The goal of this project is to **match candidate resumes with job descriptions** using text similarity techniques.
By calculating how closely the text of a resume matches the text of a job posting, we can automatically identify the best-fit candidates for specific roles.

Step 1: Data Collection & Preparation

We started by collecting **sample resumes and job descriptions** in plain text format.
Each resume and job description included relevant skills, experience, and keywords.
The data was stored in structured Pandas DataFrames:

- **Resumes:** Contained fields resume_id and resume_text
- **Jobs:** Contained fields job_id and job_text

Outcome: Clean, structured dataset ready for text analysis.

Step 2: Text Preprocessing

All text was cleaned and standardized to prepare for analysis.
Steps included:

1. Converting all text to lowercase
2. Removing punctuation, numbers, and special characters
3. Tokenizing text into individual words
4. Removing English stopwords (like *is, the, and, of*)
5. Applying **lemmatization** using WordNetLemmatizer to reduce words to their base form (e.g., *developing → develop*)

Outcome: Consistent and meaningful text data for both resumes and job descriptions.

Example (before & after cleaning):

Before: "Experienced software engineer skilled in Python, machine learning, and web development."
After: "experienced software engineer skilled python machine learning web development"

Step 3: Feature Extraction (Vectorization)

To compare text documents numerically, we used **TF-IDF (Term Frequency–Inverse Document Frequency)** vectorization.
This converts text into vectors based on how important a word is in a document relative to all other documents.

Formula intuition:

- Words that appear often in a specific document but rarely in others are given higher weights.
- Common words across all documents have lower weights.

Outcome:
Each resume and job description was transformed into a vector representation sharing the same feature space.

Step 4: Similarity Computation

To measure similarity between resume and job text vectors, we used **Cosine Similarity**:

$$\text{Cosine Similarity} = \frac{A \cdot B}{||A||||B||}$$

Where AAA and BBB are TF-IDF vectors of a resume and job description.

This gives a value between **0 and 1**, where:

- **1** → Perfectly similar
- **0** → Completely dissimilar

Outcome:
A **similarity matrix** was generated showing how well each resume matches each job.

Example Output:

Resume →	**Job 1 (ML Engineer)**	**Job 2 (Frontend Developer)**
R1 (AI Engineer)	0.88	0.32
R2 (Data Analyst)	0.52	0.20
R3 (Frontend Dev)	0.30	0.91

Interpretation:

- Resume R1 best matches Job J1 (Machine Learning Engineer) with 88% similarity.
- Resume R3 best matches Job J2 (Frontend Developer) with 91% similarity.

Step 5: Visualization

Two visualizations were created to make results more interpretable:

1. **Heatmap:**
   Displayed the similarity matrix with color intensity showing stronger matches.
2. **Bar Charts:**
   For each job, displayed similarity scores across all resumes to easily identify top matches.

Outcome:
Visually clear understanding of which resumes are best suited for each job.

Step 6: Documentation & Reporting

All findings, code workflow, and limitations were documented.
We also discussed future improvements for accuracy and scalability.

Outcome:
A complete end-to-end report with methodology, analysis, and insights.

Why We Chose TF-IDF + Cosine Similarity Over Other Models

Model	Description	Why Not Used
CountVectorizer	Only counts word frequency.	Ignores word importance; overemphasizes common words like “skills” or “experience”.
Word2Vec / Doc2Vec	Captures semantic meaning using neural embeddings.	Needs a large dataset and long training time; difficult to tune for small sample text.

Model	Description	Why Not Used
BERT / Sentence Transformers	Deep contextual models; understand sentence meaning.	High computational cost; not necessary for small-scale similarity tasks.
TF-IDF + Cosine Similarity	Lightweight, interpretable, and effective for text matching.	Best trade-off between simplicity and accuracy.

Chosen Model: TF-IDF + Cosine Similarity

- Efficient even on small datasets.
- Captures word importance well.
- Easy to explain and visualize for demonstration or evaluation.
- Provides quantitative similarity scores that are easy to rank.

Final Results Summary

Job ID	Top Matching Resume	Similarity Score
Job 1 – Machine Learning Engineer	Resume R1	0.88
Job 2 – Frontend Developer	Resume R3	0.91

Conclusion:

The model accurately identified resumes aligned with the technical requirements of each job description.

Limitations

- TF-IDF doesn't capture context (e.g., "AI" and "Artificial Intelligence" treated differently).
- Matching depends heavily on word overlap.
- Doesn't account for experience years or proficiency levels.

Final Outcome

The Resume–Job Description Matching system:

- Cleans and preprocesses textual data.
- Converts text to vector form using TF-IDF.
- Computes cosine similarity between resumes and jobs.
- Displays ranked matches and similarity visualization.
- Suggests improvements for more intelligent matching in the future.