Fake Job Posting Detection

CS6120: Natural Language Processing
Neha Reddy Palnati (002337926)
Krishnan Narayanan (002354016)
GitHub

Contents

Abstract

The rise of online job platforms has, unfortunately, made it easier for scammers to post fraudulent job advertisements, posing serious risks to job seekers and undermining trust in digital recruitment. This literature survey explores how machine learning and deep learning techniques are being applied to address this issue. Key focus areas include feature extraction, handling class imbalance, and performance evaluation using metrics such as precision, recall, and F1-score. Our review finds that while classical machine learning models are more interpretable and easier to deploy, deep learning techniques often achieve higher accuracy by capturing complex patterns in textual data. We also highlight several research gaps, such as the need for larger and more diverse datasets, as well as improved model explainability. The survey concludes with recommendations for future research, including real-time detection, behavioral analysis, and the integration of advanced models like transformers.

1 Introduction

The digital transformation of job hunting has been a double-edged sword. While platforms like LinkedIn and Indeed have made job searches more efficient and accessible, they have also opened the door to a troubling rise in fake job postings. These scams can deceive job seekers into sharing personal information, paying unnecessary fees, or even becoming victims of identity theft.

Traditional detection methods such as keyword filters or manual reviews are increasingly ineffective against these evolving threats. In contrast, machine learning (ML) and deep learning (DL) approaches are emerging as powerful alternatives. These technologies can analyze linguistic patterns, detect metadata anomalies, and identify subtle cues across millions of postings cues that humans might easily overlook, such as inconsistent company profiles or suspicious salary ranges.

2 Methodology

This study follows a structured approach to detect fake job postings using machine learning and deep learning models. The steps involved are:

1. **Dataset**: The dataset was sourced from a public repository containing real and fake job postings from Kaggle. It includes features like job title, description, location, and a binary label indicating whether a job is fraudulent. **LINK TO DATASET**

2. **Data Preprocessing**: We cleaned and prepared the data by handling missing values, removing irrelevant text, and converting text into numerical form using techniques like TF-IDF. The dataset's significant imbalance, with more real postings than fake ones, was addressed to ensure fair model training. The development cycle incorporated SMOTE (Synthetic Minority Oversampling Technique) to generate synthetic samples of the minority class (fake postings), balancing the dataset. Additionally, class weighting was applied during training to penalize misclassification of the minority class more heavily, ensuring models focused on detecting fraudulent postings effectively.

3. **Exploratory Data Analysis (EDA)**: Statistics and visualizations were used to understand patterns in the data. This included analyzing word frequency, job location trends, and description length.

4. **Modeling**: Several models were trained and compared: A variety of machine learning and deep learning models were implemented and evaluated to classify job postings as real or fake.

    - **Logistic Regression:**
      The Logistic Regression model was developed as a foundational baseline for the binary classification task of detecting fake job postings, leveraging its simplicity and interpretability. The model was instantiated using the LogisticRegression class from scikit-learn, with hyperparameters configured to ensure effective training: max_iter= 1000 was set to allow sufficient iterations for convergence, C=0.1 was chosen to apply moderate regularization and mitigate the risk of overfitting, and class_weight='balanced' was employed to address the dataset's class imbalance by assigning weights inversely proportional to class frequencies, thus ensuring equitable learning across both real and fake postings. The model was trained on preprocessed training data (X_train_features, y_train), which consisted of TF-IDF vectorized textual features and engineered variables such as description length, following data cleaning and feature extraction steps.
    - **Random Forest:**
      The Random Forest model was developed to capture non-linear relationships and automatically rank feature importance within the heterogeneous dataset of job postings, employing an ensemble approach to enhance predictive capability. The RandomForestClassifier was instantiated with specific hyperparameters: n_estimators=100 to construct 100 decision trees, max_depth=5 to constrain tree depth and prevent overfitting, min_samples_leaf=5 to ensure a minimum of five samples per leaf for robustness, random_state=42 for reproducibility, and class_weight='balanced' to address class imbalance by weighting classes inversely proportional to their frequencies, aligning with the preprocessing use of SMOTE. The model was trained on the preprocessed training data (X_train_features, y_train), which included TF-IDF vectorized features and engineered variables such as description length, following data cleaning and feature extraction.

