Resume Grader: An NLP-Based ATS Analysis Tool

https://github.com/matth101/6320-final

Final Report

Motivation

In today's job market, over 90% of Fortune 500 companies use Applicant Tracking Systems (ATS) to screen resumes. As a student applying for internships, understanding how these systems evaluate resumes became a crucial interest to me! While many tools exist for job searching and application tracking, few provide transparent insights into how ATS systems analyze resumes. As an exploratory exercise, and to also help demystify this process, I decided to create a tool that shows candidates exactly how their resumes match against job descriptions.

Technical Approach and Evolution

Approach evolved from simple pattern matching to a sophisticated NLP pipeline. Initially, I started with basic keyword matching, which quickly proved insufficient for understanding the nuanced relationships between job requirements and candidate qualifications. Here's how our solution evolved:

Initial Implementation:

```
def calculate_keyword_match(self, resume_keywords, job_keywords):
    matching_keywords = set(resume_keywords_lower).intersection(set(job_keyw
    return len(matching_keywords) / len(job_keywords_lower)
```

The journey in building this resume grader began with the most straightforward approach: basic keyword matching. Initially, I implemented simple counting of overlapping words between resumes and job descriptions. While this approach was easy to implement, it quickly showed its limitations. For example, it would miss that a "machine learning engineer" was qualified for an "ML developer" position, or that "Python programming" and "Python development" were effectively the same skill.

This realization led to the first major enhancement: a comprehensive skills detection system. I built extensive lists of both technical and soft skills, and more importantly, implemented a sophisticated variation handling system. This meant the tool could now understand that "JS" and "JavaScript" were the same skill, or that "ML" and "Machine Learning" referred to the same capability. I also introduced weights to the scoring system, recognizing that technical skills often carry more weight in technical positions than soft skills.

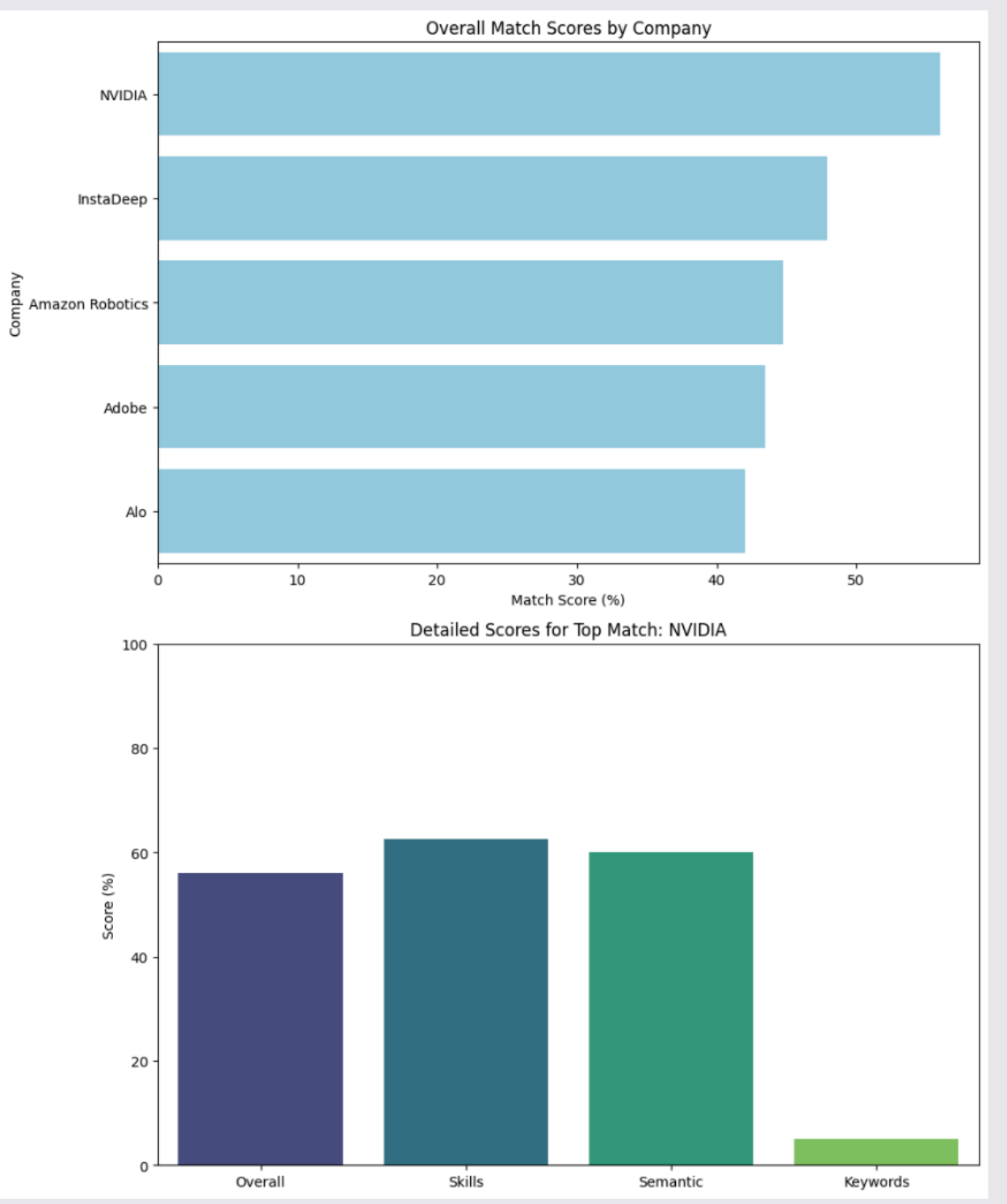
```
# Example of enhanced skills detection
skill_variations = {
    "machine learning": ["ml", "machine learning", "deep learning"],
    "artificial intelligence": ["ai", "artificial intelligence"],
    "javascript": ["javascript", "js", "node.js"]
}
```

