Resume scanning using machine learning and AI

by Pelle Vedsmand and Nicolai Rosendahl

Abstract

This project evaluates and compares three approaches for the assessment of resumes against job descriptions: A machine learning solution implementing and evaluating 4 different models, a local AI solution leveraging Retrieval-Augmented Generation and prompt engineering, and ChatGPT using prompt engineering. Using a consistent test dataset, we measure six key metrics—match score accuracy, label score accuracy, execution speed, response consistency, response clarity from an HR user perspective, and sensitivity to input variations. The goal is to determine the strengths and limitations of each method.

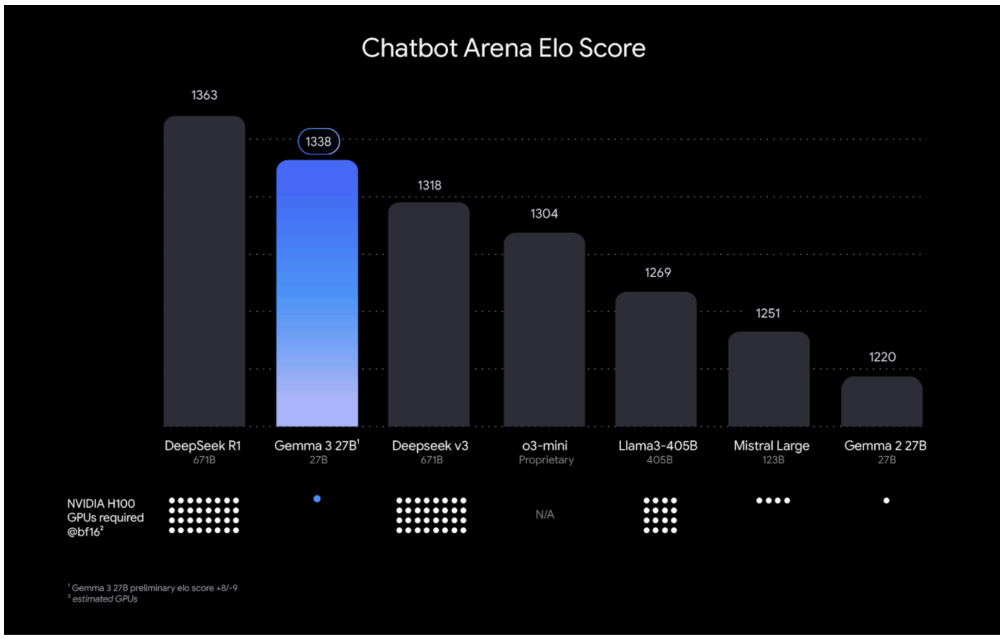
Goal of the project

The objective is to investigate how different AI-driven approaches perform when tasked with rating resumes for specific job fits, with an emphasis on not only accuracy but also interpretability, consistency, and usability for HR professionals. By comparing machine learning models, a local RAG-based AI, and prompt-engineered well known LLM, we seek insights into their suitability for real-world resume evaluation.

Process of finding our models

Local AI model

Given the context of the project - the important features of the model we were looking for were proficiency in semantic language understanding and the possibility of running said model on a machine with relatively limited hardware resources.



gemma3

The decision ended up being the Gemma3 model, based on a combination of a high Elo score (a relative score used to grade models against each other) and the fact that Gemma3 seemingly could perform as well as other top performers utilizing fewer GPUs and parameters. The decision ended up being Gemma3:4b, since downscaling the model to a size able to run efficiently on a laptop was necessary.

As for fine tuning the model prompt engineering was utilized in order to make the desired result as tailored to the application as possible, by defining its traits, task, tone and target, and tweaking these until a desired result was achieved. (Link to the final prompt: AIML-Exam/Webapp/openwebui_api.py at main · pelle112112/AIML-Exam)

The model was provided with a knowledge base with the intention of providing necessary context about the company's requirements, including job descriptions and their values and culture. Fine tuning of RAG included feeding the knowledge base different formatted files, I.E. plain text- or json files, structuring the knowledge base in different ways, I.E. using directories, and ensuring a consistent file naming structure that would allow the model to find the referred files as consistently as possible.

Baseline LLM

To make the results measurable, a well known LLM is included as a baseline to better compare our findings. GPT-4.1-mini was chosen because ChatGPT generally is considered one the best AI chatbots. (The best AI chatbots of 2025: ChatGPT, Copilot, and notable alternatives | ZDNET) Their 4.1-mini model would combine the desire of keeping the experiment as fair as possible, by not choosing the biggest model available, while also being free to use.

Machine Learning models - fine tuned and trained

The goal of this stage was to categorize resumes into predefined job categories (e.g., Software Developer, Data Scientist, Web Developer) as a first layer of automated screening.

Research would indicate that focusing on fine-tuning a BERT model would provide a very accurate and context-aware solution.
*(*Reference*:* https://onlinelibrary.wiley.com/doi/full/10.1155/cplx/1884264 *)*
The model trained with BERT therefore became the baseline of the classification models.

The BERT model was fine-tuned using the Kaggle Resume Dataset and trained with the Adam optimizer, which has a dynamic learning rate, and a batch size of 8 over three epochs. No learning scheduler was added for the training, which could have further increased the models performance.

