AUTOMATIC ANSWER GENERATION FOR MATH WORD PROBLEMS

Kulakshi Fernando

178090C

Thesis/Dissertation 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 2019

DECLARATION

I declare that this is my own work and this dissertation 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 acknowledgement is made in the text.

Also, I hereby grant to University of Moratuwa the non-exclusive right to reproduce and distribute my dissertation, 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 (such as articles or books).

Signature: Date:

The above candidate has carried out research for the Masters Dissertation under my supervision.

Name of the Supervisor: Dr. Surangika Ranathunga
Signature of the Supervisor: Date:

Name of the Supervisor: Prof. Gihan Dias
Signature of the Supervisor: Date:

ACKNOWLEDGEMENTS

I am sincerely grateful for the advice and guidance of my supervisors, Dr. Surangika Ranathunga and Prof. Gihan Dias. Without their support, mentoring, and encouragement this project would not have been completed. I would like to thank them for taking time out of their busy schedule to be available anytime that was needed with help and advice.

I would also like to thank my progress review committee, Dr. Dulani Meedeniya and Dr. Supunmali Ahangama. Their valuable insights and guidance helped me to enhance the quality of my research work.

I would like to thank the entire staff of the Department of Computer Science and Engineering, both academic and non-academic for all their help during the course of this work and for providing me with any resources necessary to conduct my research.

This work was funded by a Senate Research Committee (SRC) Grant of the University of Moratuwa.

Finally, I would like to express my gratitude to my friends, specially Mr. Mokanarangan Thayaparan and Ms. Ishadi Jayasinghe who were always supporting me at my research work and to my family for their immense support.

ABSTRACT

Automatic Answer Generation for Math Word Problems

A math word problem (MWP) is a mathematical problem expressed using natural language. In this research, elementary level set-related word problems in which information is given in set notation are considered. As per our knowledge, this is the first research addressing set theory related word problems.

This research introduces an abstract representation to interpret mathematical semantics of set expressions and relations between sets. Two methods to extract given set related expressions were implemented: rule based method and a statistical method. Results show that statistical method is more robust to typing errors and unexpected expression formats. A parser based on a context free grammar is introduced to validate set related expressions and give feedback to the user when there are incorrect expressions. Along with these functionalities, we present a complete set problem solver system that understand and solve a given set word problem.

In addition to the solver, we experiment in extracting mathematical expressions from unstructured plain text using sequential classifiers. Several sequential classification models including conditional random-fields (CRF) and Long-Short Term Memory (LSTM) networks were compared with word and character level features. The results show that using character level features significantly increase the performance of mathematical expression extraction.

Keywords: Set theory. Answer generation. Math word problems.

LIST OF FIGURES

Figure 3.1	Overview of the system	21
Figure 3.2	Venn diagram of set A, the set of all positive integers between 1 and 9	23
Figure 3.3	The Venn diagram denoting information in question Q1. Numbers denote the cardinalities of sets and bordered labels denote the set that represent by each region	25
Figure 3.4	The graphical representation of subset-superset relations between all the possible sets in Q1. The binary representation of the sets are written with set names. Edges of the graph point subsets to supersets.	29
Figure 3.5	Extracting information from Q2 using rule based approach	31
Figure 3.6	Network architecture of the W-LSTM model	34
Figure 3.7	Network architecture of the W-Bi-LSTM model	34
Figure 3.8	Network architecture of the W-CH-Bi-LSTM model	36
Figure 3.9	Finding the universal set in Q2	39
Figure 4.1	CRF performance against cumulatively added feature sets from the set A to set G listed in Table 3.4	49
Figure 4.2	Performance of RNN models with respect to number of epochs	50

LIST OF TABLES

Table 3.1	Problem categories based on information part of the problem	24
Table 3.2	Binary representation of sets given in Q1 and cardinality information about each set	26
Table 3.3	Regular expressions used to extract set information	30
Table 3.4	Features used for CRF divided into categories. The left most column contains a label for each of the set of features	35
Table 3.5	Examples for set expressions that can be validated by the CFG parser	36
Table 3.6	Generated equations for sets in Q1	42
Table 4.1	Datasets used to evaluate solver system	46
Table 4.2	Statistics of problems in the dataset, DExpr	47
Table 4.3	Accuracy, Recall, Precision and F1-score of the best performance of all models	50
Table 4.4	Performance of statistical and rule-based expression extractors	51
Table 4.5	Statistics of the data used to evaluate the parser	51
Table 4.6	Evaluation results of the parser	52
Table 4.7	Statistics of the data used to evaluate the solver	52
Table 4.8	Evaluation results of the solver	52
Table 4.9	Evaluation results of the complete system	53

LIST OF ABBREVIATIONS

MWP	Math Word Problem
GCE	General Certificate of Education
IGCSE	International General Certificate of Secondary Education
GCSE	General Certificate of Secondary Education
JCE	Junior Certificate Examination
NCERT	National Council of Educational Research and Training
NLP	natural Language Processing
CRF	Conditional Random Field
LSTM	Long-Short Term Memory
Bi-LSTM	Bi-directional Long-Short Term Memory
CFG	Context Free Grammar

TABLE OF CONTENTS

Declaration of the Candidate & Supervisor i

Ackowledgement ii

Abstract iii

List of Figures iv

List of Tables v

List of Abbreviations vi

Table of Contents vii

1 Introduction 1

- 1.1 Background 1
- 1.2 Research Problem 2
- 1.3 Research Objectives 3
- 1.4 Contributions 3
- 1.5 Scope and Limitations 4
- 1.6 Publications 4
- 1.7 Organization 4

2 Literature Survey 6

- 2.1 Math Word Problem Solving Systems 6
  - 2.1.1 Rule-based approaches 7
  - 2.1.2 Statistical approaches 8
  - 2.1.3 Hybrid approaches 9
  - 2.1.4 MWP domains addressed in previous research 15
  - 2.1.5 Discussion on problem solving systems 15
- 2.2 Expression Extraction Methods 16
- 2.3 Information Extraction Using Sequential Classifiers 17
- 2.4 Summary 19

3 Methodology 20

- 3.1 System Overview 20
- 3.2 Set Problem Categorization 20

3.2.1 Presentation formats of sets 20
3.2.2 Properties of sets 22
3.2.3 Problem categorization 23
3.3 Abstract Representation of Sets 23
3.3.1 Representing any number of Sets 26
3.3.2 Representing main sets 26
3.3.3 Representing derivable sets 27
3.3.4 Finding subsets and supersets 28
3.4 Expression Extraction 28
3.4.1 Rule based expression extraction 29
3.4.2 Limitations of regular expressions 30
3.4.3 Expression extraction using statistical approach 32
3.5 Expressions Parsing 33
3.6 Mapping to Data Representation 38
3.7 Question Validation 39
3.8 Answer Generation 41
3.8.1 Generating equations and calculating cardinalities 41
4 Evaluation 45
4.1 Data Sets 45
4.2 Experimental Setup 46
4.3 Evaluations 48
4.3.1 Evaluation of math expressions extraction based on sequential classifiers 48
4.3.2 Evaluation of set expressions extraction 49
4.3.3 Parsing sets expressions 51
4.3.4 Solving a set problem 52
4.3.5 End-to-end performance 52
5 Conclusion 54
5.1 Future Work 55
References 56

Chapter 1

INTRODUCTION

A ‘math word problem’ (MWP) is a mathematical problem expressed using natural language. This thesis introduces an automatic solver for MWPs related to elementary set theory. The solver understands and solves a subset of problems in elementary set theory questions in ordinary level (O/L) examination papers.

This chapter introduces the background of the research problem, describes the motivation behind the research, presents research objectives, contributions and research publications.

1.1 Background

Mathematics is a fundamental subject that students must pass in almost every O/L examination around the world. Hence there are many online educational platforms for students to learning mathematics. MWPs given in elementary and ordinary level examinations are presented in human readable format using natural languages. They may also include descriptions and explained scenarios relevant to the problem.

E-learning systems support student-centered learning by helping students to learn and practice mathematical knowledge they gain in the classroom. In e-learning systems, it is important to give assignments to students, assess students’ answers and give feedback on students’ performance. In the process of fully automating such a system, understanding problems and generating answers for the problems given in assignments are quite useful. Given this capability, teachers can simply submit questions and let the system to validate questions, find answers, create marking schemes and assess students’ answers and give feedback on them.

There is much research conducted to automatically solve MWPs in several mathematical domains since 1964. Simple MWPs may be easily comprehensible even to children in elementary schools. However, given the complexities of natural language and the requirement of common sense, understanding MWPs is quite a challenge to computers. For example, consider the following problem;

Select equivalent sets from the following.
$A = \{$*Letters of the word "LONDON"*$\}$,
$B = \{27, 72\}$,
$C = \{$*Positive odd numbers less than 10*$\}$,
$D = \{$*Multiples of 2 less than 10*$\}$".

In order to solve this problem, a machine should understand semantics of the notation and common sense such as a written word is consists of letters. In addition, it needs mathematical knowledge such as how two sets become equivalent and what odd numbers are. In other types of questions it is also critical to filter out irrelevant information, specially when a scenario is described. Given all these requirements, even though answer generation for MWPs has been experimented through several decades, still there is not any system that performs adequately for all types of MWPs written in English language.

Research conducted to answer MWPs are mostly focused in a single mathematical domain in order to reduce the complexity in understanding MWPs. On the other hand, characteristics of MWPs are specific for different domains of mathematics. Most of this research covers elementary level arithmetic and algebraic problems[1, 2, 3, 4, 5]. Also, there is a few systems that address other domains such as kinetics, geometry and probability questions[6, 7].

1.2 Research Problem

Basic set theory is an essential mathematical domain in mathematical syllabi around the world including GCE O/L mathematics syllabus in Sri Lanka. Local GCE O/L mathematics syllabus[1] includes the knowledge of simple set manipulations that can be used in day today life. The syllabus includes formats of

[1]`http://nie.lk/pdffiles/tg/e10tim109.pdf` and `http://nie.lk/pdffiles/tg/e11tim168.pdf`

presenting a set, basic set operations and solving simple problems using set theory and Venn diagrams. Usually, the GCE O/L maths paper includes 3-5 number of questions from set theory in each year including multiple-choice-questions (MCQs) and a structured question. As well as the local O/L examination, general and junior level certified international math examinations including GCE, IGCSE, GCSE, and JCE and government examinations of other countries such as NCERT contains set theory and Venn diagram related questions in every year.

There are semi-automatic platforms available that support different aspects in learning set theory. For example, Wijesinghe et al. [8, 9] present systems that represent Venn diagrams and a system that assesses answers provided for Venn diagram related questions. However, the second system expects teachers to provide marking schemes.

So far, we are not aware of any system that automatically generates answers for set theory related MWPs. This is a gap that should be fulfilled in order to make a step forward towards an e-learning platform that can automatically generate answers and assess students' answers for a given sets related questions.

1.3 Research Objectives

Objectives of this research are as follows:

1. Introduce an abstract representation that captures semantics of sets problems.
2. Implement a solver that can solve O/L sets related MWPs.
3. Provide a baseline to evaluate systems that solve sets related MWPs in future.

1.4 Contributions

Following contributions are made in this thesis;

- Introduced a categorisation of set related questions.
- A dataset of sets related word problems.
- A dataset of math problems, which is annotated for math expressions belonging to several mathematical domains.

- An abstract data representation to store set related information in a problem, based on Knuth's representation of 16 logical connectives [10].
- A context-free grammar (CFG) parser that can identify errors in an mathematical expression written in set notation.
- A solver for set type MWPs using above elements.
- Comparison of sequential classifiers for the task of math expression extraction from unstructured plain text

1.5 Scope and Limitations

1. This research focuses only on textual set related problems where related information to solve the problem is presented in set notation.
2. The information types that the implemented system can understand is limited to set names, subset and superset relations, set cardinality and enumerated elements of sets. The system does not currently recognize other information provided using set notation such as set builder form of a set.
3. The system is implemented to recognize symbols in Unicode standard.

1.6 Publications

- "Answer Generation for Set Type Math Word Problems", in 2018 International Conference on Advances in ICT for Emerging Regions (ICTer) at University of Colombo School of Computing, Sri Lanka.
- "Mathematical Expression Extraction from Unstructured Plain Text", in 24th International Conference on Applications of Natural Language to Information Systems.

1.7 Organization

This thesis is organized as follows. Chapter 2 provides the background and related work for MWP solvers and math expression extraction. The methodology used to implement the solver and its components are described in Chapter 3. The implementation details, evaluation details, and results of each element of the solver is mentioned in chapter 4 along with result discussions. Finally, chapter 5 provides the conclusion of the thesis and future work possible in the research.

Chapter 2

LITERATURE SURVEY

This chapter describes the literature related to understanding and answering MWPs. The chapter is divided into three sections.

Section 2.1 describes previous approaches taken in solving MWPs. Recent MWP systems are reviewed based on the technical approach used and the data representation adapted to interpret the problem.

Correctly identifying related text from the given problem is one of the key factors that contributes to the accuracy of an MWP solver. Identifying given math expressions from ordinary text is challenging when the text is unstructured. Section 2.2 presents few previous work related to math expression extraction from unstructured plain text.

Statistical math expression extraction can be mapped into a problem of sequential text classification. Section 2.3 describes the contribution in sequential classifiers.

2.1 Math Word Problem Solving Systems

A system for answering MWPs has first implemented on 1964 which named 'STUDENT'[11], for simple arithmetic and algebraic problems. Mukherjee and Garain [12] provide a descriptive review on math problem answering systems introduced in the last few decades starting from 'STUDENT'[11]. Systems explained in this survey use rule based inference methods to understand and solve problems. There are many domains of mathematics covered in these systems, including arithmetic, algebra, geometry, probability, kinetics, mechanics, metrics and physics. However, Mukherjee and Garain [12] show that most of these systems are not evaluated empirically nor test data is provided to compare with other systems. Mandal and Naskar [13] provide a survey on recent arithmetic word problem solvers. They show the advancement in arithmetic problem solvers in last three decades. Technologies used in solvers have gradually moved into statistical and hybrid methods from complete rule based methods.

