Hire Help – A Smart Interview Scheduler

Abhishek Singh, Aryan Sanjay Tandon, Atharva Singh, Prof. Sansar Singh Chauhan
Department of Computer Science and Engineering, G.L. Bajaj Institute of Technology and Management, Greater Noida.

***Abstract*— With the continuously expanding job market, with varying requirements, it has become essential for organizations to make their hiring process smooth and reduce the time consumed in this process as much as possible, so that they can focus more on hiring quality employees and not spend much time on redundant tasks. HireHelp – A Smart Interview Scheduler resolves the issues that traditional recruitment faces with the modern job market requirements and increases the efficiency of some steps in the traditional recruitment process by adapting to the latest technology trends specifically, Machine Learning and Large Language Models. The primary aim of this paper is to reduce the workload on hiring managers, and make the hiring process more efficient by automating redundant tasks also making the process more streamlined using ML Techniques like text similarity by embeddings generation.**

Index Terms*— Hiring Process, Interview Scheduling, Large Language Models, Machine Learning, Text Similarity

I. INTRODUCTION

Most of the existing applications that are being used for hiring focus mostly on making the job application process easier and finding the best applicants for the job but they don't focus much on scheduling of interviews, selecting an interview panel, making sure the interview panel has the most in common with the candidate in terms of skills and area of expertise. Also sometimes screening a candidate just on the basis of their resume is not enough, as there is no way of verifying the information mentioned in their resume, which could lead to a lot of time and resources wasted on a candidate who does not have the practical skills and expertise for the job. So, there needs to be a better way to screen candidates as well.

A. Background

The recruitment process has evolved a lot with the use of latest technological trends like – artificial intelligence, machine learning, large language models etc. Especially the creation of algorithms that generate Application Tracking System (ATS) scores has impacted the whole process drastically. The use of such technologies allowed many tasks to be automated which were earlier tedious and redundant, but most of the focus has been on automating the application process using keyword-based algorithms which is not enough for the modern job market. The current market requires context-based algorithms for parsing resumes, and generating ATS scores based on this algorithm, a lot of work has been done in this regard as well but till now none of the algorithms have been accurate enough that large organizations would use them for all types of hiring, they use these algorithms usually for lower-level roles.

B. Motivation

The motivation behind developing HireHelp is to make the hiring process efficient at the same time make it easier for both the candidate and organization to track and manage the interview and application status. Although, the traditional recruitment process is simple to understand, it is not efficient and has some issues that need to be resolved for an efficient hiring process. The following were the major motivating factors for this paper –

- ❖ Need to improve efficiency of the hiring process, making it more scalable and manageable.
- ❖ Making the whole process easy to track for the candidates, and making the whole hiring process easier for them, by providing real-time updates and instant interview scheduling.
- ❖ Reducing time and cost for the whole process for organizations, also improving their resource utilization.
- ❖ Reducing human bias from the hiring process, by automating most of the tasks involved in the process and making it more fair for all *candidates.*

II. RELATED WORK

A. Comparative Study

Teodor Petrican et al. in [1] propose an automatic recommendation system based on skill matching

techniques that could be used as an external tool for an online recruitment platform. They propose a similarity metric based on ontology of the skills, they use 3 algorithms – exhaustive search algorithm, breadcrumbs algorithm, multi-source shortest algorithm to obtain a ranked skill-based matching between candidates and job offers. The major issue with this approach is that they have not talked about the scalability of this algorithm over real-world data, as there is a requirement of getting the output as fast as possible but using these 3 algorithms in an application might cause lag or even system crash if there are large number of users.

