International Journal of Advance Research in Science, Management, Engineering and Technology
Volume 1, Issue 4 | September - October 2026 | www.ijarsmet.com
ISSN: 3139-1141

Study on Automated Resume Information Extraction and Candidate Profiling

A Hybrid Framework for Structured Candidate Profile Generation from PDF and DOCX Resumes

Sonali Santosh Bhosale[#1], Dr. Dinesh Banswal[*2]
#Student (1) and Guide (2), ZIBACAR, Affiliated to Savitribai Phule Pune University (SPPU), Pune, Maharashtra, India
ORCID ID: 0009-0005-3633-3899 (Sonali Bhosale); 0000-0002-9114-0919 (Dr. Dinesh Banswal)
sonalibhosale156@gmail.com, dinesh.banswal@zealeducation.com

Abstract—

Recruiters and placement cells get hundreds of resumes that come in all sorts of formats and layouts, making it tedious to read through them, inconsistent and difficult to repeat. Automated resume parsers remove this problem, but most of them stop at extracting a handful of simple details like a name, an email address or a list of keywords, leaving the recruiter with a flat set of fields instead of a useful picture of the candidate.This study examines automated resume information extraction with an explicit focus on candidate profiling. It looks at how the name, email address, mobile number, skills, educational qualifications, work experience and projects can be extracted from uploaded PDF and DOCX resumes and then combined into one structured candidate profile. A hybrid framework is proposed. The text is extracted using the Apache PDFBox and Apache POI tools, normalized, and split into sections with a header dictionary and layout hints. The patterns, curated dictionaries, and a statistical named entity recognition model from Apache OpenNLP library are applied to fill each section's fields. The profile builder links each skill to the experience and project fields, calculates the total of experience by combining the ranges, and outputs a completeness score with a confidence level for each. The paper describes the design of each stage, the data model, the Java-based implementation plan with the Spring Boot and Hibernate frameworks, and an evaluation procedure with precision, recall, and F1 measures based on annotated resumes. The main point is that the creation of the profile view with interconnected fields of skills, projects, and experience allows to transform the text of the resume into screening and search data.

Keywords: resume parsing; information extraction; candidate profiling; named entity recognition; natural language processing; PDF and DOCX parsing; Apache OpenNLP; recruitment automation.

I. INTRODUCTION

A resume is the first document a recruiter reads about a candidate, and for campus placements and online job portals, it is the only document a recruiter reads before a shortlisting decision. Organisations advertising one vacancy get hundreds if not thousands of applications, most of which have to be scanned through before finding the handful of suitable candidates. It takes time to read through each resume, and the quality of judgment on the hundredth application is unlikely to be the same as that on the first.A system that reads resumes automatically and presents the important facts in a consistent form therefore has an obvious practical value.

The difficulty is that a resume is written by a person for a person. There is no mandatory schema, and two candidates with the same qualifications may present them with different headings, different date formats, different orderings and very different visual layouts. The file formats present a second layer of complexity. A Portable Document Format (PDF) file encodes glyphs at coordinates on a page, requiring the notion of a paragraph or a heading to be reconstituted from their positions. A Word document in the DOCX format maintains paragraphs, tables and text boxes, but many authors use tables and text boxes solely for visual design, scattering related text across independent containers. Any extraction method must contend with both scenarios before beginning to parse the content.

Information extraction refers to the discipline of transforming unstructured or semi structured text into structured records. When applied to resumes, it has been studied for approximately two decades using rule based systems, statistical sequence labellers and more recently, neural and transformer based models [1], [2], [3]. Much of this work and of the student and small enterprise systems that follow it focuses on the first step: the location of individual items such as the candidate name, an email address, a telephone number and a list of skill keywords. These items are necessary, but they describe a document more than a person.

A recruiter does not inquire whether a resume contains the word Spring Boot. The recruiter asks whether the candidate has actually constructed something with it, in which role, and for how long. A hiring manager does not ask for the list of degrees on the page; the question is what the highest completed qualification is and how many months of relevant work followed it. Answering such queries requires interrelation of facts: a skill must be associated with the experience or project in which it was used, dates must be combined to a total duration, and degrees must be ordered. When the extraction stage delivers only isolated fields, this reasoning is once more left to the human reader, and much of the benefit of automation is lost. This limitation is referred to in this paper as limited candidate profiling.

The present study therefore asks how the information extracted from PDF and DOCX resumes may be assembled into a structured candidate profile which incorporates skills, education, experience and projects, and the extent to which each component of such a pipeline may be expected to yield accurate outputs. The focus of the research is the extraction of the candidate name, email address, mobile number, skills, educational qualifications, work experience and projects from uploaded resumes.The research gap addressed is the limited

ISSN: 3139-1141

depth of candidate profiling in systems that stop at basic information.

The main contributions of this paper are the following.

- A hybrid extraction framework that combines layout aware text recovery, dictionary based section segmentation, pattern rules, curated skill and degree dictionaries and a statistical named entity recogniser, designed so that each stage can be tested on its own.
- A candidate profile model in which skills are tied to the work experience and project entries which provide evidence for them, along with a skill evidence score, a union-based measure of the total experience and profile completeness index.
- A confidence and validation layer which provides a score and reason for every extracted field, allowing for fields with insufficient evidence to be passed to human reviewers instead of being blindly applied.An implementation plan on the Java platform using Apache PDFBox, Apache POI, Apache OpenNLP, Spring Boot and Hibernate, and an evaluation protocol that measures field level and profile level quality.

The rest of the paper is organised as follows. Section II reviews related work. Section III presents the research gap, problem, objectives and scope. Section IV explains the methodology. The proposed framework is described in detail in section V. Section VI discusses the implementation design, and section VII presents the evaluation protocol. The expected benefits and the ethical considerations are outlined in section VIII. The limitations and future work are listed in Section IX and it concludes.

II. Background and Related Work

A. Structure of Resumes and Nature of the Problem

Although they differ considerably, most resumes contain a recognisable set of parts: a header with contact details, an optional career objective or summary, education, work experience, projects, skills, certifications and achievements. The parts are signalled by headings, yet the headings are not standardised. Skills may appear as Technical Skills, Key Skills, Core Competencies or Tools and Technologies, and work history may be labelled Experience, Employment History or Professional Background. A parser that relies on a single fixed list of headings will fail on a visible fraction of real documents.

Inside each part the text is rarely written in full sentences. Bullet fragments, comma separated lists and tabular layouts are common, which weakens the assumptions behind general purpose language tools that were trained on prose. Dates appear as ranges in many notations, including month and year, numeric month and year, year only and open ended forms such as Present. Resumes prepared in India add further conventions: secondary and higher secondary results are reported as percentages, university results as percentage or cumulative grade point average, and personal details such as date of birth, marital status or parents names are sometimes listed. Some of these details are irrelevant to a hiring decision and some are sensitive, so a responsible extractor must know what to ignore as well as what to capture.

B. Rule Based and Dictionary Based Approaches

The earliest practical resume parsers relied on hand written patterns and word lists. Items with a regular surface form, such as email addresses and telephone numbers, are well suited to regular expressions, and closed vocabularies such as degree names, programming languages or institution keywords are easy to hold in gazetteers. Work such as that of Kopparapu [2] shows how language processing steps and lists can be combined to turn unstructured resumes into information that supports search. The strengths of this family are transparency, speed and no requirement of any training data. The weaknesses are brittleness when a layout falls outside the anticipated forms and steady maintenance burden as vocabularies change, for example, when a new framework becomes popular.

C. Statistical Sequence Labelling

A second line of work treats extraction as labelling a sequence of tokens. Hidden Markov models and conditional random fields (CRF) assign a label to each token while taking neighbouring labels into account [4], [5]. Yu, Guan and Zhou [1] proposed a cascaded hybrid model in which a first pass divides the resume into blocks and a second pass extracts detailed items from each block using different learners for different block types. Singh and co-workers [3] described PROSPECT, a screening system that applied conditional random fields to extract resume entities and then ranked candidates against requirements. These studies established two ideas that remain relevant: resumes should be segmented before details are extracted, and the extraction method may differ by section.

Resume entity recognition is a specialised case of named entity recognition (NER), a task surveyed by Nadeau and Sekine [6] and studied in depth on newswire corpora such as the CoNLL 2003 shared task data [7]. Tag schemes that mark the beginning and the inside of an entity, introduced for text chunking [8], are now standard. Ratinov and Roth [9] showed that design choices such as the tagging scheme and the use of external gazetteers have a large effect on recognition quality, a finding that supports combining dictionaries with learned models for a domain such as resumes.

D. Neural and Transformer Based Models

Neural sequence taggers removed much of the manual feature design of earlier statistical models. The bidirectional recurrent network with a CRF output layer described by Lample et al. [10] became a common baseline, and the survey by Li et al. [11] traces the later progress. Attention based architectures [12] and pretrained encoders such as BERT [13] pushed results further. Resume specific studies followed this path. The researchers Ayishathahira et al. [14] used a combination of neural networks and conditional random fields while working on resume parsing, Chen et al. [15] suggested a two step algorithm which first tackles text blocks and then pulls out information from them, Bhatia et al. [16] applied BERT for end to end parsing and for finding candidates for a job description and last, Barducci et al. [17] constructed an end to end framework for Italian language resumes.

Each of these models have their real world costs though.They need annotated resumes in sufficient numbers, which are difficult to obtain because resumes contain personal data. They require more computing resources for both training and sometimes for inference, and their behavior on an atypical

ISSN: 3139-1141

set of inputs can be less explainable than that of a rule-based system. For a college-level or small-organization deployment, the value of a design that uses learned models only where they are needed and where their behavior can be inspected seems indisputable.

E. Matching and Screening Systems

A separate body of research uses already structured resume data for downstream tasks. Convolutional networks have been applied to classify resumes into occupational categories [18], and person job fit models learn joint representations of resumes and job postings [19], [20]. Siamese networks have been used to match resumes to jobs [21], and a systematic review by Sinha et al. [22] summarizes screening methods based on language processing and machine learning. All of these methods rely on the quality and richness of the structured input data. A matching model that sees only a bag of skill words cannot tell recent hands on use from a passing mention, which is one practical reason to invest in profile construction before matching.

