# Layout-Aware Parsing Meets Efficient LLMs: A Unified, Scalable Framework for Resume Information Extraction and Evaluation


Fanwei Zhu*
zhufw@hzcu.edu.cn
Hangzhou City University
Hangzhou, China

Jinke Yu†
yujinke.yjk@alibaba-inc.com
Alibaba Group
Hangzhou, China

Zulong Chen†
zulong.czl@alibaba-inc.com
Alibaba Group
Hangzhou, China

Ying Zhou‡
zhouying@zhejianglab.org
Zhejiang Lab
Hangzhou, China

Junhao Ji†
jijunhao.jjh@alibaba-inc.com
Alibaba Group
Hangzhou, China

Zhibo Yang§
yangzhibo450@gmail.com
Alibaba Cloud
Hangzhou, China

Yuxue Zhang†
yuxue.zyx@alibaba-inc.com
Alibaba Group
Hangzhou, China

Haoyuan Hu¶
haoyuan.huhy@antgroup.com
Ant Group
Hangzhou, China

Zhenghao Liu‖
liuzhenghao@mail.neu.edu.cn
Northeastern University
Shenyang, China



## Abstract

Automated resume information extraction is critical for scaling talent acquisition, yet real-world deployment faces three major challenges: the extreme heterogeneity of resume layouts and content, the high cost and latency of large language models (LLMs), and the lack of standardized datasets and evaluation tools. In this work, we present a layout-aware and efficiency-optimized auto-extraction and evaluation framework to addresses all three challenges. Our system combines a fine-tuned layout parser to normalize diverse document formats, an inference-efficient LLM extractor based on parallel prompting and instruction tuning, and a robust two-stage automated evaluation framework supported by new benchmark datasets. Extensive experiments show that our framework significantly outperforms strong baselines in both accuracy and efficiency. In particular, we demonstrate that a fine-tuned compact 0.6B LLM achieves top-tier accuracy yet significantly reduces inference latency and computational cost. The system is fully deployed in Alibaba's intelligent HR platform, supporting real-time applications across business units.



## Keywords

Layout-Aware Parser, Parallel Prompt Routing, Automatic Resume Analysis


## 1 Introduction

The ability to efficiently and accurately screen resumes has become a critical part of the recruitment process in modern enterprises. However, manual review is slow, costly, and prone to error, making it impractical for industrial use. While automated resume analysis offers a solution, existing methods often struggle to balance accuracy, latency, and computational cost.

Early resume parsing systems, built on handcrafted rules or traditional statistical models [6, 22], offer fast processing but fail to generalize across the vast diversity of linguistic styles and visual layouts found in real-world resumes. Conversely, modern LLMs [4, 9, 11] provide the deep semantic understanding needed for robust extraction, but their high inference latency and computational expense are often prohibitive for real-time, large-scale deployment. An effective industrial system must therefore bridge this gap, delivering state-of-the-art accuracy without compromising on production efficiency and cost.

**Challenges.** Specifically, building a practical resume analysis system at industrial scale requires addressing three key challenges

- *Layout and content heterogeneity*: Resumes submitted by candidates are highly diverse in both structure and content. Many contain important information embedded in images or use complex, multi-column formats that disrupt standard reading order, making consistent parsing difficult.
- *High cost and latency of LLMs*: While LLMs offer strong extraction capabilities, directly applying them to raw text leads to high latency and token usage, which is costly and unsuitable for real-time, large-scale applications.
- *Lack of data and evaluation tools*: Due to privacy concerns, high-quality annotated resume datasets are rare. Evaluating extraction quality, especially for list-style entities like work experience, is hard to do manually at scale, calling for automated and reliable evaluation frameworks.

In this work, we present a practical, layout-aware, and efficiency-optimized framework for automatic resume extraction and evaluation. Our system addresses the above challenges through the following key components: *First,* we introduce a unified layout parsing model that fuses PDF metadata with OCR content and employs a fine-tuned resume layout parser to reconstruct a semantically coherent reading order from diverse, often multi-column resume layouts. *Second,* to enable efficient LLM-extraction, we adopt a task decomposition strategy with index-based pointer outputs, reducing both token usage and response time. On top of this, we fine-tune a compact 0.6B model using instruction supervision, enabling high accuracy at low cost. *Third,* we develop a two-stage evaluation framework using the Hungarian algorithm for entity alignment

and multi-strategy field matching. This enables robust, fine-grained assessment without human involvement.

Extensive experiments on a synthetic dataset with diverse layouts and a complex real-world resume dataset demonstrate that our system consistently outperforms state-of-the-art baselines in accuracy and efficiency. Notably, our fine-tuned Qwen3-0.6B-SFT model surpasses the accuracy of top-tier models like Claude-4 while offering 3–4× faster inference. The complete system is deployed within Alibaba Group's intelligent HR platform, where it supports real-time parsing with high throughput across multiple business units.

**Contributions.** In summary, our key contributions are as follows:

- We propose a unified, layout-aware resume parsing framework that robustly handles layout and content heterogeneity.
- We design an inference-efficient LLM extraction strategy and fine-tune a compact model to achieve competitive accuracy at low latency and cost.
- We introduce a robust two-stage automated evaluation protocol for field-level performance measurement.
- We open-source the full pipeline, along with the datasets to promote future research and practical adoption.[1]

## 2 Related work

**Rich document understanding.** Resume information extraction is closely related to the broader field of visually-rich document understanding. Traditional NLP models, which process text as a 1D sequence, often fail on semi-structured documents like resumes, invoices, and forms. Recent advances in this area have been driven by multimodal models that jointly captured textual content and layout structure. Xu *et al.* [20] introduced a pre-training model LayoutLM that integrates text content with bounding box information, overcoming limitations of purely text-based NLP models on various document understanding tasks. LayoutLMv2 [21] further enhanced this approach by incorporating visual features from raw document images alongside text and layout signals. LayoutLMv3 [10] unified token and patch representations in a single Transformer, demonstrating improved multimodal reasoning.

Recent efforts have also explored more efficient alternatives. Wang *et al.* [19] proposed to integrate spatial structure into a language model using disentangled attention, avoiding the use of a heavy vision encoder. Zhang *et al.* [23] focused on training-free prompting strategies for LLMs, using entity, layout, and document similarities to construct more effective in-context examples.

