Resume Reveal

Reveal your seniority level

By:
Matan Cohen
Shira Shani
Eden Menahem

Problem Description

Motivation
Accurately assessing candidate seniority from resumes is essential for fair and efficient hiring. However, resumes often contain vague or exaggerated descriptions that obscure true expertise

Application Value
Automated seniority classification supports faster, more consistent screening and better Candidate-role**.** matching in large-scale tech recruitment

Challenges

- Non-standard resume format.
- Ambiguous or inflated language Implicit seniority cues that require contextual interpretation.

Scope and Objectiv
Test LLMs' ability to classify seniority from unstructured resume text**.**
Generate synthetic resumes to control and vary seniority signals.
Combine real and generated data, including misleading examples.
Compare model performance.

Formal Task Specification

Task Specification:

- Input: Resume text + job role.
- Output: Predicted seniority level -Junior / Mid / Senior.
- Task: Multi-class classification (single-label prediction)

Multi-class single-label classification

Evaluation Metrics:

loss monitoring , Accuracy, Confusion Matrix.

High-Level Plan:

Data:

- Synthetic: LLM-generated by title & seniority.
- Real: Scraped from hireitpeople.com with labels.

Training: DistilBERT & RoBERTa fine-tuning.

Evaluation: Compare DistilBERT, RoBERTa, GPT-4

Prior Art

TITLE	[ResumeAtlas: Revisiting Resume Classification with Large-Scale Datasets and Large Language Models]	[Construction of English Resume Corpus and Test with Pre-trained Language Models]	[conSultantBERT: Fine-tuned Siamese Sentence-BERT for Matching Jobs and Job Seekers]
Task solved	Classification task: mapping full resume text to a seniority label	Classification task: classifying all detected resume text blocks into five semantic sections	Scoring task: the model gives a score showing how well a resume fits a job role.
Approach / Model	Used BERT and compared it to TF-IDF for classifying resumes into Junior, Mid, and Senior levels.	Used resume text, splits it into parts like experience or education, and uses BERT or DistilBERT to predict the correct label for each part. Results were compared to TF-IDF.	Fine-tuned a BERT model to match resumes with job descriptions, and compared it to TF-IDF using similarity scores.
Data	13,389 resumes labeled by seniority and job role.	1,484 resumes: 286 labeled, 1,198 from OCR.	270,000 resume-job pairs from real applications.
Metrics	Checked how often the top 1 or top 5 predictions matched the correct seniority.	Measured how well each part of the resume (like experience) was labeled correctly.	Compared how well resumes matched jobs using accuracy and similarity scores.
Results	BERT got better accuracy than TF-IDF (92% vs. 85.8%).	DistilBERT worked best. Experience helped the model.	SBERT gave better job-resume matches than TF-IDF.

Data Description

Dataset Summary

- 584 resumes with job role and seniority level
- Sources: Scraped + GPT API generated
- Fields:
  - Resume: Full text
  - Job Title: Declared role
  - Seniority: Labeled (Junior/Mid/Senior)

Example:

Input:
Resume -> "Senior Business Analyst with over 8 years in financial services, led cross-functional teams at XBank…."
Job Title -> "Senior Business Analyst"
Features -> WordCount = 230, TitleTokens = 3
Output *-> Senior*

