4.1. Proposed methodology of MASMEE

The general scenario of the MASMEE application, where a test paper is generated, test is conducted and evaluation is performed, is shown in Figure 4.1. Examiner Agent (EA) generates test paper based on parameters specified by human examiner and question bank in the database. The Examiner Agent prepares the test paper and saves in the database. The paper setting is performed using Selection Algorithm discussed in detail in next section.

![Diagram of the MASMEE process](image)

Figure 4.1 General Scenario of conducting examination using MASMEE

The Student Agent (SA) proactively collects the test paper from the database and presents it to the students with the help of Gateway Agent (GA) and Web Interface. On completion of prescribed time for the examination, the Student Agent collects the answers from students’ and saves in the database. The Student Agent notifies the Examiner Agent to evaluate the answers. The Examiner Agent fetches the test paper, model answers and all student answers from the database to evaluate the answers’.
The evaluation is performed based on type of examination—subjective, objective or practical. The methodology used for evaluation is shown in Figure 4.2. Subjective evaluation is accomplished using proposed hybrid technique and Domain Ontology based evaluation. The hybrid technique is based on Latent Semantic Analysis (LSA), Bilingual Evaluation Understudy (BLEU) and Fuzzy Logic. The need for second proposed solution using domain ontology was recognized because while developing the hybrid technique, it was realized that if domain ontology is combined with machine learning techniques then better results could be achieved. The design of domain ontology is included. Objective evaluation module can evaluate multiple-choice-questions (MCQ), fill-in-the-blanks (FIB) and one line answers. The MCQ are evaluated using key matching. FIB questions are evaluated using matching with synonym search. The one line answers are evaluated using BLEU technique. The proposed algorithm for Practical evaluation checks the students’ program against several parameters like syntactic correctness, testing of correct output, metrics based on style and efficiency; and semantic similarity of model program with students’ program.
The syntactic correctness is checked using programming language compilers, testing is performed using XUnit and metrics are calculated using tools. The output of all tools is read using Java code. The similarity between model and each student’s program is found using Rapid Subgraph Calculation (rascal) algorithm. All these methods and techniques used in evaluation are discussed in detail in section 4.3 and 4.4.

4.2. Automatic Test Paper Generation

a) Creation of Question Bank: The question bank is prepared (300 questions) and saved in database. MASMEE provides facility to add questions to the question bank. Each question consists of following fields:

- Subject and Subtopic: Each question can belong to one subtopic under one subject. There should be no overlap in the concepts of two questions in order to ensure non-repetition of same concepts in the paper.
- Marks: Each question is assigned marks depending on complexity.
- Difficulty Level: If a question is to state the facts and processes, then difficulty level is less. If it requires more reasoning and thinking then it is difficult. The difficulty level can be between 1 and 5. 1 means easy and 5 means most difficult.
- Type: The questions can be of Subjective, Objective or Practical type.

b) Specify format of test paper: After the question bank is available with the above mentioned fields for each question, the automatic test paper generation module can generate the test paper. The format of test paper should be provided by examiner. The format includes: subject, number of sections in test paper, number of questions in each section, type of paper (subjective, objective or practical) and difficulty level. The number of sections should be less than or equal to number of subtopics. The difficulty level of test paper is the highest difficulty level a question can have.

c) Algorithm used for Paper Generation: The problem of paper setting is now reduced to Selection Problem. It can be solved using the Random function based selection algorithm [91]. The steps in the algorithm are shown in Figure 4.3. The input to the algorithm is the question bank and format for setting up the paper. The algorithm works by selecting questions from database using SQL query that belong to given subject, subtopic, type, and difficulty level (less than or equal to difficulty level of test paper). Then random numbers are generated. System generates as many unique random numbers as the number of questions required in the section. If random numbers are not
unique then they are regenerated. Then, questions at same serial order as value of random numbers in the list of questions fetched are selected to be included in test paper. The Randomized Selection Algorithm has an assumption that there is non-overlapping concept in different questions.

Algorithm RandSelectQuestion (questions_in_database, format_of_test_paper)

1. Map the subtopics of the subject to number of sections in the paper. Assumes number of sections is less than or equal to number of subtopics.
2. Repeat for each subtopic in the subject
3. Fetch the questions from database with difficulty level less than or equal to question paper difficulty level.
4. Repeat for i=1 to number_of_questions in test paper per section
5. Generate unique random number between [1, number_of_questions_fetched]. These random numbers are used as index numbers at which the questions will be selected in list fetched in step 3.
6. From the list fetched in step 3, select the questions at positions equal to random numbers and include these in test paper.
7. end Repeat step 4
8. end Repeat step 2
9. Return the generated test paper as answer.