Due to limited time and compute resources, it was not possible to extend the training further, however monitoring loss and accuracy across epochs ensured stable performance, while avoiding overfitting. During training the model showed strong performance in correctly labeling resumes with their most likely job categories, thanks to its deep understanding of context and semantics.

To benchmark BERT's performance and understand how well lighter models perform in comparison, three traditional classification models were added:

- Random Forest
- Logistic Regression
- Decision Tree

These were implemented using TF-IDF vectorized text features and while all models were capable of producing reasonable results, they primarily relied on keyword presence rather than contextual understanding.

In addition a simple Cosine Similarity approach was implemented using TF-IDF vectors. This method calculates the cosine angle between a vectorized resume and job description in the vector space to estimate their semantic similarity.

All models were used exclusively in the first stage of resume evaluation, when resumes were assigned one or more job categories. The second stage of screening (semantic matching to job descriptions) was handled separately using the Cosine Similarity approach.

Experiment

We designed an experiment to systematically compare the different approaches using the following metrics:

1. Match Score — The raw semantic score generated by each system for each resume-job pairing.

2. Label Score — The average of the correct labeling score for the resume to the target job application.

3. Speed of Execution — Time measured from prompt submission to system response completion.

4. Consistency — Qualitative scoring (1-5) assessing how closely each system's response matches a predefined format and structure.

5. Response Clarity — Qualitative scoring (1-5) evaluating how clear and accessible the explanations are to an HR-employee, avoiding technical jargon or prompt echoes.

6. Sensitivity — Variability in outputs measured by running identical prompts multiple times and testing with slightly altered resumes to assess score and rationale stability.

The inputs consist of generated job descriptions paired with generated resumes, with each experiment entry testing for positive cases, meaning the experiment did not include cases of resumes being matched with wrong job descriptions. Each solution was fed the same inputs and relatively same prompts, only adjusted to fit the solution's specific interface. Responses were collected and evaluated against the metrics, with human raters reviewing clarity, consistency from an end-user (HR) perspective and sensitivity.

Results

These are the averaged values of our test results. To see the unedited version go to: [AIML-Exam/documentation/solution comparison/Eksperiment med AI og ML - Ark1.pdf at main · pelle112112/AIML-Exam]

Approach	Match Score	Consistency (1–5)	Response Clarity (1–5)	Speed (s)	Sensitivity
Local AI (Gemma3)	0.87	4.05	4.65	7.25	Low
ChatGPT (GPT-4.1-mini)	0.88	5	5	5.45	Low
Cosine Similarity (TF-IDF)	0.66	5	-	0.03	High

Model	Label Score	Speed (s)	Sensitivity
BERT	0.97	0.3	Medium/High
Random Forest	0.69	0.03	Medium
Logistic Regression	0.75	0.03	Low
Decision Tree	0.85	0.04	Very High

Conclusion

The LLM models all scored high on all the metrics with a high match score and low sensitivity, giving them a clear edge over the rest as an overall approach to screening resumes. They give a semantic analysis with an output easily understandable by an HR department with low technical understanding.

The cosine sim scores low compared to the LLMs, but this makes sense, as the similarity score is calculated by matching the vectorized resume against the vectorized job application, meaning it is more likely to get a high cosine sim score by uploading a job description instead of a resume.Cosine sim outperforms the LLMs more than 100x on speed, and does not require any substantial computer power, which could offer a solution for companies with low resources not in a position to pay for the Chat GPT tokens or a server strong enough to

run a local AI model. Although it must be taken into consideration that a Chat GPT solution could be suitable as a free option for these companies if they don't require automation.

The test results indicate that it is more consistent to use a well known LLM for this task rather than a local AI, but that could be a reflection of using pure prompt engineering versus a combination of prompt engineering and RAG. This is due to the consistency rating being based on the output and the response template being defined via RAG in one case and prompt engineering in the other. After light testing (not included in the experiment) it seems that providing the structure of the answer through the prompt rather than RAG yields a more consistent result.

All of the machine learning models have a tendency to value frequently occuring job categories like "Software dev" higher than the rest, which often leads to mislabeling of less common categories. Due to time constraints methods such as data resampling and tuning of class weights was not explored, but it is reasonable to believe that implementing solutions like these would improve the result of the classification models.
Of the machine learning models BERT showcases a very high label score compared to the other models, but surprisingly lacks in the sensitivity metric - adding noise to the input data (like contact information) heavily affects the result.
BERT is limited to resumes being less than 512 tokens, which can result in loss of context for longer resumes. It is likely that in a production environment, the resume texts would require text modification to get a less sensitive result, such as removing irrelevant text sections like contact information. It is possible however that this issue could be avoided by splitting the input into subtexts and feeding it into the BERT multiple times and averaging the prediction labels, but this could result in other issues such as loss of context of the subtexts, as they could be missing critical subwords defining the semantic meaning. Configuring BERT to allow consumption of text larger than 512 tokens, could also be a solution, but was not feasible without significant hardware upgrades.

Overall, it is recommended to choose a solution that aligns with the company's budget and technical capacity, as every solution provides its own upside in regards to cost and effort to maintain. For low-budget scenarios, combining models like BERT with cosine similarity offers a lightweight and efficient option, but keep in mind this solution requires extensive knowledge on how to train and tune machine learning models.
For higher-budget deployments, companies can consider purchasing a ChatGPT enterprise license or deploying their own local AI solution, with consideration towards other concerns like privacy/security.