AUTOMATIC MODEL ANSWER GENERATION FOR SIMPLE LINEAR ALGEBRA-BASED MATHEMATICS QUESTIONS

Rajpirathap Sakthithasan

168260N

Degree of Master of Science

Department of Computer Science & Engineering
University of Moratuwa
Sri Lanka

May 2018

AUTOMATIC MODEL ANSWER GENERATION FOR SIMPLE LINEAR ALGEBRA-BASED MATHEMATICS QUESTIONS

Rajpirathap Sakthithasan

168260N

Thesis submitted in partial fulfillment of the requirements for the degree Master of Science in Computer Science and Engineering

Department of Computer Science & Engineering

University of Moratuwa
Sri Lanka

May 2018

DECLARATION

I declare that this is my own work and this thesis does not incorporate without acknowledgement any material previously submitted for a Degree or Diploma in any other University or institute of higher learning and to the best of my knowledge and belief, it does not contain any material previously published or written by another person except where the acknowledgment is made in the text. Also, I hereby grant to University of Moratuwa the non-exclusive right to reproduce and distribute my thesis, in whole or in part in print, electronic or other medium. I retain the right to use this content in whole or part in future works.

....
Sakthithasan Rajpirathap Date

The above candidate has carried out research for the Masters of Science thesis under my supervision.

....
Dr. Surangika Ranathunga Date

ABSTRACT

This research is focused on automating the process of generating answers to simple linear equation related mathematical problems.

Simple linear algebra based questions are a part of most Mathematics examinations. These linear algebra questions can appear as word type problems, where the question description is given in a textual form. Addition, subtraction, multiplication, division and ratio calculation are some of the known categories for linear equation based word type problems. Addition and subtraction based problems can be further divided based on their textual information as change type (join-separate type), compare type, and whole-part type. This research focuses on linear equation questions belonging to these three categories.

Mainly four approaches are followed by existing research for answer generation for linear algebra questions. These are rule/inference based, ontology based, statistical based, and hybrid based approaches.
In this research, a statistical approach is selected to automatically generate answers for simple linear algebra based model questions. The implemented system shows better accuracy than the other statistical systems reported in previous research for the same types of questions. This result is achieved by using ensemble classifiers and smart feature selection. Also, a new data set is created for training and evaluation purposes.

ACKNOWLEDGEMENTS

I would like to express profound gratitude to my adviser, Dr. Surangika Ranathunga, for her invaluable support by providing relevant knowledge, materials, advice, supervision and useful suggestions throughout this research work. Her expertise and continuous guidance enabled me to complete my work successfully.

I am grateful for the support and advice given by Dr. Malaka Walpola, by encouraging continuing this research till the end. Further, I would like to thank all my colleagues for their help in finding relevant research material, sharing knowledge and experience and for their encouragement. Also, I would like to thank for the teachers who involved in data creation, data annotation works in this research.

I am as ever, especially indebted to my parents for their love and support throughout my life. I also wish to thank my loving friends, who supported me throughout my work. Finally, I wish to express my gratitude to all my colleagues at my workplace, for the support given to me to manage my MSc research work.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS

Abbreviation	Description
POS	Parts-of-speech
DOL	Dolphin Language
CFG	Context-free grammar
VBD	Verb, past tense
PRP	Personal pronoun
SVM	Support vector machine
NB	Naive Bayes
GCE O/L	General certificate of education ordinary level

1 INTRODUCTION

1.1 Overview

Managing assessments through computers is a trending research area. It can be considered as an important and easy way to assess a student's performance in exams. Such computer based systems not only eliminate teacher's manual intervention, but also help standardize the examination assessment process. Computer-based assessment is now becoming common for subjects such as Mathematics.

In automated assessment of student answers, for many question types, a model answer should be provided in advance. Currently, a teacher has to manually provide the model answer for each question for assessment to be carried out. However, there is research to automate the answer generation process as well.

Mathematics syllabus contains different types of questions such as linear algebra, geometry, and calculus. Out of these, simple linear algebra based questions mostly contain the addition, subtraction, division, multiplication and ratio calculations, and geometry based questions. Addition and subtraction based word problems can be further divided into sub-types by considering the textual information in those questions. These categories are change type (join-separate type), compare type, and whole-part type [1, 2].

This research focuses on "change", "compare" as well as "whole-part" type of linear algebra mathematics questions.

1.2 Question Types

As mentioned earlier, three types of the linear equation based questions are addressed in this research: change type, compare type, and whole-part type [1, 2]. Based on the textual information of a question we can determine the question type. In this research, questions with two known variables and one unknown variable related linear equations are considered. These questions are written as word-type problems, using two-three sentences.

Change Type:

In the "change" type of questions, a particular numerical value or a quantitative value of an entity is getting changed over time. Usually, the initial quantity value is changing over time or state in sentences of a question. So simply the formula can be explained as below [1],

Start value + change value = summation or result
Or
Start value - change value = difference or result

Here the 'start value' is an initial value of a quantity of an entity in a question. 'Change value' represents the change of quantity values of an entity/variable in that question. As a result, the answer is derived as a sum or difference value of a quantity in a question.

For example, consider the below change type questions:
Q1: Pete had 3 apples. Ann gave Pete 5 more apples, how many apples does Pete have now?

Q2: Vimal has 10 books and he gave 2 books to Nimal. So, how many books does Vimal have now?

Here, in Q1 the quantity value of apples is increased. So the initial value of apples '3' is later changed by additional '5' apples. So, the end result is a sum and it is 8. So, we can say that this quantity value of apples is changing with time (3→8).

In Q2 also, the quantity of books changes (10→8). The end result is the subtraction value and it is 8.

Thus the above questions can be categorized as linear algebra "change" type questions.

Comparison Type:

In "comparison" type of questions basically there is a comparison of two numerical values of entities. The comparison is implicitly captured by identifying the difference between the quantities or numerical values in a question. Simply the formula can be stated as below [1],

Initial value + or - difference value = second value

The quantity value of "initial value" is compared with the quantity value of "second value" in a question. "Difference value" denotes the quantity difference between the "initial value" and "second value" in a question. In most cases, this type of questions has keywords such as "more", "less", "than" in its context.

For example, consider the below questions,

Q1: Joe has 3 balloons. His sister Connie has 5 balloons. How many more balloons does Connie have than Joe?

Q2: Luis has 6 goldfish. Carla has 2 more goldfish than Luis. How many goldfish does Carla have?

In Q1 the formula can be expressed as "first value (5) - second value (3) = difference (unknown)" and in Q2 the formula can be expressed as "second value (unknown) - difference (2) = first value (6)"

As mentioned earlier for "comparison" type problems, the equation can be a comparison between two quantities and the question can relate to a larger quantity or smaller quantity of the difference between two quantities. For example, let's consider another model question:

Nuwan and Vimal attended an exam. Nuwan correctly answered 11 questions and Vimal correctly answered 16 questions. How many questions did Vimal correctly answer than Nuwan?

The answer to this question is in **16 -11 = 5.** Here, we can see that the difference in quantity is the answer to this question.
Thus the above questions can be categorized as linear algebra "Compare" type questions.

Whole-Part Type:

The whole-part type of problems can be simply expressed as, "part value + part value = whole value" [1].

In some cases, the whole value and one of the part values are mentioned in the context of the question and asking for the remaining part value at the end of the answer. On the other hand, some other questions mention about the quantity of 'parts' and asking about the 'whole' quantity value. For example, consider the below questions,

Q1: There are 6 boys and 8 girls in the volleyball team. How many children are in the team?

Q2: Joe and Tom have 8 marbles when they put all their marbles together. Joe has 3 marbles. How many marbles does Tom have?

In Q1, the formula can be derived as "part value + part value = _____".
From Q2, the formula is derived as "(part value + ___ = whole value) or (_____ + part value = whole value)".

In this type of questions, we can see that the same kind of variable/entity's value is divided into parts and the 'whole 'quantity is expressed by considering the 'part' values.

In Q1, the boys and girls are some part of child entity/variable. On the other hand, marbles are divided as parts in Q2.

Thus the above questions can be categorized as linear algebra "whole-part" type questions.

1.3 Problem and Motivation

In the past, there were mainly four kinds of approaches followed by researchers in this particular area, viz.: rule/inference based approach, ontology based approach, statistical approach and hybrid approach. Many issues still exist while using these approaches. For example, the main issue with a rule/inference based approach is the usage of a fixed set of patterns with manually predefined rules [3, 4]. On the other hand, the ontology based approach requires domain knowledge to create an ontology map to solve a question [5, 6, 7, and 8]. Hybrid approaches also mostly depend on rules and predefined templates for a fixed set of question types. So they also have some issues of rule/inference based approaches [9, 10, 11, 12, 13 and 14]. In the statistical approach, probabilistic models play a role. And they do not have the limitations mentioned above with respect to other categories. There has been some past research using deep learning based approaches [15, 16] as well. Although this deep learning based approach is promising, it requires a large amount of data samples to create a system with good accuracy.

Some of the past research that used statistical based approach is considered to have the ability to solve linear equation based mathematical question types. Those statistical based approaches have shown better performance [17] than the other approaches [3, 4, 5, 6, 7, 9, 10 and 13] for the same kind of linear equation based question types. Even it avoids the aforementioned issues of other approaches [3, 4, 5, 6, 7, 9, 10 and 11]. However, there are some limitations in this previous statistical based approach [17]. In this previous statistical approach, cascading classifiers are used to solve the question. So the dependency issues and uncertainty issues of classifiers are the main drawbacks of their approach. Also, this previous statistical approach is such a lengthy approach. To adapt with any new question type, each of the classifiers in their system has to be separately trained with appropriate data samples. This system was not trained with the real math samples, rather was trained with some other data sets such as 'Newswire Text', 'Sub-corpus' of the RTE data samples due to lack of real math sample problems. So the statistical based approach is a time-consuming approach when it comes to the integration task of new question types in the future.

1.4 Objectives

The objective of this research is to design and implement a system to automatically generate answers for math word problems for simple linear algebra questions of change type, compare type and whole-part types. More specifically, we focus on word problems with two known and one unknown variables.

1.5 Contributions

To address the objectives mentioned in the previous section, a system has been developed with the following capabilities:

- Automatically convert the questions into their intermediate form as parts-of-speech (POS) form and extract the features of the question samples.
- Predict the formula for a given question sample and generate answers for that sample.

Following modules are developed to achieve the above mentioned goal.

- **Feature extractor module**
  This module is implemented to extract features of the training data to train the system.

- **Parser module**
  Two types of parser modules are implemented,
  - *Word to number paraphrase parser*
    This module is developed to convert the word-based numbers into their numerical form in the questions. For example, the word "thirty-five" in a question will be converted as '35' as its numerical form.

  - *Parts of speech (POS) tag parser*
    This module is developed to parse each and every question sample and convert it into the intermediate form containing POS tags.

- **Trainer and Classifier module**
  This module is developed to create a classifier module to be trained using training samples. Multiple types of classifiers are tried out to find the best one.

- **Answer solver module**
  This module is developed to take the predicted formula as its input and to output the final answer.

1.6 Organization of the Thesis

The rest of the thesis is organized as follows.

Chapter 2 reviews theories and related work in linear equation based word-type problem solving.

Chapter 3 explains the approach used in this research and the architecture design of our system for simple linear equation based mathematics question solver.

Chapter 4 evaluates the work presented in the thesis. It consists of details of data used for evaluation, experiments carried out and the obtained results along with a discussion on the observed results.

Chapter 5 concludes the thesis with future work.

2 LITERATURE SURVEY

2.1 Overview

Mainly there are four kinds of approaches used for the automatic answer generation word-type questions that contain linear equations. They are, rule/inference based systems, ontology based systems, statistical based systems and hybrid systems.

2.2 Rule/ Inference based approaches

The rule/inference based approach basically uses some rules and inferences to generate answers for mathematical model questions. They use stored knowledge in a form of rules and templates to process them further functionalities in the system.

Matsuzaki et al [9] carried out a research to understand the complexity of mathematical word problems that are encountered by Japanese students for the Japanese university entrance examination. This study includes a method for transforming word problems into logic representations to be solved by an automatic problem solver. They have used a semantic parser on a related topic and solved algebra word problems.

Basically, this is a very small research conducted for learning purposes in their subject area. Their problem solving pipeline has been defined as shown in Figure 2.1.

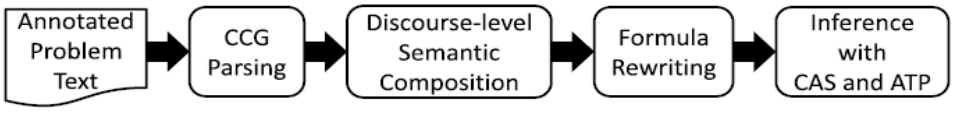

Figure 2.1: Pipeline to solve linear algebra problem [9]

The input is given as an annotated problem text. The rules and constraints are defined as annotations to help translate the sentences in the problem text into a logical form. The grammar-based parser is used for this purpose.

The aforementioned research focused on solving real and natural number arithmetic, 2D and 3D geometry, linear algebra (multiplication type, addition type), pre-calculus, and calculus related mathematical questions. Main drawback of this approach is the presence of rules and constraints that affect the flexibility to adapt to a new question type.

Dellarosa [18] also proposed an approach to solve a math problem by using a rule based method. The system takes its input as a propositional problem text of a question. The propositional based problem text is generated by some rules for each of the sentences in a question. To start a processing cycle, the system loads the first set of propositions (corresponding to the first sentence) and sets its current goal.

After that, a collection of functions then cycle through the rules, looking for rules whose conditions are satisfied by the contents of the goal stack. When no more rules are satisfied by the contents, the current cycle stops and it begins with the next set of propositions. Rule firing continues until no more rule conditions are satisfied, and no more sentences are left to read. Usually, an answer should have been derived by this point. In this research, the simple addition type of whole-part, compare type of questions are mostly addressed.

The main drawback of this approach is the usage of rule based functions. Because, in some scenarios those rule based functions fail to generate proper propositional form of a given input question.

There are some common issues that exist with this rule/inference based approach. Most of the rule/inference based approaches rely on rules/inferences for input sentences [9, 18]. The presence of rules and constraints affects the flexibility to adapt to new question types. The semantic representations of the word problem based features are not properly handled with this rule/inference based approach [9]. This is the reason why the lexicalized features cannot be generalized well for unseen words. The other issue is that the complicated noun phrase based word problems cannot be handled by this rule/inference based approach [9].

2.3 Ontology based approaches

An ontology is a set of concepts, such as things, events, and relations that are specified in some way (such as a specific natural language) in order to create an agreed-upon vocabulary for exchanging information [5]. Ontologies can be a source to generate knowledge to represent the relationships between real world entities in a problem scenario. When we take a mathematical model question in the form of a word problem, we can identify the entities and related variables of those entities in its context.

Then, the relationship map can be generated based on the domain knowledge of that problem scenario. In the working memory, the map will be created as an ontology map for a mathematical word problem and the relationships between objects are determined from that ontology map. After that, the type of the math question is identified and the equation is generated according to that problem type. The derived formula is applied with the variable values to the equations and that formula is passed to a mathematical engine to compute the formula results to generate the answer for a mathematical model question.

Figure 2.2 shows the abstract work flow of ontology approach.

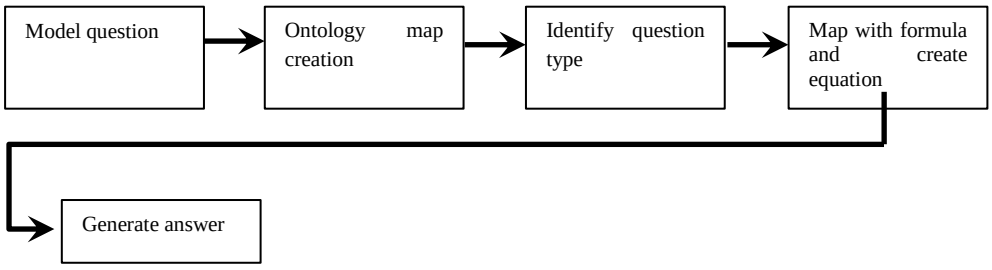

Figure 2.2: Abstract work flow of ontology approach [5]

Here, the ontology map has been used to identify the problem category of this model question with the help of fuzzy logic. Then, the formula is selected based on the equation type and the equation is created by mapping the variables in the formula. Finally, the final answer is computed with the help of other external engines. In this approach, there are some limitations that still exist, such as, the problem of needing domain knowledge to create the ontology map, and issues with mapping the appropriate variables to the correct formula.

Morton and Qu [6] present a framework that can solve some types of mathematical word problems. Their framework is based on fuzzy logic- ontology model.

Their system works for mathematics questions in the domains of investment, distance and projectile domain specific areas of the mathematical problem solving. The primary goal of their research is to create a basic system for learning purposes. So, they considered only a few question types with limited domain areas. In this research [6], addition types of linear equation based question types are addressed.

The main limitations of this approach are limited domains of question types; need for a domain expert for new question type integration, usage of third-party Wolfram [6] math engine to generate the answer for their math solver.

Shuming and et al [7] proposed a semantic parsing and reasoning approach to automatically solve math number word problems. They have designed a totally new meaning representation ontology language DOL (they called as Dolphin language for ontology representation) to bridge natural language text and mathematics expressions. A context free grammar (CFG) parser is implemented to parse natural language text to DOL trees. Reasoning module is implemented to derive math expressions from DOL trees by applying the semantic interpretation of DOL nodes.

They mentioned that they achieved a good precision and a reasonable recall on their test set of over 1,500 problems. They have used 9,600 semi-automatically created grammar rules to parse to their CFG parser.

The drawback of their approach is that general math word problems are much harder to handle through this approach, because the entity types, properties, relations and actions contained in general word problems are much larger in quantity and more complex in quality. The reason is their parser mostly failed coverage on properties, relations and actions of the word problems.

