Fake Job Posting Prediction and Analysis with Machine Learning

Shuyun Tang[1 [1111−2222−3333−4444]] and Ruiz Trevor[1 [0000−1111−2222−3333]]

University of California, Santa Barbara, CA 93106 {shuyun,tdr}@ucsb.edu

Abstract. In this project, we develop and explore data-driven strategies to identify fraudulent job posts and advertisements. The dataset we will use a publicly-available dataset from the University of the Aegean comprising 18,000 job descriptions [9], of which approximately 800 are flagged as fraudulent.
We will combine supervised machine learning (ML) and natural language processing (NLP) ideas to predict a job post's classification as fraudulent or non-fraudulent based on varying representations of the text content of the job ad. Specifically, we will seek combinations of ML and NLP methods that give high classification accuracy. Other statistical sampling methods and extensive preprocess techniques will be applied to boost the models' performances.

Keywords: NLP · Machine Learning · Recurrent Neural Network · Transformer.

1 Introduction

Nowadays, there are lots of jobs posted online or on platforms such as Linkedin and Handshake. However, among the gigantic number of jobs posted online, some of them are completely fraudulent, and some of them have misleading or unmatched information regarding the salary packages or the benefits. Identifying these posts can help job-seekers avoid scams and improve the integrity of online hiring.

To achieve this objective, this paper experiments various machine learning models including Logistic Regression, Random Forest, XGB, and the state-of-the-art language models includng Bi-directional LSTM, BERT, for the NLP tasks. The results are validated on a public dataset [9] that contains binary classification label of job fraudulence, i.e. fraudulent or non-fraudulent. We utilize statistical methods including downsampling, ROC Curve optimization, grid search for hyperparameters, and feature engineering to further improve our baseline models' performance.

The major contributions of this work are as follows:

– Survey various representation studying suitable machine learning models with extensive experiments. Also conduct experiments on the Transformer based deep learning models.

– Formulate a simple and efficient pipeline for the data preprocessing, tokenization (or vectorizing), and model fitting.
– Incorporate statistical techniques to balance and improve the models' performance.

2 Methods

2.1 Datasets

We used the Employment Scam Aegean Dataset (Employment Scam Aegean Dataset (ESAD)) from the University of Aegean [9]. It containts 18,000 job descriptions posted on an online platform, of which approximately 800 are flagged as fraudulent. The original dataset contains the columns of job id, job title, location. department, salary range, company profile, job description, job requirements, benefits, indicator of telecommuting, indicator of having company logo, indicator of having questions, employment type, required experience, required education, industry type, function, indicator of fraudulent (response variable).

2.2 Exploratory Data Analysis

In this section, we will explore the some basic statistics in this dataset. The response variable (indicator of fradulent) is extreme unbalanced, which contains 17200 non-fraudulent and 800 fraudulent.

The Fig.1 indicates that the salary range, department info, required experience and education, function, and industry have a lot of missing values. As our objective is to mainly deploy the NLP in the text related features, we will drop these columns and keep only the features listed in the table 1. in our models' training.

Table 1. ESAD Text Features

Feature Names	Explanations
title	job title, separated by the space
company profile	a basic overview of the company, usually less than 3 sentences
description	a detailed overview of the company, usually more than 3 sentences
requirements	the experience needed to apply this job
benefits	some bonus benefits, usually is specific holidays, unique cultures, etc.

The Fig.2 takes a brief look on the required experience level. Among all the job posts, we can notice that most of them requires the entry level for the graduates, or the mid-senior level and associate level for those employees leave their old companies. This is very intuitional since these three groups do play the main role in the online job applications.

Besides, the Fig.3 are the most common words generated by WordCloud(MIT licenced). Even though there does not exist a clear pattern among these two