Figure 4.3 Selection Algorithm for Test Paper generation

4.3. Subjective Evaluation

The answers submitted by students on the online system are textual information. The model answer consisting of keywords expected in the students’ answer is provided as input. The problem is to evaluate the text based answers. The evaluation of subjective answers is viewed as the task to find the correlation between each student answer and model answer. The more similar meaning words are used, the more is the correlation. The keywords in an answer are the main criteria for scoring a technical answer. Therefore, the more domain specific keywords are present in the answer, the more accurate it is. However, we cannot mark the answers by just counting the number of keywords. A more wholesome approach is required, which can evaluate based on
semantic relationship between words and concepts. From the literature review, two areas of research were identified – improving the existing techniques by creating a hybrid technique and use of domain ontology along with existing techniques. Both these areas are explored in this work. A hybrid technique is proposed in this thesis for subjective evaluation. The hybrid technique is based on clustering technique Latent Semantic Analysis (LSA), Bilingual Evaluation Understudy (BLEU) and Fuzzy Logic. It provides the student freedom of word choice and helps in overcoming problems inherent in LSA technique. Several Machine Learning (ML) techniques exist that try to capture the latent relationship between words. The techniques explored in this work are - Latent Semantic Analysis (LSA), Generalized Latent Semantic Analysis (GLSA), Maximum Entropy (MaxEnt) and BiLingual Evaluation Understudy (BLEU). These techniques are applied to Subjective Evaluation in previous work of other authors. The existing techniques show an accuracy of results from 59 to 93 percent. However, the evaluation is purely based on keyword presence. An evaluation process is required which measures the wholesome relationship between words, words and concepts and among concepts. Ontology is a concept map of domain knowledge. This work explores the effect of using Ontology with ML techniques. All the above mentioned techniques are implemented both with and without Ontology and tested on common input data. The use of Ontology with these techniques ensures proper semantic based evaluation as the answers are matched against an exhaustive Knowledge Base. This ensures categorization of answers based on concept category to which the answers belong. Ontology makes the evaluation process holistic as presence of keywords, synonyms, the right word combination and coverage of concepts can be checked.

4.3.1. Proposed Hybrid Technique

The evaluation of subjective answers is performed using a Hybrid of Statistical techniques and Soft Computing technique [11]. Statistical technique for information retrieval- Latent Semantic Analysis (LSA) [92] and technique for Machine Translation- Bilingual Evaluation Understudy (BLEU) [44] are modified and used along with Soft Computing technique- Fuzzy Logic. The Latent Semantic Analysis was selected for this work because it is a clustering technique unlike all other techniques, which are classification techniques. The clustering technique is not dependent on any pre-specified classes/categories. It identifies the classes itself based on clustering of data points. The classification techniques require the categories as input and therefore
cannot model the unknown. However, LSA has two drawbacks. Firstly, LSA cannot distinguish between necessary and unnecessary repetition of keywords. To deal with this, BLEU is combined with LSA. BLEU acts as a clip on the maximum keyword usage. Secondly, LSA looks for exact word matches. To overcome this problem Lexicon Ontology is applied on the initial input. Soft Computing technique namely Fuzzy Logic is used to map the outputs of LSA and BLEU. The relationship between outputs of LSA and BLEU are defined using Fuzzy Logic.

Importance of Fuzzy Logic to Subjective Evaluation: The application of fuzzy logic to Subjective Evaluation using output of modified LSA and BLEU is the novelty of this work. Earlier, combination of LSA and BLEU has been tried by many authors, but they used linear equations with the coefficient of LSA and BLEU defined as variables as shown in Equation 4.1 [47]. The weightage that should be given to BLEU and LSA was left as open question. The final value is sensitive to the value of a, which is a variable and can take any value. Fuzzy Logic provides a consistent treatment to this sensitivity of ‘a’ using membership functions and is shown in this work.

Final_VALUE = (a * LSA_VALUE) + ((1-a) * BLEU_VALUE) …. Equation 4.1

The high level steps of the Hybrid technique are given in Figure 4.4. The input is all the students’ answer and one model answer per question. The output is final marks of the students. First pre-processing of input is done to prepare it for use in evaluation. The tokenization, synonym search and stemming of student answers and model answer is performed. After the pre-processing, Latent Semantic Analysis (LSA) and Bilingual Evaluation Understudy (BLEU) techniques are applied independently. LSA measures the semantic relation between words using dimension reduction technique. BLEU calculates the word average and assigns marks. The output of LSA and BLEU are given as input to fuzzy logic. The output of Fuzzy Logic is final marks of the student.

The steps in algorithm are discussed in detail below:

a) Student Answers and Model answer : The model answer is provided by examiner and student answers are submitted by students when the examination is conducted. Both are typed in English language.
b) Tokenization and Stop Word Removal: Tokenization is breaking the stream of text into single words. All students’ answers and model answer are tokenized into individual words. Some common words that do not add much to technical meaning of the answers are removed entirely from the answers. These words are called stop words. If stop word is not removed, its frequency is counted which has an unfavorable effect on the correlation value. The stop words are identified using regular expressions. Examples of minimal stop word lists that are commonly used include the following:

- Determiners - Determiners tend to mark nouns where a determiner usually will be followed by a noun. Examples: the, a, an, another.
- Coordinating conjunctions – Coordinating conjunctions connect words, phrases, and clauses. Examples: for, an, nor, but, or, yet, so.
- Prepositions - Prepositions express temporal or spatial relations. Examples: in, under, towards, before.