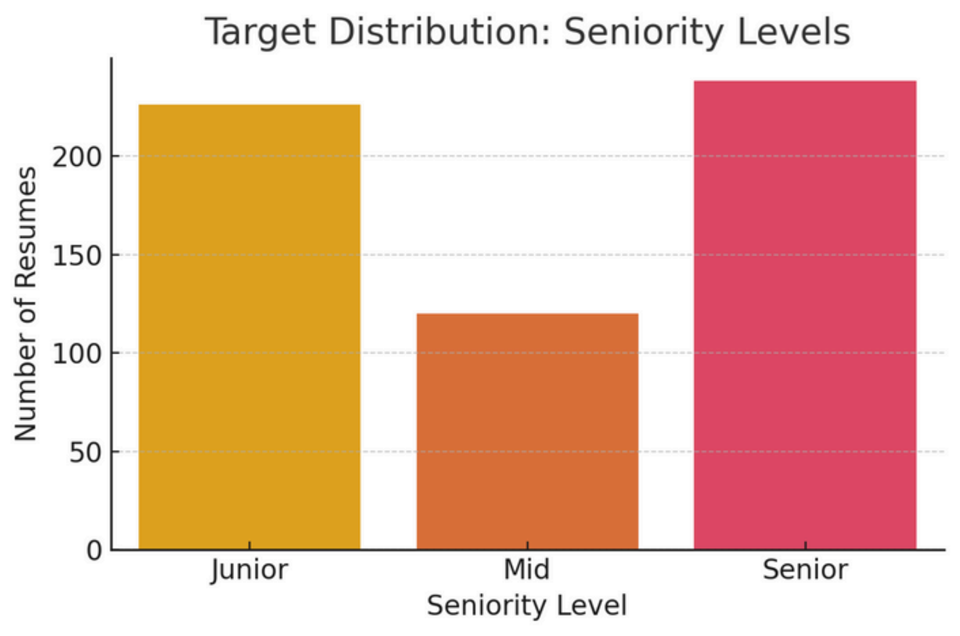
EDA

Dataset Overview

- Labels: Senior (41%), Junior (39%), Mid (21%) – slight class imbalance

Resume Length

- Mean: 462.7 words
- By Class:
  - Junior: 226 → 411.8 words, 2.28 job role tokens
  - Mid: 120 → 264.4 words, 2.28 job role tokens
  - Senior: 238 → 610.9 words, 3.28 job role tokens



Data Generation & Labeling Process

- Data cleaning: Standardized to Title Case and trimmed whitespace
- Validation: No missing labels
- Feature Augmentation:
  - WordCount – calculated by splitting text on whitespace
  - Interquartile Range: 137–602 words

Models and processing pipelines

Models/Pipelines used:
Data Collection & Generation:
– Web scraping (BeautifulSoup/Selenium)
– Synthetic data via GPT for augmentation
Models:

- **distilBERT:**
  – Tokenizer: DistilBertTokenizerFast (max_length=512, padding="max_length")
  – Model: DistilBertForSequenceClassification
- **RoBERTa:**
  – Tokenizer: RobertaTokenizerFast (max_length=512, padding="max_length")
  – Model: RobertaForSequenceClassification
- **GPT-4:**
  – Tokenizer: Not required – handled internally by GPT-4 API
  – Model: openai/gpt-4.1 via OpenRouter (generative classification)

Training Details:
Data split: 80% train / 20% validation -

- **distilBERT:** epochs=10, lr=2e-5, batch_size=8
- **RoBERTa:** epochs=10, lr=2e-5, batch_size=8
- **GPT-4:** with prompt and predicted result.

Platform:

- Google Colab Pro (Tesla T4 GPU)

Metrics

Metrics used at each step:

- Accuracy
- loss monitoring
- Confusion Matrix

How metrics are computed:

- During training:
  – Compute metrics on validation set after each epoch
  – Use model.eval() + no_grad() to predict labels
  – Aggregate true vs. predicted labels (batch-wise)
- During final evaluation:
  – Report overall loss monitoring , Accuracy, Confusion Matrix.
- Details:
  – Metrics calculated with scikit-learn’s accuracy_score

Code Organization

GitHub Repository : [ResumeRevealNLP]

Data Files :

- DATA.xlsx (located in SRC folder)
  - Resume: Full text of the candidate's resume.
  - Job Role: Declared job role.
  - Seniority: Manually assigned label indicating level (junior, mid, senior).

Major tasks and code files :

- DATA_GENERATION_AND_SCRAPING.ipynb → Data Generation and scraping code.
- EDA+BASELINE.ipynb → EDA and Baseline code.
- MODELS_FINAL.ipynb → Training & Evaluation code.

Results and Graph files:

