Job recommendation system based on LinkedIn API

First Author: Yutong Chen
UNI: yc3993

Second Author: Yuqin Zhao
UNI: yz4131

Third Author: Shuai Ren
UNI: sr3849

Abstract

Nowadays, hundreds of millions of jobs are posted on LinkedIn, which is the home to countless job seekers and employers. How to recommend and notify these jobs to the job seekers who are interested and suited to correctly and efficiently can be non-trivial. To realize this, a classification distribution should be generated among all job seekers in order to personalize their needs. Thus, what we are working at is to use the personal description job seekers posted on LinkedIn to classify them in different domains. Then we can use the domain to recommend related jobs to them. In addition to recommend jobs to different people with different preferences, we also aim to provide the user with some glance about their expected income according to our domain suggestions. In a word, we try to personalize work domain recommendations and personalize salary prediction. For the Job category prediction, the RNN model could predict in 0.9056. For the salary prediction, the Regression Model has a test set RMSE of 19.72.(10% approx). And besides we build a Web UI for our system.

1. Introduction

1.1. Focused Problems

Nowadays, hundreds of millions of jobs are posted on LinkedIn, which is the home to countless job seekers and employers. According to data from New York Fed as shown in Fig 1.1.1, since the outbreak of the epidemic in 2020, the unemployment rate has risen sharply.

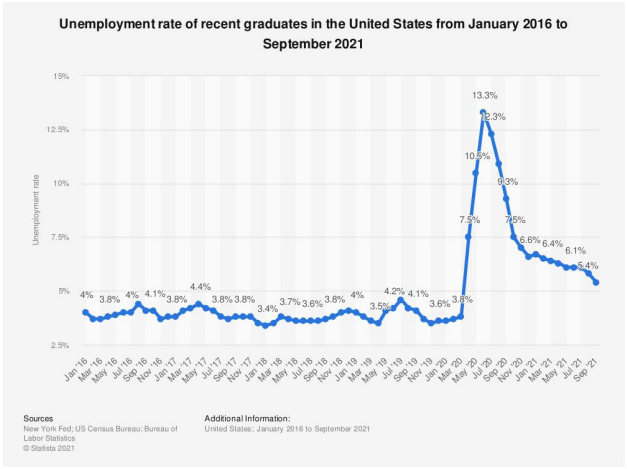

Figure 1.1.1: Unemployment rate of recent graduates in US

How to recommend and notify these jobs to the job seekers who are interested and suited to correctly and efficiently can be non-trivial. To realize this, a classification distribution should be generated among all job seekers in order to personalize their needs.

Thus, what we are working at is to use the personal description job seekers posted on LinkedIn to classify them in different domains. Then we can use the domain to recommend related jobs to them. For example, as shown in Fig 1.1.2, these two are personal descriptions (or resumes) information from LinkedIn. We are trying to use this information to classify them to different job domain so that we can recommend related jobs.

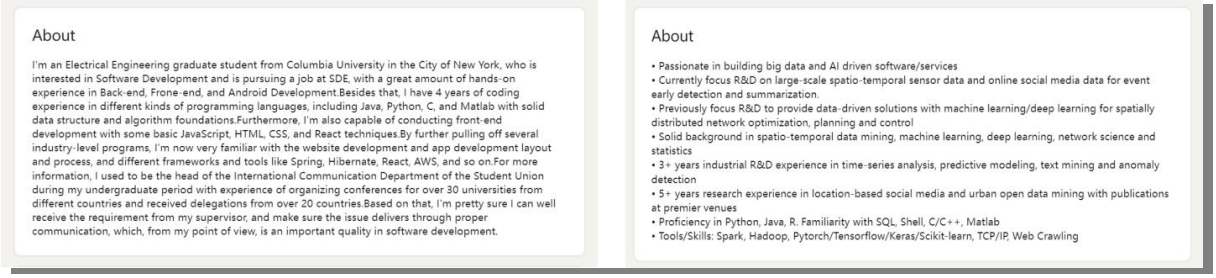

Figure 1.1.2

In addition to recommend jobs to different people with different preferences, we also aim to provide the user with glance about their expected income according to our domain suggestions. For instance, if a user is classified as suitable to HR, we can give them a prediction of their salary they can expect as long as they provide some basic information, like working experience and degree, etc.

