SHANGHAI JIAO TONG UNIVERSITY

IEEE CLASS

Mobile Network

The Report of Final Project

Author:
Xianyu Chen
Chenzhengyi Liu
Nianzu Yang
Ziheng Zhao

Student Number:
517030910285
517030910037
517030910301
517030910304

June 21, 2020

Since this a team work and we have actually done a lot of things in this project, this report maybe a little bit long although we have cut many parts such as background knowledge for recommendation system, some techniques and more details about our experiments(**we provide all the code along with partial training records**). **Thanks for your reading.**

Contents

1 Introduction

In an Internet era, the connections with academia are increasingly intense and we have witnessed more and more impressive papers and outstanding scholars springing up. For a student interested yet unfamiliar in a specific research topic, however, it's quite annoying to find a suitable mentor in this area since the numerous papers and scholars make him hard to get a clue. For example, the mentor selection system we used in IEEE class is quite simple and crude, we can only see the basic information of a professor and connect them for more details, not to mention there isn't any recommendation or tips given by the system. This dilemma calls for a effective way to understand the student's interests and that of potential mentors to get them matched efficiently, which could save them much time. This problem resembles a typical recommendation problems greatly, where we need to mine the perference of users and feature of items to get them macthed. Nevertheless, to our knowledge, there isn't any work before trying to build a recommendation system in this scenario(probably because the little commercial benefit it has), which leaves it a open literature. To this end, we focus on building a academic recommendation system in this project specifically for mentor recommendation, targetting at those self-motivated students searching for a suitable mentor matching their interests. **The main work and contribution of this project is as following:**

- We first analyze the data we have and student's needs in real scenario(Section 2).
- We then formulate this problem and conceive a feasible framework to solve it and apply the solution. Specifically, we devise a deep learning method for learning and making recommendation based on the student's historic collection of papers and scholars(Section 3.1). And extensive experiments have shown the advancement of this approach(Section 3.1.3).
- To further elevate it's performance, we innovatively separate the end-to-end recommendation model and use a pretraining method to learn the representations of scholars and papers. The recommendation model is redesigned to focus on recommendation and thus gaining an significant improvement. Based on our survey on Recommendation System, this trick is advanced and innovative(Section 3.2). Experiments on the comparison with previous approach show the benefits of such improvement(Section 3.2.3).
- We also elevate our prior experience to make the recommendation more informative, which is reflected by the design of our mentor candidates pool.(Section 4.2.5)
- Furthermore, we consider the situation when student are newcomers and come

up with a cold-boost recommendation to fill this gap before we know more about her/his interest(Section 4.3).

- Finaly, we design a website to collect the user data and demonstrate recommendation results in a visual way(Section 4).

The cooperation details and duties of this project is as following:

- Zhao Ziheng 517030910304 : Responsible for building the backend recommendation system based on user historic data, and writing Section 1, Section 2, Section 3.1 and Section 5 in this report.
- Chen Xianyu 517030910285 : Responsible for pretraining the embedding for scholars and papers, and writing Section 3.2 in this report.
- Liu Chen Zhengyi 517030910285 : Responsible for designing and implementing the fornt-end website, and writing Section 4.1 and Section 4.2 in this report.
- Yang Nianzu 517030910304 : Responsible for building the cold boot recommendation and data visualization, and writing Section 4.3, Section 4.4 and Section 6 in this report.

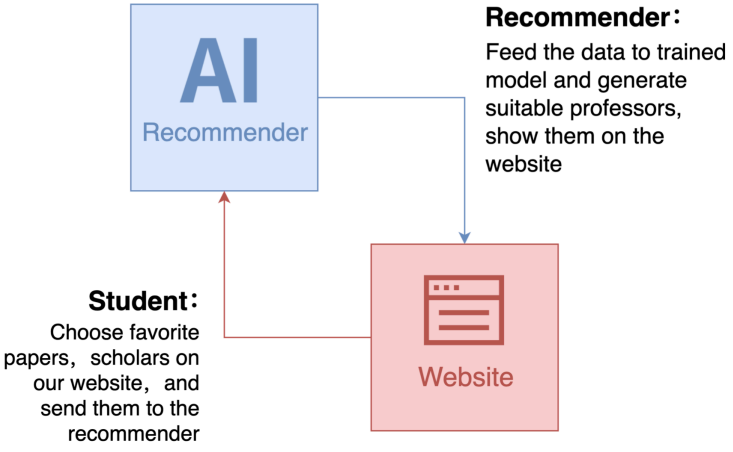


Figure 1: **Our Framework**

2 Preliminary

2.1 Analysis

When we try to recommend based on user's historic data, we can formulate all the entities in this problem as two types: scholars and papers. Thereby, they are corresponding to the classical "user-item" relationship in recommendation system. From this perspective, we might as well design a mechanisim to learn/mine the feature of each paper and scholar, which are the foundation for us to understanding the user and mentor's preference. We can then represent the user's interest based on the feature of the papers and scholars, which is mainly to aggregate their features. Now, given the interests of a student and featue of a mentor, we can score on the matching degree between them two. Clearly, the critical three steps above can all be implemented with deep neural networks: feature learning, feature aggregation and score function. This is quite rational since countless works before have proved the power of DNN in such tasks.

