**Equation Solver**

**ESAAS Project Report**

**Submitted by:**

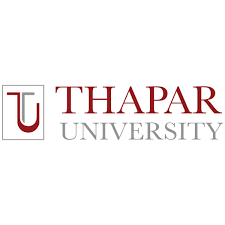
**101412029 - Purvanshi Mehta**

**101412041 - Taniya Saini**

**101412045 - Vishruty Mittal**

**Lab Group: SE 2**

**BE Fourth Year, SE**



**Computer Science and Engineering Department**

**Thapar Institute of Engineering & Technology, Patiala**

**NOVEMBER 2017**

**ABSTRACT**

This project was started with the sole purpose of helping students with their poor mathematical skills. Solutions like WolframAlpha and Euclid do exist, but their number of categories are restricted to only formulaic problems.

It was decided to build a platform where students could enter their problems, and get their solution, and the equations involved.

The project currently has two categories - Simple Interest, and Operation Prediction (simple addition, subtraction, division and multiplication problems).

Simple Interest is solved using Natural Language Processing (NLTK and Spacy for tagging entities and finding questions and values, and mapping them to their respective domains) and Machine Learning (SVM for classifying the problem).

Operation Prediction problems are solved using Natural Language Processing (extracting values) and Deep Learning (Tensorflow and Keras for classifying the problem).

It is found that Simple Interest problems, which had no existing models or solutions, were solved with an accuracy of 89 %. It was also found that Operation Prediction problems, which had an accuracy of 81 % with existing models, have been solved with an accuracy of 91 %.

Thus, we have managed to solve problems from two domains very comfortably.

**ACKNOWLEDGEMENTS**

We would like to express our thanks to our subject Faculty Mr. Anoop Jacob Thomas. He has been of great help in our venture, and an indispensable resource of technical knowledge. He is truly an amazing mentor to have.

We are also thankful to Dr. Maninder Singh, Head, Computer Science and Engineering Department, entire faculty and staff of Computer Science and Engineering Department, and also our friends who devoted their valuable time and helped us in all possible ways towards successful completion of this project. We thank all those who have contributed either directly or indirectly towards this project.

Lastly, we would also like to thank our families for their unyielding love and encouragement. They always wanted the best for us and we admire their determination and sacrifice.

**TABLE OF CONTENTS**

Section 1 - [**Introduction**](#_9u6bjw4dpw3m) **7**

Section 2 - **Requirement Analysis 15**

Section 3 - **Design Specifications 29**

Section 4 - **Results and Evaluation 36**

Section 5 - **Conclusion and Future Decisions 38**

Section 6 - **Project Metrics 40**

Section 7 - **References 40**

Section A - **Appendix** **42**

**LIST OF TABLES**

**Sr. No. Table Details Page No**

Table 2.1 Equation Generation 22

Table 2.2 Equation Solution 23

Table 4.1 Comparison with Aris 36

**LIST OF ABBREVIATIONS**

**Abbreviation Full Form**

NLP Natural Language Processing

NER Named-Entity Recognition

POS Part Of Speech

SVM Support Vector Machine

ICSE Indian Certificate of Secondary Education

CBSE Central Board of Secondary Education

ML Machine Learning

R&D Research and Development

NLTK Natural Language Toolkit

CPU Central Processing Unit

SI Simple Interest

ROI Rate Of Interest

AI Artificial Intelligence

MWP Math Word Problems

WA Washington

USA United States of America

HTML HyperText Markup Language

CSS Cascading Style Sheets

JS JavaScript

RNN Recurrent Neural Network

SRS Software Requirement Specifications

UI User Interface

OS Operating System

**SECTION 1 - INTRODUCTION**

**1.1 - PROJECT OVERVIEW**

In this section, we first aim to detail all the relevant technologies and terms used in this project. We will then explain our rationale for developing this project, and how we think it will impact students and researchers alike. We also explain the problem, the goal, our solution, any assumptions made, and potential risks.

**Technical terminology**

1. **NLP** - This is the study of natural or human language for various tasks such as building virtual assistants or chatbots, converting text to speech, translating a language to another, gathering information from images, and numerous other purposes. As one can gather, it is an extremely diverse area of study, with numerous independent, and interconnected fields of study.
   1. **NER -** NER refers to Named Entity Recognition, meaning identifying places (Paris, Tokyo), dates (6th November, 06/11/1992), percentages, money ($162, ¥50000) and so on. It involves huge datasets with pre-tagged quantities, which in turn are used to find named entities in new questions.
   2. **POS Tagging -** POS Tagging refers to Part-Of-Speech Tagging, or the act of telling whether a respective word is a noun, verb, adverb, preposition, etc. and if so, finding out which *kind* of word it is (detecting Wh-words such as What, When, Where, Who etc.)
   3. **Dependency Parsing -** This refers to the act of finding out relationships, or dependencies between various words. This is again done using huge datasets with manually tagged dependencies, from which relationships in new sentences are derived.
2. **Machine Learning -** This is the study of making a machine artificially intelligent, by "training", i.e. either making it learn the dependencies between various features (Supervised Learning), or learning hidden features from unknown data (Unsupervised Learning).
   1. **SVM -** SVM stands for Support Vector Machine. This is used for classification (separation of data into different labels on the basis of parameters the algorithm learns through Machine Learning, or the parameters are manually specified).
3. **Deep Learning** - This is the process of modelling the brain through artificial "neurons", deep learning units that change outputs depending on magnitude of the error (difference between the actual and the predicted value). Deep learning networks consists of many layers of networks (a layer consists of many neurons, and can be visualized as being connected vertically), visualized as being connected horizontally.
   1. **TensorFlow and Keras -** TensorFlow is a dataflow library (dataflow programming paradigm means dividing a certain problem into various parts, and executing them in parallel) open-sourced by Google in 2015. Various Fortune 500 companies such as Uber, Airbnb, eBAY, Snapchat etc. use it for performing operations on extremely large amounts of data [9]. Primarily, it is used for its ability to model neural networks. Keras is a high-level interface for TensorFlow.

**Problem**

As detailed below, the majority of the students do their Maths homework since it is expected of them, and not because they find Maths interesting. This is a problem often faced by teachers and curriculum designers, but it is most often found in Math students. This can even lead to a pathological hatred for Maths, and leads them asking questions like - "Why should I study Maths? If I need to do calculations, then I have a calculator for that. "

**Goal**

1. Our aim is to provide students (up to 6th grade ICSE / 7th grade CBSE / other equivalents) a platform where Maths can be made less of a chore to learn, and awaken the interest of students who have given up on the subject.
2. We will first aim to provide students with a platform where they will be provided detailed solutions for the problem they enter.
3. After finishing this, we will also work on visualization - where kinesthetic learners will be provided learning aids (i.e. - animations), so that they can get a firm grasp on the questions.

**Solution**

Our solutions looked simple on paper, but was quite difficult to implement - our aim was to model a simple problem from a student's perspective. A student looks at a problem, figures out the category, finds out the quantity asked, and applies the relevant formula. This is exactly what we implemented using NLP and ML. In operation prediction, where we find whether the operation to be performed in a question was '+', '-','\*', or '/' and then calculate the answer, we went one step further and implemented removal of irrelevant information.

**Assumptions**

1. We assume that the coming generations will become much more tech savvy. Students will depend on the Internet for studies much more than they do currently. Even the younger students will utilize the power of internet to solve their problems. Students will be able to use websites easily.
2. We assume that the main concern of students not being able to grasp the problems lies in their inability to visualize the questions.
3. We assume that the student is feeding genuine questions as input from standard textbooks, assignments and websites, where the questions are unambiguous.

**Risk Analysis**

This project carries no significant risks. The answer to "Why so?" is detailed below: 2

1. We have no hardware dependency, this being a pure software-based project.
2. We have no R&D expenditure that we have to recover, so the issue of monetization through advertising and possible loss of visitors is not present.
3. Anybody is free to contribute to the project by submitting a pull request, or forking our project on Github, so there is no risk of the project remaining stagnant or "dying".

**1.2 - NEED ANALYSIS**

This project is aimed at students and researchers alike - students for the fear they face while attempting Math Word Problems (detailed below), and researchers (for the oft inaccurate or restricted problems their models solve).

We have used a novel approach for solving both categories of problems (Simple Interest and Operation Prediction), by first pre-processing the dataset. Our training dataset consisted of 800 unique examples, which were collected from a variety of sources such as 5th and 6th grade CBSE and ICSE books, aptitude questions gathered from sites such as IndiaBix, and some were sourced from Euclid itself. We then classified this dataset by SVM, to find out which category the question belonged to (whether we had to find rate, time, etc.), and then applied NLP (NLTK and Spacy for NER Tagging, POS Tagging and Dependency Parsing). We then obtained the solution.

For solving Operation Prediction problems, our approach is initially a little different. We first collected the data from a Chinese researcher, who was glad to further our research project. This dataset consisted of 1500 training and testing examples. We then applied an algorithm of our own making for removing irrelevant data from the question, which is given to confuse the reader. We then used Deep Learning (Keras with a TensorFlow backend) to find the operation.