**Automatic Resume Analysis.** The automated analysis of resumes has a rich history, with research evolving from traditional rule-based systems to modern neural models. Early approaches framed it as a Named Entity Recognition (NER) problem and solved it with hierarchical extraction pipelines. For example, Yu *et al.* [22] proposed a two-pass cascaded hybrid model that first used a Hidden Markov Model (HMM) to segment the resume into general sections (*e.g.*, Education, Experience) and then applied specialized models (HMM or SVM) within each block. Chen *et al.* [6] advanced the cascaded approach by explicitly incorporating PDF-specific layout features (*e.g.*, font size, coordinates) into both their SVM-based block classifier and their CRF-based field extractor. More recently, Zu *et al.* [24] modernized this pipeline by replacing feature-engineered models with neural networks. Their system first employs neural classifiers for a sophisticated line-by-line segmentation of the resume into text blocks, which are then processed by a BiLSTM-CNN-CRF model for final entity extraction. Beyond direct extraction, research has also explored more complex and application-oriented tasks. For instance, Pawar *et al.* [16] moved beyond simple NER to tackle the extraction of complex, N-ary, cross-sentence relations, using a joint hierarchical neural model. Daryani *et al.*[7] built an end-to-end candidate ranking system using an NLP parser followed by IR-based resume-to-job matching. Ali *et al.* [2] focused on document-level classification to sort resumes into job categories using SVMs.

While prior methods have advanced the field, our work uniquely targets the practical challenges of deploying resume extraction systems in real-world hiring workflows. We explicitly address the layout and content heterogeneity of resumes, the high latency and cost of LLM inference at scale, and the inefficiency of manual evaluation—enabling accurate, scalable, and production-ready resume analysis for industrial use.

## 3 Approach

Our layout-aware automatic resume information extraction system follows a three-stage architecture as illustrated in Figure 1. First, a *Layout-Aware Parsing and Regeneration* stage performs hybrid content fusion (integrating PDF metadata and OCR) and uses a fine-tuned layout regenerator to re-order text from complex, multi-column layouts into a single, indexed sequence. Second, a *Parallelized, Instruction-Tuned LLM Extractor* processes this normalized text, using parallelized task decomposition and an index-based pointer mechanism with our Qwen-0.6B-SFT model to balance high accuracy with low-latency inference. Finally, a *Two-Stage Automated Evaluation framework* ensures objective assessment by first using the Hungarian algorithm for robust entity alignment and then applying a multi-strategy matching logic for a fine-grained, field-level comparison.

### 3.1 Layout-Aware Parsing and Regeneration

*3.1.1 PDF Parser: Text Extraction & Fusion.* Resume files submitted by candidates vary widely in format such as PDF, Word, *etc.*. To support consistent downstream processing, we first convert all files to PDF for unified processing. However, PDF resumes still include challenges such as embedded images, non-standard fonts, and custom encodings that hinder reliable text extraction. To address this, we adopt a hybrid content extraction strategy that combines metadata-based parsing with OCR.

**Hybrid PDF Content Extraction.** We extract available metadata from the PDF, which typically includes structured text content and associated positional metadata (*e.g.*, bounding boxes) for each text object. In parallel, we render each PDF page into an image and identify candidate image regions by masking out known text regions using metadata bounding boxes. Remaining unmasked areas are treated as image regions and processed with a state-of-the-art

[1]The code and dataset will be released at Alibaba Open Source: https://github.com/ALIBABA upon completion of the internal approval process.

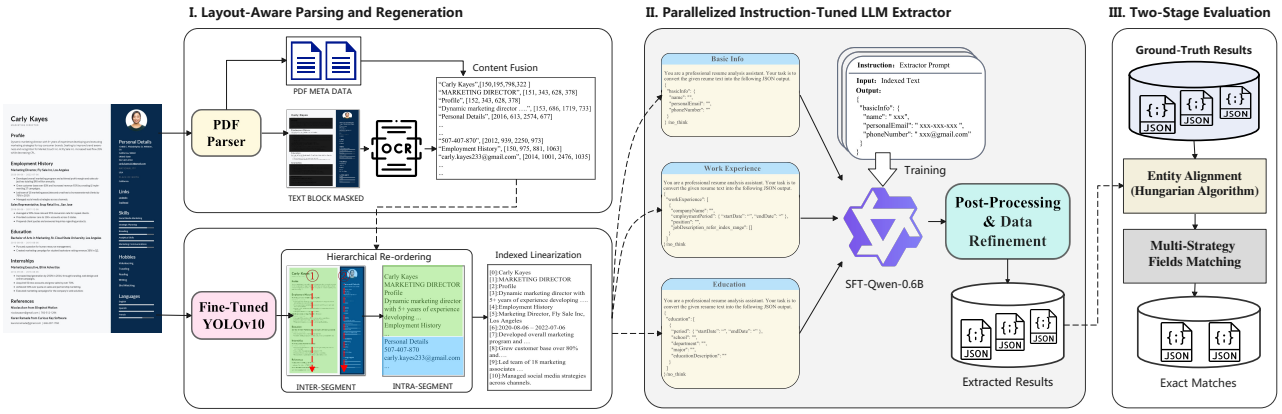


**Figure 1: Overview of the layout-aware, LLM-powered resume extraction and evaluation pipeline.**

OCR engine. The OCR results are then spatially aligned back to their corresponding coordinates within the document.

**Content Fusion.** The final step in this stage is to fuse the textual content obtained from the metadata-based PDF parsing with OCR-derived image region text. The result is a single, layout-aware set of content primitives. Each primitive is a tuple containing a text string and its corresponding bounding box coordinates (text, $x_{min}$, $y_{min}$, $x_{max}$, $y_{max}$). This fusion ensures that all textual information from the resume is captured, creating a complete, but structurally unordered collection of resume content that serves as the input for our layout unification stage.

*3.1.2 Layout Regeneration: Fine-tuned Layout Parser & Unification.* Having obtained a complete but unordered set of text primitives, the next critical task is to reconstruct a semantically coherent reading flow.

Our empirical analysis shows that approximately 20% of resumes employ non-linear, multi-column layouts that break the standard top-to-bottom, left-to-right reading flow. This makes content-level parsing alone insufficient; layout understanding and unification are essential. Conventional document layout analysis requires fine-grained annotations, which are costly and impractical for privacy-sensitive resume data. To address this, we propose a lightweight layout reconstruction approach that segments resumes into logically coherent blocks and reorders them hierarchically to produce a unified reading sequence.

**Layout Segments Identification.** We frame the problem of identifying layout regions (*e.g.*, sidebars, main content columns) as an object detection task with a simplified objective: to partition a non-linear page into a set of smaller, linearly-structured layout segments. A segment is considered linear if its internal content can be correctly read by a simple top-to-bottom, left-to-right sort.

To achieve this, we fine-tune the YOLOv10 object detection model [18] for the resume domain. We construct a resume-specific segmentation dataset of around 500 resumes, annotating only the bounding boxes of the major layout segments required to enforce linearity within each box. This lightweight annotation strategy avoids the burden of fine-grained labeling while effectively capturing internally linear blocks.

