DESIGN AND DEVELOPMENT OF AN INTELLIGENT ONLINE PERSONAL ASSISTANT IN SOCIAL LEARNING MANAGEMENT SYSTEMS

A Thesis

Submitted to the Faculty

of

Purdue University

by

Seyed Mahmood Hosseini Asanjan

In Partial Fulfillment of the

Requirements for the Degree

of

Master of Science in Electrical and Computer Engineering

May 2019

Purdue University

Indianapolis, Indiana

THE PURDUE UNIVERSITY GRADUATE SCHOOL
STATEMENT OF THESIS APPROVAL

Dr. Brian King, Chair

Department of Electrical and Computer Engineering

Dr. Ali Jafari

Department of Computer Information Technology

Dr. Zina Ben Miled

Department of Electrical and Computer Engineering

Approved by:

Dr. Brian King

Head of the Graduate Program

To my parents, Houriyeh Farsad and Mohammad Hosseini.

ACKNOWLEDGMENTS

I would like to thank Dr. Ali Jafari for his support throughout this research project and providing an incredible opportunity to research and develop the proposed personal assistant.

I would like to appreciate Dr. Brian King for his guidance during my Master's program and being part of my thesis committee.

I would like to express my gratitude to Dr. Zina Ben Miled for sharing her experience and knowledge with me, supporting me in my research projects, and being part of my thesis committee.

TABLE OF CONTENTS

Page

LIST OF TABLES

LIST OF FIGURES

ABSTRACT

Hosseini Asanjan, Seyed Mahmood. M.S.E.C.E., Purdue University, May 2019. Design and Development of an Intelligent Online Personal Assistant in Social Learning Management Systems. Major Professor: Brian King.

Over the past decade, universities had a significant improvement in using online learning tools. A standard learning management system provides fundamental functionalities to satisfy the basic needs of its users. The new generation of learning management systems have introduced a novel system that provides social networking features. An unprecedented number of users use the social aspects of such platforms to create their profile, collaborate with other users, and find their desired career path. Nowadays there are many learning systems which provide learning materials, certificates, and course management systems. This allows us to utilize such information to help the students and the instructors in their academic life.

The presented research work's primary goal is to focus on creating an intelligent personal assistant within the social learning systems. The proposed personal assistant has a human-like persona, learns about the users, and recommends useful and meaningful materials for them. The designed system offers a set of features for both institutions and members to achieve their goal within the learning system. It recommends jobs and friends for the users based on their profile. The proposed agent also prioritizes the messages and shows the most important message to the user.

The developed software supports model-controller-view architecture and provides a set of RESTful APIs which allows the institutions to integrate the proposed intelligent agent with their learning system.

1. INTRODUCTION

Since the beginning of the information age, a variety of information technologies have influenced human lives and shaped social interactions. Online social platforms have attained numerous numbers of users who interact with each other every day. The growth in such environments has attracted business leaders to create more efficient software and platforms for their users. With the overwhelming increase in data, finding a reliable and useful material has become challenging for the users. This has enabled researchers and scientists to find different solutions for upcoming issues within these systems. Leading digital companies such as Apple, Google, and Amazon have announced their intelligent personal assistants to help their users in their daily life. Canalys in an article [1] has mentioned that the smart speakers market has grown 137% from the third quarter of 2017 to the third quarter of 2018. In another article, Gartner has predicted that the virtual personal assistants market will reach $2.1 billion by 2020 [2]. The educational area is not an exception. The development in online learning systems has allowed users to use online platforms for learning and engaging in a global education environment. The users throughout the world have been using online platforms to learn and earn their certificates and degrees. Modern social platforms have enabled users to create their professional and academic profile to find jobs and pursue their desired career. They also have helped the researchers to collaborate in a global setting.

The new generation of learning systems allow users to interact with each other socially and have access to global learning materials. However, creating a learning environment that the users feel connected to it, and finding a way to provide guidance is challenging. CourseNetworking as a social learning management system (LMS) created a platform that enables the users to not only have access to the LMS functionalities but also to interact and collaborate with other global users. The data in

such platform allows us to create an intelligent agent that learns about users and recommends useful content to them. The primary goal of this project is to design and develop a prototype of Rumi. Cyberlab at IUPUI with the collaboration of CourseNetworking is conceptualizing, researching and developing an intelligent agent called Rumi. Rumi's name is intentionally picked after a Persian poet who was known for his wisdom, and it reflects the personality of a wise man. This project focuses on the basic needs and interests of the learning systems, the institutions, and the users which are as follows:

- Social environments encourage their users to engage in global learning environments and collaborate.
- Institutions want to inform their members about important messages.
- Students create their ePortfolio within the social platforms and search for their desired jobs.

This project proposes an intelligent personal assistant which allows the institutions to communicate with their members in a personalized way, recommends jobs to the students, and recommends friends to the members.

2. RELATED WORK

2.1 Recommender Systems

Nowadays, recommender systems are being used in numerous industrial and academic areas. The overload of information and resources are sometimes misleading the user to find his/her desired material. The recommender systems help the users to find jobs, friends, or items based on their preferences. Users, who can be classified under educational resource seekers (like students, teachers) can benefit from the recommender systems. For instance, a student, who is looking for useful materials to prepare himself/herself for an upcoming exam, or an instructor, who wants to provide helpful resources to the students, are precisely objectives of recommender systems. The recommender systems use prediction methods which utilize the users' features like basic information, demographic data, purchase history and ratings, and the items data like description, price, and sale ranking. There are a variety of challenges such as sparse data-sets or cold start problem in the development of recommender systems [3]. The fundamental approaches in the design and development of the recommender systems are described in Section 2.1.1. Some of the existing recommender systems are mentioned in Section 2.1.2. Section 2.1.3 focuses on the existing methods for link recommendations and common practices.

2.1.1 Fundamental Approaches

Collaborative Filtering

The name of Collaborative Filtering (CF) is adapted from one of the earliest recommender systems, Tapestry [4]. The hypothesis of collaborative filtering is if two users A and B perform similar actions like purchasing the similar items, then

they will have similar behavior on other items [5]. The common way to implement CF is to create two lists of $Users$ and $Items$ and predict the missing values in the $Users - Items$ matrix [6]. There are variety of challenges for CF recommender systems. The high number of users and items creates an extremely sparse data-set. The memory-based CF algorithms that use user data to find similarities between users or items are highly used in commercial websites [7] [8]. Some of the notable challenges of the collaborative filtering recommender systems are:

- **Data Sparsity:** Most of the commercial projects have large databases. This makes the user-item matrices very sparse and makes the recommendation challenging. In [9], the authors argue that common data-sets have more than 90 percent sparsity level. Authors in [10] illustrate that Naïve Bayes models perform better than other models in highly sparse environments. There are two common approaches to handle sparse data-sets. First, eliminating zero values from the data-set [11] [7] [12]. Second, approximating the missing values. Authors in [11] mention that replacing the most frequent value with the missing values is the most common technique to handle the data sparsity problem.

- **Cold Start:** Once a new item or a new user is created in the platform, there is not enough data to display a meaningful recommendation [13]. Some techniques such as Singular Value Decomposition (SVD) try to reduce the dimension of the matrix by removing some of the items and the users from the recommendation process [14]. Such approaches also are challenged by losing essential data about the discarded users or items, and reduction in recommendation quality [7] [15]. In other approaches such as content-boosted CF [16] and taxonomy-driven recommender systems [17], hybrid CF methods are proposed that use external data about the new items or new users. In another approach [18], authors suggest using a probabilistic method to recommend items based on the Gaussian distribution of the user ratings.

Naïve Bayes Model-based Collaborative Filtering

Using the predictive models in the design and development of the intelligent agents allows the platform to learn from the data, train itself and make intelligent decisions. Model-based CFs such as Bayesian CF models are researched in several articles [12] [19]. In a naïve Bayes CF it is assumed that the features are independent of each other, but they have a direct impact on the class attribute. In this method, the probability of the class attribute happening is calculated based on the given features [19]. The author in [12] propose Equation 2.1 to predict a user's vote on an item. The highest calculated probability of the user's vote, given the user's features, is used as the predicted result of the recommender system.

$$predicted = arg\ max\ p(class_i)\ \Pi\ P(X_o = x_o|class_i) \tag{2.1}$$

In [20] the authors compare collaborating filtering and Bayesian classifier models, and use the same Equation 2.1 to predict the probabilities. The data is transformed to binary and then to a vector, which makes the prediction model perform faster. However, this approach may cause loss of data. The same authors used this model on a data-set that includes only binary data [19].

2.1.2 Existing Recommender Systems

In [7], for each user, Amazons item-to-item collaborative filtering recommendation system matches purchased and rated items with similar items. By combining similar items, a recommendation list is created. Similar items are distinguished by extracting items that users buy together. In this paper, to compute the similarity between two items, a vector corresponding to an item with M dimensions has been used in which each dimension is a customer who has bought the item. Considering this, the worst case time complexity of the algorithm is extremely high $O(N^2M)$. Where N is the

number of items and M is the number of users. However, as the number of purchased items for each user is so small, the time complexity becomes closer to $O(NM)$.

The proposed recommender system in [21] extracts data about each book based on a simple pattern-based information-extraction tool. This tool finds a list of the sub-strings from each document (fillers) for each specific slot. Slots are the title, authors, description, published reviews, customer comments, related authors, related titles, and subject terms. To extract information about a book, at least an abstract or a review is enough for this method to extract its desired data. To process each slot, the bag of words technique has been used (a bag for each slot). The Bayesian text classifier is used as the learning method in the bag of words technique. Naïve Bayes assumption illustrates that the probability of occurrence a word is dependent on the document, but it is independent from the word's location on the document. Therefore, a multinomial text model can be created which models a document as a sequence of words. The authors use Laplace to eliminate zero probabilities. To avoid the arithmetic underflow, the authors use logarithms of the calculated probabilities.

