REAL/FAKE JOB POSTING PREDICTION

Team 23: Qinglin Li(xep4ez), Zekui Fu(pry7cp), Jinshuo Xiao(zmy4fg), Zongdi Qiu(zq9ms)

1 PROBLEM DEFINITION

1.1 MOTIVATION

With the help of rapid growth of technology, it's convenient for job seekers to find suitable jobs through social media and online advertisements. However, some people take advantage of it to advertise fake jobs post. In this way, they can ask you to pay deposit before you get a job and glean your personal information. In particular, due to the pandemic, job seekers are facing a perfect storm when it comes to these fake job postings. For one thing, some necessary changes actually make fake job opportunities seem more believable. For example, remote work is usually one of the requirements of these fake job advertisements, which makes it more appealing because people are willing to work at home nowadays. Similarly, during normal time, it would be a red flag if your potential employer avoid face-to-face meetings or interviews. Nevertheless it becomes a reasonable request. Therefore, an automated model to predict false jobs post is a valuable tool to face the difficulties in the field of job seeking.

1.2 MODEL STATEMENT

Our goal is to create a classifier to find job descriptions that are fraudulent. We abstract 18 defining attributes from 1.8k observations with one outcome variable. The input of the model, in other words, the type of the attributes including numeric and text features. The final result will be evaluated based on two different models, one for numeric data and the other is for text data. The predictive variable is fraudulent and has binary values. Zero indicates a genuine job, while one indicates that the job description is fraudulent.

2 PROPOSED IDEA

2.1 ONE-HOT AND EMBEDDING

The preprocessed data is still text, and the data cannot be machine-learned at this time, so we need to vectorize the text. Our approach is to first encode all the words that appear in the preprocessed text through one-hot, and then use embedding to vectorize the words through self-learning of the text to make them meaningful. Embedding is very critical, it is this step that makes RNN meaningful. For example, given a machine learning related material, through self-learning of the text, we can get that the relationship between "RNN" and "LSTM" is very close, and the response in vector space is that the direction and size of these two vectors are very similar, thereby giving meaning to words.

	#
narz	1
advertisingpitch	2
teamassist	3
preach	4
proudly	5
industryanalyze	6
ideasdevelop	7
tigkeitsbereich	8
applicationshave	9
psychographic	10
assessmentsco	11
special	12
proje	13
processautomate	14
regulationsensuring	15
environmentcoordinate	16
decisionsoutstanding	17
raymore	18
communicators	19
assurancecomfortable	20

Figure 1: Figures and word comparison charts of one-hot(1-20)

	0	1	2	3	4	5	6	7	8	9	...	40	41	42	43
0	0.008375	0.015300	0.040773	0.040974	0.003742	0.015290	-0.015773	-0.018650	-0.017240	-0.005166	...	-0.022979	0.008238	0.014625	-0.015602
1	0.044104	0.016911	0.033605	0.001626	0.031226	-0.001775	0.016182	0.024055	0.029936	-0.012372	...	-0.039473	-0.034800	-0.036071	0.002394
2	0.023356	0.045563	-0.007740	0.007882	0.022173	0.036547	-0.037307	-0.017676	0.000870	0.033811	...	-0.044967	0.029555	-0.014789	0.006142
3	-0.024792	-0.021394	0.032053	0.031545	0.035923	0.016303	-0.015986	-0.034888	0.033644	-0.020079	...	-0.020465	0.019077	0.009715	0.006955
4	0.025719	0.019930	0.008137	0.048560	0.037626	0.026307	0.012180	0.007349	-0.024847	0.043640	...	-0.018760	-0.030230	0.024878	-0.020458
5	-0.025857	0.028682	-0.039538	-0.005785	0.025149	-0.001585	-0.025095	0.024721	-0.005293	0.009363	...	0.004140	-0.024360	-0.044273	0.015956
6	-0.005116	-0.039160	0.043666	0.030911	-0.002898	0.040279	-0.017385	0.037435	-0.029418	0.005907	...	0.025315	0.034638	0.021799	-0.002588
7	-0.045693	-0.012849	-0.005860	0.023519	0.019123	-0.001801	0.046389	0.042365	0.034431	-0.047544	...	0.029116	-0.042583	-0.037614	-0.042468
8	-0.013549	0.022434	0.037652	0.040070	0.002295	-0.029078	0.038782	-0.038559	0.034022	0.042973	...	0.015818	-0.041446	-0.031245	-0.040872
9	0.026733	-0.046148	0.049991	-0.039795	-0.011253	0.046764	-0.013954	-0.009937	0.014241	0.004949	...	-0.020999	0.015485	0.022866	0.005024
10	-0.039463	-0.035713	0.022619	0.044891	-0.025983	-0.002099	-0.032065	0.008332	-0.042873	0.049385	...	0.010099	0.047879	0.023524	0.013191
11	-0.023510	0.002756	-0.042736	-0.025336	0.046141	0.008070	0.016121	0.004742	0.016697	0.045266	...	0.010949	0.012848	-0.019224	-0.014190
12	-0.009909	0.019746	-0.010226	0.034377	-0.029050	0.040581	0.002186	-0.007121	0.041426	0.037359	...	-0.008746	0.017960	-0.025634	0.000506