This approach is quite different from sites such as WolframAlpha and Euclid, who solve such problems using brute force (tons of computing power for simple tasks). Our method is drastically faster, and is exponentially lighter on memory and CPU usage.

**1.3 - PROBLEM DEFINITION AND SCOPE**

It is a likely for a middle-grade students to skimp on homework, but it is even more likely that the student skips his Math homework. In fact, according to a study conducted in 2005 by Ipsos [1], 4 in 10 Americans “strongly” dislike Maths.

It is quite said that mathematics is considered to be the toughest subject in spite of availability of numerous online and offline subject materials. We aim to rectify this.

There are various platforms like Wolfram Alpha which give step by step solution for inputted equations, but there is no place where a student can get a solution for even elementary level Math word problems. (Our first objective is the solution.)

We also plan to visually represent both the question and the answer to make it easy for kinesthetic learners to understand what the problem is trying to tell them. (Our second objective is the visualization).

**1.4 - APPROVED OBJECTIVES**

1. Research work - We aim to contribute to the field of Natural Language Processing by combining Mathematics as well as Artificial Intelligence. We aim to improve the accuracy of Word Problem Solving (which is currently around 60% for addition and subtraction problems, and goes even lower for more complex ones).
2. Currently, we aim to solve problems from unsolved categories such as Profit/Loss, Unitary Method, and so on.
3. We then aim to improve the accuracy of already existing sites and work and improve the complexity of the problems that they can handle.

**1.5 - METHODOLOGY USED**

1. Classification: In this step, we will classify the problem and assign to it its category. This is performed by a classifier which we have built in Python. It works by detecting keywords (principal, amount, etc.).
2. Finding the unknown quantities: In this step, we will find the unknown quantity (the values to be found). Suppose a person enters a question as “Ram invested Rs. 500 in a bank at the rate of 5 p.a. Find the simple interest incurred in one year. ”

Then, the unknown quantity / question to be extracted is “Find the S.I. … “ This task is accomplished by POS Tagging and Wh-word (or question words, such as who, what etc.) recognition.

1. Extraction of known quantities: In this step, we will find the known quantities (the quantities which are surreptitiously mentioned in the question). For the same example above, the known quantities extracted are “Principal = 500”, “ROI = 5”, and “Time = 1”. This task is accomplished by Named Entity Recognition.
2. Generation of answer: Each basic word problem can be classified into subcategories - for example, SI problems can be divided into
   1. Finding a quantity when three others are given.
   2. Finding amount.
   3. Investment of parts (example - splitting investment into two parts, and finding the cumulative interest after a year). This also includes profit sharing.

This task will call the function according to the subcategory found using previous steps.

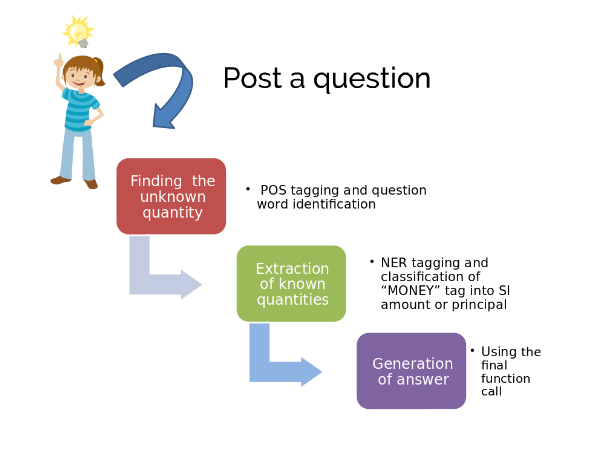


Figure 1.1 - Flowchart for Simple Interest Solving

**1.6 - PROJECT OUTCOMES AND DELIVERABLES**  
As mentioned above, our first outcome is the solution. Here, a student should enter his/her problem, and get a step-by-step solution. If the student so desires, he/she can demand an animation for both the question as well as the solution, for better understanding of how we arrived at the solution from the question.

Finally, since we are aim to attract researchers in the field of NLP and AI, we will improve the accuracy of pre-existing models.

**1.7 - NOVELTY OF WORK**

* We have used a novel approach for solving both categories of problems (Simple Interest and Operation Prediction).
* In SI problems, we first classified the collected dataset by SVM, to find out which category the question belonged to (whether we had to find rate, time, etc.), and then applied NLP (NLTK and Spacy for NER Tagging, POS Tagging and Dependency Parsing). We then obtained the solution.
* For solving Operation Prediction problems we first apply an algorithm of our own making, for removing irrelevant data from the question, which is given to confuse the reader. We then used Deep Learning (Keras with a TensorFlow backend) to find the operation.
* This approach is quite different from sites such as WolframAlpha and Euclid, who solve such problems using brute force (tons of computing power for simple tasks).
* Our method is drastically faster, and is exponentially lighter on memory and CPU usage.

**SECTION 2 - REQUIREMENT ANALYSIS**

**2.1 - LITERATURE SURVEY**

**Theory associated with problem area**

Mathematical word problems (MWP) constitute an integral part of a child’s elementary schooling curriculum. Solving a MWP is a complex task involving critical aspects of reading comprehension (understanding the components of the problem), and generating a solution that agrees with the “story” in the problem. Children are trained through the process of problem solving by the use of various strategies.

The general approach to any MWP solver is this : categorize problem; extract known quantities, and the relations between them; extract unknown quantities; finally call function responsible for solution.

**Existing Systems and Solutions**

1. Euclid - Allen Institute for AI is an institute based out of Seattle, WA, USA whose mission is to “contribute to humanity through high-impact AI research and engineering.” [10] Their project Euclid aims to solve SAT-style Math problems, which include simple problems like “How many multiples of 12 are less than 25?”. But even a slight change in the phrasing of the words can result in the problem not being parsed at all. Also, the maximum complexity of the problems is only upto 5th grade CBSE. The accuracy of the model is around 60-65 %.
2. WolframAlpha - WolframAlpha, a computational engine made by Stephen Wolfram and his team over a period of 10-15 years, started solving simple word problems - which a second grader would be expected to solve (for example - A boy has 12 apples more than Jon. Jon has 2 apples less than George. If George has 3 apples, how many apples are there in all? [3] ) But WolframAlpha, despite its high accuracy, can only solve extremely simple word problems.

These are the only two large-scale and accurate implementations that we could find. Many research papers have been published on the topic, but very few have great accuracy, and even fewer have been successfully implemented on the Internet.

**The Problem that has been identified**

An ordinary child has to learn the required algebra, but will easily grasp the narrative utilizing extensive world knowledge, large vocabulary, wordsense disambiguation, coreference resolution, mastery of syntax, and the ability to combine individual sentences into a coherent mental model.

In contrast, the challenge for an NLP system is to “make sense” of the narrative, which may refer to arbitrary activities like renting bikes, collecting coins, or eating cookies.[4]

**Related Work**

Prior studies attempting to solve mathematical word problems in an automatic manner fall into two primary categories: those intended to understand the cognitive aspects of problem solving in children and those intended for intelligent tutoring systems

Prototypical systems such as WordPro [5], Solution [6], and ArithPro [7] are representations of cognitive models of human mathematical word problem solving processes. These operate on propositional representations of the problem text later solved in a rule-based manner.

In the realm of intelligent automatic tutoring systems, MWP solvers were based on either using specific sentence structures and keywords[8], or using templates (schema) limited in scope by variety and problem types.

STUDENT [8] handles algebraic problems by first transforming NL sentences into kernel sentences using a small set of transformation patterns. The kernel sentences are then transformed to math expressions by recursive use of pattern matching. The major difference is the introduction of a tree structure as the internal representation of the information gathered for one object.  
Almost all of the approaches parse NL text by simply applying pattern matching rules in an ad-hoc manner. For example, as mentioned in Bobrow [8], due to the pat-  
tern “($, AND $)”, the system would incorrectly divide “Tom has 2 apples, 3 bananas, and 4 pears.” into two “sentences”: “Tom has 2 apples, 3 bananas.” and “4 pears.”  
**Methods and Tools**

1. **Algorithm**

We require no particular tool for this project, except for the usual software required for such a project, such as -

1. Linking, using Flask.

The method used is the ones used in almost all of the research papers that we studied, but more specifically, we are improving upon the solution provided by Euclid [4]:

1. Classify problem, and assign some category.
2. Go to the program for the category.
3. Find the variable(s) to be found.
4. Find the known variables.
5. Find the relationships between the known variables.
6. Using the above three steps, go to the function for your specific purpose. For example, function findPfromIntTimeAndRate() is called when you are given Simple Interest, Time, and Rate of Interest, and you need to find the principal.

**b) Tools used**

We require no particular tool for this project, except for the usual software required for such a project, such as -

1. Linking - Flask.
2. Responsive webpage - HTML, CSS, Bootstrap, JS, jQuery.
3. Python libraries - NumPy, SciPy, NLTK, Spacy, Keras, TensorFlow.