The proposed algorithm in [22], Regression-based Latent Factor Models (RLFM), can be described as two-stage hierarchical latent factor models which are trying to find similarities in the given data-set. The primary goal of this algorithm is to predict users' ratings on the items. By mapping item features and user features to the specific ratings, the algorithm learns the relationship between features and the ratings. Then, the model calculates affinity values based on observed features and latent vectors corresponded to them. In the first stage of the model, the affinity values is calculated based on the observed features and latent vectors when they are all known. Then, in the second stage, a generative model for the latent vectors is determined.

The research in [23] proposes a taxonomy-based recommendation system that uses available taxonomies to overcome the well-known problems of latent vector models. First, because of sparsity in purchase history, learning from latent factors is hard. Second, once new items are added to the catalog, these models won't be able to learn correctly about such items (cold start problem). Third, these models struggle in

recommending time-related items based on the user's behavior. The algorithm defines vector features for each user and item with the dimension of $1 * K$ (K is the model's dimensionality). User factor is responsible for capturing long-term interests of the user. Also, features are created for all of the nodes in the item taxonomy. Therefore, items under the same sub-category in the item taxonomy have similar features. By having these, for each user and for each time step the score which is showing the likelihood that the user purchases an item is calculated. This score is the summation of the long-term interest and the short-term interest of the user. Being interested in buying a flash drive after buying a camera is an example of the short-term interest of a user. Researchers in [24] propose a taxonomy-based recommendation system that learns and discovers possible taxonomies from raw data. The advantages of this project that are also mentioned in the paper can be categorized into three major points. First, in comparison to other approaches, this method is not using human-created taxonomies. Creating proficient taxonomies can be costly, and most of the data-sets do not have human-induced taxonomies. Therefore, employing this method can be useful to extract a proper taxonomy for those data-sets. Second, human-created taxonomies are stationary and do not change by adding new items to the catalog. Third, sometimes human errors can cause noises in the taxonomy. Using automatic taxonomy generator can solve this problem. First, the items is organized in a tree. Starting from the root, an item is attached to the root with the probability that is proportional to the item's frequency. Each item has a latent vector which is sampled from its parent and has a description based on multinomial distribution based on its parent's description. Also, the item includes a popularity measure generated by a normal distribution.

A Cyberlab graduate assistant fellow, Mirzaeibonehkhater has proposed a dynamic recommendation system to personalize educational context with the CourseNetworking platform [25]. The system's aim is to deliver a robust recommendation system that recommends the most relevant posts to the users. The recommended posts are discovered based on specific course categories that the current user is enrolled in or

the topics that he/she has used in his/her posts. The proposed system uses a ground truth matrix (GTM) and matrix factorization methods to develop the desired platform. The main factors that are considered in the development process are accuracy, cold start problem, up-to-date results, and self-updating system. The proposed recommender system uses both user and posts features. The selected features for the users are the role of the current user (student / instructor), user's rating on a post, and user's reflection. The selected features from the posts are content, creation time, hashtags, attachments, and a binary value that shows if a given post was chosen as post of the week. The developed method uses natural language processing (NLP) methods to analyze the content of the posts. The system performs accurate predictions for user ratings with an accuracy of 98% in the user-post matrix.

2.1.3 Link Recommender Systems

The objective of the recommender systems varies based on the business perspective. The recommender systems are widely used in recommending movies, books, and music. The growth in the social networking area created a new domain of research studies [26]. In [27], the authors argue that link recommendations are one of the standard features in social networking websites. Social networks can be described as graphs in which the vertices are the users, and the edges indicate an association between the users. The goal of link recommender systems is to find the likelihood of a potential link between two users. The research [28] defines link prediction as a graph $G(V, E)$ where $e = (v, u) \in E$ shows an association between v and u at time $t(e)$. As $G[t, t']$ is a subgraph of G, we can formulate the training set and testing set as $G[t_0, t'_0]$ and $G[t_1, t'_1]$ respectively where $t'_0 < t_1$.

Model-based recommendation methods create a model from existing data and learn the patterns based on the existing links. The goal of the model-based methods is to predict the linkage likelihoods based on the created model. In a social network, we can export the data based on the existing features and class attribute. As it is dis-

cussed in [29] and [30], topological features such as the shortest distance between two users and the number of mutual neighbors are commonly used model-based recommender systems. It is also common to use nodal information such as users' locations, age, and education in such models. A variety of machine learning (ML) techniques have been used in link prediction models. In [29] authors use logistic regression on the data from CiteSeer and Enron data-sets to predict the likelihood of the users interacting with each other. The researchers in [31] employ both logistic regression and Markov random field methods on DBLP and PubMed data-sets to predict the likelihood of collaborations between the authors. In [32], the authors use DBLP data-set to predict co-authorship likelihood between the users by using decision tree classifiers. The authors in [33] use support vector machine (SVM) to predict links within the Google+ platform.

Some other methods use the proximity between the users as the main factor of their recommendation systems. In [34], the authors argue that similar users have a higher likelihood to interact with each other. In [35], the proposed link recommender system uses keywords to find similarities between the users. Some of the used similarity calculation functions are as follows:

- Manhattan Distance: The Manhattan distance between two nodes M and P is calculated by $d(M, P) = |M_x - P_x| + |M_y - P_y|$
- Cosine Similarity: The cosine similarity of two vectors A and B is calculated by $cos(\theta) = \frac{A.B}{||A||||B||}$
- Jaccard Index (Coefficient): The Jaccard similarity coefficient of two vectors A and B is calculated by $J(A, B) = \frac{A \cap B}{A \cup B}$

2.2 Intelligent Agents in Education

An intelligent agent program is a software that can collect data and analyze and process the information to act autonomously [36]. Today, the agents are individ-

ualistic applications which can be connected to other online or offline applications and databases, learn about their environment, and do something purposeful in return [37]. Such systems help the users to interact with the current software in a better way. Also, the agent personalizes its actions based on the users' preferences. As it is mentioned in [37], an agent software's can behave like a human secretary or an assistant, who can prioritize the tasks based on their importance, and make decisions on behalf of the users. Negroponte in Being Digital defines the best agents as a "digital sister-in-law" [38] and explains that the best agents not only have a comprehensive knowledge in a certain area but also consider the user's precedencies. In [39] and [40] the authors categorize an intelligent agent's characteristics into nine categories:

- Reactivity: collecting and perceiving data, and acting accordingly.
- Autonomy: acting objectively and independent.
- Collaborative behavior: collaborating with other agents to accomplish a particular purpose.
- Communication: interacting with humans and connecting with other agents.
- Inferential capability: using previous data and user's preferences to reach a better result.
- Temporal continuity: being consistent with previous actions.
- Personality: being human-like and displaying such features as emotion.
- Adaptivity: learning, training and adapting itself.
- Mobility: being able to transfer itself to another platform.

Considering above-mentioned categories, scientists categorize the agents in other different classification layouts. This allows us to distinguish the importance of the different features of an agent and improve the design and development process of the agent software.

3. COURSENETWORKING

3.1 Introduction

As it is elaborated in the CourseNetworking's white paper [41], CourseNetworking is a social learning platform, that is focused on creating the next generation of the learning management systems (LMS.) Today's standard learning management systems' main goal is to create a course management platform that allows the institutions to deliver courses and manage them. However, as [42] illustrates, using social networking in higher education increases the students' sense of connection to the other students. Besides, it shows that the students in higher ed., who use social networking, seem to have better feelings towards their learning process. To address the gap between old generation learning management systems and the current needs in academia, CourseNetworking is built with a commonly accepted approach in social networking platforms. Jafari in [41] indicates that CourseNeworking's approach is to create an environment that is rewarding, engaging and entertaining for the current generation.

The CourseNetworking (CN) platform is an integration between traditional learning management systems and social networking environments. As it is shown in Figure 3.1, the users have access to CN through the cloud. The users can use any device that is connected to the internet to use CN's functionalities. CN's learning model includes unique features that are explained as follows:

- Social Network: CourseNetworking offers a global social environment for users with similar fields of study to collaborate. It enables the users to use CN's tool to share their desired content and interact with the other users.

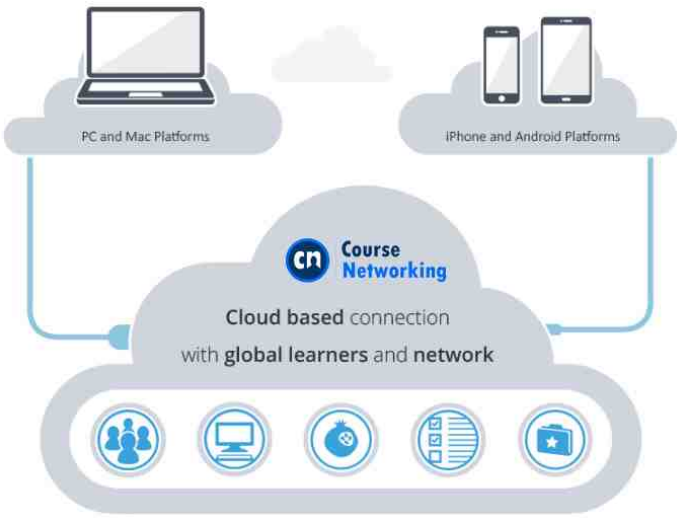


Fig. 3.1. CourseNetworking, an academic social networking site [43]

- Common LMS Tools: CourseNetworking's platform also covers most of the learning management systems' tools. It includes tasks (modules), assignments, quizzes, grade-book, files, roster, calendar, and so forth.
- Anar Seeds: CN's rewarding system enables the learning platform to be more engaging and enjoyable. The users will collect Anar seeds (pomegranate seeds in Persian) for a variety of reasons such as: finishing an assignment, posting, interacting on others' posts, and so forth.
- Task Tool: Task tool is one of the most sophisticated tools that enables the instructors to create learning schemes and activities based on their needs. It allows the instructors to design an outline for their course. For instance, a teacher can assign an assignment to the students, use other learning tools, attach a file, and monitor the students' interaction.
- Lifelong Access: CourseNetworking give the users a lifelong profile. The users can access their content even if they graduate or move to another institution.

