        if (nsubj != null && dobj != null) {
            String w2n = nsubj.gov().nodeString();
            String w3 = dobj.dep().nodeString(), w1 =
                    nsubj.dep().nodeString();
            attributeTuples.add(new Tuple(forGeneralAttr ? w2n
                    : "unique", w1, w3));
        }
        if (nsubj != null && conj != null) {
            String w2n = nsubj.gov().nodeString();
            String w1 = nsubj.dep().nodeString();
            String w3 = conj.dep().nodeString();
            String w4 = conj.gov().nodeString();
            if (forGeneralAttr) {
                attributeTuples.add(new Tuple(forGeneralAttr ? w2n
                        : "unique", w1, w3));
                attributeTuples.add(new Tuple(forGeneralAttr ? w2n
                        : "unique", w1, w4));
            }
        }
    }
}
```

```
/**
 * Extracts cardinalities
 * @param selected
 * @param sentence
 * @return
 */
private CustomError extractCardinalities(
        List<TypedDependency> selected, String sentence) {
    CustomError error =
            ConflictChecker.getCardinalityConflict(cardinalityTuples,
                    selected, sentence);
    if (error != null) {
        return error;
    }
    TypedDependency nsubj = null;
    TypedDependency dobj = null;
    TypedDependency amod = null;
    for (TypedDependency typedD : selected) {
        if (typedD.reln().toString().contains("nsubj")) {
            nsubj = typedD;
        }
        if ("dobj".equals(typedD.reln().toString())
                || typedD.reln().toString().contains("prep")) {
            dobj = typedD;
        }
        if ("amod".equals(typedD.reln().toString())) {
            amod = typedD;
        }
        if (nsubj != null && dobj != null && amod != null) {
            String w3 = dobj.dep().nodeString();
            String w1 = nsubj.dep().nodeString();
            String w4 = amod.dep().nodeString();
            cardinalityTuples.add(new Tuple(w4, w1, w3));
        }
        if (nsubj != null && dobj != null && amod == null) {
            String w1 = dobj.gov().nodeString();
            String w2 = nsubj.dep().nodeString();
            String w3 = dobj.dep().nodeString();
            cardinalityTuples.add(new Tuple(w1, w2, w3));
        }
    }
    return null;
}

/**
 * Extracts subtypes
 * @param selected
 * @param relation
 */
private void extractSubtypes(List<TypedDependency> selected,
        String relation) {
    TypedDependency nsubj = null;
    TypedDependency conj = null;
    String supertype = "";
    for (TypedDependency typedD : selected) {
        if (typedD.reln().toString().contains("nsubj")) {
            nsubj = typedD;
            supertype = nsubj.dep().nodeString();
        }
        if (typedD.reln().toString().contains("conj_")) {
            conj = typedD;
        }
        if (nsubj != null && conj != null) {
            String w1 = conj.gov().nodeString();
            String w2 = conj.dep().nodeString();
            subtypeTuples.add(new Tuple(relation, supertype, w1));
            subtypeTuples.add(new Tuple(relation, supertype, w2));
        }
```

```
            if (nsubj != null && selected.size() == 1) {
                String w1 = nsubj.gov().nodeString();
                subtypeTuples.add(new Tuple(relation, supertype, w1));
            }
        }
    }
}

/**
 * SentenceChecker.java
 * Class that performs sentence checking
 */
package com.kimcortez.sp.parser;

import java.io.StringReader;
import java.util.List;
import java.util.StringTokenizer;

import javax.swing.JOptionPane;

import com.kimcortez.sp.parser.models.CustomError;
import com.kimcortez.sp.utils.Dictionary;

import edu.stanford.nlp.ling.HasWord;
import edu.stanford.nlp.parser.lexparser.LexicalizedParser;
import edu.stanford.nlp.process.Tokenizer;
import edu.stanford.nlp.trees.GrammaticalStructureFactory;
import edu.stanford.nlp.trees.Tree;
import edu.stanford.nlp.trees.TreebankLanguagePack;