In this article, we used several approaches to deploy our job recommendation system. Including:

- Used python crawler in the webdriver to obtain data in the comment on LinkedIn.
- Applied SparkSQL and SQL to do data preprocessing.
- Implemented TensorFlow on GCP to build Recurrent neural network.
- Used Spark ML and sklearn to compare and train Linear regression model.
- Conducted Flask to deploy machine learning model on internet.
- Used HTML, CSS, JavaScript to visualize our data and salary prediction website.

2. Related Work

2.1 Recommender Systems

As a general rule, recommendation engines don't have a very clear purpose, or they don't have a very clear purpose. Generally speaking, users don't even know what they want. In the present time, the placement of the recommendation engine is in the hands of the user [1]. It uses the recommendation algorithm to generate the list of items that users may be interested in based on historical behavior, interest preferences and demographic characteristics of the user. Then, it uses the recommendation algorithm for the search engine to generate the list. It has been a trend in recent years for recommender systems to become much more prevalent. Among the four classifications in the Recommendation algorithm are: Content-based filtering, Collaborative filtering, Rule-based approaches, and Hybrid approaches [2].

Collaborative Filtering (CF)

Collaborative Filtering is a way for organizations to achieve filtering through collective wisdom to remove products and information users are not interested in. Basically, collaborative filtering is a way of recommending content to users based on their shared interests to help find what they are really interested in for them on the Web [3]. This method assumes that the easiest way to find the content a user is interested in is to find other users with similar interests to that user and then recommend the content that they are interested into that user. As an extension of the memory-based recommendation system (CF) that primarily makes recommendations based on heuristics.

Content-based filtering (CBF)

the content-based filtering (CBF) method extracts recommendations based on the content itself. A step in the process of performing this function is the selection of the similarity function. This is one of the most important considerations in the process. Choosing the most appropriate similarity function to measure the similarity of two items or users is one of the primary tasks that must be accomplished when selecting the best similarity function. The next major task is recommending similar items and users. To recommend projects in a simple manner, it is a good idea to follow the most common recommendation strategy, which is to suggest projects that people have a tendency toward, but the target user does not [4].

Hybrid filtering (HF)

As for hybrid filtering (HF): Taobao, as a result of a fusion of the above algorithms, has an algorithm that offers both content-based recommendations and collaborative filtering recommendations. In addition to the specific fusion, the specific application scenarios should also be included in the fusion, for example, a feature fusion or an algorithm fusion.

2.2 Literature Review

Using algorithmic approaches such as textcnn to classify the types of fields is what the article [5] uses to approach the classification problem. Then a pre-trained glove vector and a tokenization sequence are used in combination to generate a sentence matrix that recognizes text as an image and provides advantages for identifying important features. At last, the output for the 25 categorizations is derived from the soft Max layer. Textcnn has the advantage of replacing the traditional machine learning method with high accuracy at the beginning with textcnn's improved accuracy, and textcnn's advantage of improving accuracy are the benefits. As a result of collaborative filtering, the second article [6] creates a system of recommendations. One of the disadvantages associated with CF is the fact that it may not be enough interactive data available to users or projects (for example, users who have abnormal tastes, or new projects or users). It is the purpose of this article to address this shortcoming by proposing the idea of developing another type of recommendation system, the so-called content-based method, which is based on the use of clear domain-specific functions of users and projects to construct models. This hybrid recommendation system that combines both content-based methods and CF is referred to as a hybrid recommendation system. It is supposed that as a hybrid methodology, it would combine it with content-based signals into a CF-based core system, so it is considered to be a hybrid procedure [7]. In this article, we distinguish between two scenarios in order to present

content tailored to the requirements of two different scenarios based on the tasks and the state of a user when he or she starts "cold.". If the user is starting fresh, the content-based similarity measure is compiled by configuring the neural network embedded within the learning process so as to evaluate the similarity between different jobs. As a result, even if jobs do not interact with each other directly, they can still be associated with one another. This article addressed one of the problems users without any activity face, which is the process of categorizing their resume content into fine-grain job categories, and then recommending popular jobs that fall within these categories to them [8].