The only problem existing now is, we don't have user data to train the network. We address this by treat each scholar(with known mentor) as a user. Accordingly, his published paper can be seen as the favorite ones and cooperated authors as favorite scholars.

2.2 Problem Formulation

Assume we have scholar set $\mathcal{A}$ and paper set $\mathcal{P}$ in this problem. Given the favorite papers $\boldsymbol{p}^u \in R^{n_p}, p_i^u \in \mathcal{P}$ and favorite authors $\boldsymbol{a}^u \in R^{n_a}, a_j^u \in \mathcal{A}$ of user u, our goal is to find the optimal mentor $w^* \in \boldsymbol{w}^u$ for u. Here, $\boldsymbol{w}^u \in R^{n_w}$ is the candidate mentors for u and the optimal means $w^* = argmax_w(r(\boldsymbol{a}^u, \boldsymbol{p}^u, w))$. $r(\cdot)$ is the recommendation model that predict a score for given input.

We summerize our notations in the following table:

Table 1: Notation

Notation	Description
$\mathcal{A}, \mathcal{P}$	scholar and paper set
$\boldsymbol{a}^u, \boldsymbol{p}^u$	the authors and papers sequence of a user's favorite list
$\boldsymbol{A}_u^e, \boldsymbol{P}_u^e$	the embedding matrix of $\boldsymbol{a}_u, \boldsymbol{p}_u$
$\boldsymbol{w}^u$	candidate mentors
$\boldsymbol{W}_u^e$	the embedding matrix of $\boldsymbol{w}_u$
$\boldsymbol{z}^u$	the interest representation of u
$r(\cdot)$	our recommendation model

2.3 Dataset

We derive our dataset from Acemap. The dataset statistics is as following table.

Table 2: Dataset Statistics

Term	Statistics
Number of scholars	59941
Number of papers	74780
Number of student-mentor relationships	117375
Average publications per scholar	4.6
Average cooperated scholars per scholar	21

3 Back-End

3.1 Recommendation System Based on User Interests

3.1.1 Model

The three steps forementioned in our algorithmn are : feature learning, feature aggregation and score function. They are implemented respectively by : an Embedding Layer, two Convolutional Layer and a Fully-connected Layer, matrix multiplication. And **the critical component is the design of Convolutional Layer to learn user interests form favorite papers and scholars**.

We first utilize 2 Embedding Layers to generate and update the representation of scholars and papers, i.e. $\boldsymbol{A}_u^e \in R^{n_a \times d}$ and $\boldsymbol{P}_u^e \in R^{n_p \times d}$. The input of them are