c) Synonyms of words: After extracting individual words from the input answers, the next step is to find synonyms, hyponyms and all forms of the keywords given in model answer [48]. This step is required because student may not always make use of standard keywords and uses some words with similar meaning. For example, word ‘transfer’ has synonyms ‘shift’, ‘convey’, ‘transferred’ etc. The student can use any of these words in his answer. Synonyms are found using an available Lexicon Ontology.
d) Stemming using Replacements: The words have a basic stem from which it can take multiple forms [93]. For example, the word education has forms: educated, education, educating, educate. The stem in all these words is “educ”. Similarly, all words have a stem. The general structure of a word consisting of consonants (c) and vowels (v) is: \([c](vc)^m[v]\), where, \(m\) is the number of vowel-consonants pairs. There are different values of \(m\) between 0,1 and 2 for which replace operations are to be performed. Some of the stemming replacements are shown in Figure 4.5. The order of the replacements must be kept intact (line by line, left to right). The value of \(m\) must be calculated by counting letters. The detailed algorithm is in [93].

```
replaceAll("ssess ","ss ");
replaceAll("[r]s "," ");
replaceAll("ed "," ");
replaceAll("ion\[s\] "," ");

replaceAll("ies ","i ");
replaceAll("eed ","ee ");
replaceAll("ing "," ");
replaceAll("at ","ate ");
```

Figure 4.5 Example rules for Stemming using Replacements

e) Latent Semantic Analysis: Latent Semantic Analysis (LSA) [92] is a technique in natural language processing for analyzing relationships between a set of documents and the terms they contain. The basic assumption is that there exists a hidden semantic space in each text that is the accumulation of all words meaning. It usually takes three steps to compress the semantic space - filtering, selection and feature extraction. First, the stop words are filtered. Second, word frequency matrix is constructed by selecting reference texts. Third, singular value decomposition is done to extract features by factorizing the feature matrix. Last, cosine similarity is found between feature vectors. The cosine similarity of these vectors (correlation value) signifies the degree of relation between the student answer and the keywords. The semantic presence of the keywords in student answers is indicated by higher cosine correlation value. The standard LSA technique is modified in two ways. Firstly, Pre-processing the input to find synonyms and performing stemming. This makes the output of LSA more precise as all the forms of the words are provided for. It gives the students freedom to use any similar word or form of same word depending on tense and sentence structure being used. Secondly, in this work the selection of words (second step above) in LSA is modified. The vocabulary required for calculating word frequency is originally prepared by counting unique terms in model answer or reference text. This is modified by counting unique terms in model answer and unique terms in all student answers, thus building a complete model of all the student answers and model answer.
The detailed steps of modified LSA technique are given in Figure 4.6. First, the pre-processing steps of stop word removal, tokenization, synonym search and stemming are performed (discussed above). Then, word frequency matrix is calculated. Singular valued decomposition (Eigen decomposition of non-invertible matrices) is performed on term-frequency matrix. This generates vectors of terms and answers. These term and answer vectors are multiplied to get cosine similarity. An example of working of LSA is given in Appendix B.

Algorithm Latent Semantic Analysis (model answer, Student Answer)

Variables Used in Algorithm

- Terms matrix of order 1XM. It has all the keywords from model answer and unique words from students’ answer.
- Stu matrix of order NX1. It contains all the students’ answer.
- Matrix ‘A’ of order MXN. It is the frequency value a[i][j] of number of times term[i] appera in stu[j].
- S is the left Eigen vectors of $A^T A$.
- E is the square roots of Eigen values of A.
- U is right Eigen vectors of $AA^T$.
- SE matrix of order based on rank of E is product of S and E.
- D matrix is product of E and U.
- Q is sum of all rows of SE matrix where, se[i][j] is added if the ith element in Term is included in model answer.

1. Find out all the unique keywords in model answer as Term 1XM matrix. Add the list of unique words in student answers to this matrix. Note that no words are repeated in Term matrix.
2. Keep all student answers in an NX1 matrix, Stu NX1.
3. Perform stop word removal, tokenization, synonym search and stemming on model answer and all student answers.
4. Construct the term to document frequency (tdf) matrix- $A_{mXn}$. Here m is number of terms and n is number of answers. The rows of A correspond to entries in Term matrix and columns correspond to Stu matrix. … continued on next page
a[i][j] = 0 if ith word in Term is not present in jth answer in Stu.

a[i][j] = x, where x is the number of times the word appears in answer j in Stu.

5. Calculate Singular Value Decomposition (SVD) => \[ \text{SVD}(A_{mXn}) = S \times E \times U^T \]

Where, \( S \) = Left Eigen Vectors of \( A^T A \). Terms in the concept space are represented by row vectors of \( S \);

\( E \) = Identity of square roots of Eigen values of \( A^T A \). It gives the degree of relationship between the \( S \) and \( U^T \) matrix.

\( U^T \) = Right Eigen Vectors of \( A A^T \). Documents are represented by column vectors of \( U^T \).