/**
 *
 * @author Kim Isle Cortez
 *
 */
public class SentenceChecker {
    private LexicalizedParser lp;
    private Dictionary dictionary;
    private String regexA =
            "(.*)/NNS?␣(HAS|HAVE)/VB(P|Z)␣(A|AN)/DT␣(.*)/NNS?"
                    + "((\\,/\\,|␣AND/CC)␣(.*)/NNS?)*␣\\./\\.";
    private String regexB =
            "(.*)/NNS?␣(HAS|HAVE)/VB(P|Z)␣A/DT␣(UNIQUE|DISTINCT)/JJ␣"
                    + "(.*)/NNS?((\\,/\\,|␣AND/CC)␣(.*)/NNS?)*␣\\./\\.";
    private String regexC =
            "(.*)/NNS?␣(MUST|MAY|MIGHT|CAN|COULD|SHOULD)/MD␣"
                    + "(.*)/VB(P|Z)?((.*)/(IN|TO))?␣(AT_MOST_ONE|ZERO_OR_ONE|"
                    + "ONLY_ONE|AT_LEAST_ONE|ONE_OR_MANY|ZERO_OR_MANY)/(JJ|VB)␣"
                    + "(.*)/NNS?((\\,/\\,|␣OR/CC)␣(.*)/NNS?)*␣\\./\\.";
    private String regexD =
            "(.*)/NNS?␣CAN/MD␣BE/VB␣EITHER/CC␣(A|AN)/DT␣"
                    + "(.*)/NNS?((\\,/\\,|␣OR/CC)␣(.*)/NNS?){1,}(␣ONLY/RB)?␣\\./\\.";
    private String regexE =
            "(.*)/NNS?␣CAN/MD␣BE/VB␣(A|AN)/DT␣(.*)/NNS?"
                    + "((\\,/\\,|␣OR/CC|␣AND/CC)␣(.*)/NNS?)*(␣ONLY/RB)?␣\\./\\.";

    public SentenceChecker(LexicalizedParser lp, Dictionary dict) {
        this.lp = lp;
        this.dictionary = dict;
    }

    public CustomError isValid(String sentence) {
        CustomError wordError = hasValidWords(sentence);
        String tagged = getTaggedSentence(reconstruct(sentence));
        CustomError formatError =
                (matchesSentenceF(tagged) || matchesSentenceE(tagged)
                        || matchesSentenceC(tagged) || matchesSentenceB(tagged))
                        || matchesSentenceA(tagged)
```

```
                ? null
                : new CustomError(sentence.replaceAll("(\\r|\\n)",
                        ""),
                        "Sentence format doesn't match recognized formats.");
    return wordError == null ? formatError : wordError;
}

/**
 * Performs POS tagging
 * @param sentence
 * @return tagged sentence
 */
public String getTaggedSentence(String sentence) {
    TreebankLanguagePack tlp = lp.getOp().langpack();
    GrammaticalStructureFactory gsf =
        tlp.grammaticalStructureFactory();
    Tokenizer<? extends HasWord> tokenizer =
        tlp.getTokenizerFactory().getTokenizer(
            new StringReader(sentence));
    List<? extends HasWord> sentence2 = tokenizer.tokenize();
    Tree parse = lp.parse(sentence2);
    gsf.newGrammaticalStructure(parse);
    String tagged = parse.taggedYield().toString();
    return tagged.toUpperCase().replaceAll("[\\[\\]]", "")
        .replaceAll(", ", " ");
}

private CustomError hasValidWords(String sentence) {
    sentence = sentence.replaceAll("[^a-zA-Z ]", "");
    StringTokenizer tok = new StringTokenizer(sentence, " ");
    while (tok.hasMoreTokens()) {
        String word = tok.nextToken();
        if (!dictionary.contains(word)) {
            int action =
                JOptionPane
                    .showConfirmDialog(
                        null,
                        "The word '"
                            + word
                            + "' is not found in the system's dictionary. "
                            + "Do you wish to add it?",
                        "Word Not Found",
                        JOptionPane.YES_NO_OPTION);
            if (action == JOptionPane.YES_OPTION) {
                dictionary.addWord(word);
            } else {
                return new CustomError(sentence,
                    "Word not found in the dictionary [" + word
                        + "]");
            }
        }
    }
    return null;
}

/**
 * Returns sentence structure of a given sentence
 *
 * @param td
 * @param sentence
 * @return
 */
public String getStructure(String sentence) {
    String tagged = getTaggedSentence(sentence);
    if (matchesSentenceF(tagged)) {
        return "ssf";
    } else if (matchesSentenceE(tagged)) {
        return "sse";
```

```
    } else if (matchesSentenceC(tagged)) {
        return "ssc";
    } else if (matchesSentenceB(tagged)) {
        return "ssb";
    } else if (matchesSentenceA(tagged)) {
        return "ssa";
    } else {
        return "none";
    }
}

private boolean matchesSentenceA(String sentence) {
    boolean otherConstraint = true;
    otherConstraint =
            sentence.contains(",") ? sentence.contains("AND/CC")
                    : otherConstraint;
    return sentence.matches(regexA) && otherConstraint;
}

private boolean matchesSentenceB(String sentence) {
    boolean otherConstraint = true;
    otherConstraint =
            sentence.contains(",") ? sentence.contains("AND/CC")
                    : otherConstraint;
    return sentence.matches(regexB) && otherConstraint;
}

private boolean matchesSentenceC(String sentence) {
    return sentence.matches(regexC);
}

private boolean matchesSentenceE(String sentence) {
    boolean otherConstraint = true;
    otherConstraint =
            sentence.contains(",") ? sentence.contains("OR/CC")
                    : otherConstraint;
    return sentence.matches(regexD) && otherConstraint;
}

private boolean matchesSentenceF(String sentence) {
    boolean otherConstraint = true;
    otherConstraint =
            sentence.contains(",") ? sentence.contains("OR/CC")
                    : otherConstraint;
    return sentence.matches(regexE) && otherConstraint;
}

