International Journal of
INTELLIGENT SYSTEMS AND APPLICATIONS IN ENGINEERING
ISSN:2147-6799 www.ijisae.org Original Research Paper

A Model to Automate the Development of Computer Science Curriculum Syllabi

Ritu Sodhi[1], Jitendra Choudhary[2], Anil Patidar[3], Laxmikant Soni[4], Ritesh Joshi[5], Kuber Datt Gautam[6]

Submitted: 04/02/2024 **Revised:** 12/03/2024 **Accepted:** 18/03/2024

Abstract:Creating curricula and syllabuses is a crucial aspect of education. How well students are trained in computer science determines how good the students will be in the subject.A given course's syllabus is determined by several factors, including the objectives of the course, the resources available, the amount of time allotted, feedback from previous students, employers, and alumni, the aptitude of the learners, etc. Before developing a syllabus, course experts use a manual method that takes into account some variables and updates the syllabus as needed. Automation of the syllabus creation process is important because the needs of the software sector are always changing.This paper put out a model to make the process of creating syllabuses easier. The capability to design, modify, and store curricula will be offered by the model. Our model takes into account two factors: input from the industry and open-source course curricula from different universities. It will provide recommendations to the person who prepared the syllabus regarding its content based on these factors. The syllabus creator will finalize the contents by considering the suggestions given by the model and other attributes upon which it depends.

Keywords:Computer Science, Curricula,Feedback, Industry, Syllabus, University.

1. Introduction

Universities are pivotal in society, serving as the primary distributors of knowledge. A critical aspect of educational delivery involves the development of curricula, at the heart of which lies the creation of course syllabi. Experts in course development take into account various factors before formulating a syllabus, ensuring its periodic update as necessary.

A key determinant in shaping curriculum and course content is the evolving demands of the software industry. Given the dynamic nature of these requirements, collaboration between software companies and academic institutions is essential to enhance educational quality. This involves soliciting and integrating industry feedback into course design and content, necessitating an automated approach to keep pace with frequent changes.

Another influence on syllabus content is the availability of open-source syllabi from other academic institutions. Traditionally, syllabus developers manually review these resources, highlighting the need for a systematic approach to streamline the syllabus development process.

This document introduces a comprehensive model aimed at automating the curriculum creation process, enabling:

- The ability to generate, modify, and delete curriculum content.
- Bridging the industry-academia divide by incorporating feedback from software companies.
- Creating a repository of open-source course outlines for content extraction.
- Offering guidance to syllabus developers on course recommendations.

[1,2,3,4,6] *Medi-Caps University, Indore (M..P.) – India*
[5] *Ganpat University,Kherva, Mehsana (Gujrat) – India*
Email : [1]ritu.sodhi@medicaps.ac.in, [2]jitendra.scsit@gmail.com, [3]anil.patidar@gmail.com, [4]laxmikant.soni@medicaps.ac.in, [5]riteshjoshi.indore@gmail.com, [6]kuber.datt@gmail.com
** Corresponding Author :* Ritu Sodhi
Email: ritu.sodhi@medicaps.ac.in

The structure of this paper is as follows: Section 2 delves into the background and related studies. Section 3 identifies factors influencing the Syllabus Creation Process. In Section 4, we outline the proposed automation model for syllabus development, aimed at minimizing manual efforts. Section 5 compares traditional and new approaches to syllabus creation, evaluating them against various criteria. Finally, Section 6 summarizes the findings of our research.

2. Background and Related Work

This section reviews literature across three distinct realms: bridging the university-industry divide, developing a centralized syllabus repository and management tools, and exploring sentence similarity techniques.

2.1 Bridging the Gap between Universities and Industries

Efforts to narrow the divide between academia and the commercial sector have been widespread, with many scholars seeking employer feedback to align educational offerings with industry needs.

Almi and colleagues[1] conducted industry surveys revealing that employers value on-the-job training for students, enhancing their confidence and practical understanding. Key findings suggest that students should:

- Receive industry-specific training,
- Acquire technical skills,
- Understand industry expectations,
- Learn about emerging technologies, programming languages, and software.

S. Zeidan and M.M. Bishnoi [2] gathered insights on the skills required for 21st-century careers through surveys and focus groups involving students, industry professionals, academics, and alumni. The consensus indicated that current curricula fall short of meeting Industry 4.0's employability standards, highlighting the need for incorporating contemporary technological trends, soft skills enhancement, project-based learning, and internships. They advocate for active university-industry collaborations through practical engagements like internships and corporate interactions.

Exploring the agile software development approach, Pathak and team [3] conducted a case study using the SCRUM framework on academic projects, observing improved project quality and student awareness of agile methodologies.