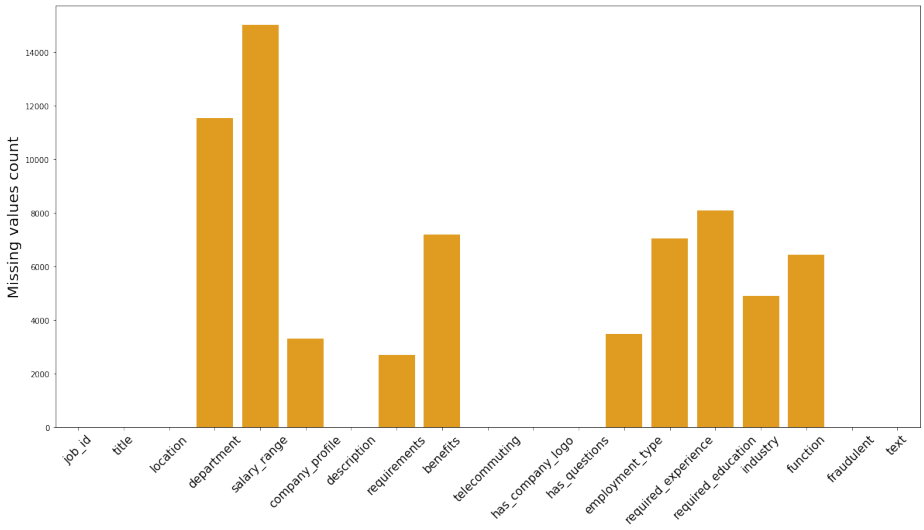

Fig. 1. Missing values in the features

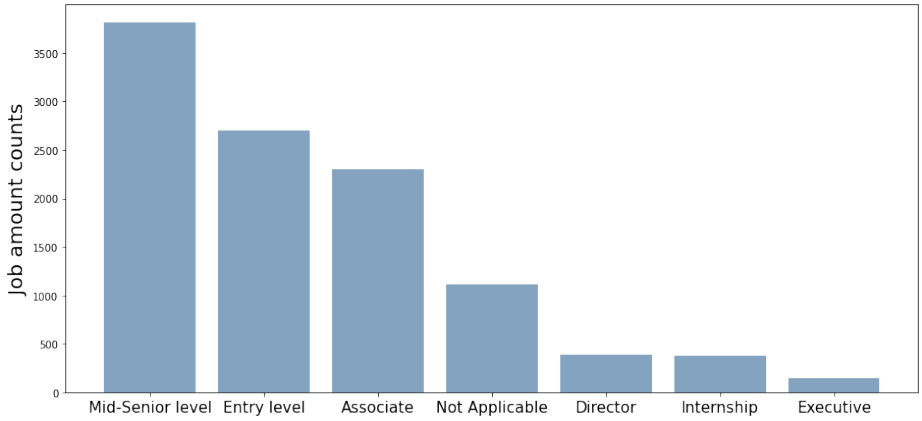

Fig. 2. Number of job postings for different required experience

WordCloud figures, we can still notice that the fraudulent job posts tend to have more words about promising such as ensure, aptitude, project, which usually gives the candidates a sense of easy to get in. In contrast, the real job posts tend to have more diverse words covering in team work, communication, location, social media, work pace, etc. Among these real keywords, they usually have a equally spread and weights as their sizes don't vary too much, while the fraudulent key words usually have very large proportion for the specific words as the ensure, customer, product have very large sizes.

Fig. 3. WordCloud of Non-fraudulent (left) Fraudulent (right)

2.3 Preprocessing

Since our main goal is to let the model study the texts in the job posts, we will merge the text features from the Table 1. into an mega text column and apply the preprocess pipeline to them. In order to optimize different models' performance, we use different preprocessing methods for different models.

For the traditional machine learning models, we utilize the Spacy [5] tokenizer to create our token object, which is equal to linguistic annotations, lemmatize each token, remove spaces, convert into lowercase, and then remove the stop words. We then create the bag of words, which vectorizes the tokens and creates a vector for each of them. The advantage of using this method is that different vectors can have relationships to others, which implicitly formulate the latent corresponse between the words, i.e. the vectors of 'apple' and 'pear', which belong to the fruit, may have a smaller Euclidean Distance compared to the vector of 'tiger'.

For the Recurrent Neural Networks or the BERT model, we use the word embedding instead. The advantage of this method is that most of the pretrained word embeddings [6] are based on gigantic pretrained dataset, which will be accurate enough for this task. And for other non-pretrained word embeddings,

the built-in functionalities would help the corresponded model better study the representation and converge fast.

We then setup the pipeline using the Bag of Words and combine the above preprocess methods as well as the models below.

The models should be generalized in both classes even though the fraudulent has a extreme small sample. Besides the modification of the dataset columns, we also try the downsampling of the non-fradulence job posts as it is extreme unbalanced. We also apply the ROC curve optimization in order to alleviate the recall score, which corresponded the fraudulent prediction.

2.4 Evaluation and Metrics

We split our dataset properly and 20 percent is our test dataset, which contains 177 supports and 3399 supports samples for the fraudulent and non-fradulent classes respectively. After the downsampling, the total dataset has 2000 fraudulent samples and 830 non-fraudulent samples and same train-test split is applied. We evaluate our models using the precision, recall, and F1-score, which are all the macro average between the two classes as we think the unbalanced classes will make the weighted average biased. We add the weighted F1-score for the extra references. However, the weighted F1 is strongly dependent on the sample sizes of different classes, which should not be used to reflect a model's recall and precision balance when the sample is not balanced.

