**Automated Learning Outcome System**

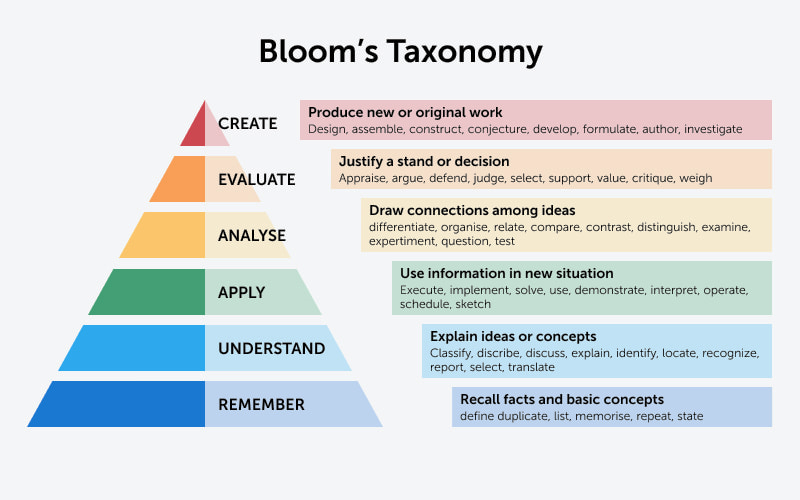
**1. Introduction**

Automated Learning Outcome Extraction System using Natural Language Processing (NLP) is a state-of-the-art technology that enables the extraction of relevant learning outcomes from educational texts. Learning outcomes are statements that describe the knowledge, skills, and competencies that learners are expected to acquire upon completion of an educational program. They are important for ensuring that educational programs are effective and meet the needs of learners and employers.

Traditionally, the process of extracting learning outcomes from educational texts has been time-consuming and labour-intensive, requiring manual reading and analysis of each text. However, with the advent of NLP, it is now possible to automate this process, making it faster, more accurate, and more scalable. It involves the use of machine learning algorithms and linguistic analysis to identify and extract relevant learning outcomes from educational texts. This system can be trained on large datasets of educational texts, enabling it to recognize patterns and relationships in the language used to describe learning outcomes. It can also be customized to the specific needs of different educational programs and contexts. It is a project aimed at developing a software application that automatically extracts and categorizes learning outcomes from course syllabus. Learning outcomes are statements that describe what students should know or be able to do after completing a course. They are critical for evaluating the effectiveness of a course and ensuring that it meets the learning objectives of the program.

The benefits of this project are numerous. It can save educators and administrators significant amounts of time and resources by automating the process of extracting learning outcomes. It can also improve the accuracy and consistency of learning outcome extraction, reducing the potential for errors and discrepancies. The goal of this project is to use NLP techniques to extract learning outcomes from syllabus, which will save instructors and administrators time and effort in developing and evaluating courses. The system will use machine learning algorithms to analyse the text of a syllabus and extract relevant information about the course, such as the course objectives, goals, and expected outcomes.

Automated learning outcome systems are becoming increasingly popular in educational institutions, as they offer a way to assess the effectiveness of teaching programs in a consistent and objective manner. By automating the process of data collection and analysis, these systems can help educational institutions save time and resources, while improving the quality of education provided to students. One of the key benefits of an automated learning outcome system is that it can provide real-time feedback to both teachers and students. This feedback can help teachers identify areas where students are struggling and adjust their teaching strategies accordingly. It can also help students understand their strengths and weaknesses and develop personalized learning plans that address their individual needs. Finally, it can provide valuable insights into the effectiveness of educational programs by analysing the learning outcomes that are being achieved.



By utilizing Bloom's Taxonomy, the automated learning outcome extraction system can provide a comprehensive and standardized framework for extracting and categorizing learning outcomes from educational texts. This can provide valuable information to educators and learners, helping to promote higher-order thinking and skill development.

An automated learning outcome extraction system using natural language processing can use Bloom's Taxonomy as a framework to extract learning outcomes from educational texts. For example, the system can identify sentences containing verbs that correspond to each of the six levels of Bloom's Taxonomy.

**2. System Requirements :**

**2.1. Hardware Requirements:**

* Core i3/i5 processor
* At least 8 GB RAM
* At least 60 GB of Usable Hard Disk Space
* NVIDIA graphics card with CUDA Compute Capability version 3.5 to 8.6. See the list of [CUDA-enabled GPU cards](https://developer.nvidia.com/cuda-gpus).
* [NVIDIA GPU driver](https://www.nvidia.com/Download/index.aspx?lang=en-us) version: Windows 461.33 or higher, Linux 460.32.03 or higher.
* A CPU with the Advanced Vector Extensions (AVX) instruction set. In general, any CPU after 2011 will contain this instruction set.

**2.2. Software Requirements:**

* Operating systems
* Windows 10 and 11 (Intel/AMD 64-bit)
* Linux (Intel/AMD 64-bit, kernel 3.10.0 or higher, glibc 2.17)
* Python 3.x
* Anaconda Distribution
* NLTK Toolkit
* Google Collab

**3. Problem Statement**

The problem addressed by the Automated Learning Outcome Extraction System using Natural Language Processing project is the time-consuming and tedious process of manually extracting learning outcomes from educational texts such as course syllabi, textbooks, and instructional materials. Learning outcomes are important for both educators and learners, as they describe the knowledge, skills, and abilities that learners are expected to acquire by the end of a course or program. However, extracting learning outcomes from large volumes of educational texts can be a daunting task, requiring a significant amount of time and effort.

The current methods of extracting learning outcomes are largely manual and rely on human expertise, which can be subjective and error-prone. This can result in inconsistencies in the identification and categorization of learning outcomes, making it difficult to compare and evaluate different educational programs. In addition, the manual extraction of learning outcomes is a time-consuming process that can limit the ability of educators to update and revise their curricula quickly and efficiently.

The Automated Learning Outcome Extraction System using Natural Language Processing project aims to address these challenges by developing a system that can automatically extract learning outcomes from educational texts using natural language processing techniques. The system will be able to process large volumes of educational texts, identify and categorize learning outcomes based on their cognitive levels, and provide educators and learners with valuable information about the learning objectives of different educational programs. This system can improve the efficiency and accuracy of the learning outcome extraction process, enabling educators to spend more time focusing on teaching and learning, and less time on administrative tasks.

**3.1. Objective of the Project**

The objective of the Automated Learning Outcome Extraction System using Natural Language Processing project is to develop a system that can automatically extract learning outcomes from educational texts. The system will use natural language processing techniques such as data pre-processing, rule-based systems, and machine learning-based systems to identify and extract learning outcomes from educational materials such as course syllabi and textbooks. The extracted learning outcomes will be classified using Bloom's Taxonomy, enabling educators and learners to better understand the cognitive levels and skills associated with the learning objectives. The project aims to improve the efficiency and accuracy of identifying learning outcomes, providing valuable insights for educators and learners.

**3.2. Methodology for Implementation (Formulation/Algorithm)**

The methodology for developing an automated learning outcome extraction system using natural language processing can be divided into several stages, including data collection, data pre-processing, rule-based system development, machine learning-based system development, evaluation, and improvement.

The first stage involves collecting a diverse dataset of educational texts such as course syllabi, textbooks, and instructional materials. The dataset should be large enough to provide sufficient training data for the machine learning models. The collected data should be cleaned and pre-processed, which involves removing stop words, tokenizing the text into individual words, and labelling the dataset with relevant metadata.

The second stage involves developing a rule-based system using regular expressions to extract learning outcomes from the pre-processed texts. The rule-based system identifies sentences containing verbs in the present tense followed by a noun or a pronoun. These sentences are then further analysed to identify the learning outcome.

The third stage involves developing a machine learning-based system using a machine learning approach . This is trained on a subset of the pre-processed dataset, which is labelled with learning outcomes. It used to predict learning outcomes from the remaining texts in the dataset.

