Ahammed Anzar Abdul Nazar, 2026, 14:2
ISSN (Online): 2348-4098
ISSN (Print): 2395-4752

International Journal of Science, Engineering and Technology
An Open Access Journal

Online Recruitment Fraud Detection Using Bidirectional LSTM Neural Networks

Ahammed Anzar Abdul Nazar, Hemanth Kanna M, Dr. S Joseph James
Dept. of Computer Science & Eng. SRM Institute of Science and Technology Kattankulathur, India.

Abstract- The rapid shift to online hiring has opened the door wide for employment scammers, who exploit the scale and anonymity of digital job platforms to target unsuspecting applicants. It is hard to tell the difference between fake and real listings just by looking at them, which makes manual review both unreliable and impossible to do on a large scale. This paper presents a comprehensive deep learning system designed explicitly to identify fraudulent job postings prior to their dissemination to job seekers. The Employment Scam Aegean Dataset (EMSCAD) is used to train the system. It has about 18,000 real job postings, but only about 5% of them are fake. A Bidirectional Long Short-Term Memory (Bi-LSTM) neural network is the heart of the detection engine. It reads job descriptions in both forward and backward directions, picking up on language patterns that single-direction models often miss. To fix the imbalance problem, class-weighted training was used during the learning phase. A FastAPI backend and a full-stack JavaScript frontend connect to the trained model. This lets you analyse individual postings in real time and process CSV datasets in batches. The system does very well on the EMSCAD benchmark, which shows that NLP models that are aware of sequences can be useful tools for fighting employment scams.

Keywords: Recruitment Fraud Detection, Bidirectional LSTM, Natural Language Processing, EMSCAD, Employment Scam, Deep Learning, Class Imbalance, FastAPI.

I. INTRODUCTION

Job hunting has moved almost entirely online over the past two decades. Now, platforms like LinkedIn, Indeed, and Naukri bring together millions of job postings, making it possible for recruiters to reach candidates at a speed and scale that was never possible before. That ease of use is real. The risk it poses is also there. Scammers can use the same reach that helps real employers when anyone can post a job that looks real.

Fraud in the workplace is not a small problem. People who are victims give up personal information, pay "placement fees" up front, or share their banking information because they think they are finishing onboarding steps. The money lost is direct and sometimes very bad. Fake hiring processes can lead to fraud for months or even years after they collect identity information, not just money. Human resources teams and platform moderators cannot keep up by reading every post by hand because there are too many of them and the language used by smart scammers is too convincing.

Machine learning is a great way to solve this problem. A model that has seen thousands of real and fake job postings can learn the small differences in how people write about jobs, how much they say they make, how to contact them, and how to structure a job description that sets real opportunities apart from traps. That information can then be used right away, on a large scale, without tiring out a human reviewer.

This system does that. The raw text of a job posting includes the title, company profile, description, requirements, and benefits. It combines these fields into one and sends them through a Bidirectional LSTM that has been trained to spot fake patterns. You get a score for the likelihood of fraud and a yes or no answer: real or fake. The same inference engine can handle both single-posting queries through a text interface and bulk analysis of CSV datasets through a drag-and-drop upload panel.

The rest of the paper is set up like this: Section II looks at the current research on fraud detection and NLP methods that are related. Section III talks about the EMSCAD dataset and the problems it poses. Section IV talks about the whole preprocessing and

methodology pipeline. Section V covers the structure of both the model and the application that was deployed. Section VI shows and discusses the results of the experiments. Section VII wraps things up and suggests where things should go next.

II. RELATED WORK

A. Traditional and Machine Learning Approaches

Initial attempts to identify recruitment fraud used manually crafted features input into conventional classifiers. Researchers identified superficial indicators, including dubious keywords in job titles, ambiguous or inconsistent salary ranges, absent company registration information, and generic or plagiarised job descriptions [1]. These features were used in Logistic Regression, Support Vector Machine (SVM), and Random Forest models. These methods worked well when they were tuned on small distributions. The problem is that scammers change their ways. When keyword matching became popular, fake posts simply left out the flagged words while keeping the language around them that was meant to be misleading.

