Generating Word Problems Of Arbitrary Complexity

Gautham B A
Dept. of Computer Science
PES Institute of Technology
Bangalore, Karnataka, India
gautham.bangalore@gmail.com

Aditya Agarwal
Dept. of Computer Science
PES Institute of Technology
Bangalore, Karnataka, India
skymanaditya1@gmail.com

Viraj Kumar
Dept. of Computer Science
PES University
Bangalore, Karnataka, India
viraj.kumar@pes.edu

***Abstract*—The proposed method depicts how word problems can be generated by taking a non-singular system of linear equations and inspecting the solution. The corresponding English statements obtained can be treated as a "figurative" translation of the non-singular system as an equation like x+y=4 may be understood as the process of adding 2 quantities which produce a result of 4. We call it as a "figurative" translation as the two quantities 'x' and 'y' can be mapped to any entity but the semantics of the equation is always retained. This forms the basis of problem generation. The proposed method defines a set of regular expressions using which the linear equations are translated into English statements. The non-singular system of linear equations must undergo a normalization phase in order to apply the regular expressions. The complexity of the word problem that is generated is a function of the nature of the system of linear equations and hence is arbitrary. The number of linear equations in the system and the diversity in the signs of the terms contribute to the complexity of the generated problem. The proposed method provides a systematic procedure which can be employed to generate word problems from a given non-singular system of linear equations. The proposed method has been implemented as a pedagogical tool by capturing the transformations that the equations undergo and portraying the changes as a video play, which would provide a keen insight on the dynamics that are in play when one attempts to solve such a system.**

Keywords—non-singular system, linear equations, regular expression, normalization, complexity.

I. Introduction

There exists a pattern in a given system of linear equations. The process of obtaining the solution only involves recognizing the pattern and performing the corresponding actions. But the process of understanding the semantics of a linear equation involves mapping the variables in the equation to entities. In the process of obtaining the equivalent word problem, the aspect of choosing the right entity for the variables in the equation is paramount, as the goal of the proposed method also involves the solution and the word problem to be coherent. Thus, the hierarchy of classification of entities is discussed.

The mapping of variables to entities is obtained by inspecting the solution of the system. After choosing the right entity to map the variables the subsequent task is to normalize system of equations which involves translocating the terms of the equation so that the equations are completely free from the negative sign. This is a necessary step which has to be performed in order to apply the regular expression. After normalization comes the task of choosing the right regular expression for translation.

II. Literature survey

A. *Using the Natural Language Generation tool for generation syntactically correct multi-sentence MWP questions.[1]*

Generating mathematical word problems from ontologies in unrestricted domains. It builds on an existing ontology verbalizer that renders logical statements written in Web Ontology Language (OWL) as English Sentences.
Natural Language Generation (NLG) has already been applied to the problem of generating single-sentence multiple choice questions.
Generating mathematical word problems (MWPs), the questions are expressed as multi-sentence short narratives in which some numerical information is given, they end with a request for the student to calculate an unknown quantity. MWP questions may also contain additional 'distractor' numerical values that make identifying the ones required for the arithmetic operation more difficult. The method also proposes to ensure the generation of semantically valid statements.
Demonstrates a method by which MWP questions may be generated from on ontology that contains OWL Data Property Assertion statements (axioms) with literal values that are integers.
The paper also claims to be able to vary automatically -

i) The readability of the output text
ii) Include distractor numerical values
iii) Introduce extraneous information into the narrative,
iv) Modify the order of presentation of numerical values
v) Introduce conceptual difficulty of the mathematical problem.

Readability is controlled by increasing or decreasing the length of the MWP. Distractor numerical values can be added provided that an ontology contains additional data properties to increase difficulty.

The extraneous information may be removed so that the question is simplified by stripping it down to essentials.
The paper also aims to generate sentences in different orders to modify the difficulty of a MWP.
The paper also makes a seemingly easy question difficult to solve by requesting additional calculations to be performed.
The system has not yet been tested within a suitable intelligent tutoring system and with users.
The proposed method handles literal values other than integers as well which is a serious limitation in the method described above. The regular expressions ensure semantic correctness even when real numbers are used as literal values.
Our model is designed to handle units of measurement for literal values and ensuring semantic correctness. The regular expressions are carefully designed to support the inclusion of units in our MWPs and to prevent generation of erroneous or mathematically invalid statements.
We have tested the implementation of the proposed method with the students of 10th grade and improvement was witnessed in the approach that the students employed for finding the solution.

