Domain-specific NLP System to support learning path and curriculum design at tech universities

Nhi N.Y. Vo[a], Quang T. Vu[a], Nam H. Vu[a], Tu A. Vu[a], Bang D. Mach[a], Guandong Xu[b]

[a]*School of Science and Technology, RMIT University Vietnam, 702 Nguyen Van Linh Boulevard, District 7, Ho Chi Minh City, 70000, Vietnam*
[b]*Data Science Institute, University of Technology Sydney, 81 Broadway, Ultimo, 2007, New South Wales, Australia*

Abstract

The tech sector has been growing at a rapid speed, demanding a higher level of expertise from its labor force. New skills and programming languages are introduced and required by the industry every day, while the university courses are not updated adequately. Finding the high-demand skills and relevant courses to study has become essential to both students and faculty members at tech universities, which leads to a growing research interest in building an intelligence system to support decision making. Leveraging recent development in Natural Language Processing, we built an NLP-based course recommendation system specifically for the computer science (CS) and information technology (IT) fields. In particular, we built (1) a Named Entity Recognition (CSIT-NER) model to extract tech-related skills and entities, then used these skills to build (2) a personalized multi-level course recommendation system using a hybrid model (hybrid CSIT-CRS). Our CSIT-NER model, trained and fine-tuned on a large corpus of text extracted from StackOverflow and GitHub, can accurately extract the relevant skills and entities, outperforming state-of-the-art models across all evaluation metrics. Our hybrid CSIT-CRS can provide recommendations on multiple individualized levels of university courses, career paths with job listings, and industry-required with suitable online courses. The whole system received good ratings and feedback from users from our survey with 201 volunteers who are students and faculty members of tech universities in Australia and Vietnam.

This research is beneficial to students, faculty members, universities in CS/IT higher education sector, and stakeholders in tech-related industries.

Keywords: Data science applications in education, Architectures for educational technology system, Teaching/learning strategies, Adult learning

---

1. Introduction

In recent years, Computer Science (CS) and Information Technology (IT) fields have been evolving rapidly. With new programming languages, frameworks, tools, and systems coming out every month, the tech companies begin to demand new skills and knowledge when recruiting for specific technical roles. It becomes harder for universities to keep track of all the latest technologies required by the industry. Besides, along with the booming of e-learning technology and services, there are countless Massive Open Online Courses (MOOCs) available for free on different platforms, e.g. Coursera, EdX, etc. Some of these MOOCs are taught by professionals working at big tech companies, so the materials are often updated and are closer to the industry standard. The movement to online and remote learning with cost-effective micro degrees has made traditional and expensive CS/IT programs outdated [1]. As a result, the performance gap between skills taught at universities and those required by the companies is widened [2]. Many approaches are being investigated to close this gap [3, 4]. There has been a lack of intelligence systems to support student learning and curriculum design at CS/IT programs offered at universities.

On the other hand, advanced text mining techniques have been applied to solve Natural Language Processing (NLP) problems in other areas, especially using the Named Entity Recognition (NER) task [5]. There is an increasing interest in applying NER models for keyword extraction in CS/IT field [6, 7]. Initial studies on building an in-domain NLP system at tech universities have also been investigated by Ma et al. [8], Pardos and Jiang [9]. However, the current literature has either stopped at a theoretical model or a simple system without any real-world benefits to the students nor the universities.

Realizing this research gap, we conduct this research building an NLP-based course recommendation system to address the below research questions:

Q1 What skills/knowledge are still missing in these university programs?

Q2 Which future career can a student take with the current learning path?

Q3 Which courses should a student take to prepare for a specific tech career?

The research focuses on building an intelligence system to improve the quality of CS/IT programs at tech universities based on advanced text mining and NLP methods. It compares the knowledge and skills taught in the current course offerings at these universities and the real-life job requirements. The NLP system supports decision-making in the development of these programs and the improvement of universities' life-ready teaching curriculum. This research project contributes to both theoretical and practical knowledge based on both information systems and the education domain. The paper contributes significantly to both the extension of the current literature and the solving of real-world problems in two folds.

Theoretical contributions:

1. Developing a novel CS/IT domain-specific named entity recognition model to extract keywords and analyse the CS/IT courses and jobs description.
2. Being one of the first systems, according to our knowledge, to leverage a novel NLP model for a CS/IT course recommendation system based on industry requirements in higher-education universities.

Societal contributions:

1. Building a hybrid recommendation system to suggest new skills/courses for different CS/IT programs, which improves the program quality and students' satisfaction.
2. Constructing an intelligence system to assist students in choosing elective courses suitable for their career aspirations, which increases their technical capability and employability.

3. Enhancing the learning experience and quality of students, which leads to increasingly higher skilled graduates joining the workforce.

The research and the NLP course recommendation system aim to be beneficial to various stakeholders, e.g. CS/IT students (choosing suitable courses for their learning and career path), faculty members (updating curriculum design), universities (having better student experience and satisfaction), employers, and society (work-ready graduates with relevant skills).

2. Literature Review

2.1. NER models in CS/IT field

Named Entity Recognition (NER) has recently become a popular topic among NLP researchers. However, most NLP systems consist of NER models with real-world entities only, such as organization names, location, event date, etc. [10]. Multiple neural network architectures have been investigated for this task, e.g. the Convolutional Neural Networks (CNN), the Bidirectional Long Short-Term Memory Networks (BiLSTM) [11], the Gated Convolutional Neural Networks [12]. Other investigated deep learning approaches researched are adaptive co-attention network [13] and multi-task learning [14] in social media data.

Domain-specific NER is an emerging research trend, particularly in the CS/IT field. Little research has been carried on this specific topic, apart from some tech-related keyword extraction tools [15]. Some previous models of NER in CS/IT fields [6] are based on Conditional Random Field (CRF), a type of statistical modeling method for pattern recognition and structured prediction [16]. However, with the growth of artificial intelligence research, more deep learning techniques have been applied to build more advance NER models [17]. Huang et al. [18] have combined the BiLSTM with CRF to detect domain-specific entities in scientific literature and enrich their NER model with Rapid Automatic Keyword Extraction (RAKE). A Deep neural network for Bug-specific NER

(DBNER), introduced by Zhou et al. [19], has combined BiLSTM and CRF to build an enhanced NER model in the CS/IT field.

Regarding the embedding techniques, researchers have applied text mining on datasets from tech-related social media sites, particularly the two popular websites StackOverflow and GitHub. Using BERT [20] embedding methods as the base, multiple embedding models have been released, which enables the enhancement of NER models. Some of the state-of-the-art embedding methods are SciBert [21] and BERTOverflow [7]. We use BERTOverflow to benchmark our domain-specific model named CSIT-NER.

To conclude, these research projects resulted in a strong foundation for solving the NLP task with NER in CS/IT field. However, the literature has stopped at the theoretical NLP models stage. Little work has been done to apply these models to solve real-world problems. Realizing this research gap, we aspired to build and apply a CS/IT domain-specific NER model for a course recommendation system that can leverage advanced deep learning NLP techniques.

Moreover, transfer learning for the NER model has been recently investigated by Lee et al. [22] and Tabassum et al. [7]. The initial studies have shown the potential of this approach, which is the motivation of this paper. Our CSIT-NER model aims to achieve good performance compared to other state-of-the-art models and even better at transfer learning for accurately extracting the CS/IT entities. We can then leverage the CSIT-NER model for our decision support system for student learning and curriculum design.

2.2. Course Recommendation System for CS/IT students

While NER is an emerging NLP research topic, the Course Recommendation System (CRS) has been extensively investigated in the past decades [23]. Regarding traditional approaches, Lin et al. [24] have applied sparse linear method (SLIM) to query the top-N recommended courses in Information Management programs at Chinese universities, and Bhumichitr et al. [25] have used both Pearson Correlation Coefficient and Alternating Least Square (ALS) to suggest courses based on the similarity of their descriptions. Some researchers have tried

to solve the problem using ontology [26, 27], while Mondal et al. [28] have used the k-means clustering algorithm to build a grade-based CRS.