2.3 innovation

In our studies of the previous papers, we found that there are some issues that have not been solved, based on the research we conducted：

1. Using the process of calculating distance, it isn't easy to handle the works with too much data since the procedure is complex.
2. The recommendation systems written in other people's literature cannot update the data in real time, but the job requirements are changing day by day.
3. Although many jobs only work with jobs classification, the users, such as students, may need more information about the jobs they work on.

3. Data Mining

LinkedIn is a social network designed for professional networks, with a focus on employment. Its website and applications include hundreds of millions of professional user profiles and recruitment information [9].

In the part of obtaining data, we used two methods, one is the LinkedIn website crawler, through this method we can obtain information including email, phone number, address, education, and job position. However, the feature values obtained by this method are too few. First, the lack of some features including work experience leads to low accuracy of salary prediction, and the second lack of job description makes it difficult for us to achieve simple predictions of job classification. So, we choose to download the corresponding data on the official website.

In the data processing step, we have made many attempts. First, we try to use LinkedIn API to obtain data. However, LinkedIn API is used for the interface APP and it's hard to apply. Thus, we try the second method, then use python webdriver crawling web pages to obtain data in the comment on LinkedIn.

Then we start to use web drive to crawl data online. The principle of this method is that by entering the source code of the user information web page, you can use the search method to find personal information such as Email, address, education background in the HTML language of the interface. Find more people's relevant information through similar search methods.

We use Selenium to interact with web pages and obtain information. Due to the limited information of LinkedIn users that can be downloaded once, we will download multiple times. Then we download ChromeDriver, which is a separate executable that WebDriver uses to control Chrome. Also, we installed time module, parsel module and csv module in our Jupiter notebook. Then we can log in to our LinkedIn account to obtain LinkedIn information. In downloading the data module, we need to use Parsel, which is a library for extracting data points from websites. Since we have installed it at the beginning, we also need to import this module in our "script.py". As we said before, we will use the Inspect Element on the web page to locate the HTML tags we need in order to extract each data point correctly. But there is a problem here is that the format of the username may be different. When we crawled the data, we found that some users' information could not be obtained. We discovered this problem after debugging. In order to solve this problem, we searched for LinkedIn the user's data format, and write these three formats into our conditional sentences, to solve this problem. In the search step, as we said before, we put data in various formats into conditional statements, and write preprocessors to help us find the corresponding keywords. Then save the required keywords into a csv file. A large amount of data can be obtained by repeating it many times. Part of our data shows in Figure 3.1.

	Name	E-mails	Headline
0	Vaishvik Kumar	kvaishvik24@gmail.com	Software Engineer @ Samsung R&D Institute,Noida
1	Rutuja L.	ru2jalimaye11@gmail.com	Associate BI
2	Vedant Sharma	vedant.12sharma@gmail.com	Student at PES University
3	Pavan Kumar pothuru	pavankumar2302@gmail.com	Machine learning Enthusiast, Graduated from In...
4	Shikhar Tiwari	shikhartiwari.bme18@itbhu.ac.in	Analyst Intern @RoaDo | Ex- Intern @IIM Ahmeda...

Figure 3.1

Since our data only contains some of the features we need, we have downloaded some personal information and job descriptions from the government website. We mainly start from:

NYC Open Data： *https://data.cityofnewyork.us*

Data World： *https://data.world/datasets/jobs*

After that, we used SparkSQL to do data preprocessing. Finally got the data form we wanted.

4. Methods

4.1 RNN Model

For the job classification part, we choose RNN model. A convolutional network is a neural network that is specialized for processing a grid of values X such as an image, can readily scale to images with large width and height, and some convolutional networks can process images of variable size [10].

RNNs can in principle map from the entire history of previous inputs to each output. Any function computable by a Turing Machine can be computed by a finite size RNN [11].