Radermacher and colleagues[4] identified prevalent skill gaps among students in areas such as communication, project management, software tool proficiency, testing, teamwork, problem-solving, programming, and ethical practices, among others.

Qusay and team's[5] integration of mobile technology and application development into programming curricula showed significant benefits, equipping students with modern technological skills and better preparing them for the software industry.

Pan-Wei Ng and associates[6] critiqued the narrow focus on specific software development methodologies in academia, contrasting with the industry's diverse approaches. They advocated for the Software Engineering Methods and Theory (SEMAT) initiative, emphasizing the development of logical thinking to adapt to various practices and meet changing industry demands.

Dai and collaborators[7] proposed a revision of computer programming courses by reviewing top university syllabi and gathering feedback from students and seasoned educators. Their recommendations stress the importance of foundational programming concepts over language-specific details, advocating for case-based teaching and practical projects to enhance learning outcomes.

These studies collectively underscore the importance of integrating practical skills, industry expectations, and modern technologies into academic curricula to better prepare students for the dynamic demands of the workplace. However, there's a noted gap in discussions on adapting curricula to the fast-paced evolution of technology sectors.

2.2 Existing research on the development of a syllabus repository and tools for syllabus management.

The literature on the development of syllabus repositories and management tools reveals efforts to streamline syllabus access and organization. Researchers have employed web crawling techniques to aggregate computer science course syllabi, though challenges persist due to the prevalence of irrelevant links and the unstructured nature of online syllabi. Efforts to clean data and extract valuable information are crucial in this process. Additionally, some scholars have focused on creating tools for efficient syllabus management.

Xiaoyan Yu and team [8] developed an automated digital library for computer science courses to address the issues of searching for syllabi online, where irrelevant results and unstructured formats are common. Their approach involved techniques for recognizing, segmenting, and classifying information to transform unstructured syllabi into structured formats, facilitating easier access and potential educational benefits.

Mohammed Abdou[9] introduced a framework designed to simplify the tasks of syllabus creators, enabling them to generate, modify, erase, and disseminate syllabi effectively.

Bassam Hussein and collaborators [10] created an application to automate the syllabus development process, incorporating features to align course outcomes with broader program objectives, ensuring educational alignment.

Manas Tungare and colleagues[11] developed a syllabus management tool that integrates with Moodle as a plugin, offering capabilities for adding, editing, deleting, and comparing syllabi.

Chung and team [12] implemented an ontology-based syllabus classification scheme, converting unstructured syllabi into a structured format for repository storage.

Guberovic and associates[13] launched CSyllabus, a web application housing a syllabus repository from 11 universities, facilitating access and comparison of syllabi for students and educators. The application employs a course comparison algorithm using Latent Semantic Indexing, with feedback collected to evaluate its effectiveness.

Nakul Rathod and Lillian N. Cassel [14] trained machine learning classifiers to identify relevant computer science syllabi among search engine results, addressing the issue of navigating through numerous irrelevant links.

Mosharraf and team [15] adopted an Agent-Based modeling approach for syllabus repository creation, focusing on the extraction of pertinent information and the determination of syllabus contents.

These studies highlight the technological advancements in syllabus repository creation and management, aiming to enhance the accessibility and organization of educational materials. Despite these efforts, there is a noted absence of frameworks advising syllabus creators on content selection, indicating a potential area for future research and development.

2.3 Existing research on techniques for measuring sentence similarity

Significant research has been conducted in the domain of sentence similarity tests, a key component of natural language processing (NLP) with diverse applications including search engine ranking, plagiarism detection, and DNA pattern comparison. Notably, the application of sentence similarity analysis for comparing open-source syllabi and industry feedback remains underexplored.

Dastan Hussen Maulud et al.[16] evaluated various semantic analysis methods and tools, concluding that advanced techniques boast an accuracy range between 90 and 95 percent. Wael H. Gomaa et al. [17] explored different textual comparison approaches, distinguishing between lexical and semantic similarities. They found that string-based methods assess lexical similarity, while corpus-based and knowledge-based (specifically WordNet) approaches determine semantic parallels.

Issa Atoum et al. [18] analyzed methods for detecting word and sentence similarities, comparing corpus-based, knowledge-based, and hybrid approaches. Their findings suggest hybrid methods outperform the others in terms of results. Palakorn Achananuparp et al. [19] reviewed 14 semantic similarity methods, advising on the optimal method for specific data sets and highlighting its relevance in text mining, question answering, and text summarization.

In the biomedical field, Qingyu Chen et al. [20] investigated the efficacy of sentence similarity measures on PubMed abstracts for ranking purposes, discovering that neither lexical nor semantic measures fully meet ranking needs.