F. Text Recovery and Document Layout

Before anything can be done involving language processing, the text of a resume has to be recovered in some usable order, and this turns out to be less trivial than it might seem. The PDF page is a set of drawing instructions; characters are placed at coordinates, and the file records neither lines nor paragraphs nor columns, so the reader program has to group glyphs into words and words into lines by looking at positions and font sizes. When a page has two columns, or a sidebar which holds the contact details and the skills, a purely top to bottom reading will mix text from unrelated parts of the page. Many resume templates are designed precisely this way, with a narrow coloured column on one side, which makes layout handling a practical necessity and not an academic detail.

A DOCX file presents a different set of problems. Its paragraphs are explicit, but often the visual structure is built using nested tables, frames and text boxes, and the order in which these objects are stored is not necessarily the same as the order in which they are read. Also, headers and footers are stored separately, and hyperlinks hold their targets apart from the visible text. Each format therefore requires their own reader, and the readers have to deliver the same normalised representation that all later stages are independent of the original file type. This requirement was a big influence on the interface between the reader and the normalisation modules in the proposed framework.

G. Toolkits Relevant to the Java Platform

Three Open Source Apache projects cover the main extraction needs on the Java.platform. Apache PDF. The box [23] allows reading PDF files and exposes a text stripper that can order text by position. Apache POI [24] allows reading Microsoft Office formats and, through its XWPF component, provides access to the paragraphs, tables, headers and footers of DOCX files. Apache OpenNLP [25] offers sentence detection, tokenisation and a trainable maximum entropy based name finder. The Stanford CoreNLP toolkit [26] is a widely used alternative for language analysis, and the Tesseract engine [27] provides optical character recognition for scanned documents. The framework of this paper employs their first three, treating optical character recognition as a progression.

H. Evaluation Practice and Data Availability

Comparing resume extraction studies with each other proves to be a more formidable task than it might seem at first glance. Since resumes contain personal information, annotated collections of considerable size are rarely released and many studies report results on private or modestly sized sets. The metrics also differ. Some work counts a prediction correct at the level of single tokens, as is common in sequence labelling, while other work requires the whole entity to match exactly, which is the convention of the CoNLL shared tasks [7]. A system can be strong under lenient convention and weak under the strict one. A third difference is the unit of aggregation: a score pooled over all items is dominated by long resumes, whereas a score averaged over resumes treats each candidate equally.

For these reasons the present study fixes its evaluation protocol in advance (Section VII), reports strict and partial matching side by side, separates results by file type and layout, and publishes the annotation guidelines together with the results. The goal is not simply to assess the framework, but to make the assessment reproducible for others.

I. Expertise Classifications and Normalization

Skills have a vocabulary problem that is partly independent of the extraction technique. The same skill is expressed in several ways, such as JS, Javascript, and ECMAScript, and different skills share a common prefix, such as Java and JavaScript. Skills are also hierarchical: Spring Boot belongs to the Spring family, which belongs to the Java ecosystem, so a candidate who lists Spring Boot has given indirect evidence of Java. A flat list of strings cannot express this, and a recruiter who searches for Java would miss a profile that mentions only the framework. A taxonomy that includes aliases and parent relations removes the friction, at the expense of editorial effort. Consequently, the framework treats the taxonomy as a first class, versioned asset of the system.

J. Summary of the Literature

Table I summarises the approaches discussed above in terms of what they chiefly deliver. The literature is rich on how to recognise individual entities and on how to match resumes to jobs. It is thinner on the step in between, namely how recognised entities are assembled, validated and enriched into a candidate profile that a person can query. That intermediate step is the subject of this study.

TABLE I. COMPARISON OF REPRESENTATIVE APPROACHES

Approach	Representative study	Main output
Cascaded hybrid model	Yu et al. [1]	Blocks, then detailed fields
Language processing with lists	Kopparapu [2]	Information for search
CRF extraction and ranking	Singh et al. [3]	Entities, ranked candidates
Neural network with CRF	Ayishathahira et al. [14]	Entity labels
Two step block and field	Chen et al. [15]	Blocks, then fields
BERT based parsing	Bhatia et al. [16]	Entities, job matches
End to end Italian pipeline	Barducci et al. [17]	Entities from resumes
Proposed framework	This study	Linked, scored candidate profile

ISSN: 3139-1141

K. Positioning of the Present Study

The present study does not claim a new recognition algorithm. Its position in the literature is between two bodies of work: the studies that recognise entities in resumes [1], [14], [17], and the studies that use structured resume data for matching [19], [20]. The first group finishes where the second group begins, and the connecting step that gives the data its structure, meaning and quality signals gets less attention than either end. The study occupies this connecting step and treats it as a design problem: what a candidate profile should contain, how it should be built from imperfect extraction results, and how its usefulness should be measured.

III. Research Gap, Problem Statement and Objectives

A. Research Gap: Limited Candidate Profiling

The review shows that many resume extraction systems concentrate on basic information. The gap that is explored in this study is that such systems rarely create a structured candidate profile that ties together skills, education, experience and projects into a single coherent record. The gap can take three related forms.

- Flat output. Fields are returned as independent lists. A skill list does not indicate where each skill was used, and an education list is not ordered or summarised, leaving the recruiter to interpret the document.

- Missing evidence links. A skill that only occurs in a keyword block is treated the same as a skill that occurs in two projects and a job. Without evidence links, keyword stuffing is not distinguishable from demonstrated use.

- Missing derived attributes and quality signals. Total experience, highest qualification, recency of skills and a measure of how complete or trustworthy the extraction is typically not computed, even though these are what screening queries actually use.

B. Problem Statement

Given a text based resume in PDF or DOCX format, design and evaluate a method that extracts the candidate name, email address, mobile number, skills, educational qualifications, work experience and projects, and organises them into a validated, queryable candidate profile in which skills are tied to the experience and projects that evidence them.

C. Formal Problem Formulation

The task can be stated more precisely. After text recovery and normalisation, a resume is a sequence of lines R = (l1, …, ln). A segmentation assigns to every line a label from a finite set of sections Λ that includes a header label and an ignored label. For every section label a field extractor maps the lines carrying that label to a set of typed entries, for example education entries from the lines labelled Education. The outcome of all extractors is an unlinked set of entities U. The profile construction step is then a function that takes U and returns a structured profile Π in which entries are ordered, skills are linked to entries, derived attributes are computed and every value carries a confidence and a source range. Seen in this way, many existing systems solve the problem only up to the set *U*, or return a profile in which the connections are absent. The contribution of this study lies in the last function and in the evaluation of its output, while the earlier functions are built from known techniques and are chosen for transparency. The formulation also indicates where errors are propagated; a wrong segmentation corrupts all extractors that depend on a section whereas a wrong entry in one extractor affects only the connections of that entry. This explains why segmentation receives a separate evaluation in the plan.

D. Research Objectives

- To study existing approaches to resume parsing and identify where candidate profiling is weak.
- To design a text recovery and normalisation procedure that works for both PDF and DOCX resumes.
- To design extractors for the seven target fields using patterns, dictionaries and a statistical recogniser.
- To define a candidate profile model with evidence links, derived attributes, confidence values and a completeness index.
- To define an evaluation protocol that measures both field level extraction and profile level usefulness.

E. Research Questions

RQ1: What mix of rules, dictionaries and statistical recognition works best for each target field? RQ2: How reliably can segment boundaries be recovered from PDF and DOCX text, and how much does layout matter? RQ3: Is a profile built by tying skills to experience and project better than a flat list of skills? RQ4: How can the system alert the reviewer when it is unsure, so they focus on the important parts?

F. Scope and Assumptions

The study covers resumes in English with PDFs that have a text layer or DOCX files that have standard paragraphs and tables. It does not cover scanned image resumes, legacy binary Word files, or resumes in other languages, though the design leaves room. The framework builds the profile but does not rank candidates against a job description; ranking is seen as a later layer that uses the profile. Attributes that could lead to discrimination, such as gender, age, religion, caste, marital status and photographs, are intentionally excluded from the profile.

G. Requirements

The objectives were turned into a short set of functional and non functional requirements, listed in Table II, which guided the design and give a checklist for the demonstration of the artefact. Each requirement has a verification method, so that it is clear how it will be checked.

TABLE II. Requirements and Their Verification

ID	Requirement	Verification
R1	Accept PDF and DOCX uploads and reject other types	Upload tests
R2	Extract name, email and mobile number	Field level scores
R3	Extract skills with canonical names	Set match score
R4	Extract education, experience and projects as entries	Aligned entry score
R5	Link skills to supporting entries	Reviewer judgement
R6	Compute experience without double counting	Error in months
R7	Attach confidence and source range to values	Inspection, calibration

ISSN: 3139-1141

ID	Requirement	Verification
R8	Exclude sensitive personal attributes	Schema review, tests
R9	Process a resume within seconds	Timing runs
R10	Allow review and correction by a person	Usability walkthrough

R1 to R8 are functional; R9 and R10 are non functional.

IV. Research Methodology

A. Research Design

The study follows the design science research approach [28], in which a purposeful artefact is built to address an identified problem and is then evaluated. The process model of Peffers et al. [29] was used as the outline: problem identification, definition of objectives, design and development, demonstration, evaluation and communication. The problem and objectives are stated in Section III. The artefact is the extraction and profiling framework described in Sections V and VI, and the evaluation is specified in Section VII.

B. Selection of Tools

The tools were selected based on five main criteria including an open source licence that allows using them in an educational institution, active maintenance, possibility to operate on an ordinary server without graphics processor, stable programming interface, and simple connection to the web application layer. Apache PDFBox, Apache POI and Apache OpenNLP meet all the criteria since they are distributed under the Apache License 2.0. Python libraries such as PyPDF2, pdfplumber and python-docx offer comparable text recovery and were considered, particularly for experimentation. The final choice was the Java platform because the web tier, persistence layer and extraction logic can then be developed in one language and deployed as one service with Spring Boot [30].