The general steps used in any solvers is to first identify relevant information from text, then map them into a machine readable interpretation and finally solve the problem by generating proper equations based on that interpretation. Equations are then solved using computer algebraic systems(CAS). The majority of the work of automatically solving a MWP is on extracting relevant information accurately, and on interpreting information in a way that helps to generate equations. MWP solving systems can be categorized according to the technologies and information interpretation structures used in the system. A review of recent attempts taken in MWP solvers based on their technologies and the data representation structures is presented in this section.

2.1.1 Rule-based approaches

Rule-based approaches mainly try to understand the semantics of the problem by generating a corresponding meaning representation, which is then used to generate equations. When semantic representations are used for problem solving, the text is parsed using a grammar into a pre-defined structure that models the semantics of the problem.There have been systems that use combinatory categorical grammar (CCG)[14] to parse text of MWPs. CCG is based on combinatory logic and supports understanding text both syntactically and semantically.

Matsuzaki et al. [15, 16] use CCG to parse pre university MWP in Japanese language. Then via a discourse level semantic decomposition they resolve coreferences of the problem text. Problem text is then converted into a formula in Zermelo–Fraenkel set theory. They rewrite these formulae preserving their meaning into a more machine understandable format. Finally, the answer is obtained using CAS (Computer Algebra System) and ATP (Automated Theorem Proving) systems. Even though the accuracy is less, they show that the system competes well with average human performance on pre-university mathematics examinations.

Semantic representations used in systems vary with domain specific properties of mathematical problems. Shi et al. [17], Seo et al. [6], and Dries et al. [7] introduce meaning representation languages for numerical, geometry and probability word problems, respectively. Works of Seo et al. [6] and Dries et al. [7] are described under hybrid approaches in subsection 2.1.3. Shi et al. [17] present DOL, which stands for 'Dolphin Language' in their solver (named 'SigmaDolphin') for number word problems. A given word problem is mapped into a parse tree in DOL using 9600 CFG rules generated in a semi automatic way. The node types of a DOL tree are, constants, classes and functions. constants are numbers and nouns in the problem (e.g., 3.56, 'New York') and classes refer to a predefined data types for entities in the problem. For example, the class of the entity 'New York' is '*location.city*' and class of '3.56' is *math.number*. Functions are used to form larger language units and comprise with a set of built in functions such as '*nf.math.sum*'. Given a DOL tree, equations are generated in a reasoning module. They show that SigmaDolphin performs significantly better than the work of Kushman et al. [2] which was the state-of-the-art work at that time.

2.1.2 Statistical approaches

In contrast to rule-based approaches, statistical methods focus on generating equations to solve a given problem, based on statistics of the problems seen in the training phase.

The work of Roy [18] uses a cascade of classifiers for each step of their system. They address arithmetic word problems in this research. Their system originally focuses on extracting and understanding quantity information in a text, which is called 'quantity entailment'. Quantity information can be presented in various forms. Just a cardinal number can be written using numbers or letters. In addition ordinal numbers, dates, a time duration, rates, currency, etc. are related to quantity information in a text, which may need to consider for calculations. In addition the way quantities are represented can be more complex sometimes. For example "*A bomb in a Hebrew University cafeteria killed five Americans and* *four Israelis*" can also be written as "*bombing at Hebrew University in Jerusalem killed nine people, including five Americans*".

Roy [18] first finds segments of texts describing quantities, using a bank of classifiers that use features including word class, POS tags and character information. Then units (entities) are extracted using co-reference resolution and semantic role labeling. Finally, quantities are represented in a tuple containing values, units and changes (such as increasing and decreasing) of a quantity. As an application of quantity entailment, they have implemented an arithmetic MWP solver. Here they use additional classifiers to find related quantity pairs, arithmetic operations that should be applied to a pair of quantity, and order of operations that should be applied to a set of quantities. Even though they claimed to have high accuracy rates (over 86%), an empirical evaluation has not been carried out to compare results with other state-of-the-art systems.

2.1.3 Hybrid approaches

Solving MWPs include handling a comparatively restricted language domain and problem traits specific to mathematical domains. Even though rule based approaches seem more suitable to the problem they are hard to scale and require lot of hard work to tackle all the complexities of the problems. In contrast, statistical approaches are more scalable but require more data and computational power. In the problem of answer generating to MWPs, hybrid approaches that combine both rule based and statistical based approaches are widely used.

Using equation templates : In template matching approaches, the text is converted directly into the necessary system of equations by filling slots of predefined equation templates. The matching template for a given problem is selected using a trained statistical model. Kushman et al. [2] first introduced template matching approach for automatically solving math word problems. They addressed algebraic word problems and their work is a well-recognized research in this field. They find suitable template of system of equations for the problem, aligned with numbers and nouns given in the text. The combination of a given word problem, matching system of equation templates, aligned equations with nouns and numbers and the answer of the problem is named as a derivation. The best derivation for a given problem is matched using a log-linear model and beam-search. Much research [19, 20, 21, 5] was then conducted to enhance the performance of this method on solving algebraic word problems. Zhou et al. [19] update the features and the search algorithms used in the work of Kushman et al. [2]. Also, instead of considering both nouns and numbers in the problem text, they only consider numbers to align with equation templates which drastically reduce the space and time complexity of the process. With respect to Kushman et al. [2] they achieve 10% more accuracy on the same dataset.

Both the work of Kushman et al. [2] and Zhou et al. [19] is empirically evaluated by Huang et al. [20] on a large-scale dataset called Dolphin18K that comprises 18K MWPs. They show that these systems perform poorly in such a large dataset in contrast to less diverse small-scale datasets used in the original research. They also confirm that using more training data does not provide much of a benefit to improve the performance of given two statistical approaches. Huang et al. [20] also introduce an algebraic word problems solver named SIM which performed best in their experiment for the large scale dataset. SIM is also a template based approach, but the approach for aligning templates with data employs a similarity based method. They calculate the weighted Jaccard similarity between vectors of word TF-IDF scores of a given problem and every problem in the training set. The template corresponding to the most similar problem is selected to solve the given problem.

Taking a step ahead on the work of Zhou et al. [19], Upadhyay and Chang [21] found that accuracy of the answer can be increased by evaluating all candidate derivations and selecting the best one, instead of evaluating only the final answer. The problem of evaluating only the answer is that the correct answer can be generated via incorrect derivations. Huang et al. [5] also use the strategy of finding several candidate templates. In their approach, instead of only extracting numbers from the text, they find for textual expressions that can be found in algebraic word problems and they map these phrases into math expressions.

For example, textual expressions such as '*[NUM]% discount*' and '*[NUM]% off*' should be mapped to the math expression '$(1-n)$' when writing equations. They show that their system achieves 10% higher accuracy with respect to state-of-the-art systems over 'Dolphin18K'[20] dataset.

Using expression trees : Another approach used to representing information of arithmetic word problems is using expression trees, which is a tree representation of the required mathematical equation, where nodes of the tree are operators and leaves are operands. Koncel-Kedziorski et al. [22] introduce expression trees to solve multi-line algebra word problems in their solver names ALGES. They first identify 'Qsets' of the problem which is a tuple of the entity, and information about the entity such as adjectives, verbs and location information. Candidate expression trees are generated using log-linear programming and these tree structures are scored based on their likelihood of capturing mathematical computation expressed in the word problem. This likelihood score is found by learning discriminative models; a local model trained to map spans of text to math operations and global model trained for the coherence of the entire equation with respect to global problem text structure.

Roy and Roth [23] ground a given problem containing all arithmetic operators (multiplication, division, addition and subtraction) which may need to be solved using multiple steps into an expression tree. They first extract candidate operands from the text and then from all candidate operands they find relevant set of operands using classifiers. The least common operators between pairs of operands are predicted using a multi-class SVM classifier and the best operator is selected using a join inference procedure. Roy and Roth [4] introduce ILLINOIS Math Solver as the implementation of the described concept.

'Unit dependency graphs (UDG)' are another tree representation of problems introduced in Roy and Roth [24]. They use UDG of a problem when creating the expression tree to make the expression tree more accurate and consistent with the text. A UDG maps relationships between quantities and their units. An example of such a relationship is application of rates, as in, "*40 miles per hour*". One of the vertices of a unit dependency graph represents the question asked and other vertices represent quantities in the question whereas the links connecting vertices are labeled with the relationship between two quantities. Vertices and edges of the graph are found using a binary classifier and a multi-class classifier respectively. It achieves accuracy over 80% on available datasets.

Using problem classification : In this approach problems are first classified based on procedures that can be used to solve the given problem. Then equations are generated by matching numbers and nouns in the problem into predefined equation templates according to the problem category. A common classification used for arithmetic categorizing [3, 25] is dividing problems into '*change*' (or '*join and separate*'), '*part-whole*' and '*compare*' classes. For example, the arithmetic problem '*There are 7 dogs to walk on Henry's street. Henry walked 4 of them. How many dogs does Henry have left to walk?*' belongs to the '*Change*' category. Problems in '*Change*' category represents a quantity change of some entity; the changing entity in the above problem is '*dogs*'. 'Change' class is mapped with the equation format,

$$Val_{start} + \sum_{g \epsilon gains} Val_g = \sum_{l \epsilon loses} Val_l + Val_{end} \tag{2.1}$$

Mitra and Baral [25] use further sub categorization step for arithmetic problems according to the placement of unknown variable of the equation. They use a probabilistic model with a set of features related to each category. They use ConceptNet[1] to compare the type of entities and verbs associated with quantities. They claim an accuracy over 86%. Among the challenges to these systems, a common one is failing to handle complex questions. In such questions properties of several categories may applicable. For example, '*There are 48 erasers in the drawer and 30 erasers on the desk. Alyssa placed 39 erasers and 45 rulers on the desk . How many erasers are now there in total ?*' combines both 'change' and 'part-whole' properties. Also, these systems are not capable of handling questions that require common sense information such as a week has seven days.

The work of Cetintas et al. [26] is worth mentioning here. They experiment in

[1]`http://conceptnet.io/`

categorizing arithmetic word problems into two categories, 'multiplicative compare' and 'equal group' using an SVM classifier. They conduct the classification with different settings of data such as avoiding stop word removal, stemming and using POS tagging. They identify that stemming and stop word removal reduce the accuracy of text classification. Also, they show identifying discriminative and non-discriminative POS tags related to the classification in the problem text is important to enhance the accuracy of the classification.

State/Frame identification and verb categorization : An arithmetic word problem usually describes a scenario in real world. This method identifies world states of the problem and update quantities in these states according to verbs that govern entities in the states. There are some research [22, 1, 7, 27] that identify objects from the problem that represent an entity along with related information about entities. For example, Koncel-Kedziorski et al. [22] identify a tuple of entity, and other information including quantity, adjectives, location, verbs, syntactic and positional information, and container of the entity. They define this combination as a 'Qset'. Dries et al. [7] identify entities, their properties, containers of entities, and actions related to an entity.

Hosseini et al. [1] introduced states that may include several objects. They use verb categorization to track the state changes in arithmetic word problems in their solver named 'ARIS'. They introduce 7 categories of verbs. An SVM classifier is used for the classification. Mishra et al. [27] recently introduced frames that consist of entity-objects that they name as slots. They use two kinds of frames, state frames and action frames. Action frames are what act upon state frames that cause a state change. The creation of frames is triggered by verbs in the sentence and the frame identification is done using SVM and random forest models. They try to answer any type of question asked about the scenario given in the problem such as 'who' and 'what' problems. They also present the answer with step by step illustrations of the solution to help students. However, the performance in answering in this system is much lower compared to the work of Hosseini et al. [1].

The main limitation of using verbs to understand a problem is that it fails when common sense is required to match two verbs. For instance consider the problem,‘*Last week Tom had $74. He washed cars over the weekend and now has $86. How much money did he make washing cars?*’. The relationship between washing cars and making money needs common sense knowledge.

Using meaning tags : In this approach the selected text is tagged using role-tags such as ‘verb’ and ‘subject’, in order to understand the semantics of the problem. Liang et al. [28] solves arithmetic word problems that are solvable using one operator. Their approach is a tag-based statistical method. The system first uses a linguistic analyzer to obtain corresponding linguistic representations of the problem such as dependency trees and co-reference chains. Solution type classifier, which is an SVM with linear kernel functions, classifies problems according to their solution types such as ‘addition’, ‘subtraction’, ‘multiplication’, ‘time variant quantities’ and etc. They use verb category, key-word indicators and aggregation pattern indicator related features to classify problems into solution types. According to the solution type, logical form converter converts the problem into a logical representation using first order logic based on a rule based approach. Afterwards, based on the logical form, equations are inferred and answers are generated by the inference engine. The system has achieved high accuracy over 90% which is higher than state-of-the-art systems at that time.

Using declarative languages : Some research ([6, 17, 7]) introduces new declarative languages to represent semantics of the problem according to the domain of mathematics. Seo et al. [6] introduced a fully automated system named GEoS, for solving multiple answer questions in SAT level geometric problems. They statistically parse both text and images into an expression in a newly introduced declarative language, Ω. Ω is a subset of typed first order logic that includes ‘constants’ (e.g., 0, 5), ‘variables’ (e.g., ‘AB’ in ‘line AB’), ‘predicates’ (e.g., Equals, IsDiameter) and ‘functions’ (e.g., LengthOf, SumOf, AreaOf). Types of these components are either numeric or Boolean.

Dries et al. [7] recently addressed answering probability related questions using a similar approach as Seo et al. [6]. They introduced a dataset of 2160 probability word problems. Given a problem, they identify entities, containers and attributes as in Hosseini et al. [1] from the problem text using a set of rules. Their language includes a set of containers (multisets), a set of attributes and their associated values, a set of size containers, a set of mutiset relations, and a set of observations and a set of queries. Given the probability problem defined using the aforementioned language, the problem is solved using a Bayesian network by calculating conditional probabilities.