Bhushan Kinge et al. in [2] propose a system to automate the resume screening process for large-scale hiring, which would reduce the manual work, the proposed system extracts information from the resume using NLP techniques, and uses a dual model approach which employs 22 models, a prediction model based on K-Nearest Neighbours, and a recommendation model using cosine-similarity based recommendation engine. The issues with the approach are that it lacks adaptability, as it uses a static machine learning model that does not learn over time, it also has scalability issues as it uses a limited dataset, and since the dataset lacks diversity which might lead to unwanted biasness in the dataset. Magdalena Velciu et al. in [3] provides a detailed examination of job mismatch, focusing on employees with higher education qualifications, it discusses how educational mismatches affect wages, job satisfaction and work, there is a need for better alignment between educational offerings and occupational requirements to address skill gap. The issue with the study is that it fails to offer insights specific to sectors where job mismatches are prevalent.

B. Existing Systems

There are several existing systems that aim to streamline the recruitment process but most of them have limitations as they focus mostly on only a single aspect of the recruitment process.

2.1) Comparison Between Existing System

Existing Application	Approach	Scope of Improvement
LinkedIn [4][6]	Uses recommendation engine based on user connections, activity for personalized job suggestions and networking.	Lacks job matching based on context-based skill matching.
Indeed [7]	Uses Web scarping to provide job listings from various sources.	Lacks ML techniques for personalized job alerts based on user behaviour.
Glassdoor [8]	Provides insights on salary, company culture, job market condition.	Lacks real-time data feed for jobs, no job matching/ recccomendation.
Monster [9]	Offers an interface with advanced filtering options, ATS to recommend jobs.	Lacks real-time job tracking for job seekers.
ZipRecruiter [10]	AI-driven job matching algorithms.	Needs improvement in the training datasets for reducing bias.

Table 1: This above table shows the comparison of different existing systems for recruitment.

III. METHODOLOGY

The automated interview scheduling and assigning interview panel is implemented using a custom machine learning model, which uses some pre-trained models to assign interview panel. For scheduling interview, the availability of each interviewer is checked and based on this a common date and time is calculated on which all selected interviewers are available.

Fig 1: The Figure is a level-0 DFD for HireHelp.

The interface is designed such that it is easy for anyone to navigate and use the platform, the interface is divided into 2 dashboards – interviewer dashboard, and candidate dashboard. Each dashboard contains relevant functionalities for each role, some common functionalities include – viewing jobs, viewing scheduled interviews.

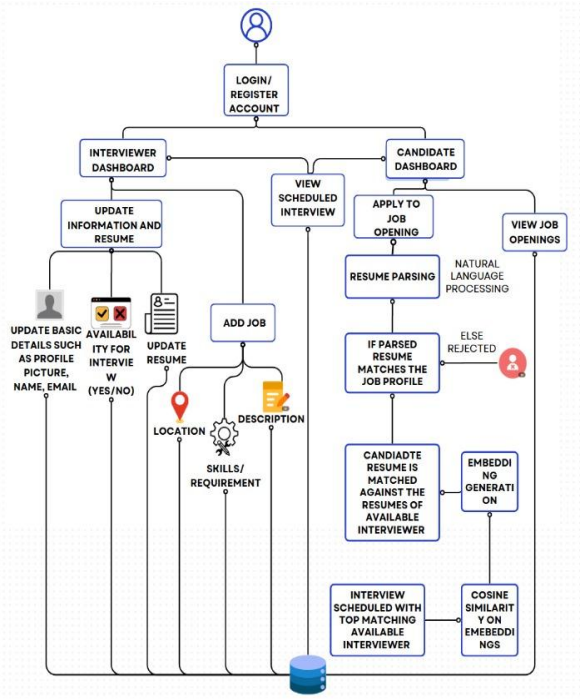

Fig 2: The figure contains a flow chart of HireHelp.

A. Screening Process

The screening process, uses the resume of the candidate to generate an ATS score, which is used to decide whether the candidate is eligible to apply for the job or not, but the information in the resume cannot be verified at the time of application. So, to make the screening process more accurate in identifying fake resumes, before uploading their resume the candidate needs to answer 5 questions that are generated using LLM, which uses the job description and the job requirements to generate these questions. After answering these 5 questions, these answers are again given to the LLM which generates a screening score for the candidate, after this the candidate can upload their resume, which is then used to generate a resume score. If both the resume and screening score of the candidate are above the cut-off score, then they are eligible to apply for the job.