Different recommendation system approaches such as collaborative filtering have been studied by Hui et al. [29] to improve the CRS. Due to the popularity and data availability, there are more research works on recommendation systems for MOOCs than traditional university ones [30, 31]. However, most of these CRS is not constructed specifically for any domain, especially CS/IT programs, which is the research interest of this paper.

Regarding the application of NLP techniques to improve the CRS, some initial studies have been carried out in recent years [32]. Simple Word2Vec pipeline has been used for calculating semantic similarity between courses [8]. Pardos and Jiang [9] have also implemented a Course2Vec model (based on the Word2Vec model) with Recurrent Neural Networks (RNN) in their CRS. RNN has also been the chosen network for a goal-based CRS proposed by Jiang et al. [33]. Other NLP methods such as topic and sentiment analysis have been combined with survey data for a personalized CRS built by Ng and Linn [34].

While these NLP approaches have done a thorough job on the course description, none of them has incorporated the industry side by applying text mining on the job description. Their systems also focus only on finding similar courses, which is unrealistic as students would not want to learn the same topic again. Realizing this research gap, we aim to build a CS/IT domain-specific CRS (CSIT-CRS) that can recommend the courses with the new skills required by the industry and tech they have not learned yet. The CRS is personalized based on both their career interest and their learning history, combining both the CSIT-NER model and collaborative filtering with other user preferences.

3. Datasets and Methods

3.1. Datasets

We used multiple datasets to build our whole NLP system. The named entities of the StackOverflow and GitHub datasets were annotated manually.

Table 1: Overview of all datasets

Dataset	#docs	#sentences	#tokens	#entities
StackOverflow train	1,638	9,263	78,329	6,422
StackOverflow val	536	2,936	24,937	2,268
StackOverflow test	564	3,108	26,324	2,272
GitHub	143	8,023	50,447	4,095
Job description	2,154	30,293	1,085,731	2,907
Course description	1,385	9,399	270,543	1,886

The entities of the job and course description datasets are referring to only the skills extracted from the Emsi API. We combined these with the output predictions from the CSIT-NER model before feeding it to the CSIT-CRS.

3.1.1. NER datasets

To benchmark our CSIT-NER model, we used two public NER datasets in the CS/IT field, namely the annotated StackOverflow and GitHub NER datasets. Tabassum et al. [7] defined 20 types of annotated entities, including 8 coding language entities and 12 natural language entities. We focused only on the 11 relevant entities in this research. The entities are LIBRARY, APPLICATION, UI ELEMENT, LANGUAGE, DATA STRUCTURE, FILE TYPE, FILE NAME, VERSION, DEVICE, OS, and WEBSITE (plus the tag O for non-entity words).

3.1.2. Course description dataset

The dataset contains titles and descriptions of the available courses from 10 universities in Vietnam and Australia, namely University of Adelaide, Australian National University, University of Melbourne, Monash University, University of New South Wales, University of Queensland, University of Sydney, University of Western Australia, University of Technology Sydney and RMIT University Vietnam. In total, we had scraped 1,385 courses (updated in June

2021) from these schools. This dataset consists of 4 attributes: UNIVERSITY, COURSE CODE, COURSE NAME, and COURSE DESCRIPTION.

3.1.3. Job description dataset

The dataset was scraped from Glassdoor, including 2,154 jobs (updated in June 2021) from 12 different SE/IT career paths in Vietnam and Australia. We filtered the job listings using the 12 search terms: Android Developer, Cyber Security, Data Engineer, Data Scientist, DevOps, Full Stack Developer, iOS Developer, Machine Learning, QA, Software Engineer, Software Developer, and Web Developer. As we use the automatic scraping bots, the number of jobs could vary by 15%, depending on the current vacancies. This dataset consists of 4 attributes: JOB TITLE, JOB DESCRIPTION, COMPANY NAME, URL.

3.2. NLP System Overview

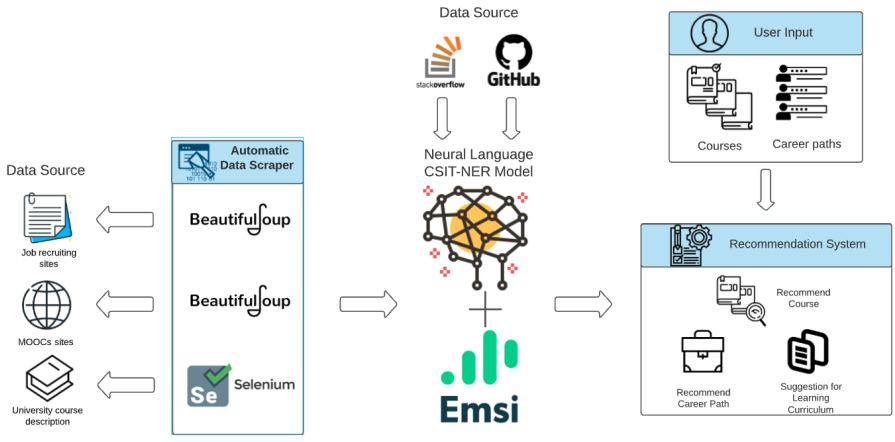


Figure 1: NLP System overview

Our NLP system used data from various sources. Firstly, we used BeautifulSoup and Selenium to scrape data from multiple websites which contained the job postings, course descriptions, and MOOCs online courses information (see Section 3.3). Secondly, we built and fine-tuned the CSIT-NER model as in

Section 3.4 by using the annotated corpus from StackOverflow and GitHub provided by Tabassum et al. [7]. The trained CSIT-NER was then applied to the scraped job and course datasets to extract relevant entities. Next, the predicted CS/IT skills were combined with output tokens from Emsi[1] to further extend our list of skills. The combined skill lists had been thoroughly checked and updated frequently, which further enhanced our hybrid CSIT-CRS (see Section 3.5). Finally, we built a web application to provide user interaction with the three distinctive use cases as described in Section 4.1.

3.3. Automatic Data Scraper

We build multiple scraping bots to automatically collect data from tech university courses, job listings, and MOOCs. The real-time data we use is susceptible to changes in the current vacancies and curriculum of universities. To ensure our system maintains a good level of data integrity and can react with the change of data sources, we take two aspects into considerations: (1) Framework for scraping bots to optimize resources and achieve high efficiency, and (2) Update frequency for each data source to ensure the database is up to date.

3.3.1. Framework

We use the Selenium and Beautiful Soup frameworks for our automatic scraping bots due to their efficiency and resource optimization [35]. The framework applied to each data source is as below:

1. Course Descriptions: The Selenium technique is used to simulate the login process and user behavior as some university websites require login to access their course guides.
2. Job Descriptions: Beautiful Soup technique is selected as the jobs are public on Glassdoor[2] with no restriction.

[1] Emsi is an open-source library with over 30,000 skills gathered from hundreds of millions of job postings, profiles, and resumes. See https://api.emsidata.com/

[2] https://www.glassdoor.com/

3. MOOCs: The Beautiful Soup technique is selected as The MOOCs are public on Class Central[3] with no restriction.

3.3.2. Update Frequency

All data sources are likely to change after a certain amount of time. We need to identify the sweet spot of the period to update our data to maintain data integrity while still optimizing resources.

The university course information is usually updated before the academic semester begins if there are any changes in the teaching materials. Universities often have the semester time of 18-22 weeks and 6-8 weeks of semester break. To keep up with the latest update, we update the scraping bots every two months.

The available job postings in Glassdoor change at a much faster pace as the industry grows quickly and tech companies are all looking out for talents. Each job posting often lasts for 1-3 months, depending on the company recruitment campaign. The scraping bot automatically updates weekly.

The online courses from MOOCs data help for both personal learning and getting hands-on experiences. Many courses last for years and receive feedback for over five years. Even good quality online courses come out every few months, but find difficulty competing with the reputable ones introduced in the past. Therefore, the bot for this data source automatically updates monthly.