CourseNetworking as a student-centered learning system provides a set of tools for the users to interact with each other. As CN's focus is to create a global learning platform in the educational area, its functionalities operate socially. Some of the main characteristics of CN as a social learning system are as follows:

- ePortfolio: CN allows the users to create their ePortfolio within the CN platform. The users have life-long access to their ePortfolios with a unique ID. Section 3.2 discusses the ePortfolios comprehensively.

- Posts: The users in CN use posts feature to write and share their desired content with the other users. The posts can contain title, content, links, attachments, images, YouTube videos, and SCORM packages. The users can control the visibility settings of the posts and the related keywords. The posts can also be attached to the users' profiles.

- Polls: This feature can be used to create a question in CN. The poll types can be multiple choice, short answer, true or false, or scale 1-5. The users can also define the correct answer for the given poll.

- Events: The users can create an event in CN to set up a meeting or a group activity. The users can use other content like images and attachments in the events.

- Social Interactions: The users can interact with the other users within the CN platform. They can like a post, leave a comment on a post, follow other users, or chat with them.

- Hashtags: The hashtags are associated with keywords with social activities. The users can use both predefined hashtags within the courses or networks that they are enrolled in, or define their hashtags. This allows the users to create and follow a certain topic throughout the CN platform.

3.2 ePortfolio

CourseNetworking like most of the social networking websites such as Facebook, Twitter, and LinkedIn allows the users to create their profile. The user profiles in the CN platform are specifically designed for educational purposes, where the users can share their resume, showcases, files, experience, and skills.

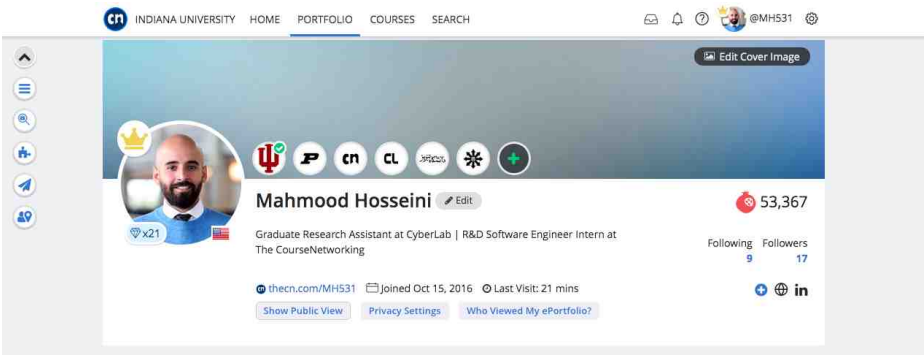


Fig. 3.2. ePortfolio in CourseNetworking

Figure 3.2 shows an example of an ePortfolio in CN. The users can add the logos of the institutions that they are involved in. The institutions can verify the users, and the verified institution logs will have a green check-mark on their logo.

The ePortfolio shows the number of Anar seeds that the user has achieved in CN. The side-bar menu has different functionalities which are as follows:

- Menu: The menu button allows visitors to navigate through the profile.
- Job Search: This enables the users to search for jobs in their field of study and location in the Indeed platform.
- Versions: The users can create different versions of their ePortfolio. This allows the users to emphasize their strength in certain areas based on the jobs that they are looking for.
- Share: The users can share their ePortfolio on other social media, with an email, or with a unique link.

- Visitor Tracking: This feature tracks interactions of the visitors. The users can track which aspect the visitors viewed in their profiles. It stores the visitors' internet protocol (IP) and their location.

Fig. 3.3. Skills in CourseNetworking ePortfolio

As Figure 3.3 shows, the users can add their skills to their ePortfolio. This unique feature allows the users to pick from the predefined skills in the CN system or create their own. The users can see the other users who use the same skills, which enables them to collaborate globally with other users.

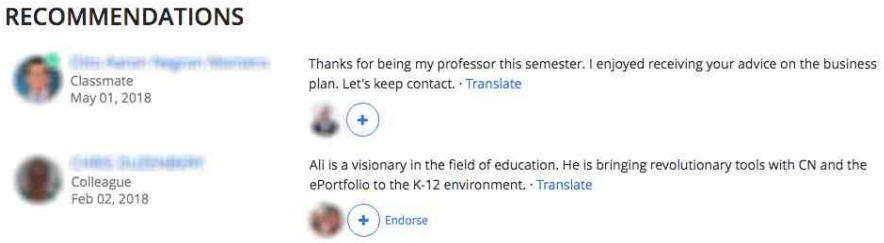


Fig. 3.4. Compliments/Recommendations in CourseNetworking

Figure 3.4 shows an example for compliments that the users can write on other users' ePortfolio. The current user's followers can also endorse compliments.

3.3 Institutions

Institutions that purchase the CN platform license and use it as their main learning management system have access to CN channel. CN channel is the institution admin panel that offers several tools and features that allow the institutions to manage their learning environment.

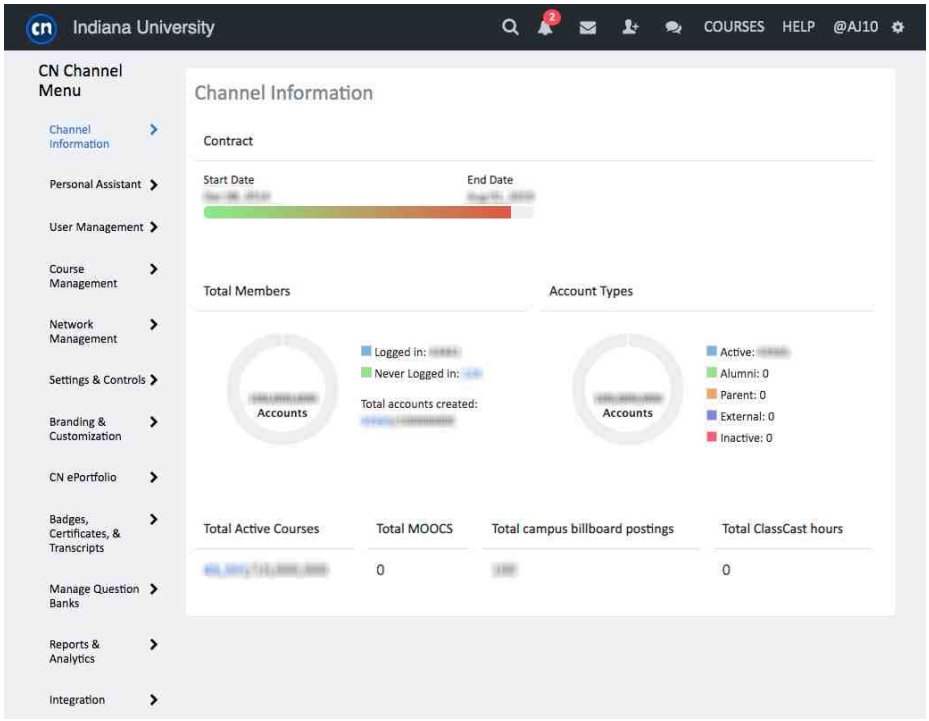


Fig. 3.5. Institution Administration (CN Channel) in CourseNetworking

The CN channel allows organizations to use all of CourseNetworking's functionalitie and offers the following features:

- Channel Information: This section provides basic information about the institution. It includes start and end date of the license, the number of the total, active, logged in users, the number of the active courses, and total campus billboard posts.

- Personal Assistant: The institutions can control the personal assistant within CN. The institutions can also use the announcement engine to interact with their users. This is discussed in Chapter 4.

- User Management: This tool enables organizations to create, edit, and delete users. The users can be imported from other learning platforms and exported as a standard CSV file. The institutions can create email templates and send

emails. This feature offers additional labels to help the institutions to organize members such as categorizing the members based on the schools and the departments.

- Course Management: The institutions can use this tool to create, import and export their courses. The users can be assigned to different courses.
- Network Management: It enables the organizations to create, import and export CN networks.
- Settings and Controls: This tool offers a variety of controls for the institutions such as controlling the course access, course invitation, viewing sharing public posts, enabling the users to change their names, calendar, and LTI settings.
- Branding and Customization: CN allows the organizations to customize the look and feel of their learning system. The institutions can edit the logo, header name, fonts, and colors.
- Badges, Certificates, and Transcripts: This tool offers the institutions to create their badges. The badges can be used within the courses in an institution. The certificates can be used to on user ePortfolios. The users who accomplish a task or finish a course can obtain a certificate. The organizations can upload and export the transcripts for their members.
- CN ePortfolio: This tool can be used to change the certified institution logo that appears on the members' profile. The institutions can create predefined skill tags and showcases. The ePortfolio sections can be managed by this tool.
- Manage Question Banks: This tool can be used to create and manage question banks. The instructors can use the questions banks to create quizzed in their courses. The admins can import IMS global's standard question and test interoperability (QTI) packages [44].

- Reports and Analytics: This feature offers a variety of reporting systems such as institution reports, ePortfolio reports for skills, badges, showcases, and recommendations, course reports, students' information and Anar reports, and instructor reports. The institutions can view previous reports and create new reports.

- Integration: CN allows the institutions to integrate other platforms to CN by using RESTful APIs. The consumer key and secret keys are assigned to the organizations. CN is integrated with lightweight directory access protocol (LDAP) and Google, and the institutions can use this feature to set up their authentication service by using the single sign-on method. Other external tools can be added by using this tool. CN can also be used as an external learning tools interoperability (LTI) package in other learning systems. A former Cyberlab graduate assistant, M. AboualizadehBehbahani, has created CN post, which offers some of CN's features as CN's LTI external package [45]. Any learning system that supports LTI 1.0 standard can add CN post to their environment. CN post connects the users who use the traditional learning management systems to a social learning environment. The users can use CN's global courses within their host LMS and connect to other users throughout the world.