C. Data Collection and Preparation

Resumes contain personal data so the evaluation set is planned around consented and anonymisedmaterial. Volunteer resumes from students/and professionals will be collected with written consent, any direct identifiers replaced with realistic dummy values prior to annotation and their original files stored in only an access controlled location for duration of the study. To cover layouts that are not provided by volunteers, additional resumes will be created from public template styles using invented content. The collection will be balanced across file type (PDF and DOCX), and seniority (freshers and experienced candidates) and layout (single column, two column and table based).

D. Annotation

Each resume is annotated by two people against written guidelines that define every field, how to treat ambiguous cases and which personal details must be ignored. Agreement is measured with Cohen's kappa [31] on a pilot subset, disagreements are resolved by discussion, and the guidelines are revised until agreement is acceptable. The resulting gold records are stored as structured files that follow the profile schema of Section V.

E. Sample Size and Pilot Study

A small pilot sample of about twenty resumes is annotated initially to evaluate the system. The goal is to assess the annotation guidelines, identify fields for which annotators differ systematically, and uncover unexpected variations in layout. For the main study, sufficient resumes are chosen within each stratum to yield stable estimates of the per-field scores, and confidence intervals as described in Section VII make the uncertainty explicit rather than hiding it in a single number.

F. Consent and Ethical Handling of Resumes

Due to the nature of the documents being processed, the volunteering basis of participation, the indication of how the resumes will be used and stored, and the ability to withdraw at any moment, resulting in the deletion of the file and annotations, the study appears to have addressed confidentiality issues adequately. The subjects' personal identifying information is removed before annotations are made, and the annotators only work with the confidential version of the documents. The study does not attempt to re-identify the subjects based on the information in their resumes or match them with other resources.

G. Iterative Development

Development proceeds in cycles, each of which consists of a period of iteration on the training data. During each iteration, a particular set of updates to the dictionaries or pattern matching rules is made, and the performance on the held-out test set, which is only evaluated once per cycle, is measured. The errors in this test are then classified into categories (e.g., incorrectly ordered words, unrecognized heading, ambiguously marked tokens) and the category that accounts for the largest proportion of errors is addressed first in the next development cycle. The process is continued until the system performance is satisfactory on the test set and the error analysis can be found in Sect. VII.

V. Proposed Framework

A. Architecture Overview

The framework consists of a pipeline of seven stages, depicted in Fig. 1. An uploaded file is validated, converted to text by a format specific reader, normalised, divided into sections, passed to field extractors, assembled into a candidate profile and finally stored and exposed through a service interface. Each stage takes a well defined input and produces a well defined output, enabling every stage to be unit tested with small fixtures and to be replaced independently. For instance, the statistical recogniser can be swapped for a transformer model without touching the segmentation or the profile

ISSN: 3139-1141

builder.

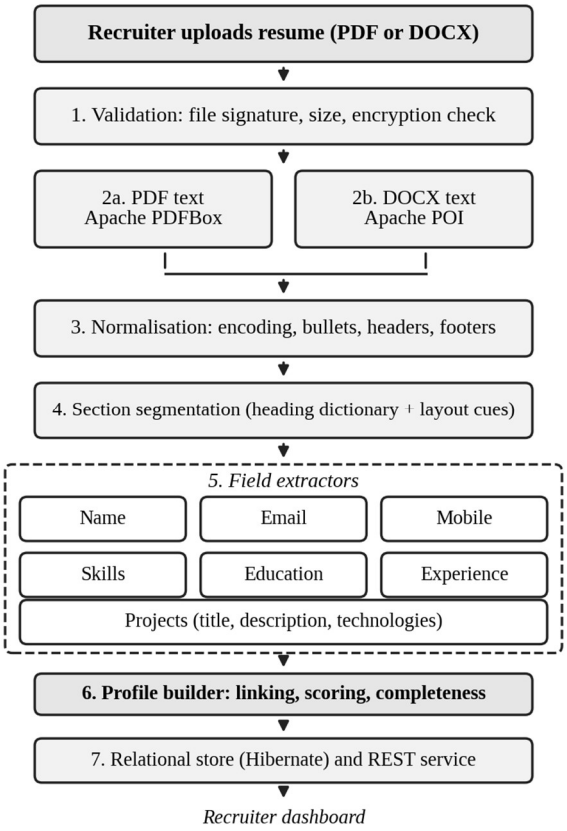


Fig. 1. Architecture of the proposed resume information extraction and candidate profiling pipeline.

A second design decision is that the value of each stage retains a reference to the source text. Each extracted value includes the line numbers and character range of its source. This provenance enables the dashboard to highlight the source of a value, to support analysis of error, and underpins the evidence links in Section V-N.

B. Ingestion and Validation

The service handles a multipart upload and performs checks before attempting to parse it. The file extension is compared to the real content type by examining the first bytes. A PDF document begins with the signature %PDF-, and a DOCX is, in reality, a ZIP archive, which begins with the bytes PK and contains word/document.xml entry. Files that claim to be one thing but contain another, encrypted PDF files, and too big files are rejected with an explanatory message. Decompression for DOCX files is limited to avoid archive bombs; this is achieved by secure ZIP settings in Apache POI. The SHA-256 checksum of the file is calculated to recognize the same file re-uploaded and link it to the existing record instead of creating the duplicate candidate.

C. Text Extraction from PDF and DOCX

For PDF files the text stripper of Apache PDFBox [23] is used with sorting by position enabled. Sorting is a good idea because PDF content streams are not required to list text in reading order and some exporters write a right hand column before a left hand one. Even with sorting enabled two column layouts can interleave lines from both columns. To reduce this the extractor also obtains text positions and applies a simple column detection: if the horizontal distribution of line starts shows two stable clusters separated by a gap then they are regrouped column by column and the left column is read first. Lines that repeat at the same vertical position on every page (typically headers, footers and page numbers) are removed.

For DOCX files Apache POI [24] is used via the XWPF classes. The body elements of the document are visited in their stored order so that paragraphs and tables stay in sequence. Table cells are read row by row, and the cells of a row are joined with a separator so that a two cell row containing a label and a value remains recognisable. Header and footer parts are read separately because some authors place their name and contact details there. Text inside drawing objects and text boxes is not exposed by the paragraph interface, so the underlying XML is scanned for text box content and, when present, that text is added in document order. Hyperlink targets are retained, since an email address or a profile link is sometimes visible only as link text.

If a PDF yields almost no text, as happens with scanned images, the system reports that no text layer was found rather than returning an empty profile. Optical character recognition [27] is a possible extension and is listed among the directions for future work.

D. Normalisation

Raw text is cleaned before interpretation. Unicode compatibility normalisation (form NFKC) [32] converts ligatures, full width characters and special dashes to their ordinary counterparts, which prevents a pattern from failing because a hyphen in a date range is actually an en dash or a minus sign. Non breaking spaces and zero width characters are removed. Decorative bullet glyphs from symbol fonts are mapped to a single bullet marker, words broken by a line end hyphen are rejoined when the joined form is a known word, and runs of blank lines are collapsed. The normaliser keeps a mapping from cleaned positions to original positions so that provenance remains exact.

E. Section Segmentation

Segmentation partitions the line stream into labelled blocks that follow the observation, made in earlier cascaded models [1], [15], that field extraction is easier when the extractor knows which part of the resume it is reading. A line is a heading candidate if it is short, has no sentence punctuation and is not part of a bullet list. The candidate is compared with a heading dictionary that maps many surface forms to a canonical section label (Table III). Matching is first exact after case and punctuation folding, then approximate using an edit distance [33] of one for short headings and two for longer ones, tolerating typing slips and the loss of a character during extraction. Because a heading word may also occur inside ordinary text, for example the word experience inside a sentence, a candidate is accepted only when a layout cue supports it: capital letters, a bold run in a DOCX file, larger font size in a PDF, or an empty line before and after.

Fig. 2 shows the decision flow.

TABLE III. SECTION HEADING DICTIONARY (EXCERPT)

Canonical section	Example heading variants
Education	Education; Academic Details; Qualifications; Academic Background
Experience	Work Experience; Employment History; Professional Experience; Internships

ISSN: 3139-1141

Canonical section	Example heading variants
Skills	Skills; Technical Skills; Key Skills; Core Competencies; Tools and Technologies
Projects	Projects; Academic Projects; Personal Projects; Project Work
Certifications	Certifications; Courses; Training; Licences
Summary	Summary; Career Objective; Profile; About Me
Ignored	Declaration; Personal Details; Hobbies; References

Sections labelled Ignored are located so that their text is excluded from extraction.

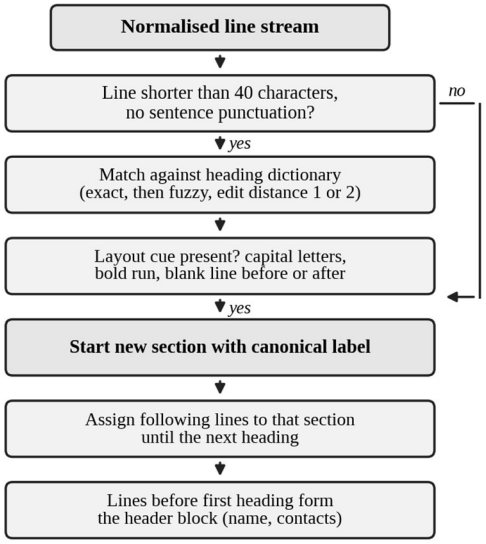


Fig. 2. Decision flow used to detect section headings and assign lines to sections.

Lines that occur before the first accepted heading form the header block, where the name and contact details are normally found. If a resume has no recognisable headings at all, the segmenter falls back to a content based classification of paragraphs using the same dictionaries, and marks the whole result with low confidence. Segments that remain unlabelled are kept rather than discarded, because a skill or project description may legitimately sit under an unusual heading.

Algorithm 1 summarises the segmentation procedure. It is deliberately simple: one pass over the lines, with all knowledge held in the dictionary and in the layout test, so that its behaviour on a failing resume can be understood by reading the dictionary.