3.4. CS/IT Named Entity Recognition model (CSIT-NER)

Figure 2 summarizes an end-to-end data pipeline for our CSIT-NER model. First of all, we obtained datasets from StackOverflow/GitHub and scraped data for jobs and courses from multiple websites. We split the StackOverflow dataset into a train, a validation, and a test set for our model training and evaluation. Secondly, we pre-processed the raw texts by removing English stop words using the NLTK Python package [36] and tokenized them with a pre-trained StackOverflow tokenizer. We then passed the tokenized corpus through embedding

[3] https://www.classcentral.com/

layers using in-domain fine-tuned BertOverflow and Fasttext models. After the sets were embedded, we used the train and validation sets to build and fine-tune our NER model. The embedding vectors were fed into a stacked BiLSTM and Fully Connected Layers as in Figure 3.

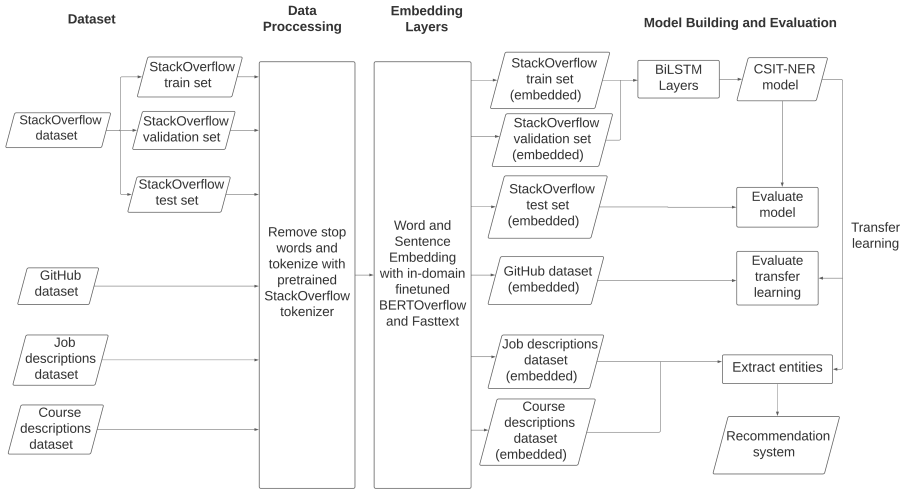


Figure 2: CSIT-NER Model Pipeline

Our CSIT-NER model containing 6,757,136 parameters was trained using Adamax optimizer with a learning rate of 0.001 and a batch size of 128 for 50 epochs (see Section 6.3.2 for parameter tuning). All datasets were imbalanced as more than 90% of words were not entities, so we used the bias initializer based on the class weight in the last layer to help improve the model accuracy. We also looked into the comparison of models with and without class weight in Section 6.3.1

We then evaluated the model on the StackOverflow test set. We further tested our model capability of transfer learning by assessing its performance on the GitHub dataset. Finally, we used the final CSIT-NER model to extract entities on the course and job datasets and integrated them into our CSIT-CRS.

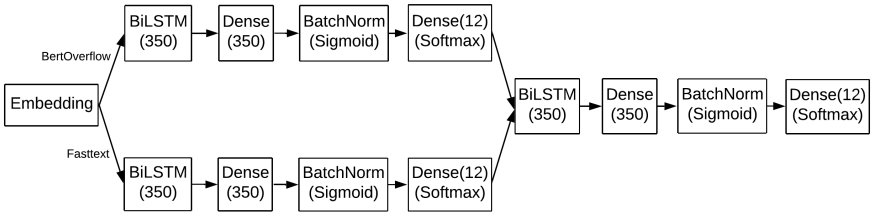


Figure 3: CSIT-NER neural network layers (units and activation in the brackets)

3.5. CS/IT Course Recommendation System (CSIT-CRS)

3.5.1. Use cases

Our CSIT-CRS is designed to support students and academic faculty members at universities in multiple situations.

For students, there are two different scenarios. The first scenario is when the students want to know which specific skills or courses they need to pursue a career path. They can enter the career options into the system and receive a list of most-demanded skills and tools required by the industry for that particular career path, together with a list of current courses at their university offering the previous skills and tools (Use case 1 on the Web Application). Alternatively, they can add their study history, including all the past courses, to receive a list of additional skills which they still need to improve on and the links to available MOOCs to help them acquire those skills (Use case 3 on the Web Application).

The second scenario is when the students are uncertain of which career options to follow, our system can recommend the most potential career paths based on their strengths and past performance. More specifically, they are interested in finding a job posted by companies that can match their skills and levels of expertise. The students can enter the courses they have taken and receive the following: a list of potential career paths together with a link to related job listings online so they can conveniently apply for those positions (Use case 2 on the Web Application).

For academic faculty members, our system can help them to keep track of new technologies/skills from the market to keep their courses up to date. The

lecturers or professors can enter a course or a list of their teaching courses together with a chosen field in the market. The system returns the following items: a list of skills/technologies from the chosen field which are still missing from their courses and the links to available online resources that can help them update or redesign their courses (Use case 3 on the Web Application).

3.5.2. Hybrid CSIT-CRS model

At the early stage, our system faced a cold start problem due to the lack of user interactions. Therefore, we implemented a context-aware RecSys using the extracted skills from the CSIT-NER models and Emsi API. The NLP-based model was further enhanced by a supervised ranking fusion method.

First of all, we denote τ_i as the list of extracted entities for course i and τ_j as those for job j, $i \in \{1, 2, ..., I\}$ and $j \in \{1, 2, ..., J\}$. We also have the context notions of the user's chosen career path cp_u and study history sh_u for user u, $u \in \{1, 2, ..., U\}$. We initially compute a map-reduce function by counting the entities, ranking the order of entity aggregations in τ_i and τ_j then getting the top ϵ most popular entities for each list. We denote these T_i^ϵ and T_i^ϵ.

After that, we can use T_i and T_j to calculate the context-aware similarity score between course i and job j as:

$$\omega(T_i^\epsilon \subset T_j^\epsilon) = \frac{T_i^\epsilon \cup T_j^\epsilon}{T_i^\epsilon \oplus T_j^\epsilon} \tag{1}$$

On the other hand, the context-aware dissimilarity score is given by:

$$\omega(T_i^\epsilon \not\subset T_j^\epsilon) = \frac{T_i^\epsilon \cap T_j^\epsilon}{T_i^\epsilon \oplus T_j^\epsilon} \tag{2}$$

We then can get the ϕ ranked top k courses by:

$$\rho_{ij}^\epsilon(k) = \phi(\omega(T_i^\epsilon \subset T_j^\epsilon), k) \tag{3}$$

Similarly, we have τ_s as the list of all the courses taken by student s, $s \in \{1, 2, ..., S\}$. Let T_s^ϵ be the top ϵ skills that student s has obtained from those courses, we can calculate the context-aware similarity score ω and get the ϕ ranked top k career paths and jobs by:

$$\rho_{js}^\epsilon(k) = \phi(\omega(T_j^\epsilon \subset T_s^\epsilon), k) \tag{4}$$

Lastly, the supervised ranking fusion approach combines pair-wise similarity and dissimilarity ranked lists to compile the final top k recommendations for each use case in our CSIT-CRS. We can calculate the dissimilarity score Δ to get the top l missing skills of the students, then compute the context-aware similarity score ω and get the ϕ ranked top k MOOCs by:

$$\Delta_j s^{\epsilon}(l) = \phi(\omega(T_j^{\epsilon} \not\subset T_s^{\epsilon}), l) \tag{5}$$

$$\rho_{mjs}^{\epsilon}(k) = \phi(\omega(T_m^{\epsilon} \subset \Delta_j s^{\epsilon}(l)), k) \tag{6}$$

where τ_m is the list of extracted entities for MOOC m, $m \in \{1, 2, ..., M\}$; and τ_m^{ϵ} be the top ϵ extracted entities of MOOC m.