**2.2 - ANALYSIS / WORKING PRINCIPLES**

We have already described the technological terminology used above in the "Project Overview" section. We detail how the project actually works in this section:

**Simple Interest**

This task involves five parts:

1. Finding out the rate, time and any "money" values - this task is accomplished by NER Tagging using Spacy, an industrial-strength Natural Language Library. The number of "money" values are also computed using Spacy.
2. Finding the quantity to be computed - This can be one of the following - principal, rate, time, interest and amount. This task is accomplished by extracting the question from the input using POS Tagging, and then finding the words "principal","rate", etc. in the question, and then assigning category.
3. Getting the pattern for the question - This is accomplished by applying POS Tagging to the above result, and then getting what we call a "pattern".
4. Matching the pattern - We had already applied SVM to the collected dataset, and applied NLTK to the result to get the 13 most common patterns occurring in SI problems. The SequenceMatcher is used to get similarity scores for the question pattern with all of the 13 patterns, and then selecting the pattern with the maximum similarity score.
5. Calculating the quantities - On the basis of the unknown quantity and the above selected pattern, we apply the standard formula (SI = P\*R\*T / 100) or any variation of it, and then obtain the result. If the quantity to be found is the amount, then the formula A = P + I can also be used.

**Operation Prediction**

The task of predicting the operator and finding the relevant quantity involves three parts:

1. Dividing the question - The question is divided into two parts - the question (which contains the quantity to be computed) and the story (which contains the facts relevant to the problem).
2. Removing irrelevant information - For every line in the stories list, if the story data is not relevant to the question, then remove the offending line.
3. Pre-process the data - This task is accomplished using Keras's preprocessing functions - the story and the query are preprocessed separately.
4. Building the model and evaluation - A sequential neural network with dropout, embedding and RNN is built, and the weights and other details of the network are loaded using the saved model with the same architecture. The model is then evaluated. The model is then used to compute the solution.

# **2.3.1 Introduction**

## **2.3.1.1 Purpose**

This manual is for describing the SRS, or the software Requirements Specification for **Equation Solver**, a product which will be used for animation of elementary-level Mathematics problems (and their generated solutions) to make it easy for an average student to have a strong grasp over Math questions.

## **2.3.1.2 Intended Audience and Reading Suggestions**

This SRS is deliberately written in common man language so that even the normal user, with no technical skills can easily understand the gist of the document. This SRS contains everything a manual might contain - how the project came into being, its features, its requirements, and so on.

For better understanding, it is suggested that the manual be read in the same order it is written.

## **2.3.1.3 Project Scope**

The project is intended for students who either have trouble solving (or visualizing) Math problems, and the benefits are - better understanding, and the student can solve similar problems of the same type easily.

Our short-term goal is expanding the difficulty level for the problems, while keeping the categories same (ex - inculcating more difficult Profit/Loss problems).

Our long-term goal is - making sure that the project works even for problems on sites such as GeeksForGeeks, HackerRank, etc.

# **2.3.2 Overall Description**

## **2.3.2.1 Product Perspective**

Despite the controversies surrounding Math, the proponents and naysayers of Mathematics both agree on one thing - that it is complex. Complexity may be tantalising - you keep on working even though knowing it is somewhat futile, and reach the solution at the end. But the complexity may be frustrating too - you have tried every possible method under the Sun, but the problem refuses to budge. We have seen that due to a variety of reasons (poor instruction, poor material, and lethargy, poor visualization power) students generally treat Maths with contempt; not worth their time.

We found that when a student is asked a word problem, he struggles with *visualization*, which is where our project comes in.

When the user opens the product web site, he will be treated with a sleek, minimalistic design, and an input box in which to enter his question in. He will then be treated to a solution to his question.

We hope to bring purity to the muddy waters of Math word problems, and to give kinesthetic learners the boost they so desperately need.

## **2.3.2.2 Product Features**

Since we are building our project on the core tenet of *visualization*, it is only fitting if we describe the features through a class diagram rather than writing on and on.

## **2.3.2.2.1 User Classes and Characteristics**

We anticipate that only students will use the project.

## **2.3.2.2.2 Operating Environment**

The project will operate on any system having

1. A processor that is not *too* old (example - Intel Celeron/Pentium).
2. A browser that is fully compatible with HTML5 (which means no Internet Explorer).
3. Any operating system (as long as it has a *modern* browser).

Since the projects runs on the Web, it does not conflict with any other application/website/etc.

## **2.3.2.2.3 Design and Implementation Constraints**

One constraint arises from the fact that the browser required should be fully HTML5 compatible.

Also, the majority of the research on Natural Language Processing takes place In English (due to there being more data to work on in English). This, coupled with the fact that our dataset comprises of only English problems, means that the only language we will be supporting is English.

## **2.3.2.2.4 Generation of Equation(s)**

Table 2.1 - Equation Generation

|  |  |
| --- | --- |
| **System feature** | Equation Generation |
| **Priority** | High |
| **Description** | The equations are generated after confirmation of the category. |
| **Action** | The most important part of the project - using a classifier for categorizing, and then using NLTK / Keras for extraction of features, and eventual formation of the equations. |
| **Result** | Category is found out and equations are created. |

## 

## **2.3.2.2.5 Solution of Equations**

Table 2.2 - Equation Solution

|  |  |
| --- | --- |
| **System feature** | Equation Solution |
| **Priority** | Medium |
| **Description** | Self-evident title. |
| **Action** | The questions will be solved, and the equation returned. |
| **Result** | The solution(s) for the equation(s) are returned. |

## 

# **2.3.3 External Interface Requirements**

## **2.3.3.1 User Interfaces**

The User Interface (UI) is purposefully kept as minimal and clean as possible. The interface consists of a single page that refreshes parts of content using Ajax when new content is fetched or parsed.

The first part of the interface consists of an input box for entering the Math problem in question. The second part of the interface displays the category for the questions, and asks the user whether this is the category he was expecting. If the user selects yes, then the equations are generated, solved, animated and displayed. The interface will visualize the features and functionalities listed in this document for this prototype as the included below not limited to:

1. Input box for entering the equations.
2. Animations and solution displayed in a clean interface.
3. Help button.

## **2.3.3.2 Hardware Interfaces**

* The server/PC hosting the files for using Spacy and CoreNLP.
* Windows/Mac OS X/Linux users’ computers.

## **2.3.3.3 Software Interfaces**

For the prototype, we will launch the portal over the Internet, and other than the hardware specified in the hardware interface section, the software requirements boil down to any OS that has the capacity to support a HTML5 compatible browser.

## **2.3.3.4 Communications Interfaces**

Internet connection and a web browser are required in order to make use of several functions and to be executed such as searching, viewing and downloading.

# **2.3.4 Other Nonfunctional Requirements**

## **2.3.4.1 Performance Requirements**

For this prototype version, we will keep on detecting if the system crashed, hanged or an operating system error occurred. Also detecting the performance of the system in terms of the efficiency of integration of the different components.

## **2.3.4.2 Safety Requirements**

Our software will be completely safe to use. Taking help from this will not result in any kind of loss, damage or harm to the product as well as user. On the contrary, the software will help the user to attain a solution to the problem.

## **2.3.4.3 Security Requirements**

Equation Solver doesn’t require any user credentials to use it. The software will be open to all who are willing to solve a maths question. Since we will not be saving any personal information about the user, the security issues won’t be raised.