6. Reduce the dimension of \( S, E \) and \( U \) by finding Rank of \( E \). This step removes noise in the data.

7. Calculate \( SE = S \times E \). It gives the participation of term in the model answer (concept).

8. Calculate \( d = E \times U^T \). It describes the participation of student answer in the model (concept).

9. Compute the weight vector of the keywords in model answer as \( q \) using following method:

\[
\text{for } j=1 \text{ to } N \quad q[j] = \sum_{i=1}^{n} se[i][j] \quad \text{where, } se[i][j] \text{ is an element of } SE.
\]

Each row of \( SE \), \( se[i] \) signifies a word vector. By adding these individual vectors for the words included in list of keywords in model answer, the term vector weight is found. The result in \( q \) is a column matrix of order 1XN.

10. Calculate cosine similarity between \( q \) and \( d_i \), where \( i \) is the student answer number in Stu matrix, as:

\[
(\text{Vector}(q)\times\text{Vector}(d_i))/(\text{mod}(q)\times\text{mod}(d_i))
\]

It gives the correlation between the query words and student answer. Higher the value of cosine correlation, more are the chances of occurrence.

Figure 4.6 Steps in Latent Semantic Analysis Technique

f) Bilingual Evaluation Understudy: The BLEU technique [44] is used to overcome the drawback of LSA technique. LSA overrates the answers that repeat the keywords many times. It means if a student repeats the same sentence a number of times then LSA evaluates it as good answer. However, in reality it is a bad answer. To overcome this
problem, BLEU is used. BLEU generates a metric value ranging between 0 and 1. The value indicates how similar student answers are to the standard answer. The frequency of a keyword in student answer and total number of words in student answer are calculated. These values cannot be more than frequency of each keyword in model answer and total number of keywords in model answer respectively. If any value is more, then the corresponding value calculated from model answer is used. Divide the frequency of each keyword in student answer by the total number of words. The algorithm as used is given in Figure 4.7.

A modified version of BiLingual Evaluation Understudy (BLEU) algorithm is used.

- Firstly, the original BLEU algorithm makes use of n-grams (phrases of words). In this work, individual words are used because stop words are removed and phrases cannot be constructed.
- Secondly, the original method calculates brevity factor and multiplies the bleu value with it. Brevity factor (BF) is exponent value of r, where r is number of words in student answer divided by total words in model answer. The significance is not to penalize a short response. In this work, BF is not calculated. BLEU is used to clip the maximum usage of keywords so that unnecessary repetition of keywords does not fetch more marks to students.

Algorithm BLEU (model answer, students’ answer)

Variables Used
- Term matrix of order 1XM – unique words in model answer.
- masterFrequencyMatrix – frequency of words in Term in model answer.
- Stu matrix of order NX1 – all student answers
- bleuRepOfKeywords matrix of order NXM – repetition of each model answer keyword in students’ answer. N is the number of Answers in Stu and M is the number of keywords in Term.
- num- stores the numerator for word average for bleuVal
- den – stores the denominator of word average for bleuVal
- bleuVal- the final bleu score

1. The Preprocessing steps- tokenization, stop word removal and synonym search are applied to input.

.... Continued on next page
g) Use of Fuzzy Logic in hybrid technique: The output of BLEU and LSA are mapped using Fuzzy Logic. Fuzzy Logic is an extension of two-valued logic to handle the concept of partial truth. Compared to traditional crisp variables, a fuzzy variable has a truth value varying between 0 and 1 showing there degree of membership. LSA and BLEU both generate correlation value between 0-1. In sharp set, zero correlation is interpreted as No relationship. A correlation value less than or equal to 0.30 represents weak relationship, 0.50 represents moderate relationship and 0.70 represents strong relationship. Value less than or equal to One represents a perfect relationship. However, this interpretation does not deal well with boundary cases like correlation value of 0.51 is in moderate relationship (0.50) or strong relationship (0.70). Therefore, Fuzzy variables are used to define the correlation values. Both LSA and BLEU give output as degree of correlation, so they are used as fuzzy input variables. These two independent
variables are pointing towards the different aspects of level of similarity between model answer and students’ answer. This work defines two input variables named LSA and BLEU with three membership functions (bad, average, and excellent) and one output variable Final with four membership functions (bad, ok, average and excellent). Trapezoidal class is used for all membership functions. The inference engine rules of fuzzy logic are shown in Table 4.1. Whenever the BLEU and LSA technique both give bad correlation value, the output is bad. When any one is average or excellent and the other is bad, the result is ok. When both are average, result is average. Lastly, when both give excellent correlation, result is excellent.