Table 2: CSIT-CRS use case 1 results

	Model	MRR@k	MRP@k	MAP@k
k=5	Baseline 1	0.7271	0.7271	0.7236
	Baseline 2	0.7281	0.6049	0.6656
	CSIT-CRS	**0.9012**	**0.7992**	**0.8501**
k=10	Baseline 1	0.7306	0.7015	0.7190
	Baseline 2	0.7364	0.5508	0.6177
	CSIT-CRS	**0.9042**	**0.7334**	**0.8081**
k=15	Baseline 1	0.7305	0.6401	0.6976
	Baseline 2	0.7379	0.5197	0.5843
	CSIT-CRS	**0.9042**	**0.6999**	**0.7867**

Volunteers had rated the courses in the ranked order for the top 15 recommendations. We quantified the context-aware CSIT-CRS performance by the three ranking evaluation metrics measured at the top k (k=5,10,15), namely the Mean Reciprocal Rank (MRR) [37], the Mean R-Precision (MRP) [38], and the Mean Average Precision (MAP) [39]. We benchmarked our model in Table 2 against two common approaches in content-based RecSys: (1) similarity ranking with extracted skill and (2) similarity ranking with TF-IDF word vectors [40].

Our NLP-based RecSys received good feedback from the users and significantly outperformed the two baselines by 8% to 20%. This proved that our supervised ranking fusion approach was better than the traditional similarity ranking method. The CSIT-NER model also helped improve the performance significantly compared to the other common NLP approach using TF-IDF. The volunteers also rated our model on use cases 2 and 3 as in Table 3.

Table 3: CSIT-CRS rating use case 2 and 3 results

	k	MRR@k	MRP@k	MAP@k
Use case 2 - level 1	5	0.8896	0.8170	0.8479
Use case 2 - level 2	5	0.7828	0.7332	0.7550
Use case 3 - level 1	5	0.8442	0.7910	0.8083
Use case 3 - level 2	5	0.6808	0.6390	0.6559

All the rated courses and user interaction of the content-based model will be one of the inputs to build our hybrid CSIT-CRS. Figure 4 illustrates the whole framework of our hybrid CSIT-CRS model, where we combine both context-aware CSIT-CRS and collaborative filtering module with a Restricted Boltzmann Machine model. The hybridization layer is calculated as a weighted average dynamically adjusted based on the weight α regarding the context (cp_u, sh_u) of each individual user u.

$$r = \alpha_u^{(cp_u, sh_u)} r_1 + (1 - \alpha_u^{(cp_u, sh_u)}) r_2 \tag{7}$$

where r_1, r_2, and r are the recommendation 1 (context-aware), recommendation 2 (collaborative filtering), and the final recommendations of our hybrid CSIT-CRS model accordingly.

3.6. Baselines and Evaluation Metrics

We benchmarked our CSIT-NER model against other baselines and state-of-the-art models as described by Tabassum et al. [7], namely the Feature-based Linear CRF and SoftNER. We also included two baseline model variations using

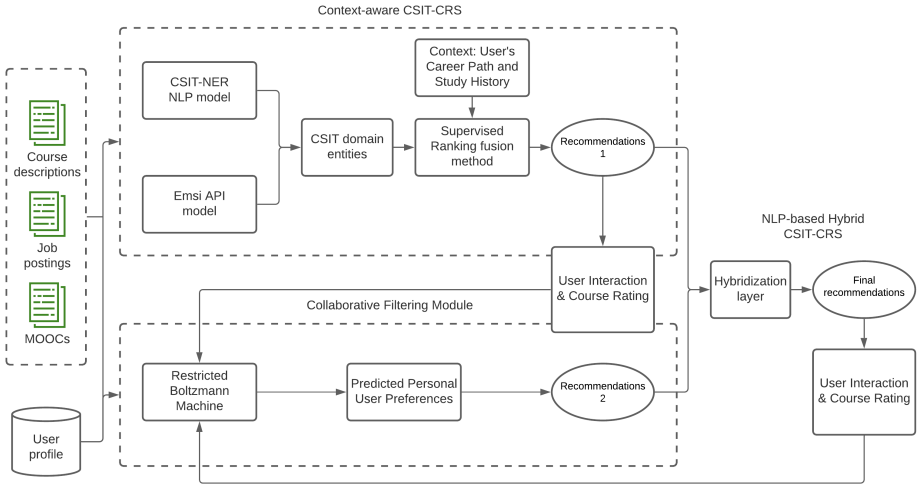


Figure 4: Hybrid CSIT-CRS framework

BiLSTM with in-domain BERT (BERTOverflow) or Fasttext embeddings. To test the transfer learning performance, we benchmarked our CSIT-NER model against a BiLSTM-CRF neural network trained on the GitHub dataset and the transfer learning from other mentioned approaches.

We used three different evaluation metrics to evaluate our CSIT-NER model, namely the precision, recall, and F1 score. The calculation for these metrics is as follows, where TP means True Positive and FP means False Positive:

$$Precision = \frac{TP}{TP + FP} \tag{8}$$

$$Recall = \frac{TP}{TP + FN} \tag{9}$$

$$F1 = \frac{2 * Precision * Recall}{Precision + Recall} = \frac{2 * TP}{2 * TP + FP + FN} \tag{10}$$

To benchmark our hybrid CSIT-CRS method, we built two baselines with popular collaborative filtering models, namely the Bayesian Personalized Ranking Matrix Factorization (BPRMF) [41] and the Restricted Boltzmann Machine (RBM) [42]. For BPRMF, we set the number of factors to 200, the learning rate to 0.01, lambda regularization to 0.001, and train for 100 iterations. For RBM,

we set the hidden units to 600 and batch size to 60, then train for 100 epochs. We reported the performances evaluated using a 5-fold cross-validation strategy with three ranking metrics Recall@k, NDCG@k, and MAP@k with k=5,10,15 [43].

4. Web Application and User Analysis

4.1. Web Application

We created a web application to implement the work result. The front-end of the web app used ReactJS [44] where the Material Design principle was applied as it is the current Google standard in web design[4]. For the back-end of the application, we chose Flask [45] as it is often preferred for deploying machine learning models thanks to its easy learning curve and flexibility.

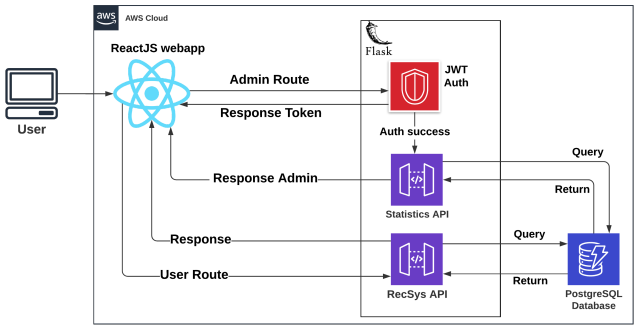


Figure 5: Web Application Architecture

Finally, the team used PostgreSQL [46] as the main database framework for the web application. PostgreSQL is well known for supporting machine learning related projects by integrating extensions, hence using PostgreSQL would provide easy implementation and scalability for our web application. To deploy the web application to the cloud service, the team used Amazon Web Services (AWS)[5], one of the most popular cloud platforms which are scalable

[4]https://material.io/design
[5]https://aws.amazon.com/

and reasonable in price. The AWS Relational Database Service (RDS) coming from the same package would be used to implement PostgreSQL instance on the AWS cloud services and Elastic Compute Cloud (AWS EC2) would be used to implement the front-end and the back-end of the web application.

The web application has two defined routes for users: Common User and Administrator. The normal user is only able to access the User Web Page. The User Web Page provides an interface to the 3 use cases as mentioned in Section 3.5. The Admin Web Page can be accessed using a pre-registered Administrator account with a hashed password. The Admin Web Page is secured by using JSON Web Token (JWT Token) [47] method to reduce the number of authenticating username and password for each request.

