Pages 1–7
2023

Identifying Fraudulent Job Postings Online using Natural Language Processing Machine Learning Models

Morris Chang, Gina Shuyao Wang, Eric Yuxin Miao
INFO 251
University of California, Berkeley
May 5 2023

Keywords: Job, Employment, Online, Fraud, Scam, NLP, Machine Learning, Detection

CONTENTS

1 INTRODUCTION

The development of information and communication technology in the past decades has changed human daily life in different ways. In the past, information was often communicated through physical means, such as papers and books. The development of the internet and other technology has pushed the content that used to be published on paper to migrate to online platforms, including news, advertisements, and job postings. In the past, newspapers were often the first place where people could look for new job postings and opportunities, and this often included a very minimal amount of information due to the high prices and limitations in space. The introduction of the Internet and employment search websites have increased the accessibility of job postings and enabled employers with limited budgets and resources to acquire the talents they needed. However, this flexibility and reduction in barriers have also led to an increase in online fraudulent job postings.

Fraudulent job postings online usually carry malicious goals and intentions, including the attempt to acquire confidential personal information from the applicants or solicit payments from applicants as application fees. The personal information acquired by the fraudulent job postings is used as a part of a larger scam, or illegally sold to third parties. This may endanger the applicant's privacy or accidentally involve the application in illegal activities. In addition, certain applications attempt to solicit payments from job applicants as "application fees", and this often leads to financial losses for the applicant.

Therefore, developing a system to detect fraudulent job postings is crucial to protect job seekers and prevent the spread of these malicious activities on job posting platforms. The employment posting websites would be able to use such models to predict, filter, and remove fraudulent postings. This project aims to build and develop a Natural Language Processing (NLP) model that could analyze the textual descriptions of job postings and identify any suspicious patterns or anomalies that may indicate fraudulent activity. This includes utilizing different kinds of machine learning models and algorithms to identify patterns or anomalies, and this system can help job seekers to identify legitimate job postings and avoid falling victim to fraudulent activities. Furthermore, this research can contribute to the development of better fraud detection systems in other domains as well.

2 DATA ACQUISITION

The dataset that would be used in this project is the Employment Scam Aegean Dataset (EMSCAD) as shown in Table 1. It is published by the University of the Aegean's Laboratory of Information and Communication Systems Security. This dataset is publicly available for download and contains 17880 real-life job postings from online platforms, including 17014 legitimate job postings and 866 fraudulent job postings. The instances in the EMSCAD dataset were manually annotated and classified into two categories.

Table 1. Data Type in the dataset

Type	Name	Type	Name
String	Title	**Binary**	Telecommuting
	Location		Company logo
	Department		Questions
	Salary range		**Fraudulent**
HTML fragment	Company profile	**Nominal**	Employment type
	Description		Required experience
	Requirements		Required education
	Benefits		Industry
			Function

3 EXPLORATORY DATA ANALYSIS

During the exploratory data analysis phase, our team explored the different features within the dataset, and many of the features contained too many null values or were too noisy in general. The string features,

such as title, location, and salary range of the job postings did not show any significant relationships with fraudulent postings. In addition, there are significant amounts of null values in features such as departments and salary ranges, making it difficult to interpret the features.

In addition, binary and categorical features, such as telecommuting, company logo, and questions also did not show significant changes in regards to fraudulent postings. The nominal values such as industries and functions did contain a few items that had a more significant fraudulent frequency than other items in the dataset.