Further, they also mentioned that sometimes they were able to parse a problem successfully, but they were unable to derive math expressions in the reasoning stage. This is often because some relations or actions in the problem were not modeled appropriately. But still this approach did not rely on any third party tool like Wolfram [6] search engine to compute the answer. This research mostly addresses the comparison type and ratio, type of word problems in different domain areas in linear equation question type.

LeBlanc and Weber-Russell [8] carried out research on word problem simulation. They have used a bottom-up approach to implement their word problem solver. They create a relationship map in memory by using situated or action oriented interpretation of text of a question to solve a given word problem.

Their approach is based on two components, EDUCE and SELAH [8]. EDUCE is a kind of parser that is distinct for each word problem that is considered. EDUCE focuses on quantities, pronominal reference, noun compounds, set partitions, time-sequence, ellipsis and the references to previous sets.

SELAH is an integrated tool that is used to integrate text information that exists in a working memory represented by EDUCE. The drawback of this model is in its bottom-up approach. Most of the time it does not allow building proper data relationships. Also, the model is not fit to design the relationship between data sets and objects. Further, the model allows computing a solution only for some change type and compare type related linear equation based questions.

There are issues that exist in ontology based approaches. In this approach, researchers commonly considered only a few types of mathematical questions in limited domain areas [6]. Domain experts are needed to derive the domain knowledge to write logic to create an ontology map to solve a question with this ontology approach. In this ontology based approach, when they used the parser, it mostly failed coverage on properties, relations, and actions. Also they mostly covered only a limited amount of domains. The bottom-up approach does not allow building proper data relationships using prior knowledge. Because their model does not fit to correctly design the relationship between data sets and objects. A common issue in this approach is to map the proper variables to the correct formula to create equations to solve [7, 8].

2.4 Statistical approaches

Statistical approaches rely on a probabilistic related methodology to generate an answer for a mathematical model question. The success of this approach lies in relying on the training process of the system with the data samples we take for a particular question type. This approach is not dependent on rules or patterns as in the rule based or ontology based approaches. There are some parameter-tuning techniques that can be used to improve the accuracy of this type of statistical based systems. The accuracy can be increased by increasing the amount of correct training samples that we give to this system in its learning process.

The basic flow of this approach is as shown in Figure 2.3.

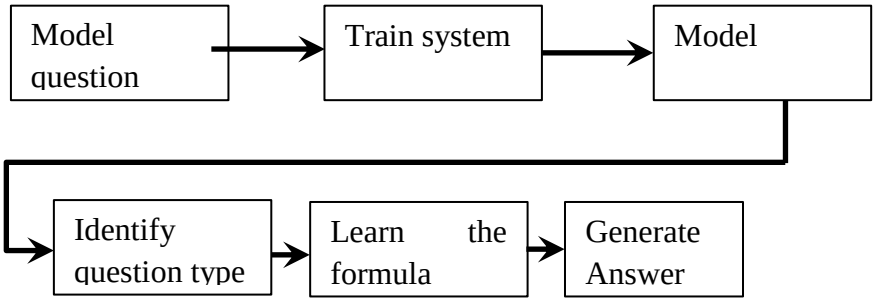

Figure 2.3: Abstract flow diagram of statistical approach

This type of statistical based system approach is more robust to the irrelevant information mapping and can more accurately provide the answer for ambiguous words contained in mathematical problems. The models have been used to identify the question types of the mathematical problems and the system learns the formula to be mapped for that particular kind of question.

Mostly, arithmetic related addition type, subtraction type; surplus type and multiplication type questions are addressed using this statistical based approach.

The most recent statistical approach is proposed by Roy et al [17]. This approach uses four types of statistical classifiers to solve the problems that are part of elementary school mathematics word problems. These include a quantity identifier classifier that helps in finding related quantities, a quantity pair classifier that is used in finding the operands, an operation classifier that is used in selecting an arithmetic operation, and finally an order classifier that is used to order the operands for subtraction and division cases. They have used perceptron algorithm to create the classifier in this approach.

The authors mention that their approach mostly avoids the limitations of rule based and ontology based approaches [9 and 10]. However, this approach limits itself by allowing only one arithmetic operation (among addition, multiplication, subtraction and division) at a time with two or three operand candidates of linear equation based question types. This approach did not use any predefined templates. Rather, it is using classifiers to identify the operations to solve questions. So it is eliminating the issue of rules and hybrid (rule plus statistical) [11 and 14] approaches.

Still this approach has some issues. This approach used four different classifiers in their system to do different works in their problem solving approach. Those classifiers are cascading classifiers. They are dependent on each other for their continued work. So if any of the classifiers fails, then this system will fail to continue to solve a math problem as we expect. This is one of the major issues in this existing statistical approach. In their approach the classifiers are not actually trained with real math samples. Rather they have used a different type of corpuses (Newswire Text, Sub-corpus of the RTE Datasets) to train their classifiers. This can affect the overall system accuracy. Also their approach is a long approach because they have to train their classifiers in a serial manner which will take more time to adapt to a new question type.

2.5 Hybrid approaches

Hybrid approaches are mostly implemented with the help of rule/inference based and statistical-based implementations. In some hybrid approaches, the rule/inference logics are used in the initial stage of the system's process, and they used statistical models in their final stage of the question-solving approach. On the other hand, some other approaches use the statistical models in their initial stage of the question-solving process and rules/inference logics are used in the final stage in their

approach. Mostly the statistical models are used to predict the necessary operation that they want to take in the question-solving process. The rules are mostly used to identify the entities and mapping the numerical values to a predefined formula template in their solution process.

Mukherjee and Garain [19] survey these works. They mentioned that Kushman's [20] algorithms try to map the mathematical word problem to a system template that contains a set of equation templates. For example, a predefined equation template ax+ by = c can be used as a template plotting the variables that exist in the mathematical question sentence. These system templates are collected from training samples by analyzing the pattern and category of question by considering the many numbers of features defined in the system. These kinds of hybrid approaches separately define the individual templates for each question. Here the drawback is they assume that these templates will reoccur for new questions too. In this approach majorly, addition, division and multiplication types of linear equation related questions are addressed by researchers.

Kushman's [20] method has been implemented with a set of pre-defined templates that are used to define the linear equation based formulas to represent the mathematical model questions. They have defined templates with a set of slots with two types; number slots and unknown slots. Number slots have been used to fill up the number values derived from the model question and the unknown slots have been used to assign the nouns that indicate the answers of questions.

The main problem with this approach is that we can have many possible assignments for each number slot. Thus it uses more hypothesis space to generate the answer for a question. Also in determining how to map natural language into equation templates, the system examines hundreds of thousands of "features" of training examples. These features also use more space. In this approach, improper assignments can happen when we map the values to separate slots. We always need to stick with the same format of questions for a particular template. So if we are going to generate an answer for a new question that is slightly different from this question type, then we may need to define a new set of templates. This will lead us to take more memory requirements as well [20]. In this research, the addition, subtraction and multiplication types of linear equation related questions are mostly addressed and they have handled up to two unknown variables.

Zhou's et al [10] approach has considered reducing the total number of slots to reduce the space required to avoid more memory usage.

Here, they only considered having numerical slots in a model question and avoided considering the noun slots to map the nouns in a model question. They defined effective features to describe the internal relationship within the numbers and unknowns (nouns) in the model question. They did not consider creating separate unknown slots to map the nouns in a model question. So this approach basically reduced the hypothesis space, giving the advantage of learning the inference process when generating the answers to a model question.

This approach used a quadratic programming model to describe the template selection and number assignment in a model question.
However, it was mentioned that there are some issues that still exist with this approach. For example, it is hard for their algorithm to relate the word “forfeits” to “minus”. The other problem is that their algorithm only considers the single noun as the entity of a word problem. For complicated noun phrases, their algorithm may fail to respond.

In this research [10] as well, addition, subtraction and multiplication types of linear equation related questions are addressed.

Hosseini et al [11] have proposed a container-entity based approach that solves the mathematical question sentence with a state transition sequence. They have also mostly addressed solving arithmetic word problems that include addition of whole-part type of linear equation question types. There is a set of containers included within the states and each of the containers specified some set of entities. Those entities are identified by some heuristic rules. After that the quantity of each entity type change encountered by looking at the associated verb category for that entity type. After that, they have used a classifier to classify the sentence into one of the seven categories.
They have presented a novel approach towards finding the solution for simple arithmetic word problems. It is called ARIS. This system analyzes each sentence in the problem statement to identify the relevant variables and their values. After that, ARIS maps this information into an equation that represents the problem and enables its (trivial) solution. In their approach, a problem text is divided into fragment parts where each part is represented as a transition between two world states in which the quantities of entities are updated or observed. They referred to these fragment parts as sentences. Their system ARIS learns to predict verb categories in sentences using syntactic and semantic features from small, easy-to-obtain training data. ARIS represents the world state in a math problem into entities, sets, quantities, attributes and their relations and takes advantage of the circumscription assumption and successfully fills in the information gaps. Finally, ARIS makes use of attributes and avoids the irrelevant information on the math word problems.

But there are some drawbacks in their system. The authors mention that they have come across text entailment issues, implicit action errors, usage of irrelevant information mapping issues, and parsing issues. This system highly depends on verb categorization. The statistical part of this approach is used only for the purpose of this verb categorization prediction and then their system uses a set of equations to map with the variables found in questions.

Lin et al [12] has presented a tag-based hybrid statistical math word problem solver with reasoning and explanation. It combines the statistical framework with logic inference. Their system analyzes the body and question texts into their associated tag-based logic forms and then draws inferences from them. This particular research is done for Japanese language based mathematical problems.

Comparing with the rule-based approaches done previously, this proposed hybrid statistical approach alleviates rules coverage and ambiguity resolution problems. Their tag-based approach also provides the flexibility of handling various kinds of related questions with the same body logic form.

Moreover, this system [12] supports 11 types of mathematical problems. An important fact to be remembered regarding this research is that they have done it only for Japanese language related mathematical question solving problems. Further, this research is a combination of statistical and logic based inference approaches. Thus new inference rules have to be defined for a new question type.

Amnueypornsakul and Bhat [14] presented a system that can identify the discourse structure of the mathematical story problem that will enable comprehension in mathematics, and it utilizes the information in the discourse structure towards generating the solution in a systematic manner.

They build a random forest based multistage classifier model that predicts the problem type, identifies the function of sentences in each problem, and extracts the necessary information from the question to generate the corresponding mathematical equation. The equation generation stage is a rule-based deductive learner that combines the result of sentence type classification (and sign prediction) to derive the numerical quantities needed to plot in the equation templates.

In this approach [14], several classifiers are used for different tasks. Most of these classifiers are cascading classifiers. The problem type classifier, sign prediction classifier, sentence function identification classifier are the classifiers used in this approach. Their system initially predicts the sign and the sentence type of the question. Then in the next stage the rule based deductive learner is combines the result of sentence type classifier and sign prediction classifier to derive the numerical quantities needed to plot in the equation templates.

In this study [14], they mostly focused on “change” and “whole- part” type of linear equation related questions. There are drawbacks exist in this hybrid approach. Their rule based learner is the main module of their equation generator component. So there is an issue of integration of new question types. This is because each type of the question needs to be integrated with new rules in the system. And here they have used multistage classifiers in their statistical part. So any classifiers can fail at any stage and system may go to the unexpected stage.

2.6 Summary

Mainly there are four kinds of approaches followed by existing research. Those approaches are, the rule/inference based approach, ontology based approach, and statistical based approach and the hybrid approach. There are still many issues that exist with these approaches, as mentioned above. For example, the main issue of a rule/inference based approach is the usage of a fixed set of manually defined patterns

with predefined rules/inferences [10]. On the other hand, the ontology based approach has a lack of domain knowledge to create an ontology map to solve a question [5, 6, 7, and 8]. Hybrid systems have faced entailment, implicit action errors, usage of irrelevant information related issues and parsing issues [11]. Also, some hybrid systems highly depend on a verb categorization method and the predefined set of templates give less flexibility to new question types [12].

In the statistical approach, probabilistic models play a role. Past research in statistical approaches considered solving linear equation based mathematical question types. However, dependency of several classifiers, uncertainty of the cascading classifiers are the main issues in the past statistical approach [17].

3 METHODOLOGY

3.1 Overview

This chapter describes the methodology employed in developing the system for automatic model answer generation for word-type Mathematics questions. More specifically, we focus on simple linear algebra questions with one addition or subtraction. The questions contain one unknown variable and two known variables in its context.

We employ a statistical technique that involves the creation of a probabilistic model to generate answers to questions that are input to the system. The probabilistic model is generated using training data samples belonging to "compare, change and whole-part" types of linear equation based mathematical questions. Figure 3.1 shows the system architecture.

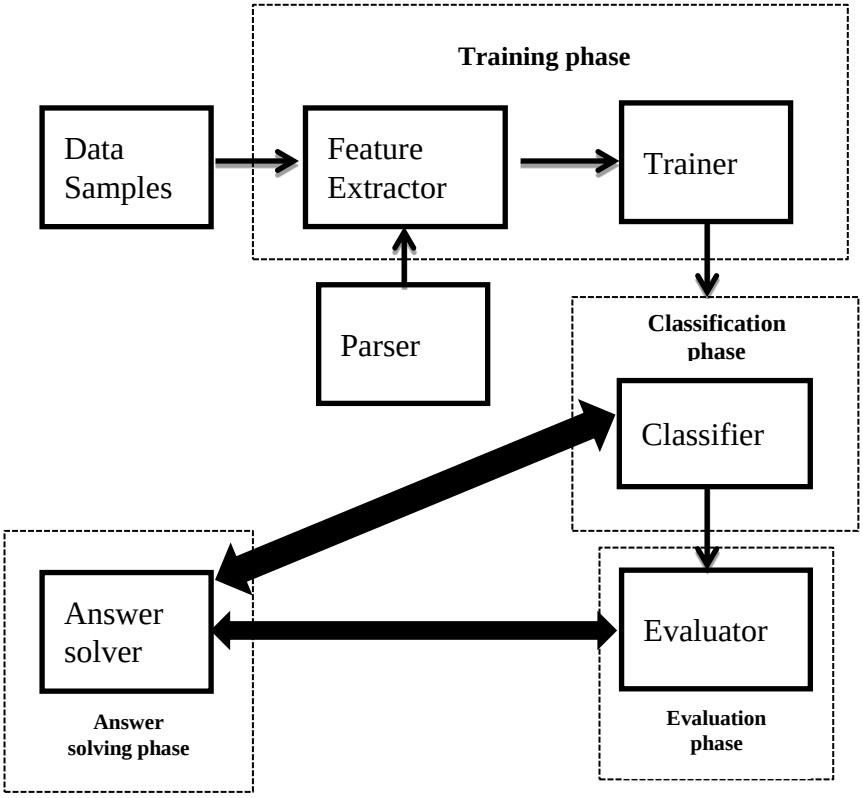

Figure 3.1: System Architecture

There are four main phases in this system as showed in Figure 3.1. The phases are training phase, classification phase, answer solving phase, and evaluation phase. This particular system takes the annotated training samples as input in the training phase. It parses the training samples to a parser to convert the questions to an intermediate form to define and construct features from the training samples. Then the feature extractor module works towards extracting the features. After that the trainer module initiates the training process. The classifier module creates the trained model and the

evaluation module starts the evaluation process of the trained model by using test samples. Finally, the answer solver module generates the answer.

3.2 Data collection

Since this research focuses on a statistical system; it mainly relies on the data set to train itself. The three types "compare type", "change type" and "whole-part type" of linear equation based questions and their formulae are the data sources of this research. The samples are simple linear equation based addition and subtraction related mathematical word problems with 2 or 3 sentences. They also have two known variables and one unknown variable. The known variables are expressed numerically or textually, and the unknown variable is expressed by a word. For example, consider the below questions *Q1* and *Q2*,

*Q1: "There are **41** pencils in the drawer. Mike placed **30** pencils in the drawer. **How many pencils** are now there in total?"*

*Q2: "Pamela had **six** fresh oranges in her bag. Her mother gave her **seven** more oranges. **How many oranges** does Pamela have now?"*

Question *Q1* contains two numerical variables as **41** and **30**. On the other hand *Q2* contains the word-typed numeric values as **six** and **seven**. But in both questions, the unknown variable is implicitly expressed in the last sentence.

There are some publicly available question datasets. The Add-Sub dataset [21] contains linear algebra questions similar to the question types that are addressed in this research. Also, the ARIS Dataset [11] contains questions of one mathematical operation based linear equation story problems. Other than that, Roy et al's [17] dataset also contains one operation based linear equation mathematical problems. Also, the SingleOP dataset [22] contains one operation based linear equation problems that are suitable for our research. The Dolphin dataset [7] is freely available on the web. But it mostly contains complex simultaneous equation based algebra questions of addition, subtraction as well as multiplication, and division. There we could not find simple linear algebra questions that have two known variables and one unknown variable. Even though it contains the change, compare, and whole-part type of questions, those questions are complex questions with more than 3 variables and more than three sentences in the question description. Also, the questions in this data set [7] are mixed and not divided into the question types. So the Dolphin dataset [7] is not taken in this research.

The primary source of data is Sri Lanka's GCE O/L model questions, which are collected from Sri Lanka's GCE O/L teachers. Out of the 1713 data samples were collected for this research, 782 samples are collected from SriLanka's O/L teachers, and 389 samples are taken from Add-Sub dataset. Also, 112 samples are collected from the ARIS dataset [11]. 230 samples are collected from Roy et al's [17] and

remaining 200 samples were collected from SingleOP dataset [22]. Table 3.1 shows the statistics of the dataset.

Table 3.1: Data sources and number of samples

Data source	**Number of samples**
SriLanka's O/L teachers	782
Add-Sub dataset	389
ARIS dataset	112
SingleOP	200
Roy et al dataset	230
Total	**1713**

The data samples are filtered and collected from these public datasets by considering the type of the questions we consider in this research. But it is to be noted that SingleOP dataset [22] is used only for the testing purpose in this research.