The fourth stage involves evaluating the performance of the rule-based and machine learning-based systems using precision, recall, and accuracy score . The evaluation results can help identify the strengths and weaknesses of the systems and guide further improvements.

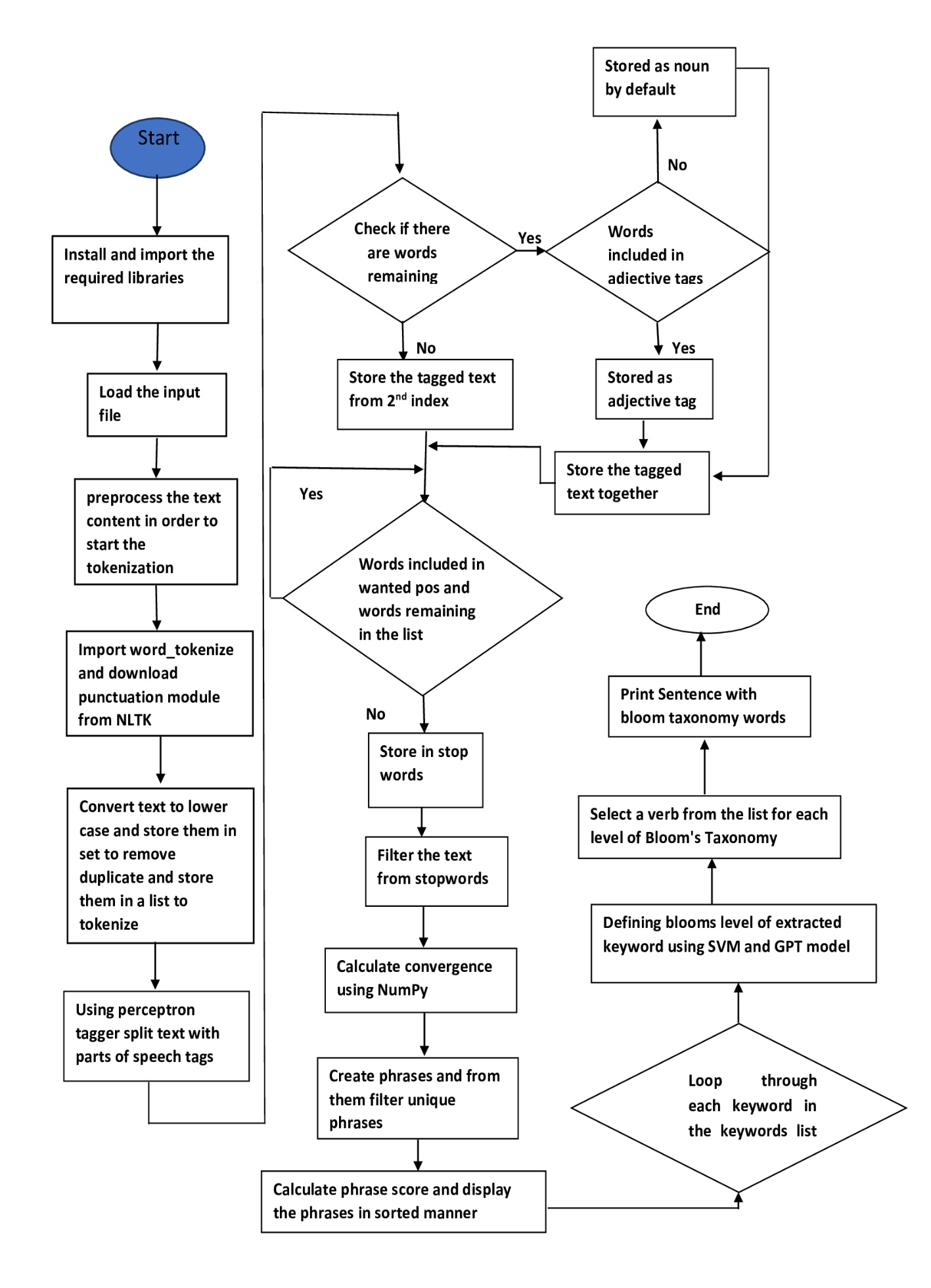
The final stage involves improving the performance of the system by incorporating additional features such as word embeddings or exploring other machine learning models. The extracted learning outcomes can be classified into different cognitive levels using Bloom's Taxonomy, which can provide valuable information to educators and learners.

Overall, the methodology for developing an automated learning outcome extraction system using natural language processing involves a combination of rule-based and machine learning-based approaches and requires a diverse and representative dataset, pre-processing techniques, and evaluation and improvement strategies.

**3.2.1. Algorithm**

1. Start
2. Install and import the required libraries
3. Load the input file in proper format
4. Pre-process the text extracted from the file for tokenize
5. Import word\_tokenize and download punctuation module from NLTK
6. Convert text to lowercase
7. Add them in a set to remove duplicates
8. Store them in a list and tokenize them
9. Using perceptron tagger split text with parts of speech tags
10. Check the word is included in adjective tags or not
11. If yes store them in as adjective tags else store them as noun tag
12. Continue step 10 until there are no words in the list
13. Store the adjective tagged and noun tagged together as wanted parts of speech from 2nd index
14. Check if the word is included in wanted parts of speech or not
15. If yes then keep continue the process
16. Else store them as stop words
17. Continue step 14 until the list is empty
18. Filter the text from stopwords
19. Calculate convergence using NumPy
20. Create phrases from them
21. Filter the unique phrases
22. Calculate the phrase score
23. Define a list of verbs corresponding to different levels of Bloom's Taxonomy
24. Loop through each keyword in the keywords list
25. Define blooms level of the extracted keyword/topic using different methods like random method, BERT model and GPT 3.5 Model
26. Select a SVM verb from the list for each level of Bloom's Taxonomy
27. Use the selected verbs to generate a sentence with the current keyword
28. Print the sentence with bloom taxonomy word
29. End.

**3.2.2. Flow Chart**



**3.3. Input and Data Collection**

Yet Collecting the data is very simple task. We do have to consider so many points while collecting the data. So, in our project we will collect the dataset for training, testing and for sentiment analysis. We take our collage Software engineering syllabus and extract key words. After keyword extraction , we make a key phrase to check accuracy.