**Hierarchical Re-ordering and Indexed Linearization.** After segmenting each document into layout regions (also referred to as *big blocks*), we impose a two-level hierarchical sorting strategy to regenerate a uniform layout with consistent reading order:

- Inter-segment sorting: The detected layout segments are sorted globally according to their visual positions (*i.e.*, the coordinates of their top-left corners), following a standard top-to-bottom, left-to-right rule. This establishes the high-level reading flow, such as processing the main content column before a right-aligned sidebar if they start at the same vertical position.
- Intra-segment text block sorting: Within each layout segment, we further sort the individual text primitives (*i.e.*, text blocks) that fall within its boundaries. This local sort also follows the top-to-bottom, left-to-right principle, arranging the lines and words into a coherent, readable sequence.

These sorted blocks are concatenated to form a single, linearized text stream that conforms to human reading patterns. Noticeably, when constructing this sequence, we assign a unique, sequential index (*i.e.*, a line number) to each line in the text block. This indexing brings significant efficiency gain as we will discuss in the subsequent information extraction stage (Section 3.3).

## 3.2 Parallelized Instruction-Tuned LLM Extractor

With layout unification completed, the document is converted into an indexed, linear text sequence. The final step is to extract structured key-value information suitable for automated recruitment systems. To achieve higher accuracy and robustness, our framework leverages the advanced understanding and reasoning of LLMs to design an LLM-based extraction strategy with a particular focus on the efficiency optimizations.

*3.2.1 Parallelized Extraction via Task Decomposition.* Our primary goal is to extract four categories of resume content essential for

downstream HR applications: *Basic Information* (*e.g.*, name, contact details, location), *Work Experience* (*e.g.*, company, title, dates, description), and *Education Background* (*e.g.*, institution, degree, major, dates). A naïve approach would be to prompt the LLM to extract all fields in a single pass. However, this method is suboptimal, as complex prompts requiring the model to identify disparate types of information simultaneously can degrade performance. Instead, we adopt a *parallelized task decomposition* strategy, decomposing the task into three independent sub-tasks, one for each information category. The complete, linearized resume text is processed in three parallel threads, with each thread invoking the LLM with a highly specialized prompt tailored to its specific extraction target. For example, the prompt for basic information extraction explicitly requests name, phone, email, *etc.*, while the work experience prompt specifies extraction of job titles, time spans, and description. This parallelized, decomposed approach not only improves extraction accuracy but also significantly reduces end-to-end latency. Example prompts are provided in Appendix A.

**Index-based Pointer Mechanism.** To further address the challenges of high latency and content drift when extracting long, descriptive fields (*e.g.*, *work description*, *project description*), we apply a key optimization of index-based pointer mechanism in our framework. Rather than prompting the LLM to generate the full verbatim content, a process that is slow, token-intensive, and prone to generative errors like paraphrasing or hallucination, we explicitly prompt it to return a line number range (*e.g.*, [15, 25]) referencing the indexed text from our layout regeneration stage (Section 3.1.2).

This transforms the task for the LLM from a complex, open-ended generative task into a much simpler and more constrained span identification task, offering two major benefits:

- Efficiency: Returning index ranges requires only a few tokens, significantly reducing inference time and cost compared to generating full descriptions.
- Content Fidelity: During our post-processing stage, we use these returned indices to re-extract the exact text block from the original source document, guaranteeing 100% content fidelity.

*3.2.2 Model Selection and Fine-Tuning.* Although large language models such as GPT-4o and Qwen-max exhibit strong generalization and semantic capabilities, their high latency and inference cost are often prohibitive for real-time, large-scale enterprise applications. In contrast, compact models like Qwen-0.6B offer significantly faster inference (approximately 150 tokens per second), enabling them to process a typical resume (300–400 tokens) in 1–2 seconds per extraction sub-task. However, the lightweight Qwen-0.6B lacks sufficient performance for complex extraction tasks due to limited capacity.

To address this, we fine-tune Qwen-0.6B on a carefully constructed supervised dataset tailored for resume parsing. The supervised fine-tuning (SFT) dataset consists of 15,500 resumes and 59,500 instruction-based training samples. Each training sample is represented as a triplet of (instruction, input, output), where the instruction specifies the extraction task (*e.g.*, extracting work experience), the input is the indexed resume text, and the output is corresponding human-verified JSON-format label. The dataset combines both synthetic and real-world resumes covering diverse formats and content styles. Further construction details are provided in Appendix B. After fine-tuning, the adapted Qwen-0.6B model achieves strong extraction accuracy comparable to much larger LLMs, while maintaining high inference efficiency, making it practical and scalable for real-time resume information extraction in industrial deployment.

**Output Formatting.** To decide the suitable output format, we conducted experiments comparing several formats and adopt JSON as the output format. While alternatives like YAML or Markdown reduce token length, JSON yields the most stable results, likely due to Qwen's fine-tuning on JSON-style data. Instead of using JSON-compatible decoding modes (*e.g.*, automaton decoder), which often compromise the model's parsing capabilities, we adopt a prompt-based strategy that naturally guides the model to produce valid JSON and apply lightweight string-based extraction (*e.g.*, text.find("{") and text.rfind("}")) to retrieve valid JSON blocks.

*3.2.3 Post-Processing and Data Refinement.* As the raw model outputs often contain formatting inconsistencies or hallucinated content, we further implement a multi-stage post-processing and data refinement pipeline to enhance the data fidelity. The process consists of four key stages:

- Grounded content re-extraction: To mitigate content drift in long-form fields (*e.g.*, *job* or *project descriptions*), we use the line indices returned by the LLM as pointers to re-extract all descriptive text directly from the original source document, eliminating any content drift introduced by the model.
- Domain-specific normalization: We apply a set of domain-specific cleaning and normalization rules to standardize volatile fields, such as normalizing diverse date formats and removing suffix noise in organization names.
- Context-aware de-duplication: We identify and filter redundant entries, such as a project that is also mentioned within a work experience, by comparing the source text spans (*i.e.*, line number ranges) of all extracted entities.
- Source text verification: Finally, we perform a verification step for every extracted record, discarding any entity whose key identifying fields (*e.g.*, *company name* and *job title*) cannot be found in the original document text, thereby pruning model hallucinations.

Through this rigorous four-stage pipeline, we transform raw LLM outputs into clean, trustworthy, and application-ready structured data that faithfully reflects the content of the original document.

## 3.3 Two-Stage Automatic Evaluation

After extracting the key information, another critical task is to evaluate the accuracy of extraction. Manual evaluation is impractical for large-scale system development as it is prohibitively time-consuming, expensive, and often subject to human inconsistency. To enable efficient comparison and provide objective, reliable results, an automated evaluation methodology is essential.

For clarity, we define two key terms in resume information extraction:

- An *Entity* refers to a complete block of information. For example, a single "Work Experience" or one "Education History" entry is considered one entity.

- A *Field* refers to a specific attribute within an entity. For a "Work Experience" entity, its fields would be company name, position, start date, end date, and description.