5. User Analysis

The user survey is divided into 3 sections: CSIT-CRS rating, UI/UX, and demographic questions. In the first and most important section of the survey, the CSIT-CRS rating is used to evaluate the participant regarding the relevancy and accuracy of the recommendation system regarding the three use cases.

The second section of the survey is the user rating for the UI/UX aspects of the web application. The questions consist of six questions in 5-level Likert-type scale to measure the satisfaction of the web application to the user:

Q1 How easy is it to navigate through the system?

Q2 How fast were the responses?

Q3 How easy to use do you think of the system?

Q4 What do you think of the system's design?

Q5 How likely will you use the system to help with study plan/course design?

Q6 How likely are you going to recommend the system to your peers?

The last section of the survey collects user demographics to group participants accordingly. The first question asks whether the participant is a student or faculty member. If they are a student, the second question asks for their

school year in university. Otherwise, it asks for their current position in the university. The final question is optional, asking for their general opinion about the system in form of short written feedback.

We collected 201 responses in total, of which 171 volunteers are students. There are 31 people in Year 1, 43 in Year 2, 57 in Year 3, 32 in Year 4, and 8 in Year 5 or above. The rest are 30 responses from faculty members from five different roles in their universities. We are interested in the preferences of our survey respondents based on either their chosen career paths or their demographic groups. As illustrated in Figure 6, the career aspirations distribution is relatively equal among all the volunteers for the twelve options, with a slight surge of interest in some trending careers such as Data Scientist and Machine Learning. This further validates the CSIT-CRS ratings Table 2 and Table 3, showing that the volunteers got a varied range of personalized recommendations.

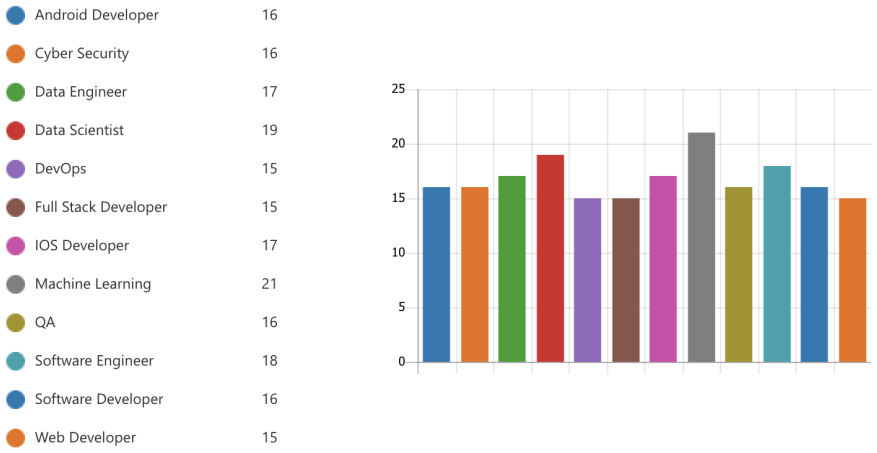


Figure 6: Career choices distribution

The results in Table 4 show that respondents are satisfied with our system in general. The scores range from 3.3860 (Year 3 - Q4) to 4.4333 (Staff - Q3). Questions 2 and 4, which measure our system response time and design, receive slightly lower scores than other questions. We will focus on improving these two aspects when we fully integrate our NLP system into the university network.

Table 4: Average UI/UX scores by group

Group	Q1	Q2	Q3	Q4	Q5	Q6
Year 1	**4.3226**	3.5484	4.1935	3.7097	4.1290	4.1290
Year 2	4.1628	3.4419	3.8605	3.4186	4.0465	4.0930
Year 3	4.2281	3.5263	3.8772	3.3860	4.0000	3.7895
Year 4	4.2500	3.5625	4.2188	3.4063	4.2188	4.1563
Year 5 or above	3.5000	3.5000	3.8750	3.3750	4.0000	**4.2500**
Staff	4.3000	**4.0333**	**4.4333**	**3.7333**	**4.2333**	4.0667

6. Empirical Results

6.1. CSIT-NER model results

Table 5 shows the experimental results of previous works and different proposed architectures of the CSIT-NER model on the StackOverflow validation and test sets. The CSIT-NER with character-sentence embedding achieved a significant improvement of 8% to 10% in all metrics compared to the state-of-the-art SoftNER model for the validation set. For the test set, it outperformed other models by at least 2% to 4%. More importantly, the model with character-sentence embedding had a trade-off balance between precision and recall and the highest F1 scores for both datasets, not suffering from high precision and low recall as in model BiLSTM with only BERTOverflow or Fasttext.

Table 6 demonstrates the model transfer learning performance on GitHub NER datasets in comparison with other baselines. Our CSIT-NER with character and sentence embedding can outperform the BiLSTM-CRF model which was trained specifically on the GitHub dataset. Moreover, our final CSIT-NER performed better than the model with only character embedding by 4% in the recall score. To conclude, our CSIT-NER model achieved great performance with both the StackOverflow and GitHub datasets. Especially with the significant results in the GitHub dataset, we could prove the transfer learning capability of our CSIT-NER model, which is deemed suitable for application to the job and

Table 5: CSIT-NER model performance on StackOverflow NER dataset

Dataset	Model	Precision	Recall	F1
Val set	Feature-based CRF	0.6945	0.4713	0.5674
	SoftNER	0.7058	0.6462	0.6683
	BiLSTM (BERTOverflow)	0.7373	0.5935	0.6293
	BiLSTM (Fasttext)	0.7340	0.6378	0.6739
	CSIT-NER (char-emb)	0.7302	0.6706	0.6915
	CSIT-NER (char-sent-emb)	**0.8054**	**0.7270**	**0.7618**
Test set	Feature-based CRF	0.7287	0.3996	0.5462
	SoftNER	0.7646	0.6873	0.7183
	BiLSTM (BERTOverflow)	0.7630	0.6059	0.6345
	BiLSTM (Fasttext)	0.7700	0.6389	0.6903
	CSIT-NER (char-emb)	**0.7833**	0.6934	0.7302
	CSIT-NER (char-sent-emb)	0.7708	**0.7288**	**0.7442**

Table 6: CSIT-NER model transfer learning performance on GitHub NER dataset

Model	Precision	Recall	F1
BiLSTM-CRF (trained on GitHub)	0.6453	0.6096	0.6269
Feature-based CRF	0.4513	0.3761	0.3978
SoftNER	0.6132	0.6066	0.6055
BiLSTM (BERTOverflow)	0.6330	0.5868	0.5883
BiLSTM (Fasttext)	0.6180	0.5870	0.5945
CSIT-NER (char-emb)	0.6386	0.6451	0.6326
CSIT-NER (char-sent-emb)	**0.6491**	**0.6954**	**0.6447**

course descriptions datasets in our CSIT-CRS.

6.2. Hybrid CSIT-CRS results

Table 7: Hybrid CSIT-CRS results

	Model	Recall@k	NDCG@k	MAP@k
k=5	BPRMF	0.4765	0.3870	0.3103
	RBM	**0.9821**	0.8323	0.7604
	Hybrid CSIT-CRS	0.9752	**0.8605**	**0.8112**
k=10	BPRMF	0.6407	0.4490	0.3454
	RBM	0.9820	0.8323	0.7604
	Hybrid CSIT-CRS	**0.9872**	**0.8655**	**0.8149**
k=15	BPRMF	0.7719	0.4907	0.3627
	RBM	0.9782	0.8390	0.7694
	Hybrid CSIT-CRS	**0.9872**	**0.8655**	**0.8149**

From Table 7, we can see that our hybrid CSIT-CRS outperforms both baselines on almost all the metrics for k=5,10,15. Particularly, we believe our context-aware model with CSIT-NER and supervised ranking fusion method has pushed the preferred courses higher in the recommended list, leading to an increase of 3% to 5% in NDCG@k and MAP@k. The BPRMF performance is significantly bad but slightly improved with k=15. This might be mainly due to the small number of users in our case. We can conclude that our hybrid CSIT-CRS is suitable for solving our cold-start problem.

