ResumeScoreX

Transforming the Resume Screening

Using NLP concepts

next slide →

02

Introduction

In today's fast-paced recruitment landscape, sifting through hundreds or even thousands of resumes can be time-consuming and challenging for hiring teams. To streamline this process, we've developed the NLP-Based Resume Screening Tool — a solution that leverages Natural Language Processing (NLP) and machine learning techniques to automatically assess the relevance of resumes to job descriptions

next slide →

03

Working of ResumeScoreX

This tool extracts the content from resumes, compares it with the specific skills and qualifications mentioned in a job description, and generates a match score to rank candidates based on how well their resumes align with the position. The tool utilizes TF-IDF vectorization and cosine similarity to measure textual similarity, offering an efficient and objective way to shortlist the most suitable candidates.

By automating this process, this tool not only saves valuable time for recruiters but also improves the accuracy and consistency of resume screening, ensuring a fairer and more effective hiring process.

04

Concepts Used in ResumeScoreX

HTML FOR FRONTEND

Users can interact with the tool via a simple web form that lets them upload PDF resumes HTML forms send the uploaded resume file to the backend (Flask) for processing.

FLASK FOR BACKEND

Flask is a lightweight web framework written in Python. It is used to build the backend of web applications and handle HTTP requests.

RESUME MATCHING LOGIC

The resume text is extracted using pdfplumber, which reads the PDF file and converts it into a machine-readable format. The resume and job description texts are converted into numerical vectors using the TF-IDF.Once converted into vectors, cosine similarity is used to compare the similarity between the resume and the job description, resulting in a match score

next slide →

05

Key Python Libraries

PDFPLUMBER

Extracts text from PDF resumes. Reads each page of the uploaded PDF and compiles the text for processing.

NLTK (NATURAL LANGUAGE TOOLKIT)

Provides tools for text preprocessing and tokenization.Helps clean and structure the resume and job description texts for analysis.

SCIKIT-LEARN

Contains the TF-IDF Vectorizer to convert text into numerical form. Includes cosine_similarity to compute the similarity between the resume and job description.

FLASK-CORS

Enables Cross-Origin Resource Sharing, allowing the frontend (if hosted elsewhere) to communicate with the backend API securely.

next slide →

06

Text Extraction Function

extract_text_from_pdf(file_path)

This Function is used to convert uploaded PDF resumes into raw, readable text that can be processed using NLP techniques. Resumes are usually in PDF format, which are not directly readable as plain text. NLP models require clean, structured text input.

The uploaded PDF file is passed to pdfplumber.
All text from each page is extracted and combined into a single string.The final output is passed to the NLP engine for scoring.

Matching Score Logic

Objective :