Below is an implementation of the proposed resume parser, in Algorithm 1.

Algorithm 1 Resume Parser

```
Input: PDF path of the uploaded resume
Output: List of extracted skills

procedure extract_text_from_pdf(pdf_path)
    Open PDF document from pdf_path
    Initialize text as an empty string
    for each page in the PDF:
        Append page text to text variable
    Close the PDF document
    return text
end procedure

procedure parse_resume(pdf_path)
    # Step 1: Authenticate and obtain access token
    Initialize auth_url as "https://auth.emsicloud.com/connect/token"
    Set payload with client credentials (client_id, client_secret, grant_type, and scope)
    Set headers with "Content-Type" as "application/x-www-form-urlencoded"
    Make a POST request to auth_url with payload and headers
    if response status is 200:
        Extract access_token from response

    # Step 2: Extract text from the PDF
    Call extract_text_from_pdf(pdf_path) to get the resume_text
    Convert resume_text to lowercase

    # Step 3: Send resume text to the skills extraction API
    Initialize url as "https://emsiservices.com/skills/versions/latest/extract"
    Set headers with "Authorization" as "Bearer {access_token}" and "Content-Type" as
"application/json"
    Set API payload with "text" as resume_text and "confidenceThreshold" as 0.8
    Make a POST request to url with API payload and headers

    # Step 4: Process the response
    Extract skills list from the response JSON (key: "data")
    Initialize skills_json as an empty list
    for each skill in skills:
        Extract skill name and append it to skills_json

    return skills_json
end procedure
```

B. Assigning Interview Panel

A custom machine learning model is used to generate scores for each interviewer and candidate pair, for each candidate. Based on these scores, the top interviewers are chosen for the interview panel.

It comprises of 5 sub-models, all these models are pre-trained, each uses a different technique to generate embeddings for the resume of both interviewer and candidate. After generating the embeddings, a score is generated based on the similarity of the embeddings. All 5 models, give a score for each pair of candidate and interviewer, the final score of a pair is taken as the average of the scores given by the models.

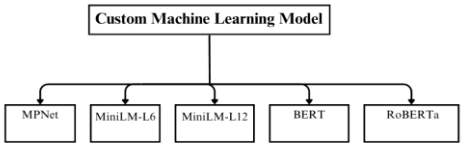

Fig 3: The figure contains the models used for selecting interview panel.

C. Performance Evaluation

3.1) ML Model Score

Comparision	M1	M2	M3	M4	M5	Mean
Atharva Vs Abhishek	0.70	0.69	0.69	0.89	0.64	0.72
Atharva Vs Shikhar	0.49	0.48	0.47	0.82	0.44	0.54
Atharva Vs Vinay	0.62	0.62	0.57	0.86	0.56	0.65
Atharva Vs Non-Tech	0.11	0.12	0.09	0.57	0.06	0.19
Abhishek Vs Shikhar	0.54	0.52	0.52	0.81	0.52	0.58
Abhishek Vs Vinay	0.59	0.56	0.52	0.83	0.58	0.62
Abhishek Vs Non-Tech	0.11	0.05	0.12	0.58	0.11	0.19
Shikhar Vs Vinay	0.43	0.38	0.44	0.80	0.43	0.49
Shikhar Vs Non-Tech	0.08	0.09	0.09	0.62	0.12	0.20
Vinay Vs Non-Tech	0.16	0.12	0.16	0.57	0.15	0.23

Table 2: The Table contains the similarity score generated by each ML model for 5 pairs of resume, 4 out of the 5 resumes are similar as they are from a technical background, and 1 is a non-tech resume, as we can see the score generated by the model after comparison of non-tech resume with other resumes is low, and also for some resumes the score is near 0.50, as they were neither similar nor different.(M1 – MPNet, M2 - MiniLM-L6, M3 – MiniLM-L12, M4 – BERT, M5 - RoBERTa).

