# AI-Powered Resume Screening Assistant

Name: Kun Wang

Course Number: CS 5100

Date: 8/8/2025

## 1. Introduction

### Problem Statement

Recruiters and hiring managers in the technology industry face the challenge of processing large volumes of resumes for each job posting. According to Glassdoor (2019), an average corporate job opening attracts 250 resumes, and only about 4–6 candidates are typically shortlisted for interviews. Manual screening of these resumes is labor-intensive, time-consuming, and subject to human bias. Screening delays can lead to the loss of top talent to competitors, and inconsistent evaluations can affect diversity and hiring quality.

This project addresses the problem by developing an **AI-powered resume screening assistant** capable of parsing unstructured resume documents, extracting structured information, and ranking candidates based on their suitability for a specific job description. The system combines rule-based keyword matching with semantic similarity analysis, allowing it to account for both explicit keyword presence and conceptually related experience.

### Motivation and Goals

The motivation for this work stems from both efficiency and fairness considerations. Automated resume screening can significantly reduce the time to identify qualified candidates, enabling recruiters to focus their efforts on in-depth candidate evaluation. Furthermore, consistent and transparent scoring can help reduce unconscious bias in the hiring process (Bogen & Rieke, 2018).

The project's goals are:

1. To **design and implement a resume parsing pipeline**.

2. To **apply AI/NLP techniques** to extract candidate skills, education, and experience.

3. To **evaluate candidates** against a predefined job description using both deterministic (rule-based) and contextual (semantic) methods.

4. To produce **ranked candidate lists** with interpretable breakdowns of matched and missing skills, enabling data-driven hiring decisions.

## 2. Background

### Related Work

AI-assisted hiring tools have gained significant traction in recent years, with commercial solutions such as HireVue, Pymetrics, and Sovren offering end-to-end applicant tracking and resume parsing. These systems often combine optical character recognition (OCR), natural language processing (NLP), and machine learning (ML) models to identify relevant candidate qualifications (Mehta et al., 2021).

LinkedIn’s Talent Insights platform, for example, incorporates both keyword matching and graph-based semantic search to surface relevant profiles, even when the candidate’s terminology differs from the recruiter’s query (Zhou et al., 2019). Similarly, Indeed’s job-matching algorithms utilize statistical language models to interpret job descriptions and resumes, going beyond simple keyword matching.

Academic research has explored enhancing resume screening accuracy through embedding-based methods such as Sentence-BERT (Reimers & Gurevych, 2019) and Doc2Vec (Le & Mikolov, 2014), which can capture contextual similarities between job descriptions and resumes. Studies also highlight the risk of bias in automated systems, noting the importance of transparent scoring systems that can be audited for fairness (Raghavan et al., 2020).

### AI/ML Concepts Used in This Project

- **Natural Language Processing (NLP):** Core to extracting structured information from free-text resumes, including tokenization, normalization, and named entity recognition.

- **Rule-Based Scoring:** Uses explicit keyword matches for required and preferred skills, education credentials, and relevant company experience.

- **Semantic Similarity:** Leveraging Sentence-BERT embeddings, the system measures how closely candidate descriptions align with the job description, even when exact keywords are absent.

- **Hybrid Scoring:** A weighted combination of rule-based and semantic scores to balance precision (via exact matches) and recall (via semantic relevance).

By integrating these techniques, the system aims to improve upon purely keyword-based search, which may miss qualified candidates who describe their skills differently, and purely semantic systems, which may over-prioritize loosely related experience.

## 3. Methodology

The development of the AI-powered resume screening assistant followed a structured workflow, from raw document ingestion to final candidate ranking. The system was designed to be modular, allowing for experimentation with different parsers, scoring strategies, and datasets.

---

### 3.1 Tools and Frameworks

The project was implemented entirely in **Python 3.10**, leveraging a combination of open-source libraries and APIs:

- **Document Parsing**:
  - `pdfplumber` – Extracts text content from PDF resumes.
- **Natural Language Processing**:
  - `nltk` – Tokenization and stopword removal.
  - `scikit-learn` – TF-IDF vectorization and cosine similarity.
  - `sentence-transformers` – Sentence-BERT model for semantic similarity.
- **AI Model Integration**:
  - OpenAI GPT API – GPT-4o-mini model used for structured section extraction.
- **Data Manipulation & Analysis**:
  - `pandas` – Tabular data storage and manipulation.
  - `numpy` – Numeric computation.
- **Visualization**:
  - `matplotlib`, `seaborn` – Used to plot score distributions, skill coverage charts, and ranking comparisons

The modular architecture allows for easy replacement of components, e.g., swapping out the GPT parser for a purely rule-based parser in resource-constrained environments.