Algorithm 1. Section segmentation

```
Input: lines L[1..n], heading dictionary H
Output: mapping S from section label to lines
current <- HEADER; S[HEADER] <- empty list
for i <- 1 to n do
    if isCandidate(L[i]) then
        label <- match(L[i], H)   // exact, then edit distance
        if label != none and layoutCue(L[i]) then
            current <- label; S[label] <- empty list
            continue
    append L[i] to S[current]
return S
```

F. Extraction of Name, Email and Mobile Number

Name. The candidate name is the field where simple rules fail most visibly, because a name is not marked by any pattern. The extractor generates candidates from the header block and scores them. A candidate is a line, or part of a line, of two to four tokens that begin with capital letters or are entirely in capitals, contain no digits and no symbols, and are not dictionary words such as Resume, Curriculum Vitae or a section heading. Each candidate is scored by

$$N(c) = w_1P(c) + w_2F(c) + w_3R(c) + w_4E(c) - d(c) \quad (1)$$

where P rewards a position near the top of the document, F rewards a prominent format such as a larger font or capital letters, R is the probability returned by the OpenNLP person model [25] for the candidate tokens, E measures string similarity between the candidate and the local part of the email address (the portion before the at sign), and d is a penalty for words typical of job titles, organisations and places. The weights w_1 to w_4 sum to one and are initially set to 0.35, 0.15, 0.30 and 0.20, then tuned on the development set. The highest scoring candidate is selected. When no candidate reaches a minimum score, the name is reconstructed from the email address when possible and flagged for review.

Email address. Email addresses adhere to a standardised structure defined within the Internet Message Format [34] of which a subset is adequate for resumes. Once normalisation is complete the pattern shown in Table IV is applied to the whole text. Obfuscated notations such as name [at] domain [dot] com are rewritten before being matched. In the case of multiple addresses being found, the first occurrence in the header block is prefered and addresses to well known free mail providers are not penalised due to their prevalence amongst candidates. Syntactic validity is checked but not deliverability, as no message is sent.

Mobile number. The extractor targets ten digit Indian mobile numbers which begin with 6, 7, 8 or 9, with an optional country prefix and any mixture of spaces, hyphens, brackets or dots as separators. The digits are collected, the country prefix removed and the value validated. The accepted number is stored in the international format of ITU-T Recommendation E.164 [35] as +91 followed by ten digits. Numbers which appear adjacent to words such as fax, landline or reference or are part of a longer digit string such as an identity number are ignored.

TABLE IV. PATTERNS FOR CONTACT FIELDS

Field	Pattern (simplified)
Email	`[A-Za-z0-9._%+-]+@[A-Za-z0-9.-]+\.[A-Za-z]{2,}`
Mobile	`(?:\+?91[\s-]?)?[6-9]\d{4}[\s-]?\d{5}`
Year	`(?:19|20)\d{2}`
Percentage	`\d{2}(?:\.\d{1,2})?\s?%`
Grade point	`\d{1,2}\.\d{1,2}\s?(?:CGPA|GPA|/10)`

Patterns are shown in simplified form; digit boundaries and context checks are added in the implementation.

A fabricated example illustrates how equation (1) behaves. Suppose the header of the resume includes the lines Asha Verma and Computer Applications. For the first line, P = 1.0 as it is the first line, F = 0.9 for a larger font, R = 0.8 from the person model and E = 0.7 as it resembles the email prefix asha.verma with no penalty so N = 0.35 × 1.0 + 0.15 × 0.9 + 0.30 × 0.8 + 0.20 × 0.7 = 0.865. For the second line, P = 0.6, F = 0.5, R = 0.1 and E = 0, and the penalty for typical heading words is 0.3, giving 0.015. The first line is chosen with a wide margin. The margin itself is useful; if the best two candidates are within a small distance of each other, the name is accepted

ISSN: 3139-1141

but the confidence is reduced and the alternative is displayed to the reviewer.

G. Skill Extraction

Skills constitute a large, open and evolving vocabulary so that the extractor is built on a curated taxonomy rather than a fixed list. Each entry in the taxonomy has a canonical name, a category such as programming language, framework, database, tool, cloud service or soft skill, plus a set of aliases. For example, JS and Javascript map to JavaScript and Spring-Boot maps to Spring Boot. Matching proceeds with a multi pattern string search using Aho–Corasick algorithm [36] on lower cased token sequences, which finds every taxonomy entry in a single pass independent of dictionary size.

A few practical issues need explicit handling. Tokenisation must preserve names such as C++, C#, .NET and Node.js that ordinary tokenisers split. Short or common names are ambiguous; the letter R, the word Go and the single letter C could appear in unrelated contexts such as R and D or option C. These names are only allowed when inside the skills section, inside a technology line of a project, or next to a context word such as language or programming. Names that include other names, for instance Java and JavaScript are resolved by preferring the longest match at a position.

The taxonomy is supplemented by the statistical recogniser. An OpenNLP name finder model could be trained on annotated resume sentences which label skill mentions not present in the dictionary, such as a newly released library. Mentions that the model suggests but the dictionary does not contain are stored in a review queue and once a reviewer has confirmed them they are added to the taxonomy. In this way, human corrections improve the dictionary over time and the system behaves as a learning loop, an idea consistent with the use of gazetteers and learned models together [9].

H. Dictionary Construction and Maintenance

Dictionaries determine much of the recall of the framework, so their construction is described as a process. The first version of the skill taxonomy is assembled from public documentation of programming languages, frameworks, databases and tools, and from the vocabulary of technology job advertisements, and is then reviewed by the authors for duplicates and ambiguous names. Every entry receives a canonical name, a category, aliases and, where appropriate, a parent entry. Rules for aliases are written down: case differences, spacing and hyphen variants are handled by normalisation, and only true synonyms, such as a product and its former name, are listed.

Maintenance follows the review queue described earlier. Candidate terms proposed by the statistical recogniser, together with terms that reviewers add by hand, are listed with the number of resumes in which they occur. A term is added to the taxonomy when it has been verified by a reviewer, and each change is documented in a version history with dates and reasons. The heading and degree dictionaries follow the same procedure. Versioning allows an evaluation run to be repeated with the exact dictionaries that were used, which matters for reproducibility.

I. Education Extraction

Education entries are located in the education section, and, when no such section exists, by searching the whole text for degree expressions. A degree dictionary groups qualifications into ordered levels (Table V), covering the usual forms found in Indian resumes together with international equivalents. Dotted and undotted spellings, such as B.E. and BE, are folded together before matching.

TABLE V. QUALIFICATION LEVELS USED FOR ORDERING

Level	Examples of recognised forms
1 Secondary	SSC; Class X; Matriculation; Secondary
2 Higher secondary	HSC; Class XII; Intermediate; Higher Secondary
3 Diploma	Diploma; Polytechnic; Advanced Diploma
4 Bachelor	B.E.; B.Tech; BCA; B.Sc; B.Com; BBA; Bachelor of ...
5 Master	M.E.; M.Tech; MCA; M.Sc; MBA; Master of ...
6 Doctorate	Ph.D.; Doctorate; D.Sc

Level numbers are used to derive the highest qualification.

Each degree expression acts as an anchor for one entry. The lines within a small window around the anchor are searched for an institution, recognised by words such as University, College, Institute, School, Academy, Board or Polytechnic, for a specialisation introduced by words such as in or with, for a year or year range using the pattern in Table IV, and for a result expressed as a percentage or a grade point. When the result is a grade point value, its scale is taken from the text if it is stated and is otherwise inferred from the value, for example a value above 4.5 cannot be on a four point scale. Entries are finally sorted by level and end year, from which the highest qualification and the most recent qualification are derived.

J. Work Experience Extraction

In the experience section, entries are anchored by date ranges. A date parser (Table VI) accepts month names in full or abbreviated form, numeric month and year pairs, and bare years, joined by hyphens, en dashes, the word to, or similar connectors, and open ended endings such as Present, Current, Till date and Ongoing, which are resolved to the date of processing. Each parsed range becomes the anchor of an entry. The job title is found on the same line, or adjacent lines, with a dictionary of role words such as Developer, Engineer, Analyst, Intern, Trainee, Consultant, Administrator, Lead and Manager. The employer is located as the neighbouring line or line fragment that carries an organisation marker such as Limited, Technologies, Solutions, Systems or Inc., or that follows a separator such as at, a comma or a vertical bar. The bullet lines that follow, until the next anchor, are stored as responsibilities and achievements.

Overlapping or duplicated ranges are common when a candidate lists a promotion within the same employer or a freelance engagement alongside a job. Adding up durations naively would count some months twice. The total experience is therefore calculated on the union of the ranges. If the ranges after sorting and merging are $[s_1, e_1]$, …, $[s_k, e_k]$, the total in months is

$$T = \Sigma_{j=1}^{k} \text{months}(e_j - s_j) \quad (2)$$

In this context, the function months uses calendar months, inclusive of both ends, so that months ranging from June to August would be three. Internships identified by the role dictionary are reported separately so that a recruiter can decide if they should be counted or not. The validation rules reject ranges where the end precedes the start or lies in the future (except open endings) and raise a warning when the first employment began before the plausible age of completing

ISSN: 3139-1141

secondary education; the check compares the experience start with the earliest education year.

TABLE VI. DATE NOTATIONS ACCEPTED BY THE PARSER

Notation	Example
Month name and year	Jan 2023 – Mar 2024; January 2023 to March 2024
Numeric month and year	01/2023 – 03/2024; 2023-01 to 2024-03
Year only	2021 – 2023; 2021 to 2023
Open ended	Jun 2024 – Present; 2022 – Till date
Duration phrase	Three months; 2 years 4 months
Season and year	Summer 2024; Fall 2023

A year without a month is read as spanning January to December unless the context shows otherwise.

K. Project Extraction

Projects are particularly valuable to freshers who have limited work experience. A project entry usually has a title line that can be identified by a short line at the beginning of a bullet or a numbered item or followed by a field label such as Technologies Used, Tech Stack, Tools, Role, Duration, Team Size or Description. The text following that until the next anchor line is the project body. From the body the extractor collects a short description, the role if stated, the duration if stated, and the technologies. Technologies are obtained by applying the skill taxonomy to the label line and to the description, with higher confidence for the label line because it is an explicit statement. A project with no title is created only when the body is long enough to be a coherent description, so that stray bullets are not turned into projects.

