Multi-Modal Candidate Shortlisting System

Pathri Vidya Praveen

June 24, 2025

Abstract

This project presents a multi-modal deep learning system for predicting whether a job candidate should be shortlisted based on their resume text and structured profile data. The system leverages a dual-branch architecture combining a transformer-based text encoder (BERT) with a multilayer perceptron (MLP) for tabular features. The aim is to simulate a real-world HR analytics pipeline for intelligent candidate screening using Natural Language Processing (NLP) and tabular data modeling techniques.

1 Introduction

Recruitment processes are often time-consuming and subjective. To streamline initial candidate filtering, we propose a deep learning model that assesses resumes through both natural language and structured data. The motivation stems from real-world applications in Human Resource Management Systems (HRMS), where AI can significantly reduce manual efforts while maintaining fairness and consistency.

2 Dataset and Preprocessing

2.1 Data Composition

The dataset comprises:

- **Resume Text**: Free-form descriptions of education, skills, experience.
- **Profile Features**: Degree, GPA, years of experience, job role, etc.
- **Label**: A binary indicator of shortlisting (1) or rejection (0).

2.2 Label Generation with Noise

Labels were initially generated through keyword matching based on role-specific skills. To introduce realism and prevent overfitting, we flipped labels with a global noise probability of 0.1.

2.3 Text Cleaning and Tokenization

Resume texts were lowercased and cleaned using regular expressions. Tokenization was performed using Keras's Tokenizer. For BERT, the Huggingface tokenizer was used to ensure compatibility with the transformer model.

2.4 Embedding Layer

GloVe 100D embeddings were loaded and converted into an embedding matrix, later used to initialize an embedding layer (though BERT supersedes this in the final model).

2.5 Tabular Features

Categorical variables such as job role were one-hot encoded. GPA and Experience were synthetically generated and normalized using StandardScaler to simulate a real-world profile table.

3 Model Architecture

3.1 Overview

The model consists of two branches:

- **Text Branch (BERT)**: A pre-trained transformer extracts semantic features from resume text. The `pooler_output` is passed through dropout.
- **Tabular Branch (MLP)**: Experience and GPA are passed through a multilayer perceptron with ReLU, batch normalization, and dropout.

3.2 Fusion and Classification

Features from both branches are concatenated and passed through a final classifier (2 linear layers with dropout and ReLU) to produce logits for binary classification.

4 Training Strategy

4.1 Loss Function

Binary Cross Entropy with Logits Loss (BCEWithLogitsLoss) is used. Class imbalance is handled using `pos_weight` computed from training label distribution.

4.2 Optimization

AdamW optimizer with a learning rate of 2×10^{-5} is used. A linear warmup scheduler is implemented.

4.3 Regularization and Early Stopping

Dropout is applied in both branches and the final classifier. Early stopping is triggered if validation loss does not improve over 2 epochs.

5 Evaluation and Metrics

5.1 Metrics Used

- **Precision**: 0.8961
- **Recall**: 0.9324
- **F1 Score**: 0.9139
- **ROC-AUC Score**: 0.7483