2.1.4 MWP domains addressed in previous research

Clearly, most of the MWP solving research is conducted to solve arithmetic and algebraic word problems. In the survey of Mukherjee and Garain [12], they summarize research conducted in other domains, including distance and volume rate (calculus) problems [29], matrices [30], mechanics [31, 32], physics [33, 34], and mathematical proofs [35].

Investment, distance, projectiles and percents related problems are addressed by Morton and Qu [36] recently. Matsuzaki et al. [15] address any MWP in pre-university mathematics examinations including mathematical proofs. In addition, recent research are available that solve number word problems [17], geometry problems [6], and probability word problems [7].

Other essential mathematical domains such as statistics and graphs, set and Venn diagrams, trigonometry, surface area and volume of solids, inequalities, and geometric constructions are not yet addressed by any previous research so far, as per our knowledge.

2.1.5 Discussion on problem solving systems

Recent approaches for answer generating for MWPs widely adapt hybrid methods. That eliminates the overhead of preparing data in statistical methods and utilize more domain level properties of MWPs. Among hybrid approaches, the template matching method is a widely scalable and a remarkable method which attracted lot of attention recently. The highest benefit of this method is that it can be used in any math problem which can be solved using a system of linear equations. However, the performance of this method heavily depend upon the training dataset and the systems may fail to solve unseen types of problems. Also, the challenge of correctly filtering relevant and irrelevant information and correct alignment with slots in the template is still there which is later improved using methods including template ranking and deep learning. The other well performing method is using a semantic based approach where text is parsed into a representation using a language that is build to understand problems in the particular domain. Even though this method brings better performance, this is less scalable across different domains.

Solving problems in arithmetic and algebra is more researched than other domains like geometry and probability. Therefore, there are standard datasets and baseline methods available for these mathematical domains. For example, Koncel-Kedziorski et al. [37] provide an online repository for math word problems, MAWPS with a unified test-bed to evaluate new algorithms. There repository includes arithmetic and algebraic problem datasets provided by research explained above. Cetintas et al. [38] research on identifying relevant and irrelevant information given in arithmetic problems by considering joint probability between words in question and information parts of the problem. In contrast, other mathematical domains do not have such facilities. Also, some domains like sets related problems are not have been addressed so far, as per our knowledge.

2.2 Expression Extraction Methods

Most of the times the math expressions in the World Wide Web are presented in the form of images, where they are written using tools like latex or MathML that converts the structured math expressions into pictorial expressions. Extracting textual math expressions written in unstructured format needs a mechanism to separately identified from other ordinary text.

Most of the MWP solving systems in arithmetic and algebraic domains do not require a special attention for extracting and understanding math expressions, rather they focus on selecting relevant information which is given as constants and nouns. Other domains such as Geometry requires understanding given formulae and other math expressions. Seo et al. [6] use regular expressions to extract numbers and variables. At implementation level, they use a simple equation parser that parse arithmetic equations into prefix notation. Matsuzaki et al. [15] address pre-university math problems that contain complex mathematical expressions, but they expect these expressions to be annotated in the input using MathML.

The work of Tian et al. [39] is the only research that focuses on extracting math expressions in unstructured plain text according to our knowledge. They use a Hidden Markov Model (HMM) with 8 hidden states which are parts of math expressions such as variables, constants, monocular and binocular operations, left and right delimitation, etc. These states correspond to observable mathematical symbols in the text accordingly. The model is trained with a dataset that consists of 13,423 expressions and achieved over 89% accuracy and 77% recall. However, the dataset is not publicly available in order to be used as a baseline.

2.3 Information Extraction Using Sequential Classifiers

In NLP, automatically extracting structured information from semi-structured or unstructured text is known as information extraction. Extracting unstructured textual mathematical expressions is a special kind of information extraction which needs to be handled specifically with respect to ordinary information extraction methods. This can be considered as a sequential classification task for two classes of text; ordinary text and math expressions related text. Sequential text classifiers are trained by taking a sequence of labeled text, and learning correlations between units of text to classify the text into predefined classes. Given an unlabeled sequence of text, a sequential text classifier then predicts the classes of the units of the text considered in the application.

Early approaches in text classification used rule-based methods such as domain-specific gazetteers[40], rule inference and dictionary based approaches. Then unsupervised learning based methods[40] like clustering and feature based supervised learning methods were used. Feature based supervised learning approaches include Hidden Markov Models (HMM) [41, 42], decision trees [43], Maximum Entropy Models [44], Support Vector Machines(SVM) [45] and Conditional Random Fields (CRF) [46] which were dominating methods in sequential text classification in past decades. With respect to other models, CRFs can take both past and future text into account when learning the model. The work of Finkel et al. [47] was a milestone model that uses CRF which performed well on CoNLL 2003 dataset at that time. Nadeau and Sekine [48], Sharnagat [49], Yadav and Bethard [50] provide detailed surveys on these approaches.

Deep learning technologies are the state-of-the-art procedure for sequential classification nowadays[51]. With respect to previous methods, deep learning methods support non-linear mapping between the input and output. Therefore, it can learn complex features from data. It reduces the effort of feature engineering that is required for other learning approaches. The input for a deep learning model is usually word-level, character-level or hybrid vector representations of text, known as embeddings, where dimensions of a vector represent latent features. They capture both semantic and syntactic properties of text. But on the other hand, deep learning methods require a lot of data to train the model. Convolution neural networks (CNN) and recurrent neural networks (RNN) are the two main types of deep learning models. Yin et al. [52] show that RNNs such as Long Short Term Memory (LSTM) networks and gated recurrent units (GRU) performs better and more robustly than CNNs for sequential classification tasks.

Combining both deep learning and traditional supervised learning methods is proven to be beneficial in NER tagging. Huang et al. [53] experimented with Long Short Term Memory (LSTM) networks combined with CRFs for NER. LSTM networks reduce the vanishing gradient problem in standard RNNs and perform well for long sequences of data. Bidirectional LSTM (Bi-LSTM) networks increase the accuracy of sequence tagging by considering both past and future inputs. Huang et al. [53] experiment with different combinations and show that the model architecture which combines a Bidirectional-LSTM (Bi-LSTM) with a CRF layer at the output performs best in their experiment. The work of Chiu and

Nichols [54] is another milestone in sequential text classification which combined Bi-LSTM with CNN. Research including the work of Ling et al. [55] show that using character level information with LSTM networks is effective in increasing the performance of language modeling. Li et al. [51] provide a detailed survey on modern deep learning strategies used for in sequential text classification. Now the burden of feature engineering is minimized, they identify main challenges in text classification to be annotating data and handling unseen data.

Mathematical expression extraction is a special task of sequential text classification. OIn summary, deep learning methods are the state-of-the-art technology used in sequential classification. Recent research use combined models[53] of both deep learning and traditional supervised learning methods to enhance the accuracy in sequential text classification.

2.4 Summary

When considering technologies used in MWP solvers, it is clear that hybrid approaches that employ both rule based and statistical methods are widely used. Most of the MWP solvers in previous research solve arithmetic and algebraic word problems. Therefore, resources like datasets and online testing platforms that facilitate research for solving arithmetic and algebraic problems are available. Many domains of essential mathematics involving set theory is not yet addressed as per our knowledge.

Extracting math expressions in unstructured plain text is a special task in information extraction that can be solved using sequential text classification. The state-of-the-art procedure used in sequential text classification is deep learning methods. Tian et al. [39] is the only research available for math expression extraction from plain text, and they use an HMM based model in their research.

Chapter 3

METHODOLOGY

This chapter explains the methodology used in this research to solve a given set related problem. Section 3.1 provides an overview of the system. Section 3.2 describes the categories of sets related problems. Section 3.3 presents the abstract data representation used to capture the mathematical semantics of the problems. Section 3.4 describes the approaches used to extract information from a given set problem. Finally, section 3.8 describes the steps of generating the answer.

3.1 System Overview

Figure 3.1 depicts the overview of the set problem solving system. Given a problem, first expressions presented in set notation are extracted out. Then these expressions are parsed to check syntactical errors. Next, expressions in set notation is mapped into a data representation which is capable to model the semantics of set related information. After that, using this representation the problem is validated. Any syntactical error of expressions or a logical error in the problem is displayed to the user as a feedback. Finally, given the information mapped into the aforementioned representation, necessary equations are generated and solved to obtain the answer.

3.2 Set Problem Categorization

3.2.1 Presentation formats of sets

Information of sets can be presented using set notation, set builder notation, roster form, Venn diagrams and descriptions. Set notation is the well known formal way of expressing information of sets. In set notation, a set is considered as an object. When describing a set, a set is denoted using curly braces ('{}') and the definition or elements of the corresponding set is written inside the curly braces

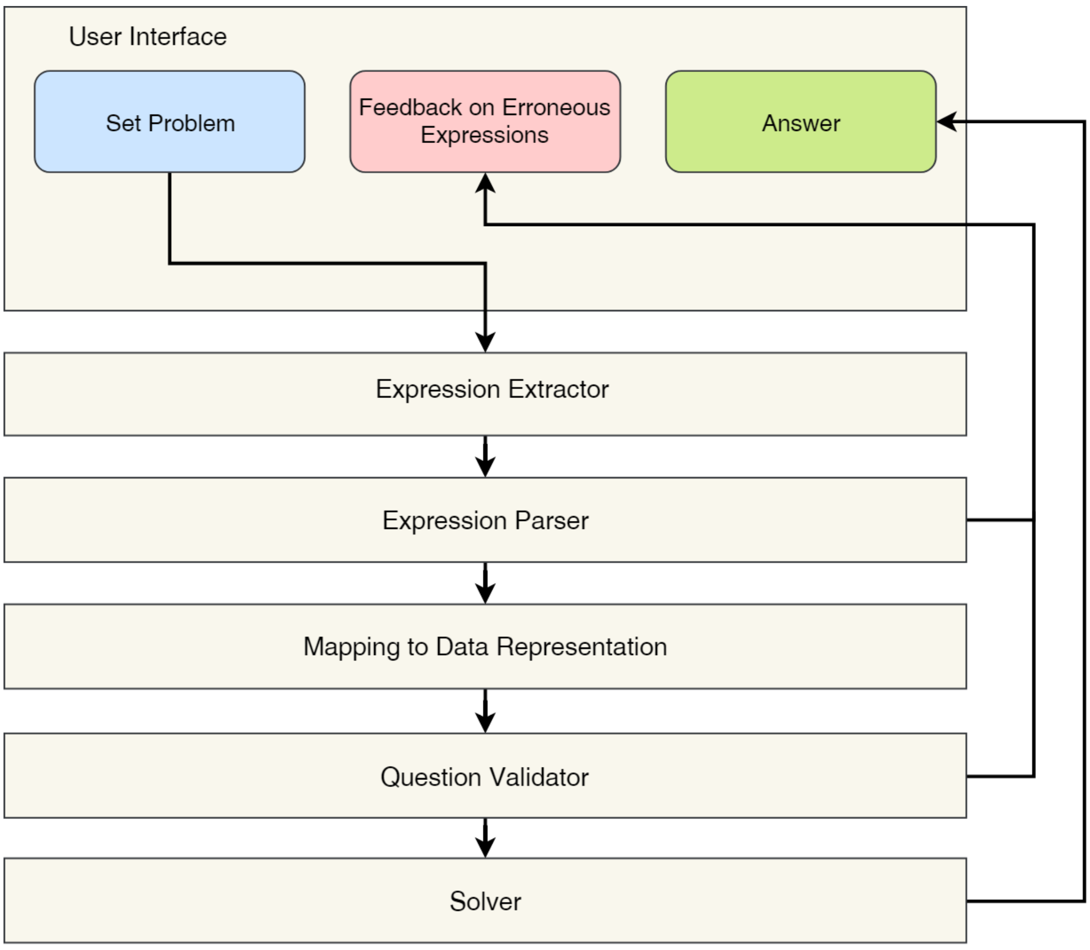


Figure 3.1: Overview of the system

(e.g.,‘A ={Monday,Tuesday,Wednesday,..}). Presenting a set using its members is known as roster form of the set. Instead of listing all elements, the way to find the members of a set can be formally written using set builder notation. For example, $\{x \in \mathbb{N} \mid x \neq 0\}$ denotes the set of all positive numbers in set builder notation. Venn diagrams [56] are another famous illustrative model to represent information about sets. Apart from these formal methods, a set can be described in words (e.g., Set A denotes the set of all days in a week).

3.2.2 Properties of sets

A set has a name, definition, elements and a cardinality. Set name is the identity of a set. Usually, set name is a capital English letter. By convention, some symbols are reserved for widely used and important sets. For example, $\mathbb{N}$ denotes the set of natural numbers and $\mathbb{R}$ denoted the set of real numbers. In addition ξ denotes the universal set and ϕ denotes the empty set.

The definition of a set is basically an expression of what the set contains. For example, ‘*A is the set of positive integers between 1 and 10*’ is a definition of the set A. This can be written either as an informal description, in set notation or in set builder notation. Set A is represented in set notation and set builder notation as follows; $A = \{\textit{positive integers between 1 and 10}\}$ and $A = \{x \mid x \in; 1 < x < 10\}$.

Elements are the members of a set. Elements can be expressed in roster form as an unordered, comma-separated list inside the set. For example, elements of set A is written as $A = \{2, 3, 4, 5, 6, 7, 8, 9\}$. This can be illustrated in a Venn diagram as shown in Figure3.2.

Cardinality is the number of elements in a set. It can be written in a description. For example, ‘*the size of set A is 8*’ or ‘*the set A has 8 elements*’. Formally, cardinality is denoted in set notation as $n(A) = 8$ or $\mid A \mid= 8$. Set cardinality is illustrated in a Venn diagram by writing the cardinality on the boundary of a set as denoted in Figure3.2.