**2.4 - ANALYSIS MODEL**

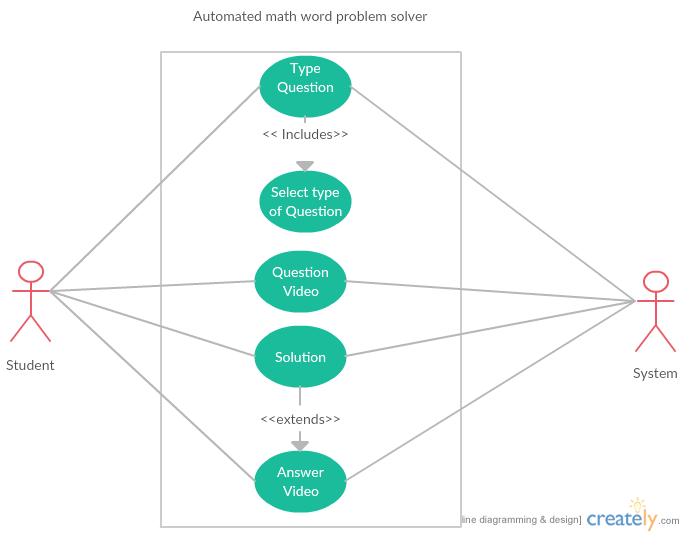


Figure 2.1 - Use Case Diagram

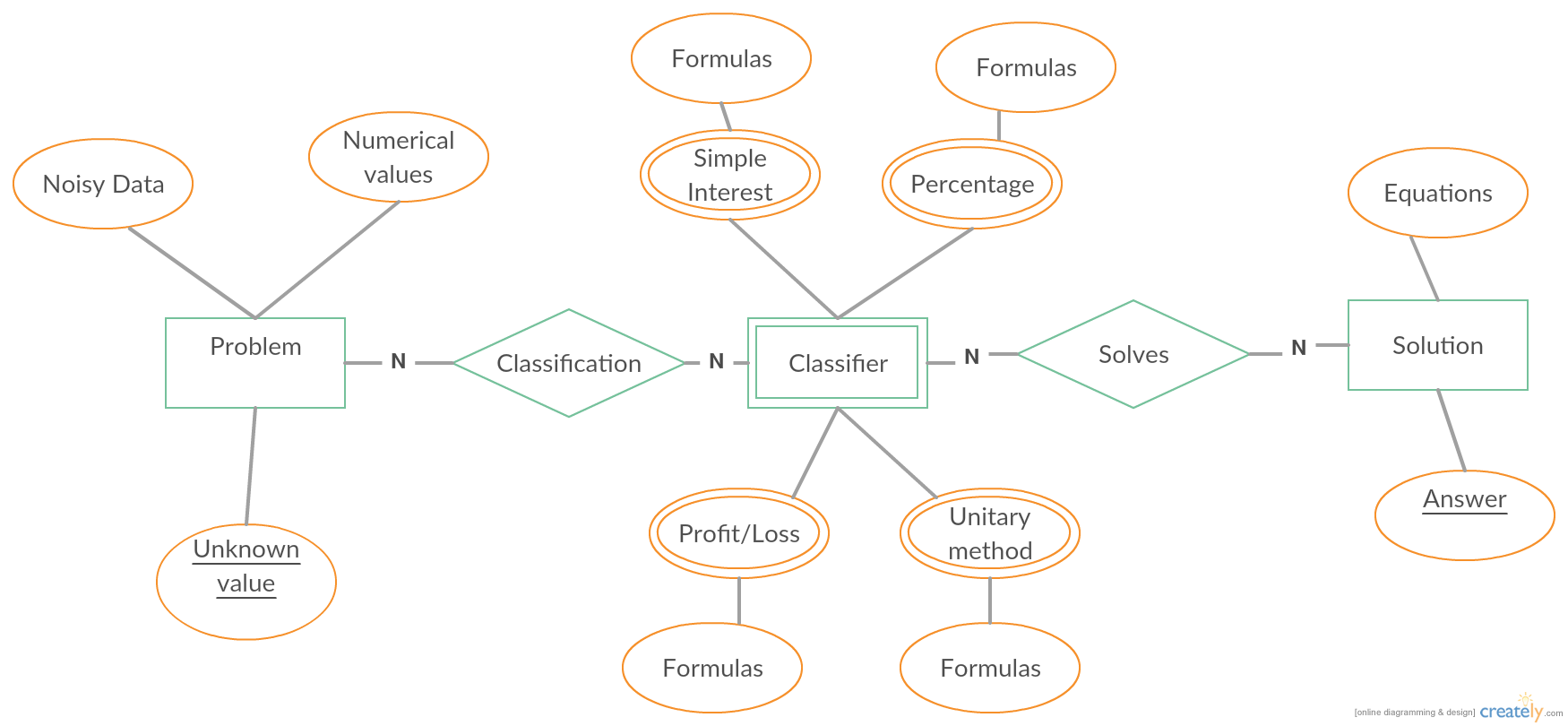
****

Figure 2.2 - ER Diagram (Data Model)

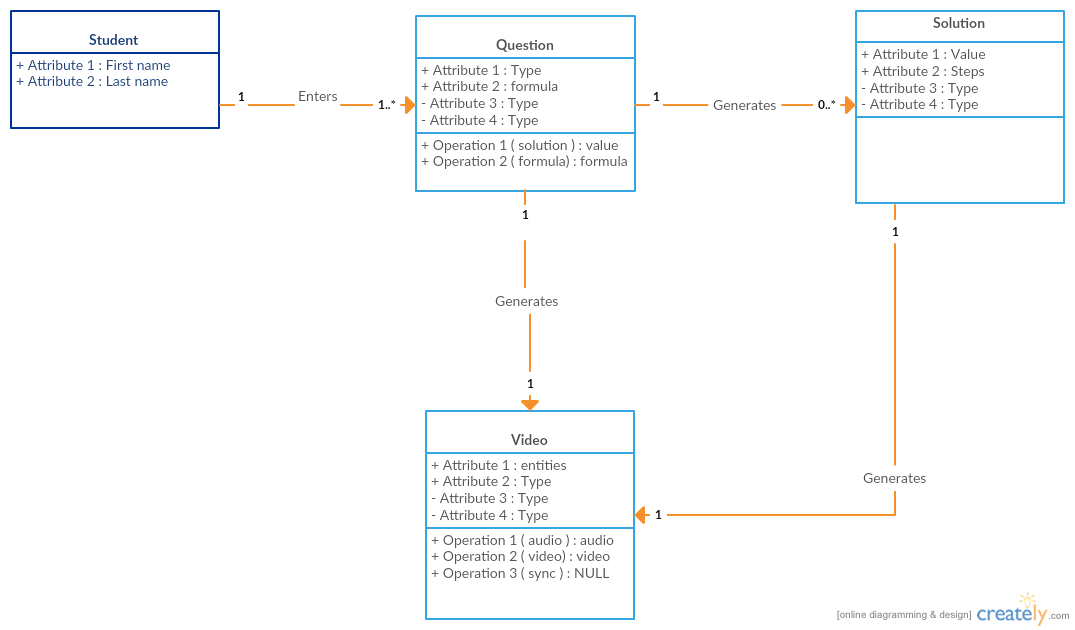
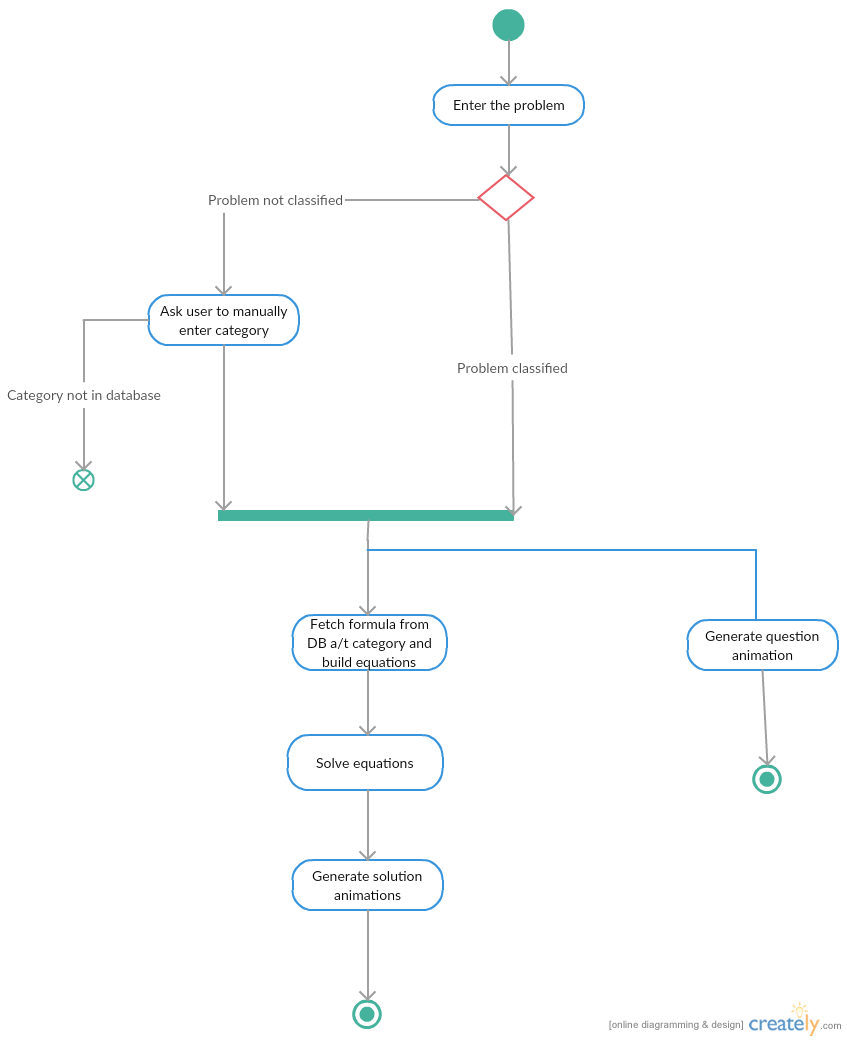