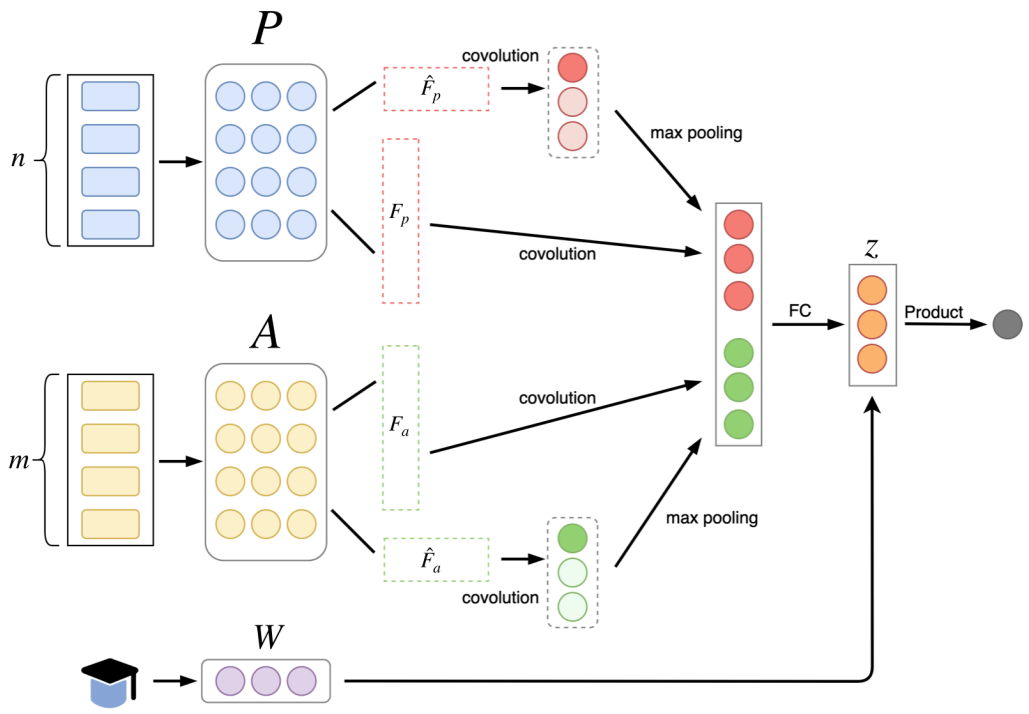


Figure 2: **Our Recommendation Model**

$\boldsymbol{a}^u, \boldsymbol{p}^u$, respectively. After that, two symmetric convolutional neural networks will extract higher level information from $\boldsymbol{A}_u^e, \boldsymbol{P}_u^e$ simultaneously and a fully-connected layer will aggregate them into a vector describing the interests of an user, $\boldsymbol{z}^u$.

For $\boldsymbol{P}_u^e$ the CNN has n_h horizontal filters $\hat{F}_p \in R^{k \times d}$ with different k. For a $\hat{F}_p^i$, it will slide from the top of $\boldsymbol{P}_u^e$ to the last row with stride 1, which means it will interact with every paper $j, 1 \leq j \leq n_p - k + 1$. The interaction is

$$c_j^i = \phi_c(\boldsymbol{P}_{u;j:j+k-1}^e \odot \hat{F}_p^i) \tag{1}$$

Here ϕ_c is the activation function and $\odot$ is inner product operator. So the result for $\hat{F}_p^i$ is

$$\hat{c^i} = [\hat{c_1^i}, \hat{c_2^i}, ..., \hat{c^i}_{n_p-k+1}] \tag{2}$$

And we apply max pooling on it to keep the most significant feature extracted by a horizontal filter. So the last output of all the n_h filter $\hat{F}_p$ is

$$\hat{o_p} = \{max(\hat{c^1}), max(\hat{c^2}), ..., max(c^{\hat{n}_h})\} \tag{3}$$

Besides, there are n_v vertical filters $F_p \in R^{1 \times n_p}$, sliding form the left of $\boldsymbol{P}_u^e$ to right. This outputs

$$o_p = \{c^1, c^2, ..., c^{n_h}\} \tag{4}$$

Every latent dimension in $\boldsymbol{P}_u^e$ has different meanings, so we don't apply max pooling on o_p.

Similarily, we have another CNN for $\boldsymbol{A}_u^e$ which yields o_p and $\hat{o_a}$ and o_a.

After that, we use a fully-connected neural network to aggregate the outputs of CNNs and generate a representation for user's interest

$$z = \phi_f(f(concat(\hat{o}_p, o_p, \hat{o}_a, o_a))) \tag{5}$$

Here, ϕ_f means the activation function.

Finally, we inner product $z \in R^d$ and $\boldsymbol{W}_u^e \in R^{w,d}$ to get the final score for each candidate(Note that $\boldsymbol{W}_u^e$ is extracted from scholar embedding layer).

$$score = Wz + b \tag{6}$$

Here, b is the bias for each mentor.

3.1.2 Training

For each sample(a student-mentor relationship), we derive 1 true mentor and 3 randomly selected fake mentor. The we have y_{true} and y_{pred} and define loss with binary cross entropy loss(BCELoss).

Besides, for bacth training, we limit the length of $\boldsymbol{a}^u$ as 20 and $\boldsymbol{p}^u$ as 10. For those longer, we only keep the most recent ones; For those shorter, we add padding term to them to align them. Other hyparameters setting are as following:

Table 3: Training settings

Hyparameters	Value
Epoch number	200
Batch size	512
Learning rate	0.001
L2 normalization	0.001
Dropout rate	0.5
n_h	10
n_v	4
d	64

By the way, we record the hyparameters for some trainings in best_metric_record.txt and fixed random seeds so it's easy to reproduce every record.

3.1.3 Testing

When testing, we pick 1 true mentor and 99 randomly selected fake mentors for each sample, and return the top 10 mentors with highest result. From a classification view, we give these 10 mentors positive label. And based on that, many classification metrics can be used here. However, the superbound of them are not necessarily 100% because true positve number is at most 1 here. We group the testing metrics here

- **Accuracy-based**: Precision (P@10), Recall (R@10), F1 Score (F1@10).
- **Ranking-based**: Area Under the ROC Curve (AUC), Normalized Discounted Cumulative Gain (NDCG@10), Mean Average Precision (MAP@10), Average Rank(AR).