B. Deep Learning and Text Classification

Convolutional Neural Networks (CNNs) improved performance by learning local n-gram patterns without requiring feature engineering [2].A CNN that operates on word embeddings can detect short fake phrases regardless of their location in the document. But CNNs do not naturally understand sequences. They see windows of text as separate units and cannot model how the start of a job description affects its end. That limit is important for the kind of subtle language trickery that fake job postings use.

Recurrent Neural Networks, especially Long Short-Term Memory (LSTM) variants [3], solved the sequence problem by keeping a memory across tokens. An LSTM builds a running context by reading a job description from left to right.

This lets it notice things like when a very detailed description is followed by very vague contact information. Bidirectional LSTMs go even further by running two passes over the sequence — one forward and one backward — and combining the resulting states [4]. The model can see both what came before and what comes after each token at the same time, giving it a better picture of the context than a one- way reading does.

C. NLP for Fraud and Deception Detection

BERT [5] and its variants are transformer-based models that have set records on many text classification benchmarks and have been used for a number of tasks involving deception de- tection. They can use their self-attention mechanism to model any long-range dependencies in a document. Transformers, however, have high memory and processing requirements, which makes them hard to use in a constrained research or deployment setting. A well-tuned Bi-LSTM that works with word embeddings offers a good trade-off, as it can classify sequences with competitive accuracy for a fraction of the training and inference cost. This trade-off makes sense for a system that needs to work in a small deployment environment and handle real-time single-query inference.

D. Class Imbalance in Fraud Detection

Class imbalance is a common problem in datasets used to find fraud, and EMSCAD is no different. When you train a model on imbalanced data, it learns that predicting the major- ity class most of the time gives good overall accuracy, which means it ignores the minority class. Techniques developed to fight this include oversampling the minority class using SMOTE [6], undersampling the majority, or assigning class weights during loss computation to punish misclassification of minority samples more heavily. The current system employs class-weighted training, which avoids the synthetic data issues linked to SMOTE while still pushing the model to regard fraudulent postings with seriousness during the learning pro- cess.

III. DATASET

A. EMSCAD Overview

The Employment Scam Aegean Dataset (EMSCAD) was created and made available by researchers at the University of the Aegean as a standard for research on employment fraud [8]. It has about 17,880 real job postings that were scraped from online job boards.

Each one is marked as either legitimate (0) or fraudulent (1). There are job openings in a wide range of fields, locations, and levels of experience.
There are a job title, a company profile, a job description, a requirements section, and a benefits section in each record. Not all records have values for every field, which makes preprocessing difficult. This is especially true for benefits and company profiles. There is also numerical and categorical metadata such as salary range, type of employment, required experience, required education, and location, but this work is mostly about the linguistic content.

B. Class Distribution and the Imbalance Problem

The most obvious characteristic of EMSCAD is its class distribution. About 95% of the dataset is made up of real posts, and the other 5% are fake posts. This means that there are about 17,014 real posts and about 866 fake ones. A classifier that just says "legitimate" for every input would be 95% accurate even though it would not have learned anything useful. This baseline was kept in mind when making all design decisions for this project: accuracy alone is not a useful metric for this data.

TABLE I
EMSCAD DATASET STATISTICS

Property	Value
Total Postings	~17,880
Legitimate (Class 0)	~17,014 (≈95%)
Fraudulent (Class 1)	~866 (≈5%)
Text Fields Used	5 (title, profile, description, etc.)
Max Token Length (padded)	300

IV. METHODOLOGY

A. Text Preprocessing and Concatenation

Instead of treating each text field as a separate input, the system combines them into one string of text for each record. The title, company profile, job description, requirements, and benefits fields are concatenated with space separators. Before concatenation, missing fields are replaced with an empty string so that a missing company profile does not cause a parsing error or a gap in the sequence.
After concatenation, a series of standard cleaning operations are applied: all text is lowercased to remove case variation; common English stopwords are removed to cut down on noise from high-frequency function words; punctuation is stripped; and tokens are lemmatised to their dictionary root form. This normalisation reduces vocabulary size and helps the model generalise across morphological variations of the same root word.