evaluation.csv –

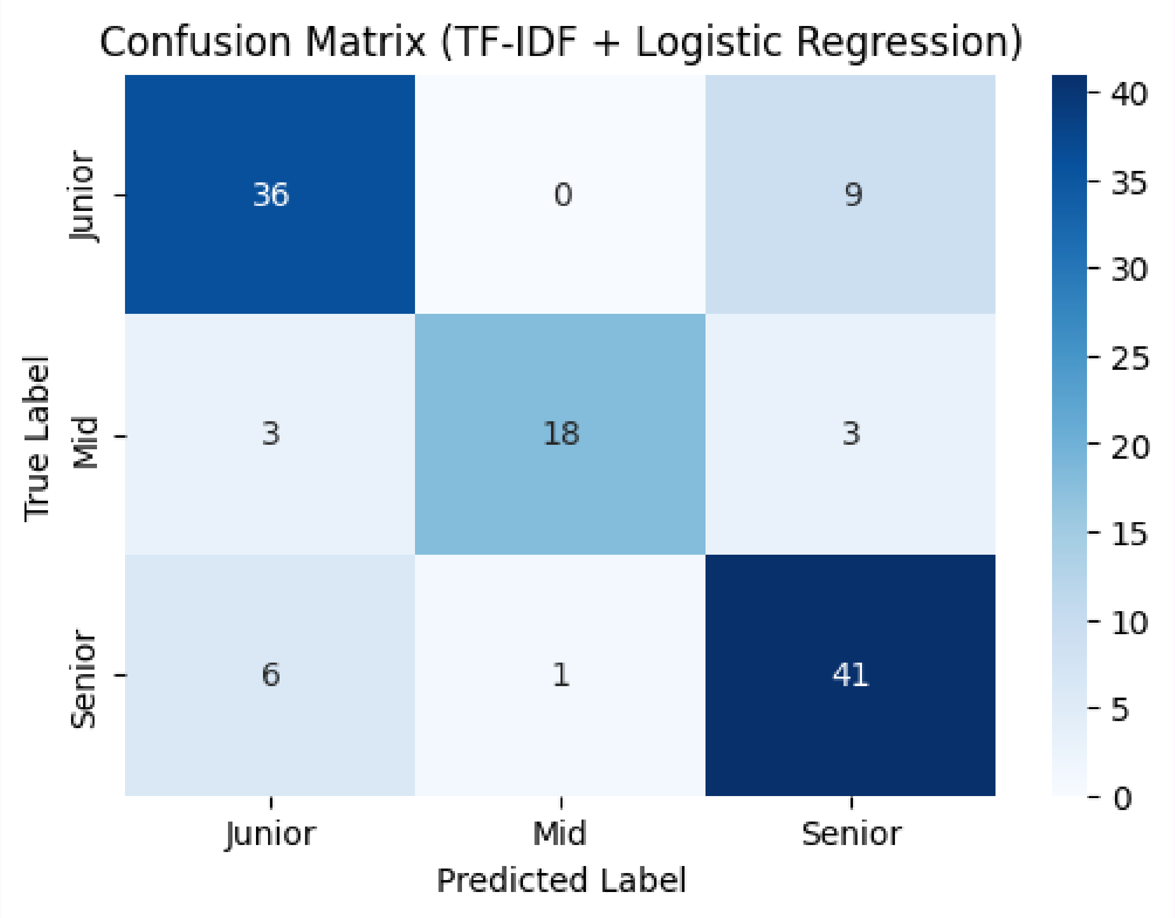
- columns - evaluation metrics.
- rows - models.

Evaluation_file.pdf –

PNGs showing validation curves and confusion matrices for DistilBERT & RoBERTa, plus final accuracy comparison.

Baseline

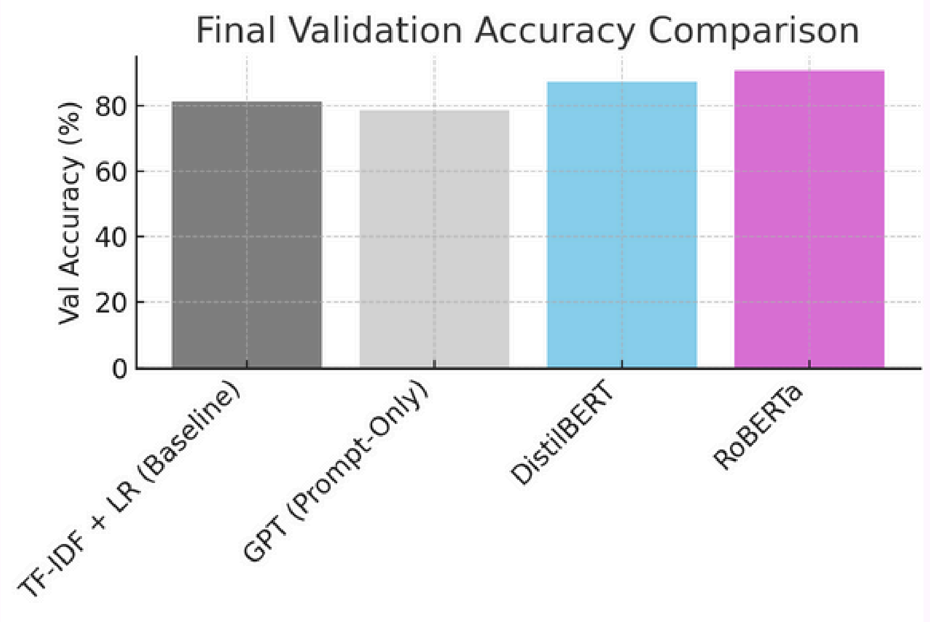
- Baseline model: TF-IDF (1–2 grams, 5,000 features) + Logistic Regression
- Input: Combined job title and resume text
- Data split: 80/20 stratified by seniority
- Accuracy: 81.2%