3.3 Labeling the data samples

Labeling the samples is manually done. The questions we considered in this research contain 2 numerical known variables. As mentioned earlier, "change", "compare" and "whole-part" types are considered in this research. By looking at the samples, there are 8 types of labels/classes that are identified and associated with our data set. Those labels/classes are **X-Y, X+Y, Y-X, Y+Z, Y-Z, Z-Y, Z+W and Z-W**. The **X-Y** and **Y-X** labels are for Compare type of questions and **Y-Z, Y+Z, Z-Y** are for Change type of questions. **Z+W** and **Z-W** classes are associated with the Whole-Part type of questions.

This particular labeling approach is introduced and proposed in our research because it enables us to identify that the same type of formula (E.g.: **X+Y, X-Y, Y-Z**... etc.) supports a set of questions. So the formula is a kind of category and is considered to be used as a label/class in our approach.

In the above labeling approach, the letter '**X**' represents the 1st variable and letter '**Y**' represents the 2nd variable in a "Compare" type of question. On the other hand for "Change" type of questions, 1st variable is represented by letter "**Y**" and the 2nd variable is represented by letter "**Z**" to differentiate the variety of question types. For "Whole-part" type of questions, the 1st variable is represented by letter "**Z**" and the 2nd variable is represented by letter "**W"**.

For example, the label **X-Y** represents the deduction operation between the 1st numerical variable and the 2nd numerical variable in a question and **Y-X** represents the deduction operation between the 2nd numerical variable and the 1st numerical variable in a "compare" type of question. This labeling scheme was selected because there are some cases where we can find the larger numeric value is located as the 2nd

numerical variable and the smaller numeric variable is located as the 1st variable in a question context. So, in this case, we have to do the deduction operation from the 2nd numerical variable to 1st numerical variable.

Label **Y+Z** represents the addition operation between the 1st numerical variable and the 2nd numerical variable of a "Change" type of question. They are labeled like this to differentiate the formula type and to separately identify the negative/positive impact of change, compare and whole-part type of questions.

To represent the above 8 labels/classes, a considerable amount of "change", "compare" and "whole-part" type of questions are collected as mentioned in the previous paragraph. Also in the dataset, there are questions with either two or three sentences, Table 3.2 shows the number of questions per each type.

Table 3.2: Amount of questions collected for training purpose

Label	Question Type	No. of questions
X-Y	compare	200
X+Y	compare	115
Y-X	compare	250
Y-Z	change	320
Y+Z	change	250
Z-Y	change	200
Z+W	whole-part	200
Z-W	whole-part	178

For example, consider the following questions. While Q1 has 2 sentences, Q2 has 3 sentences. But both questions are having 2 numerical known variables and one unknown variable.
Each question is annotated with its corresponding formula.

Q1: *Marry found 63 marbles but 24 were broken. How many unbroken marbles did Marry find?,* ***X-Y***

Q2: *Vimala had 11 pounds in her hand bag. Her friend gave her 30 more pounds. How many pounds does Vimala have now?* ***X+Y***

The Kappa statistic measurement of this collected dataset is 0.8598, which is calculated from the results given by two domain experts. The calculation is as given in Table 3.3.

Table 3.3: Kappa statistics calculation

	Judge 1			
Judge 2		**Correctly annotated questions**	**Wrongly annotated questions**	**Total**
	Correctly annotated questions	1692	2	1694
	Wrongly annotated questions	3	16	19
	Total	1695	18	1713

$K = \frac{(p_0 - p_e)}{(1 - p_e)}$

$P_0 = (1692 + 16) /1713 = 0.997$

$P_e = (1694/1713)* (1695/1713) + (19/1713)* (18/1713)$

$= 0.9786$

$K = (0.997 – 0.9786) / (1 – 0. 9786) = 0.0202/ 0.0242 = 0.8598$

Here K refers to Kappa statistic. Judge 1 and Judge 2 represent Sri Lanka's GCE O/L mathematics teachers who participated in this dataset evaluation process.

3.4 Pre-processing

Each of the questions is programmatically tokenized to add a white space between the sentences and full stops in question. This allows the parser to easily identify the sentences. Leading and trailing white spaces of the questions are cleaned before passing to parser component for further processing. Unwanted spaces between the sentences in questions are also removed programmatically in preprocessing step.

3.5 Parser module

Two different parsers are implemented in this research: word-to-number paraphrase converter and the POS converter.

- **Word-to-number paraphrase converter**

In some of the model questions, the numeric value can appear in word format. In those cases, this particular parser takes the tokenized plain input sentence and converts only the word-based numeric values into number based numerical value in the question. The formats of the word-based numeric values are in standard format [23] as in the following examples shown in Table 3.4.

Table 3.4: Word to number conversion samples

Word based numeric values	Number based numeric values
One thousand, two hundred eighty-nine	1289
One thousand, twenty-five	1025
Ten thousand, sixty-eight	10068
Fifteen	15
Three hundred fourteen	314
Three hundred fifty-two	352

For example consider the following model questions,

- ***Q1:*** *Sunil has* ***eighty five*** *packets. Kamal has* ***twenty*** *less packets than Sunil. How many packets does Kamal have?*

And the output from the parser is,

- ***Q1:*** *Sunil has* ***85*** *packets. Kamal has* ***20*** *less packets than Sunil. How many packets does Kamal have?*

The parser separately identifies only the word-based numeric values in the question context and then converts them to the actual number based numerical values.
This word-to-number paraphraser module is implemented with the help of freely available word2Num open source package [40].

- **POS converter**

Parser outputs a POS tagged sentence. For this, Pen Tree Bank based POS tagger [17] was used. For example, consider the below model question,

Question*: Ravi had 9 deposits in his bank. His dad gave him 7 more deposits. How many deposits does Ravi have now?*

The parser will output the below result:

1st sentence - *Ravi had 9 deposits in his bank*
Custom POS structure:
{'noun1': 'deposits', 'verb_base': None, 'val': '9', 'sentence': 'Ravi had 9 deposits in his bank.', 'adjective': None, 'verb1': 'had', 'singular_noun2': None, 'prop_noun': 'Ravi', 'index': 0, 'preposition': 'in', 'singular_noun1': 'bank', 'val2': None, 'verb2': None, 'ques_length': 3, 'noun2': None, 'ques_form': None, 'prop_noun2': None}

2nd sentence - *His dad gave him 7 more deposits*
Custom POS structure:
{'noun1': 'dad', 'verb_base': None, 'val': '7', 'sentence': 'His dad gave him 7 more deposits .', 'adjective': 'more', 'verb1': 'gave', 'singular_noun2': None, 'prop_noun': None, 'index': 1, 'preposition': None, 'singular_noun1': None, 'val2': None, 'verb2': None, 'ques_length': 3, 'noun2': 'deposits', 'ques_form': None, 'prop_noun2': None}

3rd sentence - *How many deposits does Ravi have now?*
Custom POS structure:
{'noun1': 'deposits', 'verb_base': 'have', 'val': None, 'sentence': 'How many deposits does Ravi have now ?', 'adjective': None, 'verb1': None, 'singular_noun2': None, 'prop noun': 'Ravi', 'index': 2, 'preposition': None, 'singular_noun1': None, 'val2': None, 'verb2': None, 'ques_length': 3, 'noun2': None, 'ques_form': 'How', 'prop_noun2': None}

3.6 Feature extractor

Feature extractor is a component in this system to extract the features of questions. This component takes the POS tagged representation of a question as input to extract the features. The output from this module is a feature vector that can be passed to our trainer module.

This POS tag based feature extraction technique has been followed by Roy et al [17]. But there it has been used for extraction of very simple features. In contrast, in this research, our system used this technique to extract the entire set of features (both complex and simple).

There are 36 features that are defined and extracted by a feature extractor for change, compare and whole-part type of questions. The individual features are activated whenever they satisfy the condition for a particular question. In activated mode, these features return **True** and otherwise they return **False**.

3.6.1 Feature set description

As mentioned earlier, 36 features were identified for the three question types in this research.

20 out of 36 features are newly introduced in this research and 10 features are derived from Amnueypornsakul and Bhat's [14] work (first 10 features given below). The last 6 features are taken from Roy et al [17]. But actually there are 11 features identified from Roy et al [17]. However, 5 out of 11 of Roy et al's [17] features are overlapping with Amnueypornsakul and Bhat's [14] 10 features. So, 36 features are considered as unique features in this research.

1. Comparative adjective that exists near to a noun in a question

Most of the compare type of questions exists with comparative words as adjectives. The words 'less', and 'more' are a type of comparative based adjectives existing in questions of compare type.

Further, these types of adjectives mostly exist in 2nd or 3rd sentence in a 'compare' type of question with nouns.

For example, a sentence from a compare type of question is '*Saman has 6* ***pens*** ***less*** *than Muru'.* The statement contains '*less'* as a comparative adjective near to plural noun '*pens'*.

2. Positive comparative adjective present in a question

There are words in a question that can give a positive impact on the numerical values or quantities existing in question. By considering the positive comparative adjectives, we can identify whether the variable/quantity value of a particular noun is getting positive based improvement or not.

There are many keywords that can exist in questions to make positive changes in a variable/quantity value. The most common keyword is the comparative adjective word "*more*".

So, this particular feature will get activated whenever there is a positive comparative word in the 2nd or 3rd sentence of a question.

3. Negative comparative adjective existing in a question

There are words in a question that can give a negative impact on the numerical values or quantities existing in a question. By considering the negative comparative adjectives, we can identify whether the variable/quantity value of a particular noun is getting a negative value or not.

There are many keywords that can exist in questions to make a negative change in a variable/quantity value. The most common keyword is the comparative adjective word "*less*". So, this particular feature will get activated whenever there is a negative comparative in the 2nd or 3rd sentence of a question.

4. Question having conjunctive prepositions in its context

Almost all the questions of compare type are having conjunctive propositions. These conjunctive prepositions provide a clear way to identify a question as compare type. The keyword "*than*" is the most occurring conjunctive preposition in the dataset.

Conjunctive prepositions can exist in any sentence of a particular question. For example, consider the below compare type questions,

"Vimala is 34 years old. Kamala is 9 years younger ***than*** *Vimala. How old is Kamala?"*

In this question the keyword "*than*" exists in the middle of the sentence of the question.

*"Udari walked 1 mile and ran 0.6 miles. How much further did Udari walk **than** run?"*

In the above question, the keyword "*than*" exists in the last sentence.

5. Last sentence contains the unknown variable part in a question

In this research, as mentioned earlier, we considered questions with 2 known variables and 1 unknown variable. The unknown variable is mostly associated with the same proper noun entity that is getting changed by some actions. So, by identifying the existence of an unknown variable in the last sentence of the question, the final value of the known variable can be mapped to that unknown variable at the end of the solution of the question.

For example, let's consider the below sentence:

*"Dan picked 9 **grapes** and gave Sara 4 of the grapes. How many **grapes** does Dan have now?"*

Here the unknown variable is "final quantity of *grapes*", which is present in the 2nd sentence of this question. And we already know that the quantity change for the known quantity of "*grapes*" happened in the first sentence itself.

6. Question having two matching singular nouns and a coordinating conjunction

Some of the three sentence based compare type questions are having singular nouns in their second sentence with their quantity values. The third sentence is about the difference between the quantity values mentioned in the second sentence of the model question.
Both these singular nouns are joined with a coordinating conjunction in the second sentence.

For example, consider the below question:

*"Diana made cookies. She used 0.625 kg **flour** and 0.25 kg **sugar**. How much more **flour** than **sugar** did Diana use?"*

In this question "*flour*" and "*sugar*" are singular nouns that are attached to their quantities and are residing in the 2nd sentence of the question. The 3rd sentence is about the quantity difference between those singular nouns in the question.

7. Subordinating conjunction exists within the proper nouns.

This particular feature works with compare type questions having three sentences. This feature ensures that the mostly occurring conjunctions such as "*than*" exist in

the compare type of questions. This conjunction is mostly located within the proper nouns or action makers in the 3rd sentence of the question.
For example, consider the below question:

*"Rosy and Mary planned a trip. Rosy marked 6 places and Mary marked 5 places on a map. How many more places did **Rosy** mark **than** **Mary**?"*

The underlined subordinating conjunction word "*than*" exists within the proper nouns "*Rosy*" and "*Mary*" and they are the proper nouns/action makers in the question.

8. Question contains verb lemmas

In most of the questions, the quantity changes happen for a particular noun that is related to a proper noun in that question. And the relationship between the quantity change and a particular proper noun entity can be identified using the verb based lemmas that exist near to the proper noun and related noun.

For example consider the below question,

*"Grandma **had** 8 strawberries. Grandpa **gave** her 3 more strawberries. How many strawberries does Grandma have now?"*

The verb of the given sentence "*Grandma had 8 strawberries*" has changed in the 2nd sentence as "*Grandpa gave her 8 more strawberries*". And thus the problem has 2 verb lemmas (*have* and *give*). The quantity of noun entity "*strawberries*" is changed due to the change of verb form which can be identified using the verb lemmas in the above question.

So if a relationship between the proper noun and the noun exists by considering the verb lemmas, this feature gets activated.

9. Question contains different proper nouns for the same action

This feature is mostly suitable for the whole-part type of questions. In questions there can be same actions performed by different proper nouns/action makers to initiate an event/action.

For example, let's consider the below whole-part type of question,

*"**Keith** grew 6 turnips. **Alyssa** grew 9 turnips. How many turnips did they **grow** in all?"*

In the above question, the proper nouns *Keith* and *Alyssa* are making an action *'grow'*. And those actions are separately done by both proper nouns. At the end the question, it is asking about the total result of the actions performed by these separate entities.

So, this feature gets activated if separate proper nouns exist to make an action on the question.

10. Splitting collective nouns

There are entities represented by a collective noun in a question. This pattern mostly occurs in a whole-part type of questions.

For example, consider the below question sample,

"Mike has 10 ***coins****. 7 of his coins are* ***dimes*** *and the rest are* ***pennies****. How many are* ***pennies****?"*

In the above sample, the *dimes* and *pennies* are two nouns and the collective noun *'coins'* represents *dimes* and *pennies*. The quantity of the *dimes* is given in the question itself and the quantity of the *pennies* is asked at the end of the question.

So this feature ensures that the 3rd sentence contains the noun which is related to the collective noun in the previous sentence.

To identify the relationship between the collective noun and its related nouns, the list of collective nouns and their related nouns are maintained in the system.

11. Index position, distance between comparative adjective word and noun in a threshold value

This feature is activated whenever a compare type of question contains a comparative adjective word that is closely located to a noun in the same sentence in a question. A fixed threshold value is predefined to activate this feature by considering the distance between the comparative adjective word and a noun.

This feature gets activated if this distance is less than or equal to the threshold value. 2 unit of distance is set as the default value for this threshold. This threshold value is defined by looking at the training samples of compare type of questions.

Also, this newly introduced feature is closely related to the previous feature, but it differs from the distance measurement of words.

For example, let's consider the sentence "*Kamal has 8* ***more*** *pieces of* ***cake*** *than Ravi*". Here the distance between the comparative adjective "*more*" and the noun "*cake*" is calculated as 2 units of distance.

12. 1st and 2nd sentences having the same proper nouns

There are questions with 2 sentences. In most of this type of questions, the 2nd sentence is asking for a final value of some verb/noun based quantity. Mostly the

proper nouns in the sentences can act as action makers. The initial sentence in this type of questions also mentions about a quantitative value of a verb/noun based entity for a particular proper noun referring to a person/place.

So this newly introduced feature will ensure that the proper noun present in the 2nd sentence is matched with the proper noun present in the 1st sentence for the same verb / nouns. For example,

*"**Nimal** walked 1 mile and ran 0.6 mile. How much further did **Nimal** walk than run?"*

In the above question, the proper noun "*Nimal*" matches in 1st and 2nd sentences. So, we can conclude that "*Nimal*" is the one that made an action change in this particular question.

13. 1st and 3rd sentences having the same proper nouns

There are questions with 3 sentences. In most of these types of questions, the 3rd sentence is asking for the final value of some verb/noun based quantity of a person or a place, which made action change in the question. The initial sentence in these types of questions also mentions about a quantitative value of a verb/noun based entity for a particular proper noun referring to a person/place.

So, this newly introduced particular feature will ensure that the proper noun present in the 3rd sentence is matched with the proper noun present in the 1st sentence for same verb / nouns. For example,

*"**Dina** made cookies. She used 0.625 kg flour and 0.25 kg sugar. How much more flour than sugar did **Dina** use?"*

In the above question, the proper noun "*Dina*" is matching in 1st and 3rd sentences. So, we can conclude that "*Dina*" is the one that made an action change in this particular question.

14. Question having matching verbs and its singular noun

Some of the compare type questions with two sentences are having verbs in their first sentence that made the sentence indicate the actions done by someone. The second sentence contains the singular noun form of those verbs that convert the second sentence into a question by considering the actions mentioned in the first sentence.

So, this feature ensures that the question in the second sentence is about the actions that have been mentioned in the first sentence. In this case, the feature will return true Boolean value.

For example,
*"Eve **ran** 0.7 miles and **walked** 0.6 miles. How much further did Eve **run** than **walk**?"*

In the above question "*ran*" and "*walked*" are verbs that mention about actions in this question. The singular nouns are "*run*" and "*walk*" which are matched with the singular noun form of verbs in the first sentence.

The verbs and singular noun comparison is done by getting the lemmatized version of the verbs in the first sentence by matching with the nouns in the second sentence. WordNetLemmatizer is used to do the lemmatization task in the implementation of this feature.

15. First numerical value is greater than the second numerical value

In this research, we consider only questions with 2 numerical variables. Also, the numerical value is changing with some actions based on the contextual information in the question.

By looking at the changes that occurred in a particular numerical variable, we can understand if an action is positive or negative in the question's context. So, this particular feature gets activated when the 1st numerical value in a question is greater than the 2nd numerical value of the question.