Our primary evaluation challenge is to accurately compare the list of entities, along with their fields predicted by our model against a ground truth list. However, designing such an automatic evaluation model is non-trivial. A naive approach, such as a one-to-one comparison of entities based on their sequential order often fails due to the following challenges:

- Quantity mismatch: The number of predicted entities may differ from ground-truth, leading to either missed extractions or spurious extractions.
- Order mismatch: Even if both lists contain the same entities, the model may extract them in an order that differs from the ground-truth, causing a naive index-based comparison to fail.
- Partial match: An extracted entity could be only a partial match as some fields may incorrectly extracted or even missing. For instance, a predicted *Work Experience* entity might have the correct company name and position fields, but a different data format or incomplete description text.

To overcome the above challenges, we propose a robust, two-step automated evaluation framework that intelligently aligns entities, performs a fine-grained field-level comparison, and aggregates the results into clear, quantitative metrics.

**Entity Alignment via the Hungarian Algorithm.** For any pair of lists to be compared, for instance, a ground-truth list of $M$ work experiences $G = \{g_1, \ldots, g_M\}$ and a predicted list of $N$ experiences $P = \{p_1, \ldots, p_N\}$, we construct an $M \times N$ similarity matrix $S$. Each element $S_i j$ represents the similarity between $g_i$ and $p_j$, which is computed as the average normalized string similarity of their key fields, such as company name and position. We then apply the Hungarian algorithm [12] to $S$ to find one-to-one assignment that maximizes the total similarity of all matched pairs $\{(p_i, q_j)\}$. This naturally resolves the aforementioned challenges: it is impervious to order mismatch and can find the optimal partial assignment even when $M \neq N$.

**Multi-Strategy Fields Matching.** Once entity pairs are aligned, we proceed to a fine-grained comparison of their constituent fields for each aligned entity pair. Recognizing that a single "exact match" rule is inadequate for diverse data types, we designed a multi-strategy comparison function that dynamically selects the most appropriate validation rule based on the field's semantic nature.

- Period Fields (*e.g.*, dates): Normalized into (year, month) format to support flexible matching across date representations.
- Named Entities (*e.g.*, organizations, schools, job titles): Compared using partial substring matching to tolerate abbreviations or suffix differences.
- Long Descriptions (*e.g.*, project or job descriptions): Matched via edit-distance-based similarity (*e.g.*, edit distance > 0.9) to allow for minor paraphrasing.
- Other Fields (*e.g.*, names, email addresses): Evaluated using normalized exact match after lowercasing and punctuation removal.

An extracted field is considered as correct only if it is *both* successfully aligned with a ground-truth entity by the Hungarian algorithm *and* subsequently passes the multi-strategy field matching criteria.

To validate the reliability of this automatic evaluation framework, we conducted human verification on a subset of results. The evaluation consistently aligned with human judgment, confirming its effectiveness in producing accurate matching outcomes.

Finally, we aggregate the field-level matching outcomes across all evaluated resumes to compute quantitative performance metrics for each field. Such per-field evaluation not only aggregates to overall extraction quality but also highlights field-specific weaknesses-offering valuable insights for further model improvement.

# 4 Experiments

## 4.1 Experimental Settings

*4.1.1 Datasets.* As there is no public resume dataset for evaluation, we construct two distinct datasets to evaluate our model's performance.

- **SynthResume**. SynthResume is a synthetic corpus of 2,994 resumes with different layouts, which is constructed through a semi-automated, LLM-based generation pipeline (Figure 8). This process involves curating a diverse set of resume templates and using an LLM to populate them with new, randomized yet plausible content. Each sample undergoes layout parsing and pre-labeling via Qwen-max, followed by manual correction to ensure annotation quality. Details of dataset construction is provided in Appendiex C.
- **RealResume**. RealResume is a real-world private dataset of 13,100 real-world resumes collected from Alibaba's HR system. This dataset is inherently more complex, featuring documents with challenging characteristics such as custom fonts, intricate layouts, and mixed Chinese-English content. The annotation process for this dataset followed the same methodology as for SynthResume.

*4.1.2 Methods for Comparison.* To comprehensively evaluate our proposed framework, we compare it against a wide range of baseline systems, as well as benchmark different LLMs in our pipeline. We categorize the compared methods into four groups:

- **Non-LLM Baselines.** This includes a commercial resume parsing system widely used in industry Bello [5], and a deep learning-based information extraction pipeline built on the PaddlePaddle framework PaddleNLP [17].
- **Naïve LLM Baseline.** This approach directly applies a large language model (Claude-4 [4]) to OCR-extracted resume text, without any layout-aware preprocessing or task decomposition.
- **Our Pipeline (Zero-Shot LLM).** We evaluate our full layout-aware pipeline paired with various off-the-shelf LLMs, including Claude-4 [4], Gemini-2.5flash [9], GPT-4o [15], Deepseek-v3 [11], Qwen-max [1], and Qwen3 series models [3], to examine how different models perform in a zero-shot setting within our architecture.
- **Our Pipeline (Fine-Tuned LLM).** This variant uses our pipeline with a supervised fine-tuned model, Qwen3-0.6B-SFT, trained on our SFT dataset. It represents our final, production-oriented system, designed to balance top-tier accuracy with the efficiency required for large-scale deployment.

Details about these models are provided in Appendix D.

**Table 1: Statistics of Datasets.**

| Dataset | Source | Size | Fields | Language | Layout | Data Split (Train/Test) |
|---|---|---|---|---|---|---|
| **SynthResume** | Semi-Automated Synthetic | 2,994 | 15 | Chinese | Linear & Non-Linear Templates | 2,500 / 494 |
| **RealResume** | Real-World | 13,100 | 19 | Mixed (Eng/Chn) | More Complex, Custom Fonts | 13,000 / 100 |

*4.1.3 Metrics.* We evaluate our system using standard information extraction metrics: *Precision*, *Recall*, and *F1-score*, following prior work [8]. Evaluation is conducted at the field level, where an extracted field is considered a correct match only if it satisfies both entity alignment and field-matching criteria. Additionally, we introduce a novel alignment *Accuracy* metric that measures the fraction of correctly extracted fields among all aligned fields. This metric helps us distinguish errors caused by the alignment algorithm from those caused by our field-matching rules.

Formal definitions of the metrics are provided in Appendix E.

*4.1.4 Implementation Details.* We fine-tune the Qwen-0.6B model using a full-parameter supervised fine-tuning approach, updating all trainable parameters to maximize task-specific performance. For optimization, we employ the AdamW optimizer [14]. The training process is stabilized by a low initial learning rate of 5e-6, which helps prevent catastrophic forgetting. The learning rate is managed by a cosine annealing scheduler [13] to facilitate smooth convergence in the later stages of training. To accommodate GPU memory constraints, we set the per-device batch size to 2. We apply gradient accumulation over 2 steps, resulting in an effective batch size of 4. This configuration provides a robust balance between training stability and computational efficiency, allowing us to effectively train the model within our hardware limitations.