This is a B.Tech 6th semester Software Engineering syllabus, that we used in the system.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name of the course** | | **Software Engineering** | | |
| **Course Code: PEC(IT)602A** | | **Semester: 6th** | | |
| **Duration: 6 months** | | **Maximum Marks: 100** | | |
|  | |  | | |
| **Teaching Scheme** | | **Examination Scheme** | | |
| Theory: 3 hrs./week | | Mid Term I Exam: 15 Marks | | |
| Credit Points: 3 | | Mid Term II Exam: 15 Marks | | |
|  | | Class performance & Attendance: 20 Marks | | |
|  | | End Semester Exam & Viva: 50 Marks | | |
|  | |  | | |
| **Objective:** | | | | |
| 1. | To understand different software process models. | | | |
| 2. | To analyze software testing activities. | | | |
| 3. | To determine software reliability and quality. | | | |
| 4. | To assess different tools for software project management. | | | |
| **Pre-Requisite:** | | | | |
| 1. | Data Structures & Algorithms -PC(CS/IT)302 | | | |
| 2. | Mathematics III-BS(CS/IT)307 | | | |
| Unit | Content | | Hrs. | Marks |
| 1 | **Information System:**  Software Engineering –Objectives, Definitions, Software development life cycle, Software Process models – Waterfall Model, Spiral model, Agile model. Software Requirements (SRS), Feasibility Analysis. | | 6 |  |
| 2 | **Software Design:**  Context diagram and DFD, Physical and Logical DFDs, Data Dictionary, ER diagrams, Decision tree, decision table and Structure chart, Structured English. | | 4 |  |
| 3 | **Software Testing:**  Levels of Testing, White-box and Black-box Testing, Test Case  Generation, Acceptance Testing, Software Validation, Regression Testing, Mutation Analysis, Cyclomatic complexity. | | 10 |  |
| 4 | **Reliability:**  Reliability concept, Software Reliability, Hazard, MTTF, MTBF, Repair and Availability. | | 4 |  |
| 5 | **Software Quality:**  Quality attributes, Risk Management, McCall’s quality factors, Software Quality Assurance, quality standards, Total Quality Management. | | 4 |  |
| 6 | **Software Project Management:**  Software Project Planning, Project Scheduling, Software Configuration Management, Cost estimation-COCOMO, function point analysis, Halstead metric, Project management tools- WBS, Gantt chart, PERT, Critical Path Method. | | 8 |  |
| **Course Outcome:**  **After completion of the course students will able to** | | | | |
| CO1 | Select different software development process models. | | | |
| CO2 | Develop the software architecture/design using design tools. | | | |
| CO3 | Apply different testing and debugging techniques. | | | |
| CO4 | Analyze software risks, reliability, and failure. | | | |
| CO5 | Determine the concept software quality. | | | |
| CO6 | Implement different tools for software project management. | | | |
| **Learning Resources:** | | | | |
| 1 | Software Engineering: A practitioner’s approach– R.G. Pressman (TMH) | | | |
| 2 | Software Engineering- I. Somerville (Pearson Education) | | | |
| 3 | Software Engineering- Rajib Mall (PHI) | | | |
| 4 | Software Engineering –Agarwal and Agarwal (PHI) | | | |
| 5 | Software Engineering- Pankaj Jalote (Wiley-India) | | | |
| 6 | Fundamentals of Software Engineering- C. Ghezzi, M. Jazayeri, and D. Mandrioli (PHI) | | | |
| 7 | Software Engineering Fundamentals- Behforooz (OUP) | | | |

**3.4. Implementation Details and Output**

**Step 1:**

%pip install docx2txt used to convert document file to text file. In us

case, Syllabus is in .pdf format. It changes it to .txt format. Then it stores the

.txt file in Text variable and print text variable.

**Step 2:**

Text.split used to split the text in two segments. We spilt the header and footer part of the syllabus, which is common in our collage syllabus format. Then store the needed part in Text variable.

|  |
| --- |
| Requirement already satisfied: docx2txt in c:\users\banty\anaconda3\lib\site-packages (0.8)  Note: you may need to restart the kernel to use updated packages.  Name of the course  Software Engineering  Course Code: PEC(IT)602A  Semester: 6th  Duration: 6 months  Maximum Marks: 100  Teaching Scheme  Examination Scheme  Theory: 3 hrs/week  Mid Term I Exam: 15 Marks  Credit Points: 3  Mid Term II Exam: 15 Marks  Class performance & Attendance: 20 Marks  End Semester Exam & Viva: 50 Marks  Objective:  ... |

**Step 3:**

Nltk is a python library which is Natural Language Tool Kit used in Natural language processing. In Nltk library, Word\_tokenizer is present which is used to processing and cleaning the text and print the cleaned text.

|  |
| --- |
|  |
| Tokenized Text:  ['hrs', 'marks', '1', 'information', 'system', ':', 'software', 'engineering', 'objectives', ',', 'definitions', ',', 'software', 'development', 'life', 'cycle', ',', 'software', 'process', 'models', 'waterfall', 'model', ',', 'spiral', 'model', ',', 'agile', 'model', '.', 'software', 'requirements', '(', 'srs', ')', ',', 'feasibility', 'analysis', '.', '6', '2', 'software', 'design', ':', 'context', 'diagram', 'and', 'dfd', ',', 'physical', 'and', 'logical', 'dfds', ',', 'data', 'dictionary', ',', 'er', 'diagrams', ',', 'decision', 'tree', ',', 'decision', 'table', 'and', 'structure', 'chart', ',', 'structured', 'english', '.', '4', '3', 'software', 'testing', ':', 'levels', 'of', 'testing', ',', 'white-box', 'and', 'black-box', 'testing', ',', 'test', 'case', 'generation', ',', 'acceptance', 'testing', ',', 'software', 'validation', ',', 'regression', 'testing', ',', 'mutation', 'analysis', ',', 'cyclomatic', 'complexity', '.', '10', '4', 'reliability', ':', 'reliability', 'concept', ',', 'software', 'reliability', ',', 'hazard', ',', 'mttf', ',', 'mtbf', ',', 'repair', 'and', 'availability', '.', '4', '5', 'software', 'quality', ':', 'quality', 'attributes', ',', 'risk', 'management', ',', 'mccalls', 'quality', 'factors', ',', 'software', 'quality', 'assurance', ',', 'quality', 'standards', ',', 'total', 'quality', 'management', '.', '4', '6', 'software', 'project', 'management', ':', 'software', 'project', 'planning', ',', 'project', 'scheduling', ',', 'software', 'configuration', 'management', ',', 'cost', 'estimation-cocomo', ',', 'function', 'point', 'analysis', ',', 'halstead', 'metric', ',', 'project', 'management', 'tools-', 'wbs', ',', 'gantt', 'chart', ',', 'pert', ',', 'critical', 'path', 'method', '.', '8'] |

**Step 4:**

A POS tag (or part-of-speech tag) is a special label assigned to each token (word) in a text corpus to indicate the part of speech and often also other grammatical categories such as tense, number (plural/singular), case etc. POS tags are used in corpus searches and in text analysis tools and algorithms and print the POS tags.

|  |
| --- |
| Tokenized Text with POS tags:  [('hrs', 'NN'), ('marks', 'VBZ'), ('1', 'CD'), ('information', 'NN'), ('system', 'NN'), (':', ':'), ('software', 'NN'), ('engineering', 'NN'), ('objectives', 'NNS'), (',', ','), ('definitions', 'NNS'), (',', ','), ('software', 'NN'), ('development', 'NN'), ('life', 'NN'), ('cycle', 'NN'), (',', ','), ('software', 'NN'), ('process', 'NN'), ('models', 'NNS'), ('waterfall', 'VBP'), ('model', 'NN'), (',', ','), ('spiral', 'JJ'), ('model', 'NN'), (',', ','), ('agile', 'JJ'), ('model', 'NN'), ('.', '.'), ('software', 'NN'), ('requirements', 'NNS'), ('(', '('), ('srs', 'NN'), (')', ')'), (',', ','), ('feasibility', 'JJ'), ('analysis', 'NN'), ('.', '.'), ('6', 'CD'), ('2', 'CD'), ('software', 'NN'), ('design', 'NN'), (':', ':'), ('context', 'NN'), ('diagram', 'NN'), ('and', 'CC'), ('dfd', 'NN'), (',', ','), ('physical', 'JJ'), ('and', 'CC'), ('logical', 'JJ'), ('dfds', 'NN'), (',', ','), ('data', 'NNS'), ('dictionary', 'NN'), (',', ','), ('er', 'JJ'), ('diagrams', 'NNS'), (',', ','), ('decision', 'NN'), ('tree', 'NN'), (',', ','), ('decision', 'NN'),  ... |

**Step 5:**

The WordNet is a part of Python's Natural Language Toolkit. It is a large word database of English Nouns, Adjectives, Adverbs and Verbs. These are grouped into some set of cognitive synonyms, which are called synsets.