The most significant leap forward came with the integration of BERT-based sentence transformers for semantic analysis. This advanced NLP technique allowed the system to understand the context and meaning of text, not just match exact words. For instance, if a resume mentioned "building scalable web applications," the system could now understand this was relevant to a job requiring "experience in developing enterprise-level software systems," even though the exact words didn't match.

```
def calculate_semantic_similarity(self, resume_text, job_text):
    resume_embedding = self.semantic_model.encode([resume_text_clean])[0]
    job_embedding = self.semantic_model.encode([job_text_clean])[0]
    return cosine_similarity([resume_embedding], [job_embedding])[0][0]
```

The final scoring system became multi-dimensional, carefully weighted to reflect real-world hiring priorities. I allocated 60% of the score to direct skills matching, recognizing this as the primary indicator of qualification. Semantic similarity accounts for 30%, capturing the broader context and relevant experience that might not be expressed in exact skill matches. The remaining 10% goes to keyword analysis, maintaining a connection to traditional ATS systems' methods.

Visualization and Output



The visualization system provides two key insights through intuitive charts. The first is a comparative bar chart showing match percentages across different job

positions, allowing quick identification of the most promising opportunities. The second visualization focuses on skills analysis, using a color-coded system (green for matching skills, red for missing skills) to highlight areas for resume improvement. These visualizations transform complex matching calculations into actionable insights, helping users understand exactly where their resume needs enhancement. The charts were implemented using matplotlib and seaborn, with careful attention to readability and clear data presentation.

Future Work, Lessons Learned

The development of this resume grader has opened up several interesting avenues for future exploration. One particularly promising direction would be integration with job search APIs, allowing real-time analysis of newly posted positions. Imagine a system that not only grades your resume but actively suggests relevant job postings and provides targeted improvement recommendations!

I also discovered the complexity of context-aware text analysis during development. While the current implementation uses pre-trained models effectively, there's potential for fine-tuning these models specifically for the technical recruiting domain. This could involve training on a large corpus of successful resume-job matches to better understand what makes a strong candidate-position fit.

Another fascinating extension would be the incorporation of industry-specific scoring adjustments. We found that different sectors emphasize different aspects of a candidate's profile - what makes a strong match for a software engineering role might differ significantly from what's valued in a data science position. Building this intelligence into our scoring system would provide more nuanced and actionable feedback.

The most significant technical lesson was the importance of balancing sophisticated NLP techniques with practical usability. While I could have implemented more complex analysis methods, I found that clear, actionable feedback was more valuable to users than marginally more accurate but less interpretable results.

Contributions

Self-Scoring

Matthew Hui

- 80 points - Significant exploration beyond baseline
  - Implemented semantic matching using transformer models
  - Developed comprehensive skills detection system
  - Created visualization components for analysis
- 25 points - Innovation/Creativity
  - Flipped traditional ATS approach to benefit candidates
  - Developed multi-dimensional scoring system
  - Could have integrated more complicated features to improve idea
- 10 points - Highlighted complexity
  - Integration of multiple NLP techniques
  - Handling various document formats and structures
- 5 points - Visualization/documentation
  - Created visualizations - could have been more interactive though, in a web app for instance
  - Comprehensive documentation and demo

Total: 120 points