B. Personalized Mathematical Word Problem Generation.[2]

[2] Proposes a novel technique for automatic generation of personalized word problems. Word problems are generated from general specifications using answer-set programming (ASP). The specifications include tutor requirements and student requirements and takes a logical encoding of the specification, synthesizes a word problem narrative graph, and realizes the problem in natural language.
It also claims to have the following properties:

- It is automatic: a mathematical model, a plot, and a discourse of a word problem are generated automatically from general specifications.
- It is personalized: students can set preferences for a problem's setting, characters, and their relationships.
- It is sensible: the system enforces coherence in a synthesized plot using a novel technique called discourse tropes.
- It is fir for scaffolding: varying requirements to different layers of a word problem enables a tutor to scaffold a unique educational progression.

Synthesis of logical graphs is implemented with Answer Set Programming (ASP) and also includes a Natural Language Generation (NLG) module for generating a textual representation of a synthesized logical graph. To reduce the impact of poor language clarity on problem complexity, the system includes an algorithm for unambiguous and non-repetitive reference resolution in problem text.
[2] Claims to build templates automatically with respect to the set of requirements, provided by students and tutors. These templates are generalization of existing problems and are built by exploring a space of solutions that fit this template.
The system uses ASP for automatic generation of a word problem. It also uses a saturation technique for automatic verification of a universally quantified property. The system for word problem generation procedure takes as input a set of requirements R (student and tutor requirements) to the problem and produces as output a textual representation of the problem. The plot and mathematical model satisfy all the requirements as specified by student and tutor.
The process of generating word problems consist of two phases: logic generation and natural language generation. The logic generation builds a mathematical model of the problem, taking into account tutor requirements. Given a specific equation E, the system generates a problem plot. The NLG phase takes a generated logical graph, and realizes it into a concrete textual representation.
Equation generation works by encoding an equation E as an expression tree. The encoding first guesses a tree shape and an assignment of mathematical operators to internal binary nodes. Then it deduces whether every mathematical requirement in tutor requirements is covered by some sub expression of the guessed tree. Finally, it forbids the solutions that do not cover all of the requirements of tutor or do not represent a valid equation.
A logical graph G models an equation E, if its subgraph is isomorphic to the expression tree of E. So the system for an equation E, students requirements R_s and an ontology generates any logical graph G from types, relations and tropes in the ontology, such that it models the equation E and fits the requirements of the tutor.
The Natural Language Generation phase works by realizing a logical graph of a problem into textual representation. It first approximates the text with sentences produced using primitive templates, and then it orders sentences produced by templates, resolves entity references unambiguously and non-repetitively, and realizes the result into valid English.
Evaluation – An ontology of 100-200 types was prepared, relations, and tropes in three literary settings: “Fantasy”, “Science Fiction” and “School of Wizardry”. Randomly generated 25 problems in the domains of age, counting, and trading, with the solutions requiring 2-4 primitive arithmetic operations. Each problem was generated in less than 60 seconds.
The proposed method tries to fall into the tracks of [2] by trying to maintain the context-sensitiveness of the problem. It will produce the English statements that “figuratively” translate to the equations that were given as input. The proposed method differs from [2] in a manner that it doesn't rely on “tutor” to specify the nature of the problem to be generated. It accurately determines the nature of the problem to be generated by first solving the system of linear equations and then inspecting their solution.

III. Procedure

The systematic procedure involved in translating the non-singular system of linear equations into word problems is described in this section.

A. Solving the system of linear equations

The system of linear equations is solved using any of the well-known methods. Gaussian Elimination is employed in the implementation of the proposed method in order to obtain the solution.

The solution thus obtained is inspected and the appropriate entity is chosen for the word problem. Hence, the nature of solution of the system of linear equations is vital in deciding the aspect on which the word problem will be based on.