5.2 Synthetic Label Generation

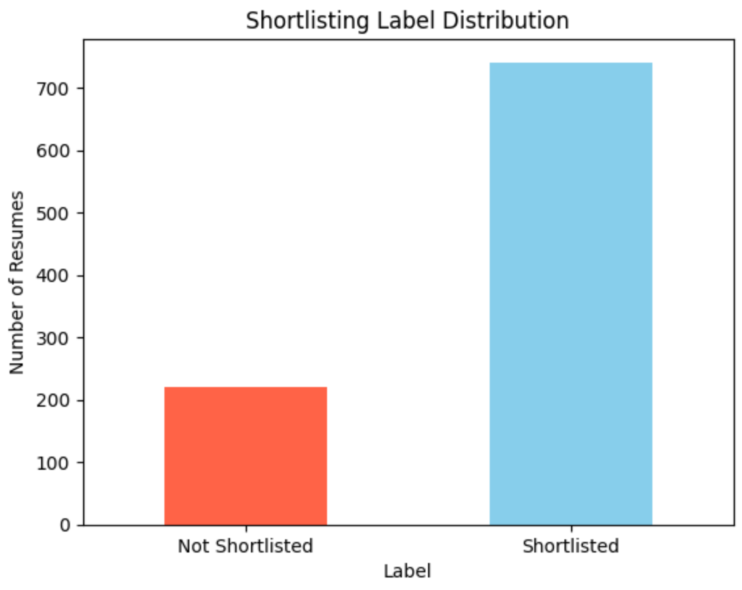

Figure 1: Plot showing labels that are synthetically generated

5.3 t-SNE visualization of GloVe Embeddings

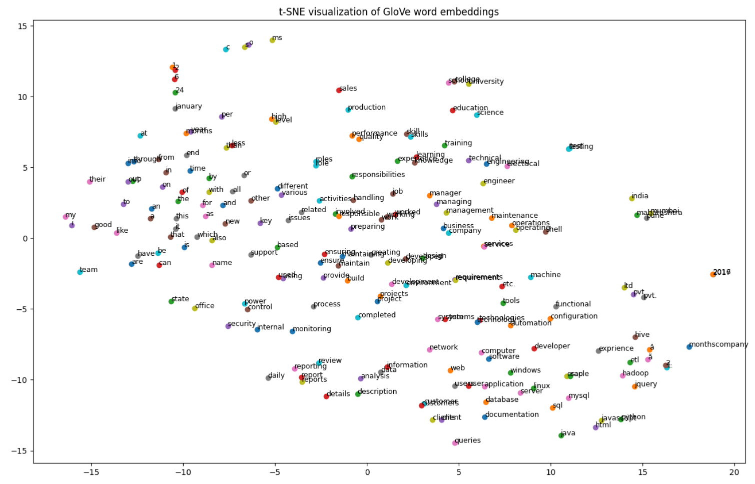

Figure 2: t-SNE visualization of GloVe embeddings(100d)

5.4 Confusion Matrix

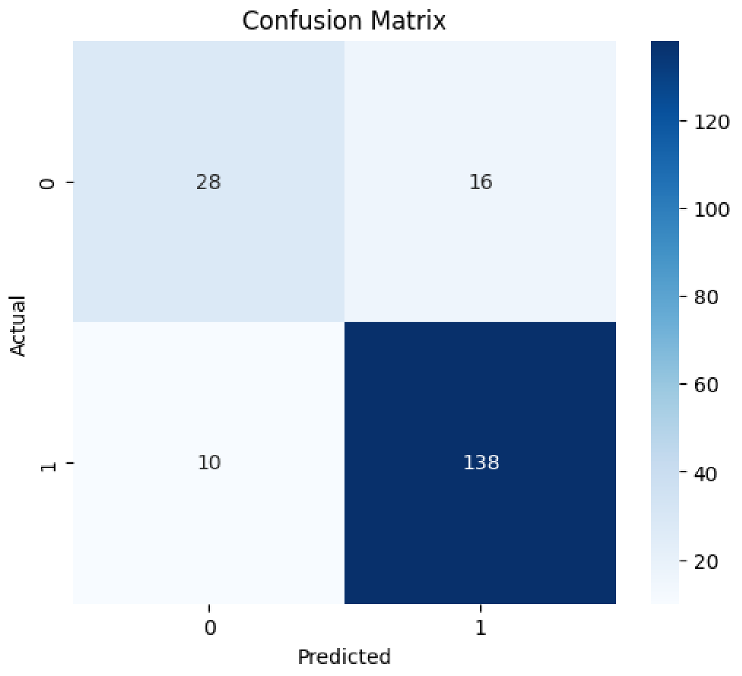

Figure 3: Confusion Matrix of Validation Set

5.5 ROC Curve

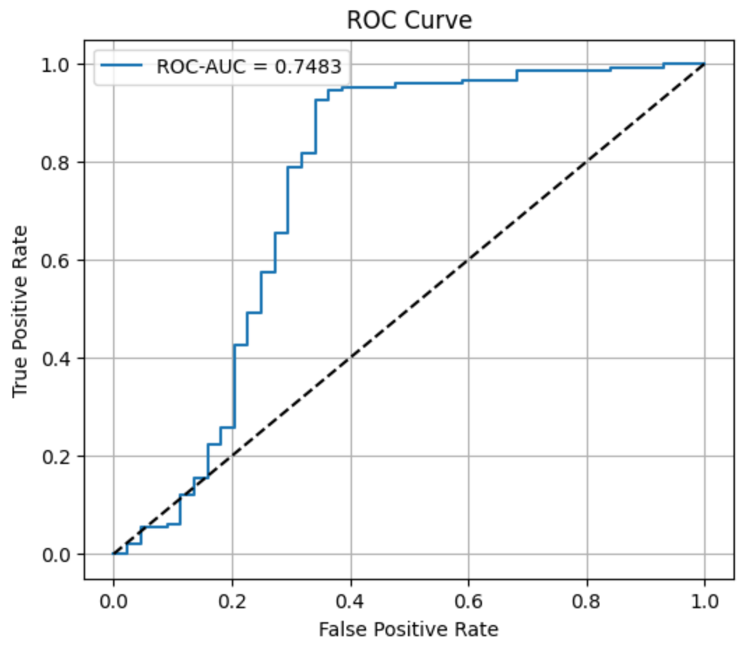

Figure 4: ROC Curve showing classifier performance

6 Conclusion and Future Work

This project demonstrates a successful integration of NLP and tabular modeling to automate candidate screening. The model achieves strong recall and F1 scores, indicating reliable shortlisting.

Possible Improvements

- Use contextual embeddings from sentence-transformers or domain-specific BERT models (e.g., SciBERT for scientific resumes).
- Replace MLP with TabNet or FT-Transformer for richer tabular understanding.
- Hyperparameter tuning using Optuna or Bayesian optimization.
- Build a basic Streamlit or FastAPI-based interface to simulate deployment.
- Include resume PDF parsing and named entity recognition (NER) for better skill extraction.
- Use attention mechanisms to highlight important sections of resumes
- Add pretraining of tabular features using an autoencoder

- Compare with a baseline model (e.g., logistic regression or XGBoost on tabular features alone).
- Implement interpretability methods such as SHAP or LIME to explain model predictions.