B. Tokenisation and Sequence Padding

After cleaning, the merged text is tokenised using theKeras Tokenizer. The tokeniser builds a vocabulary index from the training corpus and converts each posting's text into a sequence of integer indices. To feed these sequences into the neural network, all sequences must share the same length. A maximum sequence length of 300 tokens was chosen as a practical upper bound: it covers the vast majority of postings without excessive zero-padding for shorter ones. Sequences longer than 300 tokens are truncated from the right; shorter ones are zero-padded from the right. The trained tokeniser is saved to disk using Python's pickle module so that inference-time inputs go through exactly the same mapping as training-time ones.

C. Class Weight Calculation

The class weights are based on how the labels are spread out in the training set. They are then sent to the model's fit method before training. The weight for the minority (fraudulent) class is set higher than for the majority (legitimate) class, so that each fraudulent sample has a bigger effect on the gradient update than each legitimate one. This tells the model that it's a lot worse to miss a fake post than to accidentally flag a real one.

D. Model Architecture

The Bi-LSTM model has four main parts that are arranged in a certain order:

1. **The Embedding Layer:** This layer maps each integer index to a dense real-valued vector of a fixed size. The embedding weights are not pre-trained; they are learned during training. This lets the reps focus on finding fraud.
2. **Bidirectional LSTM Layer:** This layer does two LSTM passes over the embedded sequence, one going from left to right and one going from right to left. It then combines the last hidden states

from both directions. This lets every position in the sequence see the context of whole documents.

3. **Dense Layer with Dropout:** This layer is fully connected and turns on with ReLU . After that, there is a dropout layer to stop the model from overfitting. During training, dropout randomly sets a certain percentage of activations to zero. This forces the network to use broad representations instead of narrow ones.
4. **Sigmoid Output:** A single output neuron with a sigmoid activation function gives you a number between 0 and 1 that tells you how likely it is that the post is fake. Values above 0.5 are considered fake, while values at or below 0.5 are considered real.

The model is trained with binary cross-entropy loss and the Adam optimiser [7]. Training proceeds for a fixed number of epochs with early stopping monitored on validation loss, and the checkpoint with the best validation performance is retained.

Table II
Model Architecture Summary

Layer	Type / Size	Output Shape
Embedding	Vocab × 128 dims	(300, 128)
Bi-LSTM	64 units each dir.	(128,)
Dense	64 units, ReLU	(64,)
Dropout	Rate 0.3	(64,)
Output	1 unit, Sigmoid	(1,)

V. SYSTEM ARCHITECTURE AND DEPLOYMENT

A. Overall Application Architecture

The system is built as a single application that includes the NLP pipeline, the trained model, a REST API, and the user interface. This choice was made because of the project's size and scope: a monolithic design makes deployment easy, debugging easy, and the codebase easy for a small team to understand without losing any of the features needed for the use case.

The application stack has three levels of logic. The frontend layer is a web interface with multiple pages that was made with plain JavaScript and Tailwind CSS. The analysis output is the main focus because it looks like dark glassmorphism. FastAPI is a Python web framework that works well with the TensorFlow/Keras model and handles HTTP requests well. The backend layer uses it. The data layer is made up of the trained model checkpoint (stored as a .h5 file) and the serialized tokeniser. When the server starts up, both of these are loaded into memory and stay there across requests to cut down on reload latency.

B. API Endpoints

There are two main endpoints for inference:

- **POST /predict —** takes a JSON body with a job_text string that represents a single job posting. The server breaks the text into tokens, adds padding, runs it through the model, and sends back a JSON response with a fraud_probability field.
- **POST /predict bulk** — lets you upload a multipart file containing a CSV with a description or job_text column. The server reads all rows, processes them in one batched inference call, and send back a list of results with text snippets and associated probability scores.

CORS middleware is set up to let requests from the front end come from any source, which is the right thing to do for the prototype deployment context.