4. DESIGN AND DEVELOPMENT

4.1 Introduction

The intention behind this project is to research, design and develop an online intelligent agent that helps the students and instructor in their academic life. As it is discussed in the previous chapters, the focus of the intelligent agents in the learning system is to create an environment that is beneficial for the users. The proposed personal assistant has a character that the users can feel connected to. As it is illustrated in Section 2.2, there are 9 important factors that an intelligent agent can have. The personal assistant in this project is created to be reactive. It can extract data, learn from data and user interactions, and perform autonomously. The agent collaborates with other learning systems like CourseNetworking. It consistently interacts with users over time. Machine learning techniques allow the personal assistant to be adaptive and learn from user interactions. Other learning systems can integrate this project with their system and use it within their platform.

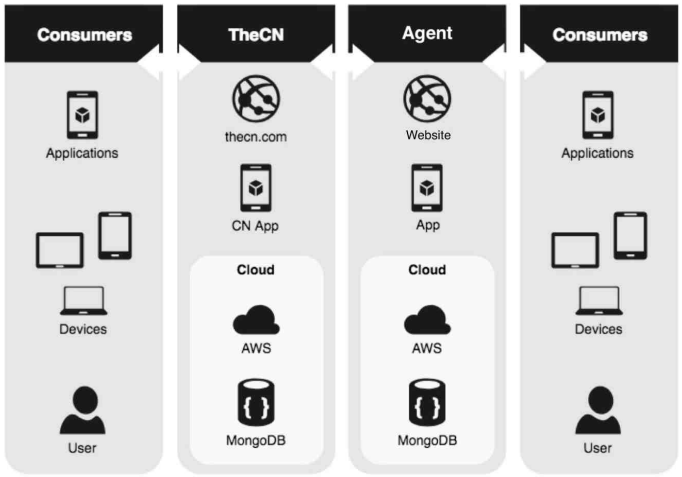


Fig. 4.1. Integration with CourseNetworking

This project as an independent software can be integrated with any learning system that provides adequate data for machine learning techniques. As Figure 4.1 shows, this agent can be accessed through its website. This enables the users to use the agent's functionalities through their learning management system like CN or independently. The proposed software has its database and runs on Amazon web services (AWS) cloud platform.

4.2 Architecture

This project like other modern web platforms has a database, a back-end, and a front-end. The software design follows model-controller-view (MVC) architecture.

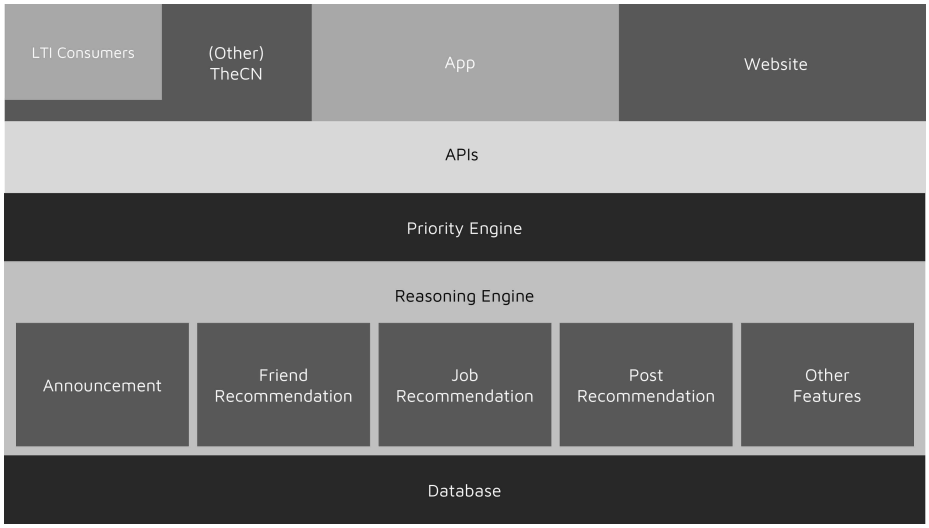


Fig. 4.2. Architecture

The users can access the proposed system through its website or within the other platforms like CourseNetworking. Other LTI consumers can use the personal assistant through their connection with the host platform. The recommendations and announcements are computed and handled in Reasoning engine, which is fully described in Section 4.2.1. Priority engine is the decision maker component of this project, where it prioritizes the features and decides what message should the user

see first. Section 4.2.2 comprehensively explains the priority engine. Section 4.2.3 demonstrate the architecture of the database. Section 4.2.4 explains how the system's APIs work. In Section 4.3, the integration process of this personal assistant with a learning management system is explained. The proposed prototype's backend is written in PHP 7.1 and uses Laravel framework to support MVC architecture. Docker containers are used for development and testing purposes, which allow the researchers and the developers to see the project from the same point of view and eliminates the potential cross-platform errors.

4.2.1 Reasoning Engine

Reasoning engine is responsible for collecting data, analyzing and returning content to the next layer. Each of the features compute contents for the current user. The features are described in 4.4. Reasoning engine includes a set of controllers which are responsible for computing the required response for each API request.

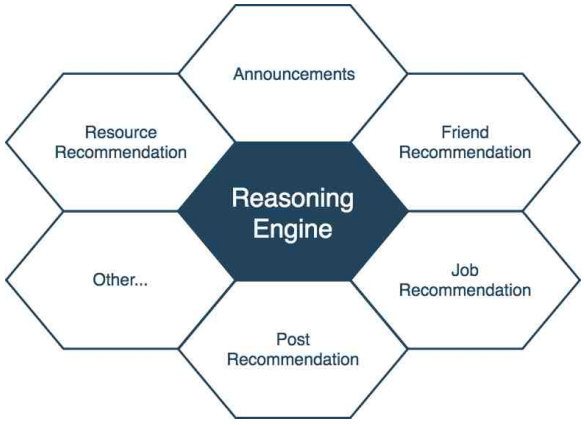


Fig. 4.3. Features

4.2.2 Priority Engine

This layer controls the importance of each content. When a user logs into the learning platform, the priority engine is responsible for deciding what content has the highest priority. The system creates a vector for each user that contains the priority of all the features or the given user. The default values are all the same, and the announcements that come from the institutions have the highest priority. This allows the institutions to make sure that their users see the announcements.

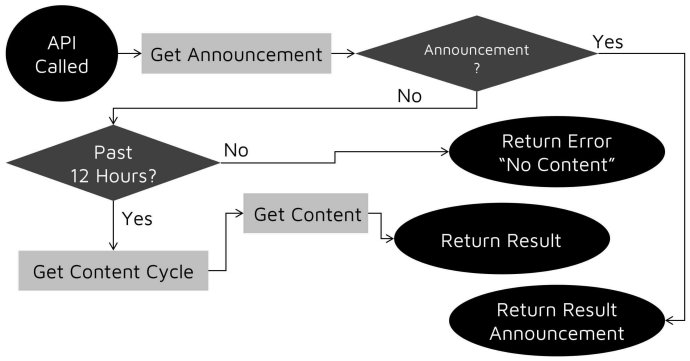


Fig. 4.4. Priority Engine Flowchart

As Figure 4.4 shows, once an API is requested from the learning management system, the agent returns the announcements if there is any. If not, and if it is past 12 hours from the last time that the personal assistant has communicated with the user, it will return a useful content based on the priorities. A 12-hour time span is the default value between two different messages. This allows the personal assistant to be connected to the users. Institutions can change the default value. Based on the content priority vector, this project creates a cycle for each user. Within the given cycle, the contents will be sorted based on their priority and return each of them at each API call. There are three specific fields in the users collection which store the data for Priority engine:

- Priorities: The priorities of all of the features for each user.
- Priority_cycle: Sorted list of the features based on their priorities.

- Priority_next_index: Index of the next feature in the priority_cycle list.

For instance, for a user the above-mentioned fields can look like:

```
"priorities" : {
    "friend_recommendation" : 0.85,
    "job_recommendation" : 0.8
},
"priority_cycle" : [
    "friend_recommendation",
    "job_recommendation"
],
"priority_next_index" : NumberInt(1)
```

Listing 4.1 An example of Priority engine fields in the database

Where the user has more interest in friend recommendations than job recommendations. The user interactions will have impact on the priority of each feature with a 0.1 learning rate.

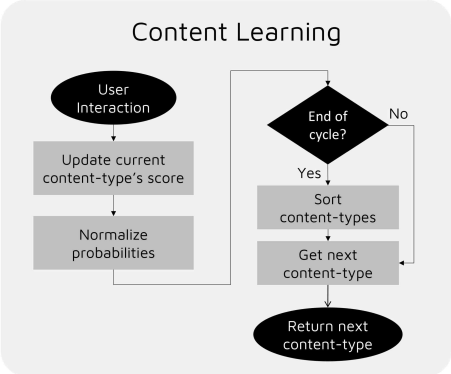
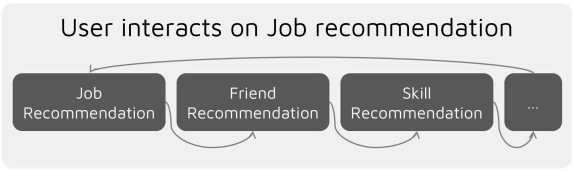


Fig. 4.5. Content Learning

Figure 4.5 shows the flowchart of the learning process. Once a user interacts with any of the recommendations, the priority of it increases. For instance, if the user in 4.1 viewed a recommended job, the priority of the job recommendation feature will become 0.88 in the next cycle.

(a) A 12 Hour Content Cycle



(b) Updated Content Cycle Based on User Interaction on Job Recommendation

Fig. 4.6. Content Cycle Before and After User Interaction

At the end of each cycle, the features are sorted based on their priorities. The learned values based on user preferences show their impact in this stage. The user interaction increases the priority of a job recommendation, and it becomes the most important feature in the next cycle.

4.2.3 Database

The proposed agent uses MongoDB as its database management system (DBMS) for several reasons. First, to keep the personal assistant's database consistent with CourseNetworking. Second, NoSQL databases perform better in social networking websites [46]. Third, MongoDB stores data in JSON documents format that makes RESTful API support easier. The database contains several collections:

Users The schema includes:

- Cn_id: Unique user identifier in the CN platform.
- Institution_id: ID of the institution that the user belongs to.
- Role: Role of the user.