<table>
<thead>
<tr>
<th>Input1 (LSA)</th>
<th>Input2 (BLEU)</th>
<th>Output (final)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bad</td>
<td>Bad</td>
<td>Bad</td>
</tr>
<tr>
<td>Bad</td>
<td>Average</td>
<td>Ok</td>
</tr>
<tr>
<td>Bad</td>
<td>Excellent</td>
<td>Ok</td>
</tr>
<tr>
<td>Average</td>
<td>Bad</td>
<td>Ok</td>
</tr>
<tr>
<td>Average</td>
<td>Average</td>
<td>Average</td>
</tr>
<tr>
<td>Average</td>
<td>Excellent</td>
<td>Average</td>
</tr>
<tr>
<td>Excellent</td>
<td>Bad</td>
<td>Ok</td>
</tr>
<tr>
<td>Excellent</td>
<td>Average</td>
<td>Average</td>
</tr>
<tr>
<td>Excellent</td>
<td>Excellent</td>
<td>Excellent</td>
</tr>
</tbody>
</table>

Table 4.1 Rules for Inference Engine of Fuzzy Logic

4.3.2. Proposed Ontology based Evaluation

In the previous section, subjective evaluation is performed using hybrid technique based on LSA, BLEU and Fuzzy Logic. The Lexicon Ontology tool was used to find synonyms of words in student answers. It was identified that if domain ontology is used instead of model answer then better evaluation can be performed. Therefore, in this section several statistical techniques like LSA, Maximum Entropy (MaxEnt) are combined with domain ontology to perform subjective evaluation.

The scheme of evaluation is given in Figure 4.8. First, the pre-processing steps are performed. The processed answers are given as input to statistical techniques along with subject specific ontology. The statistical techniques used are – Latent Semantic Analysis, Generalized Latent Semantic Analysis, BiLingual Evaluation Understudy, proposed hybrid technique and Maximum Entropy. These techniques are discussed below.
4.3.2.1. Information Retrieval Statistical Techniques

The input to all the techniques is Students’ Answers. The output is a similarity measure in the range of [0, 1], where a value of 0 indicates no similarity and 1 indicates high similarity. The input is pre-processed before applying the techniques. The steps of pre-processing are: tokenization (find all words in all students’ answer), stop word removal (removing common words like a, the, as, an etc.) and synonym search (for each word left after stop word removal, find its synonyms) as already discussed in previous section. Stemming is not performed as Ontology concept classes cannot be stemmed so matching will become difficult.

Latent Semantic Analysis (LSA), Bilingual Evaluation Understudy (BLEU) and Hybrid techniques are already discussed in previous section.

Generalized LSA (GLSA): GLSA [29] finds the Term matrix as used in LSA algorithm (Figure 4.6) by generating phrases (ngrams) from model answers and students’ answers. Ngrams are formed by placing the n adjacent words in model answer together and moving right word by word. The phrase length used is 2, 3 and 4 neighbouring words. Example, ‘computer_graphics_system’ is a 3 gram. These phrases constitute the Term matrix. The frequency of each phrase in students answer is calculated to generate tdf (AMxN matrix). The remaining steps are same as in LSA. The Algorithm is included in review of literature in Chapter 2.
Maximum Entropy (MAXENT): In MaxEnt [33], the input is training essays (multiple model answers) and student answers. The training essays are labelled with category to which they belong like good, bad, etc. No pre-processing steps are performed. The training data is used to study the word context – that is words that mostly follow and precede the word under consideration. The entropy is calculated for the current word to appear in a given context. This entropy is calculated for each word in model answer. The entropy for all word pairs in the model answers and corresponding target category are given to Perceptron for training. This builds a model using which the student answer evaluation is performed. The detailed algorithm is included in review of literature in Chapter 2.

4.3.2.2. Design of Ontology

Ontology is the study of groups or classes of things [50], [53], [94]. It is a classification of concepts in a domain. Domain Ontology represents the concepts which belong to part of the world. Particular meanings of terms applied to that domain ontology. Example, the word ‘card’ has many meanings. Ontology about domain of ‘poker’ has meaning playing card while ontology of computer hardware has a meaning of ‘punch cards’ and ‘video cards’. Ontology is a formal explicit description of concepts in a domain of discourse (classes or concepts), properties of each concept describing features and attributes of concepts (slots) and restrictions on slots (facets). Ontology along with set of individual instances of classes constitutes a knowledge base. The Domain Ontology of Computer Graphics is prepared using subject-predicate-object representation, where subject is a class or instance, predicate is property of class or instance and object is values of properties. The following components of Ontology are defined for Computer Graphics subject:

1) Classes: Sets, collections, concepts and types of objects. These represent the concepts in a domain that cannot be initialized. These are names of categories, events, algorithms, etc. For example, the main class is ‘thing’. Under ‘thing’, the starting class is ‘Computer_graphics’. ‘Computer_graphics’ has subclasses including ‘computer_graphics_applications’, ‘Computer_graphics_systems’ and ‘types_of_media’. The ‘Computer_graphics_systems’ has subclasses ‘hardware’ and ‘software’. Further classification of each is performed as depicted in Figure 4.9(a-c).
The algorithms and processes are designed as classes with steps as individuals as properties. The classification is depicted in Protégé tool in Chapter 5.

2) Individuals: The classes have instances or objects of classes. For example, ‘image_processing’ class has several applications. These applications are designed as individuals of ‘image_processing’ namely, ‘color_coding’, ‘improve_picture_quality’, ‘improve_quality_of_shading’, ‘machine_perception’ and ‘rearrange_picture_parts’. The instances further have properties. Instances cannot have children or subclasses or instances but has attributes like graphics packages implementing these applications – ‘hasPackage’. The algorithms implementing these applications are represented as classes and linked to individuals of respective ‘image_processing’ applications.