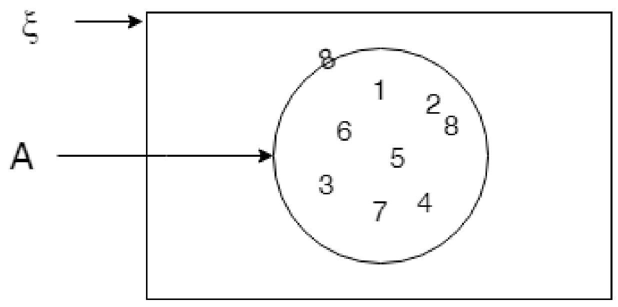


Figure 3.2: Venn diagram of set A, the set of all positive integers between 1 and 9

3.2.3 Problem categorization

According to the presentation formats and properties of sets, we categorized problems in the collected dataset. Prior to the categorization, the information and the question parts of the problem is separated. Table 3.1 shows the categories for information part of the problem along with examples. The same procedure is applied when categorizing the question part of the problem text.

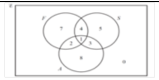
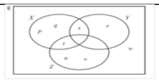
3.3 Abstract Representation of Sets

In this section we describe the data representation we use to denote sets related information. It is based on the Venn diagram representation of sets. Venn diagrams can depict,

- Universal set and its subsets with names,
- Elements of sets,
- Cardinalities of sets,
- Superset-subset relationships among sets,
- Sets derived from applying set operations (complement, union and intersection) between given sets

Since derived sets are also sets we refer given sets as ‘main sets’ and sets gen-

Table 3.1: Problem categories based on information part of the problem

Category	Format of the Representation	Given Information	Number of Equations	Example
1	Set notation	Cardinality	220	if |A|=37, |B|=50 and A⊂B, find |A∪ B| and |A∩B| If n(ξ)=24, n(A)=11, n(B)=18,find the greatest value of n(A∪B)?
2		Elements, Cardinality	161	S = {s, q, u, a, r, e}, V = {a, e, i, o, u} I. List the members of S∩V II. Find n(S∪V)
3	Description	Cardinality	106	There are 30 people in a group. 17 own a car, 11 own a bicycle and 5 do not own either a car or a bicycle. Using a Venn diagram, find how many people in this group own a car but not a bicycle.
4	Set builder notation / set notation	Definition of the set	97	ξ={x:1⩽x⩽12, x is an integer}, M={odd numbers}, N={multiples of 3} I.Find n(N). II. Write down the set M∩N.
5	Venn diagram	Cardinality	156	 The Venn diagram shows the number of students who study French (F), Spanish (S) and Arabic (A) I. Find n(A∪(F∩S)). II. On the Venn diagram, shade the region F'∩S.
6		Elements	96	 Use set notation to complete the statement: {u, v} ____ Z Shade X∩(Z∪Y)'.
7	Other types of Problems			

erated by applying operations to main sets as ‘derived sets’. In a Venn diagram, we refer a Jordan curve (non-self intersecting closed-curve) as a ‘region’. We refer combinations of regions as ‘combined-regions’ which also depicts sets.

Our representation utilizes the presentation used by Knuth [10] when describing 16 logical connectives between two binary variables. Since logical operations are similar to set operations, these 16 logical connectives which are represented in a binary value can be mapped into a shaded Venn diagram. So that, when considered a Venn diagram drawn for two variable (considering the sets are intersecting), each possible shading in the Venn diagram denotes each of the binary connectives between two binary variables. This is achieved by assigning each region of the corresponding Venn diagram into a bit position of the binary value. If the corresponding bit of a region is 1, the region is shaded; if the bit is zero, the region is kept unshaded.

To simplify the description consider the question Q1,
‘*It is given that* $n(\xi) = 40$, $n(P) = 18$, $n(Q) = 20$, *and* $n(P \cap Q) = 7$. *Find,*
(i) $n(P \cup Q)$
(ii) $n(P' \cap Q')$

The Venn diagram that represent above information is illustrated in Figure 3.3. Using the binary representation for Venn diagrams described above, we can map the Venn diagram depicted in figure 3.3 into binary values in Table 3.2.

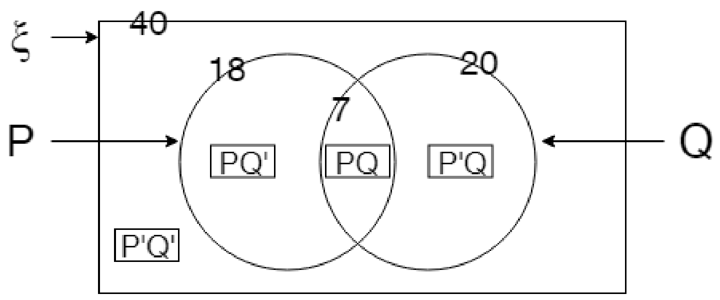


Figure 3.3: The Venn diagram denoting information in question Q1. Numbers denote the cardinalities of sets and bordered labels denote the set that represent by each region

Table 3.2: Binary representation of sets given in Q1 and cardinality information about each set

Region\ Combined-region	**Bit positions**				**Decimal value of the binary representation**	**Cardinality of the set**
	P’Q’	**PQ’**	**P’Q**	**PQ**		
$P \cap Q$	0	0	0	1	1	7
Q	0	0	1	1	3	20
P	0	1	0	1	5	18
$P \cup Q$	0	1	1	1	12	NA
$P' \cap Q'$	1	0	0	0	14	NA
ξ	1	1	1	1	15	40

3.3.1 Representing any number of Sets

In this research, we extend the mapping used in the illustration in Knuth [10] into any number of variables instead of two variables. Given n number of main sets, a Venn diagram should have 2^{n} maximum number of regions (when each set intersect with other sets). This can be shown using simple induction; we omit the proof since it is trivial. Consider Γ, the set of all regions in a Venn diagram denoting n number of main sets which intersect with each other. By taking the power set of Γ we can say that the number of all regions and combined-regions (or in other words, number of possible shadings of the Venn diagram or number of all sets that can be denoted by the Venn diagram) is $2^{2^{\mathrm{n}}}$. To map this information into binary representation we need 2^{n} bit positions, so that we can denote any shading in the Venn diagram, or any set combination given n main sets, using 2^{n} number of bits and $2^{2^{\mathrm{n}}}$ number of binary values.

3.3.2 Representing main sets

In order to keep track of which bit position represents which region in the corresponding Venn diagram, there should be a particular order of representing regions. We chose an order where leftmost bits carry the presence of complements of main sets and rightmost bit positions carry the presence of main sets. For example, refer the Table 3.2. There we ordered sets which participate complements of both main sets in Q1 into leftmost bit. As move to the right, the complement of a main set is omitted. Finally, we get the order of regions in Q1 as $P'Q'$, $P'Q$, PQ', PQ.

When regions always follow a particular order the binary representation of main sets remain unchanged. Given n number of main sets, we can calculate the decimal value of the binary representation of main sets as follows. Consider a matrix of ones and zeros. Let columns of this matrix denote regions in the corresponding Venn diagram where regions are ordered in the above mentioned manner. Let rows of the matrix denote all possible sets that can be represented in the corresponding Venn diagram. Let V be i^{th} row vector in the matrix and n be number of sets. Then, equation 3.1 shows $v_{mainseti}$, the decimal value of the binary representing of the i^{th} main set.

$$v_{mainseti} = \sum_{j=0}^{n-1} V_{i,j} \times 2^{n-1-j} \tag{3.1}$$

3.3.3 Representing derivable sets

Once binary representations of main sets are calculated (equation 3.1), all representations of derived sets can also be obtained. The binary representation of the derived set obtained by applying union operation between two sets is the result of bitwise-OR of the binary representations of participating sets. Likewise, the binary representation of intersection and complement is found by using bitwise-AND and bitwise-complement between binary values of participating sets, respectively. For instance, consider sets mentioned in the question, Q1. Let decimal values of binary representations of set P and Q, in Q1, be 5 and 3, respectively. The binary representation of the set $(P' \cap Q')$ is calculated as follows.

Binary representation of set P $= 0101_2$

Binary representation of set P'

$=$ *Bit-wise complement of* 0101_2

$= 1010_2$

Binary representation of Q' = 1100_2

Binary representation of $(P' \cap Q')$

= *Bitwise AND of binary values of P' and Q'*

= 0100_2

3.3.4 Finding subsets and supersets

In a Venn diagram, supersets comprised of regions representing their subsets. Therefore, in the binary representation of a set, each combination of non-zero bits denotes each of its subsets. This relation between binary values representing sets in Q1 is depicted in Figure 3.4. In this graph, directed lines denote the subset-superset relationship where arrow heads are pointed to supersets.

3.4 Expression Extraction

To extract information we employ two approaches and compare the performance of each approach. First method is a rule-based approach where expressions are extracted using regular expressions and filtered using simple heuristics. The second is a statistical approach where expressions like text is selected using a sequential text classifier. We compare several sequential classification approaches and used the best performing classifier for set expressions in our system.

In addition to rule-based and statistical approaches, keyword matching is also used to extract certain information. For example, a problem in category 1 (refer the Table 3.1) may ask to find the least possible value of the cardinality of a set. We maintain a glossaries of words such as “least, minimum" and “maximum, greatest" for each type of expected information.

Consider the problem, Q1 mentioned previously. Information expected to extract from Q1 are the following;

n(ξ) = 40, n(P) = 18, n(Q) = 20,

n(P ∩ Q) = 7, $n(P \cup Q)$, $n(P' \cap Q')$.

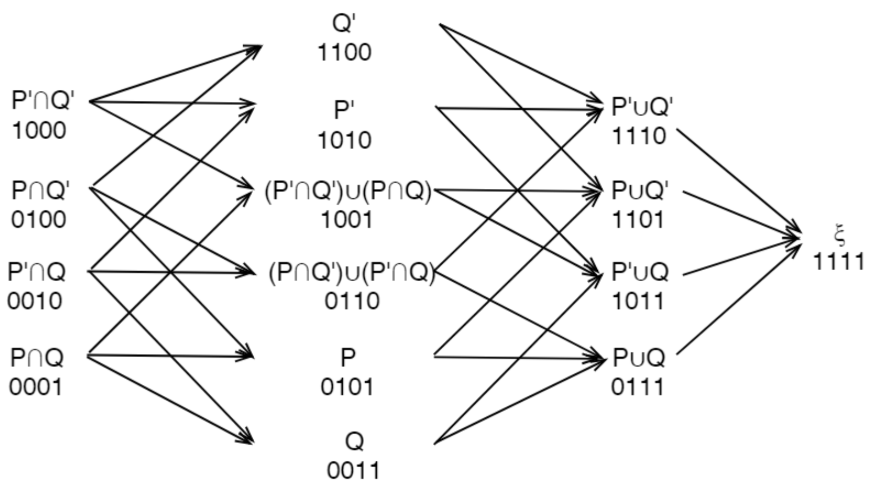


Figure 3.4: The graphical representation of subset-superset relations between all the possible sets in Q1. The binary representation of the sets are written with set names. Edges of the graph point subsets to supersets.

3.4.1 Rule based expression extraction

We used a list of regular expressions to extract sets related expressions. Table 3.3 shows the list of regular expressions used to extract set information from text. We tried to cover as much as possible variations in our rules giving flexibility for different formats. For example, using spaces, allowing multiple-word set names, using 'N' instead of 'n' in cardinality expressions (e.g., '$N(A) = 8$'), and mistakenly using different bracket types (e.g., 'B={1,2,3}' , 'B=[1,2,3]', 'B={1,2,3)'). This flexibility sometimes cause irrelevant text to be extracted by regular expressions such as extracting ordinary bracketed phrases.

In order to filter out irrelevant text, few heuristics are applied. Prior to using regular expressions common keywords that appear before set names are replaced with a special delimiter '#'. Few such common keywords are 'set', 'universal set', 'and' and 'the'. After extracting expressions, set names are separated from equations by considering the left had side of equations. Then the non noun words in the beginning of the set names are removed by considering the POS tags of set names. An example of rule based information extraction is illustrated in the Figure 3.5 considering the following problem (Q2);

Table 3.3: Regular expressions used to extract set information

Regular Expression	**Example**
[Nn][ ]*([\(\[ ]([A−Za−z∩∪ξ\−\(\)\′\ ]+)[\)\]])|\\\|([A−Za−z∩∪ξ\−\(\)\′\ ]+)\\\|	$n(Girls \cup Boys)$, $\lvert P \cup Q\rvert$
[Nn]{0,1}[ ]*(\(([A−Za−z∩∪ξ\−\(\)\′\ ]+)\))[ ]*=[ ]*[−]{0,1}[ ]*\d+|\\\|([A−Za−z∩∪ξ\−\(\)\′\ ]+)\\\|[ ]*=[ ]*[−]{0,1}\d+	$n(Girls{\cup}Boys) = 5$, $n(\xi) = 100$
([A−Za−z∩∪ξ\−\(\)\′\ ]+)[ ]*	$(Girls{\cup}Boys)$
([A−Za−z∩∪ξ\−\(\)\′\ ]+)[ ]*=[ ]*[\{\(\[ ][\w, .]*[\)\}\]]	Colours = {red, blue, green}
([A−Za−z∩∪\′] + [ ]*[⊂⊆][ ]*[A−Za−z∩∪ξ\′]+)|([A−Za−z∩∪ξ\′] + [ ]*[⊃⊇][ ]*[A−Za−z∩∪\′]+)	$A \subset B$
[Nn][ ]*(\(([A−Za−z∩∪ξ\−\(\)\′\ ]+)\))[ ]*=[ ]*[Nn][]*(\(([A−Za−z∩∪ξ\−\(\)\′\ ]+)\))|\\\|[ ]*(([A−Za−z∩∪ξ\−\(\)\′\ ]+))\\\|([ ]*=[ ]*\\\|[ ]*(([A−Za−z∩∪ξ\−\(\)\′\ ]+))\\\|)*	$n(P{\cup}Q){=}n(R)$