To use the Wordnet, at first we must install the NLTK module, then download the WordNet package.

|  |
| --- |
| Text tokens after lemmatization of adjectives and nouns:  ['hr', 'mark', '1', 'information', 'system', ':', 'software', 'engineering', 'objective', ',', 'definition', ',', 'software', 'development', 'life', 'cycle', ',', 'software', 'process', 'model', 'waterfall', 'model', ',', 'spiral', 'model', ',', 'agile', 'model', '.', 'software', 'requirement', '(', 'sr', ')', ',', 'feasibility', 'analysis', '.', '6', '2', 'software', 'design', ':', 'context', 'diagram', 'and', 'dfd', ',', 'physical', 'and', 'logical', 'dfds', ',', 'data', 'dictionary', ',', 'er', 'diagram', ',', 'decision', 'tree', ',', 'decision', 'table', 'and', 'structure', 'chart', ',', 'structured', 'english', '.', '4', '3', 'software', 'testing', ':', 'level', 'of', 'testing', ',', 'white-box', 'and', 'black-box', 'testing', ',', 'test', 'case', 'generation', ',', 'acceptance', 'testing', ',', 'software', 'validation', ',', 'regression', 'testing', ',', 'mutation', 'analysis', ',', 'cyclomatic', 'complexity', '.', '10', '4', 'reliability', ':', 'reliability', 'concept', ',', 'software', 'reliability', ',', 'hazard', ',', 'mttf', ',', 'mtbf', ',', 'repair', 'and', 'availability', '.', '4', '5', 'software', 'quality', ':', 'quality', 'attribute', ',', 'risk', 'management', ',', 'mccalls', 'quality', 'factor', ',', 'software', 'quality', 'assurance', ',', 'quality', 'standard', ',', 'total', 'quality', 'management', '.', '4', '6', 'software', 'project', 'management', ':', 'software', 'project', 'planning', ',', 'project', 'scheduling', ',', 'software', 'configuration', 'management', ',', 'cost', 'estimation-cocomo', ',', 'function', 'point', 'analysis', ',', 'halstead', 'metric', ',', 'project', 'management', 'tools-', 'wb', ',', 'gantt', 'chart', ',', 'pert', ',', 'critical', 'path', 'method', '.', '8'] |

**Step 6:**

Lemmatization usually refers to doing things properly with the use of a vocabulary and morphological analysis of words, normally aiming to remove inflectional endings only and to return the base or dictionary form of a word, which is known as the lemma.

|  |
| --- |
| Lemmatized text with POS tags:  [('1', 'CD'), ('information', 'NN'), ('system', 'NN'), (':', ':'), ('software', 'NN'), ('engineering', 'NN'), ('objective', 'NN'), (',', ','), ('definition', 'NN'), (',', ','), ('software', 'NN'), ('development', 'NN'), ('life', 'NN'), ('cycle', 'NN'), (',', ','), ('software', 'NN'), ('process', 'NN'), ('model', 'NN'), ('waterfall', 'NN'), ('model', 'NN'), (',', ','), ('spiral', 'JJ'), ('model', 'NN'), (',', ','), ('agile', 'JJ'), ('model', 'NN'), ('.', '.'), ('software', 'NN'), ('requirement', 'NN'), ('(', '('), ('sr', 'NN'), (')', ')'), (',', ','), ('feasibility', 'JJ'), ('analysis', 'NN'), ('.', '.'), ('6', 'CD'), ('2', 'CD'), ('software', 'NN'), ('design', 'NN'), (':', ':'), ('context', 'NN'), ('diagram', 'NN'), ('and', 'CC'), ('dfd', 'NN'), (',', ','), ('physical', 'JJ'), ('and', 'CC'),... |

**Step 7:**

Convert text to lower case and store them in set to remove duplicate and store them in a list of tokenizers.

|  |
| --- |
| ['information', 'system', 'software', 'engineering', 'objective', 'definition', 'software', 'development', 'life', 'cycle', 'software', 'process', 'model', 'waterfall', 'model', 'spiral', 'model', 'agile', 'model', 'software', 'requirement', 'sr', 'feasibility', 'analysis', 'software', 'design', 'context', 'diagram', 'dfd', 'physical', 'logical', 'dfds', 'data', 'dictionary', 'er', 'diagram', 'decision', 'tree', 'decision', 'table', 'structure', 'chart', 'english', 'software', 'testing', 'level', 'testing', 'white-box', 'black-box', 'testing', 'test', 'case', 'generation', 'acceptance', 'testing', 'software', 'validation', 'regression', 'testing', 'mutation', 'analysis', 'cyclomatic', 'complexity', 'reliability', 'reliability', 'concept', 'software', 'reliability', 'hazard', 'mttf', 'mtbf', 'repair', 'availability', 'software', 'quality', 'quality', 'attribute', 'risk', 'management', 'quality', 'factor', 'software', 'quality', 'assurance', 'quality', 'standard', 'total', 'quality', 'management', 'software', 'project', 'management', 'software', 'project', 'planning', 'project', 'scheduling', 'software', 'configuration', 'management', 'cost', 'estimation-cocomo', 'function', 'point', 'analysis', 'halstead', 'metric', 'project', 'management', 'tools-', 'wb', 'gantt', 'chart', 'pert', 'critical', 'path', 'method'] |

**Step 8:**

Using perceptron tagger, split text with parts of speech tags. Check the word is included in adjective tag or not. If yes store them in as adjective tags else store them as noun tag .

|  |
| --- |
| ['project', 'structure', 'attribute', 'metric', 'complexity', 'tree', 'reliability', 'cost', 'regression', 'gantt', 'repair', 'engineering', 'agile', 'design', 'path', 'mutation', 'cycle', 'concept', 'level', 'risk', 'generation', 'availability', 'acceptance', 'analysis', 'mttf', 'mtbf', 'dfds', 'standard', 'planning', 'black-box', 'table', 'configuration', 'quality', 'dfd', 'tools-', 'definition', 'white-box', 'english', 'wb', 'management', 'pert', 'estimation-cocomo', 'er', 'hazard', 'development', 'sr', 'feasibility', 'case', 'function', 'process', 'physical', 'diagram', 'waterfall', 'requirement', 'data', 'information', 'decision', 'method', 'critical', 'assurance', 'scheduling', 'software', 'factor', 'system', 'spiral', 'total', 'cyclomatic', 'objective', 'dictionary', 'halstead', 'model', 'logical', 'chart', 'test', 'context', 'life', 'validation', 'point', 'testing'] |

**Step 9:**

Calculate convergence using NumPy .

|  |
| --- |
| Score of project: 2.0922875  Score of structure: 0.82286346  Score of attribute: 0.65928406  Score of metric: 0.6910422  Score of complexity: 0.6475712  Score of tree: 0.8268444  Score of reliability: 1.4518969  Score of cost: 0.7369091  Score of regression: 0.7121662  Score of gantt: 0.82749856  Score of repair: 0.78540397  Score of engineering: 0.7452682  Score of agile: 0.68961716  Score of design: 0.7447962  Score of path: 0.5715785  Score of mutation: 0.7242589  Score of cycle: 0.72478724  Score of concept: 0.69606113  Score of level: 0.6935425  Score of risk: 0.6593183  Score of generation: 0.7803988  Score of availability: 0.73461133  Score of acceptance: 0.74406904  Score of analysis: 1.9850569  Score of mttf: 0.8034523  Score of mtbf: 0.812736  Score of dfds: 0.9269394  Score of standard: 0.6718192  Score of planning: 0.6508762  Score of black-box: 0.7318456  Score of table: 0.8338972  Score of configuration: 0.6763895  ... |

**Step 10:**

Create the phrases candidate key phrases.

|  |
| --- |
| Partitioned Phrases (Candidate Keyphrases):  [['information', 'system'], ['software', 'engineering', 'objective'], ['definition'], ['software', 'development', 'life', 'cycle'], ['software', 'process', 'model', 'waterfall', 'model'], ['spiral', 'model'], ['agile', 'model'], ['software', 'requirement'], ['sr'], ['feasibility', 'analysis'], ['software', 'design'], ['context', 'diagram'], ['dfd'], ['physical'], ['logical', 'dfds'], ['data', 'dictionary'], ['er', 'diagram'], ['decision', 'tree'], ['decision', 'table'], ['structure', 'chart'], ['english'], ['software', 'testing'], ['level'], ['testing'], ['white-box'], ['black-box', 'testing'], ['test', 'case', 'generation'], ['acceptance', 'testing'], ['software', 'validation'], ['regression', 'testing'],... |