- Cn_number: Unique string assigned to the user in the CN platform.
- View_count: Number of the times that the user has viewed a message from the personal assistant.
- Job_recommendation_config: Configuration of the job recommendation feature for this user.
- Friend_recommendation_config: Configuration of the friend recommendation feature for this user.
- Priorities: Priorities of the features.
- Last_view: Timestamp of the last time that the user has seen a message from the agent.
- Priority_cycle: Sorted array of the features.
- Priority_next_index: Index of the next feature in the priority_cycle array.
- Deleted_friend_recommendations: Array of IDs of the deleted friend recommendations.
- Visited_announcements: Array of IDs of the visited announcements.

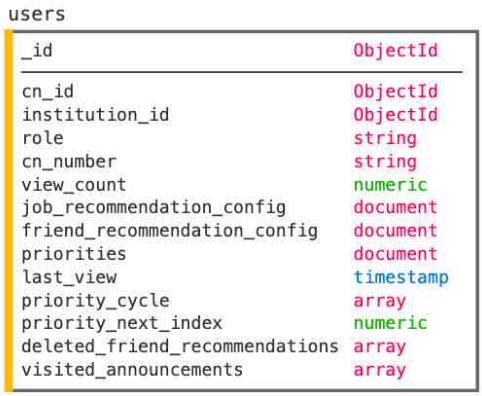


Fig. 4.7. Users Collection

Friend Recommendations Recommended friends' data for each user is stored in this collection.

- Cn_id: Unique user identifier in the CN platform.
- Recommended_cn_id: Recommended user's ID in CN.
- Score: Score of the recommendation.
- Causes: Array of the recommendation's reasons.
- User_user_skills: Array of the matching skills between the current user and the recommended user.

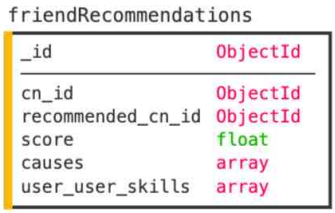


Fig. 4.8. Friend Recommendations Collection

Institutions The agent uses this collection to store the institutions' settings.

- Institution_id: Unique institution identifier in the CN platform.
- Settings: Document of the features with a binary value for each that shows if the feature is enabled.

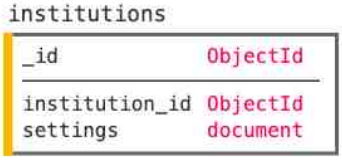


Fig. 4.9. Institutions Collection

User Interactions This is the collection of user interactions that contains data for job views, feedback, and interactions with friend recommendations.

- Cn_id: Unique user identifier in the CN platform.
- Interaction_type: Type of the interaction. This field can contain one of $follow\_user$ or $delete\_user$ values, and defines another field based on its value.
- Followed_cn_id*: If the value of interaction_type is $follow\_user$, this field will contain CN ID of the followed user.
- Deleted_cn_id*: If the value of interaction_type is $delete\_user$, this field will contain CN ID of the deleted friend recommendation.

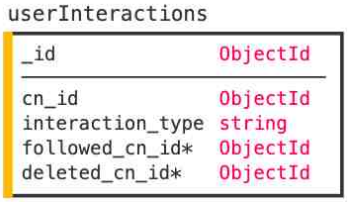


Fig. 4.10. User Interactions Collection

Announcements The institution-generated announcements are stored in this collection.

- Institution_id: Unique institution identifier in the CN platform.
- Institution_name: Name of the institution.
- Logo_url: URL of the institution's logo.
- Title: Title of the announcement.
- Body: Main text of the announcement.

- Start_time: Timestamp that stores start time of the announcement. The users who log in before this time will not see this announcement.
- End_time: Timestamp that stores end time of the announcement. The users who log in after this time will not see this announcement. If the announcement does not expire, this field is left empty.
- Ctime: Timestamp of the creation time of the announcement.
- Audience: Array of the audience of the announcement. This field can contain one or more values of *admin*, *instructor*, and *student*.
- Priority: The importance of the announcement in the scale of (1 Low priority - 10 High priority).

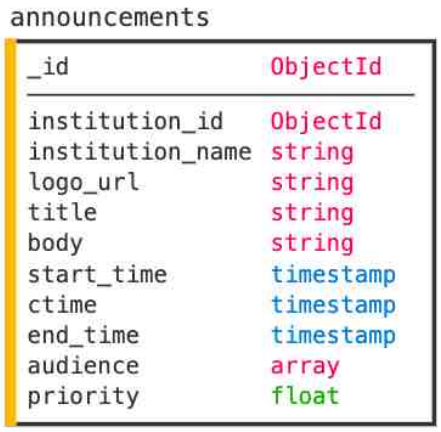


Fig. 4.11. Announcements Collection

Skills This collection is created to store the user IDs which use the same skills. A document is created for each skill, and stores the user IDs for the given skill. This enables the agent to compute skill matching with a higher performance.

4.2.4 APIs

The agent supports RESTful API architecture to perform high-quality data transmission. The online learning systems can use the APIs to have access to the agent's data and computational components. To use the APIs, the user must have a token from the host learning system (CourseNetworking).

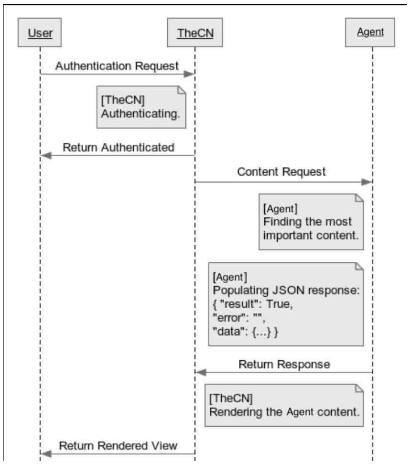


Fig. 4.12. CourseNetworking Using an API

In this project, every request API request goes through a middle-ware. This enables the system to filter the requests before computing the features. The $CNHandshake$ middle-ware is defined in this project that allows the agent to validate if the requests are coming from an authenticated user. As Figure 4.12 shows, first, the user authenticates with CN. Then, CN sends an API request to the personal assistant to get the most important content. The request has a unique 32 character token set in its header. In the $CNHandshake$ middle-ware, the agent validates the given token with CN. If the given token is valid for the given user, the API request will be routed through the software. Once the API request is routed to the specified controller, the agent creates and returns a standard JSON response to CN. The content view within the CN platform will be rendered after CN receives the API response. Following Algorithm 1 shows the validation process of an API request before it being handled.

Algorithm 1: CNHandshake Middle-ware

```
Receive API request from CN:
    if token is not set then
        Response Error.
    Request Validate token with CN:
        if token is not valid then
            Response Invalid token.
Handle set a session for the user
```

Following Table 4.1 shows the APIs and their available methods. All of the APIs go through the middle-ware before entering the software. As in every API request token of the current user is set, the agent can access the user data based on the given token.

Table 4.1.
APIs

API	Description	Methods
getCNOne	The most important message	GET
announcement	Institution announcements	GET
recommendFriends	Recommended friends	GET
recommendFriend	Recall recommendFriends (limit 1)	GET
recommendJobs	Returns the recommended jobs	GET
recommendJob	Recall recommendJobs (limit 1)	GET
deleteFriendRecommendation	Delete friend recommendation	POST
userInteration	Store user interaction	POST
j	Job view	GET
admin/announcement	View/create/edit announcements	GET,POST
admin/settings	Edit settings	POST

The deleteFriendRecommendation API requires a *deleted_cn_id* parameter, which must include a valid CN ID for a user. The userInteraction API requires a *interaction−type* parameter which identifies type of the user interaction. This API can contain more fields based on the interaction type.

4.3 Integration

As it is mentioned in Table 4.1, the institution can use the personal assistant's APIs to integrate it with their learning system. This project provides a set of functionalities that the channels (institutions) can control. The institution admins can enable/disable any of the features in their system. The institutions can manage the announcements within their system. It allows the institution admins to view, delete, or edit previously created announcements, or create a new announcement.