3) Attributes: The classes and individuals have properties. Some of the attributes identified are: ‘uses’ with values depicting application of device, ‘technique’ depicting name of technique used in the device, ‘purpose’ depicting the purpose like input or output or pointing device, ‘standard’ depicting the universal standards followed and adopted by the devices. Each class has separate attributes. The ‘Events’ class has subclasses depicting working of devices. These classes have attributes like ‘actor’ depicting the part of device performing the action, ‘target’ depicting the part of device effected by event, ‘output_of_event’ depicting the result of action or event.

4) Relations among Classes: Apart from subclass relation, Ontology uses disjoint and equivalence relations for linking classes and subclasses of different hierarchy. Example, ‘Working_of_CRT’ is a subclass of ‘video_display_devices’. Former is also a subclass of ‘Events’. So equivalence is created between the child nodes of the ‘Events’ and ‘video_display_devices’ classes.

5) Process representation: There are many processes and algorithms in graphics for example, ‘working_of_CRT’, ‘working_of_plasma_panel’, etc. These are represented as event class individuals. These have properties like ‘actor’, ‘target’, ‘output_of_event’, ‘input_to_event’, ‘part_of_event’, ‘predecessor’ and ‘successor’.

The implementation of Ontology and related snapshots are included in Chapter 5 in Section 5.3.3. The designed Ontology is a dummy Ontology and does not cover exhaustively all the concepts of Computer Graphics. The complete Ontology is includes in Appendix C.
(a) Ontology class – Computer Graphics

(b) Ontology Class – Computer_graphics_system
4.3.2.3. Using Ontology with existing statistical techniques

When the students’ answers are to be evaluated, the details related to the concept are fetched from the Ontology. The point of access is provided by human examiner along with model answer. When the students’ answers for the question are to be evaluated, the details related to the concept are fetched from the Ontology. The level of detail will depend on the type of question as shown in Table 4.2. When short questions are to be answered then directly related information is fetched. When longer questions are to be answered then more details are fetched.

The combination of Ontology with the statistical techniques is performed using the design depicted in Figure 4.8 [12]. After performing pre-processing, the students’ answers are given as input along with Subject Specific Ontology to statistical techniques.

The steps performed for each statistical technique (discussed in previous section) are shown in Figure 4.10. The sentences in each student answer are classified as belonging to an ontology concept using statistical techniques. Then, total number of concepts
found in each answer is divided by total number of concepts in concept map to generate the similarity score between concept map and student answer.

<table>
<thead>
<tr>
<th>Type of Question</th>
<th>Level of Detail fetched from Ontology</th>
</tr>
</thead>
<tbody>
<tr>
<td>One-line questions</td>
<td>Concepts direct properties, instances and subclasses.</td>
</tr>
<tr>
<td>Short Length questions</td>
<td>All the concepts under the main concept.</td>
</tr>
<tr>
<td>Long Answer Questions, stating facts and phenomenon</td>
<td>All the concepts under the main concept along with equivalent and inverse classes.</td>
</tr>
<tr>
<td>Essay Length questions, reflective and open ended</td>
<td>All the concepts directly under the main concept and related concepts with inverse, equivalence, part-of, steps in phenomenon, etc are fetched. All the nodes directly related to main concept and indirectly related are fetched in the form of Triples.</td>
</tr>
</tbody>
</table>

Table 4.2 Criteria to fetch information from Ontology

Algorithm OntologyIREvaluation(Ontology, model answer, student answer)
1. Extract the Ontology related with question for which answers are to be evaluated taking the ontology access point mentioned at time of adding the question to database of MASMEE.
2. Represent all students’ answers (i) and sentence in student answer (j) in matrix form.
3. Classify each sentence in students’ answer as belonging to one of the ontology concept using following steps.
   a) Split the student answer into constituent sentences.
   b) Then perform the following steps.
      BLEU- count the frequency of each word in each answer and all words in each concept. Find bleuVal for each sentence with each concept. The concept with which bleuVal is highest, is the concept to which sentence belongs.
      LSA – perform SVD for each sentence with the ontology concepts replacing the Term matrix of order 1XM (Figure 4.6).
      GLSA- form phrases of length 2,3, and 4 from ontology concepts and students’ answer and then perform LSA. 

.....Continued on next page
4.4. Objective Evaluation

Objective questions can be multiple-choice questions, fill-in-the-blanks and one-line questions. The input is student answers and model answer. The multiple choice questions and fill-in-the-blanks require matching of model and student answer. The one-line question requires calculating correlation between student answer and model answer.