16. Question contains exactly two numeric values in 1st or 2nd sentences

In this research, we are considering only two numerical values based model questions that require solving. The first numerical value is placed in the 1st or 2nd sentences and the second numerical value always resides in the 2nd sentence of the question in the dataset.

So, this feature makes sure that those two numeric values actually exist in the 1st or 2nd sentence of a question and it returns a True Boolean after getting activated.

17. Question contains the same proper nouns in all the sentences

This particular feature gets activated only when there are three sentences in the questions. In some questions, there are two participants or action makers. The answer to the question is about the difference between the quantities of those action markers.

For example,
"***Amal*** *and* ***Vimal*** *are running a farm business.* ***Amal*** *bought 8 cows and* ***Vimal*** *bought 5 cows. How many more cows did* ***Amal*** *buy than* ***Vimal****?"*

In the above question, the first sentence mentions about "*Amal*" and "*Vimal*", which are two proper nouns. The second sentence is about their actions and the third sentence is about their final value.

So, this particular newly introduced feature ensures that the action makers, which are proper nouns, are correctly exist in the sentences. Further, the order of the proper noun is also considered in this feature.

18. 2nd sentence having the same type of verb in the question

This particular feature works with 3 sentences based compare type of questions, because, in most of the questions, the second sentence contains information about an action performed by any persons or proper nouns. The third sentence is asking for the quantitative difference between the actions that is mentioned in the second sentence. In this type of questions, the same action is performed by two different pronouns.

So, this feature is implemented to ensure that the action mentioned in the last sentence is actually performed by the proper nouns in the second sentence.

For example, consider the below sample model question,
"*Rosy and Mary planned a trip. Rosy **marked** 6 places and Mary **marked** 5 places in a map. How many more places did Rosy **mark** than **Mary**?"*

In the above question, the action/verb "*marked*" is matched with the proper nouns "*Rosy*" and "*Mary*". The last sentence also asks about the quantity difference between the action performed by "*Rosy*" and "*Mary*".

So, this newly introduced feature gets activated when it identifies a matching action in the second sentence.

19. Question having exactly two proper nouns

In most of the 3 sentence based questions, there are two proper nouns that exist as action makers in the second sentence. So, here this feature ensures that the second sentence of the question contains exactly two proper nouns.

20. Matching proper nouns that exist in the same order in the sentences

The order of the proper noun is an important factor while getting the final answer of the question. The same proper nouns can exist in a different order in the sentences of a question or they can exist in the same order. In most of the cases, the final answer value of the model question is dependent on the order of proper nouns in some of the compare type of questions with 3 sentences. For example, consider the below model question:

"***Sehan** and **Nihal** own a tea factory. **Sehan** bought 250 shares and **Nihal** bought 650 shares. How many less shares did **Sehan** buy than **Nihal**?"*

In the above question, the proper noun order is "*Sehan*", "*Nihal*". The last sentence in this question also has the order of proper nouns as "*Sehan*" and "*Nihal*".

In some cases, the last sentence can be changed as "*How many more shares did Nihal buy than Sehan?*" Here the order of proper nouns is "*Nihal* and *Sehan*".

So this newly introduced feature is useful in identifying the positive/ negative effect that occurs due to the order of the proper nouns/ action makers in a question. The feature gets activated when the order of the proper noun is maintained in the same order in the all sentences of a question.

21. The final value of the question is related to the main proper noun in the question

This is implemented for "change type" of questions with three sentences. With some of the questions, the first sentence starts with a proper noun having some quantity value based nouns on it. The middle sentence explains the ways this quantity is getting changed. The final sentence is about the final quantity value of that related proper noun that is mentioned in the first sentence, which is also considered as the main proper noun entity in that question.

So, this feature ensures that the quantity change is happening only to the noun that is related to the main proper noun of the question. It checks the proper noun of the last sentence and ensures it is similar to the noun in the first sentence.

For example, consider the below question,

*"**Rosita** has 12 cake pieces in her bag. Her mother gave her 7 cake pieces. How many cake pieces does **Rosita** have now?"*

Here the question is about the main proper noun "*Rosita*" and the related noun is "*cake pieces*". The final value also is asked about the quantity value of cake pieces that Rosita has.

22. Possessive pronoun of the main proper noun exists near the noun/quantity changer in a question

In some of the "change type" questions, the quantity changes are made by some noun entities that can be called as "quantity changers" of those questions. Those noun entities mostly exist near to the possessive pronoun of the main proper noun in a particular question. By identifying this pattern, we can make sure that the quantity based changes that happened because of some other noun entities in that question. Further, we can also ensure that the quantity based changes that occurred in a variable and that are related to the main proper noun identified in the first sentence of this question.

For example, consider the below question:

"***Sam*** *has 9 dimes in his bank.* ***His*** ***dad*** *gave him 7 dimes. How many dimes does* ***Sam*** *have now?"*

By considering the "*His dad*" word combination, we can confirm that some action/quantity changes are done by Sam's dad. Also, those two words "*His*" and "*dad*" are next to each other in the sentence. So, by checking the position based distance, we can confirm if the relationship of those words/entities is high or low. A predefined threshold value is set as 2 units of position based distance. So if the position based distance is less than or equal to our threshold value then the relationship of those words is considered as high otherwise low.

23. The change action has made a negative impact on a quantity

This particular feature mostly gets activated in change type of questions. There are quantity based changes that happen in change type of sentences due to some negative and positive actions. By correctly identifying the negative or positive effect, we can decide whether we have to deduct the quantity or add the quantity to the initial quantity value of a noun in a change type of question.

The negative impact in a question can be captured by looking at the verb and its corresponding position in a sentence. In many "change type" of questions, the words *'but'* and *'and'* come as coordinating conjunctions by placing themselves in the middle of the question indicating that a negative change happening in a quantity of a noun. For example, consider the below questions:

"*Tom found 7 seashells* ***but*** *4 were* ***broken****. How many unbroken seashells did Tom find?"*

In this question after the word "*but*" the negative word "*broken*" is appearing. So the "*but*" word can be considered as an indicator to identify a negative change in this question.

"Kamal picked 20 oranges ***and*** ***threw*** *10 oranges. How many did Kamal pick than throw?"*

In the above model question the word "*and*" is appearing as a coordinating conjunction which is very close to the negative word "*threw*" in its second sentence.

In questions with three sentences also, the negative impact of quantity related patterns is captured. For example, consider the below question:

"*There are 7 crayons in the drawer.* ***Mary took*** *3 crayons out of the drawer. How many crayons are there now?"*

In the above question the verb "*took*" appears next to a proper noun "*Mary*". And the verb '*took*' is making a negative impact on the quantity in this question. So, by checking this closely appearing pattern, we can confirm that the quantity of the crayons is reduced by an action "*took*" performed by "*Mary*".

To identify the negative words in this research, currently, a set of predefined negative words/verbs are manually added to the system by looking at the training dataset that we collected for this research. For example 'broken', 'damaged', 'passed', 'wasted' are such kind of keywords.

24. Change action made a positive impact on quantity in the question

The quantity variable in a "change type" question can be changed by a positive impact/action on the question such that the final value of that variable can increase at the end.

Mostly the action related sentence is located in the second sentence in these types of questions. So, the positive impact can be easily captured in the second sentence. For example, let's consider the below sentence,

"Sam had 9 dimes in his bank. His dad ***gave him*** *7 dimes. How many dimes does Sam have now?"*

Here the phrase "*gave him*" can make a positive change in the quantity of dimes. The phrase is a kind of bigram in this particular sentence. By checking the action word/verb in this sentence, we can confirm that the quantity change happened to the noun "*dime*" of the main proper noun "*Sam*".

So, this feature will ensure that the bigram pattern of "Verb (VBD) - Personal Pronoun (PRP)" is appearing in the second sentence with positive verbs.

To identify the positive words in this research, currently, a set of predefined positive words/verbs is manually added to the system by looking at the training dataset that we collected for this research. 'Added', ''collected', 'integrated' are such keywords in the positive word list.

25. Action maker/proper noun exist with the nearest verb

In "change type" of sentences, the quantitative reduction mostly occurs because of some proper noun based entities. In most of the cases, the action/verb and the quantity value exist in the nearest index positions in a sentence in this kind of questions.

For example,
"*Rosy had 9 apples in her bag. Her* ***mother gave*** *her* ***7*** *more apples. How many apples does Rosy have now?*

The proper noun "*mother*", verb "*gave*" and the quantity "7" are placed next to each other in the nearest index positions in the second sentence in this question.
Here the "*mother*" is an action maker and the quantity change done by her is "7". So, this particular feature will ensure that the action maker and the action with its quantity are in the nearest index positions in the sentence of change type of questions.

26. Action/quantity change happened for the same entity in a question

In this research, we have considered the quantity changes only for a particular variable in a question. This ensures that the change of quantity happens in the same entity variable of a question. We have to consider the quantities only for that particular entity variable in a question. For example,

"*Dan picked **9 limes** and gave Sara **4** of the **limes**. How many limes does Dan have now?*"

In the above question, "*lime*" is an entity and the quantities 9 and 4 are only related to that "*lime*" entity. Also, the quantity values 9 and 4 are positioned very close to the noun entity "*lime*". By looking at this pattern, we can assume that the particular quantity is related to the nearest noun entity in a sentence. This particular pattern should exist to ensure that the change in a quantity of a noun is happening to the same noun entity of the question.

27. Co-reference resolution to identify the existence of the proper noun in the following sentences

This feature is applicable for questions with both two and three sentences. In some questions, the first sentence has the proper noun of the question. But the following sentences in that question may have co-references which can be used instead of the proper noun that we identified in the first sentence itself. For example, consider the following questions,

*Q1: **Diana** made biscuits. She used 0.625 kg flour and 0.25 kg sugar. How much more flour than sugar did **she** use?*

*Q2: **Sam** had 9 dimes in his bank. His dad gave him 7 dimes. How many dimes does **he** have now?*

In Q1 the proper noun '*Diana'* is referred as its co-reference *'she'* in the subsequent sentences. Q2 also has the same pattern for the proper noun '*Sam'*.

By checking the personal pronouns in the second or third sentences in a question, the proper noun in the first sentence is mapped for this feature.

28. Collective noun exists in sentences

A collective noun represents some sub nouns. In that case some of the whole-part types of questions, there are collective nouns which can be considered as a super class type of some other nouns existing in a question.

For example, consider the below sample,

"There are 6 ***boys*** *and 8* ***girls*** *on the volleyball team. How many* ***children*** *are on the team?"*

In the above question, the 1st sentence mentions about two nouns as '*boys'* and '*girls'* and the 2nd sentence contains the collective noun of these two nouns as '*Children'*.

So, this newly introduced feature will ensure that the collective noun exists in some nouns in the question and gets activated if it finds the particular pattern.

To identify the collective nouns a list of collective nouns and its nouns are collected from web resources [39] and maintained in the system. For example, the collective noun 'children' can represent the group of 'boys' and 'girls'. Also, the collective noun 'mammals' can represent the group of elephants and cows and some other animals. So our list contains this kind of information to identify the collective nouns in the system.

29. Question explicitly mentions some keywords related to an addition operation

In some of the whole-part type of questions, there are some keywords explicitly mentioning the result of the action happening in a question. Words such as "total", "sum and "all" exist in sentences. And these keywords explicitly mention that there is an addition operation needed to get the answer. For example, consider the below question,

"There are 12 pens in the drawer. Mike placed 15 pens in the drawer. How many pens are now there in ***total****?"*

In the above question, the keyword "*total*" explicitly mentions about the operation that we need to perform in this question to get an answer. And in this case, it is addition operation.

So, this newly introduced feature checks whether there are any such keywords in the question and gets activated if it finds the particular pattern. The keywords are manually extracted from the data samples for this purpose.

Such keywords are kept in a keyword bank to use in the system for this feature implementation.

30. Proper nouns separately perform same action

This feature is most suitable for whole-part type of questions because some such questions have more than one proper noun in a question. These proper nouns are separately performing some actions. So the questions of this kind contain some quantity change done by a particular proper noun, but it is asking about the quantity change done by the other proper noun entity/action maker. For example, consider the below sample question,

*"**Bobbi** and **Sandi** put 12 dimes into a change purse. **Sandi** put in 8. How many did **Bobbi** put in?"*

In the above question, the proper nouns '*Bobbi'* and '*Sandi'* perform the same action. The question itself mentions about the quantity put by '*Sandi'*. At the end, it asks about the quantity put by '*Bobbi'*.

31. Checks the matching-units present in the question

This feature checks whether the units of the quantities are matching in all sentences in a question.

32. Checks bigrams from sentences containing quantities

This feature checks whether the quantity in a sentence appears with any bigrams in the sentences in a question.

33. Checks matching proper nouns

This feature checks whether the proper nouns in the questions are matching in the sentences of that question. So we can make sure that the quantity changes happen only related to that entity.

34. Checks the order of quantities

This features checks and confirms the order of the quantities appears in a question's sentences. So that, numerical values can be plotted in the formula in a correct order.

35. Quantity change happens near to a noun

This feature checks whether the quantity change actually happens near to a noun in a question.

36. Checks matching nouns in the sentences

This feature checks whether the nouns in the last sentence are matching with other sentences in a question.

3.6.2 Training and classification module

In this research, pure statistical techniques are used to generate answers to model questions. 1350 questions are entered into the system as training data samples to create a classifier for this system.

Different types of machine learning techniques are used to create a classifiers and the best is selected as classifier to classify the new sample.

In this research, initially, we have considered Gaussian Naïve Bayes [24], Random Forest [14], Decision Tree [24], Perceptron [17] and SVM [12]. Later, these classifiers were combined to create classifier ensembles. There are various ways to create ensemble classifiers. Combining the same type of classifiers with different parameters or combining the different type of classifiers with different parameters are some of the known ways of creating ensemble classifiers [25].

3.7 Answer solver module

The final answer to the question is generated by the answer solver module. The inputs for this component are the predicted formula/class (from the classifier) and the two numerical values derived from the question itself. The formula is predicted by the classifier and post to the answer solver component. After that, the first and second numerical values are derived from the question context and those are passed to this component for further calculation. At the end, the numeric values are plotted in the predicted formula and the final answer is generated.

Figure 3.2 shows the workflow of this module,

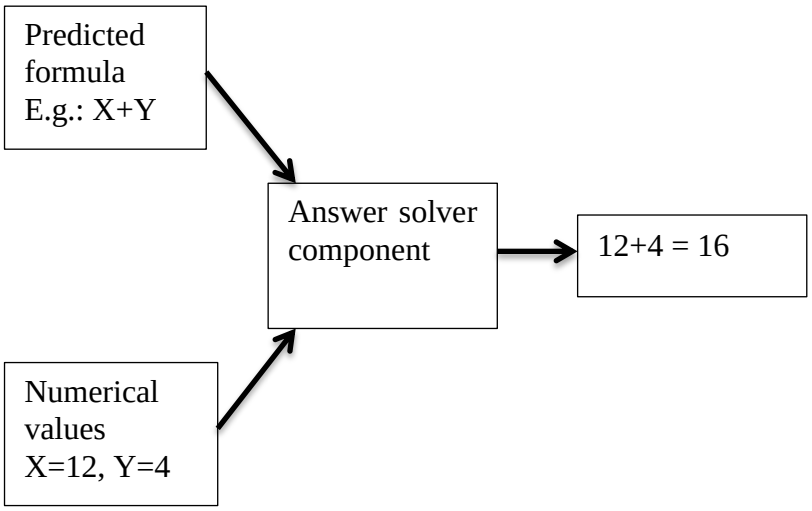

Figure 3.2: Workflow of answer solver module

Since this module is working totally independent from the other modules, the separation of concern can be easily maintained as well. Because of this new formula

types also can be easily integrated. Also, SymPy [41] kind of answer solver packages can be easily plugged-in with this module.

4 EVALUATION

4.1 Evaluation techniques

As mentioned in section 3.6.2, there are two different ways to create classifiers. They are individual classifiers and ensemble based classifiers. Evaluation has been done for all those classifiers to identify their performance.

For individual and ensemble-based classifiers, 10 fold cross-validation, and hold-out based evaluation methods are used to evaluate this system. Precision, recall, F-measure, and overall accuracy measurements are calculated for these evaluation methods. Data samples are shuffled two times before they were passed to the evaluator module to randomize the samples and ensure that the samples are well spread in the train feature vector.

In this research, we separately evaluated the accuracy of the system for different set of feature sets such as 10 features, 20 features, 30 features, 31 features and 36 features. And these features sets are selected according to the below scenarios shown in this Table 4.1.

Table 4.1: Selected feature sets for evaluation

Scenarios	Feature sets	Description
Scenario1	10 features	Amnueypornsakul and Bhat's [23] feature set
Scenario2	11 features	Roy et al [14] feature set
Scenario3	20 features	our own newly introduced feature set
Scenario4	30 features	Amnueypornsakul Bhat's [23] feature set + our own newly introduced feature set
Scenario5	31 features	Roy et al [14] feature set + our own newly introduced feature set
Scenario6	36 features	Amnueypornsakul Bhat's [23] feature set + Roy et al [14] feature set + our own feature set

Since 5 features from Roy et al's [17] feature set are overlapped with Amnueypornsakul and Bhat's [14] features, scenario 6 contains 36 features (10+ 20+11 – 5).
Apart from that, another experiment was carried out with SingleOP dataset [22] and precision, recall, F-measure and accuracy values are calculated. It is noted that no questions from this dataset are included in training set.

4.2 Evaluation results

4.2.1 Evaluation with different feature sets

The system is evaluated with different set of features with different classifiers such as Random forest, Gaussian NB, Decision Tree, and SVM, Perceptron as well as ensemble classifier to identify the model with the highest accuracy.

Accuracy, precision, recall, and f-measure are separately measured for all the feature sets.

The following figure 4.1 represents the comparison of accuracy for 10-fold cross validation and figure 4.2 represents the comparison of accuracy for hold-out evaluation. Both of these figures are used the values from table A.1 in the **APPENDIX A** section. Other than that precision, recall, f-measure values for different feature sets with different type of classifier algorithms are represented of Table A.3, Table A.4 and Table A.5 in **APPENDIX A** section.