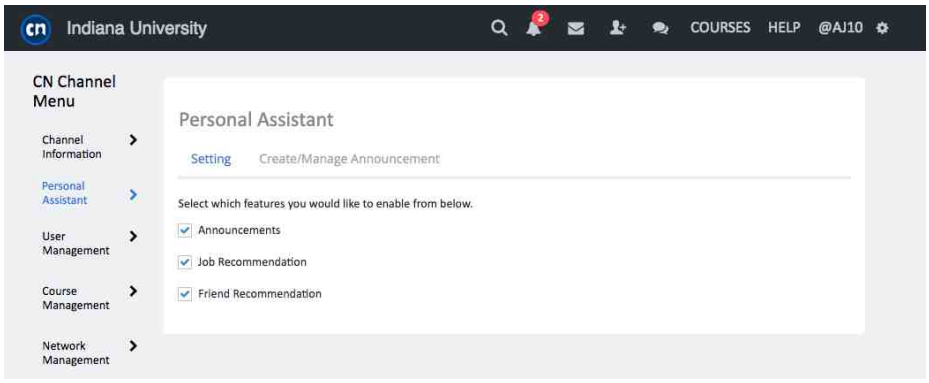


Fig. 4.13. Admin

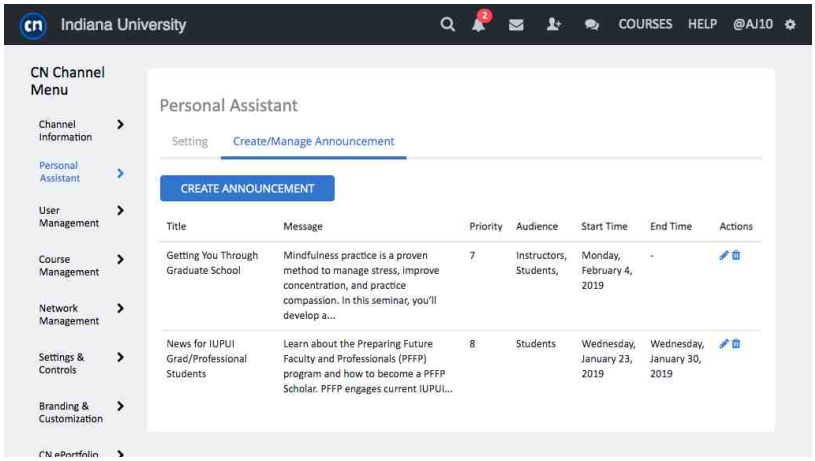


Fig. 4.14. Admin - Announcements

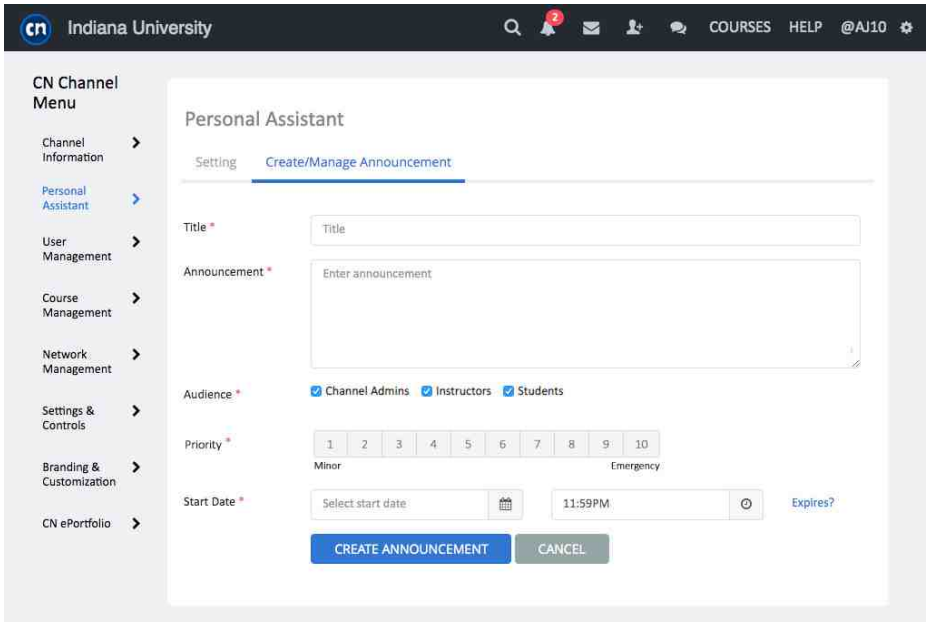


Fig. 4.15. Admin - Create an Announcement

As Figure 4.15 shows, channel admins can create the announcements through the admin panel. The channel admins can control title, announcement text, the audience of the announcement (Channel Admins/Instructors/Students), the priority of the announcement(1 Low priority/10 High priority), start date, and end date (optional) in the announcement creation panel. If the end date of an announcement is not defined, the audience of the announcement will see the message in their first visit after the start date.

4.4 Features

The Figure 4.3 shows that the proposed agent provides several features. The purpose of each of the features is to provide a meaningful and helpful message for the user. Each feature can use the data which is available from the integrated learning platform's system and construct a robust model that helps the users in their academic journey. As it is shown in Figure 4.3, the personal assistant uses the main banner location in the CN platform to communicate with the users.

4.4.1 Announcements

Announcements are the core elements of the relationship between institutions and their students. Institutions can use the announcements feature to deliver their important messages to the students and the instructors. This allows the institutions to keep their users notified with the latest notifications and keep track of the users that see the announcements.

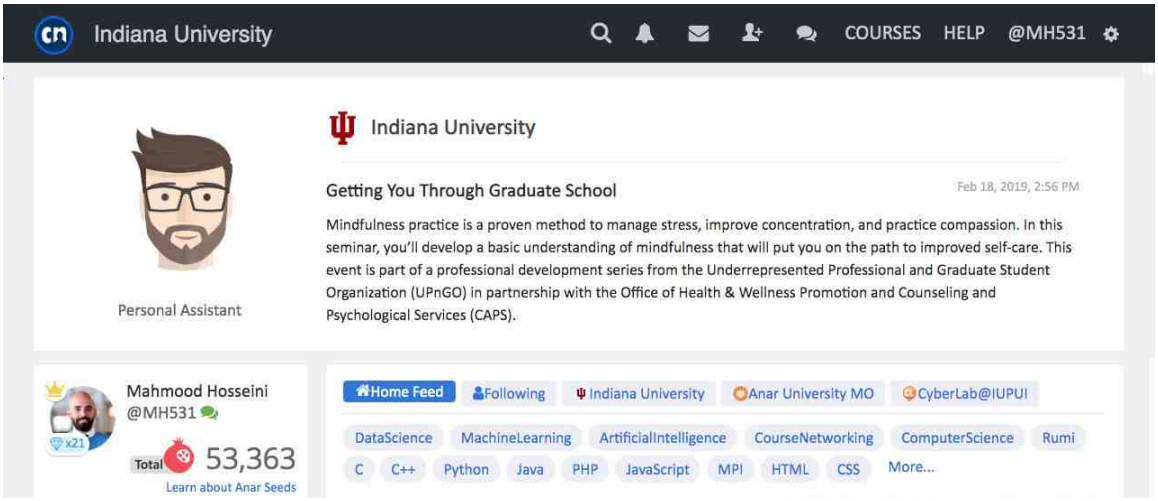


Fig. 4.16. Announcements

4.4.2 Job Recommendation

As it is mentioned in Chapter 3, the users can create their portfolio within the CourseNetworking platform. Skills, major and location of the users are the primary data that the agent uses for job recommendations. This is done by integrating the proposed system with Indeed through its application programming interface (API). The API requests and responses are handled with the standard JSON format. The major interests and the location of the user are selected and sent as the parameters of an API request to Indeed. The default number of job recommendations in CN is 3, but it can be changed in the configurations.

Algorithm 2: Job Recommendation

if *job settings is not set* **then**
  if *field of study is set* **then**
    $KEYWORDS$ = keywords from field of study.
  else
    $KEYWORDS$ = 5 most used skills of the user.
  $LOCATION$ = current city and state.
  Store $KEYWORDS$ and $LOCATION$ as job settings.
Request *jobs from Indeed*:
  if *if returned jobs* **then**
    Response *jobs from Indeed*


```
{
    "cn_id": "4f67936a91d408bf2a000002",
    "recommendations": {
        "77c95f2a908103b3": {
            "name": "Communication Intern",
            "description": "The Communication Intern...",
            "company_name": "Indiana University",
            "url": "www.indeed.com/viewjob?jk=77c95f2a908103b3",
            "source": "Indiana University",
            "address": "Indianapolis, IN",
            "date_posted": "Thu, 28 Mar 2019 19:24:28 GMT",
            "formatted_data": "21 days ago"
        }
    },
    "content_type": "job_recommendations",
    "face_type": "neutral"
}
```


Listing 4.2 An example of a job recommendation API response

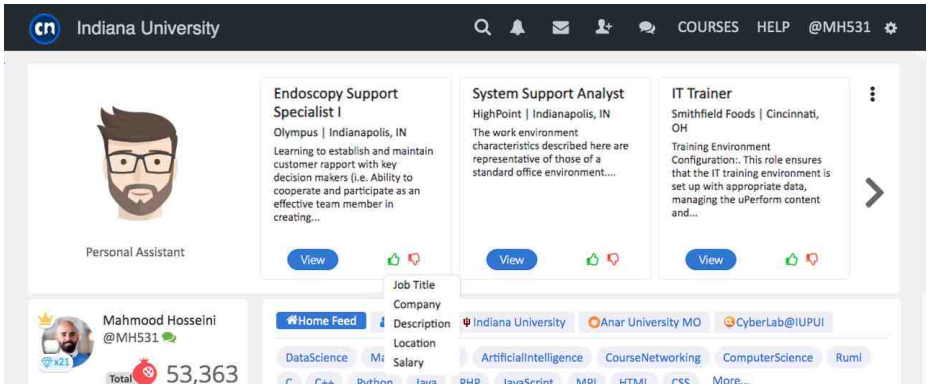


Fig. 4.17. Job Recommendation

4.4.3 Friend Recommendation

Friend recommendation is currently the most sophisticated feature of the proposed system. The agent recommends potential connections within the learning system and enables the users to connect to the other users. The connections can be in each user's interest from different aspects. For instance, for a user who wants to connect with his/her classmates, the agent will learn from their actions and recommend more people with the same characteristics. For the users who would be interested in global collaboration with the users who share similar skills, recommended friends will be related to the user's skill-set. The default number of recommended friends is 3.

Link recommendation is one of the most important features of modern social media websites [27]. CourseNetworking as a social learning system encourages its users to collaborate in global classrooms. This creates a unique space for the agent to recommend friends to the users and help them grow their network. The connections network in CN is a directed graph. The users can follow other users, be followed, or both.