First, we do data prepossessing, we use word embedding method to encode our text into vectors. Label after encoded shown in Figure 4.1.1. Instead of using a one-hot vector to represent each word, we can add a word embedding matrix in which each word is represented as a low-dimensional vector. Note that this representation is not sparse anymore, because we're working in a continuous vector space now. Words that share similar/related semantic meaning should be 'close to each other' in this vector space (we could define a distance measure to estimate the closeness). We also tried one hot encoder method, but the accuracy is far below our expect (acc=0.73).

	text	job_category	cat_id
5931	Responsibilities 1. Responsible for the end-to...	data scientist	25
6833	RESPONSIBILITIES: Kforce has a client seeking ...	software design	26
9617	Willing to relocate candidate to the Seattle, ...	software design	26
9230	Under general supervision and with a high degr...	software design	26
2587	Seeking a Data Scientist advanced in Python An...	data scientist	25

Figure 4.1.1

Second, we define our RNN model with two LSTM layers. There are two reasons why we used RNN. First, the dataset is large enough for deep learning model to fit the prediction. We also tried Random Forest in SparkML, the result was below 0.9. Second, instead of using CNN or other deep learning model, we apply RNN model because CNN can only take and process one input individually, and the previous input has nothing to do with the next input. However, some tasks need to be able to better process the information of the sequence, that is, the previous input is related to the subsequent input. The model summery is shown is Figure 4.1.2. During our training, we applied several optimize methods including early stopping, adam optimizer and drop out.

Third, we estimate our model. Loss and Accuracy graph shown is Figure 4.1.3 and Figure 4.1.4. The accuracy reaches about 90%. We use tsne and PCA to visualize our data(Figure 4.1.5). As we can see, different labels are fairly separated from each other. Then we put different text into our model. For example, I put “Making the right decision is a difficult but interesting thing, you must calculate the outcome of each choice and calculate the impact of each outcome. After I took the course of probability theory and statistics in college, I became fascinating in decision making methods. This passion carried me through my education at PKU. I studied new languages including SQL” into our model, the result successfully show that I am a data scientist. We also try to into a whole resume, the model also makes correct prediction.

Layer (type)	Output Shape	Param #
embedding_4 (Embedding)	(None, 250, 100)	50000
lstm_8 (LSTM)	(None, 250, 50)	30200
lstm_9 (LSTM)	(None, 50)	20200
dense_4 (Dense)	(None, 27)	1377