Below is a sample implementation of MPNet Model (M1), in Algorithm 2.

Algorithm 2 ML Model

```
Input: Skills of both interviewer and candidate (skills_1,skills_2)
Output: Match score between both skills

procedure sbert_base_v2 (skills_1,skills_2)
        """Calculate similarity between two skill lists using SBERT (all-mpnet-base-v2)."""
        sen = skills_1 + skills_2
        model = SentenceTransformer('all-mpnet-base-v2')
        sen_embed = model.encode(sen)
        score = 0
        # Iterate through each skill in skills_1
        for i in range(len(skills_1)):
                if skills_1[i] not in skills_2:
                        # Cosine similarity between the current skill in skills_1 and all skills in skills_2
                        similarities = cosine_similarity([sen_embed[i]], sen_embed[len(skills_1):])[0]
                        # Add the maximum similarity if it's above the threshold (0.4)
                        max_similarity = max(similarities)
                        if max_similarity >= 0.4:
                                score += max_similarity
                else:
                        score += 1  # Add 1 for exact match
        # Normalize the score by the number of skills in the first list (skills_1)
        score = score / len(skills_1)
        return round(score, 3)
end procedure
procedure similarity (model_func,skills_1,skills_2):
        with torch.no_grad(): #Disable gradients to save memory
                result = model_func(skills_1,skills_2)
                return float(result)
end procedure

procedure mlscore (skills_1,skills_2)
        # Calculate similarity scores using different models
        ssbert_base_v2 = similarity(sbert_base_v2, skills_1, skills_2)
        sparaphrase_minilm_l6_v2 = similarity(sbert_paraphrase_minilm_l6_v2, skills_1, skills_2)
        sall_roberta_large_v1 = similarity(all_roberta_large_v1, skills_1, skills_2)
        average = (ssbert_base_v2+sparaphrase_minilm_l6_v2+sall_roberta_large_v1)/3.0
        return average
end procedure
```

D. Scheduling Interviews

Once an interview panel is assigned to the candidate, the portal then uses the availability data present for all the interviewers in the interview panel to schedule the interview on a common time slot. To find the common time slot the maximum and minimum time slots for the interviewers are calculated based on which an idle time slot of at least 1hour is searched in the availability data for each interviewer, if such a time slot is found for all the interviewers, then the interview is scheduled successfully.

In case no common time slot is available, an appropriate message is displayed to the candidate and the interview is set to be scheduled for a later period of time, where all the interviewers in the interview panel are available.

Below is an implementation of the interview scheduling, in Algorithm 3.

Algorithm 3 Interview Scheduler

```
procedure schedule_interview(task_id, data)
    # Step 1: Unpack candidate details
    Extract inserted_id, jid, firstname, lastname, mobile, email, experience_year,
    candidate_skills from data
    # Step 2: Check if an interview is already scheduled
        Execute SQL query to count entries in `scheduled_interview` table
    where `cid = inserted_id`
    if count > 0:
        Print "Interview already scheduled for this candidate!"
    return
    # Step 4: Filter interviewers based on experience and skills match
    Initialize filtered_interviewers as an empty list
    for each interviewer:
        Ensure interviewer_experience and experience_year are integers
        if interviewer_experience >= 2 * experience_year:
            Parse interviewer_skills JSON by Algorithm 1
            # Using Algorithm 2 ML Model
            Calculate skill matching score using `
        mlscore(candidate_skills, interviewer_skills)`
            Append (id, firstname, lastname, score, availability) to
            filtered_interviewers
    # Step 5: Sort interviewers by skill matching score
    Select top 3 interviewers from the sorted list
    # Step 6: Find common availability
    Extract availability data for top interviewers
    # Find a common 1-hour time slot where all 3, 2, or at least 1 is available
    # Step 7: Calculate the interview date
    Map day to the corresponding weekday index
    Calculate `days_ahead` to reach the target day in the current week
    Calculate `interview_date` as `current_date + days_ahead`
    # Step 8: Prepare the interview panel and time slot
    Prepare the panel string with interviewer names
    # Step 9: Insert scheduled interview into the database
end procedure
```