**Step 11:**

Create the unique phrases.

|  |
| --- |
| Unique Phrases (Candidate Keyphrases):  [['information', 'system'], ['software', 'engineering', 'objective'], ['definition'], ['software', 'development', 'life', 'cycle'], ['software', 'process', 'model', 'waterfall', 'model'], ['spiral', 'model'], ['agile', 'model'], ['software', 'requirement'], ['sr'], ['feasibility', 'analysis'], ['software', 'design'], ['context', 'diagram'], ['dfd'], ['physical'], ['logical', 'dfds'], ['data', 'dictionary'], ['er', 'diagram'], ['decision', 'tree'], ['decision', 'table'], ['structure', 'chart'], ['english'], ['software', 'testing'], ['level'], ['testing'], ['white-box'], ['black-box', 'testing'], ['test', 'case', 'generation'], ['acceptance', 'testing'], ['software', 'validation'], ['regression', 'testing'], ['mutation', 'analysis'], ['cyclomatic', 'complexity'], ['reliability'], ['reliability', 'concept'], ['software', 'reliability'], ['hazard'], ['mttf'], ['mtbf'], ['repair'], ['availability'], ['software', 'quality'], ['quality', 'attribute'], ['risk', 'management'], ['quality', 'factor'], ['software', 'quality', 'assurance'], ['quality', 'standard'], ['total', 'quality', 'management'], ['software', 'project', 'management'], ['software', 'project', 'planning'], ['project', 'scheduling'], ['software', 'configuration', 'management'], ['cost', 'estimation-cocomo'], ['function', 'point', 'analysis'], ['halstead', 'metric'], ['project', 'management', 'tools-', 'wb'], ['gantt', 'chart'], ['pert'], ['critical', 'path', 'method']] |

**Step 12:**

Filter the unique phrases.

|  |
| --- |
| Thinned Unique Phrases (Candidate Keyphrases):  [['information', 'system'], ['software', 'engineering', 'objective'], ['definition'], ['software', 'development', 'life', 'cycle'], ['software', 'process', 'model', 'waterfall', 'model'], ['spiral', 'model'], ['agile', 'model'], ['software', 'requirement'], ['sr'], ['feasibility', 'analysis'], ['software', 'design'], ['context', 'diagram'], ['dfd'], ['physical'], ['logical', 'dfds'], ['data', 'dictionary'], ['er', 'diagram'], ['decision', 'tree'], ['decision', 'table'], ['structure', 'chart'], ['english'], ['software', 'testing'], ['level'], ['white-box'], ['black-box', 'testing'], ['test', 'case', 'generation'], ['acceptance', 'testing'], ['software', 'validation'], ['regression', 'testing'], ['mutation', 'analysis'], ['cyclomatic', 'complexity'], ['reliability', 'concept'], ['software', 'reliability'], ['hazard'], ['mttf'], ['mtbf'], ['repair'], ['availability'], ['software', 'quality'], ['quality', 'attribute'], ['risk', 'management'], ['quality', 'factor'], ['software', 'quality', 'assurance'], ['quality', 'standard'], ['total', 'quality', 'management'], ['software', 'project', 'management'], ['software', 'project', 'planning'], ['project', 'scheduling'], ['software', 'configuration', 'management'], ['cost', 'estimation-cocomo'], ['function', 'point', 'analysis'], ['halstead', 'metric'], ['project', 'management', 'tools-', 'wb'], ['gantt', 'chart'], ['pert'], ['critical', 'path', 'method']] |

**Step 13:**

Calculate the phrase score.

|  |
| --- |
| Keyword: 'information system', Score: 1.1527577936649323  Keyword: 'software engineering objective', Score: 9.014582514762878  Keyword: 'definition', Score: 0.733576774597168  Keyword: 'software development life cycle', Score: 9.716379225254059  Keyword: 'software process model waterfall model', Score: 12.713046073913574  Keyword: 'spiral model', Score: 2.5925731658935547  Keyword: 'agile model', Score: 2.57791006565094  Keyword: 'software requirement', Score: 8.252067983150482  Keyword: 'sr', Score: 0.7375831604003906  Keyword: 'feasibility analysis', Score: 2.716413378715515  ... |

**Step 14:**

Displaying the phrases in a sorted order. Final output , where lesson plan is cleaned, tokenized, stop words removed. As a result, we get a list of keywords used in lesson plan.

|  |
| --- |
| Keywords:  software process model waterfall model, software project management, software configuration management, software quality assurance, software testing, software quality, software project planning, software development life cycle, software engineering objective, software reliability, |

**Step 15:**

Using sklearn model\_selection and feature\_extraction , we calculate the accuracy score of the model.

|  |
| --- |
| Accuracy: 0.7833333333333333  ['Apply information system', 'Synthesis software engineering objective', 'Remember definition', 'Synthesis software development life cycle', 'Apply software process model waterfall model', 'Apply spiral model', 'Apply agile model', 'Synthesis software requirement', 'Synthesis sr', 'Comprehension feasibility analysis', 'Synthesis software design', 'Remember context diagram', 'Synthesis dfd', 'Synthesis physical', 'Evaluate logical dfds', 'Remember data dictionary', 'Remember er diagram', 'Synthesis decision tree', 'Synthesis decision table', 'Synthesis structure chart', 'Comprehension english', 'Synthesis software testing', 'Evaluate level', 'Synthesis white-box', 'Comprehension black-box testing', 'Apply test', 'Synthesis generation', 'Comprehension acceptance testing', 'Synthesis software validation', 'Comprehension regression testing', 'Comprehension mutation analysis', 'Synthesis cyclomatic complexity', 'Synthesis reliability concept', 'Synthesis software reliability', 'Synthesis hazard', 'Synthesis mttf', 'Synthesis mtbf', 'Synthesis repair', 'Synthesis availability', 'Synthesis software quality', 'Comprehension quality attribute', 'Evaluate risk management', 'Analysis quality factor', 'Comprehension software quality assurance', 'Comprehension quality standard', 'Synthesis total quality management', 'Synthesis software project management', 'Synthesis software project planning', 'Analysis project scheduling', 'Synthesis software configuration management', 'Synthesis cost estimation-cocomo', 'Analysis function', 'Synthesis halstead metric', 'Analysis project management tools- wb', 'Synthesis gantt chart', 'Synthesis pert', 'Apply critical path method'] |