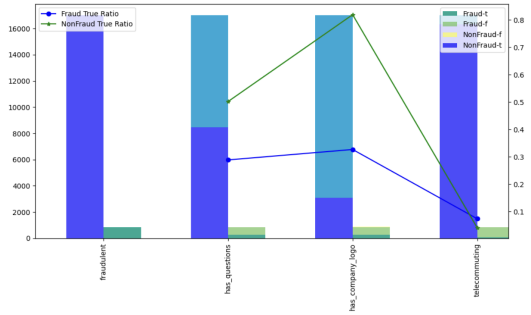

Figure 1. EDA: Binary Features and Fraudulent Label

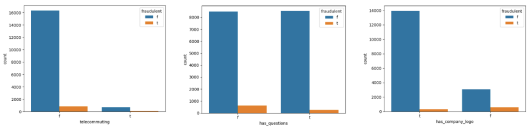

Figure 2. EDA: Individual Binary Features and Fraudulent Label

Figure 1 and Figure 2 shows that there are no significant differences in fraudulent postings for the three categorical variables, as the number of fraudulent postings does not change significantly within these features. The only feature with a higher frequency of fraudulent postings is telecommuting, when the feature is 'True', it is less likely that it would be a fraudulent job posting.

In addition to the features discussed above, the features 'functions' and 'industry' have certain inputs that have higher fraudulent postings compared to other inputs, as shown in Figure 3 and Figure 4. For example, job postings in the oil and energy, and accounting industries have a higher number of fraudulent posts, while job functions such as administrative, and engineering, also have a higher number of fraudulent postings.

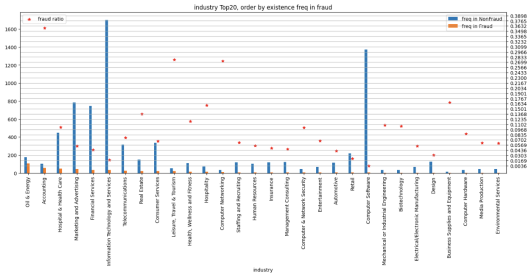

Figure 3. EDA: Industry Feature

The original dataset is composed of 16 features, including five nominal features, four categorical features, four string features, and four features

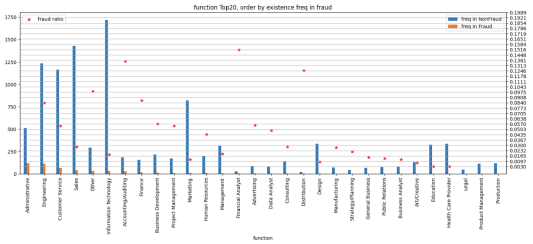

Figure 4. EDA: Job Function Feature

that contained HTML fragments. In order to simplify the modeling process, our team preprocessed the data in the following ways.

1. Removed binary and categorical variables from the dataset, as most of the categorical variables, such as if a company logo is attached or if the job involves telecommuting does not significantly affect the chances of fraudulent postings.
2. Concatenated nominal and string variables with HTML variables into a single feature that would be processed afterward.
3. Removed duplicate contents and values from the new feature. For example, if the job title is included multiple times originally in different parts of the dataset, only one instance of it would be kept.
4. Removed HTML tags, and fragments from the text, stop words, Unicodes, and punctuation using the NLTK package.
5. One-hot encoded the fraudulent column into 0 for legitimate postings and 1 for fraudulent postings.
6. The dataset is also divided into training and testing datasets by using sci-kit-learn's built-in train_test_split function, using 75% of the dataset as training data, and 25% of the dataset as testing data, with a random state of 42.

4 MODELING

Upon completion of the exploratory data analysis process, our team began the modeling process by building baseline models with the bag-of-words approach with traditional models such as logistic regression and random forests. The following section would discuss the modeling process and the results in each of the stages.

The Bag of Words (BoW) approach is a natural language processing technique to represent texts and words, disregarding order, and grammar. The string text is transformed into a matrix that represents the frequency of each word in the document. In other words, Bag-of-Words is a way of representing texts in a numeral format by their frequency.

4.1 Approach 1: Customized Top n-words Model

The first approach that our team pursued is evaluating the fraudulent postings in the training set data, extracting all of the strings and texts in the dataset, and counting the number of occurrences for each of the words. The words are sorted by the number of occurrences in the dataset, and stored in a list. The top 1200 words that are often seen in fraudulent postings in the training set are stored in a list.