Figure 2: Data after embedding(Each line represents a word)

2.2 LSTM

Because this is a text data, the RNN model is a powerful tool for processing text data, but there is a fatal problem that the gradient disappears, especially for long text analysis. To overcome this shortcoming, we decided to use the LSTM model and improve the model performance by further improvement of the model. The reason why LSTM performs better is: the traditional RNN node output is only determined by the weights, biases and activation functions, which is a chain structure, and the same parameters are used for each time slice. The reason why LSTM can solve the gradient disappearance problem of RNN is because LSTM introduces a gate mechanism to control the circulation and loss of features. For example, LSTM can pass the features of time t2 at time t9, which solves the problem of gradient disappearance. LSTM is composed of a series of LSTM units, and its chain structure is as shown in the figure below.

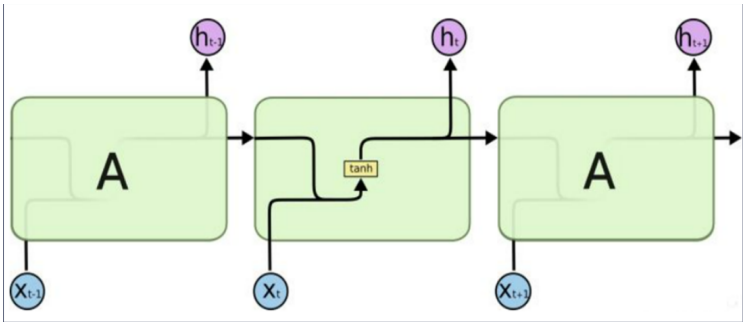

Figure 3: RNN

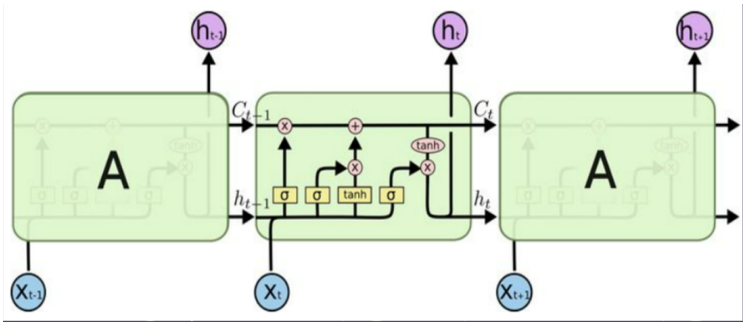

Figure 4: LSTM

3 Related work

3.1 Fake Job Prediction using Sequential Network

Nowadays, there are more and more privacy leaks in the world, and more and more graduates are persecuted by fake positions. Some companies use a variety of methods to deceive job seekers for their benefit. The goal of our team's final project is the same as this paper. The paper [1] uses a machine learning model to predict the chances of a job being fake, thereby greatly reducing the amount of such fraud, which can help job seekers make smart decisions and avoid bad choices.The paper used NLP to analyze the data, and then the model was trained as a sequential neural network.