Zhe Quan et al. [21] combined semantic and syntactic information with an attention weight mechanism into a novel tree kernel, the ACVT kernel, enhancing sentence similarity analysis.

Shuang Peng et al. [22] introduced an Enhanced Recurrent Convolutional Neural Network (Enhanced-RCNN) model, offering a less complex architecture than BERT but achieving competitive performance on two datasets.

This body of work demonstrates the breadth of applications and ongoing innovations in sentence similarity analysis within NLP, indicating potential for further exploration in academic and industrial feedback comparison.

3. Factors Influencing the Syllabus Development Process

The process of curriculum development within universities is influenced by a multitude of factors, including:

- The mission and vision of the institution
- Desired outcomes of the program, including educational objectives and specific goals tailored to the program
- Resources available, such as faculty expertise, textbooks, and other educational materials
- Constraints related to time
- Characteristics and needs of the student body
- Regulations and standards set by government bodies and higher education authorities
- Feedback from various stakeholders, including employers, alumni, current students, and parents.

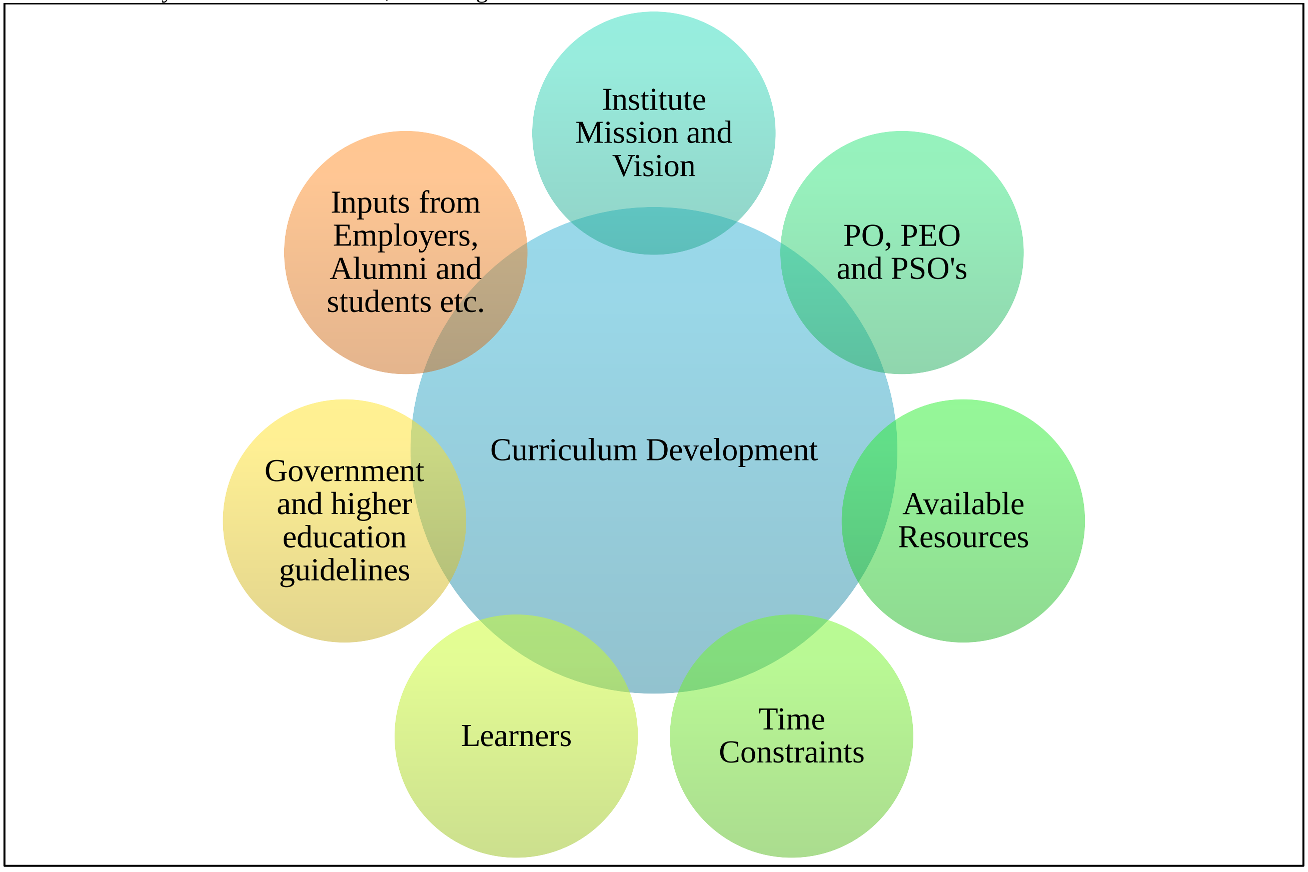

Institute Mission and Vision
PO, PEO and PSO's
Inputs from Employers, Alumni and students etc.
Curriculum Development
Government and higher education guidelines
Available Resources
Learners
Time Constraints