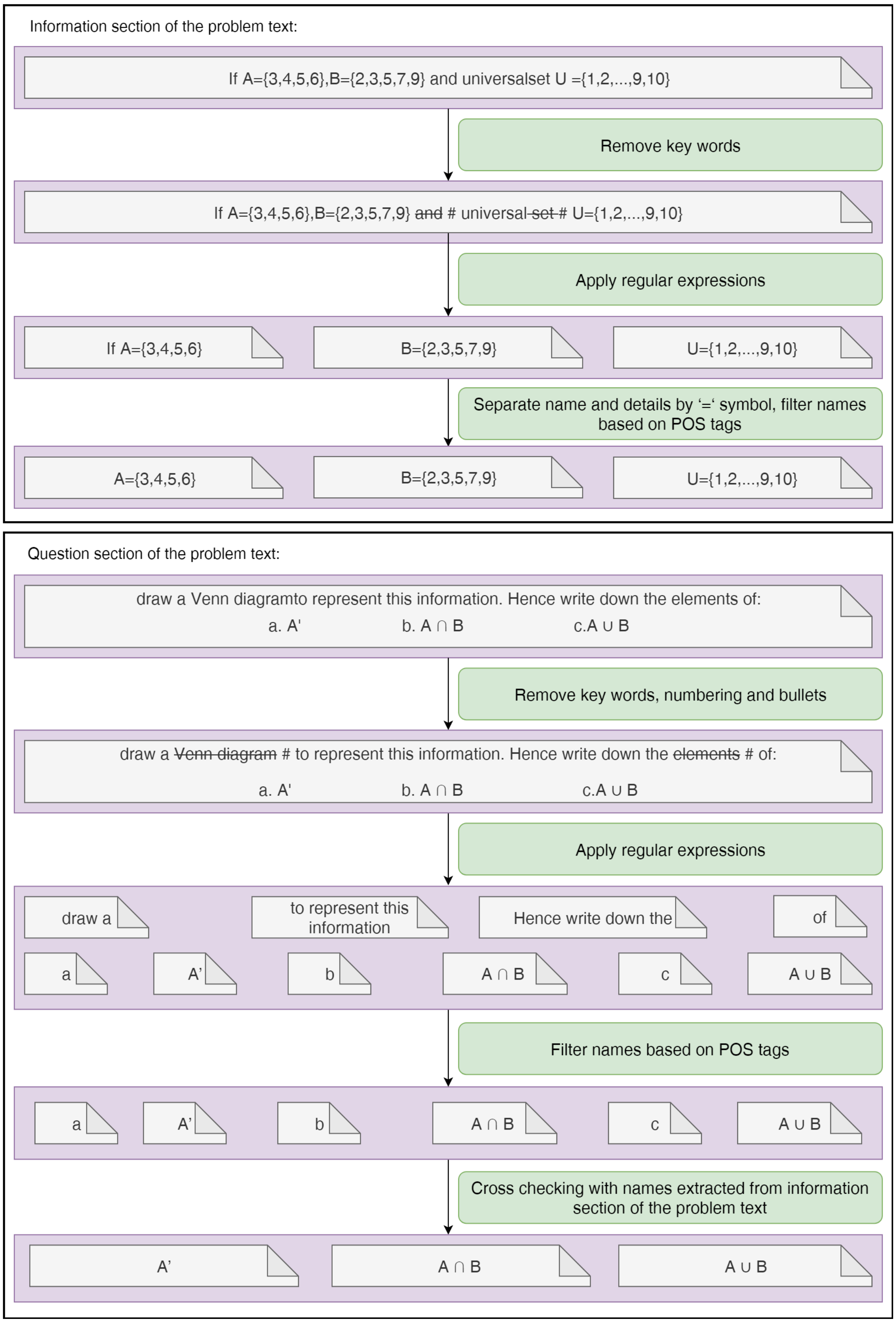
If A={3,4,5,6}, B={2,3,5,7,9} and universal set U={1,2,...,9,10} draw a Venn diagram to represent this information. Hence write down the elements of:

a. A' *b. A ∩ B* *c. A ∪ B*

3.4.2 Limitations of regular expressions

Math expression include many types of tokens such as digits, numbers and operators, delimiters and variables. Variables used in math expression can always be ambiguous to separately identify from ordinary text since variables are often indicated by letters and nouns, which can be misleading. For instance, the token *'a'* in *'Let a be a positive integer'* is both a math expression and a stop word. Digits and letters used to number questions in examination papers can be misinterpreted as relevant numbers or set names that require to solving problems. In addition, years, table or figure labels, and abbreviated names for irrelevant entities are few other occasions that can be misinterpreted as math expressions.

When considering general math expressions, another ambiguity is writing lists of elements to denote separate elements in a list with respect to mathematical



Information section of the problem text:
If A={3,4,5,6},B={2,3,5,7,9} and universalset U ={1,2,...,9,10}
Remove key words
If A={3,4,5,6},B={2,3,5,7,9} ~~and~~ # universal ~~set~~ # U={1,2,...,9,10}
Apply regular expressions
If A={3,4,5,6}
B={2,3,5,7,9}
U={1,2,...,9,10}
Separate name and details by '=' symbol, filter names based on POS tags
A={3,4,5,6}
B={2,3,5,7,9}
U={1,2,...,9,10}
Question section of the problem text:
draw a Venn diagramto represent this information. Hence write down the elements of:
a. A' b. A ∩ B c.A ∪ B
Remove key words, numbering and bullets
draw a ~~Venn diagram~~ # to represent this information. Hence write down the ~~elements~~ # of:
a. A' b. A ∩ B c.A ∪ B
Apply regular expressions
draw a
to represent this information
Hence write down the
of
a
A'
b
A ∩ B
c
A ∪ B
Filter names based on POS tags
a
A'
b
A ∩ B
c
A ∪ B
Cross checking with names extracted from information section of the problem text
A'
A ∩ B
A ∪ B

Figure 3.5: Extracting information from Q2 using rule based approach

sequences. In contrast to a list, sequences should be identified as a single math expression. For example, in the text *"In the sequence 7, 14, 28, x, 112.. what is the value of x?"*, the mathematical expression is the sequence:*' 7, 14, 28, x, 112..'* whereas in *"Find arithmetic mean of the numbers in the list 8 - a, 8, 8 + a"*, math expressions are individual expressions separated by commas.

Regular expressions alone cannot handle any of the above challenges when extracting math expressions. Apart from above mentioned ambiguities in semantic level, there can be syntactic level limitations to extract math expressions by regular expressions when typing errors are present. For example, if $A = \{1, 2\}$ was written with a typo as $A = 1, 2\}$, regular expressions will fail to identify the expression. Another important fact is the start and end of a math expression. An expression that contains words such as *'Let A = total area of five circles of radius r'*, or an expression with typos like *'Let set A={Students of grade five and set B = {Girls in grade five}'* do not have a clear lexical separation from the usual text. It is less practical to predict all the possible typing errors and write flexible regular expressions to capture them.

3.4.3 Expression extraction using statistical approach

The second approach for extracting expressions is using sequential classifier. In addition to the set related problems, we used problems in arithmetic, open vocabulary algebra, closed vocabulary algebra, geometry, sets and few problems from other domains such as probability and data representations. Collected problems were tokenized based on spaces and punctuation marks and annotated for math expressions using the IOB format to prepare the dataset. For example, the sentence *Write down the set* $M \cap N$ is tagged as O O O O B-EXP I-EXP I-EXP O.

Models

We compared several sequential models to extract expressions and selected the best performing model for the solver.

Conditional Random Field (CRF) model :

First models is a Conditional Random Field (CRF) with features listed in Table 3.4. The features were selected based on the works of Finkel et al. [47], Huang et al. [53] which focus on NER for English language using CRF models. Few more features were added which seem relevant to math expressions, and they are distinguished from aforementioned features by the italic font in the Table 3.4).

LSTM network models:

Other models used to extract text based on Long Short Term Memory (LSTM) networks and Bi-directional LSTM (Bi-LSTM) networks. The basic LSTM and Bi-LSTM network architectures were used for the experiment. We used word embeddings created for words in the training dataset with a LSTM and a Bi-LSTM network. These models are referred as W-LSTM and W-Bi-LSTM in this document. The architecture of the W-LSTM and W-Bi-LSTM networks are illustrated in Figure 3.6 and Figure 3.8 respectively. We trained another Bi-LSTM with both word embeddings and character embeddings created for each word in the dataset. We refer this model as W-CH-Bi-LSTM.

3.5 Expressions Parsing

Sets related expressions which are extracted from the problem text are validated using a LL(1) CFG grammar. It covers a wide range of expected set expressions in set notation shown in the Table 3.5.

Grammer rules for parsing set expressions are defined as follows;

<S> ⟶ <Eqn>

<Eqn> ⟶ <Expr> { <equal> {<Expr> | <Rhs> | <emptySet>}}... | <RelationExpr> | <CardEqn> | <Expr>

<Expr> ⟶ <specialSet> | <emptySet> | <AndExpr> [<OrExprRHS>]*>

<Rhs> ⟶ <StatementRhs> | <SetBuilderRhs> | <ElemRhs>

<RelationExpr> ⟶ <SetRelationExpr> <BelongToExpr>

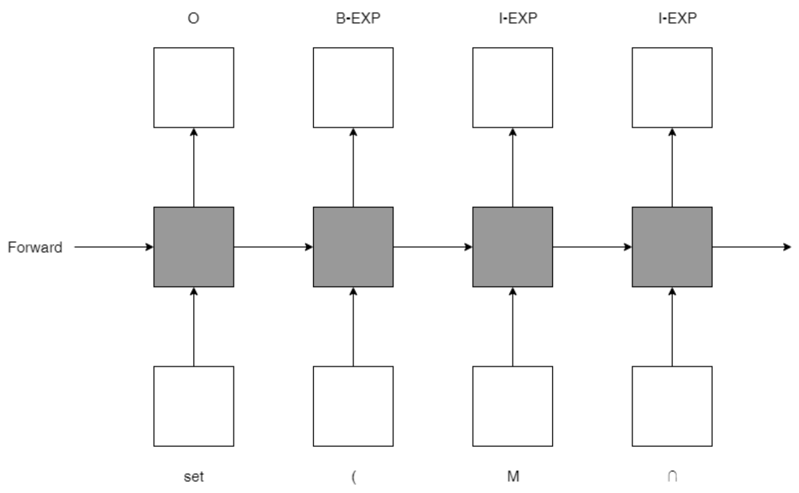


Figure 3.6: Network architecture of the W-LSTM model

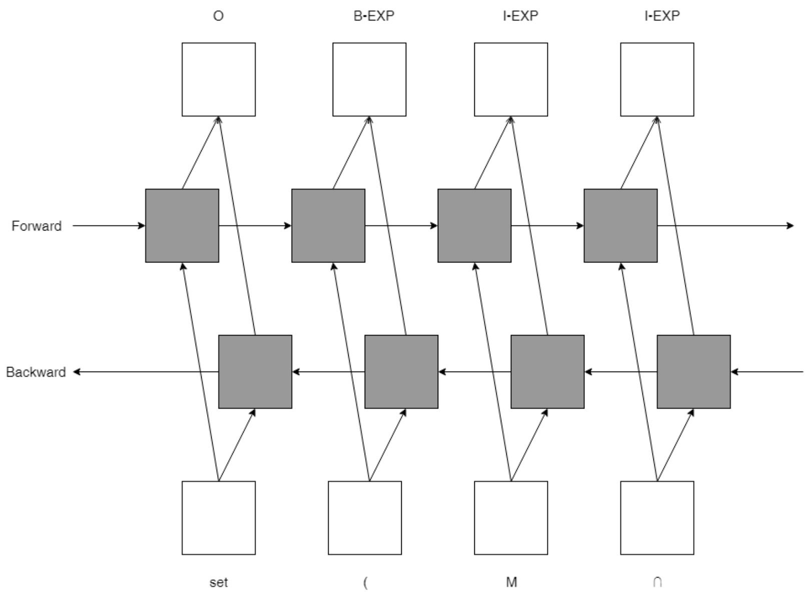


Figure 3.7: Network architecture of the W-Bi-LSTM model

Table 3.4: Features used for CRF divided into categories. The left most column contains a label for each of the set of features

Context features	
A	Uni-grams, bi-grams, tri-grams and their frequencies
Token level features	
B	*Whether the token is a single character*
C	case related features (all upper case, all lower case, starts with capital, contains non-initial capital letters)
D	features related to character type (*contains only digits*, *contains bracket delimiters*, contains only letters, is a mix of digits and letters, contains punctuation marks, word shape, word shape summarization[53])
E	Last two and last three suffixes
Semantic level features	
F	POS tag of the token and surrounding tokens (window size 5)

<CardEqn> ⟶ <CompCardExpr> [<equal> <CompCardExpr>]* [<equal> <number>] [<equal> <CompCardExpr>]* | {<number> { <equal> <CompCardExpr>}*}

<StatementRhs> ⟶ <leftCurlyBrace> <Statement> <rightCurlyBrace>

<SetBuilderRhs> ⟶ <leftCurlyBrace> [<Statement>] [<Expr>] {<separator> <Statement>}... <rightCurlyBrace>

<ElemRhs> ⟶ <leftCurlyBrace> <Element>... <rightCurlyBrace>

<Statement> ⟶ <name> { <name> [<quotes>]*} | <inequality> | <RelationExpr>

<Quotes> ⟶ <leftQuote> <name> < name>... <rightQuote>

<BelongToExpr> ⟶ {<Expr> | <Element>} <belongToOP> <Expr>

<SetRelationExpr> ⟶ <Expr> {<subsetOP> | <supersetOP>} <Expr>

<inequality> ⟶ {<Expr> | <AnyNumber>} {<inequalityOp>, {<Expr> | <AnyNumber>}...