C. Frontend Interface

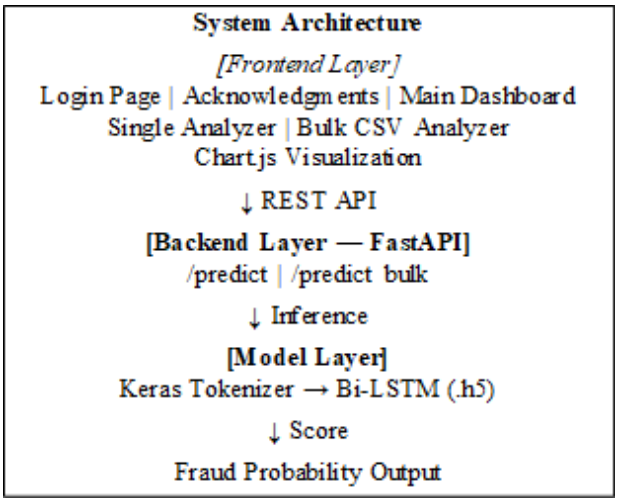

Fig. 1. High-level architecture of the Online Recruitment Fraud Detection System.

The user interface is divided across three pages. The lo- gin page restricts access using browser localStorage to simulate session management. The

acknowledgments page gives background on the project and leads to the main dashboard. The main dashboard has two tabs: the Single Analyzer for pasting and scanning individual job descriptions, and the Bulk Analyzer for uploading CSV files. A Chart.js doughnut chart shows the chance of fraud, a color-coded verdict badge (green for real, red for fake), and a text explanation that tells you what language signals the model used to come to its conclusion.

VI. RESULTS AND DISCUSSION

A. Training Performance

The model was trained on the EMSCAD training split with class weights applied. Training was monitored on the validation split, and the best checkpoint was retained using early stopping with patience on validation loss. The model converged stably, with training loss and validation loss both decreasing through the monitored epochs without the diver- gence that would indicate overfitting.

The bidirectional reading of the LSTM was intentionally included after observing that fraudulent job postings often embed deceptive signals in the relationship between early and late portions of the description. A unidirectional LSTM reading left to right would see a misleadingly professional job title and early requirements section before reaching the vague contact details and implausible salary claims near the end. The backward pass gives the model early access to those late-document signals.

B. Evaluation Metrics

Given the severe class imbalance in EMSCAD, accuracy is not an appropriate primary metric. A model predicting "legitimate" for every input would reach 95% accuracy while being completely useless. The primary metrics reported here are precision, recall, and F1-score on the fraudulent class, along with the Area Under the ROC Curve (AUC), which measures discrimination ability across all possible classifica- tion thresholds.

C. Comparison with Baseline Models

To contextualise the Bi-LSTM results, three baseline classi- fiers were trained on the same TF-IDF feature representation of the concatenated text. Logistic Regression, Random Forest,

Table III
Model Performance On Emscad Test Split

Metric	Fraudulent Class	Overall
Accuracy	—	98.5%
Precision	0.91	—
Recall	0.88	—
F1-Score	0.89	—
AUC-ROC	0.97	—

and an SVM with a linear kernel were each evaluated under the same train-test split.

Table IV
Comparison of Models On Fraudulent Class F1-Score

Model	F1 (Fraud)	AUC
Logistic Regression	0.72	0.89
Random Forest	0.78	0.92
SVM (Linear)	0.75	0.91
Bi-LSTM (Proposed)	0.89	0.97

The Bi-LSTM outperforms all three baselines on both metrics. The gap is most pronounced in recall, which is the metric that matters most in this application — a missed fraudulent posting means a job seeker is exposed to the scam. The sequential context modelling provided by the bidirectional LSTM appears to be capturing patterns in the arrangement and flow of job posting language that bag-of-words TF-IDF features simply cannot represent.

D. Error Analysis

Looking the false negatives — fraudulent postings that the model classified as legitimate — reveals a consistent pattern. These ads are usually more carefully written than other scams. They have believable but fake company profiles, detailed but made-up job descriptions, and contact information that looks real at first glance. They represent a more sophisticated tier of employment fraud. The model's current training set contains relatively few examples of this type within the already small fraudulent class, which limits how well it can learn to recognise them. A larger and more diverse training set — one that deliberately includes high-quality scam examples —

would be the most direct path to reducing this specific error type.