In the generated recommendation list, accuracy-based metrics focus on how many true mentors are included while ranking-based metrics are sensitive about the ranked position of true mentors.

The following table shows in details the best experiment results of our model.

Table 4: Experiment results

	P@10	R@10	F1@10	AUC	NDCG@10	MAP@10	AR
Our Model	0.0911	0.9112	0.1657	0.9471	0.8632	0.0091	6.2417
Random Score	0.0101	0.1007	0.0183	0.5042	0.0454	0.0010	50.0849

3.2 Feature Learning

3.2.1 Motivation

In the recommendation system, we need to use the embedding layer to obtain the feature matrix of the paper and the author as input. At the beginning, we directly initialized it after random training, but then we thought that the relationship between the paper and the author can actually be used as an effective information to recommend the tutor. For example, if we find that two authors have published many papers in collaboration, then we can guess that they may be students of the same teacher in the same laboratory, or simply that there is a teacher-student relationship between them.

3.2.2 Algorithm

In the original data, there is the author-paper relationship, from which we can extract two other relationships, namely the author-author relationship and the paper-paper relationship. Formally form these three relationships into three relationship matrices, strictly defined as follows:

1. **Author-Paper relationship** $\boldsymbol{R} = [r_{ij}] \in \{0,1\}^{\mathcal{P}\times\mathcal{A}}$, where $r_{ij} = 1$ means author i publishes a paper j; otherwise $r_{ij} = 0$.
2. **Paper-Paper similarity** $\boldsymbol{R^P} = [r^p_{ij}] \in \{0,1\}^{\mathcal{P}\times\mathcal{P}}$, where $r^p_{ij} = 1$ means paper i and paper j have some common authors; otherwise $r^p_{ij} = 0$.
3. **Author-Author similarity** $\boldsymbol{R^A} = [r^a_{ij}] \in \{0,1\}^{\mathcal{A}\times\mathcal{A}}$, where $r^a_{ij} = 1$ means author i and author j have some common papers; otherwise $r^a_{ij} = 0$.

we model relations by considering the local proximity between author and paper vertices. In detail, we use inner products to estimate the local proximity between

author and paper vertices in the embedding space,

$$\hat{r_{ij}} = \sigma(-(\boldsymbol{p_i^e})^T \boldsymbol{a_j^e}), i \in [1, \cdots, |\mathcal{P}|], j \in [1, \cdots, |\mathcal{A}|]$$

where $\sigma(x) = \frac{1}{1+e^{-x}}$is the sigmoid function, which transforms the relation value to a probability.

To preserve the explicit relations, the local proximity is enforced to be close to author-paper relations in the bipartite graph via a cross-entropy loss function:

$$L_1(\boldsymbol{P^e}, \boldsymbol{A^e}) = -\sum_{i=1}^{|\mathcal{P}|}\sum_{j=1}^{|\mathcal{A}|}(r_{ij}\log\hat{r_{ij}} + (1-r_{ij})\log(1-\hat{r_{ij}}))$$

Similarity, for the Paper-Paper similarity, and the Author-Author similarity, we can also get the similar formula:

$$\hat{r_{ij}^p} = \sigma(-(\boldsymbol{p_i^e})^T \boldsymbol{p_j^e}), i \in [1, \cdots, |\mathcal{P}|], j \in [1, \cdots, |\mathcal{P}|]$$
$$\hat{r_{ij}^a} = \sigma(-(\boldsymbol{a_i^e})^T \boldsymbol{a_j^e}), i \in [1, \cdots, |\mathcal{A}|], j \in [1, \cdots, |\mathcal{A}|]$$

So, we can also use cross-entropy loss function to enforce it to close the similarity

$$L_2(\boldsymbol{P^e}) = -\sum_{i=1}^{|\mathcal{P}|}\sum_{j=1}^{|\mathcal{P}|}(r_{ij}^p\log\hat{r_{ij}^p} + (1-r_{ij}^p)\log(1-\hat{r_{ij}^p}))$$
$$L_3(\boldsymbol{A^e}) = -\sum_{i=1}^{|\mathcal{A}|}\sum_{j=1}^{|\mathcal{A}|}(r_{ij}^a\log\hat{r_{ij}^a} + (1-r_{ij}^a)\log(1-\hat{r_{ij}^a}))$$

To generate question embeddings that preserve Author-Paper relationship, Paper-Paper similarity, and Author-Author similarity simultaneously, we combine all the loss functions to form a joint optimization framework, namely, we solve:

$$min_{\boldsymbol{P^e}, \boldsymbol{A^e}}(L_1(\boldsymbol{P^e}, \boldsymbol{A^e}) + L_2(\boldsymbol{P^e}) + L_3(\boldsymbol{A^e}))$$

Once the joint optimization is finished, we can obtain the pre-trained embeddings $\boldsymbol{P^e}, \boldsymbol{A^e}$, which can be used as the input of existing recommendation system.