Figure 4.19 shows the connections in the CN platform. The solid black links show the existing connections and the gray dashed links show the potential connections. This feature's goal is to find the best potential connections that a user might be interested in considering it. To develop a friend recommender system, this project

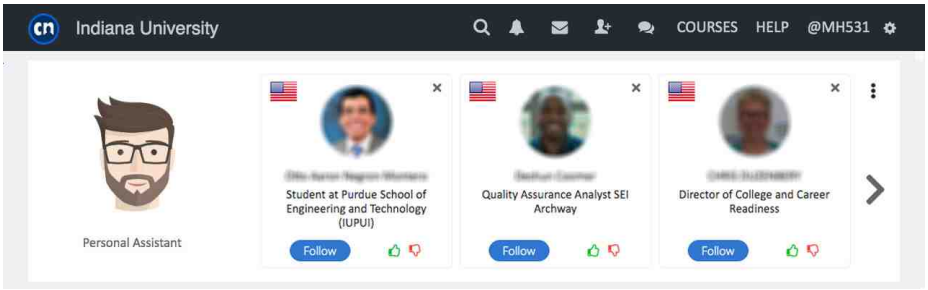


Fig. 4.18. Friend Recommendation

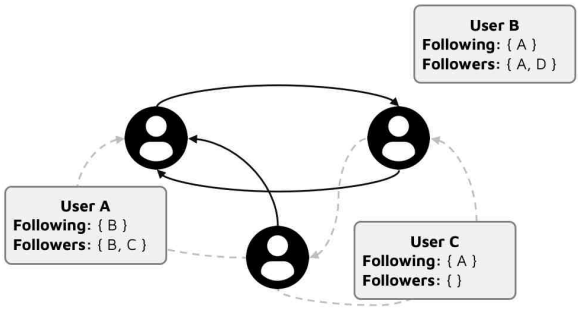


Fig. 4.19. User connection network in CourseNetworking

uses a hybrid method that uses both nodal proximity or structural proximity features. This is done by creating a vector of for each user, which includes the probabilities of the user following another user, given the corresponding feature. For instance, if an arbitrary user A has written a recommendation within CourseNetworking platform for another arbitrary user B, the vector for user A will contain a value for that feature. As following Equation 4.1 shows, the value of each attribute is the direct cause of the link likelihood of a link based on having that attribute. Where v_i is the value of feature i in each vector. It contains normalized value of the probability of a user following another user (h) with given the effect (e_i) as a feature.

$$v_i = P(h|e_i) \tag{4.1}$$

The features are assumed to be independent from each other. The probability values are exported from CN, and the features are picked based on their value. The features that have smaller normalized values than 0.01 are assumed to be insignificant based on their low impact on the likelihood of a linkage. The selected features of each vector:

- Wrote recommendation: If the current user A wrote a compliment / recommendation for user B.
- Got recommended: If user A got a recommendation from user B.
- Visited profile: If user A has visited user B's profile.
- Got visited: If user A's profile got visited by user B.
- Colleague: If user A and user B were colleagues in a course or a network.
- Follow back: If user B follows user A, but user A doesn't follow user B back.
- Followings of follower: If user B is following one or more of user A's followings.
- Followings of following: If user B is followed by one or more of user A's followings.
- Skill match: Number if matching skills between user A and user B.

Following Algorithm 3 is used to extract probabilities of the features, and Table 4.2 shows the normalized value of each exported feature. Equation 4.2 is used to calculate the score of user A following user B. Where S_{AB} is sum of the values of existing features F between them user A and user B.

$$S_{AB} = \sum_{i \in F} f_i \tag{4.2}$$

Algorithm 3: Probability Values Extraction for Friend Recommendation

Extract *existing links based on the features*:

visited_following = links where user A has visited user B
visitor_following = links where user B has visited user A
recommended_following = links where user A has recommended user B
got_recommended_following = links where user B has recommended user A
colleague_following = links where user A and user B were colleagues
skills_following = links where user A and user B have common skills
follow_back = followers of A where A does not follow them
`/* if B follows A but A does not follows B, then B is in this array */`
following_followers = followers of users that current user follows
`/* if A follows B and C follows B, then C is in this array */`
following_following = followings of users that current user follows
`/* if A follows B and B follows C, then C is in this array */`

Calculate *probabilities of the features*:

prob_skill_match = skills_following / total_links
prob_visited_profile = visited_following / total_links
prob_got_visited = visitor_following / total_links
prob_colleague = colleague_following / total_links
prob_wrote_recommendation = recommended_following / total_links
prob_got_recommended = got_recommended_following / total_links
prob_follow_back = follow_back / total_links
prob_following_following = following_following / total_links
prob_following_follower = following_followers / total_links

Table 4.2.
Friend Recommendation Vector Features

Feature	**Value**
Skill match	0.05
Visited profile	0.06
Got visited	0.01
Colleague	0.13
Wrote recommendation	0.26
Got recommended	0.26
Follow back	0.06
Followings of following	0.06
Followings of follower	0.06
Total	**1**

5. CONCLUSION AND FUTURE WORK

This research project proposes design and development of an intelligent online personal assistant for a social learning system. The proposed personal assistant is integrated with an existing learning management system, CourseNetworking. It allows the institutions to communicate with their members using a human-like persona. This agent is integrated with the most used job searching website, Indeed, to help the users to find their desired jobs. A hybrid link recommender system is developed that helps the members to find other friends and connect with other users within the platform. This project as an intelligent software platform provides the infrastructure for other researchers to contribute to the current system. The users' needs in social learning systems are constantly changing, and the available data can be used to make the current system better. Following features can be added to the proposed agent's functionalities:

- Post Recommendation [25]: The proposed post recommender system can be used in this project. The standard way that the proposed personal assistant interacts with each member creates the opportunity for the other users' posts to be seen by other users.

- Resource Recommendation: The growth of open and free resources provides accessible data for the researchers to apply natural language processing (NLP) techniques on their content and recommend them to the users. This will enable the members to find materials in their field of study.

- Certificate, Degree, and Credential Recommendation: The students who are looking for a career path can benefit from this feature. This agent can recommend useful information about the existing certificates and degrees based on the users' preferences.

- Skill Recommendation: By learning the patterns about the skill-set that certain careers or job descriptions require, the personal assistant can recommend useful skills which a user can consider. For instance, for computer science students who are looking for software engineering jobs and do not have some of the most important programming languages like Java in their skill-set, this agent can recommend the missing skills.

- Entertainment: Next generation of the learning systems can provide entertainment tools. It will enable the users to feel more connected to their personal assistant and will help them enjoy their learning experience.

Figure 4.17 shows that the proposed personal assistant will show job title, location and the description of the recommended jobs to the user, and there are several ways that the user can interact with the recommended jobs:

- View: View the recommended job.

- Feedback: Positive or negative feedback for the recommended jobs can be specified for the following characteristics of the job.

  - Job title
  - Company
  - Description
  - Location
  - Salary

In the future, user interactions help the system to learn about the users' preferences. As the search query for each recommendation are stored in the database, user interactions can have an impact in the future queries. This helps the users to find their desired jobs based on their interest and proffered location. Figure 4.18 shows that there are 4 options for the user to interact with the given friend recommendations:

- Follow: Current user follows the recommended user.

- Delete: Deletes the recommended user from recommendation database. The deleted users are avoided in the recommendation process.

- Thumbs Up: A good feedback about the recommendation.
- Thumbs Down: A bad feedback about the recommendation.

In the future, a naïve Bayesian network model can be created for each user that learns based on the users' interactions with each friend recommendation. It can allow the personal assistant to make recommendations that are biased towards the users' preferences.

In addition, to include the time parameter in the learning process, a modification can be made in the algorithm. It will allow the users' latest interactions to have more impact on the learning process. In [47] authors illustrate that media causes changes in social networks. Social mood of a user can change over time, and adding a time factor in the learning procedure can help the recommendations to adapt the users' behavior.

REFERENCES

REFERENCES

[1] "Canalys newsroom," https://www.canalys.com/newsroom/amazon-reclaims-top-spot-in-smart-speaker-market-in-q3-2018, [online] Last Date Accessed: 2019-02-20.

[2] "Gartner," https://www.gartner.com/en/newsroom/press-releases/2016-10-03-gartner-says-worldwide-spending-on-vpa-enabled-wireless-speakers-will-top-2-billion-by-2020, [online] Last Date Accessed: 2019-02-20.

[3] C. C. Aggarwal *et al.*, *Recommender systems*. Springer, 2016.

[4] D. Goldberg, D. Nichols, B. M. Oki, and D. Terry, "Using collaborative filtering to weave an information tapestry," *Communications of the ACM*, vol. 35, no. 12, pp. 61–71, 1992.

[5] K. Goldberg, T. Roeder, D. Gupta, and C. Perkins, "Eigentaste: A constant time collaborative filtering algorithm," *information retrieval*, vol. 4, no. 2, pp. 133–151, 2001.

[6] B. N. Miller, J. A. Konstan, and J. Riedl, "Pocketlens: Toward a personal recommender system," *ACM Transactions on Information Systems (TOIS)*, vol. 22, no. 3, pp. 437–476, 2004.

[7] G. Linden, B. Smith, and J. York, "Amazon. com recommendations: Item-to-item collaborative filtering," *IEEE Internet computing*, no. 1, pp. 76–80, 2003.

[8] T. Hofmann, "Latent semantic models for collaborative filtering," *ACM Transactions on Information Systems (TOIS)*, vol. 22, no. 1, pp. 89–115, 2004.

[9] G. Adomavicius and J. Zhang, "Stability of collaborative filtering recommendation algorithms," *citeseer, doi*, vol. 10, no. 1.221, p. 7584, 2012.

[10] X. Li, C. X. Ling, and H. Wang, "The convergence behavior of naive bayes on large sparse datasets," *ACM Transactions on Knowledge Discovery from Data (TKDD)*, vol. 11, no. 1, p. 10, 2016.

[11] A. Sharma, N. Mehta, and I. Sharma, "Reasoning with missing values in multi attribute datasets," *International Journal of Advanced Research in Computer Science and Software Engineering*, vol. 3, no. 5, 2013.

[12] J. S. Breese, D. Heckerman, and C. Kadie, "Empirical analysis of predictive algorithms for collaborative filtering," in *Proceedings of the Fourteenth conference on Uncertainty in artificial intelligence*. Morgan Kaufmann Publishers Inc., 1998, pp. 43–52.

[13] G. Adomavicius and A. Tuzhilin, "Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions," *IEEE Transactions on Knowledge & Data Engineering*, no. 6, pp. 734–749, 2005.

[14] D. Billsus and M. J. Pazzani, "Learning collaborative information filters." in *Icml*, vol. 98, 1998, pp. 46–54.

[15] B. Sarwar, G. Karypis, J. Konstan, J. Riedl *et al.*, "Analysis of recommendation algorithms for e-commerce," in *EC*, 2000, pp. 158–167.

[16] P. Melville, R. J. Mooney, and R. Nagarajan, "Content-boosted collaborative filtering for improved recommendations," *Aaai/iaai*, vol. 23, pp. 187–192, 2002.

[17] C.-N. Ziegler, G. Lausen, and L. Schmidt-Thieme, "Taxonomy-driven computation of product recommendations," in *Proceedings of the thirteenth ACM international conference on Information and knowledge management*. ACM, 2004, pp. 406–415.