Figure 2.3 - Class Diagram

Figure 2.4 - Activity Diagram

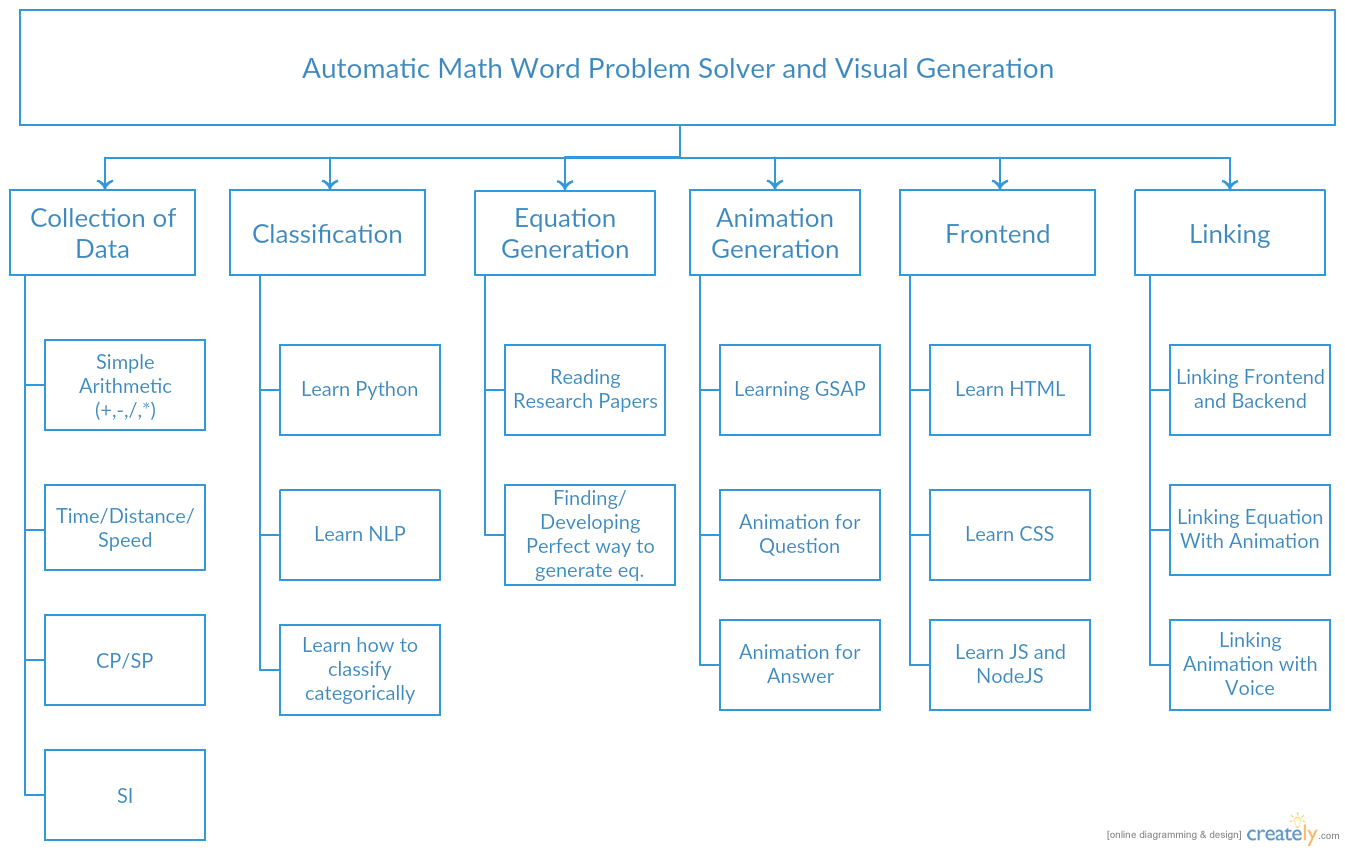


Figure 2.5 - Work Breakdown Structure (WBS)

**2.5 - COST ANALYSIS**

Since there is no use of hardware involved in the project, there is no need for cost analysis. Also, all the libraries used were reusable and open-source.

**2.6 - ASSUMPTIONS AND CONSTRAINTS**

For Simple Interest:

* '$' is used for money values. More specifically, symbols such as 'Rs.' cannot be used.
* Time is in years, not months.

For Operation Prediction:

* Only one operation is involved in the question.

Common to both:

* The question is unambiguous, and without spelling errors.

**SECTION 3 - DESIGN SPECIFICATIONS**

**3.1 - FLOWCHART OF THE SYSTEM**

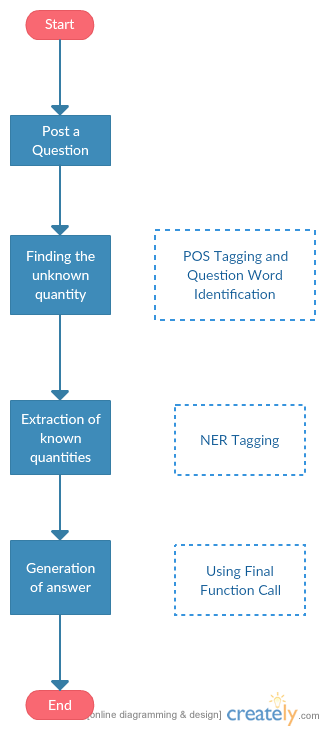


Figure 3.1 - Diagram showing flow of data in the system

**3.2 - DESIGN PHASES**

Not used.

**3.3 - FABRICATION SEQUENCES**

Not used.