The function words_in_text would then take in a list of words, and a list of texts, and would count the number of occurrences for each of the words in the texts, and return an array matrix. The training set array matrix is then used to fit and train the model, and the scores are calculated based on the results from the testing set.

In order to experiment and test the relationship and changes in the top number of words in the dataset, our team attempted to fine-tune the model by using a different number of words to train the model, and the results

on the test set predictions are shown in Table 2. As shown in the table, the precision and recall reach a certain threshold after using 900+ words in the model, and the improvement would slow down once passing this threshold. However, it is still worth noting that the increase in words does lead to an increase in precision and recall.

Table 2. Results of the TOP-n word model

	Accuracy	Precision	Recall	F1
50	0.950	0.013	0.429	0.026
100	0.958	0.238	0.757	0.361
200	0.965	0.395	0.815	0.531
300	0.967	0.489	0.773	0.599
400	0.972	0.556	0.821	0.663
500	0.974	0.655	0.798	0.719
600	0.975	0.673	0.798	0.730
700	0.976	0.664	0.813	0.731
800	0.976	0.682	0.800	0.736
900	0.977	0.695	0.812	0.749
1000	0.978	0.700	0.825	0.757
1100	0.979	0.709	0.840	0.769
1200	0.979	0.727	0.839	0.779

4.2 Approach 2: Using all words with CountVectorizer

With the results from the previous section, our team wanted to attempt to increase the precision and recall score by using all the words that are included in the training set, and by turning all the words in the training set into vectors and counting the frequency of each of those words. The testing set would also be transformed based on the same CountVectorizer.

The resulting matrix of word counts would be inputted into machine learning models, such as logistic regression to classify if the job posting is fraudulent. The function also has the built-in capability to remove stop words or convert all text to lowercase. In addition, the function also allows different inputs of n-grams, which would help to preserve patterns and sequences of the text. Our team attempted three types of n-grams, the first with single words, single and two words, and only two words. The result of the model's prediction on the test set is shown in table 3, as indicated, the precision and recall are performing better compared to the previous model, especially when both 1-grams and 2-grams are taken into account. This means that reversing the order and sequence of the text would be helpful in building the model. This has led to the continued development of the LSTM and BERT models in the following sections. However, before using neural networks, our team attempted to tackle the problem of imbalance in the dataset by manually creating a random balanced sample and conducting a similar analysis.

Table 3. Results of CountVectorizer n-gram model

	Accuracy	Precision	Recall	F1
(1, 1)	0.986	0.762	0.950	0.846
(1, 2)	0.987	0.749	0.988	0.852
(2, 2)	0.984	0.682	1.000	0.811

4.3 Approach 3: LSTM Model

The LSTM (Long Short-Term Memory) model is designed to overcome the gradient vanishing problem in traditional RNNs with its memory cell that can choose to forget or remember information over time. The LSTM model is useful in tasks that require the processing of sequential data, such as text data that have been shown in previous sections.

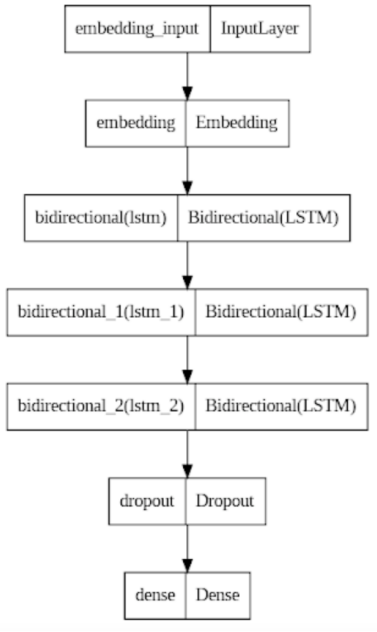

Figure 5. The structure of the LSTM model

The hyper-parameters that are chosen to be tuned in our model are embedding vector size, LSTM_layers, and the class weight of our prediction.