[18] B. M. Kim and Q. Li, "Probabilistic model estimation for collaborative filtering based on items attributes," in *Proceedings of the 2004 IEEE/WIC/ACM international conference on web intelligence*. IEEE Computer Society, 2004, pp. 185–191.

[19] K. Miyahara and M. J. Pazzani, "Improvement of collaborative filtering with the simple bayesian classifier," *Information Processing Society of Japan*, vol. 43, no. 11, 2002.

[20] K. Miyahara and M. Pazzani, "Collaborative filtering with the simple bayesian classifier," in *Pacific Rim International conference on artificial intelligence*. Springer, 2000, pp. 679–689.

[21] R. J. Mooney and L. Roy, "Content-based book recommending using learning for text categorization," in *Proceedings of the fifth ACM conference on Digital libraries*. ACM, 2000, pp. 195–204.

[22] D. Agarwal and B.-C. Chen, "Regression-based latent factor models," in *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 2009, pp. 19–28.

[23] B. Kanagal, A. Ahmed, S. Pandey, V. Josifovski, J. Yuan, and L. Garcia-Pueyo, "Supercharging recommender systems using taxonomies for learning user purchase behavior," *Proceedings of the VLDB Endowment*, vol. 5, no. 10, pp. 956–967, 2012.

[24] Y. Zhang, A. Ahmed, V. Josifovski, and A. Smola, "Taxonomy discovery for personalized recommendation," in *Proceedings of the 7th ACM international conference on Web search and data mining*. ACM, 2014, pp. 243–252.

[25] M. Mirzaeibonehkhater, "Developing a dynamic recommendation system for personalizing educational content within an e-learning network." *Master's Thesis, Purdue University, Indiana University-Purdue University Indianapolis*, 2018.

[26] X. Zhou, Y. Xu, Y. Li, A. Josang, and C. Cox, "The state-of-the-art in personalized recommender systems for social networking," *Artificial Intelligence Review*, vol. 37, no. 2, pp. 119–132, 2012.

[27] T. H. Davenport and D. Patil, "Data scientist," *Harvard business review*, vol. 90, no. 5, pp. 70–76, 2012.

[28] D. Liben-Nowell and J. Kleinberg, "The link-prediction problem for social networks," *Journal of the American society for information science and technology*, vol. 58, no. 7, pp. 1019–1031, 2007.

[29] J. OMadadhain, D. Fisher, P. Smyth, S. White, and Y.-B. Boey, "Analysis and visualization of network data using jung," *Journal of Statistical Software*, vol. 10, no. 2, pp. 1–35, 2005.

[30] R. N. Lichtenwalter, J. T. Lussier, and N. V. Chawla, "New perspectives and methods in link prediction," in *Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 2010, pp. 243–252.

[31] C. Wang, V. Satuluri, and S. Parthasarathy, "Local probabilistic models for link prediction," in *Seventh IEEE international conference on data mining (ICDM 2007)*. IEEE, 2007, pp. 322–331.

[32] N. Benchettara, R. Kanawati, and C. Rouveirol, "Supervised machine learning applied to link prediction in bipartite social networks," in *2010 International Conference on Advances in Social Networks Analysis and Mining*. IEEE, 2010, pp. 326–330.

[33] N. Z. Gong, A. Talwalkar, L. Mackey, L. Huang, E. C. R. Shin, E. Stefanov, D. Song *et al.*, "Jointly predicting links and inferring attributes using a social-attribute network (san)," *arXiv preprint arXiv:1112.3265*, 2011.

[34] M. McPherson, L. Smith-Lovin, and J. M. Cook, "Birds of a feather: Homophily in social networks," *Annual review of sociology*, vol. 27, no. 1, pp. 415–444, 2001.

[35] D. Shen, J.-T. Sun, Q. Yang, and Z. Chen, "Latent friend mining from blog data," in *Sixth International Conference on Data Mining (ICDM'06)*. IEEE, 2006, pp. 552–561.

[36] N. Jennings, N. R. Jennings, and M. J. Wooldridge, *Agent technology: foundations, applications, and markets*. Springer Science & Business Media, 1998.

[37] A. Jafari, "Conceptualizing intelligent agents for teaching and learning," *Educause Quarterly*, vol. 25, no. 3, pp. 28–34, 2002.

[38] N. Negroponte, *Being digital*. Vintage, 1996.

[39] O. Etzioni and D. S. Weld, "Intelligent agents on the internet: Fact, fiction, and forecast," *IEEE expert*, vol. 10, no. 4, pp. 44–49, 1995.

[40] S. Franklin and A. Graesser, "Is it an agent, or just a program?: A taxonomy for autonomous agents," in *International Workshop on Agent Theories, Architectures, and Languages*. Springer, 1996, pp. 21–35.

[41] "Coursenetworking, white paper," https://www.thecn.com/aboutus, 2012, [online] Last Date Accessed: 2019-02-20.

[42] H.-T. Hung and S. C.-Y. Yuen, "Educational use of social networking technology in higher education," *Teaching in Higher Education*, vol. 15, no. 6, pp. 703–714, 2010.

[43] "Coursenetworking," https://www.thecn.com/, [online] Last Date Accessed: 2019-02-20.

[44] "Ims global learning consortium — better learning from better learning technology," http://www.imsglobal.org/, [online] Last Date Accessed: 2019-02-20.

[45] M. AboualizadehBehbahani, "Proposing a new system architecture for next generation learning environment." *Master's Thesis, Purdue University, Indiana University-Purdue University Indianapolis*, 2016.

[46] S. Kanoje, V. Powar, and D. Mukhopadhyay, "Using mongodb for social networking website deciphering the pros and cons," in *2015 International Conference on Innovations in Information, Embedded and Communication Systems (ICIIECS)*. IEEE, 2015, pp. 1–3.

[47] C. Haythornthwaite, "Social networks and internet connectivity effects," *Information, Community & Society*, vol. 8, no. 2, pp. 125–147, 2005.

[48] "W3techs - web technology surveys," https://w3techs.com/technologies/details/pl-php/all/all, [online] Last Date Accessed: 2019-02-20.

[49] "Google trends," https://trends.google.com/trends/explore?q=laravel,Symfony,CodeIgniter,CakePHP,Zend, [online] Last Date Accessed: 2019-02-20.

[50] "Docker," https://www.docker.com/, [online] Last Date Accessed: 2019-02-20.

[51] "What is docker? — opensource.com," https://opensource.com/resources/what-docker, [online] Last Date Accessed: 2019-02-20.

[52] "Indeed — https://www.indeed.com/about," https://www.indeed.com/about, [online] Last Date Accessed: 2019-02-20.

APPENDICES

A. BACKGROUND TECHNOLOGIES

A.1 PHP

Hypertext Preprocessor (PHP) is a programming language that is used for a variety of computer software. PHP is widely used in web development. It offers a command line interface (CLI) and can be integrated into HTML code. PHP is the most used programming language on the web which is used by 79% of the users [48].

A.2 Laravel

Laravel is a free, open-source, and the most popular framework written for PHP [49]. Laravel uses model-controller-view (MVC) method and allows the users to use other software packages within the framework.

A.3 Docker

Docker is a platform that eliminates the barrier of technology mismatches and enables organizations to build, develop and deploy their desired software. [50] By using docker containers, developers can design the application infrastructure, based on their needs. Containers can include a variety of technologies such as operations systems, back-end and front-end development languages, database management systems, libraries, and dependencies and so forth. As it is mentioned in [51], this allows the developers to develop the software as one package with their personal configurations.

A.3.1 Docker Containers

Containers are blocks of software that include the code, libraries and related materials. Each docker container is a stand-alone unit that can run on any computing structure. This allows the developers to have lightweight packages which include their desired tools and settings and can run reliably on any environment [50]. Docker containers are the abstraction of the software layer. Several containers can run on the same computer and share the operation system. Each container is a stand-alone package and requires less space than virtual machine packages. Containers are made from container images when they are being used in the Docker Engine. This makes the Dockerized (containerized) programs run identically on any Linux, Mac OS or Windows machines. Containers are isolated software units, and they share the host operating system. It enables docker containers to have smaller packages which makes them easier to deliver and reduces the server sizes and licensing costs. Docker engine is a layer on top of the host operating system that creates the environment for the containers to run on any infrastructure. It is available for most of the common operating systems including Linux, Windows, and Mac OS.

A.4 Indeed

Indeed is the leading job searching and posting website in the world, and the research shows that it has 250 million unique monthly visitors [52]. Indeed offers a job searching API for publishers. Users need to request a publisher API key to have access to the job posting data. Table A.1 shows the detailed description of its APIs.

Table A.1.
Indeed API

Parameter	**Type**	**Required**	**Description**	**Default**
publisher	string	Yes	Indeed publisher key	–
v	int	Yes	Indeed API version	–
userip	sting	Yes	End-user's IP	–
useragent	sting	Yes	End-user's browser details	–
format	string	No	Response format (json/xml)	xml
callback	string	No	Javascript callback function	–
q	string	No	Search query	–
l	string	No	Search location	–
sort	string	No	Results sort (relevance/date)	relevance
radius	int	No	Maximum distance from location	–
st	string	No	Website type (employer/jobsite)	
jt	string	No	Job type(fulltime/parttime)	–
start	int	No	Start ID of the jobs	–
limit	int	No	Maximum number of jobs	–
fromage	int	No	Maximum lifetime of the job	–
highlight	int	No	Highlight the keywords (1/0)	0
filter	int	No	Filter the duplicate jobs (1/0)	1
latlong	int	No	Geographic information (1/0)	0
co	string	No	Country of the job	us
chnl	string	No	Channel group	–

B. SOURCE CODE

The source code of the proposed personal assistant is provided in a CD as an attachment which is accessible in the */agent* folder. To develop this software, an open source framework, Laravel, is used. Most of the developed algorithms are in */agent/app* folder, which are developed by the author of this thesis. Laravel includes other public packages in the */agent/vendor* folder, which are not the results of this research project. The development environment is created based on Docker standards. The source code of this environment is in */docker* folder.