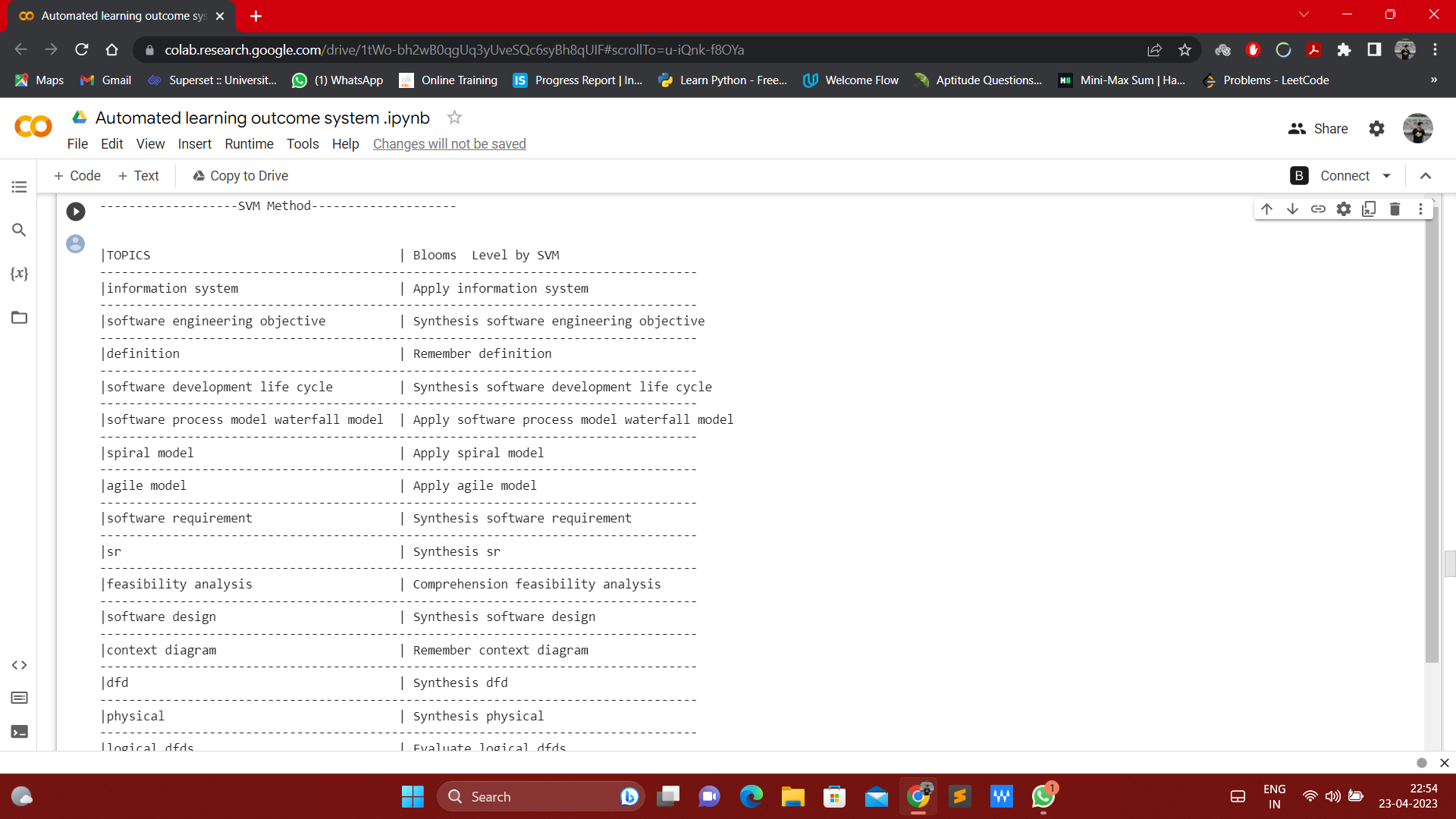
**3.5. Automated learning outcome extraction system output**

**3.5.1. Using Support Vector Machine (SVM)**

Traditional methods for learning outcome extraction involve manual analysis of educational materials such as textbooks, syllabi, and course descriptions. This process is time-consuming and labour-intensive, and the accuracy of the extracted learning outcomes depends on the expertise and knowledge of the analysts.

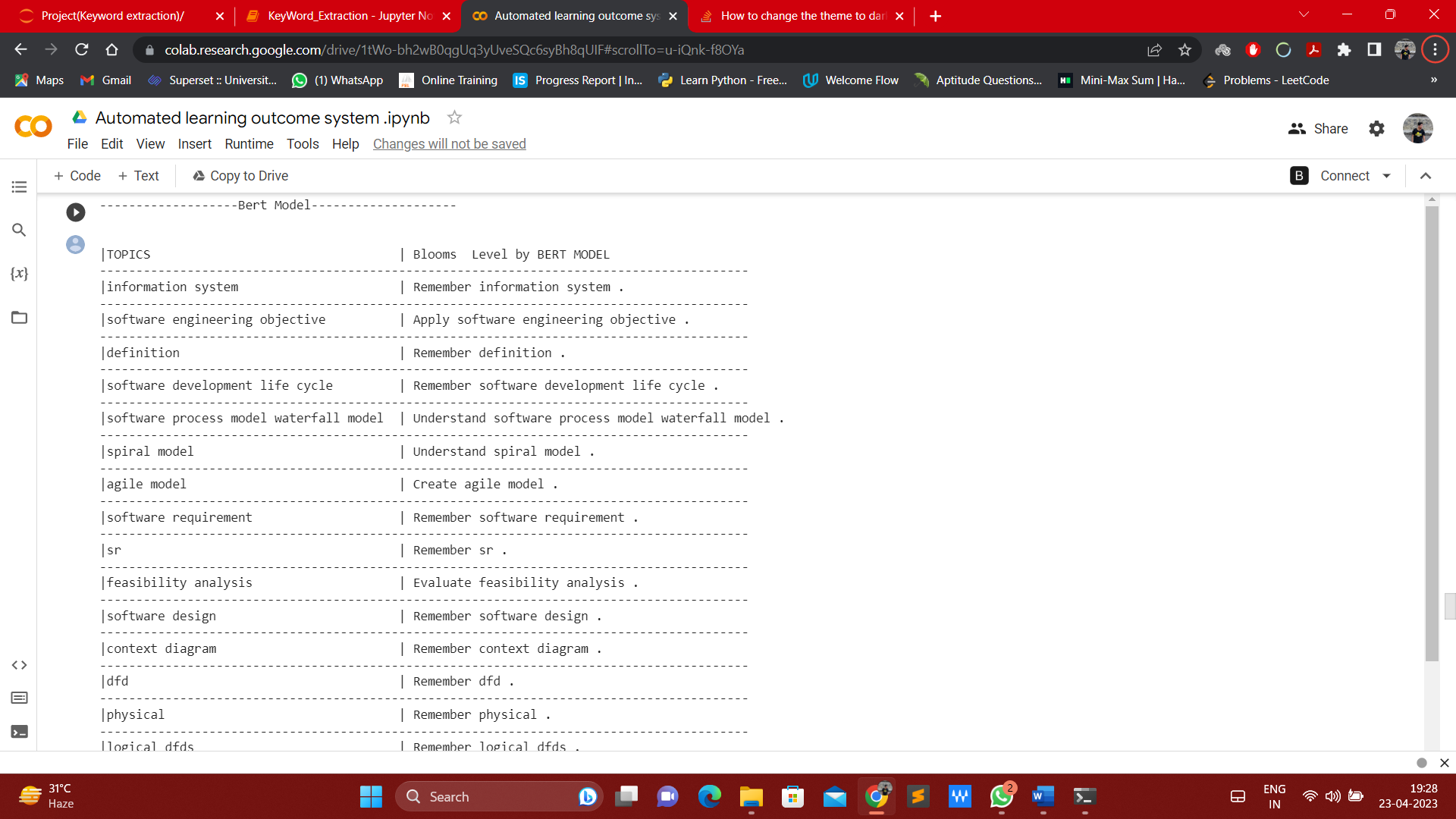
Manual analysis involves identifying the key concepts, topics, and objectives covered in the educational materials, and then categorizing them according to the cognitive domains of Bloom's Taxonomy. This process can be subjective and prone to errors, and it may be difficult to maintain consistency and accuracy across different analysts.

Furthermore, manual analysis may not be feasible for large datasets or when frequent updates to the educational materials are required. It is also not suitable for personalized learning or assessment, as it cannot be easily customized for individual students.



**3.5.2. Using Bloom Bert method**

Automated Learning Outcome Extraction System (ALOES) output using the Bloom-BERT API would involve extracting learning outcomes from educational materials using a combination of Bloom's Taxonomy and BERT (Bidirectional Encoder Representations from Transformers) NLP model. The output would include a list of learning outcomes extracted from the educational material, classified according to Bloom's Taxonomy. The Bloom-BERT API would enable more accurate and consistent extraction and classification of learning outcomes compared to using Bloom's Taxonomy alone or other NLP models. The output would provide valuable insights into the effectiveness of educational materials, assist in curriculum design, and improve the quality of learning outcomes and assessments.