1. Embedding Size: [50,100,150,200]
2. LSTM_layer = [1,2,3]
3. Class Weight:
    a. Class 0: 1.0, Class 1: 1.0
    b. Class 0: 1.0, Class 1: 1.5
    c. Class 0: 1.0, Class 1: 2.0
    d. Class 0: 1.0, Class 1: 3.0
    e. Class 0: 1.0, Class 1: 5.0

The application of class weight is to help the model to better capture the minority but important scam class in the imbalance dataset. The model was tuned to find an optimal weight to give for each class. After grid search and cross-validation, the best parameters were when the embedding dimension was set to 50, with 3 LSTM layers, and a class weight of 1.0 for class 0, and 2.0 for class 1. Below is part of our cross-validation metric in table4.

The results show that the LSTM is good at increasing the recall, decreasing the false negative, and preventing scams from escaping from the screening. However, it is not good at precision, which means that it mis-classifies a lot of regular posts to be scams. This would lead to people missing opportunities.

4.4 Approach 4: BERT Model

BERT (Bidirectional Encoder Representations from Transformers) is a state-of-the-art pre-trained language model developed by Google researchers in 2018. It is based on the Transformer architecture and can be fine-tuned for various natural language processing tasks, such as text classification, question answering, and language translation.

We started with the pre-trained BERT model from Google and try to fine-tune it with our dataset. There are three hyper-parameters that are

Table 4. LSTM cross-validation results

Embedding	LSTM layer	Class_1	Recall	Precision	Accuracy	F1-Score
50	1	1	0.980	0.614	0.751	0.755
50	1	1.5	0.979	0.677	0.766	0.801
50	1	2	0.978	0.655	0.749	0.784
50	1	3	0.977	0.695	0.751	0.812
50	2	2	0.977	0.646	0.737	0.778

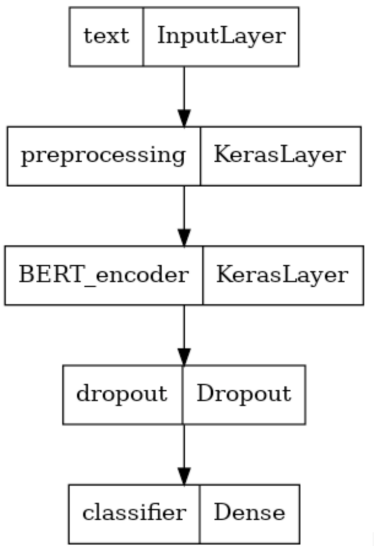

Figure 6. The structure of the BERT model

chosen to be further tuned, the basis BERT model, the dropout rate, and the class weight.

1. Model:
   a. 'small_bert/bert_en_uncased_L-4_H-512_A-8'
   b. 'small_bert/bert_en_uncased_L-8_H-512_A-8'
   c. 'bert_en_uncased_L-12_H-768_A-12'
2. Dropout_rate = [0.1,0.2,0.3,0.5,0.6]
3. Class Weight:
   a. Class 0: 1.0, Class 1: 1.0
   b. Class 0: 1.0, Class 1: 1.5
   c. Class 0: 1.0, Class 1: 2.0
   d. Class 0: 1.0, Class 1: 3.0
   e. Class 0: 1.0, Class 1: 5.0

The BERT models require significant computational resources to run, and one epoch may take up to one hour on a local device with CPU, and around 15 to 20 minutes on online platforms such as Google Colab with GPU acceleration. However, these online platforms usually carry certain limitations in credit or hours that prevent our team from testing all the parameters. Overall, our team was able to test 8 sets of parameters. The best combination of hyper-parameters is using 'small_bert/bert_en_uncased_L-4_H-512_A-8' with a dropout rate equal 0.2 and a class weight of 2 for scam labels. The cross-validation test results are below in table5. It has shown that BERT models have significantly reduced false positives, which would help to ensure that legitimate job postings are not accidentally marked as fraudulent postings. On the contrary, the BERT model performs worse on recall compared to LSTM, which means that it lets go of some of the real scams. Overall speaking, the f1 score for BERT is lower than that of the LSTM if we take both precision and recall into consideration.