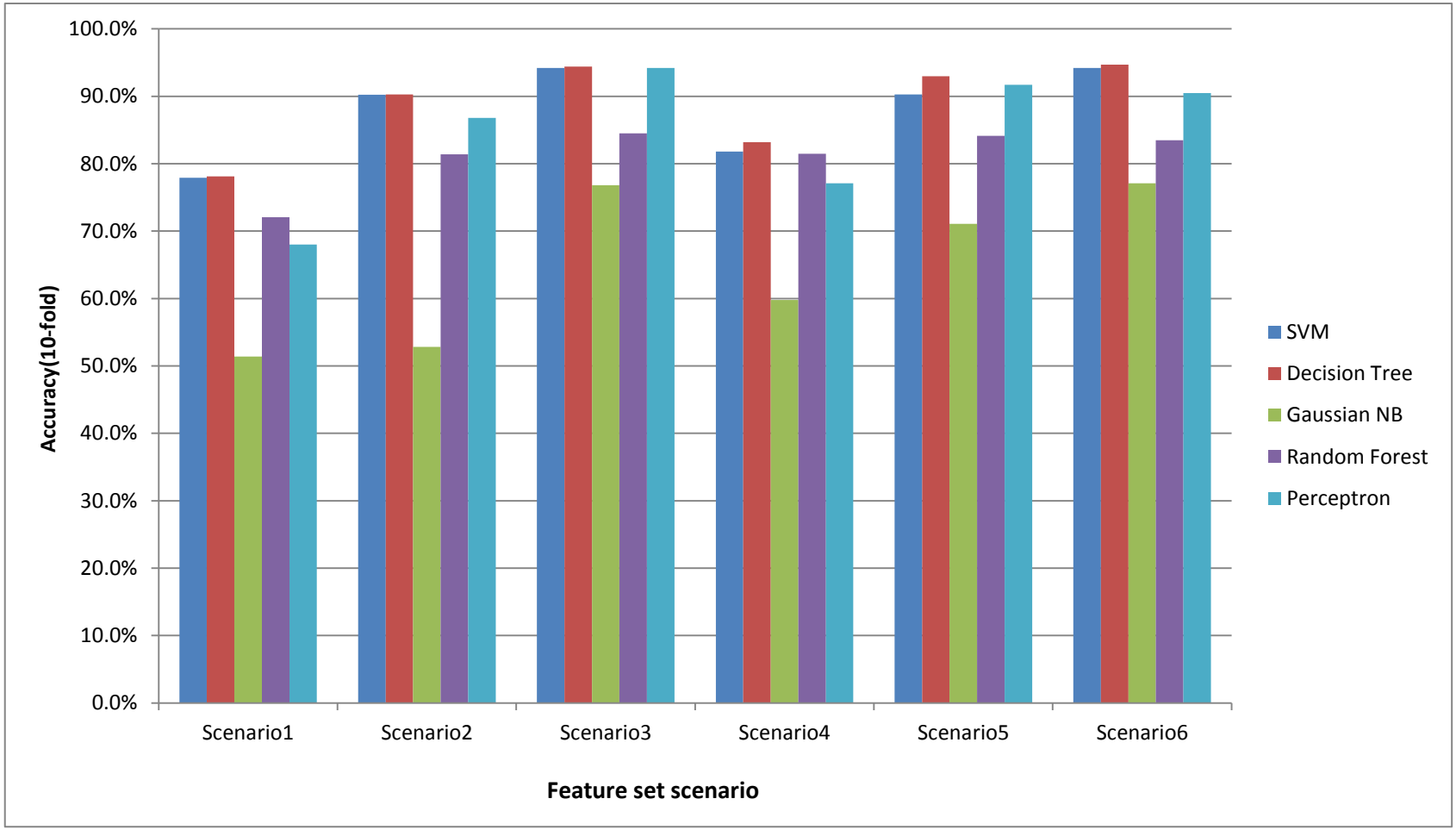

Figure 4.1:10-fold cross validation based accuracy for different features sets with classifiers

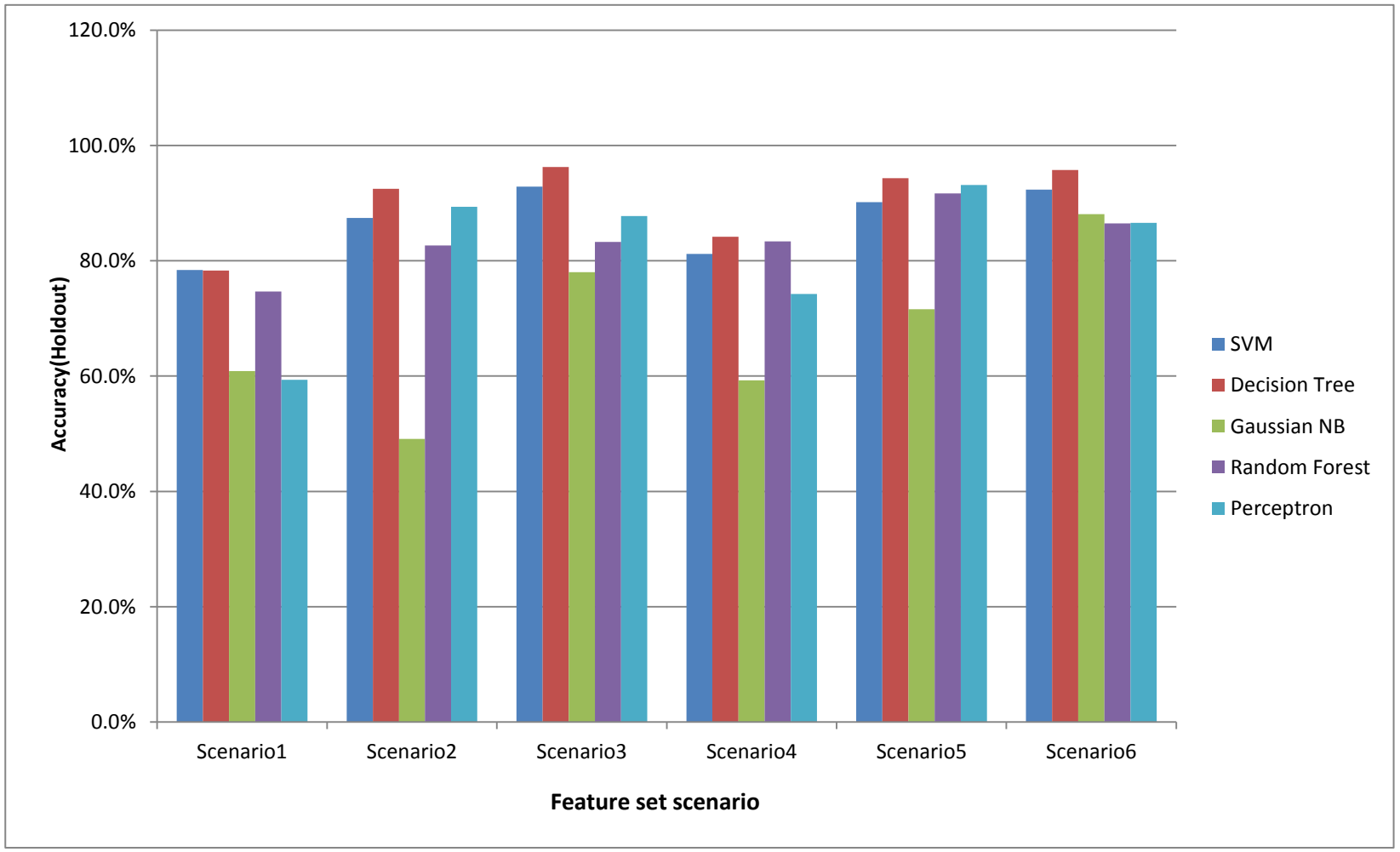

Figure 4.2: Hold-out accuracy for different features sets with classifiers

By considering the results shown in Figure 4.1 and Table A.1 for 10 fold-cross validation, SVM and decision tree show the best accuracy for 30 features. On the other hand, for 36 feature set, decision tree classifier shows the highest accuracy. It is noted that this 36 feature combination with decision tree algorithm still shows the highest accuracy than all the other classifiers and feature set combinations in the system. For Hold-out evaluation scenario also 30 and 36 feature sets with decision tree algorithm shows the highest accuracy than other feature sets and algorithms combinations. So by considering this result, the 36 feature set with decision tree algorithm is the best feature set in the system.

By comparing the accuracies with Amnueypornsakul and Bhat's [14] features and total 30 feature combinations (scenario 4 in table 4.1), the 30 feature combination shows a higher accuracy for random forest classifier.

Also, the 31 features set (scenario5 in table 4.1) shows a better accuracy than Roy et al's [17] features for perceptron classifier.

So by adding our newly introduced 20 features with Amnueypornsakul and Bhat's [14] features and Roy et al's[17] features, accuracy of the system is improved for the same type of classifiers used in Amnueypornsakul and Bhat's [14] and Roy et al's [17] approaches.

By looking at the precision results shown in Table A.3, almost all the formulae show good precision values for decision tree classifier with 36 feature set. And it is more than 0.8 out of 1. But it is noted that X-Y, Z+W formulae show better precision than other formulae. As shown in Table A.4, X-Y, Y-X formulae got the highest recall for the 36 feature set with decision tree algorithm. On the other hand, other formulae are also showing good recall for this feature set as well. Below figure 4.3 shows the F-measure value comparison for all the labels/formulae with decision tree classifier.

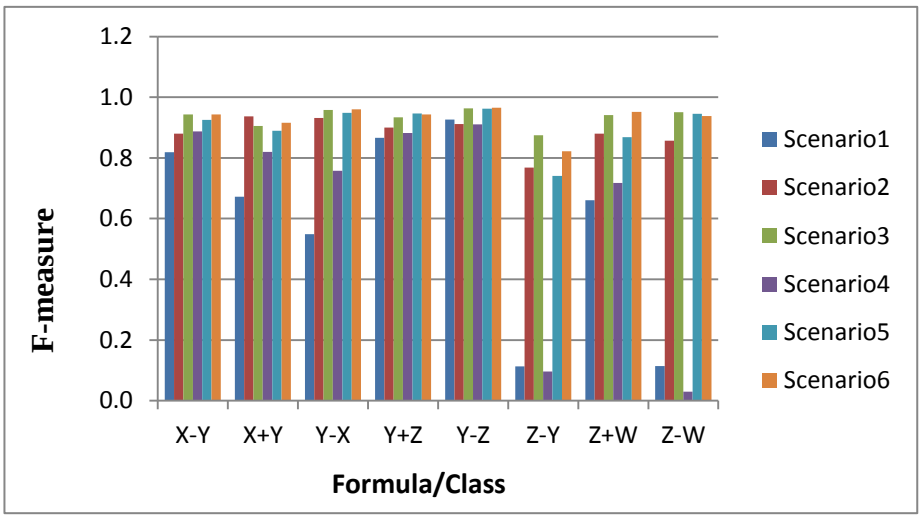

Figure 4.3: F-measure values for the labels with decision tree

The highest F-measure value was obtained for Y-X, Y-Z and Z+W formulae for 36 feature set with decision tree classifier as shown in figure 4.3 and table A.4. For other formulae also, accuracy is comparatively good for this 36 feature combination with the decision tree classifier.

4.2.2 Evaluation results for ensemble-based classifiers

As mentioned in section 3.6.2, two different evaluation criteria were used. First, the same type of classifiers is combined with different parameters. After that, different types of classifiers were combined with different parameters.

4.2.2.1. Ensemble-based evaluation results for the same classifier

Since decision tree classifier shows the highest accuracy than all other classifiers used in the system for 36 features, it is selected as the classifier for this initial experiment. In the first experiment, 3 decision tree classifiers with different parameters were combined to create an ensemble based classifier. Altogether 10 experiments are done to identify the best ensemble classifier. The different kind of parameter changes are done to min_samples_split, max_leaf_nodes, splitter, max_depth parameters of the decision tree classifier.

The parameters are explained as follows,
min_samples_split:- The minimum number of samples required for splitting an internal node. The default least value for this parameter is 2 units.

max_leaf_nodes:- Grow a tree with `max_leaf_nodes` in best-first fashion. The best nodes are defined as a relative reduction in impurity. If the parameter is ‘none’ then unlimited numbers of leaf nodes are considered. Also, the default value for this parameter is ’none’.

Splitter - The strategy used to choose the split at each node. Supported strategies are "best" to choose the best split and "random" to choose the best random split. Default value for this parameter is ‘best’.

max_depth:- The maximum depth of the tree. If the parameter is none, nodes are expanded until all leaves are pure or until all leaves contain less than min_samples_split samples. Default value for this parameter is ‘none’.

The experimented result is noted as in Table 4.2 and the parameter values are represented as follows,
Param1 - min_samples_split
Param2 - max_leaf_nodes
Param3 - splitter
Param4 - max_depth

Table 4.2: Ensemble classifier result for same type of classifiers (decision tree)

		Parameters				**Accuracy**	
	classifiers	**Param1**	**Param2**	**Param3**	**Param4**	**10-fold**	**Hold-out**
Exp1	*Classifier1*	2	None	best	None	94.51	**96.56**
	Classifier2	2	None	**'random'**	None		
	Classifier3	**10**	None	best	None		
Exp2	*Classifier1*	2	None	best	None	94.96	96.24
	Classifier2	2	None	random	None		
	Classifier3	**20**	None	best	None		
Exp3	*Classifier1*	2	None	best	None	94.51	95.6
	Classifier2	2	None	random	None		
	Classifier3	20	None	**random**	None		
Exp4	*Classifier1*	2	None	**best**	None	94.59	95.76
	Classifier2	2	None	**best**	None		
	Classifier3	20	None	**best**	None		
Exp5	*Classifier1*	2	None	best	None	94.67	96.08
	Classifier2	2	None	random	None		
	Classifier3	20	**100**	best	None		
Exp6	*Classifier1*	2	None	best	None	94.81	96.19
	Classifier2	2	None	random	None		
	Classifier3	20	**200**	best	None		
Exp7	*Classifier1*	2	None	best	None	95.03	**96.56**
	Classifier2	2	None	random	None		
	Classifier3	20	**400**	best	None		
Exp8	*Classifier1*	2	None	best	None	94.66	**96.56**
	Classifier2	2	None	random	None		
	Classifier3	20	400	best	**10**		
Exp9	*Classifier1*	2	None	best	None	94.66	96.0
	Classifier2	2	None	random	None		
	Classifier3	20	400	best	**1**		
Exp10	*Classifier1*	2	None	best	None	**95.18**	95.84
	Classifier2	2	None	random	**10**		
	Classifier3	20	400	best	**1**		

As shown in Table 4.2, in experiment 1, classifier 2 and classifier 3 are integrated with changed-parameter values and classifier 1 is kept with default parameter values. For classifier 2, the splitter parameter is set as 'random'. It is selected because that is the second option for that particular parameter. For classifier 3, the default least value of 'min_samples_split' parameter is slightly changed to a higher value than its default value in the initial experiment. In this case, the hold-out evaluation based accuracy showed the highest value for the given test data.

In experiment 2, both classifiers 1 and 2 are kept with their previous parameters. But the 'min_samples_split' parameter value of classifier 3 is increased compared to experiment 1. In this evaluation, 10-fold cross validation accuracy is slightly increased than the accuracy of experiment 1.

In experiment 3, the next changeable 'splitter' parameter value of classifier 3 is changed to identify the effect of this parameter on accuracy. But the other classifiers are kept with the same parameter values as in the Experiment 2. In this case, the

accuracy of the Hold-out evaluation and 10-fold cross validation are bit reduced than the previous experiment 2. So in experiment 4, 'splitter' parameter values of all the classifiers are now set to 'best'. In this case, the accuracy of the hold-out evaluation and 10-fold validation is slightly increased than in experiment 3. But still, these accuracy values are lower than in experiment 2. So 'splitter' parameter value of all classifiers is again set to the values in experiment 2.

In experiment 5, classifier 3 is changed with another changeable parameter 'max_leaf_nodes' with a limited number of nodes. But in default, its value is considered with unlimited nodes. So when a reduced number of leaf nodes the accuracy is increased than the accuracy got in experiment 4.

In experiment 5, the 'max_leaf_nodes' parameter value is increased in experiment 6. As a result of this, both 10-fold cross-validation and hold-out based accuracy are increased than in experiment 5. The 'max_leaf_nodes' parameter is even more increased in experiment 7. It is noted that the accuracy of both 10-fold validation and hold-out evaluation are increased than the accuracies got in previous experiments 1, 2, 3, 4, 5 and 6.

In experiment 8, classifiers 1 and 2 are kept with their parameters as in the experiment 7. But the previously unchanged 'max_depth' parameter is increased in classifier 3 from its default parameter. In this case, the accuracy is decreased than the previous experiment. In experiment 9, classifiers 1 and 2 are kept with their parameters as in the experiment 8. But 'max_depth' parameter of classifier 3 is decreased from its default parameter. In this experiment, accuracy of 10-fold validation remained the same and hold-out accuracy is slightly reduced. But still the accuracies of experiments 8 and 9 showed a lower accuracy than in experiment 7.

When considering the result in experiment 8 and 9, accuracy is not increased with the parameter changes of 'max_depth' in classifier 3. So in experiment 10, 'max_depth' parameter of the classifier 2 is increased from its default value. In this case, 10-fold validation based accuracy got the highest value than the previous experiments. On the other hand, hold-out accuracy also got good value as in the other experiments.
Based on the above experiments, we can say that almost all the ensemble based classifiers show nearly equal accuracy values. But it is noted that experiment 10 shows the highest accuracy value for 10-fold cross validation than other ensemble based classifiers. Experiment 7, 8 and 9 show the highest accuracy for Hold-Out based evaluation.