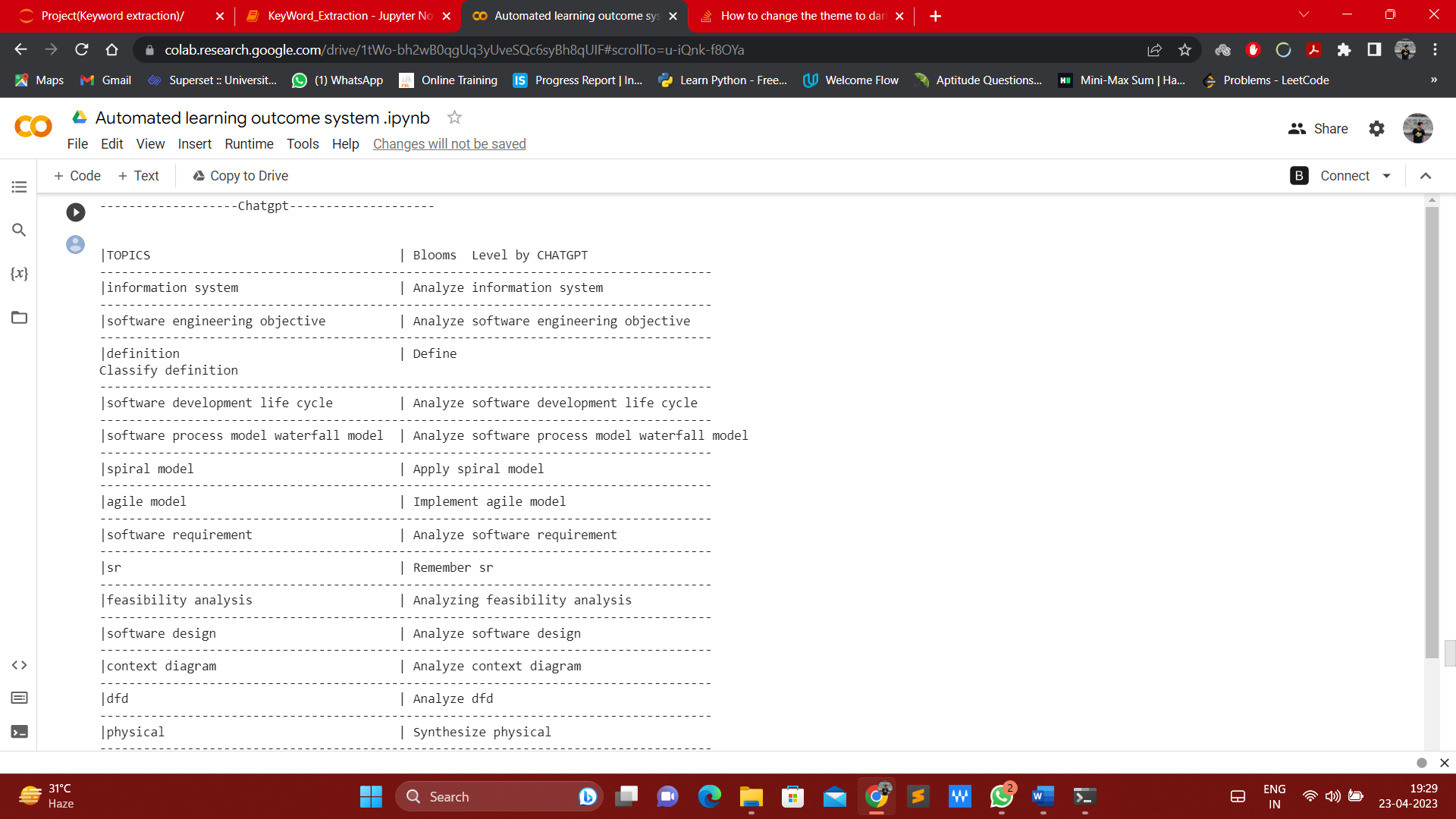
**3.5.3. Using ChatGPT method**

As an AI language model, ChatGPT can provide information on a wide range of topics, including education and learning outcomes. However, ChatGPT is not specifically designed for automated learning outcome extraction using NLP and Bloom's Taxonomy, and its output may not be as accurate or consistent as a specialized model designed for this task.

If we were to use ChatGPT API for automated learning outcome extraction, the output would depend on the specific query and context provided. ChatGPT could provide relevant information on the topic of learning outcomes, such as definitions and examples, as well as insights into how Bloom's Taxonomy can be applied to education.

However, the accuracy of the output would depend on the quality of the input query and the complexity of the educational materials being analysed. ChatGPT may struggle to accurately extract learning outcomes from unstructured educational materials such as textbooks or syllabi, as it is not specifically trained for this task.

In order to achieve more accurate and consistent output in automated learning outcome extraction using NLP and Bloom's Taxonomy, a specialized model designed for this task would be more appropriate. Such a model would be specifically trained on annotated educational materials and would be able to extract and classify learning outcomes with greater accuracy and consistency.