6.3. Robustness Analysis

6.3.1. Imbalance Learning Test

High imbalance occurred on our dataset as the tag O for non-entity words took up more than 90% of our dataset. Besides the bias initializer mentioned in Section 3.4, we could further adjust our model by using the class weights

calculated for each tag in the dataset. To investigate the effect of this method, we compared our CSIT-NER model with and without class weight adjustment in Table 8.

Table 8: Imbalance learning test

Dataset	Model	Precision	Recall	F1
Val set	CSIT-NER	**0.8054**	0.7270	**0.7618**
	CSIT-NER (class weight)	0.7636	**0.7513**	0.7539
Test set	CSIT-NER	**0.7708**	0.7288	**0.7442**
	CSIT-NER (class weight)	0.7415	**0.7504**	0.7420
GitHub set	CSIT-NER	**0.6491**	0.6664	**0.6539**
	CSIT-NER (class weight)	0.6115	**0.6954**	0.6447

From the imbalance learning test, it can be observed that adding class weight did not affect the F1 score by any significant margin. It increased the recall metrics by trading off with lower precision scores. Our CSIT-NER model without class weight had high precision and low recall. This happens when the model returns only a few tags and most of them are correct, but the model misses many tags for programming fields. Meanwhile, the model using only bias initializer still achieved the maximized F1 score, which proved that it is robust against changes in imbalance learning methods. Therefore, we keep the CSIT-NER model without class weight as our final model.

6.3.2. Hyper-parameters Tuning

Training batch size and optimizer learning rate are two hyperparameters that can affect the overall performance of the model. Therefore, we want to test the robustness of our model against variations of these two hyperparameters.

Figure 7 reports the CSIT-NER test performance when changing different values of batch size, e.g. 32, 64, 128, and 256, measured by the F1 score on the validation set. For the first 3 models with batch size values of 32, 64, and 128, the F1 scores on the validation set were almost identical after 46 epochs.

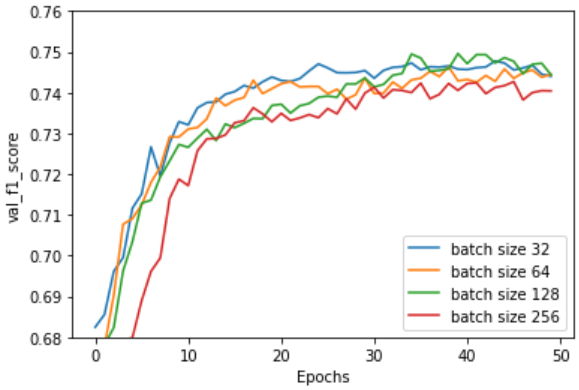


Figure 7: Parameter tuning with batch size

There was a slight decrease in the validation F1 score when the batch size was increased to 256. The differences were less than 0.5% at 50 epochs.

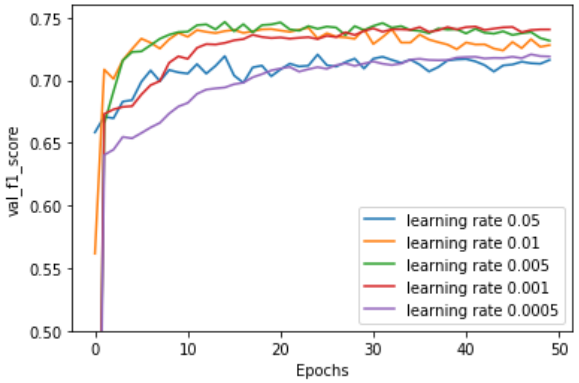


Figure 8: Parameter tuning with learning rate

Figure 8 provides the same experiences for our CSIT-NER model with the learning rate in the range from 0.05 to 0.0005. The first model with a 0.05 learning rate could reach 71% on validation f1 score. When the learning rate was decreased to 0.01, the F1 score reflected a small improvement, which was approximately 1%. The pattern continued as the learning rate was reduced further to 0.001 when the F1 score was slowly improved and reached 74%. However, further decreasing the learning rate to 0.0005 and 0.0001 reversed

the process of model performance improvement. The f1 score of this model is the same as the first model with a 0.05 learning rate. Therefore, changing the learning rate from 0.05 to 0.0005 for the model reflects minor change, which made the f1 score vary in the range of 3-4%.

Overall, variations of batch size and learning rate hyperparameters did alter the model results, but the changes were insignificant. Our CSIT-NER model still achieved a sufficient level of accuracy. Therefore, we can conclude that our model is robust against the parameter tuning effect.

6.3.3. Statistical Hypothesis Test

To validate our CSIT-CRS ratings and the survey data, we performed three different sets of statistical hypothesis tests. First of all, we conducted a quick Normality test using the Shapiro-Wilk [48]. Our data did not follow Gaussian distribution at 99% confidence level (p-value<0.01). Therefore, we decided to use non-parametric tests in the next step.

Table 9: Kruskal-Wallis H Test and Friedman Test results

		Kruskal-Wallis H Test		Friedman Test	
	Group split	t-value	p-value	t-value	p-value
Combined	A	66.0756	< 0.0001**	113.4913	< 0.0001**
	B	154.9135	< 0.0001**	216.1930	< 0.0001**
CSIT-CRS	A	83.1440	< 0.0001**	122.7711	< 0.0001**
	B	195.9316	< 0.0001**	249.0673	< 0.0001**
UI/UX	A	6.5875	0.2531	15.9569	0.0069**
	B	14.4019	0.2115	32.1921	0.0007**

** denotes statistical significant at both 95% and 99% confidence levels.

In the second set, we used the Kruskal-Wallis H Test [49] and Friedman Test [50] to compare the data between different groups of survey volunteers. We split the groups in two ways: (A) by expert level (6 groups: students in year 1, year 2, year 3, year 4, year 5 or above, and faculty members), and (B) by career

path (12 groups for 12 chosen career paths). We performed the tests for the combined CSIT-CRS ratings and UI/UX questions first and separated the two sections later. H0 is that the different groups have similar ratings and answers, while H1 is the opposite. The result in Table 9 shows that we can reject H0 and accept H1 for most tests at a 99% confidence level.

Table 10: Turkey's HSD Test results

UI/UX Question	meandiff	p-value	lower	upper	reject
Q1	−0.1012	0.5814	−0.4550	0.2527	False
Q2	−0.5187	0.0103*	−0.9136	−0.1238	True
Q3	−0.4392	0.0137*	−0.7874	−0.0910	True
Q4	−0.2772	0.1406	−0.6467	0.0923	False
Q5	−0.1573	0.3569	−0.4932	0.1786	False
Q6	−0.0491	0.7869	−0.3934	0.2952	False

* denotes statistical significant at 95% confidence level.

Finally, as we could not reject H0 in the Kruskal-Wallis H Test for UI/UX, we conducted a pair-wise Turkey's HSD test [51] at 95% confidence level to compare between students and faculty members for all 6 UI/UX questions. The H0 and H1 are similar as before. The results in Table 10 further confirm the reported values in Table 4. There are statistical differences in the answers of Q2 and Q3 among the 2 groups, but not for the other questions.

To conclude, all of the statistical hypothesis tests prove that different groups of survey respondents have different opinions and our reported results in Section 4 and Section 6.2 are statistically significant.

7. Discussion

From the empirical results, we can confirm that our NLP approach has a significant enhancement and contribution to personalized recommendation systems in CS/IT field. The CSIT-NER model achieves great results, which

lays a strong foundation for potential breakthroughs in this research direction, combining domain expertise and advanced neural network architectures. The hybrid CSIT-CRS, on the other hand, has answered the answers to all three research questions we set out when building this NLP system. Research question 1 has been answered by use case 3 where all the missing skills are listed and used for MOOCs recommendation. Use case 2 provides the answer for research question 2 with an extension of links to jobs for students to apply to. The most important research question 3 has been answered by both use case 1 and use case 3 where university courses and MOOCs are suggested based on individual preferences. The survey results and exhaustive tests have further confirmed the contribution of this paper to both current literature and society.

