PROJECT ABSTRACT

Resume Screening using NLP

An NLP pipeline that ranks resumes against a job description with TF-IDF cosine similarity and lexicon-based skill matching, delivered as a built-to-order student project with code, demo app, report, PPT and viva kit.

01. Title

Resume Screening using NLP

02. Introduction / Background

A single popular job posting can attract hundreds of resumes, and manual screening does not scale: recruiters skim each resume for seconds, strong candidates get missed, and two screeners can rank the same pile differently. Keyword-only applicant tracking systems help but are brittle to phrasing variation. Classical NLP offers a transparent middle ground: TF-IDF weighting captures which terms actually discriminate within an applicant pool, cosine similarity measures textual fit to the job description, and lexicon-based skill extraction checks required competencies explicitly. This project turns those techniques into a complete, demoable student build with fully explainable scores.

03. Problem Statement

Manual resume screening is slow, inconsistent and unscalable, while keyword-only filters miss qualified candidates over phrasing differences. The project addresses this by building an NLP screener that ranks applicants against a job description with a transparent composite score — 65% skill overlap plus 35% TF-IDF cosine similarity — and explains every ranking with matched skills, missing skills and similarity values, keeping the final decision with the human screener.

04. Objectives

- Build a preprocessing pipeline (tokenization, stopword removal, Porter stemming) and a 44-term domain skill lexicon for extraction from job descriptions and resumes.
- Implement TF-IDF vectorization with scikit-learn and a composite ranking score combining skill overlap (65%) and cosine similarity (35%).
- Build a Flask demo app with a screening analyzer, a resume inspector with keyword highlighting, and a matching-engine view (TF-IDF weights, similarity heatmap, skill gaps).

- Deliver the complete student kit: source code, evaluation notebook, report, PPT and viva Q&A document.

05. Proposed Methodology

Resumes and the job description are normalized through lowercasing, tokenization, stopword removal and Porter stemming. Required skills are extracted from the JD with regex matching against the 44-term lexicon. TF-IDF vectors are built over the JD plus all resumes so term weights reflect the applicant pool; cosine similarity between each resume vector and the JD vector gives the text-similarity component. Skill overlap (matched required skills divided by total required skills) gives the skill component. The composite score ranks candidates, and configurable thresholds assign Shortlist / Review / Reject verdicts.

06. System Architecture / Working

The system has three stages. (1) Ingestion: the recruiter pastes a job description and loads resumes as text or text-layer PDFs. (2) NLP engine: preprocessing, skill extraction, TF-IDF vectorization, cosine similarity and composite scoring run as a pure-Python pipeline. (3) Demo application: a Flask app presents the ranked shortlist, the resume inspector with matched-keyword highlighting, and the matching-engine analytics (top TF-IDF terms, resume-to-resume similarity heatmap, per-candidate skill-gap tables). Batch mode scores a folder of resumes and exports the ranked table.

07. Hardware / Software Requirements

- Any modern laptop/desktop (CPU only; no GPU needed)
- Python 3.10, scikit-learn (TfidfVectorizer, cosine_similarity), pandas, NumPy
- Regex-based skill lexicon; NLTK-style preprocessing implemented in the codebase
- Flask for the demo web app; Matplotlib for TF-IDF and heatmap charts
- Jupyter notebook for the buyer-run pipeline walkthrough and precision-at-k evaluation
- Resumes supplied by the buyer as .txt or text-layer .pdf files

08. Expected Outcomes

A working screener that ranks any resume set against any pasted job description with explainable scores; a demo that makes the ranking, the skill gaps and the similarity structure visible in seconds; and a notebook procedure that computes precision-at-k on the buyer's own labeled resume set. No ranking accuracy is pre-claimed — the report presents the build's own measured precision-at-k with the labeling procedure documented.

09. Applications

- Recruiter screening assistance for high-volume job postings (human decides, tool ranks)
- Teaching material for NLP coursework: preprocessing, TF-IDF, cosine similarity, information retrieval
- Reference implementation for skill-lexicon design and explainable scoring
- Base for embedding-based semantic matching as future scope

10. Future Scope

- Sentence-embedding (e.g. SBERT) semantic similarity to handle paraphrase beyond lexical overlap
- Tesseract OCR integration for scanned/image PDF resumes
- Expanded and auto-learned skill lexicons per industry vertical
- Fairness auditing dashboards for screening outcomes across demographic proxies

11. References

- Salton, G. and Buckley, C. — Term-weighting approaches in automatic text retrieval, Information Processing & Management, 1988 (TF-IDF).
- Pedregosa, F. et al. — Scikit-learn: Machine Learning in Python, JMLR 2011.
- Bird, S. et al. — Natural Language Processing with Python (NLTK Book), O'Reilly.
- Projectech — Resume Screening using NLP, project page and demo (projectech.in).