Resume Ranking Solution

ABSTRACT: The resume ranking describe in this docs which, although deviates from the assignment requirement, demonstrates my findings on how to make a resume ranking solution which actually solves all the problems of today's ATS.

PART I: Extraction

This is a fundamental process of the process. Apart from extracting all the text content, It is important to convert them into a semi-structured data for more optimized use. It could be done by

1. LLM Json Extraction: Easiest to configure and tune it for the use case. Might be slightly expensive in the long run than the traditional NLP extraction.
2. NER: Name Entity Extraction has the capability to do very well with very limited compute resource as compared to LLMs. Requires higher number of training data.
3. IDEAL CASE = SLM: Using LLM initially to extract the data from the resume and then train the SLM to do the same, provides LLM level performance at fraction of the cost.

The extraction includes extraction of features like:

- Education
- Experience
- Projects
- Certificates
- Custom fields
- Skills

Since we're using Language model, we can always add additional extraction fields which might vary with Job description.

Additional Requirements for HR :

ATS tends to focus more on the quantitative aspects of resume: no. of years of experience, no. of skills keyword matching, etc. Which often results in inaccurate sorting of profiles.

In our approach we'll make sure that the scoring is done in a qualitative way. For that we'll use LLM again.

LLM could be pretty random when asked to provide a score. For eg. Ask llm to rate a candidate's education between 0 – 5. It might provide two different scores for the same candidate. Hence, to avoid this we ask HR to add three levels of generic description of a candidate. Hence, we provide a benchmark for the LLM to compare the input with. The three levels might be the two extremes (0 and 5) and a median (2.5)

By this method we get the ratings of individual attributes of a candidate. The weightage of each can be configured by the hiring team.

Approach II: Fine-tuning of Embedding models.

For a very large scale of resume, fine-tuning an embedding model on individual attributes is also a great alternative. It returns a score between -1 and +1. The fine-tuning needs to be done using contrastive learning method. An HR, when provided a JD and two resumes, have to rate among them which is closer and which is not. When done for a large volume of resume, we'll have a lightweight scoring mechanism which captures a sorting intuition of a human.

FILTER 1 (MUST HAVE)

We filter out the candidates who satisfy the threshold score of MUST have features in their resume.

FILTER II (GOOD TO HAVE)

Candidates are again sorted according to the score. And top 10 candidates are filtered out.

FILTER III (AI Sorting)

The last 10 candidates can be sorted using bubble sorting to provide more refined sorting.