Table 5. BERT cross-validation results

Dropout_r	Class_1	Recall	Precision	Accuracy	F1-Score
0.1	1	0.722	0.953	0.984	0.646
0.1	1.5	0.789	0.912	0.986	0.656
0.1	2	0.789	0.903	0.985	0.672
0.1	3	0.735	0.965	0.985	0.629
0.1	5	0.717	0.976	0.985	0.623
0.2	1	0.785	0.951	0.987	0.679
0.2	1.5	0.812	0.923	0.987	0.691
0.2	2	0.821	0.943	0.989	0.694

4.5 Approach 5: Balanced Dataset

In order to address this issue, we further did the oversampling of the minority class by 1000, the smallest values which helps avoid the effect of over-fitting from the following methods described and give us the most balanced effect of the precision score and recall score. Then we experiment with the following six models: logistic regression, KNN, SVM, random forest, and Naive Bayes. However, we noticed that there may exist some over-fitting issues with logistic regression, SVM, decision tree, and random forest model with 100% precision score.

Table 6. Balanced Sample Dataset with Traditional Models

Model	Accuracy	F1-score	Precision	Recall
LogisticRegression	91.863	91.593	90.393	92.825
KNeighborsClassifier	71.306	76.325	62.974	96.861
SVC	90.15	89.64	90.045	89.238
DecisionTreeClassifier	85.439	85.153	82.979	87.444
RandomForestClassifier	91.863	91.284	93.427	89.238
MultinomialNB	89.507	89.087	88.496	89.686

In the case of detecting fake job postings, we want to minimize both false positives and false negatives since it would both neglect specific kinds of goals. A false positive occurs when the model predicts a job posting is fake when it is actually legitimate, while a false negative occurs when the model predicts a job posting is legitimate when it is actually fake. In general, minimizing false positives is important because avoids flagging legitimate job postings as fake and potentially causing harm to the organization or individuals posting the job. On the other hand, minimizing false negatives is also important because failing to detect a fake job posting can lead to negative consequences such as wasting time and resources on the application process or potentially putting applicants at risk.

As a result, our team decided to do hyper-parameter tuning using cross-validation by grid search function which allows us to find the best estimator to fix the logistic regression and random forest model since they have the best performance in all metrics. Then we adjusted the prediction threshold based on the ROC curve and produced the optimized AUC threshold. For the random forest model, the best parameters were when max depth is 30, min samples split is 10, and n_estimators is 50. The optimized AUC threshold at 0.55 and 0.87 maximum recall score with 0.02 false Positive rate after evaluating model performance. While for the logistic regression model, the best parameters with 0.01 C, l2 penalty, and liblinear solver. The optimized AUC threshold at 0.475 and 0.94 maximum recall score as well as 0.076 false positive rates.

Table 7. Fine-Tuned Hyper-parameters with Balanced Sample Dataset Models

Model	Accuracy	Precision	Recall	F1-Score
Logistic Regression	0.931	0.914	0.940	0.927
Random Forest Classifier	0.994	0.957	0.921	0.939

5 CONCLUSION

Overall, the numerous models that our team has created have shown different strengths and weaknesses, and the search for better performance in the precision score, and recall score has guided our team from traditional models such as logistic regression and random forest to more complicated LSTM and BERT neural network models. Due to the imbalance of dataset samples, the model's performance in accuracy has not been a top priority, rather our team has focused on precision and recall scores to evaluate our models. Using the f1-score as an indicator that takes both precision and recall into consideration, the results show that the CounterVectorizer version of Logistic Regression outperformed other models.