Fig. 1. Attributes Affecting Curriculum Development

The factors influencing curriculum development, illustrated in Figure 1, encompass a range of elements such as the educational institution's mission and vision, Program Outcomes (PO), Program Educational Objectives (PEO), and Program Specific Outcomes (PSOs), the availability of resources, time limitations, the demographic and needs of students, directives from government and higher education bodies, and contributions from employers, alumni, students, and parents.

Upon establishing the curriculum, the syllabus for various courses across different programs is crafted, taking into account factors like course goals aligned with program outcomes, educational objectives, and specific outcomes, available resources, time constraints, inputs from industry professionals, alumni, students, insights from open-source syllabi from other universities, and students' learning capabilities. During the development phase, the syllabus designer weighs these aspects to determine the syllabus content, ensuring the maintenance of syllabus versions to reflect updates or changes.

Designing a new syllabus presents challenges as creators must balance all these considerations to finalize the syllabus contents. This study aims to simplify the syllabus development process by proposing a model that assists both universities and course experts. Our model specifically focuses on two main factors affecting syllabus content: industry feedback and open-source syllabi from other institutions. By comparing industry feedback with various open-source syllabi, our model provides recommendations on potential syllabus content. These suggestions serve as a guide for syllabus creators, who must also take into account other pertinent factors. Given the frequent changes in industry requirements, our research advocates for the automation of the syllabus development process to ensure relevance and responsiveness to current needs.

4. Material and Methods

The proposed model for syllabus creation is detailed in the following step-by-step guide, aiming to streamline academic syllabus development:

Initiation by Syllabus Creator: When creating a new syllabus, the creator first gathers input on course content from industry stakeholders via the model and compiles a repository of open-source syllabi from various universities. Four scenarios are given below.

Utilizing the Syllabus Repository:

Content Extraction: Relevant information is extracted from the open-source syllabi, which often encompass entire semesters or programs, focusing specifically on course content, such as objectives, outcomes, and prerequisites, but prioritizing course material.
Semantic Identification: The model identifies the specific course within diverse documents, recognizing variations in course naming across institutions and pinpointing the start and end of course content based on common terminologies.
Comparison: The model compares syllabi to identify common topics, employing lexical or semantic comparison methods supported by existing libraries.

Weightage Calculation: Post-comparison, it calculates the significance of various topics.
Suggestions: Based on calculated weightages, ordered by significance, recommendations are provided to the syllabus creator for consideration in syllabus development.

Industry Input Integration:

Comparison: Similar to the repository approach, industry-provided content undergoes comparison to discern common themes.
Weightage and Suggestions: The significance of topics suggested by industry inputs is evaluated, with recommendations made accordingly.

Combining Repository and Industry Inputs: If both sources of information are available, the model integrates steps from both the syllabus repository and industry input phases.

Independent Creation: In the absence of both repository and industry inputs, the model supports the creation, modification, and management of syllabi, focusing on various educational attributes without specific content recommendations.

This structured approach facilitates the alignment of new syllabi with both academic standards and industry needs, enhancing the relevance of educational content. The model's operation, including the comparison of open-source syllabi and industry inputs to generate content suggestions, is visually represented in Fig. 2, which outlines all four scenarios and the associated steps comprehensively.