To sum up, **our pre-training model is to use the inner product of the embedding matrix to simulate the relationship between the real paper and the author as much as possible, so as to extract effective information for subsequent operations.** The whole framework is shown below

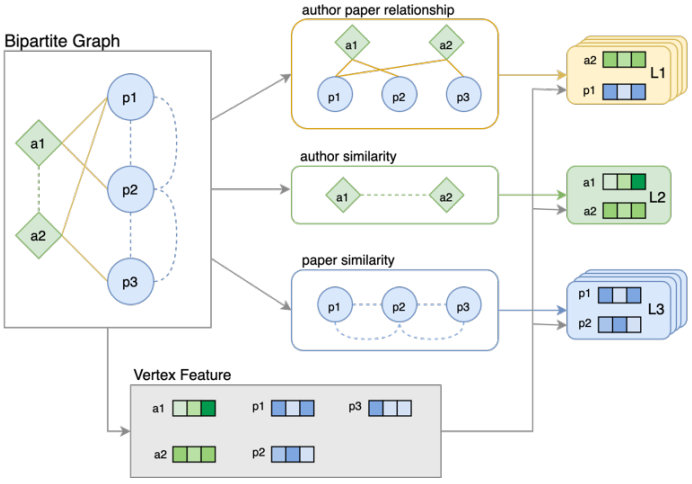


Figure 3: Pre-Framework

3.2.3 Experiments

After inputting the pre-trained embedding matrix into the recommendation system for further recommendation training, we take 1 true mentor and 99 randomly sampled false mentors for each sample, and take Top10 results to evaluate, then the results are as follows:

It can be seen that compared with the random initialization, the model using pre-trained embedding is more stable in training, and there will be no overfitting soon, and each index has also been improved to a certain extent. This illustrates the effectiveness of our model.

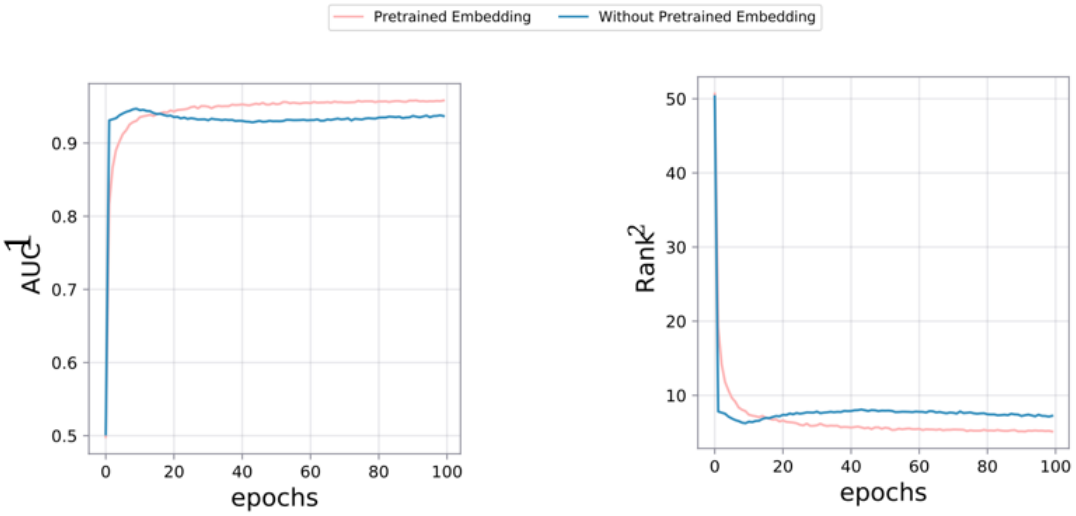


Figure 4: Result

4 Front-End

4.1 Design

Our front-end designed is based on a web design tool - Django. Django is a free and open web framework to help developers build websites with few codes. Django takes care of much of the hassle of Web development, so developers can focus on writing your app without needing to reinvent the wheel.

Figure 5 shows the core design of a django program, which can be summarized as MTV pattern:

- **M (Model)** : Model is responsible for the function of programming, and the mapping of business objects and databases (ORM).
- **T (Template)** : Template is responsible for how to display the page (HTML) to users.
- **V (View)** : View is responsible for business logic and will call Model and Template when appropriate.
- **Besides MTV**, a URL distributor is also needed. Its function is to distribute URL page requests to different views for processing, and then View calls the corresponding Model and Template.

Based on the MTV pattern of Django, we build our website framework as Fig. 6.