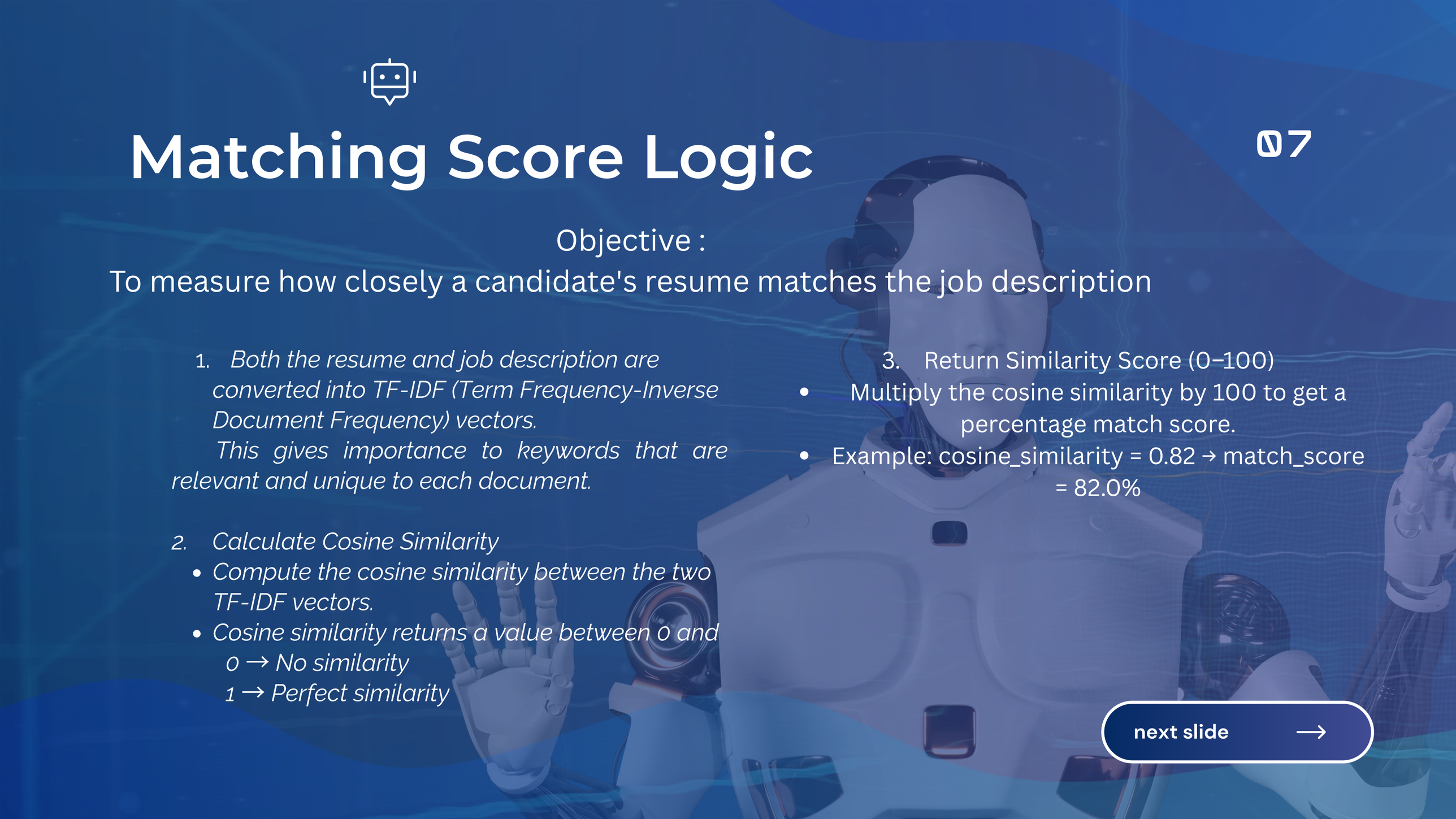
To measure how closely a candidate's resume matches the job description

1. *Both the resume and job description are converted into TF-IDF (Term Frequency-Inverse Document Frequency) vectors.*

This gives importance to keywords that are relevant and unique to each document.

2. *Calculate Cosine Similarity*
   - *Compute the cosine similarity between the two TF-IDF vectors.*
   - *Cosine similarity returns a value between 0 and*
     0 → No similarity
     1 → Perfect similarity

3. Return Similarity Score (0–100)
   - Multiply the cosine similarity by 100 to get a percentage match score.
   - Example: cosine_similarity = 0.82 → match_score = 82.0%



next slide →

08

Why We Chose Flask for ResumeScoreX

- *Easy Integration with Python & NLP. Works seamlessly with Python libraries like pdfplumber, nltk, and scikit-learn.*
- *Using Flask helps bridge the user interface and the backend NLP logic, allowing users to upload resumes and receive smart, real-time match scores via the web.*
- *Enables building routes (like /upload) to receive resume files and return results.*
- *Paired with Flask-CORS to allow frontend-backend communication (e.g., from a web page or React app).*

next slide →

Flask Endpoint – /upload

- *The /upload endpoint is the core API route in the ResumeScoreX tool. It handles the resume file upload, processes the resume, and returns the match score based on the job description.*

- *Accepts File Upload*
  Receives a PDF resume via a POST request from the frontend form
  .

- *Validates the File : Checks if the uploaded file is a valid PDF.*

- *Saves the File : Stores the uploaded file in a local uploads/ directory for processing.*

- *The /upload endpoint is the connection point between the user and the NLP engine, making real-time resume scoring possible via a simple HTTP call.*

next slide →

Project Summary

Key Features

- *PDF Resume Upload via a web interface.*
- *Text Extraction from resumes using pdfplumber.*
- *TF-IDF & Cosine Similarity to compare resumes with job descriptions.*
- *Flask API backend to handle scoring and integration.*
- *Returns a match score (0–100%) to help recruiters shortlist candidates.*

💡 *Impact*

- *Saves Time: Automates the manual resume review process.*
- *Improves Accuracy: Objective ranking based on skills and relevance.*
- *Scalable: Can be extended for different roles and larger datasets.*

next slide →

Thank You!

We appreciate your time and attention!

📄 *Built with: Python, Flask and NLP*

🚀 *Goal: Automating and enhancing the hiring process with smart resume analysis*

💬 *Open to Questions and Feedback*

Ganga Raju K N

Vishwajeet Kale