<Element> ⟶ <period>* | <AnyNumber> | <word>

<CardAddExpr> ⟶ <addSubOp> <CardFactor>

<CardFactor> ⟶ <CardPrimary> <mulDivOp> <CardPrimary>

<CardPrimary> ⟶ <CardExpr> | (<CompCardExpr>) | <Number>

<CardExpr> ⟶ <cardNotation> <leftPara> <Expr> <rightPara> | <card-

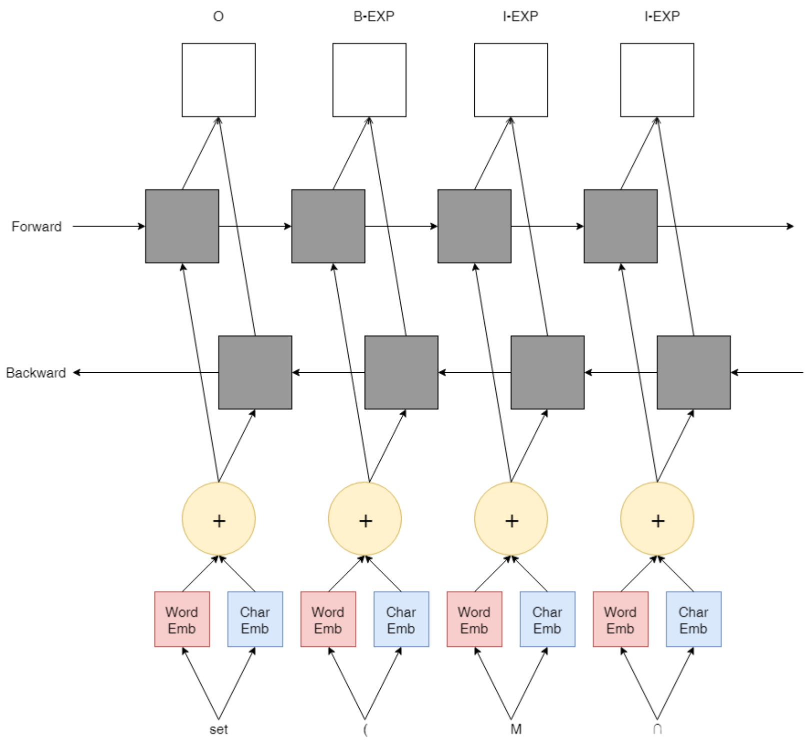


Figure 3.8: Network architecture of the W-CH-Bi-LSTM model

Table 3.5: Examples for set expressions that can be validated by the CFG parser

Expression Type	**Examples**
Set names	A, A∩(B'∪ C), Colours
Enumerated sets	{1,2,3,...,10}, {Nimal, Kamal, Amal}
Set definitions in statements	A={Whole numbers less than 10}, B={letters of "MAHARAGAMA" }
Set definitions in set builder notation	A = {x | x ∈ N }, B {x : x is a multiple of 5 ; 0 < x ≤ 20}
Set definitions in enumerated sets	A∩B = {1,2,3}
Set cardinality expressions	n(A∪B), |A∪B|
Set cardinality equations	n(A∪B')=n(C)=5, |A∪B| = 5
Complex set cardinality expressions	n(A) + n(B∩C)
Complex set cardinality equations	n(A) - n(B∩C) = 10 = n(D)
Set relationships	A ∈ $\mathbb{Z}$, A ⊆ ξ
Set equalities	(P ∩ Q) = ϕ

Notation2> <Expr> <cardNotation2>

<OrExprRHS> $\longrightarrow$ <orDiffOp> <AndExpr>

<AndExpr> $\longrightarrow$ <Primary> | <Complement> | <Rhs> [<andExprOp> <Complement>| <Primary> | <Rhs>]*

<AndExprLHS> $\longrightarrow$ <Primary> | <Complement> | <Rhs>

<Complement> $\longrightarrow$ <Primary> <complementOp>

<Primary> $\longrightarrow$ <leftPara> <Expr> <rightPara> | <word>

Terminals:

<rightQuote> $\longrightarrow$ <complementOp> | ”

<separator> $\longrightarrow$ <cardNotation2> , : , <comma> , ; , , and

<comma> $\longrightarrow$,

<emptySet> $\longrightarrow$ ϕ

<specialSet > $\longrightarrow$ ξ , Z , N

<addSubOp> $\longrightarrow$ $+, -$

<mulDivOp> $\longrightarrow$ $\times$, /

<andExprOp> $\longrightarrow$ $\cap, \setminus$

<orDiffOp> $\longrightarrow$ $\cup$, -

<inequalityOp> $\longrightarrow <, >, \leq, \geq, \neq$

<belongToOP> $\longrightarrow \in, \notin$

<supersetOP> $\longrightarrow \supset, \not\supset, \supseteq, \not\supseteq$

<subsetOP> $\longrightarrow \subset, \not\subset, \subseteq, \not\subseteq$

<complementOp> $\longrightarrow$ ’

<leftQuote> $\longrightarrow$ ‘ , “

<rightPara> $\longrightarrow$)

<leftPara> $\longrightarrow$ (

<rightCurlyBrace> $\longrightarrow$ }

<leftCurlyBrace> $\longrightarrow$ {

<cardNotation1> $\longrightarrow$ n

<cardNotation2> $\longrightarrow$ |

<period> > $\longrightarrow$.

<equal> $\longrightarrow$=

The notation of above grammar rules are as follows;

1. <word> - The variable to denote any alphanumerical string
2. <number> - The variable to denote any unsigned integer
3. ... - one or more
4. * - zero or more
5. [ ] - optional
6. {} - A sequence of symbols that should be considered as an unit

3.6 Mapping to Data Representation

Once set expressions are extracted and validated, they are mapped into the data representation described in Section 3.3. Following are the steps of mapping set expressions.

1. Expand sequences
2. Find universal set
3. Find number of sets
4. Initialize the data structure
5. Find main set indices and store
6. Find derived set indices and store

The first two pre-processing steps are applicable for questions in 2^{nd} category (Table 3.1). First, compressed sequences in set expressions such as '{1,2,3,...,10}' are expanded to get the complete list of elements. Then the universal set is selected among given sets. If a set is denoted using ξ, it is considered as the universal set. Otherwise, the superset of all sets is selected as the universal set. An example for these two steps is shown in Figure 3.9 using Q2 mentioned previously.

Other steps of mapping data into the binary representation is straight forward. First the number of sets are counted and data structure is initiated with null values. Then main set indices are calculated using Equation 3.1. When main sets are stored on the data structure indices of derived sets can be calculated as described in Section 3.3.3. According to the indices, these sets are stored in the

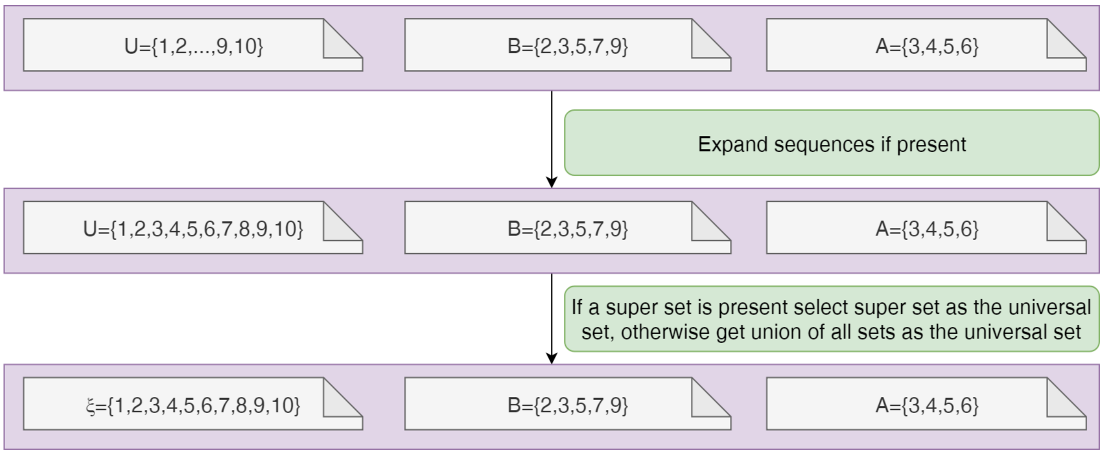

U={1,2,...,9,10}
B={2,3,5,7,9}
A={3,4,5,6}
Expand sequences if present
U={1,2,3,4,5,6,7,8,9,10}
B={2,3,5,7,9}
A={3,4,5,6}
If a super set is present select super set as the universal set, otherwise get union of all sets as the universal set
ξ={1,2,3,4,5,6,7,8,9,10}
B={2,3,5,7,9}
A={3,4,5,6}

Figure 3.9: Finding the universal set in Q2

data structure.

Algorithm 1 shows the algorithm for obtaining an index of a given derived set name. The algorithm takes a set name as the input. Given a set name, say $(P' \cap Q')$, it first converts this into prefix notation. The converted expression of $(P' \cap Q')$ is $(\cap(\neg(P), \neg(Q))$, where $\neg$ denotes the complement operation. This expression is then fed into a stack. Next, an operator and two operands are popped from the stack, calculated the binary representation of the resulting set, and then pushed back into the stack iteratively, until the stack is empty. This process finally results in the binary representation of the given derived set.

3.7 Question Validation

Problems are validated to check following properties.

- Set cardinalities are non-negative
- Set cardinalities are integers
- Cardinalities of subsets are less than supersets

Given a set, all its supersets and subsets can be found as described in Section 3.3.4. If there is an error in the problem, they are reported to the user with appropriate error messages. For example, the error message " *Inconsistent data : Cardinality cannot be negative*" is shown when a negative cardinality is present in the question.

Algorithm 1: Find index of a derived set

```
Input: setName
Output: index corresponding to setName
setOperators ← [‘AND’, ‘OR’ , ‘NOT’]
stack ← stack()
parseTree ← convertToPrefix(setName)
tokens ← tokenize(parseTree).reverse()
tokens ← reverse(tokens)
forall token ∈ tokens do
    if token ∉ setOperators then
        if token is not ‘(’ then
            stack.push(getIndex(token))
    else    /* token is a set operator; get operands of the set
     expression */
        operands = new list
        item= stack.pop()
        while item is not ‘)’ do
            operands.add(item)
            item = stack.pop()
    // compute the value of set expression
    if token = ‘AND’ then
        stack.push (bitWiseAnd(operands))
    else if token = ‘OR’ then
        stack.push (bitWiseOr(operands))
    else
        stack.push (bitWiseComplement(operands))
return stack.top ()
```

3.8 Answer Generation

When a given set problem is provided in the form of the data representation described in section 3.3, answer generation can be done on top of this representation. According to categories of problems that are addressed in this research following questions are expected.

- Find elements of a set, given elements of other sets
- Find cardinality of a set, given cardinalities or elements of other sets
- Find maximum or minimum cardinality of a set, given cardinality or elements of some other sets

When elements of a derived set is been asked answer can be trivially obtained by applying set operations over given participating sets. When it is required to find a cardinality, necessary equations should be generated to solve the problem.

3.8.1 Generating equations and calculating cardinalities

In order to find answers related to cardinality, a system of equations should be generated according to given information. Sum of cardinalities of a set of regions in a Venn diagram is equal to the cardinality of the set that combines those regions. Therefore, based on the region-wise binary representations the equations are generated for each given set in the problem.

Consider the binary representation for sets in a problem with n number of main sets. A variable, c_i is assigned to i^{th} bit position of each binary representation of sets in the problem. c_i denotes the cardinality of the i^{th} region. The bit in i^{th} bit position is b_i where $b_i \in \{0, 1\}$. The equation for the cardinality of a set, c_j where $i \in \{x : x \in \mathbb{N}; 0 \leqslant x < 2^n\}$ and $j \in \{x : x \in \mathbb{N}; 0 \leqslant x < 2^{2^n}\}$ is;

$$c_j = \sum_{i=0}^{2^n} b_i \times c_i \tag{3.2}$$

After obtaining the set of simultaneous equations for each given cardinality information, the equations are solved to find the set of cardinalities, $C = \{c_i; i \in \mathbb{N}; 0 \leqslant i < 2^n\}$ of regions in the corresponding Venn diagram. Now, any unknown cardinality can be found by calculating the sum of cardinalities of regions correspond to the given set. Table 3.6 shows generated equations for sets in Q1. The solution of system of equations generated for Q1 is; $a = 19, b = 11, c = 13, d = 7$. Thus the final answer for Q1 is; $P \cup Q (= b + c + d) = 31$ and $P' \cap Q' (= a) = 19$.

Finding minimum and maximum cardinality of a set

To Find maximum and minimum possible cardinalities we can use the knowledge of supersets and subsets. Algorithm 2 shows how to find the minimum cardinality of a given set. Inputs of the algorithm are, the set to find the minimum cardinality, number of sets, and the set of equations generated for available set information. If a set has subsets, then the minimum possible cardinality of the set is the cardinality of its largest subset. If this is the case, the size of the largest subset (max_subset_size) is returned as the output in the algorithm. If the set does not have any subset, then the minimum possible cardinality is the surplus value obtained when the cardinality of the given set is assumed to be zero. In order to find this surplus value, the algorithm updates the available set of equations by adding a new equation which equates the cardinality of the regarding set to zero. Then the system of equations are solved and the solutions are obtained. Finally, the absolute value of the negative value present in the set of solutions is returned as the minimum cardinality of the given set.

Maximum cardinality of a set is the nearest value to the cardinality of its smallest superset such that it preserves cardinality constrains of the system. It

Table 3.6: Generated equations for sets in Q1

Set Name	**Binary representation of the set**	**Cardinality of the set**	**Corresponding equation / expression**
$P \cap Q$	0001	7	d - 7 = 0
Q	0011	20	c + d - 20 = 0
P	0101	18	b + d - 18 = 0
$P \cup Q$	0111	NA	b + c + d
$P' \cap Q'$	1000	NA	a
ξ	1111	40	a + b + c + d - 40 = 0

means that the cardinality of supersets should be greater than cardinality of subsets. Algorithm 3 finds the maximum cardinality of a given set. Its inputs are, the set to find the maximum cardinality, number of sets and set of equations generated. First, the cardinality of the smallest superset (min_superset_size) is found. Then, in order to find the maximum cardinality of the given set, its cardinality is assumed to be the cardinality of its smallest superset. The equation for the given set is generated by equating its cardinality to min_superset_size and added to the set of available equations. Then the system of equations are solved. Now, the surplus value, or the difference between actual maximum cardinality of the given set and min_superset_size is equal to the negative value present in the set of solutions. Therefore the surplus is found and added to the min_superset_size to get the maximum cardinality of the given set.