## 4.2 Impact of Hyper-parameters

We analyze the effects of two key decoding parameters: *repetition penalty* and *temperature* on extraction performance. We report the results on the SynthResume dataset in in Figure 2, while similar trends observed on RealResume.

The repetition penalty discourages the model from generating repetitive content by penalizing previously generated tokens. We vary it while keeping temperature at 0 and find that a small penalty of 1.01 yields the best F1 score. The temperature controls the randomness of token sampling. For this experiment, the repetition penalty is fixed at its optimal value (1.01). We find that a moderate temperature of 0.5 provides the most stable and accurate performance. These optimal settings are adopted for all subsequent experiments.

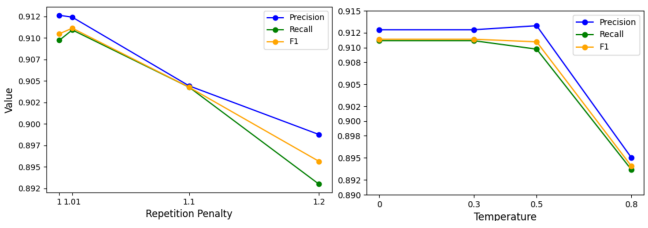


**Figure 2: Impact of Hyper-parameters.**

## 4.3 Performance Comparison

We evaluate and compare the performance of various resume information extraction methods across both the SynthResume and RealResume datasets. The overall performance averaged across all resume fields, along with average processing time per resume, are presented in Table 2. We also report fine-grained accuracy comparisons across different field groups in Table 3. Specifically, the *Period* group includes four date-related fields: employment start/end and education start/end. The *Named Entity* group includes six entity fields: company name, job title, school, major, degree, and department. The *Long Text* group comprises two descriptive fields: job description and education description. From the general and fine-grained comparison, we observe the following key findings.

**First, our layout-aware pipeline is critical for achieving state-of-the-art performance.** As shown in Table 2, our pipeline consistently outperforms all baselines. On the SynthResume dataset, the naïve LLM baseline (Claude-4) achieves an F1-score of 0.927, whereas integrating Claude-4 into our layout-aware pipeline boosts it to 0.946. The improvement is even more pronounced on the RealResume dataset, which includes more complex, mixed-language resumes- raising the F1-score from 0.919 to 0.959, a substantial 40-point gain. This significant improvement that validates the effectiveness of our framework, particularly in handling real-world resume complexity. Moreover, our pipeline outperforms traditional non-LLM methods by a wide margin. Compared to Bello, a state-of-the-art industrial system, our fine-tuned Qwen3-0.6B-SFT model improves F1-score by over 14 points (0.817 → 0.964), while also reducing inference latency (1.62s → 1.54s), making it well-suited for real-world deployment.

**Second, our fine-tuned small model offers the optimal trade-off between accuracy and efficiency.** While top-tier models such as Claude-4 and Gemini-2.5 deliver strong performance, their inference latency (4–13 seconds per resume) limits scalability in high-throughput environments. Our fine-tuned Qwen-0.6B-SFT model provides the optimal balance of performance and efficiency for a production environment. As shown in Table 2, supervised fine-tuning boosts the base Qwen3-0.6B model's F1-score on RealResume from 0.641 to 0.964- surpassing even Claude-4 (0.959) while reducing latency to just 1.54 seconds, achieving a 3–4× speedup. These results validate the effectiveness of fine-tuning compact models to meet both high accuracy and low latency demands in production-scale resume processing.

**Third, the gains of our framework are most evident in complex Long Text fields.** Long Text fields are most challenging as they require coherent extraction of multi-sentence descriptions. As shown in Table 3, on RealResume, the naïve LLM baseline (Claude-4) achieves only an F1-score of 0.548 in this group, but with our full pipeline, the score rises sharply to 0.854. A similar boost is seen

**Table 2: Comparison of Overall Model Performance on the SynthResume and RealResume Datasets. Results are averaged across all resume fields to provide a holistic evaluation of each method. Best scores are in bold, second-best are underlined.**

| | | SynthResume Dataset | | | | | RealResume Dataset | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| **Category** | **Model** | **Acc. ↑** | **Prec. ↑** | **Recall ↑** | **F1 ↑** | **Time (s) ↓** | **Acc. ↑** | **Prec. ↑** | **Recall ↑** | **F1 ↑** | **Time (s) ↓** |
| Non-LLM Baselines | Bello | 0.815 | 0.787 | 0.741 | 0.762 | 1.43 | 0.835 | 0.836 | 0.746 | 0.817 | 1.62 |
| | PaddleNLP | 0.576 | 0.669 | 0.474 | 0.523 | 22.0 | 0.515 | 0.584 | 0.422 | 0.492 | 20.9 |
| Naïve LLM Baseline | Claude-4 | 0.926 | 0.923 | 0.933 | 0.927 | 20.54 | 0.896 | 0.896 | 0.901 | 0.919 | 22.71 |
| Our Pipeline (Zero-Shot LLM) | Claude-4 | 0.949 | 0.950 | 0.943 | 0.946 | 4.98 | 0.948 | 0.937 | 0.952 | 0.959 | 4.62 |
| | Gemini2.5-flash | 0.949 | 0.958 | 0.945 | 0.951 | 11.23 | 0.947 | 0.933 | 0.955 | 0.954 | 13.67 |
| | GPT-4o | **0.952** | 0.958 | **0.948** | **0.952** | 5.50 | 0.944 | 0.936 | 0.950 | 0.954 | 6.26 |
| | Deepseek-v3 | 0.951 | **0.959** | 0.941 | 0.950 | 8.66 | 0.939 | 0.935 | 0.936 | 0.944 | 10.58 |
| | Qwen-max | 0.950 | 0.945 | 0.947 | 0.946 | 9.40 | 0.935 | 0.927 | 0.934 | 0.937 | 19.2 |
| | Qwen3-14B | 0.911 | 0.906 | 0.911 | 0.908 | 6.30 | 0.914 | 0.898 | 0.911 | 0.912 | 8.55 |
| | Qwen3-4B | 0.885 | 0.876 | 0.895 | 0.885 | 5.24 | 0.861 | 0.833 | 0.887 | 0.869 | 6.85 |
| | Qwen3-0.6B | 0.618 | 0.671 | 0.663 | 0.645 | **1.22** | 0.589 | 0.632 | 0.622 | 0.641 | **1.54** |
| Our Pipeline (Fine-Tuned Model) | Qwen3-0.6B-SFT | 0.931 | 0.918 | 0.917 | 0.917 | **1.22** | **0.961** | **0.938** | **0.964** | **0.964** | **1.54** |

with model fine-tuning: Qwen3-0.6B's F1-score for *Long Text* jumps from 0.136 to 0.846 after SFT. Accurately extracting long text (e.g., job descriptions and project summaries) is essential for downstream tasks such as candidate-job matching, making this improvement especially impactful in real-world hiring applications.