- **XGBoost:** The XGBoost model was developed to further enhance classification through gradient boosting, focusing on iterative improvement by prioritizing misclassified instances, thereby building upon the capabilities of Random Forest. The XGBClassifier was instantiated with carefully selected hyperparameters: use_label_encoder =False to disable automatic label encoding as labels were preprocessed, eval_metric= 'logloss' to use logarithmic loss as the training evaluation metric, max_depth=4 to limit tree depth and reduce overfitting, learning_rate=0.05 to ensure gradual updates during boosting, reg_alpha=1 and reg_lambda=2 to apply L1 and L2 regularization respectively for model regularization, and scale_pos_weight=sum(y_train==0)/sum(y_train==1) to dynamically adjust the weight of the positive class (fake postings) based on the class imbalance ratio in the training data, ensuring focus on the minority class. The model was trained on the preprocessed training data (X_train_features, y_train)
- **LSTM(Long Short-Term Memory):**
  The LSTM model was developed to capture sequential patterns in job descriptions for fake job posting detection. Text preprocessing involved tokenizing training data (df_train['text']) using a Tokenizer with a 10,000-word vocabulary (MAX_VOCAB_SIZE =10000) and padding sequences to a length of 200 (MAX_SEQUENCE_LENGTH=200) using pad_sequences. Class imbalance was addressed with balanced class weights computed via compute_class_weight. The Sequential model included an Embedding layer (input dimension 10,000, output dimension 200), two Bidirectional LSTM layers (128 and 64 units), Dropout layers (rate 0.4), and dense layers (64 units with ReLU and L2 regularization, sigmoid output). It was compiled with Adam optimizer (learning_rate=0.001), binary cross-entropy loss, and metrics (accuracy, precision, recall, AUC). Training ran for five epochs (batch size 32, validation split 0.2).
- **BERT:**
  The BERT model, adapted as a simplified alternative due to computational constraints, was developed to classify fake job postings by modeling contextual relationships in text data. Class imbalance was addressed by computing balanced class weights using compute_class_weight, creating a dictionary (class_weight_dict) to prioritize the minority class (fake postings) during training. The model, constructed with tf.keras.Sequential, comprised an Embedding layer (input dimension based on tokenizer vocabulary size plus one, output dimension 128, input length max_len), two Bidirectional LSTM layers (64 and 32 units, the first returning sequences), Dropout layers (rates 0.4 and 0.3), and dense layers (32 units with ReLU activation, sigmoid output for binary classification). It was compiled with the Adam optimizer (learning_rate=1e-4), binary cross-entropy loss, and accuracy as the metric. Training was conducted for five epochs (batch size 32, validation split 0.2) on preprocessed data (X_train_ids_tf), with class weights applied to handle imbalance.

3 Results and Discussions

In the evaluation of our fake job posting detection project, five models: Logistic Regression, Random Forest, XGBoost, LSTM, and a simplified BERT were assessed using Accuracy, Precision, Recall, F1-Score, and AUC-ROC on a dataset which had significantly huge class imbalance. Unlike the case in the literature review, XGBoost demonstrated superior performance, achieving the highest AUC-ROC of 0.9597, indicative of its exceptional ability to discriminate between legitimate and fraudulent postings. It also exhibited a robust Recall of 0.8728, correctly identifying 87.28 percent of fake postings a critical metric for this imbalanced dataset while maintaining a balanced F1-Score of 0.4660 and a

Precision of 0.3179, thereby outperforming Random Forest in minimizing false positives. Random Forest, despite achieving the highest Recall at 0.8786, recorded the lowest Precision (0.2386) and F1-Score (0.3753), resulting in a high rate of false positives. Logistic Regression (AUC-ROC: 0.9491, Recall: 0.8671, F1-Score: 0.3989) and BERT (AUC-ROC: 0.9565, Recall: 0.8497, F1-Score: 0.4925) delivered commendable results but fell short of XGBoost's overall balance. LSTM, with the highest F1-Score of 0.6782 and Precision of 0.6743, excelled in reducing false positives; however, its Recall of 0.6821 indicated a higher rate of missed fraudulent postings, and training logs (AUC-3: 0.9977 by Epoch 5) revealed overfitting despite a Dropout rate of 0.6. Given the priority to maximize the detection of fraudulent postings (high Recall) while ensuring strong overall performance (high AUC-ROC), XGBoost is deemed the optimal model.