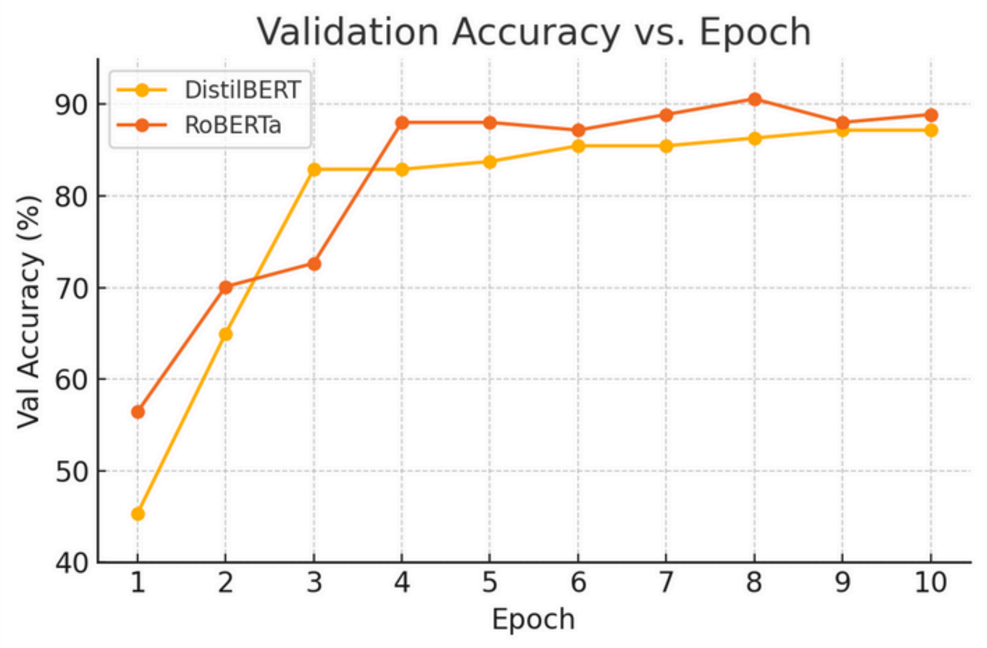


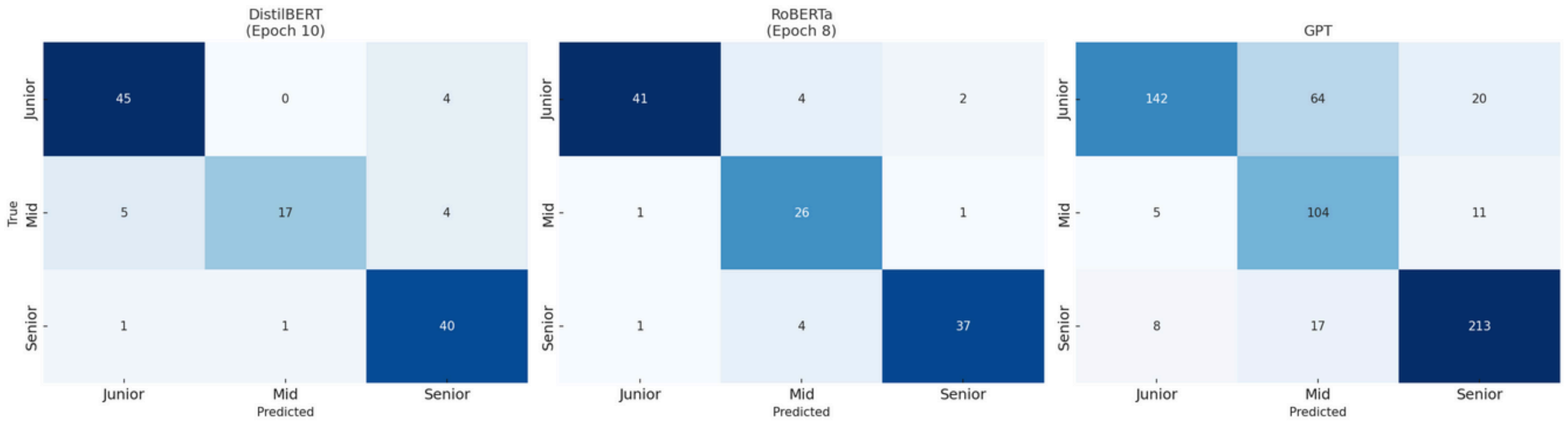
Results

Our model is **RoBERTa**, which we compare to the TF-IDF baseline and two other models - DistilBERT and GPT-4.1.

Model	Best Epoch	Val Loss	Val Acc (%)
TF-IDF + LR (Baseline)	-	-	81.20
GPT (Prompt-Only)	-	-	78.60
DistilBERT	10	0.4026	87.18
RoBERTa	8	0.3297	90.60







Main Results and conclusion

Effect of Configuration:
Extending to 10 epochs with a linear learning-rate scheduler steadily improved validation accuracy for both DistilBERT (from 45.3 % at epoch 1 to 87.18 % at epoch 10) and RoBERTa (from 56.4 % at epoch 1 to 90.6 % at epoch 8).

Objectives Achieved:
The primary goal—exceeding baseline classification accuracy— with RoBERTa achieving the highest score.

Data Support:
Validation losses decreased and accuracies rose as epochs increased under the scheduler, confirming that our configuration choices directly led to surpassing baseline performance.

Visual Abstract Slide