/**
 * Reconstructs sentence
 * @param sentence
 * @return
 */
public String reconstruct(String sentence) {
    String tagged = getTaggedSentence(sentence).toLowerCase();
    StringTokenizer tok = new StringTokenizer(tagged);
    String[] words = new String[tok.countTokens()];
    for (int i = 0; i < words.length; i++) {
        words[i] = tok.nextToken();
    }
    for (int i = 0; i < words.length - 1; i++) {
        String w1 = words[i];
        String w2 = words[i + 1];
        if (w1.contains("/nn") && w2.contains("/nn")) {
            String toReplace =
                    w1.substring(0, w1.indexOf("/")) + "␣"
                            + w2.substring(0, w2.indexOf("/"));
            String replacement = toReplace.replace('␣', '_');
            sentence = sentence.replace(toReplace, replacement);
```

```
            }
        }
        String[] cardinalities =
                { "at most one", "zero or one", "only one",
                        "at least one", "one or many", "zero or many" };
        for (int i = 0; i < cardinalities.length; i++) {
            if (sentence.contains(cardinalities[i])) {
                String replacement = cardinalities[i].replace(' ', '_');
                sentence =
                        sentence.replace(cardinalities[i], replacement);
            }
        }
        sentence = sentence.replaceAll("(\\r|\\n)", "");
        return sentence;
    }
}

/**
 * CustomError.java
 * Creates application specific errors
 */
package com.kimcortez.sp.parser.models;

/**
 *
 * @author Kim Isle Cortez
 *
 */
public class CustomError {
    private String source;
    private String error;

    public CustomError(String source, String error) {
        this.source = source;
        this.error = error;
    }

    public CustomError(CustomError error) {
        this.source = error.getSource();
        this.error = error.getError();
    }

    public String getSource() {
        return source;
    }

    public void setSource(String source) {
        this.source = source;
    }

    public String getError() {
        return error;
    }

    public void setError(String error) {
        this.error = error;
    }

    @Override
    public String toString() {
        source = source.replaceAll("_", " ");
        return "Error found: " + error + "\n" + " at sentence: '"
                + source + "'";
    }
}

/**
 * Tuple.java
```

```
 * Represents the ordered triple to represent relations
 */
package com.kimcortez.sp.parser.models;

/**
 *
 * @author Kim Isle Cortez
 *
 */
public class Tuple {
    private String firstElt;
    private String secondElt;
    private String thirdElt;

    /**
     * Default Constructor
     */
    public Tuple() {
        super();
    }

    /**
     * Constructor that initializes the tuple elements as given by the parameters
     *
     * @param firstElt
     * - the first element in the tuple
     * @param secondElt
     * - the second element in the tuple
     * @param thirdElt
     * - the third element in the tuple
     */
    public Tuple(String firstElt, String secondElt, String thirdElt) {
        this.firstElt = firstElt;
        this.secondElt = secondElt;
        this.thirdElt = thirdElt;
    }

    public String getFirstElt() {
        return firstElt;
    }

    public void setFirstElt(String firstElt) {
        this.firstElt = firstElt;
    }

    public String getSecondElt() {
        return secondElt;
    }

    public void setSecondElt(String secondElt) {
        this.secondElt = secondElt;
    }

    public String getThirdElt() {
        return thirdElt;
    }

    public void setThirdElt(String thirdElt) {
        this.thirdElt = thirdElt;
    }

    @Override
    public String toString() {
        return "<" + firstElt + "," + secondElt + "," + thirdElt + ">";
    }

    @Override
    public int hashCode() {
```

```
        final int prime = 31;
        int result = 1;
        result =
                prime * result
                        + ((firstElt == null) ? 0 : firstElt.hashCode());
        result =
                prime * result
                        + ((secondElt == null) ? 0 : secondElt.hashCode());
        result =
                prime * result
                        + ((thirdElt == null) ? 0 : thirdElt.hashCode());
        return result;
    }