Algorithm 2: Find minimum cardinality of a set

```
Input: set, no_of_sets, set_of_equations
Output: minimum cardinality of given set
sub_sets= findSubSetsOf(set.index)
max_subset_size ← max(getSizes(sub_sets))
if sub_sets.length > 0 then
    return max_subset_size
math_expression_for_set ← getMathExpr(set.index, no_of_sets)
assumed_equation_lhs ← math_expression_for_set − 0
set_of_equations_copy ← set_of_equations.add(assumed_equation_lhs)
results ← solveSystemOfEqns(set_of_equations_copy)
forall value ∈ results do
    if value < 0 then
        return abs(value)
return 0
```

Algorithm 3: Find maximum cardinality of a set

```
Input: set, no_of_sets, set_of_equations
Output: maximum cardinality of given set
max ← 0
super_sets= set.index
min_superset_size ← min(getSizes(sub_sets))
math_expression_for_set ← getMathExpr(set.index, no_of_Sets)
assumed_equation_lhs ← math_expression_for_set −
  min_superset_size
set_of_equations_copy ← set_of_equations.add(assumed_equation_lhs)
results ← solveSystemOfEqns(set_of_equations_copy)
surplus_value ← 0
forall value ∈ results do
    if value < surplus_value then
        surplus_value ← value
return min_super_set_size + surplus_value
```

Chapter 4

EVALUATION

This chapter presents evaluation methods of this research. Section 4.1 details the datasets used in the experiment. Section 4.2 presents the parameters of the experiments and important implementation details of the system. The evaluations were carried out for the performance following processes;

1. Math expressions extraction based on sequential classifiers
2. Sets expressions extraction using best performed sequential classifiers and regular expressions
3. Parsing sets expressions
4. Solving a set problem given in data representation described in Section 3.3

Section 4.3 describes each of the above evaluations, results and a discussion about the result respectively.

4.1 Data Sets

We are not aware about any available dataset created for MWPs in set theory. Thus we collected data from past papers of local GCE O/L examination and several international O/L examinations including GCE, IGCSE, GCSE and JCE examinations. Also we collected problems from local school term test papers, mock examination papers, tutorials and other online sources. The problems in Sinhalese were manually translated into English. Most of the questions taken from online educational forums such as algebra.com[1], there were many typing errors present in problems. We evaluated our system with both corrected problems and original problems. We created four datasets from sets related data as depicted in table 4.1.

In order to train sequential models to extract math expressions, we used above

[1]`https://www.algebra.com/algebra/homework/sets-and-operations/sets-and-operations.faq`

mentioned dataset combined with the SemEval-2019 task-10[2] dataset. It contains problems in mathematical domains such as elementary level arithmetic, open and closed vocabulary algebra, geometry and other domains such as probability and data representations. We refer this dataset as 'DExpr'. Table 4.2 shows statistics of the dataset, DExpr used for expression extraction.

4.2 Experimental Setup

This section first explains the experimental setting of math expressions extraction using sequential classifiers. Then the important implementation details of the complete solver system are also described.

The dataset, DExpr introduced in the Section 4.1 was used in sequential classification models. It was randomly split into training, validation and test sets in 8:1:1 ratio to train sequential classifiers. This being the first research conducted for expression extraction using RNNs, we used the standard feed-forward LSTM and Bi-LSTM models for first two settings (W-LSTM and W-Bi-LSTM), which comprised a word embedding layer where the 50-length output is subjected to a 10% dropout to avoid overfitting, one LSTM layer and an output layer with softmax normalization. The third setting (W-CH-Bi-LSTM) comprises an additional embedding layer for characters before the Bi-LSTM layer. Prior to the training we experimented with different optimizers (standard SGD, Adam

[2]`https://github.com/allenai/semeval-2019-task-10`

Table 4.1: Datasets used to evaluate solver system

Dataset	**Content**	**number of problems**
DC1-E	Problems of category 1 (Table 3.1) with typing errors	65
DC1-NE	Problems of category 1 without typing errors	64
DC2-E	Problems of category 2 with typing errors	54
DC2-NE	Problems of category 2 without typing errors	54

Table 4.2: Statistics of problems in the dataset, DExpr

Category	**Number of problems**	**Number of expressions**	**Number of tokens**	**Number of expression tokens**
Closed-algebra	1088	3832	29541	10886
Open-algebra	360	1059	16332	1372
Geometry	702	2351	22677	4288
Elementary set theory	487	2419	31380	16308
Other	86	124	3634	156
Uncata-gorized	528	1717	22796	3219

and RMSProp), dropout rates (10%, 20%, 50%) different batch sizes (10, 32, 50 100) and epochs (10, 20, 50, 100, 500 and 1000) and selected batch size of 10, 10% dropout and RMSProp optimizer with 0.001 learning rate to train the models.

In the solver system, we use an array of size 2^{2^n} to store set information of each set given in the problem, according to the binary representations of sets mentioned in section 3.3. The array index correspond to the decimal value of the binary representation of sets. The system is separately evaluated for both rule based and statistical based expression extractors. The best performing sequential classifier for set expressions was selected as the statistical model to extract set information from problem text. The CFG parser described in the Section 3.5 is implemented using a python library named pyPEG2[3] which supports writing custom expression parsers. The grammar rules are listed in the Section 3.5. We used the library 'boolean.py'[4] to convert a given expression into a parsed-tree string in prefix notation. To generate the system of equations to solve the problem, we used Sympy[57], a well-known python library that supports symbolic mathematics.

[3]`https://pypi.org/project/pyPEG2/`

[4]`https://github.com/bastikr/boolean.py`

4.3 Evaluations

All the components of the set problem solving system is evaluated for each of the datasets listed in the Table 4.1. In addition to that a dataset of 96 problems was used to training and validation testing of the system. General math expression extraction using sequential classifiers were trained and evaluated using the dataset DExpr.

4.3.1 Evaluation of math expressions extraction based on sequential classifiers

The accuracy, precision, recall and F1-score of the CRF model and RNN models were calculated based on identifying complete math expressions correctly. In order to calculate aforementioned accuracy matrices true-positives (TP), false-positives (FP) and false-negatives (FN) were calculated for each model as follows. Given the set of expected expressions, E, and the set of predicted expressions, P;

TP = {Expressions in both E and P }

FP = {Expressions in P but not in and E }

FN = {Expressions in E but not in P }

The text which is correctly predicted as expressions is counted as a true positive, text which is not an expression but predicted as an expression is counted as a false positive, and text which is an expression that is not identified as an expression is counted as a false negative.

The results of the CRF model against cumulatively added features are presented in the graph in Figure 4.1. It shows that character-based features such as containing digits or only letters and suffixes of the tokens contributes to the performance in a statistically significant rate.

W-LSTM, W-Bi-LSTM and W-CH-Bi-LSTM models were evaluated on the same test set as the CRF model. Figure 4.2 shows the accuracy of each model against the number of epochs used to train the model. The W-CH-Bi-LSTM performs much better than other two models. The W-Bi-LSTM model performs better than W-LSTM model, but it takes more number of epochs to achieve high

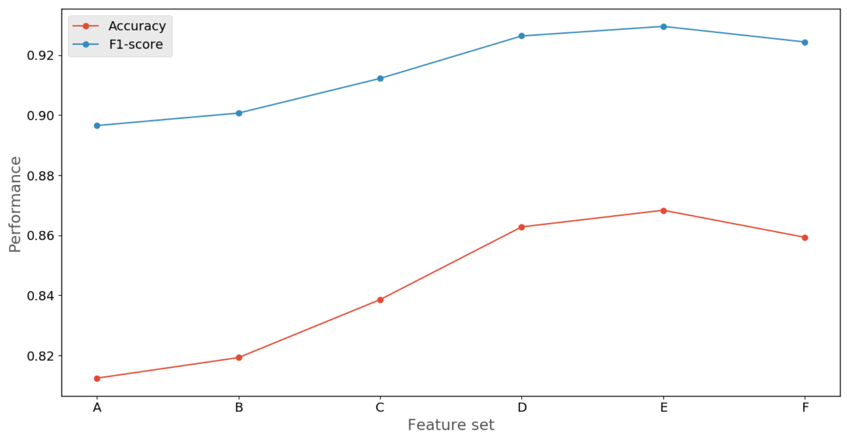


Figure 4.1: CRF performance against cumulatively added feature sets from the set A to set G listed in Table 3.4

accuracy levels.

Table 4.3 shows the best results of all sequential models. Given the dataset is small-sized, the CRF model also performs equally the W-Bi-LSTM model.

4.3.2 Evaluation of set expressions extraction

This evaluation compares the best performing statistical model and rule based extractor for set expression extraction. Statistical models were trained for extracting general math expressions. When these models were applied for sets expression extraction the CRF model performed better due to less amount of data for only set related expressions. Therefore, CRF model were used as the statistical set expression extractor. Both statistical and rule-based extractor were tested for each dataset mentioned in the Table 4.1

Rule based extractor always extracts expressions that exactly matches to the defined rules. Therefore, in DC1-E and DC1-NE, the precision of the rule-based extractor is 1. In contrast, the statistical extractor extracts all possible expressions, therefore shows a higher recall rate and lower precision and accuracy rates. Another important observation on extracted expressions are that the statistical extractor extracts other symbols combined with expressions. For example, consider expressions extracted only by statistical extractor; '$n(A \cap B) =$ ________', '$\backslash u200bn(A \cap B') = \backslash u200b(C)$' and '$n(B) = 21|$'. The first expres-

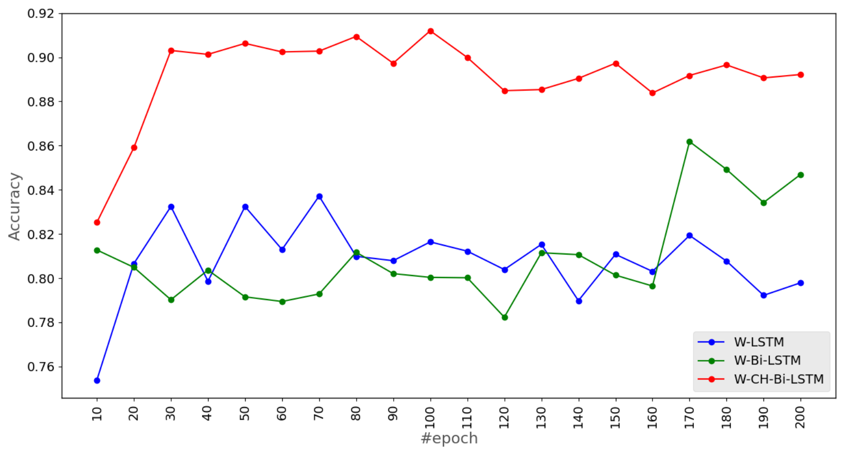


Figure 4.2: Performance of RNN models with respect to number of epochs

Table 4.3: Accuracy, Recall, Precision and F1-score of the best performance of all models

Model	**Acc.**	**Recall**	**Prec.**	**F1**
CRF	0.866	0.931	0.923	0.927
W-LSTM	0.837	0.922	0.901	0.911
W-Bi-LSTM	0.862	0.911	0.941	0.926
W-CH-Bi-LSTM	**0.912**	**0.959**	**0.949**	**0.954**

sion has an underline to write the answer, which is captured in statistical parser, but filtered out by the rule-based parser. The second expression includes $\backslash u200bn$, the zero length white space character, and the third expression includes '|' which is a typo. While extracting first and last expressions reduce the accuracy, they show the robustness of the statistical expression extractor to extracting unseen characters that are related to expressions.

Expressions in category-2 (Table 3.1) are more difficult to extract with respect to expressions of category-1, because category-1 expressions are surrounded with $n()$ whereas expressions in category-2 do not have such an obvious boundary. Therefore, the rule based extractor shows lower results than the statistical extractor in both DC2-E and DC2-NE datasets.

Table 4.4: Performance of statistical and rule-based expression extractors

Dataset	**Results**					
	Rule-based Extractor			**Statistical Extractor**		
	Accuracy	**Precision**	**Recall**	**Accuracy**	**Precision**	**Recall**
DC1-E	**0.962**	**1**	0.962	0.938	0.968	**0.968**
DC1-NE	**0.979**	**1**	0.979	0.964	0.982	**0.982**
DC2-E	0.705	0.834	**0.821**	**0.727**	**0.878**	0.809
DC2-NE	0.821	0.93	0.875	**0.835**	**0.94**	**0.882**

4.3.3 Parsing sets expressions

Given all expressions including correct and incorrect ones in each DC1-E and DC2-E datasets, the performance of the parser was evaluated according to the expression validation accuracy. Given the set of Correct expressions, CE, and incorrect expressions IE, true-positives (TP), false-positives (FP) and false-negatives (FN) were calculated as follows;

TP = {Expressions in CE identified as correct expressions}

TN = {Expressions in IE identified as incorrect expressions}

FP = {Expressions in IE identified as correct expressions }

FN = {Expressions in CE identified as incorrect expressions}

Results of the both rule based and statistical expression extractors are combined to perform the parser. The statistics of these data are shown in the Table4.5. Table 4.6 shows the evaluation results of the parser.

The expression parser fails when incorrect expressions are ambiguous with names. For example, (AB) is an erroneous expression of $(A \cup B)$; but the parser recognize it as correct since AB can also be a name. Other failures include unexpected expressions (e.g., $C \cap \phi$) and using different symbols (e.g., using $\bigcap$ instead of $\cap$).

