Predicting Fraudulent Job from Job Postings

Lili Wang, 21741780
Zihang Li, 22748446

October 26, 2023

Abstract

This project is to employ machine learning techniques for the prediction of fraudulent job postings. Three distinct models were trained using the dataset "Real/Fake Job Posting Prediction", which was vectorized by TF-IDF vectorization method. The performance of these models was evaluated using five different metrics. The Random Forest model emerged as the top-performing one. For further improvement, additional vectorization techniques and classification algorithms will be explored.

GitHub Link: Project code

1 Introduction

We were working on predicting whether a job posting is fraudulent, given its title, content, salaries, etc. as input data. This project is a binary classification task aimed at identifying fraudulent job postings from authentic ones based on the textual content of job descriptions.

The proliferation of online job boards has revolutionized the job-seeking process, but it has also given rise to a surge in fraudulent job postings. These deceptive listings can lead to financial loss and emotional distress for job seekers. In response, this report introduces a cutting-edge machine learning-based binary classification system designed to identify and filter out fraudulent job postings. This system utilizes advanced NLP techniques for text analysis and classification. We delve into the model's core components, the datasets used, and evaluation metrics, aiming to provide job seekers with a robust tool to navigate the digital job market securely. Our goal is to ensure a safer and more trustworthy employment ecosystem for all.

2 Data set

We chose Real/Fake Job Posting Prediction as our dataset, which consists of 17880 job postings that are either authentic or fraudulent. This job postings dataset includes information of each postings, such as title, location, salary range, etc., and the postings' content. Using this information as input, the model will predict if it is fraudulent, i.e. the label is variable **"fraudulent"**. The quality of this dataset is high, with more complete data for each entry and a uniform structural characterization of the data. For this purpose, there is not yet any other dataset to choose. There are several other job postings datasets, but they do not provide information on whether it is fraudulent.

3 Preprocessing

In the Preprocessing part, we aimed to prepare the textual data for subsequent analysis and modeling. To do this, we selected five essential text features from our dataset, **"ti-**

tle", **"company_profile"**, **"description"**, **"requirements"**, and **"benefits"**. These features potentially have the most significant impact on the labeling process.

Our first step was to handle missing values. Missing values in these text features were replaced with the string "unknown" to ensure that every record had a valid text entry.

Then we performed text tokenization and lemmatization to standardize and clean the text data. Tokenization aimed to break down the text into individual words as tokens. We converted all tokens to lowercase to ensure case consistency and removed punctuation marks to focus on meaningful content. We also removed stop words and common words like "and" or "the" which do not carry substantial information, to improve the quality of the text data. Lemmatization aimed to reduce words to their base or root form. This process was accomplished using the WordNetLemmatizer, which allowed us to transform words into their common lemma, such as "running" and "ran" to "run."

4 Text Vectorization

In this section, we aimed to convert textual data into numerical representations which can be used for machine learning models.

4.1 TF-IDF

For the TF-IDF vectorization method, we used the scikit-learn library to compute TF-IDF features for five key text features: "preprocessed_title", "preprocessed_company_profile", "preprocessed_description", "preprocessed_requirements" and "preprocessed_benefits". We limited the feature space to the top 100 features to manage dimensionality effectively.

TF-IDF represents the importance of each term within a document relative to the entire corpus. This method allows us to capture the uniqueness of terms in a document and their significance in distinguishing one document from another.

4.2 BERT

For BERT based vectorization, we used the pretrained deep learning model, "bert-base-uncased" which fine-tuned on large text corpora. Utilizing the Hugging Face Transformers library, we tokenized and encoded text from the same five text features mentioned earlier: "preprocessed_title", "preprocessed_company_profile", "preprocessed_description", "preprocessed_requirements" and "preprocessed_benefits".

After tokenization and encoding, we extracted word embeddings from the BERT model. These embeddings represent contextual information for each word in the text, resulting in rich, context-aware representations.