    @Override
    public boolean equals(Object obj) {
        if (this == obj)
            return true;
        if (obj == null)
            return false;
        if (getClass() != obj.getClass())
            return false;
        Tuple other = (Tuple) obj;
        if (firstElt == null) {
            if (other.firstElt != null)
                return false;
        } else if (!firstElt.equals(other.firstElt))
            return false;
        if (secondElt == null) {
            if (other.secondElt != null)
                return false;
        } else if (!secondElt.equals(other.secondElt))
            return false;
        if (thirdElt == null) {
            if (other.thirdElt != null)
                return false;
        } else if (!thirdElt.equals(other.thirdElt))
            return false;
        return true;
    }
}

/**
 * Dictionary.java
 * Dictionary class of the application
 */
package com.kimcortez.sp.utils;

import java.io.BufferedReader;
import java.io.BufferedWriter;
import java.io.FileReader;
import java.io.FileWriter;
import java.io.IOException;
import java.util.ArrayList;
import java.util.Collections;
import java.util.List;

/**
 *
 * @author Kim Isle Cortez
 *
 */
public class Dictionary {
    private List<String> words;

    public Dictionary() {
        words = new ArrayList<String>();
    }
```

```
    public List<String> getWords() {
        return words;
    }

    /**
     * Loads dictionary from txt file to list
     * @throws IOException
     */
    public void load() throws IOException {
        words = new ArrayList<String>();
        BufferedReader reader =
                new BufferedReader(new FileReader("resources/dict.txt"));
        String line = reader.readLine();
        while (line != null) {
            words.add(line);
            line = reader.readLine();
        }
        reader.close();
    }

    /**
     * Checks if word is in dictionary
     * @param word
     * @return true if found
     */
    public boolean contains(String word) {
        return words.contains(word.toLowerCase());
    }

    /**
     * Add a word to the dictionary
     * @param word
     */
    public void addWord(String word) {
        words.add(word.toLowerCase());
    }

    /**
     * Saves the current list of word to original file
     * @throws IOException
     */
    public void saveWords() throws IOException {
        Collections.sort(words);
        StringBuilder sb = new StringBuilder();
        BufferedWriter writer =
                new BufferedWriter(new FileWriter("resources/dict.txt"));
        for (String word : words) {
            sb.append(word + "\n");
        }
        writer.write(sb.toString().trim());
        writer.close();
    }
}
```

XII. Acknowledgement

Maraming tao ang nais kong pasalmatan at maraming bagay din ang nais kong ipag-pasalamt na tumulong at nakatulong sa akin upang matapos ko ang SP ko na ito.

Unang una sa lahat ay nais kong magpasalamat kay Lord dahil alam ko na hindi niya ako pinabayaan hanggang sa matapos ko ang SP ko. Alam kong kontrolado niya ang lahat ng mga bagay na naganap noong ginagawa ko ang SP ko (halimbawa, nanakaw ang laptop ko). "The Lord is in control."1 Kings 20:13.

Nais ko ring magpasalamat sa aking pamilya - Juancho (tatay), Gemma (nanay), Katherine (ate), Leslee (ate), at Kris Anne (bunso) - sa lahat ng bagay, lalo na sa aking nanay na sumuporta sa aking gastusin at nagpahiram ng laptop. Gayudin si Renz (pinsan) na nagpahiram ng laptop nang masira naman ang laptop ng aking nanay. Salamat sa inyo.

Nais ko ring pasalamatan ang aking mga kaibigan sa batch 2010 ng block 12 lalo na sina babidi - Jaye Renzo Montejo, Patrick Ballicud, Jerson Cristobal, at Jayrell Recido. Salamat sa mga kalokohan at kasiyahan. Salamat sa inyo.

Nais ko ring pasalamatan si Zee Codizar na tumiyak na matatapos ko ang SP ko, at tinulungan akong tapusin ang mga huling revisions ko. Salamat. :) Huwag kang mag-alala, tutulungan din kita, kahit ayaw mo. ;)

Nais ko ring magpasalamat sa aking adviser, Gregorio Baes, Ph.D. (*candidate*), na tinutulungan pa din ako kahit inis na inis na siya sa akin at ang kulit kulit ko. Gayundin sa aking iba pang mga guro. Salamat ma'am at sir.

Nais ko ring pasalamatan ang stackoverflow.com at google.com, gayundin ang ibang open-source projects na sobra ding nakatulong sa akin. Salamat sa inyo.

Maraming maraming maraming salamat sa mga tao, at hindi tao, na nabanggit, gayudin sa mga hindi nabanggit. Masaya ako at tunay na nagpapasalamat sa lahat ng bagay na ginawa ninyo at nagagawa ninyo. Salamat sa inyo at natapos ko ang SP ko. At ngayon, magtatapos na ako. Wakas.