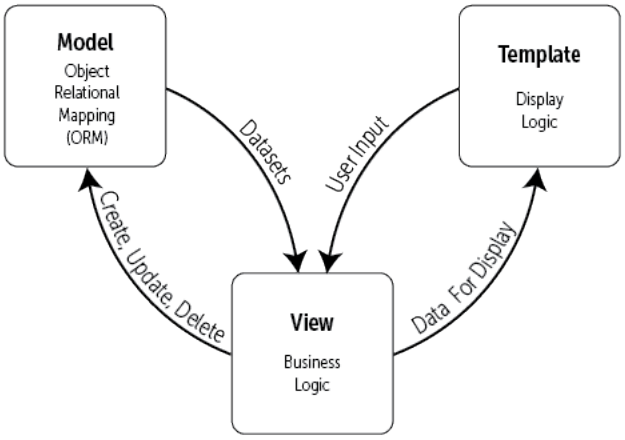


Figure 5: The MTV Diagram of Django

The Template module shows the main pages of the framework, the View module presents the corresponding python functions and the Model module is composed of our database and back-end program. When users get interaction with Template html indexes, the commands are sent from front-end into View's functions which will connect to Model's data and program. The yellow arrows show the inter connections between the three modules in our framework.

In our website framework, there are five main web pages listed as follows. Their implementation details will be discussed in the next subsection.

1. **index.html** : The login in index of users.
2. **Registration.html** : The **Registration.html** provides a sign up index for new users.
3. **Search.html** : Users can search and collect for their favorite authors or paper in the **Search.html**.
4. **Favorite.html** : The **Favorite.html** displays the collections of users' account.
5. **Recommend.html** : The recommend result of our back-end program is showed in **Recommend.html**.

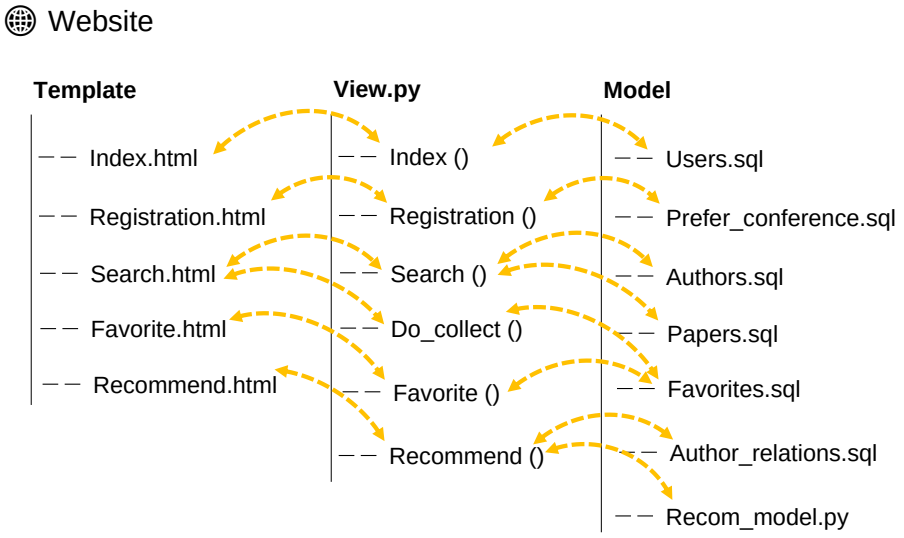


Figure 6: The Overall Framework of Our Website

4.2 Implementation

Last subsection, we present the five web pages and their connections with other modules of our front-end framework. In this subsection, we will discuss their View functions and how they are connected.

4.2.1 Index()

The index() function is designed for users' login. When new users occur, it gets users' login information and store it into Users Table. When old users sign in, this function will search Users Table and decide whether the account matches the password or not.

4.2.2 Registration()

The Registration() function is designed for new users to create new user account. What is new in this function is that it can defer users' preference by asking users' favor conferences, which is named the **Cold Start** in Recommender Systems. In implementation, we first collect users' favor conferences. Then, we search the author-conference map to get the most relative authors with these conferences. Next, for

better recommend, we keep searching other authors who are most relative to conferences in a similar categories. At last, we choose top 20 authors shown in the web page to users for Cold Start Recommendation.

4.2.3 Search() & Do_collect()

The Search() function is designed for authors and paper separately. It connects to both Authors Table and Papers Table in the database. When user searches for a author (or paper), he can just enter a part of the author's (or paper's) name in the search box, and then the all relative results will be listed in the downside page.

Users can also use the Do_collect() function to collect their favor authors (or papers). Each search result is returned to the web page with a "favorite" button which allows users to collect their favorites by it. Once a author is collected by current user, the button changes into a "delete" one, and users can remove their collections through it.

4.2.4 Favorite()

The Favorite() function is to show the collections of user. It connects the Favorites Table to get users' collections and returns them onto the web page. Similar to the Do_collect() function, it also allows user to remove their collections by clicking "delete" button.