L. Freshers and Experienced Candidates

A framework used for campus placement and for general hiring sees two very different kinds of resume. For an experienced candidate, the experience section carries most of the evidence and the analysis of dates, titles and employers dominates. For a fresher, the experience section may be empty or may list only short internships and the information is concentrated in education, projects, certifications and the skills section. The framework handles both without a switch. When no experience entries are found, total experience is set to zero, the profile is marked as a fresher profile in a derived attribute and not as an error, and the evidence score of a skill relies on project entries alone. In the completeness index the experience part is excluded from the weights for a fresher profile, so that a student is not penalised for the absence of something that was never expected. This design decision reflects the importance of projects for early career candidates, noted in the description of project extraction.

M. Design Rationale

Each major design choice was made against an alternative. Dictionary based segmentation was preferred to a trained block classifier because headings are a closed and easily extended vocabulary and that a failure can be repaired by adding a line to a file rather than needing new training data. Regular expressions were preferred for email addresses and mobile numbers because their forms are regular and a pattern is both exact and inspectable. A statistical recogniser was kept for names where no pattern exists and skills outside the taxonomy where a fixed list would age. Linking skills to entries by textual co-occurrence within an entry was preferred to a semantic similarity model because it is transparent; a reviewer can see the sentence that created the link. Finally, a weighted sum was chosen for the evidence score because its parameters are directly interpretable and can be adjusted by a recruiter who values work experience differently.

N. Candidate Profile Construction

The profile builder is the process that closes the research gap. It takes the independent results of the extractors and builds the model shown in Figure 3. This model includes an individual with the data of education, experience, and projects, and sets of skills linked to these items as entities. The critical feature is that a skill is also an entity connected to the items above.

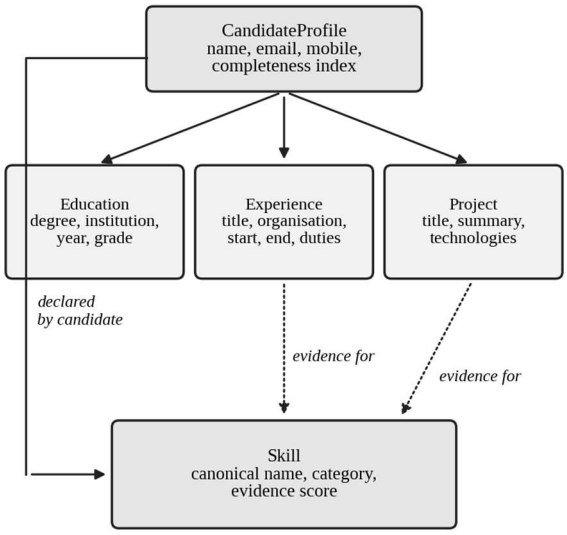


Fig. 3. Candidate profile model with skills linked to the experience and project entries that evidence them.

For each canonical skill k the builder counts the experience items Xk and the project items Pk whose text mention it and records whether it is declared in the skills section, Dk, equal to 1 if declared and 0 otherwise. The skill evidence score is

$$S(k) = \alpha D_k + \beta \min(1, X_k/\tau) + \gamma \min(1, P_k/\tau) \quad (3)$$

with $\alpha + \beta + \gamma = 1$ and a saturation constant τ. The initial values $\alpha = 0.20$, $\beta = 0.45$, $\gamma = 0.35$ and $\tau = 2$ express the view that use at work is stronger evidence than use in a project and both are stronger than a bare declaration; they are design parameters for which the evaluation will test, not claims of optimality. Fig. 4 illustrates the idea. A skill that is only declared receives a low score while one that is declared and also appears in a job and a project approaches the maximum. The scores allow the dashboard to order skills by evidence and to flag skills that are claimed but not supported.

ISSN: 3139-1141

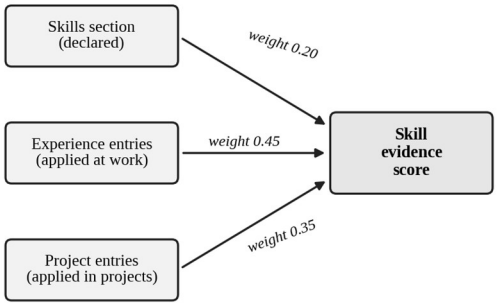


Fig. 4. Contributions of the three evidence sources to the skill evidence score.

Each extracted value has a confidence between 0 and 1. When a value is supported by several independent sources, for example an email address found both in the header block and in a hyperlink, the confidences are combined by

$$c = 1 - \Pi_i (1 - c_i) \quad (4)$$

where c_i is the confidence of source i. Confidence is also lowered by validation warnings such as an inconsistent date range. Fields below a configurable threshold are shown with a warning marker in the dashboard. Finally, a completeness index summarises the profile as

C = Σf wf qf (5)

where f ranges over the profile parts, qf in the range 0 to 1 reflects how fully part f was filled (for example the fraction of education entries that have an institution, a year and a result), and the weights wf sum to one. The index is intended for triage; it describes how much was found, and does not say anything about the quality of the candidate.

In addition to the scores, the builder derives the attributes that screening queries use: the highest qualification, the total experience T of equation (2), the most recent role, the categories in which the candidate has evidenced skills, and the age of the most recent evidence for each skill. A short textual summary is generated from a fixed template that inserts these values, for example the qualification, the experience in years and months, and the three highest scoring skills. A template is used instead of a text generation model so that every statement in the summary can be traced to extracted data.

Algorithm 2. Candidate profile construction

```
Input: header fields, entries E, X, P, declared skills D,
taxonomy K
Output: validated profile with scores
1  for each entry e in X and P do
2      e.skills <- findSkills(K, text(e))
3  for each skill k found anywhere do
4      compute D_k, X_k, P_k and S(k)
5  T <- unionMonths(ranges of X)
6  level <- highestLevel(E)
7  warnings <- validate(E, X, P, T)
8  lower confidence of fields named in warnings
9  C <- completeness(profile)
10 return profile
```

O. Confidence Calibration

A confidence value is useful only if it means something. Initial confidences are assigned by the type of evidence: a value matched by a strict pattern in the header block starts high, a value found only by fuzzy matching or only by the statistical model starts lower, and a value that fails a validation rule is reduced further. These starting values are then calibrated on the development set. Values are binned by confidence and then compared to the observed fraction of true values. If the confidence is too high for the bin, the mapping is adjusted. The threshold for review is selected from the same distribution as a trade-off between recall and the increased effort of reviewing more items. The choice of threshold, and thus the extra effort, is reported alongside the results.

P. Handling Noisy Input and Edge Cases

Real resumes have many cases not anticipated in the simplest design, and some thought is required to handle each of them correctly. The following cases are supported as described.

- Hard wrapped lines. PDF text is line wrapped at the page margin, and lines not beginning with a bullet or capital letter directly following a line ending with punctuation are joined together before processing.

- Multiple numbers or emails. A resume may list one number or address as secondary, perhaps with a note or a different format. The first email or number in the header block is used as the primary, and the rest are retained as alternates.

- Surname first formats. A name is not always in the format given in the document, and the possibility of surname-first with initial capitals or trailing initials is allowed for. The stored format always matches what is in the document to avoid inventing names.

- Table formats. When a table places a heading in one cell and the content in another on the same row, the content is associated with the heading as if they were not in a table.

- Projects within experience. A project mentioned within a job is associated both with the employer and as a project to avoid losing the context when the author does not mention it separately.

- Graphical ratings. Ratings depicted as bars or stars with no text have no text to indicate a value, and no level is recorded, only the skill name. In all these cases, the default is to retain the evidence while reducing confidence in favor of the default.

Q. Illustrative Example

Table VII gives a constructed example, written for this paper, to show how an excerpt of a resume is turned into profile entries. It is an illustration of the design, not a measured result. The evidence scores in the last row are computed from equation (3) with the initial parameters. Java is declared, appears in the internship and appears in the project, so D = 1, X = 1 and P = 1, which gives 0.20 + 0.45 × 0.5 + 0.35 × 0.5 = 0.60. Angular and Git are declared only and receive 0.20, whereas Spring Boot and MySQL are declared and used in the project and received 0.375.

TABLE VII. CONSTRUCTED EXAMPLE OF EXTRACTION AND PROFILING

Source text (excerpt)	Extracted value
ASHA VERMA, asha.verma@example.com, 90000 00000	Name Asha Verma; email; mobile +919000000000
MCA, Example Institute of Computer Applications, 2024 – 2026, CGPA 8.4	Level 5 Master; institution; years; grade 8.4 of 10

ISSN: 3139-1141

Source text (excerpt)	Extracted value
Java Developer Intern, Sample Technologies Pvt. Ltd., Jun 2025 – Aug 2025	Role; employer; 3 months; internship flag
Student Records Portal | Java, Spring Boot, MySQL	Project title; three technologies linked
Skills: Java, Spring Boot, MySQL, Angular, Git	Five declared skills
Derived	Total experience 3 months (internship); evidence: Java 0.60, Spring Boot 0.375, MySQL 0.375, Angular 0.20, Git 0.20

All names, numbers and organisations in this table are invented for illustration.

Listing 1 shows the same example as it would be returned by the candidate endpoint. Each value carries a confidence and the position of its source text; only part of the record is shown.

Listing 1. Profile returned by the service (abridged, constructed)

```
{
  "name": {"value": "Asha Verma", "conf": 0.94},
  "email": {"value": "asha.verma@example.com",
            "conf": 0.99, "src": [1, 1]},
  "mobile": {"value": "+919000000000", "conf": 0.97},
  "highestLevel": "Master",
  "totalExperienceMonths": 3,
  "skills": [
    {"name": "Java", "declared": true,
     "expCount": 1, "projCount": 1, "score": 0.60},
    {"name": "Angular", "declared": true,
     "expCount": 0, "projCount": 0, "score": 0.20}
  ],
  "projects": [{"title": "Student Records Portal",
     "skills": ["Java", "Spring Boot", "MySQL"]}],
  "completeness": 0.88,
  "warnings": []
}
```

The flat output of a basic system for the same resume would be five strings in a skill list. The profile adds the information that Java is the only skill supported by both work and a project, that the experience consists of a short internship, and that Angular and Git are claimed without support. A recruiter looking for demonstrated Java use can act on this directly.