Table 4.5: Statistics of the data used to evaluate the parser

Dataset	**# Correct Expressions**	**# Incorrect Expressions**
DC1-E	310	36
DC2-E	336	45

Table 4.6: Evaluation results of the parser

Dataset	**Results**		
	Accuracy	**Precision**	**Recall**
DC1-E	0.948	0.954	0.990
DC2-E	0.936	0.984	0.942

4.3.4 Solving a set problem

The solver component was evaluated using problems in the dataset DC1-NE and DC2-NE (refer Table 4.1). In order to conduct an independent evaluation problems that include expressions that expression extractors and parser could not handle properly were eliminated. The statistics of resulting datasets are mentioned in the Table 4.7. Evaluation results of the solver is detailed in the Table 4.9.

Table 4.7: Statistics of the data used to evaluate the solver

Dataset	**# Problems**
DC1-NE2	66
DC2-NE2	51

Table 4.8: Evaluation results of the solver

Dataset	**Accuracy**
DC1-NE2	100%
DC2-NE2	100%

4.3.5 End-to-end performance

This section presents the results of the complete system.

Table 4.9: Evaluation results of the complete system

Dataset	**Accuracy with unseen problem types**	**Accuracy without unseen problem types**
DC1-E	93.85%	96.83%
DC1-NE	95.31%	98.39%
DC2-E	85.19%	91.84%
DC2-NE	85.19%	93.88%

Chapter 5

CONCLUSION

This research takes the first step towards automatically answering set related MWPs. As a result, a complete system that understand and answer problems that include expressions written in set notation in O/L mathematics examinations.

We provide a categorization for set problems that are commonly presented in O/L examinations and introduce a dataset of 220 set problems that belongs to two identified categories. Facilitating any set related problems solver, an abstract data structure is introduced in this research, that captures mathematical semantics of set expressions in basic set theory. This representation, which adapt the binary representation used in Knuth [10], can cover all concepts and relationships that can be represented in a Venn diagram. Extracting sets related expressions were experimented with two approaches; a rule-based and statistical approach. The results show that the statistical extractor is robust for unseen expression types (e.g., expressions with typing errors) and symbols. A CFG parser is used to validate expressions. A feedback is given to the user when there are invalid expressions. We present a complete solver systems which use sets related information mapped into the above mentioned representation to validate problems, generate equations and find the solution of the problem.

This research narrows down solving any O/L sets related problem expressed in natural language to a problem of representing given information using set notation. Once expressions in set notation are provided our system can validate expressions, validate the problem and generate answer of the problem. It also provides a baseline for set problem solvers.

In addition we experimented with several sequential classifiers for extracting general math expressions from unstructured plain text. Results show that using character level features benefit in increasing accuracy of this text classification task. An annotated dataset with 102K tokens and 9K math expressions is also introduced for math problems in several domains such as algebra, geometry and basic set theory.

5.1 Future Work

In future work we try to present the solver with an API and enhanced features to be used in problem solvers for basic set theory. Expression extraction, being the major bottleneck of the system accuracy should be further enhanced by training with more data for the statistical approach. We hope to experiment more in general math expression extraction using better methods and more data.

Ultimately, the system presented in this thesis leads to a system that can solve any set related problem in O/L examinations. There are more set problem categories that is not addressed in this research, but can be solvable using the abstract data representation presented here. Therefore we can widen the system to answer problems in other categories including problems where all information is presented in natural language and problems that presents information through Venn diagrams.

References

[1] Mohammad Javad Hosseini, Hannaneh Hajishirzi, Oren Etzioni, and Nate Kushman. Learning to solve arithmetic word problems with verb categorization. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 523–533, 2014.

[2] Nate Kushman, Yoav Artzi, Luke Zettlemoyer, and Regina Barzilay. Learning to automatically solve algebra word problems. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 271–281, 2014.

[3] Bussaba Amnueypornsakul and Suma Bhat. Machine-guided solution to mathematical word problems. In *Proceedings of the 28th Pacific Asia Conference on Language, Information and Computing*, 2014.

[4] Subhro Roy and Dan Roth. Illinois math solver: math reasoning on the web. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Demonstrations*, pages 52–56, 2016.

[5] Danqing Huang, Shuming Shi, Chin-Yew Lin, and Jian Yin. Learning fine-grained expressions to solve math word problems. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 805–814, 2017.

[6] Minjoon Seo, Hannaneh Hajishirzi, Ali Farhadi, Oren Etzioni, and Clint Malcolm. Solving geometry problems: Combining text and diagram interpretation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1466–1476, 2015.

[7] Anton Dries, Angelika Kimmig, Jesse Davis, Vaishak Belle, and Luc De Raedt. Solving probability problems in natural language. In *Proceed-*

ings Twenty-Sixth International Joint Conference on Artificial Intelligence, pages 3981–3987, 2017.

[8] Diunuge B Wijesinghe, Surangika Ranathunga, and Gihan Dias. Computer representation of venn and euler diagrams. *Advances in ICT for Emerging Regions*, pages 100–105, 2016.

[9] Diunuge Buddhika Wijesinghe, Jcs Kadupitiya, Surangika Ranathunga, and Gihan Dias. Automatic assessment of student answers consisting of venn and euler diagrams. In *2017 IEEE 17th International Conference on Advanced Learning Technologies (ICALT)*, pages 243–247. IEEE, 2017.

[10] Donald Ervin Knuth. The art of computer programming volume 4, pre-fascicle 0b. a draft of section 7.1. 2: Boolean evaluation, 2007.

[11] Daniel G Bobrow. Natural language input for a computer problem solving system. 1964.

[12] Anirban Mukherjee and Utpal Garain. A review of methods for automatic understanding of natural language mathematical problems. *Artificial Intelligence Review*, 29(2):93–122, 2008.

[13] Sourav Mandal and Sudip Kumar Naskar. Solving arithmetic mathematical word problems: A review and recent advancements. In *Information Technology and Applied Mathematics*, pages 95–114. Springer, 2019.

[14] Mark Steedman and Jason Baldridge. Combinatory categorial grammar. *Non-Transformational Syntax: Formal and Explicit Models of Grammar*, pages 181–224, 2011.

[15] Takuya Matsuzaki, Hidenao Iwane, Hirokazu Anai, and Noriko H Arai. The most uncreative examinee: A first step toward wide coverage natural language math problem solving. In *AAAI*, pages 1098–1104, 2014.

[16] Takuya Matsuzaki, Takumi Ito, Hidenao Iwane, Hirokazu Anai, and Noriko H Arai. Semantic parsing of pre-university math problems. In *Pro-*

ceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), volume 1, pages 2131–2141, 2017.

[17] Shuming Shi, Yuehui Wang, Chin-Yew Lin, Xiaojiang Liu, and Yong Rui. Automatically solving number word problems by semantic parsing and reasoning. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1132–1142, 2015.

[18] Subhro Roy. *Reasoning about quantities in natural language*. PhD thesis, University of Illinois at Urbana-Champaign, 2017.

[19] Lipu Zhou, Shuaixiang Dai, and Liwei Chen. Learn to solve algebra word problems using quadratic programming. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 817–822, 2015.

[20] Danqing Huang, Shuming Shi, Chin-Yew Lin, Jian Yin, and Wei-Ying Ma. How well do computers solve math word problems? large-scale dataset construction and evaluation. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 887–896, 2016.

[21] Shyam Upadhyay and Ming-Wei Chang. Annotating derivations: A new evaluation strategy and dataset for algebra word problems. *arXiv preprint arXiv:1609.07197*, 2016.

[22] Rik Koncel-Kedziorski, Hannaneh Hajishirzi, Ashish Sabharwal, Oren Etzioni, and Siena Dumas Ang. Parsing algebraic word problems into equations. *Transactions of the Association for Computational Linguistics*, 3:585–597, 2015.

[23] Subhro Roy and Dan Roth. Solving general arithmetic word problems. *arXiv preprint arXiv:1608.01413*, 2016.

[24] Subhro Roy and Dan Roth. Unit dependency graph and its application to arithmetic word problem solving. *arXiv preprint arXiv:1612.00969*, 2016.

[25] Arindam Mitra and Chitta Baral. Learning to use formulas to solve simple arithmetic problems. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 2144–2153, 2016.

[26] Suleyman Cetintas, Luo Si, Yan Ping Xin, Dake Zhang, and Joo Young Park. Automatic text categorization of mathematical word problems. In *FLAIRS Conference*, 2009.

[27] Pruthwik Mishra, Litton J Kurisinkel, and Dipti Misra Sharma. Arithmetic word problem solver using frame identification. *arXiv preprint arXiv:1808.03028*, 2018.

[28] Chao-Chun Liang, Shih-Hong Tsai, Ting-Yun Chang, Yi-Chung Lin, and Keh-Yih Su. A meaning-based english math word problem solver with understanding, reasoning and explanation. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: System Demonstrations*, pages 151–155, 2016.

[29] Eugene Charniak. Carps: a program which solves calculus word problems. 1968.

[30] Bruce W Ballard and Alan W Biermann. Programming in natural language:âĂIJnlcâĂİ as a prototype. In *Proceedings of the 1979 annual conference*, pages 228–237. ACM, 1979.

[31] Johan De Kleer. Qualitative and quantitative knowledge in classical mechanics. 1975.

[32] Alan Bundy, George Luger, M Stone, and Robert Welham. Mecho: Year one. In *Proceedings of the 2nd Summer Conference on Artificial Intelligence and Simulation of Behaviour*, pages 94–103. IOS Press, 1976.

[33] Gordon S Novak Jr. Computer understanding of physics problems stated in natural language.(dissertation), also technical report nl-30. 1976.

[34] GE Oberem. Albert: a physics problem solving monitor and coach. In *Proceedings of the first international conference on computer assisted learning (ICCALâĂŹ87). Calgary Alberta, Canada*, pages 179–184, 1987.

[35] Xiaorong Huang. Human oriented proof presentation: A reconstructive approach. 1999.

[36] Kyle Morton and Yanzhen Qu. A novel framework for math word problem solving. *International Journal of Information and Education Technology*, 3(1):88, 2013.

[37] Rik Koncel-Kedziorski, Subhro Roy, Aida Amini, Nate Kushman, and Hannaneh Hajishirzi. Mawps: A math word problem repository. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1152–1157, 2016.

[38] Suleyman Cetintas, Luo Si, Yan Ping Xin, Dake Zhang, Joo Young Park, and Ron Tzur. A joint probabilistic classification model of relevant and irrelevant sentences in mathematical word problems. *arXiv preprint arXiv:1411.5732*, 2014.

[39] Xuedong Tian, Ruihan Bai, Fang Yang, Jinyuan Bai, and Xinfu Li. Mathematical expression extraction in text fields of documents based on hmm. *Journal of Computer and Communications*, 5(14):1, 2017.

[40] Oren Etzioni, Michael Cafarella, Doug Downey, Ana-Maria Popescu, Tal Shaked, Stephen Soderland, Daniel S Weld, and Alexander Yates. Unsupervised named-entity extraction from the web: An experimental study. *Artificial intelligence*, 165(1):91–134, 2005.

[41] Sean R. Eddy. Profile hidden markov models. *Bioinformatics (Oxford, England)*, 14(9):755–763, 1998.

[42] Daniel M Bikel, Scott Miller, Richard Schwartz, and Ralph Weischedel. Nymble: a high-performance learning name-finder. *arXiv preprint cmp-lg/9803003*, 1998.

[43] J. Ross Quinlan. Induction of decision trees. *Machine learning*, 1(1):81–106, 1986.

[44] Jagat Narain Kapur. *Maximum-entropy models in science and engineering*. John Wiley & Sons, 1989.

[45] Marti A. Hearst, Susan T Dumais, Edgar Osuna, John Platt, and Bernhard Scholkopf. Support vector machines. *IEEE Intelligent Systems and their applications*, 13(4):18–28, 1998.

[46] John Lafferty, Andrew McCallum, and Fernando CN Pereira. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. 2001.

[47] Jenny Rose Finkel, Trond Grenager, and Christopher Manning. Incorporating non-local information into information extraction systems by gibbs sampling. In *Proceedings of the 43rd annual meeting on association for computational linguistics*, pages 363–370. Association for Computational Linguistics, 2005.

[48] David Nadeau and Satoshi Sekine. A survey of named entity recognition and classification. *Lingvisticae Investigationes*, 30(1):3–26, 2007.

[49] Rahul Sharnagat. Named entity recognition: A literature survey. *Center For Indian Language Technology*, 2014.

[50] Vikas Yadav and Steven Bethard. A survey on recent advances in named entity recognition from deep learning models. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 2145–2158, 2018.

[51] Jing Li, Aixin Sun, Jianglei Han, and Chenliang Li. A survey on deep learning for named entity recognition. *arXiv preprint arXiv:1812.09449*, 2018.

[52] Wenpeng Yin, Katharina Kann, Mo Yu, and Hinrich Schütze. Comparative study of cnn and rnn for natural language processing. *arXiv preprint arXiv:1702.01923*, 2017.

[53] Zhiheng Huang, Wei Xu, and Kai Yu. Bidirectional lstm-crf models for sequence tagging. *arXiv preprint arXiv:1508.01991*, 2015.

[54] Jason PC Chiu and Eric Nichols. Named entity recognition with bidirectional lstm-cnns. *Transactions of the Association for Computational Linguistics*, 4:357–370, 2016.

[55] Wang Ling, Tiago Luís, Luís Marujo, Ramón Fernandez Astudillo, Silvio Amir, Chris Dyer, Alan W Black, and Isabel Trancoso. Finding function in form: Compositional character models for open vocabulary word representation. *arXiv preprint arXiv:1508.02096*, 2015.

[56] John Venn. I. on the diagrammatic and mechanical representation of propositions and reasonings. *The London, Edinburgh, and Dublin philosophical magazine and journal of science*, 10(59):1–18, 1880.

[57] Aaron Meurer, Christopher P Smith, Mateusz Paprocki, Ondřej Čertík, Sergey B Kirpichev, Matthew Rocklin, AMiT Kumar, Sergiu Ivanov, Jason K Moore, Sartaj Singh, et al. Sympy: symbolic computing in python. *PeerJ Computer Science*, 3:e103, 2017.