B. *Choosing the appropriate entity for the word problem*

The inspection of the solution for the system of linear equations is based on three aspects –

1. *Positive integral value:*
   If the solution is a positive integer for all the variables in the system, discrete and indiscrete entities form valid candidates for the word problem.
2. *Positive real value:*
   Since the values of the variables are non-integral values, choosing discrete entities would harm the meaning of the word problem and hence, only the entities that are indiscrete are chosen. For example, it would be meaningless to say "X has 1.5 apples". Whereas, it would be correct if it is stated as "X has 1.5 *kg* of apples".
3. *All real values:*
   Since real values contain a mix of positive and negative numbers, if at least one of the values of the variables is negative, then neither discrete nor indiscrete quantities would be appropriate for translation. For example, it would be meaningless to say "X has -1 kg of apples".
   In such a case, it would be appropriate to choose vector entities as only the vector entities are capable of making sense when negative values are assigned to them. For example, "The momentum of the car is -1 units" is a valid statement.

The classification of the entities into discrete, indiscrete and vector entities is discussed in section IV.

C. *Normalization of the system of linear equations*

The process of normalization in the proposed method is defined as "*Translocation of the terms between the right and left-hand sides in each of the linear equations till all the negative signs are eliminated*".

This process is carried out in order to make the equations suitable for translation using the appropriate regular expression.

D. *Choosing an appropriate regular expression*

Regular expressions are powerful tools to express the language of the system. The proposed method defines a set of regular expressions which are used as the templates during the translation process.

The types of regular expressions and their implications are described in section V.

E. *Translation*

After the regular expression is chosen, each of the linear equations are taken term-by-term and is expanded using the regular expression the end result of which is the English statement.

IV. CLASSIFICATION OF ENTITIES

The entities are classified with an ontological background into discrete, indiscrete and vector entities.

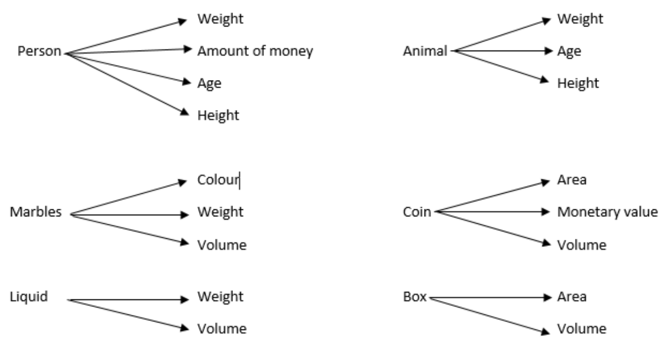
A. *Discrete entites*

Discrete entities are those entities which can be associated only with positive integral values. For example, fruits, marbles, people etc.

B. *Indiscrete entities*

Indiscrete entities are those entities which can be associated with positive real values. Ontologically, indiscrete entities are nothing but the attributes that each discrete entity possesses. For example, "apple" is a discrete entity which can be associated with only *positive integral values*, but "weight of the apple" can be associated with *positive real values*. Indiscrete entities form a superset of discrete entities.

Here's a small chart depicting the relation between the discrete and indiscrete entities –



fig(1) Relationship between discrete and indiscrete entities

C. *Vector entities*

Vector entities are those entities which can be associated with all real values. These are the only entities in the classification with which even negative values can be associated. For example, if one of the variables in the system of linear equations has a solution of -1, then the only possible way of expressing it would be with a vector entity like displacement – "The displacement of the car is -1 meter". Neither "-1 apple" (discrete entity) nor "-1 kg of apple" (indiscrete entity) would be correct.

Examples of vector entities – momentum, displacement, velocity, acceleration, force etc.

V. REGULAR EXPRESSIONS

The regular expressions form the template for translating the system of linear equations into word problem. Each equation is normalized according to the rule defined in section III.C and the regular expression is chosen depending whether the entities are discrete or indiscrete.

Symbols used in the regular expressions:

S - The sum of
c (ϵ N) - times the number of
B/C - entities
k - constant in the RHS
g_i - the number of (for discrete entity) / the 'value' of (for indiscrete entity)
. (dot) - concatenation symbol.
λ – Lambda.