3 Models and Experiments

In this section, we will give a detailed overview and the corresponded results of the proposed models in the fraudulence classification tasks. The models are implemented using Keras 2.4.3, sklearn 0.22.2, Python 3.7, and trained usingan NVIDIA Tesla K80 (12GB of RAM) provided by Google Colab Pro.

3.1 Traditional Machine Learning Models

We first experiments the following models: decision tree, random forest, logistic regression [2], and XGB [1].

From the Table 2., we can notice that the among these models, the logistic regression has the best performances in all the metrics, and is less resource demanding in the computation compared to the models like random forest. Thus, we further tune it by the hyperparameter optimization. We use a custom grid search function to find out the optimized hyperparameters, which are l_2 penalty, liblinear solver, and C (inverse of regularization strength) of 4.0.

We notice that the recall score, which is mainly affected by the unbalanced classes amounts as the fraudulent data has only 800. In order to alleviate this effect, we experiment three methods. First, we adjust the prediction threshold on the ROC curve in Fig. 4. Under the threshold of 0.01, the maximum recall score of the True Positive can be the 0.94 and the False Positive is 0.05. However,

the threshold is way too low that we don't think it would generalize in other datasets. Thus, we average the optimized AUC thresholds for both the precision and the recall score, and set the tuning threshold as 0.35.

Second, we use the downsampling method by randomly sampling 2000 from the total 18000 non-fradulent dataset. We experiment several downsampling size such as 1000, 5000, 10000 for the non-fraudulent dataset, while the size 2000 tends to have the most balanced effect on the recall and the precision. We note that the balanced dataset still does not contain 1:1 of these two classes samples since downsampling to around 1000 would make the performances drop significantly, which is probably due to insufficient training input.

Third, we use the class weight that can be set in the model building time. We set the class weight to the corresponded sample amounts in two classes. Table 3. reflects the results of the combination of the second and third methods. We can notice that the recall score increases among all the models and the F1 score also increases among the random forest, logistic regression, and XGB.

Table 2. Traditional Machine Learning Models Evaluation

Model Names	Precision	Recall	F1 Score	Weighted F1 Score
Decision Trees	0.92	0.86	0.89	0.96
Random Forest	0.99	0.84	0.90	0.98
Logistic Regression	0.98	0.86	0.91	0.98
XGB	0.97	0.73	0.80	0.97
Logistic Regression with tuning*	**0.99**	**0.88**	**0.92**	**0.99**

Table 3. Traditional Machine Learning Models Evaluation with Balanced Sample and Class Weight

Model Names	Precision	Recall	F1 Score	Weighted F1 Score
Decision Trees	0.84	0.89	0.87	0.88
Random Forest	0.92	0.90	0.91	0.92
Logistic Regression	0.92	0.92	0.92	0.93
XGB	0.93	0.88	0.90	0.92
Logistic Regression with tuning*	**0.93**	**0.92**	**0.92**	**0.93**

3.2 Deep Learning Models

We then experiment the deep learning based models including the LSTM [4], Bi-directional LSTM [7], and BERT [3]. All the models are trained using the binary cross-entropy loss, ADAM optimizer, batch size of 64, and total 5 epochs for training. We use the same metrics in the evaluation of their performances as the machine learning models. Rather than the Spacy tokenizer to directly

transform the text, we utilized the pre-trained word-embedding illustrated in the section 2.3 and we also create our own word embeddings by one hot embedding and padding. The results indicate that these two word-embedding methods don't have much difference.

LSTM (Long short-term memory) [4] is an artificial recurrent neural network architecture that is proved to be effective in modeling contextual sequence input such as text. Unlike standard feedforward neural networks, LSTM has feedback connections. It can not only process single data points, but also entire sequences of data.

The Bi-directional LSTM [7] trains two instead of one LSTMs on the input sequence and passes the connection in two directions (future to past and past to future), which can only happen when all timesteps of the input sequence are available.