The high level steps for objective answers evaluation is given in Figure 4.11. For Multiple Choice questions simple matching is performed. For fill-in-the-blanks, Synonyms based matching is used. The synonyms and all forms of the keyword are found using WordNet Tool and these are matched with student answer. If any one of these synonyms and all forms of the word matches, marks are assigned. For One-line questions the BLEU technique developed for Subjective questions is used. The bleu score of student answer is calculated by finding total number of words in student answer and frequency of each keyword in student answer. Similarly, total number of words in model answer and frequency of each keyword is calculated. Then Values obtained for each student answer are compared with values of model answer. If student answer’s value is higher, the model answer value is used to find ratio between the two calculated values of student answer.
4.5. Practical Evaluation

The objective of practical evaluation is to determine that student programs generate correct output and follow the efficiency and style metrics. The programs are evaluated based on four parameters: syntactic correctness, output correctness, metrics (style and efficiency) and similarity of logic to model program. The model program, test cases and student answer are required as input.

The size of programs evaluated by the software is from small practice questions to single file programs. The scope of the current system is limited to C, C++ and Java programming languages. The Program Dependence Graph of C++ is not generated, so similarity is not found for C++ programs.

4.5.1. Proposed Steps for Practical Evaluation

The diagram showing the different steps in practical evaluation is given in Figure-4.12. The input is students’ programs, model program and Test Cases written in XUnit packages. The details of writing XUnit package programs are given in Appendix D. The steps for evaluation are compilation, testing, metrics and similarity to model program. The steps of practical evaluation are discussed below. The tools used in evaluation at each stage are shown in Figure 4.13.
a) Compilation: in the first step, the syntax of the program is checked. The students who pass in this stage will go to next step. The Programming language compilers including Turbo C++, GNU GCC and Jdk 1.8, are used for performing compilation. In case of unsuccessful compilation, the number of errors reported by compiler is stored as reported errors (RE). These errors are processed to remove propagated error reports and to find the actual errors (AE). The propagated error reports are errors reported by compiler that are dependent on a previous error. Some syntactic mistakes like misspelling of data-types, missing comma, semi-colon, incorrect declaration of variables, use of incorrect variable names, etc leads to errors reported in surrounding lines. However, the main problem is not in all the reported lines but just the starting line.
number. In order to deal with this problem of propagated errors, the line numbers in
which errors are reported are checked for consecutive line number. If errors are
reported in consecutive lines, then only the error in first line number among them is
counted as actual error and the remaining are ignored. Sometimes the error is actually
there in all the lines but it is rare case. This way of propagated error removal helps to
give marks to students whose small mistakes lead to non-compilation of programs. A
systematic approach is to allot marks if errors are less than four errors. The marks given
are proportional to number of errors with the formula: marks for compilation X
(number of AE/4).

Figure 4.13 Tools in Practical Evaluation

b) Testing: After the program is successfully compiled, it needs to be tested for
functionality by performing testing of the programs. In testing, it is checked that
whether all the modules are working as desired or not. The test-cases are provided by
the examiner written as programs of XUnit package [95]–[97]. The test-cases are run
on model answer to find the expected output. Then the student program is evaluated for
the same test cases. If there is a match between the expected output and student
program output, it is a success. For ensuring the success of this functionality, the
signatures (number of arguments and return value) of functional units in students’
program must match the model program. The specifications for the same are given to
student in question itself. Testing is performed using XUnit framework. On successful testing 60% marks of the total marks are given. If testing is unsuccessful, then no marks are given. It is mandatory to pass all test cases.

c) Metrics Calculation: After the program is tested for correctness of output, its complexity and style needs to be evaluated. The used metrics related to style are Lines of Code (KLOC), Lines of Comment and Number of Modules. The used metrics related to efficiency are McCabe's cyclomatic complexity, time complexity and space complexity. CCCC Tool (C and C++ Code counter) is used to find the first four metrics. Time complexity is calculated by counting time taken to execute the program. The space complexity is found using Regular expressions for counting unique variable names. Arrays are not handled in space complexity. The metric values of student program are matched with that of model program. If the lines of code are more in student program, then no marks are given. If lines of comment are less, then no marks are given. If number of modules is less than model, then no marks are given. If McCabe's cyclomatic complexity is more, then no marks are given. The time and space complexity of student programs are more, then no marks are given.

d) Semantic similarity: The semantic similarity of the student program to the model program is established. This is performed by generating and comparing the Program Dependence Graphs (PDG) of student program and model program. The PDG [98] represents a program as a graph in which the nodes are statements of the program and the edges represent both dependence of data flow or control flow between statements. Dependences arise as the result of two separate effects. First, dependence exists between two statements whenever a variable is initialized or assigned values in first statement and used in second; and incorrect results will be achieved if order of the statements is changed.

For example, given \( F=X+H \ldots S1 \)
\( Y=F\times X+1 \ldots S2 \)
S2 depends on S1, since executing S2 before S1 would result in S2 using an incorrect value for A. This is data dependences. Second dependence exists between a statement and the predicate whose value immediately controls the execution of the statement. In the sequence:

\[
\text{if (A)} \quad \ldots \quad S1 \\
\text{then } B=C*D \quad ; \quad \ldots \quad S2 \\
\text{endif}
\]
S2 depends on predicate A since the value of A determines whether S2 is executed. Dependences of this type are control dependences. The details of PDG are given in Appendix D. These PDG graphs are generated using existing tools and libraries [80], [99].