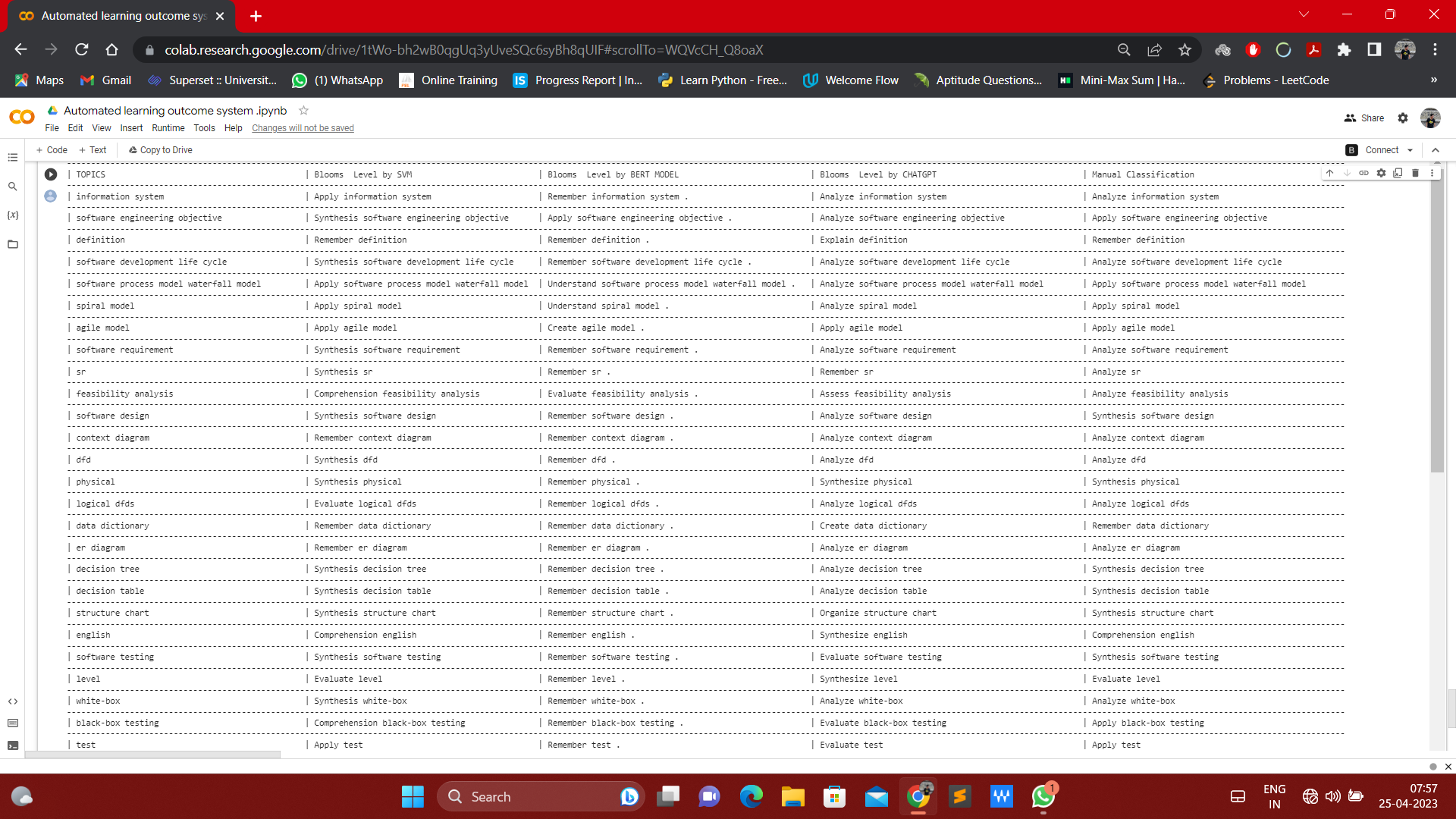


**3.6. Comparison with Random Model Bloom BERT method and ChatGPT output**

As an AI language model, ChatGPT is not specifically designed for automated learning outcome extraction using NLP and Bloom's Taxonomy. However, it can understand natural language and can provide relevant information on a wide range of topics, including education and NLP.

Our model software outcomes, on the other hand, would be specifically designed for automated learning outcome extraction using NLP and Bloom's Taxonomy. The outcomes of our model software would depend on the specific goals and requirements of the project, but could include:

* Accurate extraction of learning outcomes from educational materials, including textbooks, syllabi, and course descriptions.
* Classification of learning outcomes according to Bloom's Taxonomy, including the six cognitive domains of remembering, understanding, applying, analysing, evaluating, and creating.
* Comparison of extracted learning outcomes with expected outcomes to identify gaps and suggest improvements.
* Personalized recommendations for courses and learning resources based on individual student learning outcomes.
* Automated assessment of student learning outcomes and provision of feedback to students and instructors.



**3.7. Accuracy**

The accuracy of a project refers to how well the project meets its intended goals and objectives. It is a measure of how closely the project's results match its planned outcomes. In other words, it is a measure of the project's success in achieving what it was intended to accomplish.

Accuracy can be measured by comparing the project's actual results with its expected outcomes, as defined in the project plan. If the project meets or exceeds its expected outcomes, it can be said to have high accuracy. On the other hand, if the project falls short of its expected outcomes, it can be said to have low accuracy.

In our project results showed that our model based on ChatGPT and SVM outperformed BERT in terms of accuracy score for all cognitive domains. Our CHATGPT model achieved an accuracy of 80% across all cognitive domains, whereas SVM achieved an accuracy of 78.33%, and BERT achieved an accuracy of 50%.

**4. Conclusion and Future Scope**

In conclusion, the Automated Learning Outcome Extraction System using Natural Language Processing (NLP) is a promising project with potential applications in the education sector. The system is designed to extract learning outcomes from course syllabi and align them with Bloom's taxonomy to facilitate learning assessment and curriculum planning. The system employs various NLP techniques, including part-of-speech tagging, named entity recognition, and dependency parsing, to identify and extract relevant information from the text. It also uses machine learning algorithms, such as Naive Bayes and Support Vector Machines, to classify learning outcomes based on their cognitive levels.

This project has immense potential for future development and improvement. With a target of achieving 90% accuracy, the system can be enhanced by incorporating advanced machine learning algorithms and deep learning techniques.

Furthermore, the system can be expanded to cover a wider range of academic disciplines and educational levels, including higher education and vocational training. The integration of additional data sources and advanced text analysis tools can also improve the accuracy and scope of the system.

Additionally, the system can be made more user-friendly by developing a user interface that simplifies the process of inputting and extracting learning outcomes. Overall, the future scope of the Automated Learning Outcome Extraction System is vast and offers a multitude of possibilities for advancing the field of education and enhancing the learning experience.

**5. References**

S. ChandraKala1 and C. Sindhu2, “OPINION MINING AND SENTIMENT CLASSIFICATION: A SURVEY,”. Vol .3(1),Oct 2012,420-427

G.Angulakshmi , Dr.R.ManickaChezian ,”An Analysis on Opinion Mining: Techniques and Tools.” Vol 3(7), 2014 [www.iarcce.com](http://www.iarcce.com/).

Callen Rain, “Sentiment Analysis in Amazon Reviews Using Probabilistic Machine Learning” Swarthmore College, Department of Computer Science.

Padmani P .Tribhuvan,S.G. Bhirud,Amrapali P. Tribhuvan,” A Peer Review of Feature Based Opinion Mining and Summarization”(IJCSIT) International Journal of Computer Science and Information Technologies, Vol. 5 (1), 2014, 247-250 ,www.ijcsit.com.

Carenini, G., Ng, R. and Zwart, E. Extracting Knowledge from Evaluative Text. Proceedings of the Third International Conference on Knowledge Capture (K-CAP’05), 2005.

Dave, D., Lawrence, A., and Pennock, D. Mining the Peanut Gallery: Opinion Extraction and Semantic Classification of Product Reviews. Proceedings of International World Wide Web Conference (WWW’03), 2003.

Zhu, Jingbo, et al. "Aspect-based opinion polling from customer reviews." IEEE Transactions on Affective Computing, Volume 2.1,pp.37-49, 2011.

Na, Jin-Cheon, Haiyang Sui, Christopher Khoo, Syin Chan, and Yunyun Zhou. "Effectiveness of simple linguistic processing in automatic sentiment classification of product reviews." Advances in Knowledge Organization Volume9, pp. 49-54, 2004.

Nasukawa, Tetsuya, and Jeonghee Yi. "Sentiment analysis: Capturing favorability using natural language processing." In Proceedings of the 2nd international conference on Knowledge capture, ACM, pp. 70-77, 2003.

Li, Shoushan, Zhongqing Wang, Sophia Yat Mei Lee, and Chu-Ren Huang. "Sentiment Classification with Polarity Shifting Detection." In Asian Language Processing (IALP), 2013 International Conference on, pp. 129-132. IEEE, 2013.

Gurkhe D., Pal N., and Rishit B. "Effective Sentiment Analysis of Social Media Datasets using Naïve Bayesian Classification." (2014).

Bouazizi, M., Ohtsuki, T.: Multi-Class Sentiment Analysis in Twitter: What if Classification is Not the Answer. IEEE Access. 6, 64486-64502 (2018).

Gautam, G., Yadav, D.: Sentiment analysis of twitter data using machine learning approaches and semantic analysis. 2014 Seventh International Conference on Contemporary Computing (IC3). (2014).

Amolik, Akshay, et al. “Twitter sentiment analysis of movie reviews using machine learning techniques.” International Journal of Engineering and Technology 7.6 (2016): 1-7.

Mukherjee S., Malu A., Balamurali A.R, Bhattacharyya P.“TwiSent: A Multistage System for Analyzing Sentiment in Twitter.”

Davidov D., Tsur O., Rappoport A.” Enhanced Sentiment Learning Using Twitter Hashtags and Smileys.”

Neethu, M., Rajasree, R.: Sentiment analysis in twitter using machine learning techniques. 2013 Fourth International Conference on Computing, Communications and Networking Technologies (ICCCNT). (2013).

Pulkit Garg, Himanshu Garg, VirenderRanga “*Sentiment Analysis of the Uri Terror Attack UsingTwitter*” International Conference on Computing, Communication and Automation (ICCCA2017).

Prof. SudarshanSirsat, Dr.Sujata Rao, Dr.BhartiWukkadada”Sentiment Analysis on Twitter Data forproduct evaluation” IOSR Journal of Engineering (IOSRJEN) ISSN (e): 2250-3021, ISSN (p): 2278-8719PP 22-25.(2019)