To optimize the efficiency of our computations, we calculated the mean of word embeddings for each feature, in order to obtain the reduced dimensionality while retaining valuable information.

5 Model training

In the model training phase, our goal was to explore the performance of chosen machine learning models, and compare the impact of using the TF-IDF vectors and BERT embeddings that we generated in the previous sections.

5.1 TF-IDF-Based Models

For TF-IDF vectorization, we trained three classical machine learning models: Logistic Regression, Naive Bayes, and Random Forest. Before diving into training, we split our dataset into an 80% training set and a 20% test set to assess model performance accurately. This splitting strategy ensured that our models were evaluated on unseen data, guarding against overfitting.

5.2 BERT-Based Neural Network

In contrast, for the BERT-based vectorization, we employed a Neural Network architecture. BERT embeddings carry rich contextual information, and to make the most of this information, we concatenated the embeddings from different text features. This yielded a multi-dimensional input matrix for our Neural Network.

During the training of our Neural Network (NN) model, we encountered an error with the message: "TypeError: Unable to convert function return value to a Python type! The signature was () - handle." This error referred to a compatibility issue between the version of TensorFlow used and the GPU driver or hardware configuration. Consequently, we were unable to obtain the results of the NN model based on BERT embeddings. We attempted to utilize Colab's GPU, but the embeddings we obtained had a size of approximately 20GB, even after calculating the mean embeddings. This size was too large to load in Colab, and unfortunately, we couldn't find an alternative method to reduce the dimensionality at this time.

6 Evaluation

Since fake job posting detection is a classification task, we chose accuracy, precision, recall, F1-Score and ROC-AUC score as the evaluation methods.

The following table summarizes the evaluation results for three different machine learning models: Logistic Regression, Naive Bayes, and Random Forest.

	Logistic Regression	Naive Bayes	Random Forest
accuracy	0.9717561521252797	0.9460290827740492	0.979586129753915
precision	0.45748443499101665	0.2525107979412443	0.5970149509481706
recall	0.8636363636363636	0.46774193548387094	0.990909090909091
f1	0.6529209621993127	0.4741144414168937	0.7491408934707904
roc_auc	0.9194119498504958	0.7200066609572747	0.9850679326443897

- Logistic Regression performs well in terms of accuracy and recall, indicating it can correctly classify a high percentage of instances and effectively identify positive cases. However, its precision is relatively low, suggesting it may generate some false positives.
- Naive Bayes has a lower accuracy compared to the other models and has the lowest precision among the three. It identifies fewer positive cases with high precision but misses some positive cases, resulting in lower recall.
- Random Forest demonstrates outstanding performance across all metrics. It has high accuracy, precision, recall, F1-score, and ROC-AUC score, indicating its robustness in correctly classifying instances and distinguishing between the two classes.

In summary, Random Forest stands out as the best-performing model, providing the highest accuracy, precision, recall, F1-Score, and ROC-AUC score. Logistic Regression also

performs reasonably well, especially in terms of recall, while Naive Bayes achieving an acceptable accuracy, falls short in terms of precision and recall compared to the other models.

7 Future work

While the Random Forest model demonstrates strong overall performance, it's worth noting that the average precision score across these models remains relatively modest. To address this, we will continue our efforts to explore improved methods for reducing the dimensionality of the embeddings generated by BERT. Additionally, we will seek more robust hardware configurations to facilitate the successful training of neural network models and facilitate meaningful result comparisons. Moreover, we plan to diversify our approach by experimenting with alternative vectorization techniques beyond TF-IDF, which primarily emphasizes word frequency. Furthermore, exploring a wider range of classification algorithms is also on our agenda to further enhance model performance.

Good-performing models can be used into practice. We can make them into pre-trained models and make browser plugins, which can implement real-time rating of job postings. Also, this can be used to identify badly-written postings, helping people write better postings.