Total params: 101,777
Trainable params: 101,777
Non-trainable params: 0

None

Figure 4.1.2

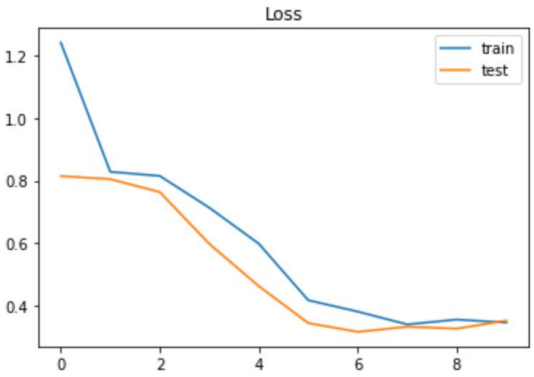

Figure 4.1.3

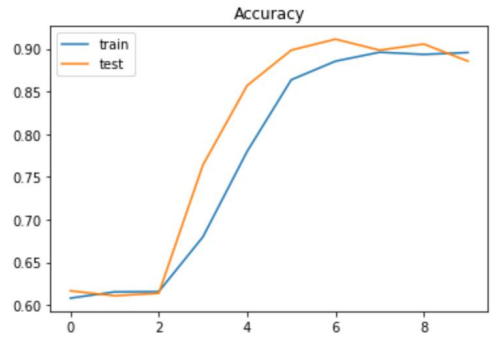

Figure 4.1.4

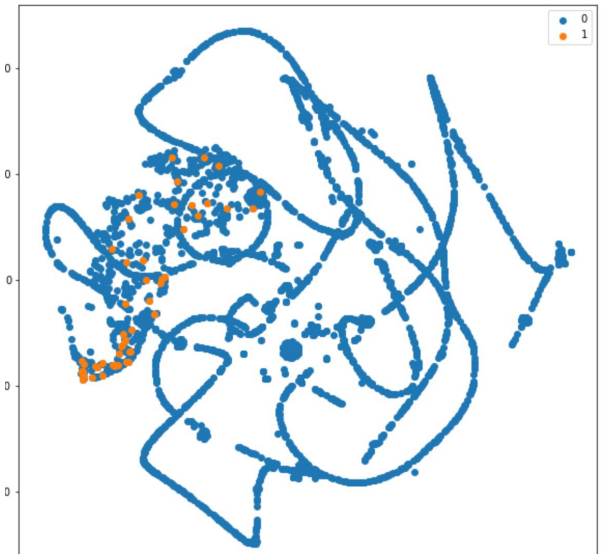

Figure 4.1.5

Also, we tried something that did not work out well.

After the model is built, we try to deploy it on the internet. However, the dataset is so large, and we don't have much knowledge about database and backend, we failed to do it. If we have more time, we will try to deploy our model and make it easier for users to use it.

4.2 Regression Model

Then for the salary prediction part, we use our datasets, which contains more than 100,000 salary data. We use 6 features for our model, which are job types, degrees, majors, industries, working experiences and working locations. Intuitively, the first four features are all categorical features. The rest are numerical since we use "miles to major cities" to represent the working locations in our datasets. This is much more useful for salary prediction than the name of location since the nearer one is from major cities, the higher income one might get.

To train the Regression Model, we use One-Hot-Encoder to generate one hot vectors for the first four categorical variables. Then we assemble all features to one vector and use it to train the model.

In order to test the model, we split the datasets to 7:3 in order to create a test set. Since the whole datasets (categorical columns) are encoded together, the test set and training set are encoded in the same way. As a result, we can use the test set to test our model. Fig 4.2.1 shows the prediction of the model. Evaluator shows that our Regression Model has a test set RMSE of 19.72(10% approx), which is quite accurate.

Now that we have the RNN Model and Regression Model, we want to build a Web User Interface so that user could interact with our model. Also, like we mention before, the dataset is so large, and we don't have much knowledge about database and backend, we failed to do it.

label	features	prediction
19	(27,[4,7,11,22,25...	21.053326297405064
20	(27,[4,7,11,22,25...	16.74219323944338
20	(27,[4,7,11,22,25...	20.272407349510146
20	(27,[4,7,11,22,26...	19.849790186483375
20	(27,[4,7,11,22,26...	19.459330712535916
20	(27,[4,7,11,22,26...	15.55473597306134
20	(27,[4,7,11,22,26...	15.164276499113896
20	(27,[4,8,11,22,25...	13.491686896564715
20	(27,[4,8,11,22,26...	18.55158121334199
20	(27,[4,8,11,22,26...	14.256526999919956
21	(27,[4,7,11,22,25...	25.33230166628742
21	(27,[4,7,11,22,25...	21.42770692681286
21	(27,[4,7,11,22,25...	20.646787978917942
21	(27,[4,7,11,22,25...	20.256328504970483
21	(27,[4,7,11,22,25...	18.304031135233217
21	(27,[4,7,11,22,26...	29.611277035169778
21	(27,[4,7,11,22,26...	14.773817025166437
21	(27,[4,8,11,22,26...	21.28479753097419
21	(27,[4,8,11,22,26...	18.16112173939453
22	(27,[4,7,11,22,25...	26.503680088129798

Figure 4.2.1

Thus, we just build a User Interface for the salary prediction model (Regression). This work could basically be divided into two parts: One is the design of the back end of the Web UI, and the other is front-end.

For the back end, we need to enable our website to use the model we build. As a result, we import pickle in our model configuration file and we use "pickle.dump" to store our model as a .pkl file. Now we can read this file and get the

model in our website.