R. *Complexity Analysis*

The cost of the pipeline is close to linear in the size of the document. Let n be the number of characters after normalisation, m the total length of the taxonomy patterns and z the number of skill matches. Text extraction and normalisation take time proportional to n. The multi pattern search of the Aho–Corasick algorithm [36] runs in $O(n + m + z)$ once its automaton has been built, and the automaton is built once at start up. Heading matching compares only the few lines that pass the candidate test with the heading dictionary, and each comparison involves short strings, so its cost is small compared with text extraction. The date parser and the entry anchors are single passes. The OpenNLP name finder processes tokens in order with a fixed beam width, so its cost is also proportional to the number of tokens. Profile construction depends on the number of entries and skills, both of which are small. In practice the dominant costs are reading the file and running the statistical model, which is why the pipeline is expected to meet the time requirement R9 without special hardware.

Summarising Section V, the framework is a chain of small, separately testable steps in which each step keeps the source ranges of its output. The decisions that matter most are the recovery of reading order, the use of section labels to steer the extractors, the preference for leaving a field empty over guessing, and the construction of links between skills and entries. Section VII describes how these decisions will be tested.

VI. IMPLEMENTATION DESIGN

A. *Technology Stack and Modules*

The system is planned as a Spring Boot [30] application with a layered structure. Table VIII presents the modules, their responsibilities and the main library used by each. The absence of web and database code from the extraction modules makes them standard Java classes that can be tested using simple unit tests and reused in a batch tool or a command line utility.

TABLE VIII. MODULES OF THE IMPLEMENTATION

Module	Responsibility	Library
ingest	Upload, type check, checksum, limits	Spring Web
reader	PDF and DOCX text with positions	PDFBox, POI
normalise	Unicode, bullets, headers, mapping	Java standard library
segment	Heading detection, section labels	Own code, edit distance
extract	Seven field extractors	OpenNLP, regular expressions
profile	Linking, scores, validation	Own code
store	Entities and repositories	Hibernate, relational database
api and ui	REST service and dashboard	Spring Web, web front end

The normalisation and segmentation modules use only the Java standard library and a small edit distance routine. The extraction module was designed such that it wraps OpenNLP [25] behind an interface so that one could supply the person model, a custom skill model, or a transformer model in the future via configuration. Dictionaries (headings, degrees, rolewords, skills, and aliases) are versioned in text files outside the code repository, making them easily editable by a placement officer or an administrator.

B. *Data Model*

The profile is stored in a relational database through the object relational mapping layer of Hibernate [37]. The main tables are candidate, resume_file, education, experience, project, skill, and candidate_skill. The candidate table stores the name, email, phone, highest qualification, total experience in months, completeness index, and review status. The resume_file table stores the original file name, type, checksum, and upload datetime, and has a foreign key to the candidate, allowing multiple versions of a resume to be stored. The skill table stores skills in a canonical format and their categories, and the candidate_skill table is a many-to-many join table that stores the evidence score, declared flag, counts X and P, and the date of the last evidence of the skill-candidate relation. The education, experience, and project tables store the source line ranges of their respective texts. The email and phone fields are unique, allowing candidate entries to be disambiguated if needed, and a deleted flag allows soft deletion of a candidate upon request.

Table IX contains the main columns of the tables described above. It can be seen that only the attributes necessary for screening are present, while gender, age, religion, caste, marital status, or photo are not present. This can be considered a reflection of the system's design as well, as the exclusion of these attributes may be a structural choice rather than just a policy decision.

ISSN: 3139-1141

TABLE IX. MAIN TABLES AND KEY COLUMNS

Table	Key columns
candidate	candidate id, name, email, mobile, highest_level, total_months, completeness, review_status
resume_file	resume_file id, candidate_id, file_name, type, sha256, uploaded_at
education	id, candidate_id, degree, level, institution, years, grade, src_range
experience	id, candidate_id, title, employer, start, end, is_internship, src_range
project	id, candidate_id, title, summary, src_range
skill	id, canonical_name, category, parent_id
candidate_skill	candidate_skill candidate_id, skill_id, declared, exp_count, proj_count, score, last_used

C. Service Interface

The service provides a few endpoints of representational state transfer (REST). A multipurpose internet mail extension (MIME)-encoded request to the resume endpoint will upload files and return candidate ids that were created or matched with warnings. The candidate endpoint will return the full profile in JavaScript Object Notation (JSON) with confidence indicators for each field and the ranges. The search endpoint takes a list of filters as input. For instance, a query might request candidates with a specific skill of a given evidence level, a certain amount of total experience, a qualification level, and a completeness threshold. Finally, the last endpoint describes the value of the current service. For instance, a question about candidates with a Master level of education and at least one project with Spring Boot would require a complex natural language processing query against a relational database if the data were stored in a conventional way. At the same time, this query is a simple join over the profile tables.

D. Dashboard and Review Workflow

The recruiter dashboard displays the list of candidates with their summary, highest qualification, experience and top skills. Opening a candidate view shows the extracted fields alongside the original text with the range of the source highlighted. Fields with low confidence carry a marker. A reviewer can correct a value, and every correction is stored with the original value. Corrections have two uses: they update the profile at once, and they accumulate as labelled examples and dictionary suggestions that are used in the next improvement cycle, so that the review effort is reinvested in the system.

E. Configuration and Extensibility

Recreate this content from scratch in simple English language. Make easy to read. MUST Randomly vary writing parameters for EACH sentence. Replace “and” / “plus” with other connectors. Avoid overly formal style. Do not use speculative focus, impersonal tone. Cut all buzzwords, clichés, and overused combinations. Do not use abbreviations or contractions. No hype, No fluff, no sales, no promotional tone, no marketing language, no clichés, no calls to action, no “let’s”, no 'flip side', no tables. No dry technical lists. Use unexpected sentence starts and end. Also don't use direct commands in writing. Very important - Keep the content the same length & Do not expand.

Input:

Behaviour that is likely to change is kept in configuration and in dictionary files instead of in code. This covers the file size limit, the minimum confidence for automatic acceptance, the weights of equations (1), (3) and (5), the paths of the OpenNLP models and the dictionaries. A developer who wishes to add a new extractor, for example for certifications or languages known, implements a single interface that receives a section and returns entries with confidences, registers it in the configuration and extends the profile schema. Because each extractor depends only on the section text and the dictionaries, new extractors can be developed and tested without any change to the rest of the pipeline.

F. Security and Privacy Controls

Several controls protect the personal data held by the system. Authentication is required for all endpoints, and roles separate reviewers, who may correct data, from viewers, who may only read it. Uploaded files are stored outside the web root with generated names, and are scanned for size and type before parsing. All access to a profile is logged. Original files and database rows are covered by a retention period after which they are removed automatically, and a candidate record can be deleted on request, together with the stored file. Statistical models run inside the deployment, so resumes are not sent to an external service for processing, which also simplifies compliance with the consent and purpose limitation obligations discussed in Section VIII.

G. Deployment Scenarios

Two deployment settings are envisaged. In the first case, the service is hosted on a server within the college or company domain, using a single machine to store the database, reducing overhead and ensuring the privacy of student data. In the second scenario, the service is offered as a container behind a web server in a cloud platform for multiple organisations, where the data for each tenant is partitioned virtually, while the application itself does not store any information apart from the database. It can be scaled horizontally by launching more instances, and the dictionaries and models are read-only mounted volumes, shared between all the instances to ensure consistency.

H. Non Functional Requirements

A single resume is processed in memory and only requires a few seconds of CPU time even with the statistical model enabled, which allows serving thousands of users on a single server. The bulk uploads, such as for a placement drive, are put on an async queue to avoid overloading the system. All the endpoints are protected by role-based authentication, and the communication is encrypted via transport encryption. The uploaded files and the profile rows are stored on disk with encryption and have a retention period set by the company policy.

I. Testing Strategy For The Development

Before the evaluation, the implementation will go through three stages of testing. The unit tests cover all the extractors, having used fixtures with both valid and invalid data, such as a ten-digit number that does not represent a phone mask or a word at the beginning of a sentence that does not have a particular meaning. The integration tests simulate the upload of files to the server and check if the resulting profile matches the expected one. Finally, a regression test suite is created from the annotated development resumes to make sure that the new changes do not break any existing layouts, and it is run before every release. The tests use either completely fake or anonymized resumes only.

ISSN: 3139-1141

VII. Evaluation Methodology

This section describes how the framework will be evaluated. The protocol is fixed before the final measurement to prevent the results from being shaped by what is observed and the numbers obtained by it are reported in full, including the cases where the framework fails.

A. Test Collection

The test collection consists of annotated resumes prepared as described in Section IV divided into a development set for tuning and a test set for the final measure. The test set is stratified by file type, seniority and layout, and each stratum is reported separately to avoid the weaknesses being hidden by an average. When a custom statistical model is trained the annotated sentences are split into training, validation and test parts at the resume level, and never at the sentence level, to prevent one candidate's text from crossing parts.

B. Annotation Schema

The gold records follow the profile schema. Each entity bears its type, its normalized value and the character range in the extracted text. Experience and project items contain the list of skills that an annotator identifies in that item as actually used, which constitutes the evidence basis for provenance links. Annotators are instructed to refrain from identifying any potentially sensitive personal data and to avoid labeling such information. A field that is absent from the resume is annotated as absent, so that the system is also credited for correctly returning nothing.

C. Matching Rules

A prediction is compared with the gold annotation under a rule suited to the field, as summarised in Table X. Email and mobile number require an exact match after normalisation. Names are compared after case folding, with a partial credit rule for a missing middle name. List valued fields such as skills are compared as sets after canonicalisation. Structured entries such as education, experience and projects are first aligned to gold entries and then compared attribute by attribute, so that a correct degree with a wrong year is counted as a partial success rather than a complete failure.

TABLE X. Evaluation Plan by Field

Field	Match rule	Expected difficulty
Name	Case folded exact; partial credit	Medium
Email	Exact after normalisation	Low
Mobile	Exact in E.164 form	Low
Skills	Set match on canonical names	Medium
Education	Aligned entries, per attribute	Medium
Experience	Aligned entries, dates, title	High
Projects	Aligned entries, title, tools	High

Difficulty is an expectation from the nature of each field, to be confirmed by measurement.