The whole system shows the social impact sides of these theoretical methods by providing effective and efficient methods that can serve multiple stakeholders. The NLP system benefits people in not only higher education sectors but also the tech industry in general. Students can find missing skills and courses to support the pursuit of their chosen career path. The academic faculty members can expand their knowledge based on the industry-required skills and update the courses using suggested online materials. Universities can leverage this to improve the student experience and attract more enrollments with better curriculum design. Companies and the industry are indirectly benefiting from this by recruiting graduates with more relevant skills.

As the transfer learning capability of our model has been proven, we can efficiently generalize the CSIT-NER model and scale the hybrid CSIT-CRS to support universities from different countries, not only Australia and Vietnam. Furthermore, cross-domain applications can be investigated in the future where we apply the same approach to the higher education sector in other fields, such as finance or marketing. Last but not least, our approaches can be expanded to various systems and applications outside of universities, e.g. using our CSIT-NER model for tech recruitment or implementing our personalized hybrid CSIT-CRS for internal training and up-skill programs at tech companies.

8. Conclusion

In this research, we have investigated the named entity recognition task specifically for CS and IT fields. We developed an in-domain CSIT-NER model based on a corpus of 15,372 sentences from StackOverflow and 6,510 sentences from GitHub. The model is one of the very first to investigate NER tasks for tech-related fields, outperforms state-of-the-art models, and is shown to be a promising benchmark for related future works. In addition, we developed an NLP-based hybrid CSIT-CRS for universities by applying the CSIT-NER model on datasets scraped from online course descriptions, job postings, and MOOCs platforms. The evaluation metrics and statistical results show that the system receives relatively high satisfaction among volunteers. Our system, with a strong societal benefit to the higher-tech education sector, is also a prospective candidate to implement in multiple universities and transfer learning to other majors of studies or different applications in the future.

Appendix A. Web Application User Interface

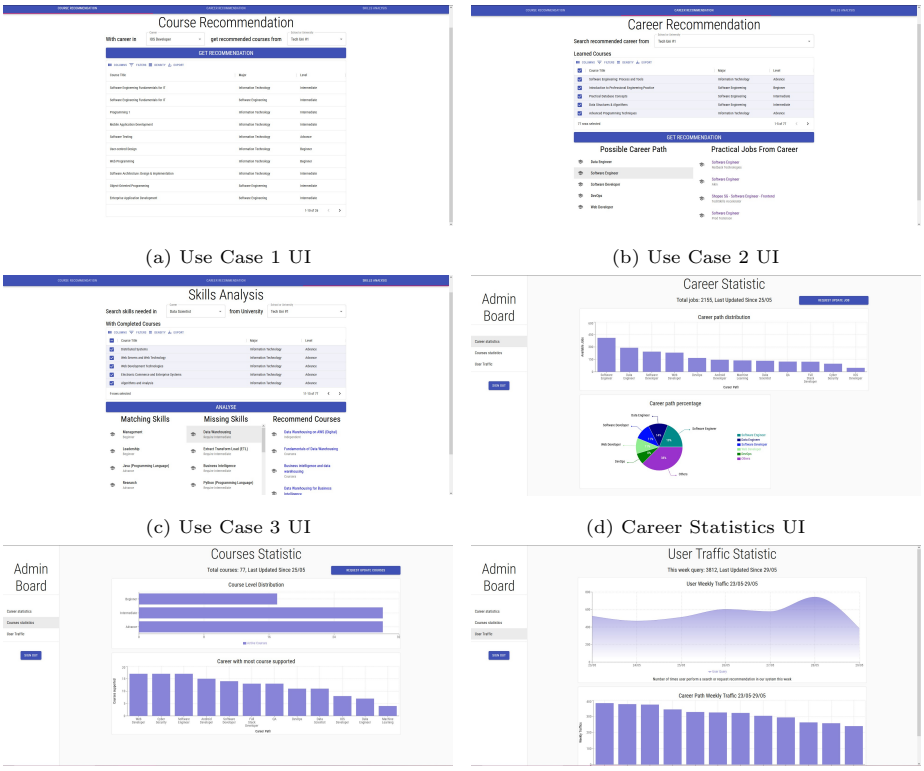
This appendix presents the User Interface of our Web Application, including the three use cases and three visualisation dashboards on the Admin portal.

References

[1] J. M. Prates, R. E. Garcia, J. C. Maldonado, MOOCs on the Context of Software Engineering Teaching and Training: Trends and Challenges, in: 2018 IEEE Frontiers in Education Conference (FIE), 2018, pp. 1–9.

[2] D. Oguz, K. Oguz, Perspectives on the Gap Between the Software Industry and the Software Engineering Education, IEEE Access 7 (2019) 117527–117543.

[3] V. Garousi, G. Giray, E. Tuzun, C. Catal, M. Felderer, Closing the Gap Between Software Engineering Education and Industrial Needs, IEEE Software 37 (2020) 68–77.



(a) Use Case 1 UI

(b) Use Case 2 UI

(c) Use Case 3 UI

(d) Career Statistics UI

(e) Courses Statistics UI

(f) User Traffics Statistics UI

Figure A.9: Web Application UI

[4] S. Valstar, Closing the Academia-Industry Gap in Undergraduate CS, in: Proceedings of the 2019 ACM Conference on International Computing Education Research, ICER '19, Association for Computing Machinery, New York, NY, USA, 2019, p. 357–358.

[5] C. J. Saju, A. S. Shaja, A Survey on Efficient Extraction of Named Entities from New Domains Using Big Data Analytics, in: 2017 Second International Conference on Recent Trends and Challenges in Computational Models (ICRTCCM), 2017, pp. 170–175.

[6] D. Ye, Z. Xing, C. Y. Foo, Z. Q. Ang, J. Li, N. Kapre, Software-Specific Named Entity Recognition in Software Engineering Social Content, in:

2016 IEEE 23rd International Conference on Software Analysis, Evolution, and Reengineering (SANER), volume 1, 2016, pp. 90–101.

[7] J. Tabassum, M. Maddela, W. Xu, A. Ritter, Code and named entity recognition in StackOverflow, in: Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, Association for Computational Linguistics, Online, 2020, pp. 4913–4926.

[8] H. Ma, X. Wang, J. Hou, Y. Lu, Course recommendation based on semantic similarity analysis, in: 2017 3rd IEEE International Conference on Control Science and Systems Engineering (ICCSSE), 2017, pp. 638–641.

[9] Z. A. Pardos, W. Jiang, Designing for Serendipity in a University Course Recommendation System, in: Proceedings of the Tenth International Conference on Learning Analytics & Knowledge, LAK '20, Association for Computing Machinery, New York, NY, USA, 2020, p. 350–359.

[10] A. Goyal, V. Gupta, M. Kumar, Recent Named Entity Recognition and Classification techniques: A systematic review, Computer Science Review 29 (2018) 21–43.

[11] Y. Shen, H. Yun, Z. C. Lipton, Y. Kronrod, A. Anandkumar, Deep Active Learning for Named Entity Recognition, in: Proceedings of the 2nd Workshop on Representation Learning for NLP, 2017, pp. 252–256.

[12] C. Wang, W. Chen, B. Xu, Named Entity Recognition with Gated Convolutional Neural Networks, in: M. Sun, X. Wang, B. Chang, D. Xiong (Eds.), Chinese Computational Linguistics and Natural Language Processing Based on Naturally Annotated Big Data, Springer International Publishing, Cham, 2017, pp. 110–121.