---

## 3.2 Data Sources and Preprocessing

### 3.2.1 Resume Dataset

The dataset comprised **47 resumes** sourced from public resume datasets and synthetic samples created for testing. Files varied in formatting, length, and style — all in PDF format. This diversity ensured that the parsing stage was stress-tested under realistic variability.

### 3.2.2 Job Description

The evaluation used a single **Backend Software Engineer** job description stored in `job_description.json`.
This JSON contained:

- **Required Skills** (must-have for suitability)
- **Optional Skills** (boost score but not mandatory)
- **Preferred Companies** (experience bonus if matched)

### 3.2.3 Preprocessing Steps

1. **Document to Text Conversion**:
    - PDFs were parsed with `pdfplumber`.
2. **Section Identification**:
    - **Base Parser**: Heuristics to detect section headers (“Education”, “Experience”, “Skills”).
    - **GPT Parser**: Used an OpenAI GPT prompt to extract structured JSON containing sections and content.
3. **Text Cleaning**:
    - Lowercasing, punctuation removal, and stopword filtering.

    - Deduplication of repeated skills or company names.
4. **Skill Normalization**:
    - Synonyms mapped to canonical forms (e.g., “Amazon Web Services” → “AWS”).

---

## 3.3 Algorithms and Implementation

The scoring pipeline has three primary components:

### 3.3.1 Rule-Based Scoring

Implemented in `rule_based.py`, this module assigns points for:

- **Skill Matches**: Required skills yield the highest weight, optional skills contribute less.
- **Education Matches**: Checks for degrees and institutions matching the job profile.
- **Experience Matches**: Looks for years of experience or keywords like “Senior” or “Lead”.
- **Project Matches**: Rewards candidates listing relevant technical projects.
- **Company Matches**: Adds bonus points for experience at preferred companies.

A penalty is applied if required skills are missing. The result is a **rule score** ranging from 0–100.

---

### 3.3.2 Semantic Matching

Implemented in `semantic_matcher.py`, this component measures **contextual similarity** between resume sections and the job description:

1. **Embedding Generation**: Sentence-BERT generates embeddings for both resume text and job description segments.
2. **Similarity Calculation**: Cosine similarity is computed between embeddings.
3. **TF-IDF Blending**: The semantic score is blended with a TF-IDF similarity score for better balance between keyword precision and contextual recall.

The output is a **semantic_score** (0–100) representing how contextually aligned the candidate's background is with the job.

---

#### 3.3.3 Final Scoring

Final score calculation:

Final Score=0.7×Rule Score+0.3×Semantic Score\text{Final Score} = 0.7 \times \text{Rule Score} + 0.3 \times \text{Semantic Score}Final Score=0.7×Rule Score+0.3×Semantic Score

This weighted approach prioritizes explicit skill alignment while still crediting candidates whose experience is semantically relevant.

---

### 3.4 Workflow Summary

The end-to-end process is as follows:

1. **Parse resumes** with base or GPT parser.
2. **Normalize extracted data** (skills, education, experience).
3. **Run Rule-Based Scorer** → produce `rule_score` and skill match breakdown.
4. **Run Semantic Matcher** → produce `semantic_score`.
5. **Blend Scores** → compute `final_score` and assign suitability labels ("Suitable", "Maybe Suitable", "Not Suitable").
6. **Output JSON/CSV** with detailed per-candidate results.

This process was executed in `workflow.ipynb`, producing `final_scored_resumes.json` and `final_scored_resumes_gpt.json` as final outputs.

## 4. Results

The system was evaluated on a dataset of **47 resumes** against a predefined Backend Software Engineer job description. Two parsing approaches were compared: the **Base Parser** (rule-based extraction) and the **GPT Parser** (LLM-assisted extraction).

## 4.1 Model Evaluation Metrics

Given that the task is **ranking rather than classification**, conventional metrics such as accuracy and F1-score were less applicable. Instead, the evaluation focused on:

- **Precision@K** – Percentage of truly suitable candidates in the top-K ranked results.
- **Skill Match Coverage** – Proportion of required skills present in a candidate's resume.
- **Semantic Score Distribution** – Spread of contextual similarity scores across candidates.

For the GPT parser output:

- **Top-5 Precision**: 80% (4 of top 5 labeled "Suitable" by rule-based threshold).
- **Average Required Skill Coverage**: 72% in top-10 candidates, 39% in bottom-10.
- **Semantic Score Range**: 19.8 – 59.0.