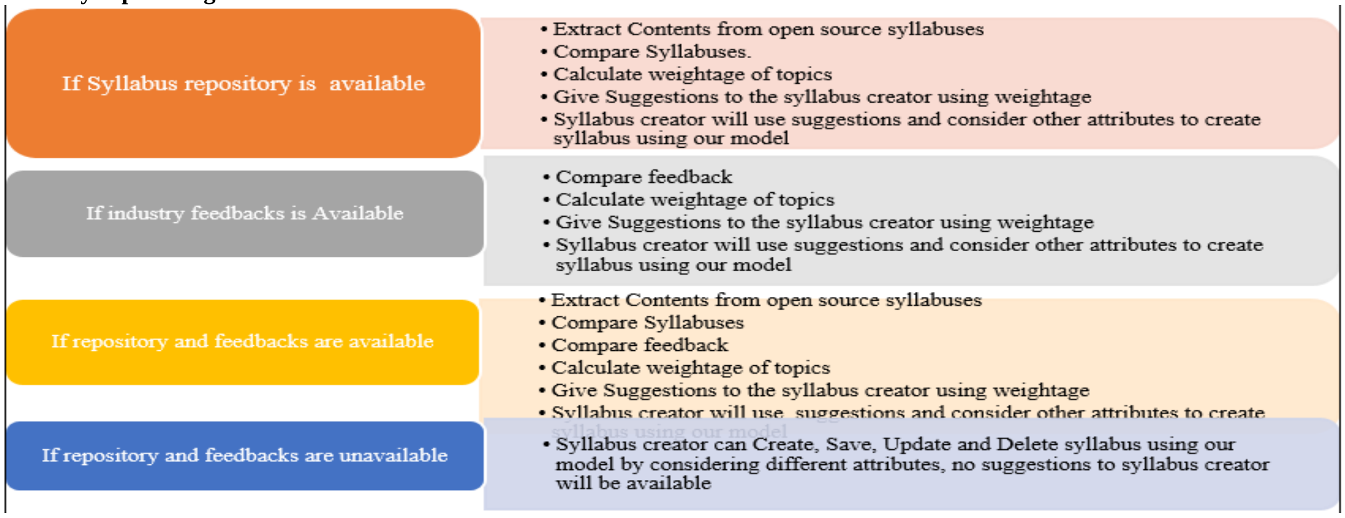
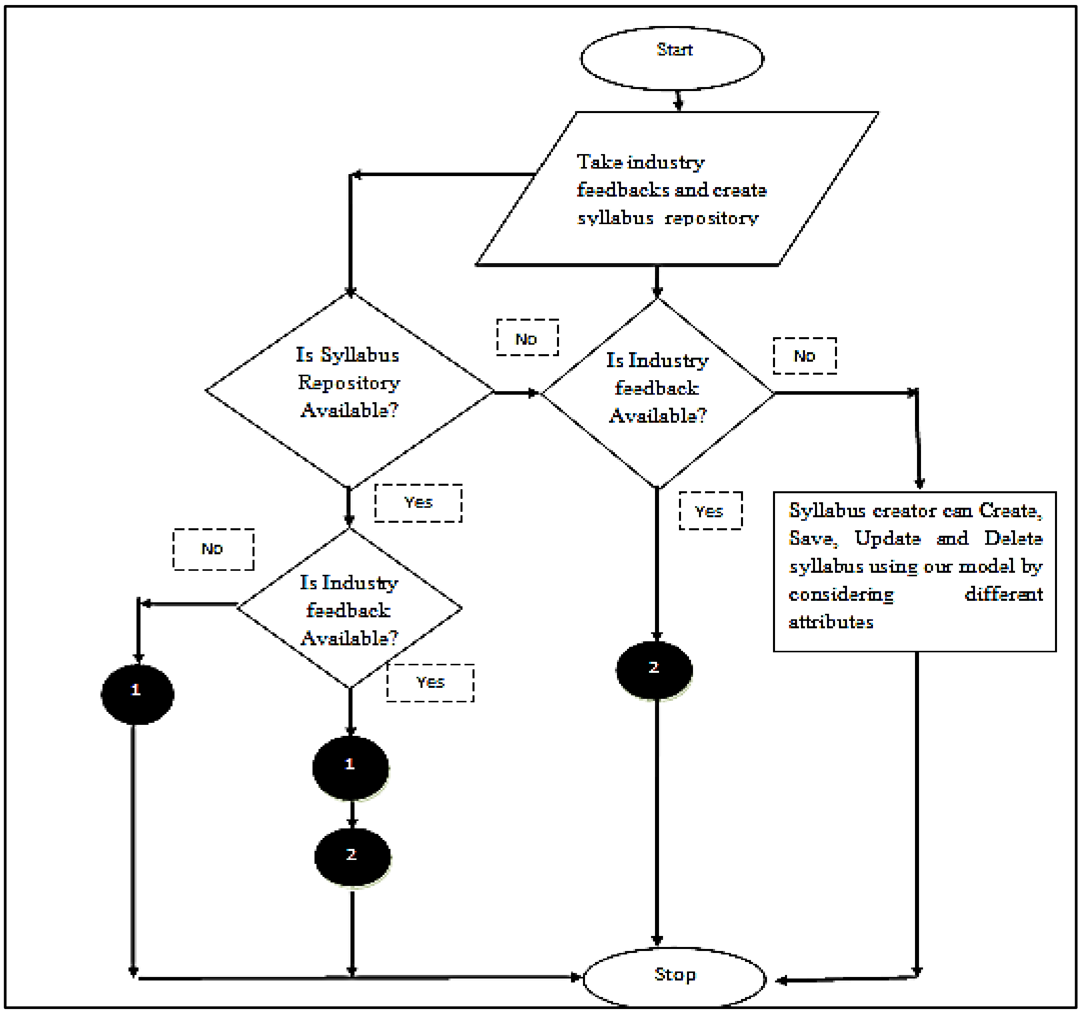
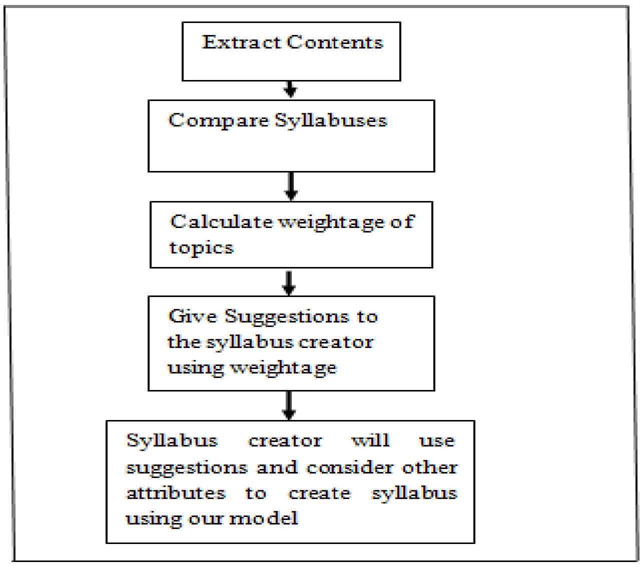