Then we have to deal with the user's input from the website. We use Flask library to solve this problem, getting the input of our model. However, user's input are just some strings and numerical variable, and we need to vectorize them. Intuitively, we should use One-Hot Encoder to encode them and assemble all features to one vector, transforming the original input to vectorized input, while in fact we cannot do this since we cannot make sure the string input to be encoded in the same way when we built the model. For example, in the feature "major", "COMPSCI" was encoded as "00001000" while now it could be encoded as "10000000". Fortunately, our categorical features are finite and each feature has finite categories. Thus, we need to process the user's input from the website, transforming them to one hot vectors in the same way as we built the model. Hence, our model can work correctly.

With regards to the front-end, it is basically just some JavaScript about website designing. Our website UI is shown in the experiment section and system overview section.

5. Experiments

5.1 RNN Model Test

First, we try to compare different kinds of model, our CNN model for text classification is shown in Fig 5.1.1. However, the result reaches 85% and start to be shaking. Although I am using NVIDIA Tesla V100, the training time is still so long due to the large parameters. So, we try RNN model. After we turn the size of the hidden layers and add drop out, our model can reach 90% in half hour.

```
def create_model():
    model = tf.keras.models.Sequential([
    tf.keras.layers.Conv2D(6, kernel_size=5, strides=1,  activation='tanh', padding='same'),
    tf.keras.layers.MaxPooling2D(pool_size=2, strides=2),
    tf.keras.layers.Conv2D(20, kernel_size=5, strides=1,  activation='tanh', padding='same'),
    tf.keras.layers.MaxPooling2D(),
    tf.keras.layers.Conv2D(120, kernel_size=5, strides=1, activation='tanh', padding='valid'), #C3
    tf.keras.layers.AveragePooling2D(), #S4
    tf.keras.layers.Flatten(), #Flatten
    tf.keras.layers.Dense(120, activation='tanh'), #C5
    tf.keras.layers.Dense(84, activation='tanh'), #F6
    tf.keras.layers.Dense(10, activation='softmax')])
```

Figure 5.1.1

Second, we download relevant information about different types of people from LinkedIn into our model to make predictions. Then we wrote a test program to convert the text into a vector, input it into our model, and then output it in the form of text. The result can accurately predict the job category. Shown in Fig 5.1.2 if we input a data of a software engineer. "Currently I am mainly working on back-end, such as micro-service, automated workflow and event processing pipeline, but I also have experience in deploying infrastructure" The model can accurately output the job category.

```
predict('Making the right decision is a difficult but interesting thing, you must calculate the outcome of each choice and calculate the impact of each outcome. Af
'data scientist'

predict('Currently I am mainly working on back-end, such as micro-service, automated workflow and event processing pipeline, but I also have experience in deployin
'software design'
```

Figure 5.1.2

5.2 Regression Model Test

In the former part, I have already shown the result of regression evaluator, which shows that our model has a test set RMSE of 19.72(10% approx). Thus, I am going to perform some test on our User Interface.

As shown in Fig 5.2.1, the first example is a man/woman who is a master's in computer science and just graduate. He/she might be a junior worker in his/her future company and probably work in New York City or other big cities, since I set the distance from major cities to 0. After pressing submit, the result shows that he/she might gain an annual salary of 106,000 dollars.

Is this result accurate? I find some real data from PayScale. It shows that "computer science professionals with a master's degree, specifically, see an average salary of $101,000" [1]. The prediction is close, but a little bit higher than the real data. This is explainable since he/she in the example works in major cities. Hence, it is logical that he/she earns more than average.

Now, for the same example, he/she becomes a manager after hard working for 20 years. According to the prediction, he/she now can earn 166,000 dollars a year, as shown in Fig 5.2.2. The increase in logical as we expected.

Then let us do another test on a man/woman who is math teacher with a bachelor's degree, and probably works in countryside, since he/she works 100 miles from major cities. As shown in Fig 5.2.3, he/she will earn 40,000 dollars a year. According to statistic data from "ZIPPIA", a math teacher with a bachelor's degree has a medium annual income of 46,676 dollars. The prediction is lower than the medium result from working location.