**3.4 - SYSTEM ARCHITECTURE**

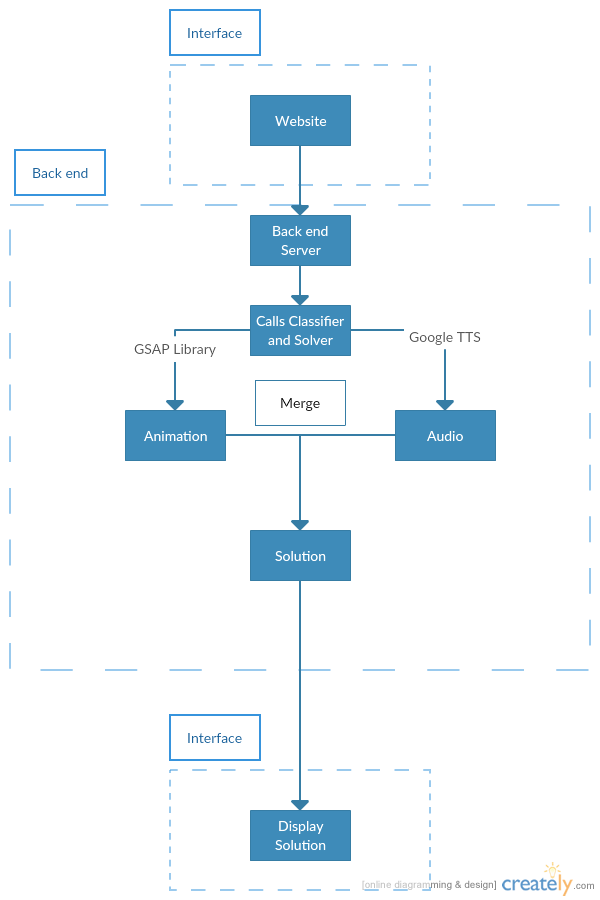
****

Figure 3.2 - Architecture Diagram

**3.5 - USER INTERFACE DIAGRAM / SCREENSHOTS**

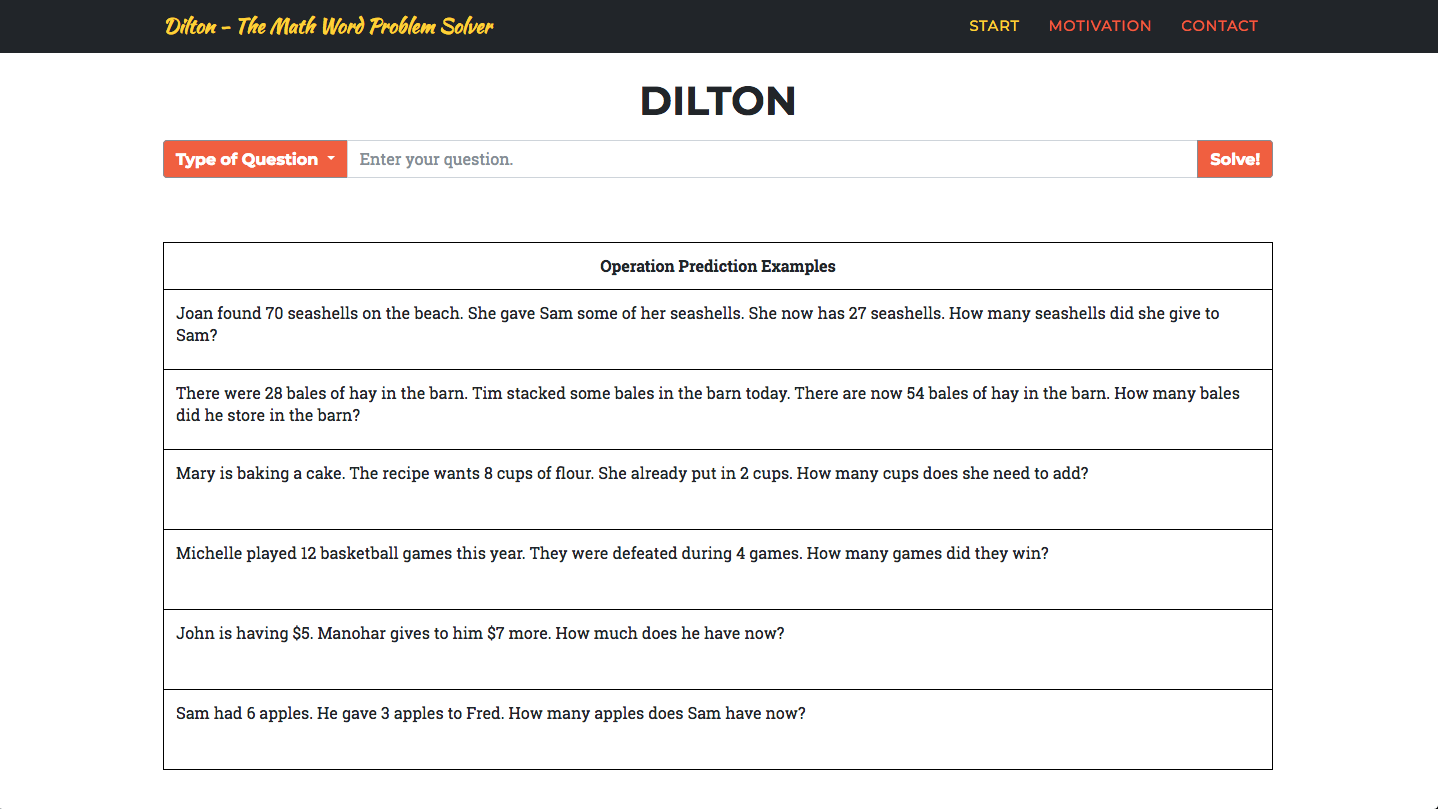
****

Figure 3.3 - Main Page - Part I

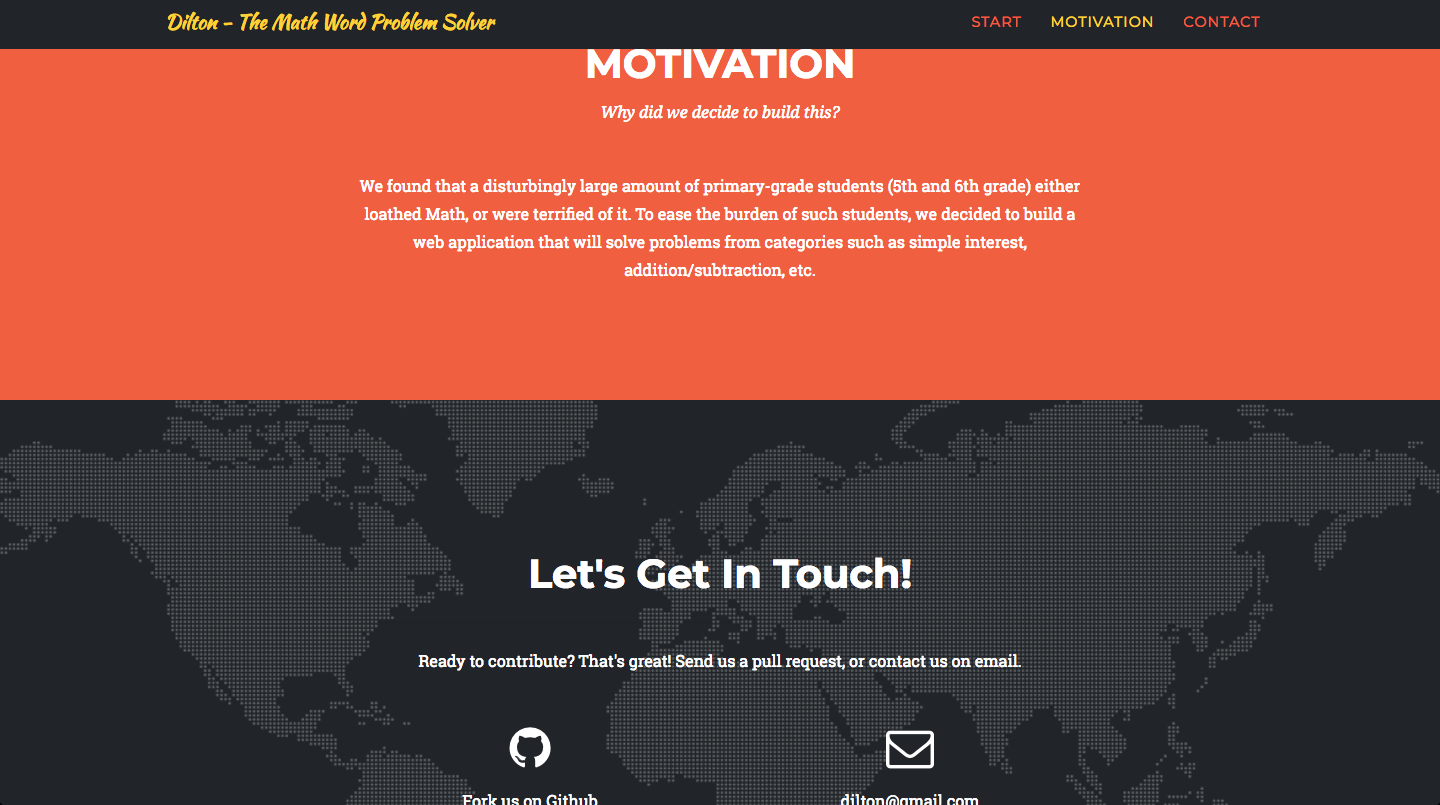
****

Figure 3.4 - Main Page - Part II

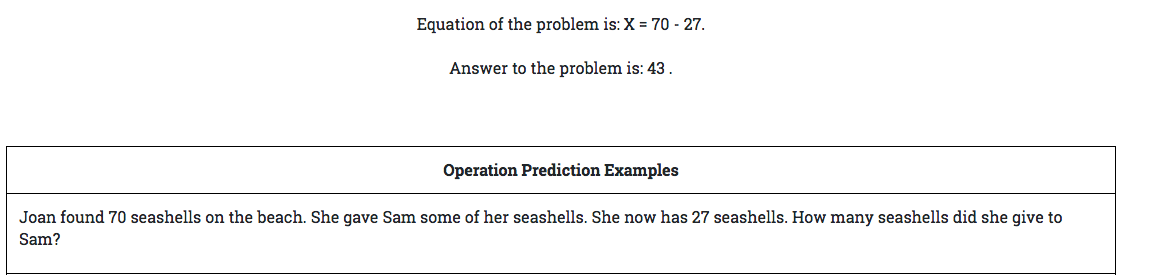
****

Figure 3.5 - Solution to an Operation Prediction problem

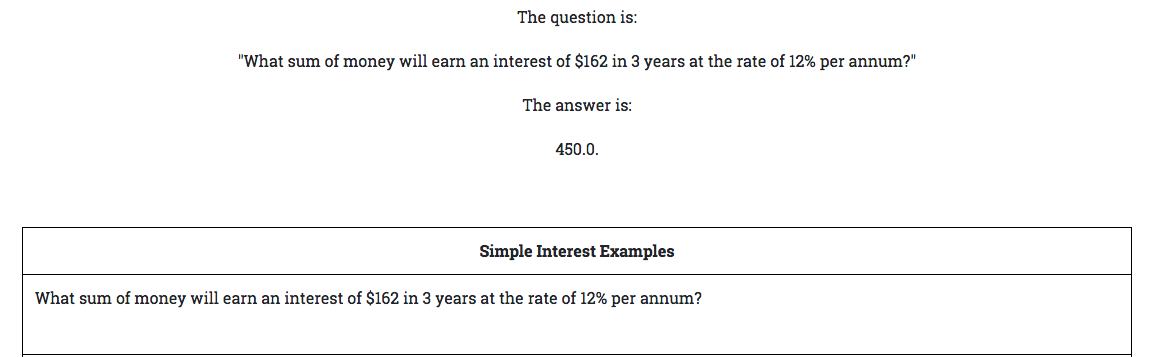
****

Figure 3.6 - Solution to a Simple Interest problem

**3.6 - SYSTEM COMPONENTS AND RELEVANT ALGORITHMS**

**Simple Interest**

* Input question
* Find tagged question by applying POS Tagger
* Apply dependency parser on question
* Apply NERTagger (Spacy) on question
* Find the category of question
* If entity text is PERCENT, set value of rate
* If entity text is DATE, set value of date
* Find the question text and category using POS Tagged input and Input
* Find question pattern using POS Tagging and Dependency Parser.
* Match pattern with max ratio using SequenceMatcher.
* Find answer using the formulae SI = P\*R\*T / 100, and A = P + I

**Operation Prediction**

* Divide the question into story and query
* Remove irrelevant information that is not related to the question
* Vectorize stories into training and testing batch data
* Build model using preloaded weights and paramters
* Find answer using built model

**SECTION 4 - RESULTS AND EVALUATION**

**4.1 - TESTING PROCESS**

**Test Plan**

1. For Simple Interest, we first built our pattern file on 850 training examples, but there wasn't sufficient data for testing. So we built our files on 650 - 700 training examples, and then planned to test on the rest. We also decided to try some other things which we detail below.
2. For Operation Prediction, we split the data into train data (1418 examples) and test data (435 examples). As the name suggests, we planned to test our built model on the test dataset. We also decided to try some other things which we detail below.