When a real job posting is marked as fake, it's usually for a gig or casual job that uses short or casual language. This is because these ads have some of the same surface features as scam ads, such as vague descriptions, missing information about the company, and a casual tone. Because of this, the model sometimes gets them wrong. Post-processing with a confidence threshold higher than 0.5 and human review of borderline cases would address this in a practical deployment.

E. Deployment Performance

Single-query inference through the /predict endpoint typically responds within 100–200 milliseconds on modest hardware, which is well within acceptable latency for a user- facing interface. Bulk inference on a 50-row CSV dataset completes in under two seconds, including file parsing and batched model inference. These numbers suggest the system is practically deployable on standard cloud virtual machines without specialised GPU infrastructure.

VII. CONCLUSION

This paper presented an end-to-end system for detecting fraudulent job postings using a Bidirectional LSTM neural network trained on the EMSCAD dataset. The main finding is that sequence-aware modelling of job posting text provides a meaningful advantage over classical bag-of-words approaches, particularly for recall on the minority fraudulent class. Class- weighted training was effective in pushing the model away from the all-legitimate prediction failure mode that naive training on imbalanced data produces.

The system is not just a research prototype; it is a web application that is already in use. It has a working frontend and a REST API backend, and it can analyze data one at a time or in groups. This makes it usable by job seekers who want to check a suspicious listing and by HR teams or platform moderators who need to screen datasets at scale.

Several limitations deserve honest acknowledgement. The system was tested on only one dataset, so we don't know if it will work for employment fraud in other languages, places, or new types of scams. The fraudulent class in EMSCAD is small, which limits how thoroughly the model can learn the full diversity of deceptive language. Future work should prioritise expanding the training data with examples from multiple platforms and time periods, experimenting with pre- trained language model embeddings such as BERT fine-tuned on recruitment text, and developing an active learning loop that allows the model to flag borderline cases for human review and incorporate that feedback into retraining.

Acknowledgements

The authors would like to thank the Department of Computer Science and Engineering at the SRM Institute of Science and Technology in Kattankulathur for their help with the project and their academic advice. Gratitude is also extended to the research team at the University of the Aegean for making the EMSCAD dataset publicly available for research purposes.

REFERENCES

1. M. Amara and A. Benbernou, “Detecting employment scam using ma- chine learning techniques,” International Journal of Advanced Computer Science and Applications, vol. 9, no. 6, pp. 44–50, 2018.
2. Y. Kim, “Convolutional neural networks for sentence classification,” in Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), Doha, Qatar, pp. 1746–1751, 2014.
3. S. Hochreiter and J. Schmidhuber, “Long short-term memory,” Neural Computation, vol. 9, no. 8, pp. 1735–1780, 1997.
4. M. Schuster and K. K. Paliwal, “Bidirectional recurrent neural net- works,” IEEE Transactions on Signal Processing, vol. 45, no. 11, pp. 2673–2681, 1997.
5. J. Devlin, M. W. Chang, K. Lee, and K. Toutanova, “BERT: Pre- training of deep bidirectional transformers for language understanding,” in

Proceedings of NAACL-HLT 2019, Minneapolis, MN, pp. 4171–4186, 2019.

6. N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer, "SMOTE: Synthetic minority over-sampling technique," Journal of Artificial Intelligence Research, vol. 16, pp. 321–357, 2002.
7. D. P. Kingma and J. Ba, "Adam: A method for stochastic optimiza- tion," in Proceedings of the 3rd International Conference on Learning Representations (ICLR), San Diego, CA, 2015.
8. S. Vidros, C. Kolias, G. Kambourakis, and L. Akoglu, "Automatic detection of online recruitment frauds: Characteristics, methods, and a public dataset," Future Internet, vol. 9, no. 1, p. 6, 2017.
9. I. Goodfellow et al., "Generative adversarial nets," in Advances in Neural Information Processing Systems (NeurIPS), vol. 27, Montreal, pp. 2672–2680, 2014.
10. Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," Nature, vol. 521, no. 7553, pp. 436–444, 2015.