In conclusion, the Regression Model enjoys great accuracy

and works well in our Web User Interface.

Figure 5.2.1

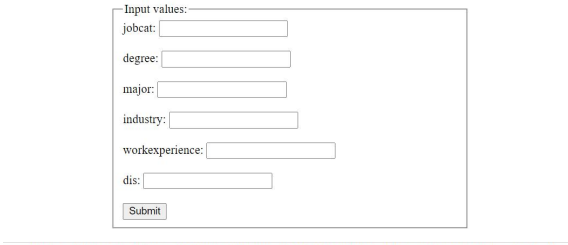

Figure 5.2.2

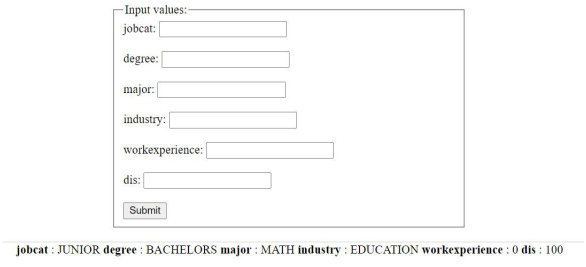

Figure 5.2.3

6. System Overview

Our software mainly consists of front-end and back-end. In the back-end part, as we said before, we first use flask to save our machine learning model into related files so that we can call it later. Then, because we cannot directly use text as the input of the model, the website must be displayed in the form of text to facilitate users to understand and use. Using the coding work, we did on the back end; we wrote a program to convert the text into a digital vector form and then input it into the model. In the front-end part, we first learned the knowledge of HTML5, CSS, JAVASCRIPT, and designed the website on this basis. The website is mainly composed of two parts: user input and system output prediction value. Every time the user enters his occupation category and other relevant information into the designated location we designed, our system will give the user a salary forecast. After that, we use the heroku platform to download the software we need on the platform and then publish the website on the platform for users to use.

But our system also has some areas that need to be improved.

1. The database problem of the RNN model. Due to the lack of relevant database knowledge in our group, there is no way to put the RNN model of big data into the interactive page.
2. The data visualization failed to be displayed in the interface, because I tried to put the RNN visualization method tsne into the web page, but it was also difficult to complete because of the large amount of data. In addition, due to time reasons, we spend most of our time debugging the parameters of the deep learning network. Each training takes a long time, which leads us to fail to complete the final visualization part, which can only be displayed in the model file Up.
3. In the process of data acquisition, what we are doing now is to use loop statements to repeatedly call to acquire data, but after we have studied airflow, I found that we can use airflow to set the acquisition time and frequency so that the computer automatically We get the data. And you can set the specific time to get the data, so that users can get the latest data. But after we learn airflow, we have completed the data acquisition part, if we have more time, we will definitely develop in this direction.

To use our application, users first need to download our RNN model from our GitHub folder. We saved the pkl file of the RNN model so that users can download it directly. Then users can use the prediction code we wrote to copy their resumes to make predictions, and then they can know their most relevant work. Shown in Fig 6.1. After that, open our web page and enter the position just predicted and other basic information about yourself to predict your salary. Shown in. In this application, we used Flask, gunicorn, scikit-learn, numpy, pandas, pickle4 and Spark.

```
model = pickle.load(open('model.pkl','rb'))

def predict(text):
        seq = tokenizer.texts_to_sequences(text)
        padded = pad_sequences(seq, maxlen=MAX_SEQUENCE_LENGTH)
        pred = model.predict(padded)
        cat_id= pred.argmax(axis=1)[0]
        return cat_id_df[cat_id_df.cat_id==cat_id]['job_category'].values[0]
```

Figure 6.1

7. Conclusion

In this project, our team mainly completed the establishment of a job recommendation system. This recommendation system can first allow users to put in their resumes and get the most suitable position for the user with an accuracy of 0.9056, and then the user can put the position we gave and some basic information into the recommendation system again, and then the recommendation system can display the expected salary of the user in 0.86 accuracy which helps the user better understand job search information.