**Test Cases**

1. For both the categories, we built our model and then proceeded to test it on the testing dataset.
2. For both Simple Interest and Operation Prediction, we had a few cases -
   1. Ambiguity in the question
   2. Incorrect spellings / punctuation
   3. Incomplete data
   4. More than one quantity to be found
   5. More than one operation (too much data)

**Test Results**

1. We found that our model performs admirably even on questions with incorrect spellings.
2. Incomplete data was a complete miss, in line with our expectations. We instead added an exception, and printed the output "Incomplete data, please enter another question".
3. The other tests were ambiguous - sometimes the model performed well, sometimes it didn't.
4. Our model performed extremely well on the test data (97 % - 99 %).

**4.2 - SIMULATION SET UP**

None whatsoever.

**4.3 - DISCUSSIONS / INFERENCES DRAWN**

The project has great results, but we realised we still have a long way to go before we can begin to popularize our project as a viable solution for frazzled students. We also have to implement visualization, and that comes if we expand the categories to Profit/Loss, Unitary Method, etc.

**4.3 - VALIDATION OF OBJECTIVES**

Table 4.1 - Comparison with Aris (Operation Prediction Problem Solver)

|  |  |
| --- | --- |
| **System** | **Categorization Accuracies (%)** |
| Aris | 81.2 |
| Our Model | 88.81 |

For Simple Interest, there is no existing solution that we know of, so the base accuracy to compare with was not determined.

**SECTION 5 - CONCLUSIONS AND FUTURE DIRECTIONS**

**5.1 - CONCLUSIONS**

We started this report by highlighting the growing dislike of students for Mathematics, and the way that even interested students lose their way while venturing in the depths of word problems. We also mentioned that present systems that intend to solve such problems get bogged by a plethora of issues, the most major being ambiguity. So, our project serves a twofold purpose - it will help students with their work, while also ensuring that the field of solving word problems is taken further.

While doing research for this project, we came across various research papers, and highlighted how they solved word problems, and the maximum complexity of the inputs which allows passable accuracy. One peculiar thing we noticed was that while the majority of the solutions followed the same approach - classify, extract, solve - the accuracy remained fixed around 65-70%, and the level of problems that were solvable remained low. This is where we come in - we intend to solve more complex problems. If we succeed, we will turn our attention to simpler problems, and try and increase accuracy.

Next, we introduced visualization of our project - through various diagrams such as WBS, Activity, Gantt, and so on.

In the next section, we introduced how the project will look like - ranging from its working, visualised through a flowchart, to screenshots of the frontend.

**5.2 - ENVIRONMENTAL, ECONOMIC & SOCIETAL BENEFITS**

* This project will benefit not just students, but also frazzled parents, who have to take up the onus of teaching their children Mathematics.
* This project will also benefit the field of NLP, through its subcategory of Word Problem Solving, as little impactful work has been done on the field.
* This project will have no tangible environmental benefits, other than the benefits of students not having to purchase some very cumbersome textbooks, and saving some poor trees.
* Economically, this project is neither a great loss nor a profit - a great amount of man hours were invested in the project, but no money was.

**5.3 - REFLECTIONS**

This project has helped us learn a lot - and not just in the field of NLP or Deep Learning. We have learnt the importance of deadlines, and increasing our performance under pressure. We have seen ourselves working sincerely on something that is not for college studies. We have got to know how stuff really works in the outside world - how people think, how ideas occur, and so on. We have also grasped the importance of working, and keeping on working - earlier on, in our formative years, we used to find interesting topics, and stop learning them after a week or two.

**5.4 - FUTURE WORK**

1. First and foremost, we intend to build classifiers and solvers for the rest of the categories, and not just Simple Interest and Operation Prediction.
2. Secondly, we intend to complete the visualization part of the project.
3. If we are done with the above two parts, then we intend to properly research the work done by other researchers in the field, and improve upon their accuracy.

**SECTION 6 - PROJECT METRICS**

**6.1 - CHALLENGES FACED**

Our project was not without its own share of challenges.

* Our first challenge was our (relatively) weak Natural Language Processing skills for this project. We had to get it up to snuff. Coursera and YouTube were a great help in this regard.
* We then had to study on various research papers. Most of the papers assumed that we already knew the topic the paper was covering, and in great detail. We had to spend inordinate amounts of time on each paper to understand them, but it all eventually paid off in the end.
* We first began manually collecting problems for building the datasets, and also began manually tagging the data. This was an extremely tedious and monotonous job, and this continued for at least a month or so. We resolved the issue by bypassing it entirely - we studied NLTK, Spacy and Keras, which did all the grindy stuff.
* Figuring out AWS was also a pain. We had to host the website somewhere, and all sites were demanding a lot of money, so we hosted our website on AWS. AWS had barely legible documentation, and a lot of stuff was figured out by trial and error. The site though, in a fit of insanity, locked us out of our own system, and deleted the SSH keys - which mean that we could no longer update the site. This was resolved by switching to DigitalOcean, which offered much greater flexibility and ease of use. The costs were hefty ($20 per month), but we didn't mind, as the site was great and snappy.
* Hosting the site on a computer is one thing, but hosting it on a site is a completely different game altogether. We had accomplished the linking through Flask, and learning to use mod\_wcgi was not so easy. Setting permissions properly, enabling security, checking the log manually each time when there was an error - it was tedious, but not boring.

**REFERENCES**

1. D. Castelvecchi, “[We Hate Math,' Say 4 in 10 -- a Majority of Americans](http://blogs.scientificamerican.com/degrees-of-freedom/we-hate-math-say-4-in-10-a-majority-of-americans/)”, October 15, 2011. Retrieved May 23, 2017.
2. B. Amnueypornsakul, (2014) “Automatic Solver for Mathematical Word Problems”, University of Illinois.
3. P. Barendse, (2012) “[Solving word problems with Wolfram|Alpha@ONLINE](http://blog.wolframalpha.com/2012/10/04/solving-word-problems-with-wolframalpha),” Retrieved May 22, 2017.
4. Rik Koncel-Kedziorski, Hannaneh Hajishirzi, Ashish Sabharwal, Oren Etzioni, Siena Dumas Ang, (2015) “Parsing Algebraic Word Problems into Equations”.
5. C. R. Fletcher, (1985) “Understanding and solving arithmetic word problems: A computer simulation,” Behavior Research Methods, vol. 17, no. 5, pp. 565–571.
6. D. Dellarosa, (March 1986) “Solution: A computer simulation of recall of children's arithmetic word problem solving,” University of Colorado Technical Report, pp. 85–148, 1985.
7. D. Dellarosa, (1996) “A computer simulation of children’s arithmetic word-problem solving.” Behavior Research Methods, vol. 18, pp. 147–154,.
8. D. G. Bobrow, (1964) “A question-answering system for high school algebra word problems,” in Proceedings of the October 27-29, 1964, Fall Joint Computer Conference, Part I, ser. AFIPS ’64 (Fall, part I). New York, NY, USA: ACM.
9. Companies using Tensorflow, [Tensorflow](https://www.tensorflow.org/). Retrieved Nov 30, 2017.
10. Allen Institute for Artificial Intelligence Mission Statement, [AllenAI](http://allenai.org/). Retrieved Nov 30, 2017.