3.2 Fake Job Recruitment Detection Using Machine Learning Approach

The purpose of this paper [2] is to avoid fraudulent posts and then use classification-based techniques of machine learning, where different classifiers are used to examine data samples. Considering classifiers alone and ensemble classifiers can enable job seekers to detect and avoid pitfalls from a large number of false job postings.

3.3 Machine Learning and Job Posting Classification: A Comparative Study

In this paper [3], the main problem is the need to identify fake job postings in numerous job listings. The author studied many machine learning classifiers, such as support vector machines, decision trees, random forests. After that, the authors processed the data and then feature extraction. Finally, the authors use various specific evaluation metrics to evaluate and summarize the results and compare them with other classifiers. Our team faced a similar problem to this paper, and our team will use Natural Language Processing and Naive Bayes Algorithm and Neural Network in the final

project to solve the problems that arise in the fake work in the paper and in the project.

4 EXPERIMENT SETUP

In this section, we describe the data collection process and proprocessing of cleaning data. We show the details of methodology that we used to implement this experiment and analysis of the training process in the following section.

4.1 DATA SETS

At the first stage, the data for our project is available at the well-know Kaggle website. However, this is a highly imbalanced dataset that consists of 17,880 observations and 18 features, among which contains 17,014 true job posts and 866 fraudulent job posts.

By analyzing the data, we find that we have 4 numerical features: job id, telecommuting, has company logo and has questions. We delete these meaningless features since we are dealing with text classification problems. The only integer column we preserve is 'fraudulent', which we label fake job posts as 1. Furthermore, we check the missing data and drop all the null values from further analysis in our dataframe. Finally we used six of the seventeen independent variables for experiments: title, description, requirements, company profile, employment type and required experience. As the label class, fraudulent is our dependent variable. Figure 1 shows the content of our dataframe. After selection process,the final dataset has 8383 records where 8173 are legitimate, and 210 are fraudulent. For the learning feature, we only select 'description' because we find that the accuracy remained basically the same when we tried other learning features.

4.2 DATA PREPROCESSING

One crucial step in data preprocessing is cleaning data. First we need to solve the imbalanced problem. When we train with the imbalance data, we get a high accuracy just by predicting the majority class, but fail to capture the 'fraudulent' class. Second, with the respect of text classification, we need to clean some ambiguous content and preserve the helpful data for training models. Since this meaningless content may create complexity and difficulty in semantic analysis in the training process, resulting suboptimal accuracy.

Two general ways to balance data is resampling and Cost-Sensitive Training. Resampling is one way to directly change the distribution of the original data. We can achieve this by duplicating the minority class or reducing the majority class samples to force the data from unbalanced to balanced. However this approach has limitation that may cause overfitting or loss of information. Therefore, we apply different weights to each category. We apply weight '5' to fraudulent class, while weight '1' to true class.

Cleaning process follow several steps. First step is removing stopwords. Stopwords are part of the text that help us to understand the sentence. We need to remove these stopwords to reduce the complexity in the training process. Our stopwords are imported form nltk library. Furthermore, we remove http and URL information and some punctuation marks. Concerning the stemming, we used the porter stemmer library for stemming of text in order to convert each word into its root form.

5 Experiment Results and Analysis

5.1 Word Embedding

After completing the data cleaning and model selection process. We should decide how to convert words to vectors.Vectorizing words is a relatively difficult task in our project. We can collect all the words in our data and use these words to form a dictionary. However, we cannot simply use the "number of pages" of a word in the dictionary to represent the meaning of this word. Because this will lose the hidden meaning between all the words. So we need to affect these words as vectors in the same space. The closer the cosine distance, the closer the meaning of the word vector.
For the generation of word vectors, we use two methods. The first is to use the already trained word2vec database. The word2vec tool takes a text corpus as input and produces the word vectors as output. The word vector table in the tool is trained by a large amount of text data, and the distance between the words corresponding to the vector can represent the generous connection between the words.
The second way is to put the embedded process into the training part in the deep learning neural network. The mapping of each word to a vector is learned through deep learning.

5.2 Training the Model

In our experiments, we used the data obtained by the two vectorization methods for model training. Because our data is too imbalanced, we increase the weight of the parameters in the loss function. In this way, when the model incorrectly classify the sample as true job, the model will suffer several times of loss.