A. *Trivial regular expression*

$$\underbrace{(S+\lambda).c_i(\lambda+g_i)B_i.(c_i(\lambda+g_i)B_i)^*.is.k.(\lambda+(more+less)\ than\ c_i(\lambda+g_i)C_i.(c_i(\lambda+g_i)C_i)^*}_{\text{Data}}.\underbrace{Find\ g_iB_i+g_iC_i(g_iB_i+g_iC_i)^*}_{\text{Question}}$$

Every word problem is composed of *data* and the *question*. The *data* part of the word problem can be generated by the part of the regular expression up to 'Find'. The *question* part can be generated by the part of the regular expression that comes after 'Find'.

e.g. – Given the set of equations 2x + 3y + 2z = 12, 3x - 2y + 8z = 1, 7x - 9y - 12z = 20. The word problem can be generated by following the procedure.

- Transforming the equations into the standard form:
  2x + 3y + 2z = 12 -------------------------------------①
  3x + 8z = 1 + 2y -------------------------------------②
  7x = 20 + 9y + 12z -------------------------------------③
- On solving the system, x = 865/221, y = 433/221, z = 29/34. We note that the solutions are fractions and not integers. Hence, we need to choose the attributes of the entity as the template owing to indiscreetness.
- Template: Marble's attributes.
- Using the trivial regular expression:
  i. The sum of 2 times the weight of red marbles, 3 blue marbles and 2 green marbles is 12. --------①
  ii. The sum of 3 times the weight of red marbles and 8 green marbles is 1 more than 2 times the weight of blue marbles. ------------------------②
  iii. 7 times the weight of red marbles is 20 more than the sum of 9 times the weight of blue marbles and 12 green marbles. ------------------③
  iv. Find the weights of red, blue and green marbles.

B. *Non-trivial/Complete regular expression*

$$(S+\lambda).c_i(\lambda+g_i)B_i.(c_i(\lambda+g_i)B_i)^*.is.k.(\lambda+(more+less)than\ c_i(\lambda+g_i)C_i.(c_i(\lambda+g_i)C_i)^*.Find\ (g_iB_i+{}_ig_iC_I).\ (\lambda+and.S)\ .(c_ig_iB_i+c_ig_iC_I)^*$$

The non-trivial regular expression differs from the trivial regular expression in the question part. Along with asking for the solution for the variables, it may also ask to find the sum of the factors of the variables.

e.g. – Given a set of equations 2x + 5y = 19, 11x – 2y = 16. The word problem can be obtained using the non-trivial regular expression as,

- Transforming the equations into the standard form:
  2x + 5y = 19 ---①
  11x = 16 + 2y ---②
- On solving the system, x = 2 and y = 3. We note that none of the variable's co-efficient is a fraction. Also, the solution obtained is not a fraction. Hence, we can choose the entity itself as the template owing to discreetness.
- Template: Box.
- Using the non-trivial regular expression:
  i. The sum of 2 times the number of square boxes and 5 rectangular boxes is 19. ---------------------①
  ii. 11 times the number of square boxes is 16 more than 2 times the number of rectangular boxes.-②
  iii. Find the number of square boxes, rectangular boxes and the sum of 7 times the number of square boxes and 9 times the number of rectangular boxes.

VI. Design and Implementation

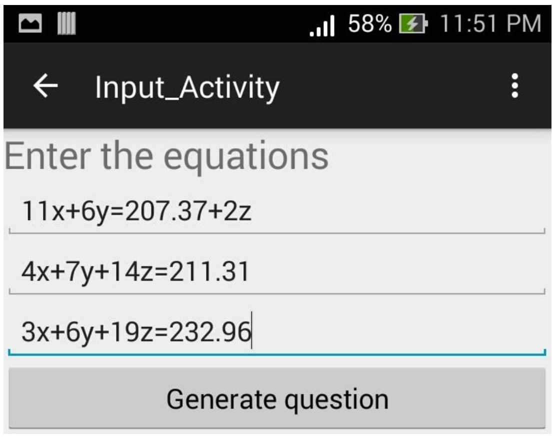
The proposed method is implemented as an Android application which serves as a pedagogical tool for understanding the dynamics that are in play when one tries to solve the system of linear equations.

The various transformations that each equation undergoes during translation is captured and is displayed sequentially which gives the effect of the equation being eaten up term-by-term and the equivalent English words are produced. This would greatly help one in realizing a methodical approach in solving a system of linear equations.

The Android application includes a seek-bar at the top which the user can slide to any point and the corresponding state of the translation is depicted. The user can step forward or backward in order to understand how a particular term gets mapped to the entity.