If Syllabus repository is available
• Extract Contents from open source syllabuses
• Compare Syllabuses.
• Calculate weightage of topics
• Give Suggestions to the syllabus creator using weightage
• Syllabus creator will use suggestions and consider other attributes to create syllabus using our model
If industry feedbacks is Available
• Compare feedback
• Calculate weightage of topics
• Give Suggestions to the syllabus creator using weightage
• Syllabus creator will use suggestions and consider other attributes to create syllabus using our model
If repository and feedbacks are available
• Extract Contents from open source syllabuses
• Compare Syllabuses
• Compare feedback
• Calculate weightage of topics
• Give Suggestions to the syllabus creator using weightage
• Syllabus creator will use suggestions and consider other attributes to create syllabus using our model
If repository and feedbacks are unavailable
• Syllabus creator can Create, Save, Update and Delete syllabus using our model by considering different attributes, no suggestions to syllabus creator will be available

Fig. 2. Proposed Model for Syllabus Creation Process

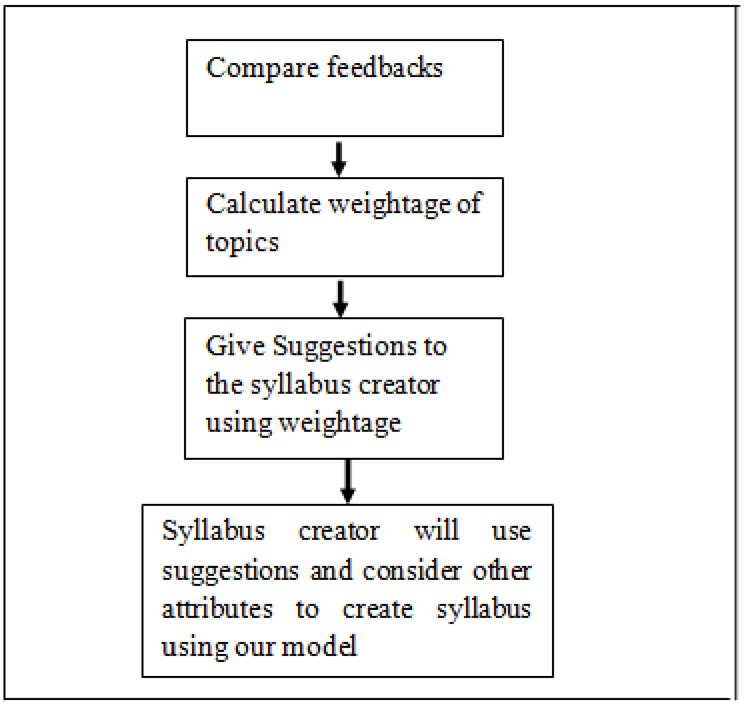
The proposed model is designed to navigate through four distinct scenarios, determined by the availability of feedback from industries and access to open-source syllabi for a given course. A visual depiction of this model is provided in Figure 3, which illustrates the comprehensive workflow. Specifically, Figure 3(a) presents an overarching flowchart of the model, while Figures 3(b) and 3(c) delve into the intricacies of subprocesses 1 and 2, respectively, elucidating their roles within the larger framework.