Interestingly, naïve LLM baseline achieves the highest F1-score on *Period* fields. This suggests that LLMs may possess strong intrinsic capabilities for handling short, visually distinct, and highly regular patterns such as dates. In such cases, our layout regenerator may introduce minor segmentation noise that slightly degrades performance. This observation motivates future work on field-specific extraction strategies to further optimize performance.

## 4.4 Ablation Study

To understand the impact of each major component in our pipeline, we conduct an ablation study with SynthResume Dataset by removing three core modules individually: (1) *w/o Text Fusion*: removing PDF's metadata-based text content and relying solely on OCR for text extraction; (2) *w/o Layout Generator*: Disabling the hierarchical layout reordering, a naive top-to-bottom, left-to-right sort is applied directly to the text boxes; (3) *w/o Post-Processor*: Skipping the post-processing step in the LLM-based extractor, the model directly generate the full long description.

**Table 4: Ablation study on key system components.**

| | | Finer-Grained Accuracy ↑ | | |
|---|---|---|---|---|
| **Variants** | **Overall Acc. ↑** | **Period** | **Named Entity** | **Long Text** |
| w/o Text Fusion | 0.907 | 0.892 | 0.945 | 0.743 |
| w/o Layout Generator | 0.916 | 0.892 | 0.950 | 0.758 |
| w/o Post Processor | 0.921 | 0.897 | 0.952 | 0.781 |
| **Full system** | **0.932** | **0.897** | **0.952** | **0.858** |

As shown in Table 4, each component of our framework contributes meaningfully to overall performance. The full system achieves the highest overall accuracy (0.932), and removing any component leads to a clear drop, validating the effectiveness of our integrated design. *Finer-grained analysis* further shows that *Long Text* fields are the most sensitive. Removing Text Fusion or Layout Generator causes over a 10-point drop in accuracy, due to OCR errors and disrupted reading order in multi-line content. Omitting the Post Processor also leads to a 7.7-point decline, highlighting the difficulty LLMs face in generating long, verbatim text accurately. Our post-processing module provides crucial robustness and traceability for such complex fields. In contrast, *Named Entity* and *Period* fields are more robust, but still benefit from Text Fusion, confirming that dual-modal PDF text extraction is consistently superior to OCR.

## 4.5 Online Deployment

With the success in offline experiments, our pipeline has been deployed to support Alibaba's Intelligent HR system (CaiMi). The deployment involves both offline training and online serving, as shown in Figure 3.

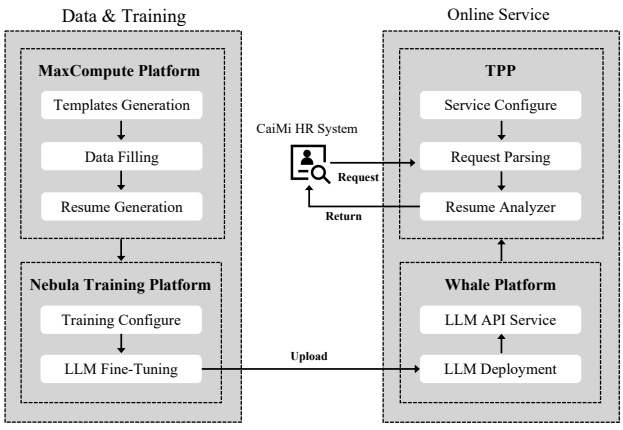


**Figure 3: System Deployment and Serving Framework.**

**Table 3: Fine-grained Accuracy and F1 Scores on the SynthResume and RealResume datasets, grouped by field types: Period, Named Entity, and Long Text. Missing values indicate that the PaddleNLP baseline does not support extraction of Long Text fields. Full results including Precision and Recall are provided in Appendix F.**

| Model | SynthResume Dataset | | | | | | RealResume Dataset | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | Period | | Named Entity | | Long Text | | Period | | Named Entity | | Long Text | |
| | Acc. ↑ | F1 ↑ | Acc. ↑ | F1 ↑ | Acc. ↑ | F1 ↑ | Acc. ↑ | F1 ↑ | Acc. ↑ | F1 ↑ | Acc. ↑ | F1 ↑ |
| *Non-LLM Baselines* | | | | | | | | | | | | |
| Bello | 0.852 | 0.885 | 0.841 | 0.749 | 0.469 | 0.259 | 0.921 | 0.879 | 0.885 | 0.769 | 0.540 | 0.500 |
| PaddleNLP | 0.433 | 0.511 | 0.818 | 0.699 | – | – | 0.387 | 0.451 | 0.722 | 0.622 | – | – |
| *Naïve LLM Baseline* | | | | | | | | | | | | |
| Claude-4 | **0.977** | **0.980** | 0.939 | 0.951 | 0.731 | 0.684 | 0.979 | **0.986** | 0.937 | **0.959** | 0.582 | 0.548 |
| *Our Pipeline (Zero-Shot LLM)* | | | | | | | | | | | | |
| Claude-4 | 0.960 | 0.973 | 0.963 | 0.956 | 0.825 | 0.801 | 0.963 | 0.972 | **0.964** | 0.949 | 0.869 | 0.854 |
| Gemini2.5-flash | 0.963 | 0.975 | 0.959 | 0.957 | 0.835 | **0.831** | 0.970 | 0.978 | 0.945 | 0.931 | **0.888** | 0.865 |
| GPT-4o | 0.960 | 0.974 | 0.966 | 0.961 | 0.840 | 0.829 | 0.963 | 0.972 | 0.950 | 0.941 | 0.880 | **0.870** |
| Deepseek-v3 | 0.961 | 0.971 | 0.968 | 0.962 | 0.829 | 0.813 | 0.960 | 0.971 | 0.939 | 0.918 | 0.867 | 0.852 |
| Qwen-max | 0.942 | 0.945 | **0.979** | **0.970** | 0.824 | 0.811 | 0.971 | 0.978 | 0.936 | 0.915 | 0.827 | 0.820 |
| Qwen3-14B | 0.888 | 0.902 | 0.961 | 0.954 | 0.698 | 0.664 | 0.963 | 0.972 | 0.939 | 0.899 | 0.724 | 0.707 |
| Qwen3-4B | 0.893 | 0.889 | 0.955 | 0.953 | 0.513 | 0.527 | 0.901 | 0.930 | 0.921 | 0.886 | 0.579 | 0.567 |
| Qwen3-0.6B | 0.619 | 0.668 | 0.637 | 0.691 | 0.256 | 0.184 | 0.680 | 0.734 | 0.647 | 0.671 | 0.120 | 0.136 |
| *Our Pipeline (Fine-Tuned Model)* | | | | | | | | | | | | |
| Qwen3-0.6B-SFT | 0.897 | 0.907 | 0.951 | 0.938 | **0.858** | 0.767 | **0.981** | 0.984 | 0.956 | 0.937 | 0.880 | 0.846 |