4.2.2.2. Ensemble based evaluation results for different types of classifiers

The other type of ensemble-based evaluation is done for the combination of different types of classifiers used in this research. GaussianNB, SVM, RandomForest, DecisionTree are the classifiers used to create an ensemble-based classifier in this different ensemble based approach. In this case, the accuracy of the ensemble

classifier is around **94.7%** for 10 fold cross-validation and the accuracy of Hold-out validation is around **96.1 %** with the default parameter values of these classifiers.

The 10 fold cross-validation based precision, recall and F-measure values for this ensemble based classifier evaluation is shown in Table 4.3,

Table 4.3: Measurements for ensemble classifier with default parameters

	Precision	**Recall**	**F-measure**
X-Y	0.9330	0.9534	0.9424
X+Y	0.9361	0.8902	0.9074
Y-X	0.9523	**0.9749**	0.9613
Y+Z	**0.9845**	0.8999	0.9395
Y-Z	0.9649	0.9686	**0.9661**
Z-Y	0.8083	0.7380	0.7336
Z+W	0.9215	0.9641	0.9402
Z-W	0.95	0.9416	0.9432

Based on the result from Table 4.3, the Y+Z formula got the highest precision. Also, Y-X formula shows the highest recall. Y-Z formula shows the highest f-measure value.

After this evaluation, the parameters of the classifiers are slightly changed to identify another accurate ensemble based classifier. The parameter changes are done as below for the classifiers mentioned above,

SVM: - degree=15, where degree is "Degree of the polynomial kernel function"
RandomForestClassifier: - max_depth=100, n_estimators=10, max_features=1, where max_depth is the maximum depth of the tree, n_estimators is the number of trees in the forest and max_features is the number of features to consider when looking for the best split.

DecisionTreeClassifier:- min_samples_split=2, min_samples_leaf=2, max_leaf_nodes=20, splitter='best'.

As a result of these parameters changes, the accuracy of this new ensemble classifier is increased as expected. It reported an accuracy of **95.48%** for 10-fold cross validation, as well as an accuracy of **95.84%** for Hold-Out based evaluation. Precision, recall and f-measure for this ensemble classifier are as in the Table 4.4. And these measurements are calculated using 10 fold cross-validation.

Table 4.4: Measurements for improved ensemble with changed parameters

	Precision	Recall	F-measure
X-Y	0.9296	0.9757	0.9515
X+Y	**0.9909**	0.8824	0.9303
Y-X	0.9420	**0.9825**	0.9597
Y+Z	0.9823	0.9071	0.9406
Y-Z	0.9671	0.9753	**0.9707**
Z-Y	0.8833	0.9266	0.8816
Z+W	0.9228	0.9593	0.9377
Z-W	0.9583	0.9675	0.9588

By considering the result shown in Table 4.4, X+Y formula got the highest precision. Also Y-X formula shows the highest recall. Y-Z formula shows highest F-measure value. And the below Figure 4.4 shows the comparison of F-measure values for the formulae with default parameters and optimized parameters.

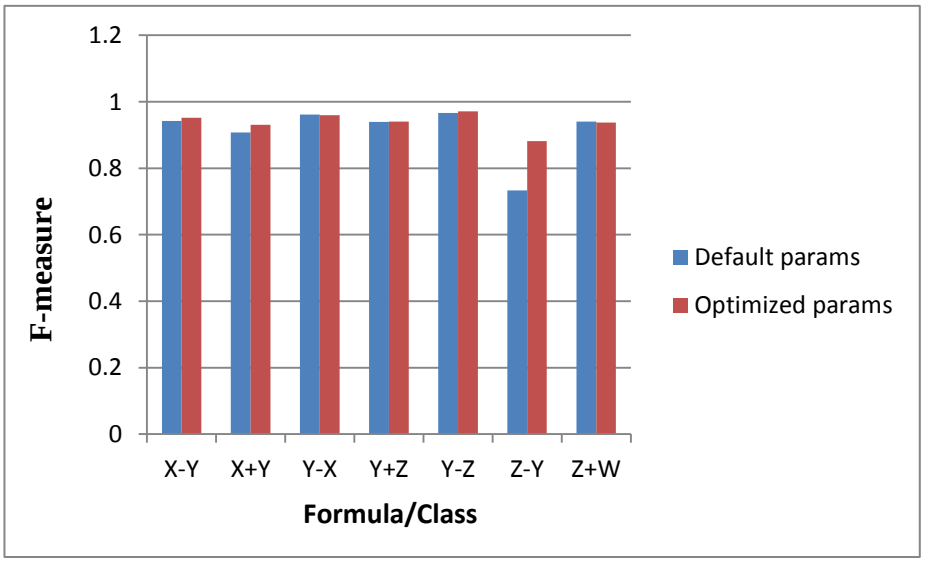

Figure 4.4: Comparison of f-measure values within default params and optimized params

Figure 4.4 shows that by doing some parameter change the f-measure values are increased for X-Y, X+Y, Y-Z and Z-Y formulae.

Discussion

From these ensemble based evaluations, we can see that the accuracy is almost similar in all of the experiments. But it is identified that by adjusting the parameters of the classifiers we can get good accurate ensemble classifiers.

4.2.3 Evaluation with SingleOP dataset [22]

The SingleOP dataset is an entirely new dataset used only for the purpose of evaluation. There are 200 test samples taken for this experiment. Accuracy, precision, recall and f-measure values are calculated for different feature sets with different classifiers. The below figure 4.5 shows the accuracy comparison of this evaluation. Also the accuracy, precision, recall, and f-measure, respectively represented by **APPENDIX A** section of table A.2, A.6, table A.7 and table A.8.

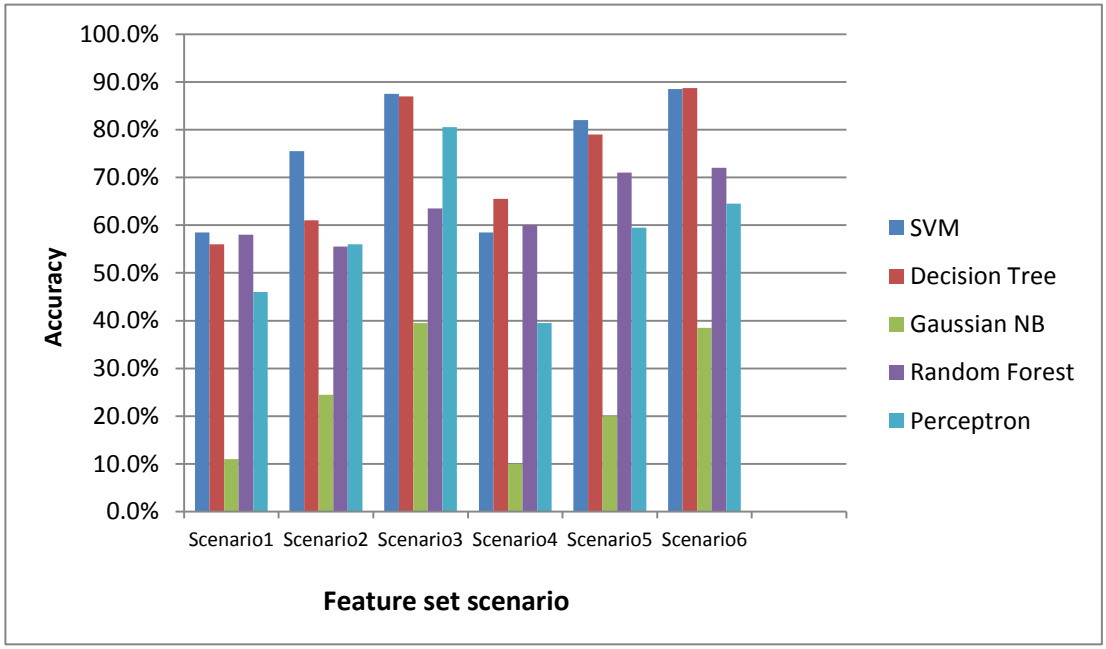

Figure 4.5: Single OP dataset accuracy of different combination of feature set with classifiers

By considering the accuracy results shown in figure 4.5 and table A.2 the SVM and decision tree show a good accuracy for 36 features. But it is noted that decision tree algorithm still shows the highest accuracy than other all classifier and feature set combinations in the system.

By considering the above figure 4.5, when compared with Amnueypornsakul and Bhat's [14] features (scenario1), 30 total feature combination (scenario4) shows the highest accuracy for random forest classifier. Also, 31 total feature set (scenario5) shows a better accuracy than Roy et al's [17] features (scenario2) for perceptron classifier. The below figures 4.6 and 4.7 are represents the comparison of f-measure values for these evaluations.

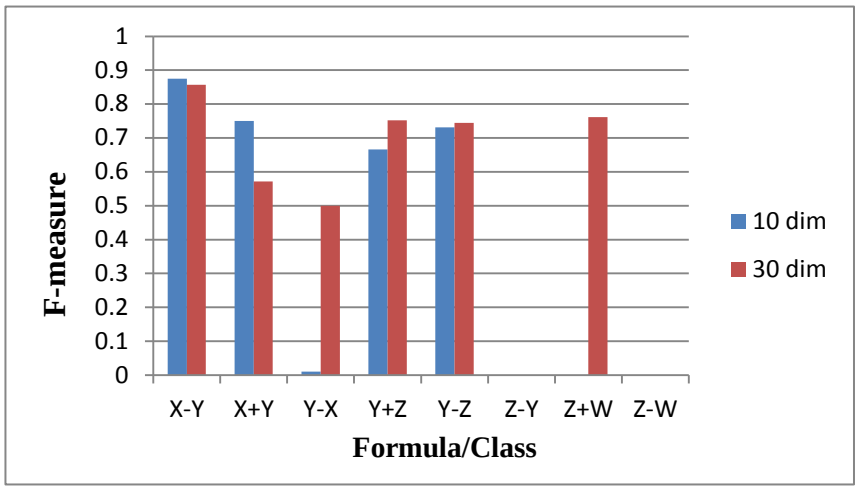

Figure 4.6: f-measure comparison of Amnueypornsakul and Bhat's [14] 10 features and 30 total features

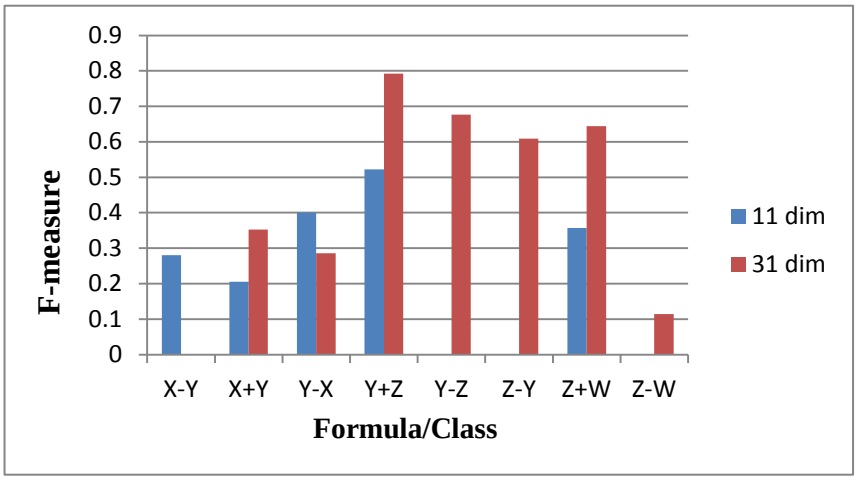

Figure 4.7: f-measure comparison of Roy's et al's [17] 11 features and 31 total features

The figures 4.6 and 4.7 show that show that accuracy is increased by our own additional features with other features derived from existing researches [14 and 17].

By looking at the precision results shown in Table A.6 in **APPENDIX A**, the formulae X-Y, X+Y, Y-X, Z-Y, Y-Z and Z+W show good precision values for decision tree classifier with 36 feature set. And the system shows less precision value for other 2 labels for the questions in SingleOP dataset.

As shown in Table A.7, X-Y, Y-X, Y+Z, Y-Z and Z+W formulae got the good recall for 36 feature set with decision tree algorithm. Out of these X-Y and Y-X formulae show the highest value than other labels. On the other hand, other formulae show less recall value for the questions in SingleOP dataset.

Good F-measure values were identified for X-Y, Y-X, Y-Z, Y+Z, Z-Y and Z+W formulae for 36 feature set with decision tree classifier as shown in Table A.8. But it

is noted that Y-X formula got the highest F-measure value for the questions in SingleOP dataset.

4.2.4 Evaluation results for SingleOP dataset with ensemble-based classifiers

First, the same type of highest accurate ensemble classifier which is mentioned in section **4.2.2.1** is used in this evaluation method. After that, the highest accurate ensemble classifier which is mentioned in section **4.2.2.2** is used to evaluate this SingleOP dataset.

4.2.4.1. SingleOP dataset evaluation for the same type ensemble classifier

As mentioned above the ensemble classifier which is mentioned in section **4.2.2.1** is used for this evaluation.

The accuracy is identified as **88.9%** for this evaluation. Precision, recall and f-measure for this ensemble classifier are as in the Table 4.5.

Table 4.5: Measurements for same type of ensemble classifier evaluation of SingleOP dataset

	Precision	**Recall**	**F-measure**
X-Y	0.9375	0.9375	0.9375
X+Y	**1.0**	0.5	0.6666
Y-X	**1.0**	**1.0**	**1.0**
Y+Z	0.6477	0.9344	0.7651
Y-Z	0.8484	0.4375	0.5773
Z-Y	0.9166	0.6875	0.7857
Z+W	0.9259	0.7142	0.8064
Z-W	0.1111	**1.0**	0.2

By considering the result shown in Table 4.7, X+Y, Y-X formulae got the highest precision. Also Y-X and Z-W formulae show the highest recall. Y-X formula shows highest F-measure value.

4.2.4.2. SingleOP dataset evaluation for the different types of ensemble classifiers

As mentioned earlier the ensemble classifier which is mentioned in section **4.2.2.2** is used for this evaluation.

The accuracy is identified as **89.2%** from this evaluation. Precision, recall and f-measure for this ensemble classifier are as in the Table 4.6.

Table 4.6: Measurements for different type of ensemble classifier evaluation of SingleOP dataset

	Precision	**Recall**	**F-measure**
X-Y	0.8461	0.6875	0.7586
X+Y	0.4444	**1.0**	0.6153
Y-X	0.0	0.0	0.0
Y+Z	0.9272	0.8360	0.8793
Y-Z	0.875	0.9843	**0.9264**
Z-Y	**1.0**	0.6875	0.8148
Z+W	0.8205	0.9142	0.8648
Z-W	0.2	**1.0**	0.12

Based on the result from Table 4.6, the Z-Y formula got the highest precision. Also, X+Y and Z-W formulae show the highest recall. Y-Z formula shows the highest f-measure value.

4.2.5 Discussion about the overall evaluation results

From all of the above experiments, the 10 fold cross-validation based evaluation results showed that the overall accuracy is very similar between the Decision Tree classifier and SVM classifier, as well as the ensemble based classifier. But it is to be noted that the Decision Tree-based classifier still shows the highest value of accuracy for the individual based classifier method. From our evaluation, It is identified that random forest algorithm shows a lower accuracy than decision tree classifier with its default configuration. Because in our feature set, some features are only related to one specific question type. So when the random forest algorithm randomly selects the features and builds the model, the overall accuracy of that random forest classifier gets lower than decision tree algorithm.

Based on the ensemble based evaluations, the highest accuracy is achieved when multiple types of classifiers are used for the ensemble based classifier approach. But it is noted that the values of all ensemble classifiers reported nearly equal values. As for multiple types of ensemble classifiers, all the 8 formulae classes showed particularly good precision, recall, and F-measure values. But it is noted that the precision is very high for X+Y type of formula. Also, Y-X type of formula showed the highest recall than other formula types. At the same time, the Y-Z type of formula showed the highest f-measure value in this evaluation. But as mentioned earlier, the other types of formula also showed good precision, recall, and F-measure values and they are between 0.85 and 1.

When our results are compared with Roy et al's [17] features; our approach shows better accuracy. While using only Amnueypornsakul's [14] features, the system shows lower accuracy. But on the other hand, when using Amnueypornsakul's [14] features with our additional features, the system shows a higher accuracy.

Also, from the SingleOP dataset evaluation, it shows that for around 88.7 % percentage of questions the formula is correctly predicted by the system. It is to be reminded that as mentioned earlier there are no questions from this corpus was in the training set. As a result, accuracy is comparatively low when compared with the evaluation of previous data set. However, still, this evaluation result shows that our approach is better than Roy's et al [17] approach and Amnueypornsakul & Bhat [14] approach for this SingleOP corpus.

5 CONCLUSION AND FUTURE WORK

5.1 Conclusion

Automatic answer generation is a trending research in Natural Language Processing. This avoids the need for manual answer creation by teachers. This thesis describes the process to automate the manual answer generation of simple linear algebra-based mathematics questions. Specifically, I consider questions with addition or subtraction on two known and one unknown variables. The statistical approach describes in this report is better than the past statistical approaches. I followed a statistical approach because it did not need manual rules/inferences or ontologies. And I experimented with available classifiers and different feature sets for this research. Also, I introduce new features and ensembles to achieve a better accuracy. Then I have experimented my work with two different data sets.

Finally, I identified that my features are more robust than what has been used in previous approaches.

5.2 Future Work

In future, this system can be adapted to new question types such as simple multiplication and division question types. To do that, we have to just derive some features of the new question types and train with appropriate samples in our system. In fact, initially this system supported only "change and compare" type of linear equation based question types. Later only "whole-part" type of questions was integrated with this system. By adding two new features, the "whole-part" type of question type is simply integrated into this system. In future, the 'SymPy' kind of open source modules can be easily integrated with our answer solver component for the answer generation of new formula types.