4.2.5 Recommend()

When users have collected a certain amount of authors (and papers), they can generate their personal recommendations through Recommend() function. It calls our back-end model, searches in all teachers and shows results.

However, in practice, there is often a large number of teachers we need to search and this is time consuming. Some searches of teachers are not suitable for current user, e.g. a medicine teacher for a computer science majored student. Therefore, we use a pre-recommend method to choose a limit number of teachers (named teacher pool) sent to back-end model. Using the Author_relations Table, we iteratively search the user's collected teachers and their co-authors and finally build the teacher pool.

subsectionCold Boot Recommendation

The recommendation system needs to make particular recommendation for users based on their previous behaviors and their interests. If we have obtained these information, it's easy to make suitable recommendation through specific recommendation algorithms or models.

Nevertheless, a problem comes to our mind naturally that how we make recommendation for new users, since we don't know their previous behaviors and their interests. This kind problem is called "Cold Boot Recommendation" in recommendation system. **In order to make our system more user-friendly, we designed our own form of Cold Boot Recommendation as below.**

In the front-end, we design a registration page for new users to create new accounts. This page contains a area where the new user can choose ther interested field and corresponding conferences. Then we will make some preliminary recommendation based on their choices after registration is done. This registration page is demonstrated as below. For example, in Fig. 7(b), we choose 5 conferences in AI field. Here, we only list three fields. We will make it completed later.

(a) Full View (b) Choose Conferences

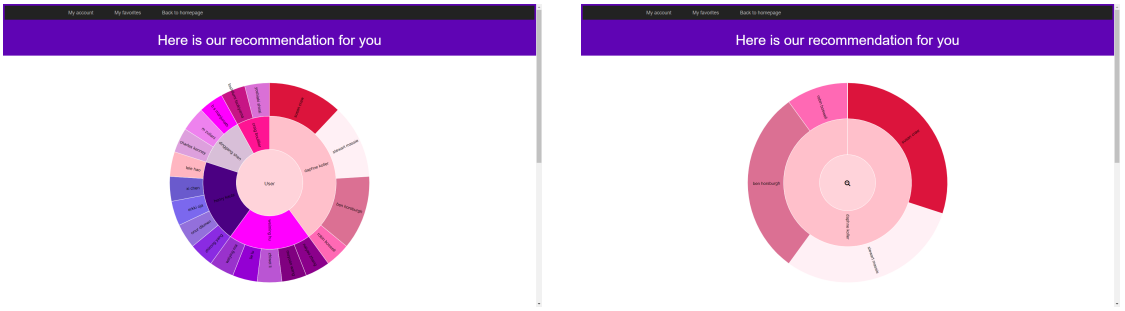
Figure 7: Registration Page

How do we recommend for new users according to their choices in this page? Now, let's discuss about our method. In the back-end, we divide the conferences according to the fields and classes (CCF has classified these conferences into 3 classes, i.e., A, B and C). For a given value n which represents the number of objects we need to list in the preliminary recommendation, we let one part of these n objects is directly linked to the conferences the user choose. The remaining candidates are chosen from the conferences in their interest fields except the ones they have chosen. It is worth

mentioning that in each part, A-class conferences holds the highest probability of being recommended. B-class ones is only second to A-class.

4.3 Data Visualization

In the final recommendation page, we visualize the results in a dynamic graph as Fig. 8 shows. It's a form of sunburst chart.