5.2.1 Using Pre-trained Embedding Data

The training set consists of about 10,000 samples. Each sample consists of a paragraph describing the job title. When doing vectorization, we use pre-trained word2vec model(the model is called googlenews-vectors-negative300) to convert each word in each sample into a vector of length 300. At the same time, the number of words in each sample is cut to the same number.

```
Model: "sequential_14"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
lstm_9 (LSTM)                (None, 300, 150)          270600
_________________________________________________________________
dropout_8 (Dropout)          (None, 300, 150)          0
_________________________________________________________________
flatten_8 (Flatten)          (None, 45000)             0
_________________________________________________________________
dense_8 (Dense)              (None, 1)                 45001
=================================================================
Total params: 315,601
Trainable params: 315,601
Non-trainable params: 0
_________________________________________________________________
```

Figure 5:

When we use the embedded data generated by the pre-trained word2vec tool as training data. The trained model is almost a complete failure. We tried to tune some model parameters and the model performance was poor. Because the dataset composed of 95 percents true jobs. So models with less than 95 percents accuracy are meaningless. From the big gap between the training and testing accuracy we can conclude that this model can't fit the true distribution of data. Due to time constraints, we gave up this approach halfway through.

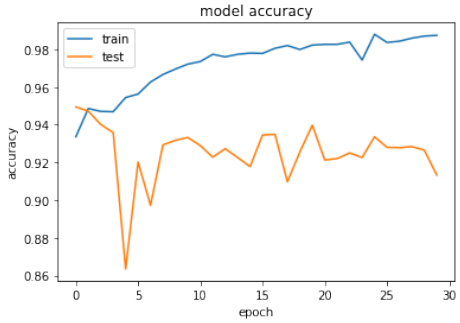

Figure 6: Acc

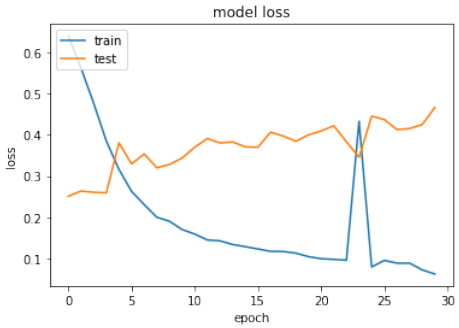

Figure 7: Loss

5.2.2 Using Embedding Layer

In this part, we skip the process of manually embedding with the word2vec tool. We just input the one-hot encoded data directly into the model and the first layer of the model, the Embedding layer, will participate in the training together with the entire model.

```
Model: "sequential_12"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
embedding_12 (Embedding)     (None, 100, 400)          2000000
_________________________________________________________________
bidirectional_12 (Bidirectio (None, 200)               400800
_________________________________________________________________
dropout_12 (Dropout)         (None, 200)               0
_________________________________________________________________
dense_12 (Dense)             (None, 1)                 201
=================================================================
Total params: 2,401,001
Trainable params: 2,401,001
Non-trainable params: 0
```

Figure 8: LSTM Model with Embedding Layer

For the parameters of this model structure, there are mainly two parameters that can be tuned. The number of features for which the word is embedded first. It determines the vector dimension to which one-hot word vectors are mapped. followed by the number of neurons in the LSTM layer. Due to the limitation of compute capacity, we performed a simple grid search in the experiment. Finally, a relatively optimal solution was found. That is to map the one-hot encoding to a 400-dimensional feature vector. and set the neurons of the LSTM layer to 100.