D. Metrics

Extraction quality is reported as precision, recall and the F1 measure [38]. With *TP*, *FP* and *FN* denoting true positives, false positives and false negatives,

$$P = TP / (TP + FP) \quad (6)$$

$$R = TP / (TP + FN) \quad (7)$$

$$F_1 = 2PR / (P + R) \quad (8)$$

Values are computed per field and aggregated in two ways: micro averaging over all items and macro averaging over resumes, because a few resumes with very long skill lists could otherwise dominate the micro average. For list valued fields with partial agreement, the overlap between a predicted set A and a gold set B is also reported as

$$J(A,B)=|A\cap B|/|A\cup B| \quad (9)$$

The accuracy of total experience is measured by the mean absolute error in months between the computed value and the annotated value over n resumes,

$$MAE=(1/n)\sum i=1n\ |Ti-T\hat{}i| \quad (10)$$

E. Baselines and Ablation

To evaluate whether each design decision made a difference, the four configurations are compared on the same test set. The first configuration uses regular expressions and dictionaries on the unsegmented text. The second configuration also uses regular expressions and dictionaries but adds section segmentation. The third configuration adds the OpenNLP recogniser and the validation rules, extracting flat fields typical of many simple systems. Finally, the fourth configuration implements the full framework, including profile construction. The comparison between the first three configurations addresses RQ1 and RQ2, while the comparison between the third and the fourth configurations addresses RQ3.

F. Hypotheses

Four hypotheses arise from the design and will be tested through the protocol. H1: Adding section segmentation improves precision on the skills, education, and experience, over the unsegmented baseline, because the extractors no longer see text that belongs to other sections. H2: Adding the statistical recogniser and the validation rule improves the name field and the alignment of structured entries over rules alone. **H3**: profile construction leaves field level scores largely unchanged, since it restructures rather than extracts, but improves the profile level measures, namely the correctness of evidence links and the agreement of skill order with reviewers. **H4**: errors are concentrated among values with low confidence. A result that contradicts a hypothesis will be reported and analysed, since a negative result is informative about the design.

The evaluation is expected to produce two kinds of outcome. If the hypotheses hold, the results will demonstrate that the early stages account for most of the field level quality, and that the profile construction is indeed valuable at the profile level while not adversely affecting the field scores. If they do not hold, the error analysis will shed light on which assumption failed, e.g., that headings are less regular than expected, or that co-occurrence links contain many false positives. Both outcomes advance our understanding of the problem, and will thus be reported.

G. Profile Level Measures

Field level scores do not show whether the profile is useful, so three profile level measures are added. The first is the fraction of resumes whose complete profile matches the gold profile on all mandatory fields. The second is the correctness of evidence links: for a sample of skills, two reviewers judge whether the entries to which the system links the skill really

mention it in a way that shows use. The third component is the agreement between the skill order based on evidence scores and the one determined by the human reviewers who examined the resume. The value of the review support is determined by measuring the ratio of incorrectly identified values with low confidence as compared to those incorrectly identified with high confidence, arguing that the former present a greater danger than the latter.

H. Reporting Format

Results will be reported in tables with one row per field and one column per stratum, giving precision, recall and F1 with confidence intervals, followed by the profile level measures, the timing figures and the distribution of error causes. Failures will be illustrated with anonymised excerpts. Reporting by stratum was performed to highlight areas in which the framework was deficient, which may be more informative to a potential end user than a single aggregate metric.

I. Statistical Treatment and Error Analysis

Differences between configurations were tested using paired bootstrap resampling over resumes, providing approximate confidence intervals with no assumption about the distribution of scores. Processing time was reported as the median and 95th percentile per resume, for each file type. Every error on the development and test sets is assigned to one cause category, as shown in Table XI, and the distribution of causes is reported with the results.

TABLE XI. ERROR CATEGORIES FOR ANALYSIS

Category	Example	Typical remedy
Text order	Columns interleaved	Column detection
Heading missed	Unusual section title	Dictionary entry
Ambiguous token	Go or R as a skill	Context rule
Date form	Unseen range notation	Parser rule
Boundary	Two projects merged	Anchor rule
Model error	Person tag on a place	More training data

VIII. DISCUSSION

A. How the Framework Addresses the Gap

The three forms of the gap identified in Section III correspond to three design elements. The flat output problem is addressed by the profile model, in which the candidate owns education, experience and project entries instead of a bag of strings. The missing evidence problem is addressed by the links between skills and entries and by the evidence score of equation (3). The missing derived attribute problem is addressed by the union based total experience, the ordered qualification levels, the confidence values and the completeness index. None of these elements needs a larger language model; they need careful modelling of what a recruiter actually wants to know, and that is the main argument of this study.

B. Comparison with Alternative Designs

The hybrid design is one of several possibilities, and it is important to note what it gains and what it gives up in comparison with the others. Table XII gives a qualitative comparison. The ratings are judgements based on the properties of each family of methods discussed in Section II, not measurements.

TABLE XII. QUALITATIVE COMPARISON OF DESIGN ALTERNATIVES

Criterion	Rules only	Neural model	Hybrid
Training data needed	None	Large	Small
Explainability	High	Low	High
Compute cost	Low	High	Low
Layout robustness	Low	Medium	Medium
Maintenance effort	High	Medium	Medium
Profile linking	Possible	Possible	Built in

Ratings are qualitative and depend on implementation quality.

A further alternative is to pass the resume text to a large language model and request it to return a structured record. Such models handle varied wording well and require no dictionaries, and are a reasonable option for a prototype. For a production system handling personal data, several concerns remain. Output can vary for the same resume between runs, which complicates testing. A model may return a plausible, but not in the document value, which is harmful when a recruiter trusts the record. Sending resumes to an external service raises privacy concerns, and running a large model locally requires substantial hardware. For these reasons the framework relies on deterministic, inspectable components and retains its source ranges, such that every value can be checked. The recogniser interface leaves the possibility of using a language model as one more component, with its output validated against the same rules and source ranges.

C. Expected Benefits

For recruiters and placement officers, a structured and searchable profile reduces the first pass reading effort and makes screening criteria explicit and repeatable. Since every value carries its source, the recruiter can verify a surprising result in seconds. For candidates a consistent reading of the document reduces the risk that a good resume is overlooked due to being formatted unusually, provided that the parser is tested on varied layouts. For researchers and developers, the modular design and the published protocol make it possible to replace one component, for instance the recogniser, and measure the effect precisely.

D. Application Scenarios

Three scenarios show how the profile would be used. In a campus placement drive, a placement officer must shortlist students for a company that requires a minimum qualification level, a minimum grade and evidence of a particular technology. With flat extraction the officer filters on keywords and still reads each resume. With profiles the officer filters on level, grade and a skill evidence threshold, and only reads the resumes that pass. In a small company without a dedicated human resources team, a manager who posts a single role can get a ranked view of applicants by evidence score and total experience, with the source text a click away to confirm each value.

A third scenario is feedback to candidates. The completeness index and the list of skills declared but not evidenced can be shown to a student before submission, suggesting for example that a claimed skill be supported by a project description. Aggregated over a batch of students, the same data shows which skills are widely claimed but rarely evidenced, which helps a department decide where practical training is needed. These uses concern the quality of the resume and the preparation of the candidate and do not involve ranking people, which keeps them free of most fairness concerns.

ISSN: 3139-1141

E. Practical Lessons for Developers

Several practical lessons emerge from the design. Firstly, effort spent on text recovery repays itself many times, and every later error caused by wrong line order looks like an error of the extractor. Secondly, dictionaries should be treated as data with an owner and version history, not as constants in code. Thirdly, a confidence value and a source range costs little to carry and make both debugging and review much faster. Fourthly, tests should include negative cases, because false positives such as nine digit numbers read as phone numbers damage trust more than a missed value. Lastly, the profile model should be fixed early, because it defines what every extractor must deliver and what the evaluation must measure.

F. Relevance to the Indian Hiring Context

Many features of the framework respond to conventions of resumes in India. Results in secondary and higher secondary education are reported as percentages, university results as percentages or grade point averages on different scales, and degrees carry abbreviations such as MCA, BCA and B.E. that international systems may not recognise. Mobile numbers have a characteristic ten digit form with an optional country code. Campus recruitment is common, hence a large share of applicants are freshers whose resumes are built around education and projects. A parser designed around Western work history formats would handle these resumes poorly, which is a reason for building and evaluating a parser with local conventions in mind, while keeping the architecture general enough for other settings.

G. Threats to Validity

Internal validity is threatened by tuning on the development set; the held out test set and the fixed protocol limit this. External validity is limited by the composition of the collection: a set dominated in one region, one industry or one resume style may not represent other settings, so the strata and the sources must be stated with the results. Construct validity depends on the annotation guidelines, since a decision such as whether an internship counts as experience changes the gold values. This is why the guidelines are written before annotation and why agreement is measured. Finally, the evidence score weights are design choices; the study treats them as parameters to be examined and not as established values.

H. Ethical, Privacy and Legal Considerations

Resume parsing touches personal data and employment decisions, and both call for care. In India, the Digital Personal Data Protection Act, 2023 [39] sets out obligations of consent, purpose limitation and data minimisation for the processing of digital personal data. A deployment should therefore show a clear notice at upload, collect only the fields needed for screening, keep data for a limited period and provide a way to correct or delete a profile.

Fairness is a separate concern. Automated screening can reproduce historical bias, as surveys of bias in machine learning explain [40] and as a widely reported recruiting tool that was withdrawn after it penalised resumes associated with women illustrates [41]. The present framework limits this risk by design: it does not extract or store gender, age, religion, caste, marital status or photographs, it does not rank candidates, and it keeps a human reviewer in the loop for uncertain values. These measures reduce, but do not remove, the risk. Any later ranking layer must be audited for disparate outcomes across groups before it is used for real decisions.