Start
Take industry feedbacks and create syllabus repository
Is Syllabus Repository Available?
No
Is Industry feedback Available?
No
Syllabus creator can Create, Save, Update and Delete syllabus using our model by considering different attributes
Yes
Yes
No
Is Industry feedback Available?
Yes
1
2
1
2
Stop

(a) Flow Chart for Automatic Syllabus Creation of Computer Science Courses


Extract Contents
Compare Syllabuses
Calculate weightage of topics
Give Suggestions to the syllabus creator using weightage
Syllabus creator will use suggestions and consider other attributes to create syllabus using our model

(b) Subprocess 2


Compare feedbacks
Calculate weightage of topics
Give Suggestions to the syllabus creator using weightage
Syllabus creator will use suggestions and consider other attributes to create syllabus using our model

(c) Subprocess 3

Fig. 3. Pictorial Representation of Proposed Model for Syllabus Creation Process

5.Comparison of the Traditional and Proposed Approaches

Table 1 showcases a comparative analysis between the conventional methods of curriculum and syllabus development and the methodology suggested by the proposed model.

Table 1. An analysis of the differences between the traditional and proposed approaches.

Parameter	Traditional Approach	Proposed Approach
Platform	In the traditional method, syllabus creators rely on an assortment of tools to perform the diverse tasks necessary for syllabus development.	Our approach offers a unified platform for creating, updating, and storing syllabi, incorporating industry feedback and assembling syllabus repositories. It facilitates the comparison of various open-source syllabi and industry feedback, integrating all these functions within a single interface.
Time	The conventional approach is more time-consuming as it requires syllabus creators to manually assess a variety of factors influencing the syllabus prior to finalizing the course content.	The proposed method is more efficient, as it provides the syllabus creator with content suggestions by analyzing open-source syllabi and industry feedback, thereby reducing the time required for syllabus development.
Input from industries	In the traditional approach, the syllabus creator manually gathers inputs from industries and conducts comparisons.	Our proposed model systematically gathers inputs from industries and conducts comparisons as part of its process.
Syllabus Repository	In the traditional method, syllabus creators are tasked with constructing a syllabus repository from scratch each time a new syllabus is developed.	In our model, the syllabus repository for a specific course is established just once by the syllabus creator. Should another individual wish to design a syllabus for the same course, they can utilize this existing repository, with the option to further enrich it by adding additional syllabi.

6. Conclusion

Before finalizing syllabus content, creators face numerous considerations, including soliciting feedback from industries and manually reviewing open-source syllabi from different universities. This meticulous process aims to bridge the significant gap often observed between the education computer science students receive and the actual demands of the software industry. To ensure relevance and efficacy in their educational offerings, universities and syllabus creators need to stay abreast of current trends and technologies.

We propose a model designed to streamline the syllabus development process and mitigate the disconnect between industry expectations and academic preparation. This model simplifies the task of syllabus creation by providing recommendations on potential content inclusions. It achieves this by analyzing two key sources of information: the wealth of open-source syllabi available from various universities and the valuable insights provided by industry feedback. Through this approach, our model aims to not only reduce the manual workload associated with syllabus development but also to enhance the alignment between academic curricula and the evolving needs of the software industry.

Acknowledgments

We thank our colleagues from Medi-Caps University Indore, who provided insight and expertise that greatly assisted the research, although they may not agree with all of the interpretations/conclusions of this paper.

Conflicts of interest

The authors declare no conflicts of interest.

References

[1] Nurul Ezza Asyikin Mohamed Almi, Najwa Abdul Rahman, Durkadavi Purusothaman and Shahida Sulaiman (2011). Software Engineering Education: The Gap Between Industry's Requirements and Graduates' Readiness. 2011 IEEE Symposium on Computers & Informatics, 2011.

[2] S. Zeidan and M.M. Bishnoi (2020). An Effective Framework for Bridging the Gap between Industry and Academia. International Journal on Emerging Technologies, vol. 11, pp. 454-461, 2020.

[3] Sonali Pathak, Pushpendra Pateriya, and Preet Pal (2012). A Case Study on Software Development Projects in Academic Knowledge Centers using SCRUM. International Journal of Computer Applications (0975 – 8887), vol. 43, pp.10, 2012.

[4] Alex Radermacher and Gursimran Walia (2013). Investigating the Skill Gap Between Graduating Students and Industry Expectations. SIGCSE '13: Proceeding of the 44th ACM technical symposium on Computer science education, pp. 525-530, 2013.

[5] Qusay H. Mahmoud, Thanh Ngo, Razieh Niazi, Pawel Popowicz, Robert Sydoryshyn and Matthew Wilks (2009). An Academic Kit for Integrating Mobile Devices into the CS Curriculum. ACM SIGCSE Bulletin, vol. 41, pp. 40-44, 2009.