The exact match of the graphs indicates complete similarity but sometimes a part of the program is common. The need was identified to find partial similarity as well. According to literature review, if two programs are common, they will have same control and data flow. If the two graphs are Isomorphic or contain Isomorphic Sub graphs then we can say they are somewhat similar. This problem is known as Maximum Common Subgraph (MCS) problem. The MCS problem is solved using Rapid Subgraph Calculation (Rascal) algorithm [14]. Rascal is a heuristic method to find common part of the graph and efficient in terms of time. Another important feature is that SDG and PDG have three types of edges. Rascal specializes in graphs with more than one type of edge.

The algorithm Rascal is given in Figure 4.14. First, all the vertices are sorted in the order of degree, in both the graphs for all the types edges i.e. if there are two types of edges then two sorted lists per graph are created. Second, each vertex pair (w,v) is matched when degree of w (student graph) is less than or equal to degree of v (model graph vertex). In case the number of nodes are less in one of the graphs then dummy vertex included in end with degree zero. Third, count the number of vertices in each type of edge in PDG of student program for non-zero degree as vgi and add its degree in eg, for each graphi. Fourth, map the nodes of student graph with model graph as performed in second step and lastly, calculate vgg and egg by counting number of vertices with non-zero degree in merged graph in vgg and summation of degree in egg.

\[
simcv(G1,G2) = \frac{(vgg + egg)^2}{\pi((vgi +egi)/2)} \quad \text{.........} \quad \text{.........} \quad \text{...Eq 4.2}
\]

where i is edge type.
The value of \( \text{simcv}(G1,G2) \) is between \([0,1]\). The more the value is towards one, the more is degree of similarity.

Algorithm MCSRascal(pdg of student program, pdg of model program)
1. Define node as (vertex number, degree).
2. Define StuClinks, Stualinks, Studlinks, modclinks, modalinks and modDlinks as arrays of type node.
3. Find the clinks, alinks and dlinks for each graph and maintain vertex number with each type of degree. Assign values to StuClinks, Stualinks, Studlinks, modclinks, modalinks and modDlinks arrays.
4. Sort all arrays StuClinks, Stualinks, Studlinks, modclinks, modalinks and modDlinks in order of degree.
5. For student graph arrays: StuClinks, Stualinks and Studlinks, count all vertices with non-zero degree in \( \text{vg}1 \) and keep adding the degree in \( \text{eg}1 \).
6. For model program graph arrays: modclinks, modalinks and modDlinks, count all vertices with non-zero degree in \( \text{vg}2 \) and keep adding the degree in \( \text{eg}2 \).
7. Compare the clinks, dlinks and alinks arrays of student graph and model graph with each other. Create mapclinks, mapalinks and mapdlinks such that:
   - if degree(stuclinks\(_i\) < modclinks\(_i\)) then mapclinks\(_i\) = stuclinks\(_i\)
   - otherwise mapclinks\(_i\) = modclinks\(_i\)
   for all \( i \) nodes in model and student graphs.
8. For each mapclinks, mapalinks and mapdlinks – count all vertices with non-zero degree in \( \text{vgg} \) and add the degree values in \( \text{egg} \).
9. Calculate \( \text{Sim} = \frac{((\text{vgg}+\text{egg})/2)/(((\text{vg}1+\text{eg}1)/2)*((\text{vg}2+\text{eg}2)/2))} \)

Figure 4.14 Rascal Algorithm for Maximum Common Subgraph

4.5.2. Integrated 4-step Evaluation

The marks assignment strategy is depicted in Figure 4.15. The total marks are divided into four parts based on weightage of each part: The first part is given to the student if the program compiles successfully, second part is given if program has semantic similarity to model program. Third part is given in full if it passes the testing stage. The Fourth part of marks is given based on efficiency and complexity metrics. The default weightage for compilation is 10 percent, semantic similarity is 10 percent, 60 percent
for testing and 20 percent for metrics. However, there is a provision to change these values in MASMEE.

```
Algorithm PracticalEvaluation (Student_Programs_list, Model_programs, Marks)

Evaluate all the four stages for model program – namely syntax, functional testing, metrics and similarity score.

For each student program in student_Program_list

• Check student program compiles successfully and save error_no if errors are reported.
• Check the functional testing and record num_of_test_passed and num_of_test_failed.
• Calculate the metrics – num_loc, num_locom, num_mod, mccabe, time_complex and space_complex.
• Compare the above parameters of student program with model answer and assign marks according to the following formula:
  • If(error_no == 0)
  • assignM = assignM+ (marks * 0.10) // compilation marks
  • assignM = assignM + (marks*0.60) // testing marks
  • if(metrics of student program same as or better than model program)
  • assignM = assignM + (marks*0.20) // assign marks for metrics
  • end if
• Find similarity of student program with model program
  • assignM = (marks*0.10)* similarity score; // assign successful compilation marks if similarity score is non-zero
• End if
• End for

End Algorithm PracticalEvaluation
```

Figure 4.15 Integration Algorithm for Practical Evaluation