Table 8. Overall All Models Test Results

Model	Precision	Recall	Accuracy	F1-Score
Top n-words Logistics Regression	0.726	0.839	0.979	0.778
CountVectorizer Logistics Regression	0.749	0.988	0.988	0.852
LSTM	0.637	0.953	0.980	0.763
BERT	0.943	0.821	0.989	0.694

In the early stages of the project, our models in logistic regression and random forest provided reasonable scores with both the top n-words, and CountVectorizer approach, with the highest precision score of 0.957, and recall score of 0.98 in different models. In the more advanced models, BERT was able to reach a precision score of 0.637 and a recall score of 0.953, and LSTM was able to reach a precision score of 0.943 and a recall score of 0.821, but with significantly more computational resources and time. In fact, BERT models are able to reduce false positives, while LSTM models have reduced false negatives, but our attempts have not shown success in achieving both in the same model. Thus, a more complex structure could be helpful in this case for example applying BERT as a bedding layer and using LSTM to make further predictions. In this way, the model should be able to capture more sophisticated relationships.

Therefore, when there are limited computation resources and time, the traditional models and approaches may be more suitable, but when the limitation in such factors are not considered, more complicated models would provide better results and avoid large numbers of both false positives and false negatives, as this may lead to real postings being disregarded, or fraudulent postings being missed.

6 CHALLENGES, CAVEATS, AND OPPORTUNITIES

There were several challenges that our team faced during the process, and these challenges may have contributed to inaccuracy in the models. One of the first challenges that were discovered was the imbalance in the original dataset, where the fraudulent emails accounted for less than 10% of the instances. Our team manually created a balanced dataset in the later steps in the modeling processes, but this still demonstrates that the lack of number and diversity of fraudulent job postings to train on could limit the word and content that the model is able to learn. A potential approach to mitigate this challenge is to input more fraudulent postings and expand the dataset until the dataset is more balanced.

Another challenge that our team has encountered in the process is the long training time required from the pre-trained BERT model. Each epoch in our attempt to fine-tune the pre-train model would take more than 20 hours to run, increasing the difficulty to experiment and find the best parameters for the model.

In addition, the format and content of online job postings have also changed over the years, the original dataset is collected from 2012 to 2014, which is almost close to ten years ago. This could indicate that our model may not incorporate a more recent format and wording of job postings. This may prevent the model's ability to accurately predict more recent job postings, and in order to improve in this aspect, it would require additional data to be collected and trained continuously.

One of the issues and questions that our team faced is the choice to either prioritize precision or recall scores for our models. Due to the imbalanced original dataset, the accuracy of the model would intuitively be high, since even when all the job postings are predicted as legitimate job postings, the accuracy would be over 90%. Therefore, accuracy would not be the best-measuring metric for this model, but the choice and balance of precision and recall is also important factor that would need to be considered. This is due to the fact that if only the recall score is optimized, it may lead to a low recall, meaning that a lot of legitimate postings would be inaccurately labeled as fraudulent postings. In contrast, if only the precision score is prioritized, then the chance of fraudulent emails being missed and labeled as legitimate emails would be higher. Therefore, in order to find a balance between false positives and false negatives, our team has prioritized models that are able to produce reasonable scores for both precision and recall.

7 CONTRIBUTION

Part	Morris Chang	Eric Yuxin Miao	Gina Shuyao Wang
Data Acquisitions	v	v	v
Exploratory Data Analysis	v	v	v
Data Preprocessing:		v	
Model Top n-word Model	v		
Model CountVectorizer	v		
Model Balanced Dataset Finetuning			v
Model LSTM		v	
Model BERT		v	
Video Editing	v		
Slide deck Final Report	v	v	v
Notebook Organization Shell script GitHub Repository		v	

REFERENCES