**ANNEXURES**

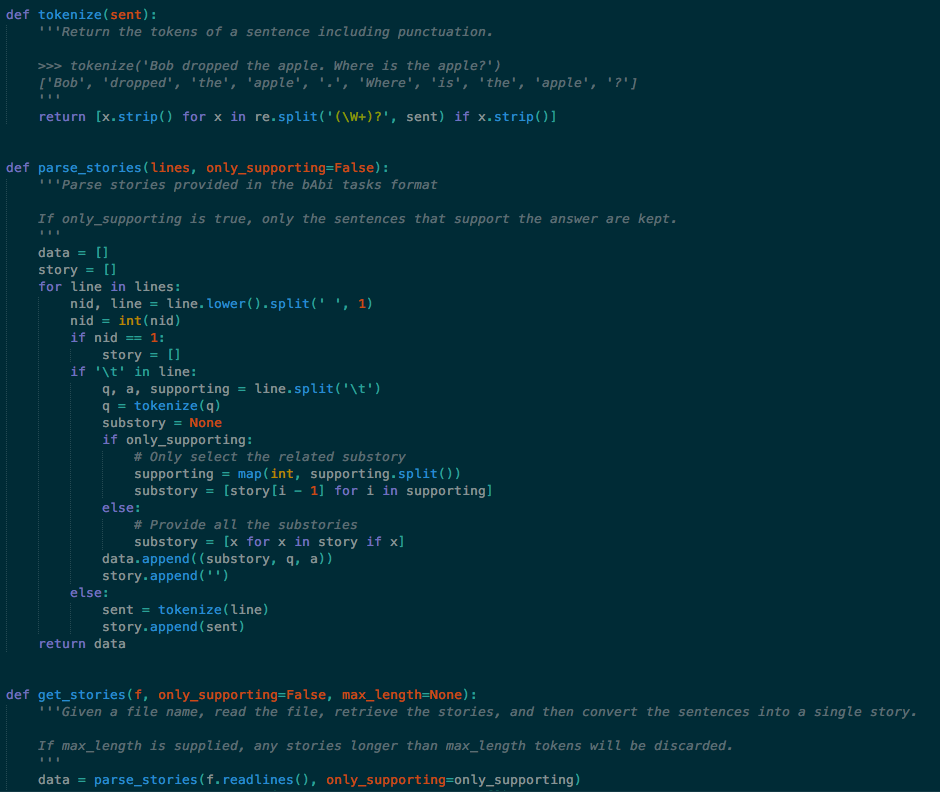


Figure A.1 - OP - I

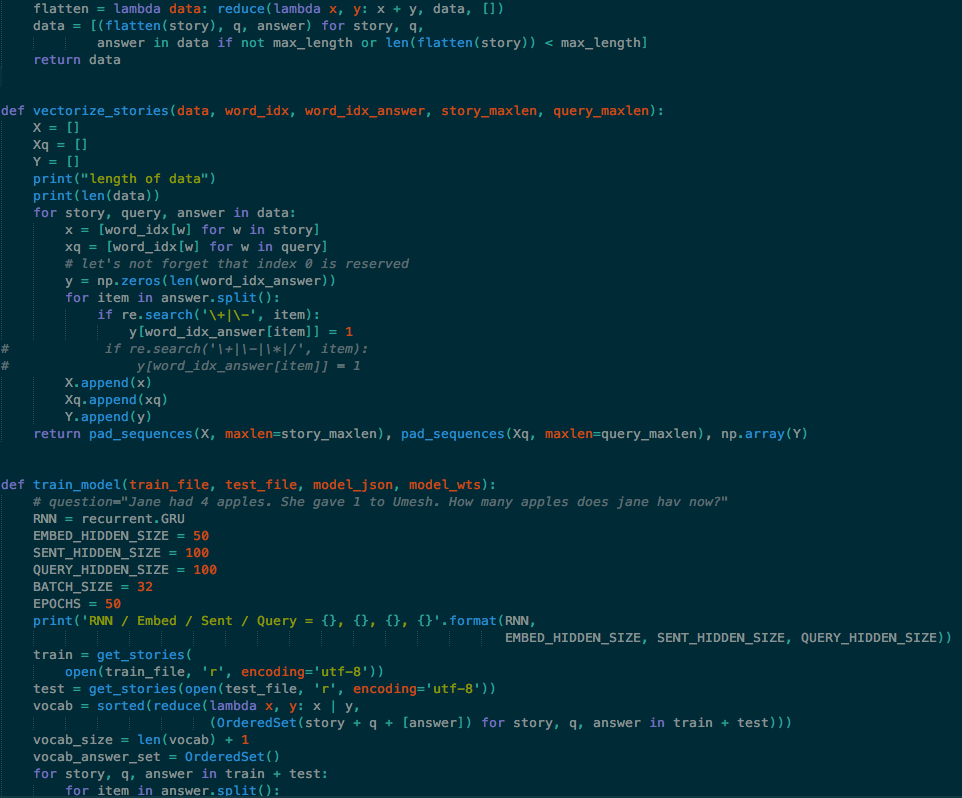


Figure A.2 - OP - II

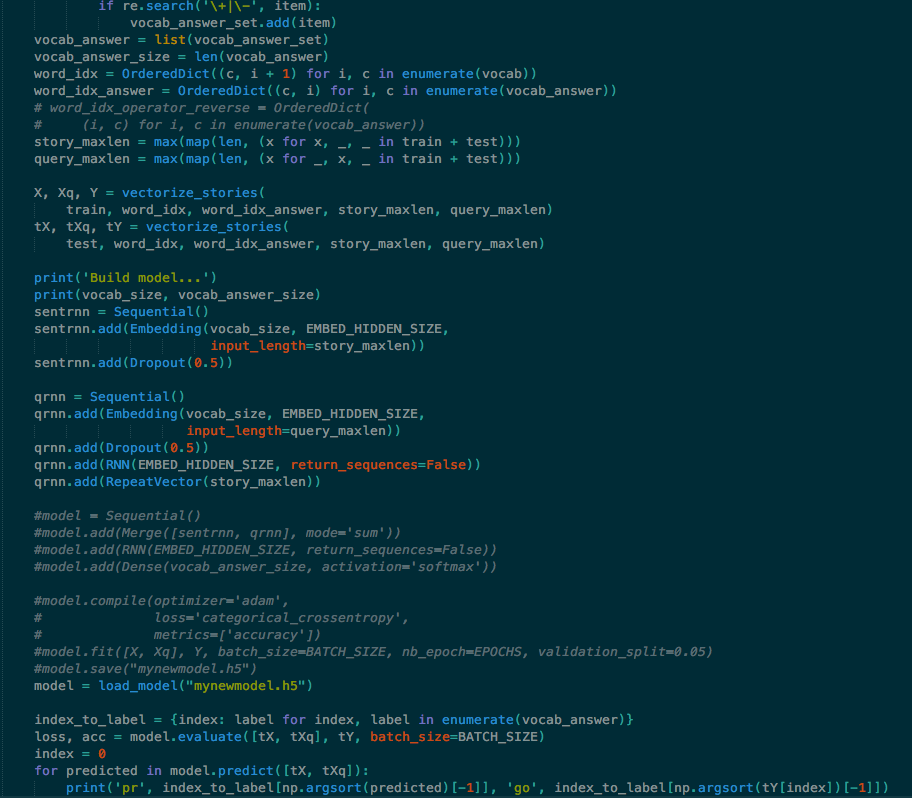


Figure A.3 - OP - III

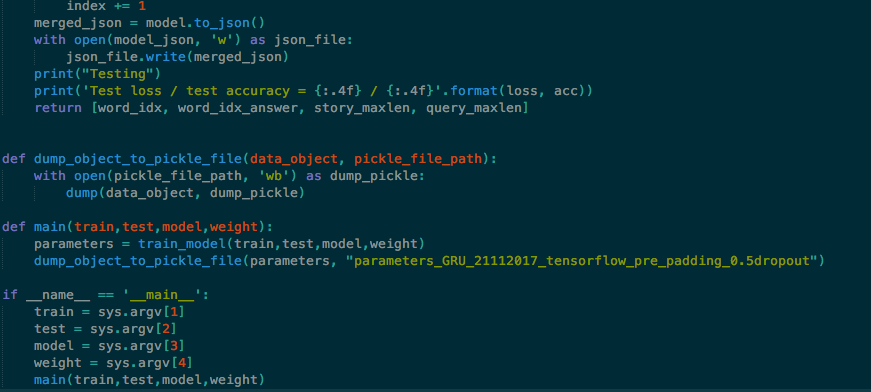


Figure A.4 - OP - IV

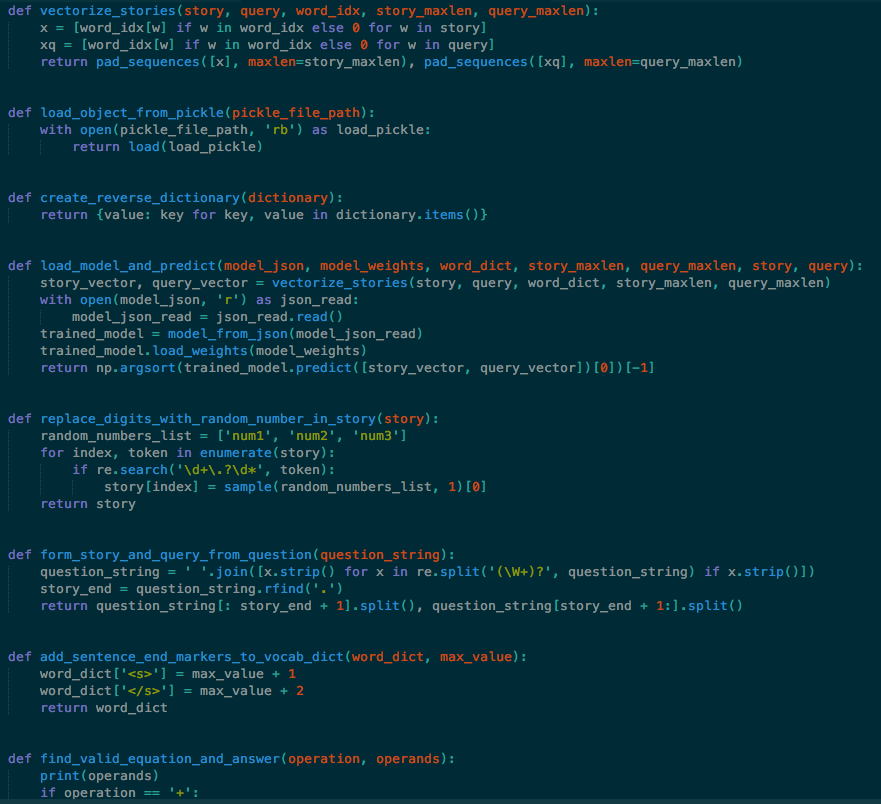


Figure A.5 - OP - V



Figure A.6 - OP - VI