The BERT (Bidirectional Encoder Representations from Transformers) [3] is developed by Google in 2018, and is the state-of-the-art model in many NLP tasks. It utilizes the attention mechanism, which is described in the formula 1. This formula shows the matrix-wise attention weights' computation. We will matrix multiply the query Q with all keys K, divide each by the square root of the keys of dimension d_k, apply a softmax function, and matrix multiply to the values V. Multi-Head Attention is computed by the cancatenation and linearly projection of several attention computation with linearly projected queries. keys, and values in parallel. This helps the model learn contextual relations between words (or sub-words) in a text. It also utilizes the encoder part from the Transformer [8] and input the entire sentences that have been replaced around 15 percent words as masked tokens.

$$Attention(Q, K, V) = softmax(\frac{QK^t}{\sqrt{d_k}})V \tag{1}$$

From the Table 4., we notice that the BERT out performs all other models, and the final F1 score slightly outperform the hyperparameter optimized logistic regression. Its self attention mechanism and pre-trained weights from the Transformer boosts its context understandings. Besides, we also experiment with the method 23 above and the results are in the Table 5. We notice that the balanced sample and class weights don't benefit the deep learning models' performance while some of the metrics such as the F1 score even decrease.

Table 4. Deep Learning Models Evaluation

Model Names	Precision	Recall	F1 Score	Weighted F1 Score
LSTM [4]	0.88	0.77	0.82	0.95
Bi-directional LSTM [7]	0.89	0.81	0.85	0.97
BERT [3]	**0.94**	**0.91**	**0.93**	**0.99**

Table 5. Deep Learning Models Evaluation with Balanced Sample and Class Weight

Model Names	Precision	Recall	F1 Score	Weighted F1 Score
LSTM [4]	0.84	0.78	0.80	0.83
Bi-directional LSTM [7]	0.84	0.82	0.82	0.85
BERT [3]	**0.93**	**0.92**	**0.92**	**0.93**

4 Discussion and Conclusion

In this paper, we tackle the task of fraudulent job predicting using the data collected from the internet and various job platforms [9]. We first preprocess the datasets using tokenizers, custom word embeddings, and pretrained word embedding [6]. We experiments the traditional machine learning models including decision tree, random forest, logistic regression, and XGB, and the deep learning based models including LSTM, Bi-directional LSTM, and BERT. We evaluate closely on the balanced and unbalanced datasets by setting the class weights and downsampling. The BERT achieves the overall best performances in different metrics. However, there exists the computation and resource trade-off that the logistic regression, which is the best model from the tradition machine learning models to tackle this task, usually requires much less time and computation during the training. However, in order to achieve such performance, the logistic regression's hyperparameters need to be optimized by the grid search.

In the future, if any industry wants employ these models to the practical circumstances or other related datasets about the fraudulence job posts prediction, we suggest to use the hyperparameters tuned logistic regression or the BERT models with transfer learning.

References

1. Chen, T., Guestrin, C.: XGBoost: A scalable tree boosting system. In: Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. pp. 785–794. KDD '16, ACM, New York, NY, USA (2016). https://doi.org/10.1145/2939672.2939785, http://doi.acm.org/10.1145/2939672.2939785
2. Cox, D.R.: The regression analysis of binary sequences. Journal of the Royal Statistical Society: Series B (Methodological) **20**(2), 215–232 (1958)
3. Devlin, J., Chang, M.W., Lee, K., Toutanova, K.: BERT: Pre-training of deep bidirectional transformers for language understanding. In: Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers). pp. 4171–4186. Association for Computational Linguistics, Minneapolis, Minnesota (Jun 2019). https://doi.org/10.18653/v1/N19-1423, https://www.aclweb.org/anthology/N19-1423
4. Hochreiter, S., Schmidhuber, J.: Long short-term memory. Neural computation **9**(8), 1735–1780 (1997)
5. Honnibal, M., Montani, I., Van Landeghem, S., Boyd, A.: spaCy: Industrial-strength Natural Language Processing in Python (2020).

https://doi.org/10.5281/zenodo.1212303, https://doi.org/10.5281/zenodo.1212303
6. Pennington, J., Socher, R., Manning, C.D.: Glove: Global vectors for word representation. In: Empirical Methods in Natural Language Processing (EMNLP). pp. 1532–1543 (2014), http://www.aclweb.org/anthology/D14-1162
7. Schuster, M., Paliwal, K.: Bidirectional recurrent neural networks. Trans. Sig. Proc. **45**(11), 2673–2681 (Nov 1997). https://doi.org/10.1109/78.650093, https://doi.org/10.1109/78.650093
8. Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, L., Polosukhin, I.: Attention is All you Need. Advances in Neural Information Processing Systems **30** (2017)
9. Vidros, S., Kolias, C., Kambourakis, G., Akoglu, L.: Automatic detection of online recruitment frauds: Characteristics, methods, and a public dataset. Future Internet **9**(1) (2017). https://doi.org/10.3390/fi9010006, https://www.mdpi.com/1999-5903/9/1/6