This research includes 8 classes/labels to represent the formula types that we considered in this research. But still, we can reduce the classes/labels by considering only the operation type that we perform in a question. For example, the formula '**X+Y**' and the formula '**Z+W**' are performing the same type of operation which is "addition" operation. But this operation is performed for different question types. Formula '**X+Y**' is associated with "Compare Type" and formula '**Z+W**' is associated with the Whole-part type of questions. These classes are separately defined to get their precision, recall and f-measure values for different formula types of all three question types considered in this research. In future, the labels can be merged as one label/class and reduced in an appropriate manner.

Currently, this research includes a maximum of two known variables and one unknown variable based linear equation based questions to generate answers. In future, it can be extended to questions with three or more known variables and some unknown variables based linear equation questions.

The assumption we considered in this research is the quantity changes happen only for a particular entity/variety in a question. That means that the change of quantity happens in the same entity variable of a question.

For example, in the question "*Dan picked 9 limes and gave Sara 4 of the limes. How many lines does Dan have now?"* In this case, we assume that the "*lime*" is a noun entity and the quantities 9 and 4 are only related to that "*lime*" entity. But in some cases, this assumption will not work when the second noun entity is referred by some other synonymous word. So in the future, this issue can be resolved by maintaining a list of synonym words for a word.
The other assumption made in this research is related to the co-reference based feature implementation. This system does not know the difference between male and female names. So there are chances of mapping the wrong personal pronoun type with a wrong name/proper noun. Also, we assume that the personal pronouns in the second and third sentences are only related to the proper noun in the first sentence in the question. This assumption is made due to the question pattern that we considered in this research. So in the future, this assumption may have to be avoided or changed for some other new question types.

In order to identify the positive and negative impact of a sentence in the question, a list of positive and negative words is defined in the system for training purpose. But in the future, this positive/negative impact can be predicted by using a separate classifier so that the system can automatically learn the positive and negative words based on the context. Also in order to identify the collective nouns in a question a list of collective-noun and its related nouns are maintained in the system and used whenever it is needed. But in future, this collective noun-noun relationship can be created using an ontology based relationship map with the help of considerable amount of training data.

REFERENCES

[1] Grouws, D. (1992). Handbook of Research on Mathematics Teaching and Learning. New York: Maxwell Macmillan International.

[2] Robert Sweetland (1992). “Types of Addition and Subtraction Problems Examples with whole numbers”, Available: http://www.homeofbob.com/math/numVluOp/ wholeNum/ addSub/ adSubTypsChrt.html

[3] Mark Hopkins, Cristian Petrescu-Prahova, Roie Levin, Ronan Le Bras, Alvaro Herrasti, and Vidur Joshi, "Beyond Sentential Semantic Parsing: Tackling the Math SAT with a Cascade of Tree Transducers", Conference on Empirical Methods in Natural Language Processing, September 2017, pp 806-815

[4] Mitra, Arindam and Baral, Chitta. "Learning to use formulas to solve simple arithmetic problems", 54th Annual Meeting of the Association for Computational Linguistics, August 2016, pp 2144-2153

[5] Margaret Rouse (2005), “Ontology”, Available: http://whatis.techtarget.com/definition/ontology.

[6] Kyle Morton and Yanzhen Qu, “A Novel Framework for Math Word Problem Solving by International Journal of Information and Education Technology” Vol. 3 No. 1 February 2013, pp 88-93.

[7] Shuming, ShiYuehui Wang, Chin, Yew Lin, Xiaojiang Liu and Yong Rui, “Automatically Solving Number Word Problems by Semantic Parsing and Reasoning”, Conference on Empirical Methods on Natural Language Processing, Sep 2015 pp 1132-1142.

[8] MD Leblanc, “Text Integration and Mathematical Connections A Computer Model of Arithmetic Word Problem Solving”, Cognitive Science 20 vol no. 3, July 1996, pp 357-407

[9] Takuya Matsuzaki, “The Most Uncreative Examinee: A First Step toward Wide Coverage Natural Language Math Problem Solving”, AAAI, July 2014, pp 1098-1104

[10] Lipu Zhou Shuaixiang Dai and Liwei Chen, “Learn to Solve Algebra Word Problems Using Quadratic Programming” *EMNLP* The Association for Computational Linguistics, Sep 2105, pp 817-822

[11] Mohammad Javad Hosseini1, Hannaneh Hajishirzi1, Oren Etzioni2, and Nate Kushman, “Learning to Solve Arithmetic Word Problems with Verb Categorization”, Conference on Empirical Methods on Natural Language Processing, October 2014, pp 523-533.

[12] Yi-Chung Lin, Chao-Chun Liang, Kuang-Yi Hsu,bChien-Tsung Huang, Shen-Yun Miao, Wei-Yun Ma, Lun-Wei Ku, Churn-Jung Liau, and Keh-Yih Su, “Designing a Tag-Based Statistical Math Word Problem Solver with Reasoning and Explanation” Vol. 20 No. 2, December 2015.

[13] Matsuzaki, Takuya, Ito, Takumi, Iwane, Hidenao, Anai, Hirokazu and Arai, Noriko H."Semantic Parsing of Pre-university Math Problems", ACL (1), August 2017, pp 2131-2141.

[14] B. Amnueypornsakul, and S. Bhat, “Machine-Guided Solution to Mathematical Word Problems”, The PACLIC 28 Organizing Committee and

PACLIC Steering Committee / ACL / Department of Linguistics, Faculty of Arts, Chulalongkorn University, 2014, pp 111-119

[15] Yan Wang, Xiaojiang Liu, Shuming Shi, "Deep Neural Solver for Math Word Problems", Conference on Empirical Methods in Natural Language Processing, September 2017, pp 856–865

[16] Ling, Wang, Yogatama, Dani, Dyer, Chris and BlunsomPhil, "Program Induction by Rationale Generation: Learning to Solve and Explain Algebraic Word Problems", ACL (1), 2017, pp 158-167

[17] Roy S. I. Vieira T. J. H. and Roth D. I, "Reasoning about Quantities in Natural Language", Transactions of the Association for Computational Linguistics, June 2015, pp 1-13

[18] Dellarosa, "A computer simulation of children's arithmetic word-problem solving", Behavior Research Methods, Instruments, & Computers 18(2), March 1989, pp 147-154

[19] Mukherjee U. Garain, "A review of methods for automatic understanding of natural language mathematical problems" Artif Intell Rev, 2008, pp 93-122

[20] Nate Kushman, Yoav Artzi, Luke Zettlemoyer, and Regina Barzilay, "Learning to Automatically Solve Algebra Word Problems", 52nd Annual Meeting of the Association for Computational Linguistics, December 2014, pp 271-281.

[21] R Koncel-Kedziorski, "MaWPS: A Math Word Problem Repository", HLT-NAACL, June 201, pp 1152-1157

[22] "MaWPS: A Math Word Problem Repository(2016)", Available: http://lang.ee.washington.edu/MAWPS/datasets/SingleOp.json

[23] Furey and Edward. "Numbers to Words Converter", Available: https://www.calculatorsoup.com

[24] Dr. Jason Brownlee (2016). "Naive Bayes for Machine Learning", Available: https://machinelearningmastery.com/ naive-bayes-for- machine-learning/

[25] Lior Rokach, "Ensemble-based classifiers", Artif. Intell. Rev. 33 (1-2),Feb 2010, pp 1-39

[26] Y.-M. Yeh et al "Research on Reasoning and Modeling of Solving - Mathematics Situation Word", Ninth IEEE International Symposium, December 2007, pp 35-41.

[27] M. Hearst, "Automatic Acquisition of Hyponyms from Large Text Corpora" In Fourteenth International Conference on Computational Linguistics, Nantes, France, 1992, pp 539-545

[28] Bussaba Amnueypornsakul and Suma Bhat, "Machine-Guided Solution to Mathematical Word Problems", Pacific Asia Conference on Language, Information and Computation, 2014, pp 111-119.

[29] Erik de corte , Lieven verschaffel, and Brian greer, "Connecting mathematics problem solving to the real world" International Conference on Mathematics Education into the 21st Century, November 2000, pp 66-73.

[30] Wanintorn Supap Kanlaya Naruedomkul and Nick Cercone, "International journal of Automatic Learning Guide for Mathematical Word Problem", Volume 17, Issue 11, 2011, pp 509-524

[31] Aparna Lalingkar, Chandrashekar Ramnathan and Srinivasan Ramani, “MONTO: A Machine-Readable Ontology for Teaching Word Problems in Mathematics” Volume 18, November 2015, pp 197-213.
[32] Chao-Chun Liang, Kuang-Yi Hsu, Chien-Tsung Huang, Chung-Min Li, Shen-Yu Miao,and Keh-Yih Su, “A Tag-based English Math Word Problem Solver With Understanding, Reasoning and Explanation”, IJCAI, June 2016, pp 4254-4255
[33] Christian Liguda and Thies Pfeiffer, “A question answer system for math word problems” First International Workshop on Algorithmic Intelligence, January 2011, pp 1-11
[34] Noam Chomsky, “Three Models for the description of language”, IRE Transactions on Information Theory 2, 1956, pp 113-124
[35] Jurafsky and Martin, “Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition”, Prentice Hall PTR, 2000, pp 1-939
[36] Eren Golge (2012). “Neural networks over decision trees”, Available: https://www.quora.com/What-are-some-advantages-of-using-neural-networks-over-decision-trees
[37] Dan Benyamin (2012). “A Gentle Introduction to Random Forests, Ensembles, and Performance Metrics in a Commercial System”,Available: http://blog.citizennet.com/blog/2012 /11/10/ random-forests-ensembles-and-performance-metrics
[38] Noel Bambrick, AYLIEN. (2016). “Support Vector Machines: A Simple Explanation”,Available: https://www.kdnuggets.com/2016/07/support-vector-machines-simple-explanation.html
[39] "Collective Nouns - A guide to collective nouns", Available: http://www.collectivenouns.biz/list-of-collective-nouns/collective-nouns-people/
[40] "word2number 1.1", Available: https://pypi.org/project/word2number/
[41] "Sympy", Available: https://github.com/sympy/sympy/releases

6 APPENDIX A:

1.Algorithm for word to number paraphrase parser

```
from word2number import w2n
def convert_to_number(sentence):
    num_start_pos = 0
    num_end_pos = 0
    formatted_string = ''
    sentence = sentence.translate({ord(x): ' ' for x in [',', '-']})
    words = sentence.split()
    for idx, word in enumerate(words):
        try:
            current = w2n.word_to_num(word)
            if current:
                current_index = idx
                if num_start_pos == 0:
                    num_start_pos = current_index
                next_word = words[current_index + 1]
                if next_word:
                    # if next_word is 'and' or 'point':
                    #     continue
                    next_current = w2n.word_to_num(next_word)
                    if next_current:
                        num_end_pos = current_index + 1
        except:
            is_num = False
            if num_start_pos > 0 and num_end_pos == 0:
                num_end_pos = num_start_pos

            if num_start_pos > 0 and num_end_pos > 0:
                num_string = ' '.join(words[num_start_pos:num_end_pos+1])
                next_current = w2n.word_to_num(num_string)
                if next_current:
                    is_num = True
                    formatted_string += ' ' + str(next_current)
                    num_start_pos = 0
                    num_end_pos = 0
            if not is_num:
                formatted_string += ' ' + word

    print(formatted_string.lstrip())
    return formatted_string.lstrip()


def formatted_string(test_data):
    numbers = [int(s) for s in test_data.split() if s.isdigit()]
    if len(numbers) == 2:
        return test_data
    else:
        return convert_to_number(test_data)
```

Table A.1: Accuracy for different combination of features with different classifiers

Classifier	Feature set	10-fold accuracy (%)	Hold-out accuracy (%)
SVM	**Scenario1**	77.9	78.4
	Scenario2	90.22	87.44
	Scenario3	**94.2**	**92.88**
	Scenario4	81.8	81.2
	Scenario5	90.29	90.16
	Scenario6	**94.2**	92.32
Decision Tree	**Scenario1**	78.14	78.32
	Scenario2	90.29	92.47
	Scenario3	94.4	**96.24**
	Scenario4	83.2	84.16
	Scenario5	92.96	94.32
	Scenario6	**94.7**	95.76
Gaussian NB	**Scenario1**	51.4	60.88
	Scenario2	52.81	49.12
	Scenario3	76.8	78.0
	Scenario4	59.8	59.28
	Scenario5	71.1	71.6
	Scenario6	**77.1**	**88.08**
Random Forest	**Scenario1**	72.07	74.64
	Scenario2	81.4	82.64
	Scenario3	**84.5**	83.28
	Scenario4	81.48	83.36
	Scenario5	84.14	**91.67**
	Scenario6	83.5	86.48
Perceptron	**Scenario1**	67.99	59.36
	Scenario2	86.8	89.36
	Scenario3	**94.2**	87.76
	Scenario4	77.1	74.24
	Scenario5	91.7	**93.12**
	Scenario6	90.5	86.56

Table A.2: Single OP dataset accuracy of different combination of feature set with classifiers

Classifier	Feature set	Accuracy (%)
SVM	**Scenario1**	58.5
	Scenario2	75.5
	Scenario3	87.5
	Scenario4	58.5
	Scenario5	82
	Scenario6	**88.5**
Decision Tree	**Scenario1**	56
	Scenario2	61
	Scenario3	87
	Scenario4	65.5
	Scenario5	79
	Scenario6	**88.7**
Gaussian NB	**Scenario1**	11
	Scenario2	24.5
	Scenario3	**39.5**
	Scenario4	10
	Scenario5	20
	Scenario6	38.5
Random Forest	**Scenario1**	58
	Scenario2	55.5
	Scenario3	63.5
	Scenario4	60
	Scenario5	71
	Scenario6	**72**
Perceptron	**Scenario1**	46
	Scenario2	56
	Scenario3	**80.5**
	Scenario4	39.5
	Scenario5	59.5
	Scenario6	64.5

2.

Table A.3: Precision of classes (10 fold cross validation)

	SVM						Decision Tree						Gaussian NB					
	10 dim	**20 dim**	**30 dim**	**11 dim**	**31 dim**	**36 dim**	**10 dim**	**20 dim**	**30 dim**	**11 dim**	**31 dim**	**36 dim**	**10 dim**	**20 dim**	**30 dim**	**11 dim**	**31 dim**	**36 dim**
X-Y	0.7671	**0.9571**	0.9280	0.8226	0.8972	0.9324	0.7777	0.9000	**0.9394**	0.8337	0.9052	0.9297	0.6	0.0666	0.9173	1.0	0.8796	0.8884
X+Y	0.5867	**0.992**	0.9857	0.7757	0.9646	0.9660	0.5842	**1.0**	0.9148	0.8526	0.9114	0.9483	0.3092	0.4614	0.6660	0.2977	0.5904	0.6694
Y-X	1.0	0.9273	**0.9444**	0.6661	0.9384	0.9346	**0.9909**	0.9113	0.9456	0.7392	0.9402	0.9499	0.2927	0.3504	0.4286	0.8761	0.4923	0.4777
Y+Z	0.8003	0.9214	**0.9673**	0.9395	0.9550	0.9631	0.8138	0.9216	0.9554	0.9221	**0.9632**	0.9585	**1.0**	0.7249	0.9161	**1.0**	**0.9444**	0.925
Y-Z	**0.9812**	0.8475	0.9431	0.8506	0.9132	0.9448	**0.9823**	0.8898	0.9688	0.8529	0.9732	0.9737	0.9949	0.9192	**0.9945**	0.9078	0.8928	**0.9929**
Z-Y	0.1224	0.9	0.8666	0.0	0.45	**0.8916**	0.1124	0.8149	**0.8933**	0.0585	0.7833	0.8633	0.0066	0.4129	0.5708	0.0595	0.5838	0.6699
Z+W	0.5349	0.9100	0.8956	0.8034	0.7975	**0.9329**	0.5363	0.8823	0.9300	0.8866	0.8623	**0.9558**	0.0625	**0.9329**	0.7851	0.8796	0.7044	0.8054
Z-W	0.1125	**1.0**	**1.0**	0.1014	0.9	**1.0**	0.1352	0.8633	0.9361	0.1752	**0.95**	0.9282	0.3206	0.4021	0.6303	0.2087	0.4400	0.6294

	Random Forest						Perceptron					
	10 dim	**20 dim**	**30 dim**	**11 dim**	**31 dim**	**36 dim**	**10 dim**	**20 dim**	**30 dim**	**11 dim**	**31 dim**	**36 dim**
X-Y	0.7636	0.9268	0.8557	0.8198	0.8756	0.8615	0.6763	0.8169	0.9478	0.8334	0.9048	0.8802
X+Y	0.5156	0.9909	0.9623	0.8054	**0.9716**	**0.9909**	0.5489	**0.9554**	0.9574	0.6617	0.9305	0.8469
Y-X	**0.9875**	0.8773	0.9539	0.6927	0.9221	0.9603	0.4034	0.9055	0.9479	0.7503	0.9491	0.9403
Y+Z	0.7987	0.9677	**0.975**	**0.9443**	0.8866	0.9584	0.9260	0.8975	0.9754	**0.9228**	0.9369	0.9512
Y-Z	0.8410	0.6888	0.7642	0.8218	0.7956	0.7656	**0.9745**	0.9020	0.9142	0.8200	0.9317	0.9624
Z-Y	0.0145	0.075	0.2374	0.0	0.0447	0.061	0.1127	0.8333	0.9	0.1392	0.7466	0.9375
Z+W	0.5280	0.9413	0.8903	0.8375	0.7977	0.8089	0.4575	0.9299	0.9598	0.8071	0.9231	0.8594
Z-W	0.7421	**1.0**	0.9	0.2147	0.8	0.8	0.1734	0.6597	**1.0**	0.2169	**0.9916**	**1.0**

Table A.4: Recall of classes (10 fold cross validation)