[13] Q. Zhang, J. Fu, X. Liu, X. Huang, Adaptive Co-attention Network for Named Entity Recognition in Tweets, Proceedings of the AAAI Conference on Artificial Intelligence 32 (2018).

[14] G. Aguilar, S. Maharjan, A. P. López-Monroy, T. Solorio, A Multi-task Approach for Named Entity Recognition in Social Media Data, in: Proceedings of the 3rd Workshop on Noisy User-generated Text, 2017, pp. 148–153.

[15] P. N. Mendes, M. Jakob, A. García-Silva, C. Bizer, DBpedia Spotlight: Shedding Light on the Web of Documents, in: Proceedings of the 7th International Conference on Semantic Systems, I-Semantics '11, Association for Computing Machinery, New York, NY, USA, 2011, p. 1–8.

[16] R. Collobert, J. Weston, L. Bottou, M. Karlen, K. Kavukcuoglu, P. Kuksa, Natural language processing (almost) from scratch, Journal of machine learning research 12 (2011) 2493–2537.

[17] J. Li and A. Sun and J. Han and C. Li, A survey on deep learning for named entity recognition, IEEE Transactions on Knowledge and Data Engineering 1 (2020) 1–1.

[18] H. Huang, X. Wang, H. Wang, NER-RAKE: An improved rapid automatic keyword extraction method for scientific literatures based on named entity recognition, Proceedings of the Association for Information Science and Technology 57 (2020) e374.

[19] C. Zhou, B. Li, X. Sun, Improving software bug-specific named entity recognition with deep neural network, Journal of Systems and Software 165 (2020) 110572.

[20] J. Devlin, M.-W. Chang, K. Lee, K. Toutanova, BERT: Pre-training of deep bidirectional transformers for language understanding, in: Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers), Association for Computational Linguistics, Minneapolis, Minnesota, 2019, pp. 4171–4186.

[21] I. Beltagy, K. Lo, A. Cohan, SciBERT: A Pretrained Language Model for Scientific Text, in: Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP), 2019, pp. 3606–3611.

[22] J. Y. Lee, F. Dernoncourt, P. Szolovits, Transfer Learning for Named-Entity Recognition with Neural Networks, in: Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018), 2018, pp. 4470–4473.

[23] H.-L. Thanh-Nhan, H.-H. Nguyen, N. Thai-Nghe, Methods for building course recommendation systems, in: 2016 Eighth International Conference on Knowledge and Systems Engineering (KSE), 2016, pp. 163–168.

[24] J. Lin, H. Pu, Y. Li, J. Lian, Intelligent Recommendation System for Course Selection in Smart Education, Procedia Computer Science 129 (2018) 449–453.

[25] K. Bhumichitr, S. Channarukul, N. Saejiem, R. Jiamthapthaksin, K. Nongpong, Recommender Systems for university elective course recommendation, in: 2017 14th International Joint Conference on Computer Science and Software Engineering (JCSSE), 2017, pp. 1–5.

[26] C.-Y. Huang, R.-C. Chen, L.-S. Chen, Course-recommendation system based on ontology, in: 2013 International Conference on Machine Learning and Cybernetics, volume 03, 2013, pp. 1168–1173.

[27] M. E. Ibrahim, Y. Yang, D. L. Ndzi, G. Yang, M. Al-Maliki, Ontology-based personalized course recommendation framework, IEEE Access 7 (2019) 5180–5199.

[28] B. Mondal, O. Patra, S. Mishra, P. Patra, A course recommendation system based on grades, in: 2020 International Conference on Computer Science, Engineering and Applications (ICCSEA), 2020, pp. 1–5.

[29] L. Hui, S. Jun, S. Zhang, H. Yun, Implementation of intelligent recommendation system for learning resources, in: 2017 12th International Conference on Computer Science and Education (ICCSE), 2017, pp. 139–144.

[30] J. Xiao, M. Wang, B. Jiang, J. Li, A personalized recommendation system with combinational algorithm for online learning, Journal of Ambient Intelligence and Humanized Computing 9 (2018) 667–677.

[31] C. De Medio, C. Limongelli, F. Sciarrone, M. Temperini, MoodleREC: A recommendation system for creating courses using the moodle e-learning platform, Computers in Human Behavior 104 (2020) 106168.

[32] S. Rao, K. Salomatin, G. Polatkan, M. Joshi, S. Chaudhari, V. Tcheprasov, J. Gee, D. Kumar, Learning to be relevant: Evolution of a course recommendation system, in: Proceedings of the 28th ACM International Conference on Information and Knowledge Management, CIKM '19, Association for Computing Machinery, New York, NY, USA, 2019, p. 2625–2633.

[33] W. Jiang, Z. A. Pardos, Q. Wei, Goal-Based Course Recommendation, in: Proceedings of the 9th International Conference on Learning Analytics & Knowledge, LAK19, Association for Computing Machinery, New York, NY, USA, 2019, p. 36–45.

[34] Y.-K. Ng, J. Linn, Crsrecs: A personalized course recommendation system for college students, in: 2017 8th International Conference on Information, Intelligence, Systems Applications (IISA), 2017, pp. 1–6.

[35] R. S. Chaulagain, S. Pandey, S. R. Basnet, S. Shakya, Cloud-based Web Scraping for Big Data Applications, in: 2017 IEEE International Conference on Smart Cloud (SmartCloud), IEEE, 2017, pp. 138–143.

[36] S. Bird, E. Klein, E. Loper, Natural language processing with Python: Analyzing text with the natural language toolkit, O'Reilly, 2009.

[37] E. M. Voorhees, D. K. Harman, et al., TREC: Experiment and evaluation in information retrieval, volume 63, MIT Press Cambridge, MA, 2005.

[38] H. Schütze, C. D. Manning, P. Raghavan, Introduction to Information Retrieval, volume 39, Cambridge University Press Cambridge, 2008.

[39] M. Zhu, Recall, Precision and Average Precision, Department of Statistics and Actuarial Science, University of Waterloo, Waterloo 2 (2004) 6.

[40] G. Salton, M. J. McGill, Introduction to modern information retrieval, McGraw-Hill, Inc., 1986.

[41] S. Rendle, C. Freudenthaler, Z. Gantner, L. Schmidt-Thieme, BPR: Bayesian personalized ranking from implicit feedback, in: Proceedings of the Twenty-Fifth Conference on Uncertainty in Artificial Intelligence, 2009, pp. 452–461.

[42] R. Salakhutdinov, A. Mnih, G. Hinton, Restricted Boltzmann machines for collaborative filtering, in: Proceedings of the 24th international conference on Machine learning, 2007, pp. 791–798.

[43] D. Valcarce, A. Bellogín, J. Parapar, P. Castells, Assessing ranking metrics in top-N recommendation, Information Retrieval Journal 23 (2020) 411–448.

[44] A. Fedosejev, React.js Essentials, Packt Publishing Ltd, 2015.

[45] M. Grinberg, Flask Web Development: Developing Web Applications with Python, O'Reilly, 2018.

[46] R. O. Obe, L. S. Hsu, PostgreSQL: Up and Running: a Practical Guide to the Advanced Open Source Database, O'Reilly, 2017.

[47] M. Jones, B. Campbell, C. Mortimore, JSON Web Token (JWT) profile for OAuth 2.0 client authentication and authorization Grants, May-2015.{Online}. Available: https://tools. ietf. org/html/rfc7523 (2015).

[48] S. S. Shapiro, M. B. Wilk, An Analysis of Variance Test for Normality (Complete Samples), Biometrika 52 (1965) 591–611.

[49] W. H. Kruskal, W. A. Wallis, Use of Ranks in One-Criterion Variance Analysis, Journal of the American Statistical Association 47 (1952) 583–621.

[50] M. Friedman, The Use of Ranks to Avoid the Assumption of Normality Implicit in the Analysis of Variance, Journal of the American Statistical Association 32 (1937) 675–701.

[51] J. W. Tukey, Comparing Individual Means in the Analysis of Variance, Biometrics 5 (1949) 99–114.