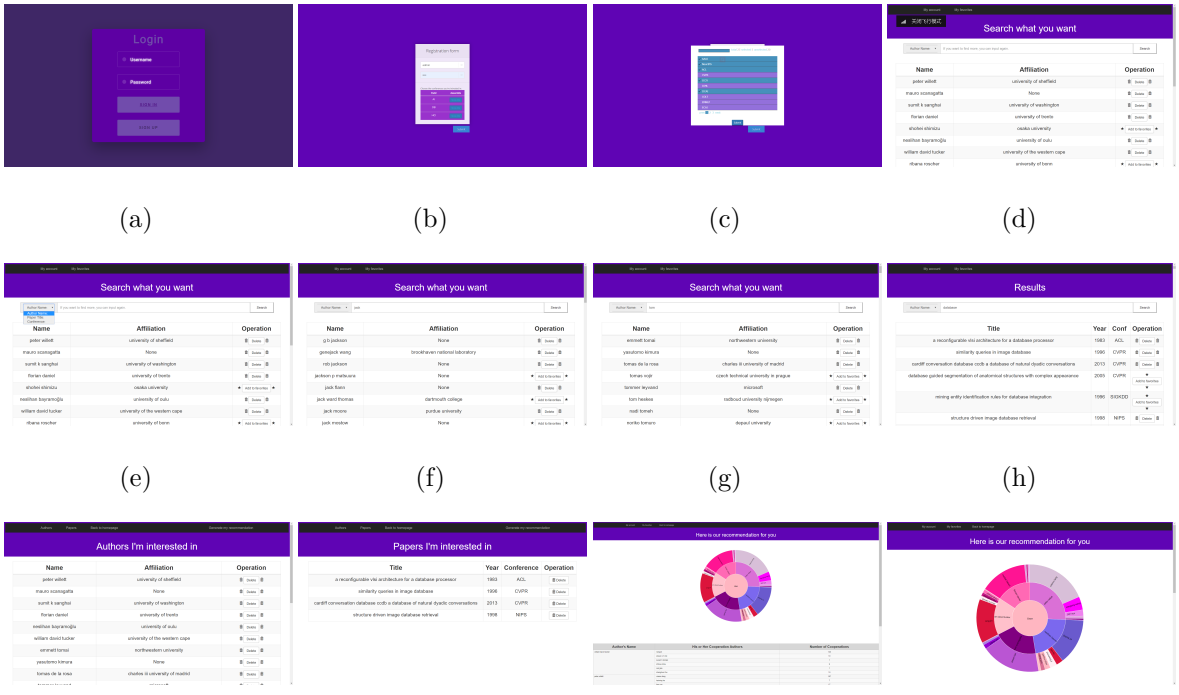
(a) Full View (b) Part View

Figure 8: Data visualization

In this page, there exists a table listing the authors shown in the sunburst chart. We write a javascript which can transfer the table below into a sunburst chart directly. The central area represents the user. The outer circular ring next to the central area contains the authors recommended through our model. The outmost circular ring contains the potential authors obtained from the inner authors. For example, as Fig. 8(b) shows, when we click on the inner author "Daphne Koller", we can get a new sunburst chart which centers on "Daphne Koller". We can click on the center again to return to the original chart. The degree to which we recommend an author is measured through the angle of the corresponding sector.

4.4 Demonstration

Since we designed a Cold Boot Recommendation function for new users, we will demonstrate a intergrated series of procedures beginning from registration as below.



(a) (b) (c) (d)

(e) (f) (g) (h)

(i) (j) (k) (l)

Figure 9: Demonstration

Figure. 9(a) shows the homepage. We click on the "Sign In" button and it will turn to registration page. After inputting the user name and password, we choose our interested fields and conferences in the Cold Boot Recommendation module. Afterwards, a page like Fig. 9(d) will be presented on the screen. The table below lists some authors we may take interest in based on our choices on the registration page and our Cold Boot Recommendation algorithm. We can add our interested authors by clicking on the "Add to the favorites" button and can remove them from our favorites through clicking on the "Delete" button.

The search range of the search box can be switched as Fig. 9(e) shows. We can search authors, papers and conferences. As Fig. 9(f)(g)(h) shows, we search for authors named "jack", authors named "tom" and papers whose title containing "database" one after the other. We add those we are interested in to our favorites.

Next, we can click on the "My favorites" button in the navigation bar to see what we have added. Through clicking on "Authors" and "Papers" button at the top left corner, as Fig. 9(i)(j) shows, we can see authors and papers we're interested in. In the end, we click the button "Generate my recommendation" at the top right corner to gain the final recommendation according to our interests.

5 Conclusion

In this project, we design and implement a Supervisor Recommendation System, which is tailored for those who has academic interests but confused about choosing suitable mentor. Based on real scenario, we devise two type of recommendations: recommendation based on user interests and cold-boost recommendation. For former, We deliberately utilize a deep learning model based on CNN to make recommendation, pretrain the embedding to further improve it and define a candidate mentor pool to squeeze the searching area and improve time efficiency. For the latter, we manually choose popular conferences from different subjects and offer to users to guess her/his interests and make initial recommendation. Besides, we bulid a website for users to browse, collect academic information and get recommendation feedbacks, which suits our model well. Finally, we conduct extensive experiments to prove the efficiency of our recommendation model.

For future work, firstly, we might try different methods for our recommendation model, such as the prevailing self-attention mechanism and GAN. Second, we

could extend the functionality of our website such as detailed demonstration page for scholars, papers, conferences and so on(since they are relatively irrelevant to the recommendation process so we save this part of work). Third, we can use more supplementary information about users to improve the quality of recommendation, such as the venues they are interested in, and the semantic feature of each paper(which will involves NLP) and so on. Fourth, we can enlarge our dataset by digging more student-relationships.

6 Acknowledgement

In the end, we want to show our gratitude to Mrs.Fu and TAs. After this semester, we have learned rather considerable knowledge and our programming ability has been enhanced. We believe that these new knowledge will benefit us a lot later. Also as the third-winner in the final contest, we appreciate your thoughtfully design of the contest and prize.

Thanks for your efforts :) !