	SVM						Decision Tree						Gaussian NB					
	10 dim	**20 dim**	**30 dim**	**11 dim**	**31 dim**	**36 dim**	**10 dim**	**20 dim**	**30 dim**	**11 dim**	**31 dim**	**36 dim**	**10 dim**	**20 dim**	**30 dim**	**11 dim**	**31 dim**	**36 dim**
X-Y	0.8680	0.8188	0.9702	0.9318	0.9298	**0.9610**	0.8715	0.8662	0.9494	0.9522	0.9481	**0.9591**	0.0408	0.0476	0.3359	0.3048	0.3612	0.3870
X+Y	0.8173	0.8861	**0.8838**	0.8167	0.8603	0.8708	0.8115	0.8867	**0.9055**	0.803	0.8870	0.8980	1.0	0.9338	0.9916	0.9505	0.9198	0.99
Y-X	0.3848	0.9520	**0.9779**	0.8541	0.9353	0.9735	0.3966	0.9618	**0.9722**	0.8239	0.9644	**0.9724**	0.4883	0.9857	0.9916	0.5326	0.9822	0.9866
Y+Z	**0.9381**	0.8847	0.8891	0.8100	0.9144	0.9031	0.9341	0.8901	0.9173	0.8552	**0.9347**	0.9334	0.6155	0.1964	0.7155	0.6170	0.6399	0.6710
Y-Z	0.8814	0.9689	0.9797	0.9721	0.9636	**0.9794**	0.8792	0.9360	0.9580	0.9766	0.9518	**0.9588**	0.8494	0.5973	0.8741	0.8374	0.8769	0.8792
Z-Y	0.112	**0.9157**	0.7666	0.0	0.4	0.9133	0.2115	0.7583	**0.8966**	0.3	0.7333	0.8300	0.05	0.93	0.8416	0.175	0.8133	0.9099
Z+W	0.8644	0.8775	**0.9593**	0.5845	0.8457	0.9435	0.8690	0.8857	**0.9558**	0.6133	0.8883	0.9502	0.0555	0.6280	0.8217	0.4109	0.4739	0.8339
Z-W	0.1224	**0.8666**	0.7464	0.02	0.6690	0.6866	0.1421	0.8583	**0.9722**	0.0147	0.9523	0.96	0.6416	0.9	1.0	0.5	1.0	1.0

	Random Forest						Perceptron					
	10 dim	**20 dim**	**30 dim**	**11 dim**	**31 dim**	**36 dim**	**10 dim**	**20 dim**	**30 dim**	**11 dim**	**31 dim**	**36 dim**
X-Y	0.8399	0.7406	0.9180	0.9115	0.8457	0.9115	0.4785	0.8404	0.9128	0.8332	0.9335	0.9093
X+Y	0.7276	0.8716	0.9111	0.8114	0.8145	0.7418	0.8179	0.8956	0.9308	0.7926	0.9116	0.7928
Y-X	0.4224	0.7816	0.7733	0.8425	0.8847	0.8968	0.2892	0.8462	0.9780	0.8420	0.9509	0.8695
Y+Z	0.9299	0.6360	0.7632	0.8254	0.8094	0.8485	0.7468	0.9005	0.9304	0.7617	0.9343	0.8931
Y-Z	0.9416	0.9791	0.9742	0.9759	0.9729	0.9821	0.8795	0.8995	0.9918	0.9778	0.9568	0.9511
Z-Y	0.015	0.1	0.2346	0.01	0.147	0.1745		0.7766	0.8166	0.35	0.6799	0.8800
Z+W	0.5221	0.7165	0.6598	0.5778	0.6730	0.5991	0.7715	0.8403	0.9035	0.3420	0.7651	0.8697
Z-W	0.5124	0.8638	0.6	0.021	0.6833	0.4640	0.3433	0.8625	0.8733	0.042	0.9166	0.8800

Table A.5: f-measure of classes (10 fold cross validation)

	SVM						Decision Tree						Gaussian NB					
	10 dim	**20 dim**	**30 dim**	**11 dim**	**31 dim**	**36 dim**	**10 dim**	**20 dim**	**30 dim**	**11 dim**	**31 dim**	**36 dim**	**10 dim**	**20 dim**	**30 dim**	**11 dim**	**31 dim**	**36 dim**
X-Y	0.8120	0.8811	0.9479	0.8719	0.9121	0.9450	0.8186	0.8800	0.9433	0.8878	0.9253	0.9430	0.0760	0.0555	0.4863	0.4552	0.5026	0.5201
X+Y	0.6662	0.9301	0.9310	0.7801	0.9072	0.9094	0.6726	**0.9376**	0.9056	0.8196	0.8900	0.9163	0.4682	0.6141	0.7939	0.4456	0.7174	0.7929
Y-X	0.5496	**0.9374**	0.9599	0.7419	0.9352	0.9524	0.5490	0.9318	0.9580	0.7582	0.9486	0.9602	0.3595	0.4831	0.5934	0.6434	0.6393	0.6249
Y+Z	0.8622	0.8970	0.9256	0.8609	0.9321	0.9305	0.8662	0.9003	0.9341	0.8820	0.9463	0.9432	0.7570	0.3040	0.7950	0.7564	0.7559	0.7642
Y-Z	**0.9280**	0.9034	**0.9607**	**0.9061**	**0.9371**	**0.9613**	**0.9270**	0.9116	**0.9631**	**0.9104**	**0.9623**	**0.9657**	**0.9158**	0.7110	**0.9299**	**0.8699**	**0.8840**	**0.9312**
Z-Y	0.1214	0.8859	0.7980	0.017	0.4200	0.8830	0.1128	0.7679	0.8753	0.0959	0.7409	0.8220	0.0117	0.5523	0.6628	0.0831	0.6642	0.7368
Z+W	0.6595	0.8916	0.9249	0.6599	0.8113	0.9358	0.6604	0.8803	0.9410	0.7175	0.8687	0.9523	0.0588	**0.7437**	0.8007	0.5461	0.5614	0.8101
Z-W	0.1120	0.9180	0.8273	0.021	0.7523	0.7961	0.1142	0.8564	0.9505	0.03	0.9457	0.9380	0.4072	0.5484	0.7638	0.2815	0.6019	0.7610

	Random Forest						Perceptron					
	10 dim	**20 dim**	**30 dim**	**11 dim**	**31 dim**	**36 dim**	**10 dim**	**20 dim**	**30 dim**	**11 dim**	**31 dim**	**36 dim**
X-Y	0.7941	0.8190	0.8800	0.8590	0.8544	0.8722	0.5293	0.8219	0.9281	0.8124	0.9158	0.8777
X+Y	0.5963	**0.9236**	**0.9336**	0.8018	0.8797	0.8185	0.6440	**0.9188**	0.9398	0.7080	0.9140	0.8091
Y-X	0.5642	0.8176	0.8406	0.7387	**0.8993**	**0.9199**	0.3269	0.8643	**0.9618**	0.7554	**0.9481**	0.8850
Y+Z	0.8568	0.7490	0.8490	0.8753	0.8307	0.8964	0.8028	0.8903	0.9505	0.8247	0.9337	0.9192
Y-Z	**0.8790**	0.8079	0.8510	**0.8905**	0.8719	0.8593	**0.9242**	0.8990	0.9500	**0.8898**	0.9423	**0.9558**
Z-Y	0.0124	0.0857	0.2146	0.018	0.015	0.01	0.5555	0.7986	0.8514	0.1916	0.6933	0.8822
Z+W	0.4832	0.8018	0.7360	0.6769	0.7190	0.6826	0.1217	0.8758	0.9287	0.4369	0.8295	0.8514
Z-W	0.504	0.9171	0.6966	0.026	0.7314	0.5628	0.2215	0.7240	0.9065	0.01	0.9423	0.9307

Table A.6: Precision for classes in SingleOP evaluation

	SVM						Decision Tree						Gaussian NB					
	10 dim	**20 dim**	**30 dim**	**11 dim**	**31 dim**	**36 dim**	**10 dim**	**20 dim**	**30 dim**	**11 dim**	**31 dim**	**36 dim**	**10 dim**	**20 dim**	**30 dim**	**11 dim**	**31 dim**	**36 dim**
X-Y	**0.875**	0.6	1.0	0.5454	0.6	1.0	**1.0**	0.08	0.875	0.5714	0.7058	**0.8421**	0.0	0.0	**1.0**	0.0	0.0	0.9
X+Y	0.75	0.6	0.8	0.0666	0.75	0.75	0.75	0.5	0.6666	0.125	0.25	**1.0**	0.0571	0.15	0.1071	0.0540	0.0967	0.1034
Y-X	0.0	0.0	**1.0**	**1.0**	0.0	**1.0**	0.4	0.25	**1.0**	0.0909	0.25	**1.0**	0.1818	0.0	0.3333	**1.0**	0.0	0.3333
Y+Z	0.8333	**0.9245**	0.9454	0.8529	**0.9464**	0.9464	0.8281	0.8032	0.9272	0.86	0.8813	0.8965	0.6	**0.875**	0.7894	0.6	0.5	0.6923
Y-Z	0.85	0.7209	0.8767	0.8142	0.7922	0.8888	0.7727	0.6153	0.8513	**0.8028**	0.8906	0.8939	**0.8571**	0.4838	0.6551	0.1428	0.4090	**0.9285**
Z-Y	0.0	0.75	**1.0**	0.0	0.8333	0.9090	0.0	**1.0**	0.9166	0.0	**0.9166**	0.8461	0.0	0.5882	0.5555	0.5	**0.8333**	0.7142
Z+W	0.2978	0.7352	0.7272	0.6818	0.7560	0.7560	0.2826	0.76	0.8378	0.75	0.7741	0.8611	0.0	0.3589	0.4	0.6666	0.3	0.4109
Z-W	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.5	0.2	0.0	0.0344	0.0	0.0	0.0333

	Random Forest						Perceptron					
	10 dim	**20 dim**	**30 dim**	**11 dim**	**31 dim**	**36 dim**	**10 dim**	**20 dim**	**30 dim**	**11 dim**	**31 dim**	**36 dim**
X-Y	**0.875**	0.0	0.6	0.8	0.6666	0.7894	**1.0**	0.3095	0.875	0.1648	0.0	0.9333
X+Y	0.75	0.6666	0.2	0.0769	0.4285	0.6666	0.0	0.6666	0.4444	0.1142	0.2307	0.6666
Y-X	0.0	0.0	0.0	0.0909	0.0	0.5	0.2222	0.0	**1.0**	0.3333	0.1666	**1.0**
Y+Z	0.8333	**1.0**	**0.9333**	**0.8378**	**0.9574**	**0.95**	0.4013	0.8518	0.9454	**0.58**	0.9333	0.5319
Y-Z	0.5765	0.5	0.5565	0.7159	0.64	0.5925	0.85	0.6551	0.7619	0.0	0.6666	0.8333
Z-Y	0.0	0.0	0.0	0.0	0.0	0.0	0.0	**1.0**	0.8571	0.0	**1.0**	0.9
Z+W	0.0	0.5365	0.8333	0.8181	0.7428	0.8571	0.0	0.6756	0.5428	0.4761	0.7916	0.62
Z-W	0.0	0.0	0.0	0.0	0.0	0.0	0.1666	0.5	0.0	0.0	0.0606	0.5

Table A.7: Recall for classes in SingleOP evaluation

	SVM						Decision Tree						Gaussian NB					
	10 dim	**20 dim**	**30 dim**	**11 dim**	**31 dim**	**36 dim**	**10 dim**	**20 dim**	**30 dim**	**11 dim**	**31 dim**	**36 dim**	**10 dim**	**20 dim**	**30 dim**	**11 dim**	**31 dim**	**36 dim**
X-Y	0.875	0.1875	0.875	0.75	0.375	0.875	**1.0**	0.125	0.875	0.75	0.75	**1.0**	0.0	0.0	0.5625	0.0	0.0	0.5625
X+Y	0.75	0.75	**1.0**	0.5	0.75	0.75	0.75	0.75	0.5	0.5	0.25	0.5	**1.0**	**0.75**	0.75	**1.0**	**0.75**	0.75
Y-X	0.0	0.0	1.0	**1.0**	0.0	**1.0**	0.4	**1.0**	**1.0**	**1.0**	0.5	**1.0**	1.0	0.0	**1.0**	0.5	0.0	**1.0**
Y+Z	**0.9016**	0.8032	0.8524	0.4754	0.8688	0.8688	0.8281	0.8032	0.8360	0.7049	**0.8524**	0.8524	0.0491	0.1147	0.2459	0.0491	0.0491	0.1475
Y-Z	0.2656	**0.9687**	**1.0**	0.8906	**0.9531**	**1.0**	0.2656	0.625	0.9843	0.8906	0.8906	0.9218	0.1875	0.2343	0.2968	0.0156	0.1406	0.2031
Z-Y	0.0	0.5625	0.4375	0.0	0.625	0.625	0.0	0.4375	0.6875	0.0	0.6875	0.6875	0.0	0.625	0.625	0.0625	0.625	0.625
Z+W	0.8	0.7142	0.9142	0.4285	0.8857	0.8857	0.7428	0.5428	**0.8857**	0.4285	0.6857	0.8857	0.0	0.4	0.5714	0.2857	0.4285	0.8571
Z-W	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.5	0.5	0.0	0.5	0.0	0.0	0.5

	Random Forest						Perceptron					
	10 dim	**20 dim**	**30 dim**	**11 dim**	**31 dim**	**36 dim**	**10 dim**	**20 dim**	**30 dim**	**11 dim**	**31 dim**	**36 dim**
X-Y	0.875	0.0	0.1875	0.25	0.25	0.9375	0.6875	**0.8125**	0.875	0.9375	0.0	0.875
X+Y	0.75	0.5	0.75	0.5	0.75	0.5	0.0	0.5	**1.0**	**1.0**	0.75	0.5
Y-X	0.0	0.0	0.0	**1.0**	0.0	0.5	**1.0**	0.0	**1.0**	0.5	**1.0**	**1.0**
Y+Z	0.9016	0.3770	0.6885	0.5081	0.7377	0.6229	**1.0**	0.7540	0.8524	0.4754	0.6885	0.8196
Y-Z	**1.0**	**1.0**	**1.0**	0.9843	**1.0**	**1.0**	0.2656	0.2968	**1.0**	0.0	0.6875	0.3125
Z-Y	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.375	0.375	0.0	0.4375	0.5625
Z+W	0.0	0.6285	0.4285	0.5142	0.7428	0.6857	0.0	0.7142	0.5428	0.2857	0.5428	0.8857
Z-W	0.0	0.0	0.0	0.0	0.0	0.0	0.5	0.5	0.0	0.0	**1.0**	0.5

Table A.8: f-measure for classes in SingleOP evaluation

	SVM						Decision Tree						Gaussian NB					
	10 dim	**20 dim**	**30 dim**	**11 dim**	**31 dim**	**36 dim**	**10 dim**	**20 dim**	**30 dim**	**11 dim**	**31 dim**	**36 dim**	**10 dim**	**20 dim**	**30 dim**	**11 dim**	**31 dim**	**36 dim**
X-Y	**0.875**	0.2857	0.9333	0.6315	0.4615	0.9333	0.8148	0.0975	0.875	0.6486	0.7272	0.9142	0.0	0.0	0.72	0.0	0.0	0.6923
X+Y	0.75	0.6666	0.8888	0.1176	0.75	0.75	0.75	0.6000	0.5714	0.2	0.25	0.6666	0.1081	0.25	0.1874	0.1025	0.1714	0.1818
Y-X	0.0	0.0	**1.0**	0.1666	0.0	**1.0**	0.5714	0.4	**1.0**	0.1666	0.3333	**1.0**	**0.3076**	0.0	0.5	**0.6666**	0.0	0.5
Y+Z	0.8661	**0.8596**	0.8965	0.6105	**0.9059**	0.9059	**0.8480**	**0.8032**	0.8793	0.7747	0.8666	0.8739	0.0909	0.2028	0.375	0.0909	0.0895	0.2432
Y-Z	0.4047	0.8266	0.9343	**0.8507**	0.8652	0.9411	0.3953	0.6201	0.9130	**0.8444**	**0.8906**	0.9076	**0.3076**	0.3157	0.4086	0.0281	0.2093	0.3333
Z-Y	0.0	0.6428	0.6086	0.0	0.7142	0.7407	0.0	0.6086	0.7857	0.0	0.7857	0.7586	0.0	**0.6060**	**0.5882**	0.1111	**0.7142**	**0.6666**
Z+W	0.4341	0.7246	0.8101	0.5263	0.8157	0.8157	0.4094	0.6333	0.8611	0.5454	0.7272	0.8732	0.0	0.3783	0.4705	0.4	0.3529	0.5555
Z-W	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.5	0.2857	0.0	0.0645	0.0	0.0	0.0625

	Random Forest						Perceptron					
	10 dim	**20 dim**	**30 dim**	**11 dim**	**31 dim**	**36 dim**	**10 dim**	**20 dim**	**30 dim**	**11 dim**	**31 dim**	**36 dim**
X-Y	**0.875**	0.0	0.2857	0.3809	0.3636	**0.8571**	**0.8148**	0.4482	0.875	0.2803	0.0	0.9032
X+Y	0.75	0.5714	0.3157	0.1333	0.5454	0.5714	0.0	0.5714	0.6153	0.2051	0.3529	0.5714
Y-X	0.0	0.0	0.0	0.1666	0.0	0.5	0.3636	0.0	**1.0**	0.4	0.2857	**1.0**
Y+Z	0.8661	0.5476	**0.7924**	0.6326	**0.8333**	0.7524	0.5727	0.8	0.8965	**0.5225**	**0.7924**	0.6451
Y-Z	0.7314	**0.6666**	0.7150	**0.8289**	0.7804	0.7441	0.4047	0.4086	0.8648	0.0	0.6769	0.4545
Z-Y	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.5454	0.5217	0.0	0.6086	0.6923
Z+W	0.0	0.5789	0.5660	0.6315	0.7428	0.7619	0.0	**0.6944**	0.6440	0.3571	0.6440	0.7294
Z-W	0.0	0.0	0.0	0.0	0.0	0.0	0.25	0.5	0.0	0.0	0.1142	0.5