Table 1: Performance Metrics of Models for Fake Job Posting Detection

Model	**Accuracy**	**Precision**	**Recall**	**F1-Score**	**AUC-ROC**
Logistic Regression	0.8736	0.2591	0.8671	0.3989	0.9491
Random Forest	0.8585	0.2386	0.8786	0.3753	0.9409
XGBoost	0.9032	0.3179	0.8728	0.4660	0.9597
LSTM	0.9687	0.6743	0.6821	0.6782	0.9385
BERT-Simplified	0.9153	0.3467	0.8497	0.4925	0.9565

The BERT and LSTM models were significantly simplified due to computational constraints, impacting their performance compared to traditional models like Logistic Regression, Random Forest, and XGBoost, which is very unlikely. The BERT, lacking the transformer architecture and pre-trained weights, used a basic Sequential model with an Embedding layer, Bidirectional LSTMs, and Dense layers, missing BERT's contextual understanding and leading to a lower F1-Score (0.4925) despite a high AUC-ROC (0.9565). Similarly, the LSTM, with only two Bidirectional layers (128 and 64 units) and learned embeddings, showed overfitting (training AUC-3: 0.9977 by Epoch 5) and a lower Recall (0.6821), despite a strong F1-Score (0.6782). True BERT and LSTM models, with their deep architectures, pre-trained embeddings, and contextual understanding, typically outperform traditional models in NLP tasks by capturing nuanced patterns and long-range dependencies, but these simplifications limited their capacity, resulting in XGBoost (AUC-ROC: 0.9597, F1-Score: 0.4660) outperforming them. Enhancing these models with pre-trained embeddings, deeper architectures, and more regularization could unlock their potential, likely surpassing traditional models in detecting fake postings.

4 Future Enchancements

In order to improve model generality, future research should concentrate on expanding datasets to encompass a broader range of job listings from diverse industries and geographical areas. Furthermore, enhancing the computational environment and interpretability of deep learning models may increase user confidence and practical utility. The efficacy of detecting fraudulent job postings could also be improved by leveraging metadata and implementing real-time detection capabilities. Additionally, developing an intuitive user interface would facilitate interaction with the detection system, enabling end-users, such as job seekers and platform administrators, to easily access and interpret results, thereby enhancing the system's accessibility and real-world applicability.

References

[1] Rajani et al., 2024. Fake job detection using machine learning. *Materials Science and Technology*, Available at: https://materialsciencetech.com/mst/uploads/2024-42440.pdf.

[2] Tran et al., 2025. Improving fake job description detection using deep learning-based NLP techniques. *Journal of Telecommunication*, Available at: https://www.tandfonline.com/doi/full/10.1080/24751839.2024.2387380?scroll=top&needAccess=true.

[3] Suparna et al., 2024. A Comparative study on fake job post prediction using different datamining techniques. *International Journal of Information Technology and Computer Engineering*, Available at: https://ijitee.org/index.php/ijitee/article/view/678/627.

[4] Rofik et al., 2024. Optimization of SVM and Gradient Boosting Models Using GridSearchCV in Detecting Fake Job Postings. *Matrik: Jurnal Sistem dan Teknologi Informasi*, Available at: https://journal.universitasbumigora.ac.id/index.php/matrix/article/view/3566.

[5] Selvanathan et al., 2025. Fake Job Post Prediction using Machine Learning. *AIP Conference Proceedings*, 3279(1):020036, Available at: https://pubs.aip.org/aip/acp/article/3279/1/020036/3341677/Fake-job-post-prediction-using-machine-learning.

[6] Aravind Sasidharan Pillai (2023). *Detecting Fake Job Postings Using Bidirectional LSTM. arXiv*, Availabionle at: https://arxiv.org/abs/2304.02019.

[7] Hina Afzal et al. (2023). *Identifying fake job posting using selective features and resampling techniques. Multimedia tools and applications*, Available at: https://link.springer.com/article/10.1007/s11042-023-15173-8.