Through this project, we learned a lot of knowledge. First, in terms of data processing, we have made many attempts. Including: 1. Use LinkedIn API to obtain data. 2. Use python webdriver to obtain data on LinkedIn. 3. Download the data in the corresponding open-source database. Through these three attempts, we exercised Spark streaming data processing experience and learned how to perform web crawlers. Although the first two methods choose to download part of the data because of insufficient feature types, we have mastered the corresponding methods.

Second, in terms of model building, we use the knowledge of Spark ML learned in class to predict more than 100,000 pieces of data on the salary prediction model. Compared with directly using the sklearn library, the speed is much faster, and it is more conducive to our follow-up configuration work. On the RNN model, we learned to use TensorFlow to build an RNN network and used a variety of tuning methods to optimize the accuracy of the model. Both models achieved high accuracy and used visualization methods to understand the data.

Third, in the display part, we have no previous web page writing and back-end experience. So, we learned the relevant language and used flask to configure our model online for users to use.

Although we encountered various problems in the course of the project, including insufficient stream data characteristics, low accuracy of the RNN model, web page compilation, etc., we solved all of them by consulting the information. Through this semester's study and the final project, we have learned a lot of software engineering knowledge. I am very grateful to the teachers and TAs for their help.

In the future, if we have more time, I would like to make the following improvements to this system. 1. There are too few features for streaming data acquisition. We can build an API to acquire related data by ourselves. 2. The RNN model has a large amount of data. The model itself includes 100,000 resumes. The large amount of text makes it difficult for us to display it on the web quickly. We want to learn the knowledge of the database if we have time to help us accomplish this goal. 3. The data visualization part is not beautiful enough, which requires us to learn relevant languages. This winter holiday our team is also preparing to learn some relevant knowledge by ourselves to help us improve this project.

References

[1] Jeremie Harris. “Ethical challenges of recommender systems”. Retrieved from https://towardsdatascience.com/ethical-problems-with-recommender-systeems-398198b5a4d2

[2] B.Thorat, Poonam & Goudar, R. & Barve, Sunita. (2015). Survey on Collaborative Filtering, Content-based Filtering and Hybrid Recommendation System. International Journal of Computer Applications. 110. 31-36. 10.5120/19308-0760.

[3] Wikipedia. Retrieved from https://en.wikipedia.org/wiki/Collaborative_filtering

[4] Wikimedia. “Recommender system”. Retrieved from https://en.wikipedia.org/wiki/Recommender_system#Content-based_filtering

[5] Minaee, Shervin & Kalchbrenner, Nal & Cambria, Erik & Nikzad Khasmakhi, Narjes & Asgari-Chenaghlu, Meysam & Gao, Jianfeng. (2021). Deep Learning--based Text Classification: A Comprehensive Review. ACM Computing Surveys. 54. 1-40. 10.1145/3439726.

[6] W. Shalaby et al., "Help me find a job: A graph-based approach for job recommendation at scale," 2017 IEEE International Conference on Big Data (Big Data), 2017, pp. 1544-1553, doi: 10.1109/BigData.2017.8258088.

[7] Al-Maliki, Murtadha. “User based hybrid algorithms for music recommendation systems.” (2018).

[8] Abhijit Roy. “Introduction To Recommender Systems- 1: Content-Based Filtering And Collaborative Filtering”. https://towardsdatascience.com/introduction-to-recommender-systems-1-971bd274f421

[9] LinkedIn. “About LinkedIn”. Retrieved from https://about.linkedin.com/zh-cn?lr=1

[10] Deep Learning by Ian Goodfellow, Yoshua Bengio and Aaron Courville, http://www.deeplearningbook.org/ , chapter 10.

[11] Lecture material by bionet group / Prof. Aurel Lazar (http://www.bionet.ee.columbia.edu/).

[12]https://www.payscale.com/research/US/Degree=Master_of_Science_(MS)%2C_Computer_Science_(CS)/Salary

Contribution:

Yutong Chen (yc3993): Data Mining, RNN model, Web UI
50%

Yuqin Zhao (yz4131): Regression Model, Web UI
40%

Shuai Ren (sr3849) : Data Mining
10%