IV. FUTURE SCOPE

Although smart interview scheduler was able to achieve its core objective, enhancement in these areas could potentially make the platform even more efficient and adapt to real-world scenarios –

- Integration with other platforms like LinkedIn, Glassdoor would make the platform even more popular and easy to manage as users can link and transfer their profiles from these platforms.
- Integrating with applications like – google calendar, outlook would also make it easy to use as these applications are widely used in the corporate industry for scheduling and managing tasks.
- Implementing a feedback system for both interviewers and candidates, would improve the overall user experience of the platform and help in enhancing features on the platform and also add new features.
- Shifting the database to a cloud-based storage, would make the platform highly scalable, as scaling the architecture to handle larger volumes of data, more users and more job postings would be crucial.
- Adding a recommendation engine to recommend jobs to users based on their location preference, experience, skills etc. would make the task of choosing a job easier for the candidate, which would also make it easier for the interviewers to get quality candidates for a job.
- Making the platform more mobile-friendly would make the system more accessible, allowing users to update their profiles and schedule interviews on the go.

V. CONCLUSION

Smart interview scheduler was successfully able to address the major challenges associated with traditional recruitment process, it was able to make the process more streamlined and efficient. The major impact has been in the time and cost reduction of the hiring process as the platform was successfully able to automate scheduling interviews and assigning interviewers to a candidate based on area of expertise and experience of the candidate, also the variety of functionalities that the platform provides is definitely helpful to organizations as it makes smart interview scheduler a single stop solution for them.

The implementation of the proposed solution is also good as it is scalable and rapid, plus the fact that it provides an easy to navigate user interface makes it simple to use and understand. The database structure is also reliable, and is able to handle real-world scenarios efficiently, with proper messages displayed to the user in case any error or unprecedented scenario occurs. The advancement of machine learning algorithms and large language models could potentially lead to a more accurate result in screening candidates and assigning interviewers.

REFERENCES

[1] Teodor Petrican, Ciprian Stan, Marcel Antal, Ioan Salomie, Tudor Cioara, Ionut Anghel. (2017). " Ontology-based skill matching algorithms". 2017 IEEE International Conference on Intelligent Computer Communication and Processing (ICCP). IEEE. https://doi.org/10.1109/ICCP.2017.8117005

[2] Bhushan Kinge, Shrinivas Mandhare, Pranali Chavan, S. M. Chaware.(2022). " Resume Screening using Machine Learning and NLP: A proposed system". International Journal of Scientific Research in Computer Science, Engineering and Information Technology,Vol. 8 No.2 . https://doi.org//10.32628/CSEIT228240

[3] Magdalena Velciu. (2018). " Matching skills and jobs: Experience of employees in Romania". New Trends and Issues Proceedings on Humanities and Social Sciences. https://doi.org/10.18844/prosoc.v4i8.3032

[4] Alessandro Ecclesie Agazzi. (2020). " Study of the usability of LinkedIn: a social media platform meant to connect employers and employees". arXiv. https://doi.org/10.48550/arXiv.2006.03931

[5] Chengguang Gan, Qinghao Zhang, Tatsunori Mori. " Application of LLM Agents in Recruitment: A Novel Framework for Resume Screening". arXiv. https://doi.org/10.48550/arXiv.2401.08315

[6] LinkedIn. (n.d.). Retrieved April 06, 2025, from https://www.linkedin.com/feed/

[7] Indeed. (n.d.). Retrieved April 06, 2025, from https://in.indeed.com/

[8] Glassdoor. Retrieved April 06, 2025, from https://www.glassdoor.co.in/Community/index.htm

[9] Monster. (n.d.). Retrieved April 06, 2025, from https://hiring.monster.com/

[10] ZipRecruiter. (n.d.). Retrieved April 06, 2025, from https://www.ziprecruiter.in/