In the *offline phase*, we construct the training dataset on the MaxCompute Platform, where a diverse set of resume templates is first generated. These templates are populated using LLM-based content synthesis followed by manual correction, producing high-quality synthetic resumes. Combined with real resumes collected from Alibaba's hiring applications, we construct a fine-tuning corpus of labeled examples. All resume texts and labels are stored on Alibaba Cloud's Object Storage Service (OSS). Model fine-tuning is conducted on Neubla, Alibaba's internal distributed AI training platform. We fine-tune the Qwen-0.6B model using full-parameter supervised learning on a compute node with 8× NVIDIA A800 GPUs. With Neubla's optimized training infrastructure, the full fine-tuning process completes within 30 minutes.

At *online serving*, the fine-tuned model is deployed on Whale Platform, Alibaba's LLM-serving infrastructure. Serving orchestration is managed by TPP, Alibaba's internal online inference engine. Upon a parsing request from the CaiMi HR system, TPP orchestrates the entire resume parsing workflow. This includes initial OCR and text fusion, layout regeneration, LLM API invocation via Whale, and finally, returning the extracted structured results back to the HR system. The entire pipeline demonstrates strong real-time performance, achieving a throughput of 240–300 resumes per minute (*i.e.*, 4–5 QPS), with an average response latency of 1.54 seconds per resume. This meets the strict latency and throughput requirements of large-scale enterprise hiring systems.

## 5 Conclusion and Future Work

In this paper, we present a layout-aware, efficiency-optimized automatic resume extraction and assessment pipeline that successfully addresses the key challenges of layout heterogeneity, LLM latency, and evaluation difficulty in industrial-scale resume information extraction. We demonstrate that by combining a robust layout-aware parsing pipeline with a lightweight fine-tuned LLM model, our approach delivers both high accuracy and low latency. The framework significantly outperforms existing baselines and has been successfully deployed in Alibaba's intelligent HR system, serving real-time scenarios with high throughput and reliability. We also open-source the entire pipeline and contribute benchmark datasets to advance future research and practical application. Future work will explore dynamic, field-specific extraction strategies that selectively apply different extraction models, such as applying simpler methods for regular fields while reserving our full pipeline for more complex, context-dependent ones, to further optimize performance.

## References

[1] Alibaba DAMO Academy. 2024. Qwen: Alibaba's Open Multilingual LLM Family. https://huggingface.co/Qwen.
[2] Irfan Ali, Nimra Mughal, Zahid Hussain Khand, Javed Ahmed, and Ghulam Mujtaba. 2022. Resume classification system using natural language processing and machine learning techniques. *Mehran University Research Journal Of Engineering & Technology* 41, 1 (2022), 65–79.
[3] Alibaba DAMO Academy. 2024. Qwen3: Think Deeper, Act Faster. https://qwenlm.github.io/blog/qwen3/. Accessed: 2025-07-26.
[4] Anthropic. 2024. Claude 4 Model Overview. https://www.anthropic.com/news/claude-4. Accessed: 2025-07-26.
[5] Bello AI. 2024. Bello Intelligent Resume Parser. https://www.belloai.com/parser?lan=en. Accessed: 2025-07-26.

[6] Jiaze Chen, Liangcai Gao, and Zhi Tang. 2016. Information extraction from resume documents in pdf format. *Electronic Imaging* 28 (2016), 1–8.
[7] Chirag Daryani, Gurneet Singh Chhabra, Harsh Patel, Indrajeet Kaur Chhabra, and Ruchi Patel. 2020. An automated resume screening system using natural language processing and similarity. *ETHICS AND INFORMATION TECHNOLOGY [Internet]. VOLKSON PRESS* (2020), 99–103.
[8] Y Gyana Deepa, Ankathi Sindhu, Alakuntla Shruthi, and Bitla Neha. 2025. Automated Resume Parsing: A Review of Techniques, Challenges and Future Directions. (2025).
[9] Google DeepMind. 2024. Gemini Flash. https://deepmind.google/technologies/gemini/#models. Accessed: 2025-07-26.
[10] Yupan Huang, Tengchao Lv, Lei Cui, Yutong Lu, and Furu Wei. 2022. Layoutlmv3: Pre-training for document ai with unified text and image masking. In *Proceedings of the 30th ACM international conference on multimedia*. 4083–4091.
[11] DeepSeek Inc. 2024. DeepSeek LLM Technical Report. *Technical Report* (2024). https://huggingface.co/deepseek-ai.
[12] Harold W Kuhn. 1955. The Hungarian method for the assignment problem. *Naval research logistics quarterly* 2, 1-2 (1955), 83–97.
[13] Ilya Loshchilov and Frank Hutter. 2017. SGDR: Stochastic Gradient Descent with Warm Restarts. In *International Conference on Learning Representations (ICLR)*. https://arxiv.org/abs/1608.03983
[14] Ilya Loshchilov and Frank Hutter. 2019. Decoupled Weight Decay Regularization. In *International Conference on Learning Representations (ICLR)*. https://arxiv.org/abs/1711.05101
[15] OpenAI. 2024. GPT-4 Technical Report. arXiv:2303.08774 [cs.CL] https://arxiv.org/abs/2303.08774
[16] Sachin Pawar, Devavrat Thosar, Nitin Ramrakhiyani, Girish K Palshikar, Anindita Sinha, and Rajiv Srivastava. 2021. Extraction of complex semantic relations from resumes. In *ASEA workshop@ IJCAI*.
[17] Baidu NLP Team. 2021. PaddleNLP: An Easy-to-Use and High-Performance NLP Library. https://github.com/PaddlePaddle/PaddleNLP.
[18] Ao Wang, Hui Chen, Lihao Liu, Kai Chen, Zijia Lin, Jungong Han, et al. 2024. Yolov10: Real-time end-to-end object detection. *Advances in Neural Information Processing Systems* 37 (2024), 107984–108011.
[19] Dongsheng Wang, Natraj Raman, Mathieu Sibue, Zhiqiang Ma, Petr Babkin, Simerjot Kaur, Yulong Pei, Armineh Nourbakhsh, and Xiaomo Liu. 2023. DocLLM: A layout-aware generative language model for multimodal document understanding. *arXiv preprint arXiv:2401.00908* (2023).
[20] Yiheng Xu, Minghao Li, Lei Cui, Shaohan Huang, Furu Wei, and Ming Zhou. 2020. Layoutlm: Pre-training of text and layout for document image understanding. In *Proceedings of the 26th ACM SIGKDD international conference on knowledge discovery & data mining*. 1192–1200.
[21] Yang Xu, Yiheng Xu, Tengchao Lv, Lei Cui, Furu Wei, Guoxin Wang, Yijuan Lu, Dinei Florencio, Cha Zhang, Wanxiang Che, et al. 2020. Layoutlmv2: Multimodal pre-training for visually-rich document understanding. *arXiv preprint arXiv:2012.14740* (2020).
[22] Kun Yu, Gang Guan, and Ming Zhou. 2005. Resume information extraction with cascaded hybrid model. In *Proceedings of the 43rd annual meeting of the Association for Computational Linguistics (ACL'05)*. 499–506.
[23] Jinyu Zhang, Zhiyuan You, Jize Wang, and Xinyi Le. 2025. Sail: Sample-centric in-context learning for document information extraction. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 39. 25868–25876.
[24] Shicheng Zu and Xiulai Wang. 2019. Resume information extraction with a novel text block segmentation algorithm. *Int J Nat Lang Comput* 8, 2019 (2019), 29–48.