[6] Pan-Wei Ng and Shihong Huang (2013). Essence: A Framework to Help Bridge the Gap between Software Engineering Education and Industry Needs. 2013 26th International Conference on Software Engineering Education and Training (CSEE&T), 2013.

[7] Kaiyu Dai, Yiming Zhao, and Ronghua Chen (2010). Research and Practice on Constructing the Course of Programming Language. 2010 10th IEEE International Conference on Computer and Information Technology, 2010.

[8] Xiaoyan Yu, Manas Tungare, Weiguo Fan, Manuel P´erez-Qui˜nones, Edward A. Fox, William Cameron, GuoFang Teng, and Lillian Cassel (2007). Using Automatic Metadata Extraction to Build a Structured Syllabus Repository. International Conference on Asian Digital Libraries, pp. 337-346, 2007.

[9] M'hammed Abdous and Wu He (2008). A Design Framework for Syllabus Generator. Article *in* Journal of Interactive Learning, pp. 541-550, 2008

[10] Bassam Hussein, Samih Abdul-Nabi, Zaher Merhi and Amin Haj-Ali (2015). A Web-Based University Courses Syllabi Generator. International Journal of Computer Applications (0975 – 8887), vol. 6, pp. 11, 2015

[11] Manas Tungare, Xiaoyan Yu, William Cameron, GuoFang Teng, Manuel A. P´erez-Qui˜nones, Lillian Cassel, Weiguo Fan and Edward A. Fox (2007), Towards a Syllabus Repository for Computer Science Courses. *SIGCSE 2007* Covington, Kentucky, USA, vol. 39, pp. 55-59, 2007.

[12] Hyunsook Chung and Jeongmin Kim (2016). An Ontological Approach for Semantic Modeling of Curriculum and Syllabus in Higher Education. International Journal of Information and Education Technology, vol. 6, pp. 5, 2016.

[13] E. Guberovic, F. Turčinovic, Z. Relja and I. Bosnic (2018). In Search of a Syllabus: Comparing Computer Science Courses. Conference: 2018 41st International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO), 2018.

[14] Nakul Rathod and Lillian N. Cassel (2013). Building a Search Engine for Computer Science Course Syllabi. Proceedings of the 13th ACM/IEEE-CS joint conference on Digital libraries, 2013.

[15] Maedeh Mosharraf and Fattaneh Taghiyareh (2020). Automatic Syllabus-Oriented Remixing of Open Educational Resources Using Agent-Based Modeling. IEEE Transactions on Learning Technologies, vol. 13, pp. 297-311, 2020.

[16] Dastan Hussen Maulud, Subhi R. M. Zeebaree, Karwan Jacksi, Mohammed A. M.Sadeeq and Karzan Hussein Sharif (2021). A State of Art for Semantic Analysis of Natural Language Processing. Qubahan Academic Journal, vol. 1(2), pp. 21-28, 2021.

[17] Wael H. Gomaa and Aly A. Fahmy (2013). A Survey of Text Similarity Approaches. International Journal of Computer Applications (0975 – 8887), vol. 68, pp. 13, 2013.

[18] Issa Atoum, Ahmed Otoom and Narayanan Kulathuramaiyer (2016). A Comprehensive Comparative Study of Word and Sentence Similarity Measures. International Journal of Computer Applications (0975 – 8887), vol. 135, pp. 1, 2016.

[19] Palakorn Achananuparp, Xiaohua Hu and Xiajiong Shen (2008). The Evaluation of Sentence Similarity Measures. 10th International Conference on Data Warehousing and Knowledge Discovery (DaWaK 2008) Turin, Italy, pp. 305-316, 2008

[20] Qingyu Chen, Sun Kim, W. John Wilbur and Zhiyong Lu (2018). Sentence Similarity Measures Revisited: Ranking Sentences in Pubmed Documents. 2018 ACM International Conference, *Washington, DC, USA*, 2018.

[21] Zhe Quan, Zhi-Jie Wang, Yuquan Le, Bin Yao, Kenli Li, and Jian Yin (2019). An Efficient Framework for Sentence Similarity Modeling. IEEE/ACM Transactions on Audio, Speech and Language Processing, vol. 27, pp. 853-865, 2019.

[22] Shuang Peng, Cui Hengbin, Niantao Xie, Sujian Li, Jiaxing Zhang, and Xiaolong Li (2020). Enhanced-RCNN: An Efficient Method for Learning Sentence Similarity. Proceedings of The Web Conference 2020, pp. 2500-2506, 2020.