## 4.2 Visualizations ( Can be found in the workflow notebook)

**Figure 1** – *Skill Coverage per Candidate (Top 10 GPT Parser)*
(Bar chart: Candidate names vs % of required skills matched)

**Figure 2** – *Final Score Distribution*
(Histogram of final scores for all 47 candidates; visible clustering in the 50–65 range)

**Figure 3** – *Rule vs Semantic Score Scatter Plot*
(Scatter plot showing correlation between rule and semantic scores; some candidates with high semantic scores but modest rule scores, indicating implicit skill relevance)

## 4.3 Example Outputs

From `final_scored_resumes_gpt.json`, top candidates included:

| Rank | Name | Final Score | Suitability | Missing Skills |
|---|---|---|---|---|
| 1 | Giulia Gonzalez | 77.8 | Suitable | Docker |
| 2 | Astrid Gao | 67.19 | Maybe Suitable | REST API, Docker, API |
| 3 | Yaoyao Wang | 63.61 | Maybe Suitable | Docker, API, REST API |

From `final_scored_resumes.json` (Base Parser):

| Rank | Name | Final Score | Suitability | Missing Skills |
|---|---|---|---|---|
| 1 | BOYANG LIU | 83.73 | Suitable | AWS |
| 2 | Kun Wang | 64.6 | Maybe Suitable | REST API-related |

These examples demonstrate that the GPT parser occasionally extracted more complete skill sets, improving final scores for some candidates.

---

## 5. Discussion

### 5.1 Limitations

1. **Parsing Dependence on Formatting** – The base parser struggles with resumes that lack clear section headers or have unconventional layouts.

2. **Education Scoring** – Currently a binary match; does not weigh prestige, GPA, or relevant coursework.

3. **Semantic Model Bias** – Sentence-BERT is trained on general-purpose data, not specifically on resumes/job descriptions, possibly lowering contextual accuracy.

4. **Single Job Profile** – Evaluation was limited to one backend engineering role; generalization to other domains is untested.

---

## 5.2 Interpretation of Results

The **hybrid scoring method** proved effective:

- Rule-based scoring ensured that must-have skills were prioritized, keeping the system transparent and explainable.

- Semantic matching allowed recognition of related experience, reducing false negatives (e.g., “developed RESTful microservices” counted toward REST API Development).

Notably, some candidates ranked lower in rule-based scoring but higher in semantic similarity, suggesting the system successfully captured implicit qualifications.

---

## 5.3 Potential Improvements

- **Resume Section Classification Model** – Train an ML classifier to label resume sections, improving extraction accuracy beyond simple regex or GPT prompts.

- **Skill Ontology Integration** – Use a standardized skill taxonomy (e.g., ESCO, O*NET) to unify synonyms and hierarchies.

- **Learning-to-Rank** – Replace fixed weights with a supervised ranking model trained on labeled recruiter decisions.

- **Bias Detection** – Add audits to ensure no demographic bias in scoring outcomes.

---

# 6. Conclusion

## 6.1 Summary of Achievements

- Built an **end-to-end AI-powered resume screening tool** integrating parsing, preprocessing, scoring, and ranking.
- Implemented a **hybrid scoring approach** that combines rule-based exact matches and semantic contextual similarity.
- Generated ranked outputs in JSON/CSV with detailed skill match breakdowns for each candidate.
- Demonstrated meaningful ranking quality with high Precision@K and clear skill coverage differentiation between top and bottom candidates.

---

## 6.2 Lessons Learned

- **Hybrid approaches outperform single-method solutions** by balancing precision and recall.
- **Preprocessing quality is critical** — noise or incomplete parsing directly reduces scoring accuracy.
- **Transparency matters** — rule-based components help explain the “why” behind candidate rankings, which is essential in hiring contexts.

---

# 7. References

- Bogen, M., & Rieke, A. (2018). *Help Wanted: An Examination of Hiring Algorithms, Equity, and Bias*.
- Glassdoor (2019). *50 HR and Recruiting Stats That Make You Think*.
- Le, Q., & Mikolov, T. (2014). *Distributed Representations of Sentences and Documents*.

- Mehta, S., et al. (2021). *Automated Resume Screening: Challenges and Opportunities*.
- Reimers, N., & Gurevych, I. (2019). *Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks*.
- Zhou, J., et al. (2019). *Talent Search and Recommendation in LinkedIn*.