I. Reproducibility and Release Plan

To support reproducibility, the study plans to publish the annotation guidelines, the evaluation scripts, the dictionary versions and the configuration used for each reported run, together with the anonymised and synthetic resumes for which sharing is permitted by the consent given. Real resumes that cannot be shared will be described by their strata and counts. The source code of the extraction modules is organised such that each module can be run on its own against the same inputs, and the exact library versions are recorded, since changes in a text extraction library can alter the line order and therefore the results of later stages.

IX. Limitations and Future Work

The framework has several limitations. It handles only text based PDF and DOCX files in English. It relies on dictionaries whose coverage dictates recall for skills, degrees and headings, and which need regular maintenance. Rules for names and dates are tuned for the conventions that dominate the development resumes and may need adaptation for other regions. Highly decorative resumes in which the text is placed freely on the page, or which are composed mostly of images and icons, will be extracted with lower confidence. Finally, the evidence score gives equal weight to every mention within an entry, so a long project description that repeats a skill many times is not distinguished from one that names it once.

Several extensions follow naturally. Optical character recognition [27] would admit scanned resumes. Transformer based entity recognition [13] could replace or complement the maximum entropy model where enough annotated data becomes available, with the interface already isolating the recogniser. Skill normalisation could be linked to an external occupational taxonomy so that related skills are recognised as such. A ranking layer that compares a profile with a job description, built on the matching research reviewed earlier [19], [20] and audited for fairness, is the next step after profile quality has been established. Support for other languages and for multilingual resumes would widen the applicability of the system.

X. Conclusion

Aim

This paper describes an investigation into automated extraction of the information contained in resumes saved in PDF or DOCX formats for candidate profiling. Beginning with the observation that many such extractors provide little beyond unconnected fields of information, this paper proposes a hybrid pipeline that extracts text, splits the resume into sections, extracts the name, email address, mobile phone, skills, education, experience and projects, and then constructs a profile containing the skills connected to the relevant information, the total experience indicated, and the confidence level in each. The design of the system, the data model, the Java implementation pipeline and evaluation procedure are all described in sufficient detail as to allow replication and testing.

Significance

The main finding is a proposition that the value of resume extraction is governed not by the quantity of fields extracted,

but their connectedness. The described extraction pipeline, once evaluated, will be able to facilitate screening, search and, with the required auditing, matching of resumes against a profile. A human auditor will always be in control of the final disposition. The implication for the wider field of recruitment technology is that the most valuable parsers do not extract the most fields, but rather provide enough information that allows the human recruiter to ask the most pertinent questions. Building that trust requires structure, evidence and honest signals of uncertainty, and these can be obtained with modest, transparent technology.

ACKNOWLEDGMENT

The authors thank the Department of Master of Computer Applications of the institute for providing the facilities for this work, and thank the volunteers who agreed to share their resumes in anonymised form for the preparation of the test collection.

REFERENCES

[1] K. Yu, G. Guan, and M. Zhou, "Resume information extraction with cascaded hybrid model," in *Proc. 43rd Annu. Meeting Assoc. Comput. Linguistics (ACL)*, Ann Arbor, MI, USA, 2005, pp. 499–506.

[2] S. K. Kopparapu, "Automatic extraction of usable information from unstructured resumes to aid search," in *Proc. IEEE Int. Conf. Progress in Informatics and Computing (PIC)*, Shanghai, China, 2010.

[3] A. Singh, C. Rose, K. Visweswariah, V. Chenthamarakshan, and N. Kambhatla, "PROSPECT: A system for screening candidates for recruitment," in *Proc. 19th ACM Int. Conf. Information and Knowledge Management (CIKM)*, Toronto, Canada, 2010.

[4] J. Lafferty, A. McCallum, and F. Pereira, "Conditional random fields: Probabilistic models for segmenting and labeling sequence data," in *Proc. 18th Int. Conf. Machine Learning (ICML)*, 2001, pp. 282–289.

[5] C. Sutton and A. McCallum, "An introduction to conditional random fields," *Foundations and Trends in Machine Learning*, vol. 4, no. 4, pp. 267–373, 2012.

[6] D. Nadeau and S. Sekine, "A survey of named entity recognition and classification," *Lingvisticae Investigationes*, vol. 30, no. 1, pp. 3–26, 2007.

[7] E. F. Tjong Kim Sang and F. De Meulder, "Introduction to the CoNLL-2003 shared task: Language-independent named entity recognition," in *Proc. 7th Conf. Natural Language Learning (CoNLL)*, 2003, pp. 142–147.

[8] L. A. Ramshaw and M. P. Marcus, "Text chunking using transformation-based learning," in *Proc. 3rd Workshop on Very Large Corpora*, 1995, pp. 82–94.

[9] L. Ratinov and D. Roth, "Design challenges and misconceptions in named entity recognition," in *Proc. 13th Conf. Computational Natural Language Learning (CoNLL)*, 2009, pp. 147–155.

[10] G. Lample, M. Ballesteros, S. Subramanian, K. Kawakami, and C. Dyer, "Neural architectures for named entity recognition," in *Proc. NAACL-HLT*, 2016, pp. 260–270.

[11] J. Li, A. Sun, J. Han, and C. Li, "A survey on deep learning for named entity recognition," *IEEE Trans. Knowledge and Data Engineering*, vol. 34, no. 1, pp. 50–70, Jan. 2022.

[12] A. Vaswani *et al.*, "Attention is all you need," in *Advances in Neural Information Processing Systems 30*, 2017, pp. 5998–6008.

[13] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of deep bidirectional transformers for language understanding," in *Proc. NAACL-HLT*, 2019, pp. 4171–4186.

[14] C. H. Ayishathahira, C. Sreejith, and C. Raseek, "Combination of neural networks and conditional random fields for efficient resume parsing," in *Proc. Int. CET Conf. Control, Communication, and Computing (IC4)*, Thiruvananthapuram, India, 2018.

[15] J. Chen, C. Zhang, and Z. Niu, "A two-step resume information extraction algorithm," *Mathematical Problems in Engineering*, vol. 2018, 2018.

[16] V. Bhatia, P. Rawat, A. Kumar, and R. R. Shah, "End-to-end resume parsing and finding candidates for a job description using BERT," arXiv:1910.03089, 2019.

[17] A. Barducci *et al.*, "An end-to-end framework for information extraction from Italian resumes," *Expert Systems with Applications*, vol. 210, 2022.

[18] L. Sayfullina, E. Malmi, Y. Liao, and A. Jung, "Domain adaptation for resume classification using convolutional neural networks," arXiv:1707.05576, 2017.

[19] C. Zhu *et al.*, "Person-job fit: Adapting the right talent for the right job with joint representation learning," *ACM Trans. Management Information Systems*, vol. 9, no. 3, Art. 12, 2018.

[20] C. Qin *et al.*, "Enhancing person-job fit for talent recruitment: An ability-aware neural network approach," in *Proc. 41st Int. ACM SIGIR Conf. Research and Development in Information Retrieval*, 2018, pp. 25–34.

[21] S. Maheshwary and H. Misra, "Matching resumes to jobs via deep Siamese network," in *Companion Proc. The Web Conf. (WWW)*, 2018, pp. 87–88.

[22] A. K. Sinha, M. A. K. Akhtar, and A. Kumar, "Resume screening using natural language processing and machine learning: A systematic review," in *Machine Learning and Information Processing*, Singapore: Springer, 2021.

[23] Apache Software Foundation, "Apache PDFBox: A Java PDF library." [Online]. Available: https://pdfbox.apache.org

[24] Apache Software Foundation, "Apache POI: Java API for Microsoft Documents." [Online]. Available: https://poi.apache.org

[25] Apache Software Foundation, "Apache OpenNLP: A machine learning based toolkit for the processing of natural language text." [Online]. Available: https://opennlp.apache.org

[26] C. D. Manning *et al.*, "The Stanford CoreNLP natural language processing toolkit," in *Proc. 52nd Annu. Meeting Assoc. Comput. Linguistics: System Demonstrations*, 2014, pp. 55–60.

[27] R. Smith, "An overview of the Tesseract OCR engine," in *Proc. 9th Int. Conf. Document Analysis and Recognition (ICDAR)*, 2007, pp. 629–633.

[28] A. R. Hevner, S. T. March, J. Park, and S. Ram, "Design science in information systems research," *MIS Quarterly*, vol. 28, no. 1, pp. 75–105, 2004.

[29] K. Peffers, T. Tuunanen, M. A. Rothenberger, and S. Chatterjee, "A design science research methodology for information systems research," *J. Management Information Systems*, vol. 24, no. 3, pp. 45–77, 2007.

[30] Spring, "Spring Boot." [Online]. Available: https://spring.io/projects/spring-boot

[31] J. Cohen, "A coefficient of agreement for nominal scales," *Educational and Psychological Measurement*, vol. 20, no. 1, pp. 37–46, 1960.

[32] Unicode Consortium, "Unicode normalization forms," Unicode Standard Annex #15. [Online]. Available: https://unicode.org/reports/tr15/

[33] V. I. Levenshtein, "Binary codes capable of correcting deletions, insertions, and reversals," *Soviet Physics Doklady*, vol. 10, no. 8, pp. 707–710, 1966.

[34] P. Resnick, Ed., "Internet message format," IETF, RFC 5322, Oct. 2008.

[35] International Telecommunication Union, "The international public telecommunication numbering plan," ITU-T Recommendation E.164, 2010.

[36] A. V. Aho and M. J. Corasick, "Efficient string matching: An aid to bibliographic search," *Commun. ACM*, vol. 18, no. 6, pp. 333–340, 1975.

[37] Hibernate, "Hibernate ORM." [Online]. Available: https://hibernate.org/orm/

[38] D. M. W. Powers, "Evaluation: From precision, recall and F-measure to ROC, informedness, markedness and correlation," *J. Machine Learning Technology*, vol. 2, no. 1, pp. 37–63, 2011.

[39] Government of India, "The Digital Personal Data Protection Act, 2023," Act No. 22 of 2023.

[40] N. Mehrabi, F. Morstatter, N. Saxena, K. Lerman, and A. Galstyan, "A survey on bias and fairness in machine learning," *ACM Computing Surveys*, vol. 54, no. 6, Art. 115, 2021.

[41] J. Dastin, "Amazon scraps secret AI recruiting tool that showed bias against women," *Reuters*, Oct. 10, 2018.