Finally, the Android application is embedded with a translator provided by Microsoft Bing Translator, so that the English statements that are obtained from the regular expression can be translated to 51 other languages.

VII. Results

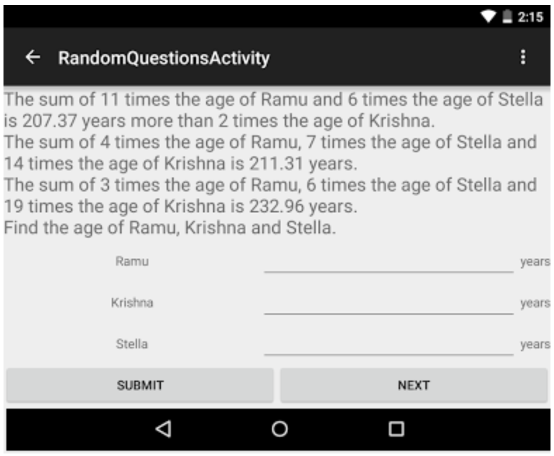


11x+6y=207.37+2z

4x+7y+14z=211.31

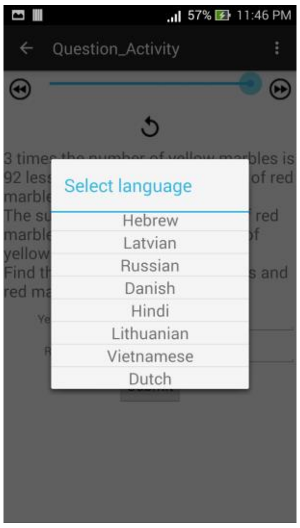
3x+6y+19z=232.96

fig(2) Entering the equations

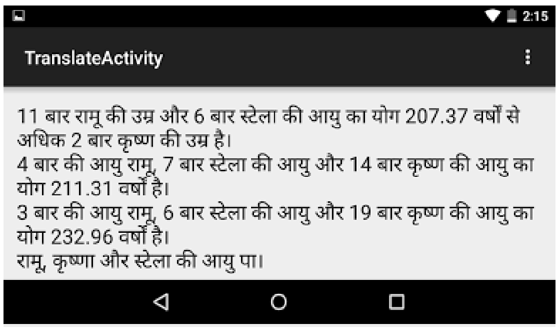


The sum of 11 times the age of Ramu and 6 times the age of Stella is 207.37 years more than 2 times the age of Krishna.
The sum of 4 times the age of Ramu, 7 times the age of Stella and 14 times the age of Krishna is 211.31 years.
The sum of 3 times the age of Ramu, 6 times the age of Stella and 19 times the age of Krishna is 232.96 years.
Find the age of Ramu, Krishna and Stella.

fig(3) Word problem



Fig(4) Translation options



11 बार रामू की उम्र और 6 बार स्टेला की आयु का योग 207.37 वर्षों से अधिक 2 बार कृष्ण की उम्र है।
4 बार की आयु रामू, 7 बार स्टेला की आयु और 14 बार कृष्ण की आयु का योग 211.31 वर्षों है।
3 बार की आयु रामू, 6 बार स्टेला की आयु और 19 बार कृष्ण की आयु का योग 232.96 वर्षों है।
रामू, कृष्णा और स्टेला की आयु पा।

fig(5) Word problem translated into one of the many languages

VIII. Conclusion

The proposed method has proved how the given system of linear equations can be translated into English statements, which are nothing but the word problems. Since the regular expressions are at the crux of the translation, the proposed method also suggests capturing the transformations which when displayed sequentially, would help one in understanding the “acts behind scenes” that are in play when one tries to solve a system of linear equations. The proposed method thus guarantees the existence of a word problem for a given non-singular system of linear equations.

IX. Acknowledgements

We thank Dr. Viraj Kumar for his constant support and guidance.

References

[1] Using the Natural Language Generation tool for generation syntactically correct multi-sentence MWP questions. Sandra Williams computing department, The Open University. 2011 AAAI Fall Symposium (FS-11-04)

[2] Personalized Mathematical Word Problem Generation. Oleksandr Polozov University of Washington, Eleanor O’Rourke University of Washington, Adam M. Smith University of Washington, Luke Zettlemoyer University of Washington, Sumit Gulwani Microsoft Research Redmond, Zoran Popovic´ University of Washington. ijcai2015.