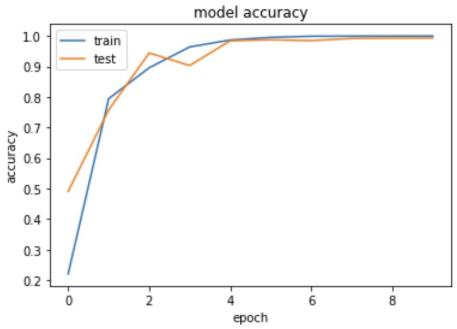

Figure 9: Acc

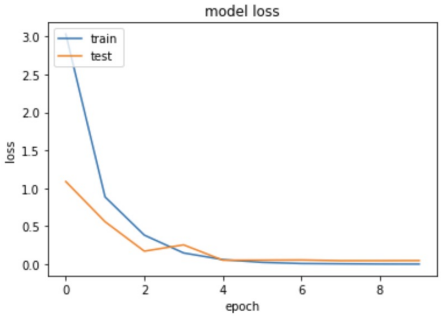

Figure 10: Loss

To ensure that the gradient will stop at the optimal position, we set our learning rate to be dynamic. The learning rate will decrease as the loss decreases. The result is that the gradient descent becomes gentler when the model is about to converge. This is very helpful for us to find the optimal solution

As shown in the acc and loss figure above. Our model can converge smoothly, and the model has this relatively good performance.During the training process of cross-validation, the training accuracy and test accuracy are almost the same, which means that our model does not overfit.

5.3 Analysis

Since our first model did not fit well, in this section we only show the performance of the second model on the test set.

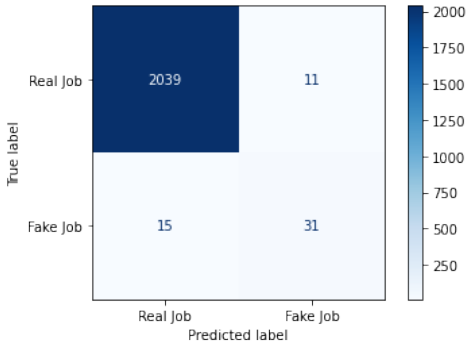

Figure 11: Confusion Matrix

This figure shows the confusion matrix generated by our model using testing dataset. From this confusion matrix, it is very intuitive to see how difficult this task is. Our data is really imbalanced. Only about 50 samples out of more than 2000 samples are fake jobs. However, our model can find more than 30 of them. This proves that our model makes sense.
The reason for the failure of the first model is that the data of the word vector library is too generic. At the same time, the degree of freedom after reducing the embedded layer is too low. This makes it impossible for us to find the distribution of the data in the hypothesis space.

6 Conclusion

More and more fake job postings are posted on various information platforms. So in this project, we created a classifier to identify fake job titles. We abstract 18 properties defined from 1.8k observations with one result variable The final results are evaluated based on two different models, one for numeric data and the other is for text data

Our approach is to first encode all words appearing in the preprocessed text, and then use embedding to vectorize the words and make them meaningful through self-learning of the text. Embedding is a critical step, and it is this step that makes RNNs meaningful. After completing the data cleaning and model selection process. We should decide how to convert words to vectors. Vectorizing words is a major task. We can collect the words in the data and use those words to form a dictionary. In our experiments, we use data obtained by two vectorization methods for model training. Because our data is so imbalanced, we increase the weights of the parameters in the loss function.

Through parameter optimization, a relatively optimal solution is found. From the confusion matrix it can be seen that our data is not quite balanced. At the end of the experiment, out of more than 2000 experimental samples, only about 50 samples were fake assignments. However, our model can find more than 30 of these fake data. Such experimental results demonstrate that our machine learning model is meaningful.

We have learned a lot in this project. For example, in processing data. There are many ways to deal with imbalanced data. To truly understand a certain method, you need to do some visual experiments on the data set; each method has its own advantages and disadvantages. In order to obtain a better result, you can try a general The processing strategy for : use a combination of several familiar and

reliable methods, and use the parameters of these methods as hyperparameters wrapped outside the learner.

REFERENCES

[1] Ranparia, Devsmit, Shaily Kumari, and Ashish Sahani. "Fake job prediction using sequential network." 2020 IEEE 15th international conference on industrial and information systems (ICIIS). IEEE, 2020.

[2] Dutta, Shawni, and Samir Kumar Bandyopadhyay. "Fake job recruitment detection using machine learning approach." International Journal of Engineering Trends and Technology 68.4 (2020): 48-53.

[3] Nasser, Ibrahim, and Amjad H. Alzaanin. "Machine learning and job posting classification: A comparative study." International Journal of Engineering and Information Systems (IJEAIS) ISSN (2020): 6-14.