# Appendix

## A Task-specific Prompts

In the LLM-based Extractor, we adopt a parallelized task decomposition strategy, where each extraction task is handled independently using a specialized instruction prompt. This modular approach improves both extraction accuracy and efficiency in production settings. The task-specific prompts for extracting basic information, education background, and work experience are illustrated in Figures 4, 5, and 6, respectively.

## B Construction of Supervised Fine-Tuning Dataset

To adapt the Qwen-6B language model for resume information extraction, we construct a high-quality supervised fine-tuning (SFT) dataset in an instruction-based format. The SFT dataset consists of two parts:

- **Synthetic Dataset:** We generate 2,500 synthetic resumes using a layout-diverse generation pipeline. For each resume, we create training instances for three extraction tasks (*i.e.*, extracting basic information, work experience, and education background), yielding a total of 7,500 instruction-format samples.
- **Real-World Dataset:** We collect 13,000 real resumes with complex formats and mixed-language content. Each resume is annotated for four extraction tasks (*i.e.*, extracting basic information, work experience, project experience, and education background), resulting in 52,000 SFT training examples.

Each training example is represented as a triplet of (instruction, input, output), where the instruction specifies the extraction task, the input is a fully indexed resume text, and the output is the corresponding structured JSON-formatted label derived from human-annotated ground truth. An example of instruction-based sample for basic information extraction is illustrated in Figure 7. This SFT dataset enables the model to learn fine-grained extraction behavior under explicit instructions and to generalize across diverse layout structures and resume styles.

## C Construction of SynthResume Dataset

To facilitate training and evaluation of our models, we construct a large-scale synthetic resume dataset, SynthResume, consisting of 2,994 richly structured samples. This dataset is generated using a semi-automated pipeline as shown in Figure 8, designed to simulate realistic resumes with diverse layout and content variations.

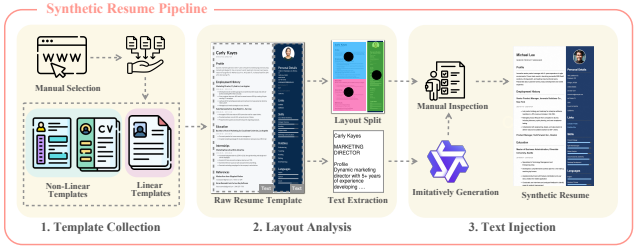


**Figure 8: Pipeline of Synthetic Resume Dataset Construction**

The construction process includes the following key stages:

*Step 1: Template Collection.* We first manually curate a collection of resume templates that reflect a wide variety of real-world styles, including both *linear templates* ( single-column, top-down formats) and *non-linear templates* (complex, multi-column layouts with sidebars).

*Step 2: Layout Analysis and Text Extraction.* Each resume template is processed through a layout analysis module to identify its major visual blocks (*e.g.*, header, main content column, sidebar) and extract the text within each block. This process resulted in a clean, ordered text representation of the original resume's content, which served as the backbone for synthesizing new resumes.

*Step 3: Content Generation and Text Injection.* Each template's extracted content is used as input context to a large language model,

**Prompt for Basic Information Extraction**

Extract the following information into a JSON object. If any field does not exist, output an empty string "".

```
{
  "basicInfo": {
    "name": "",             // Full name, e.g., "Zhang San"
    "personalEmail": "",    // Email address, e.g., "610730297@qq.com"
    "phoneNumber": "",      // Phone number, e.g., "13915732235".
                  // Preserve original format, including country/area code if present (e.g., "+1 (201) 706 1136")
    "age": "",              // Current age (numeric only)
    "born": "",             // Birth year and month if available, e.g., "1996-11"
    "gender": "",           // "Male" or "Female". Leave blank if not specified.
    "desiredLocation": ["city name", ...],
                            // Explicitly mentioned preferred job location(s), e.g., ["Beijing", "Shanghai"].
                            // Only extract if clearly stated. If none, return [].
   "jobIntention": "",   // Job intention or target position, e.g., "Algorithm Engineer". Leave blank if unclear.
    "currentLocation": "", // Current city of residence. Do NOT infer from work experience or place of origin.
    "placeOfOrigin": ""    // Place of origin or hometown. Should not be confused with current location.
  }
}
```

**Figure 4: Prompt for Extracting Basic Information in JSON Format.**

**Prompt for Education Background Extraction**

Extract the following education experiences into a JSON array. If any field does not exist, output an empty string "". If no education experience is mentioned, return an empty array [].

```
{
  "education": [
    {
      "school": "",       // Full name of the educational institution, e.g., "Tsinghua University"
      "major": "",        // Major or field of study, e.g., "Computer Science"
      "degree": "",       // Degree earned, e.g., "Bachelor", "Master", "PhD"
      "startDate": "",    // Start date in "YYYY-MM" format if available, e.g., "2018-09"
      "endDate": "",      // End date in "YYYY-MM" format. If still studying, use "present"
      "location": ""      // City or region of the school, e.g., "Beijing"
    }
  ]
}
```

**Figure 5: Prompt for Extracting Education Background in JSON Format.**

with a specific prompt designed to generate new content while preserving the original field structure: "Given the above resume as a template, generate a new resume by randomly replacing the field values while preserving the structure and semantics." The LLM-generated text undergoes a manual inspection to ensure quality and coherence. Once verified, this new text is injected back into the original visual template. The final output of this stage is a complete, fully-formatted synthetic resume PDF that retains the layout of the source template but contains entirely new, synthetic content.

*Finally,* to create high-quality labels for these resumes, we employ a two-pass annotations:

- Automated Pre-labeling: Each of the 2,994 synthetic resumes is processed by our full extraction pipeline, using a powerful LLM model Qwen-Max to generate an initial set of indexed text and pre-labeled JSON annotations.
- Manual Correction: To ensure label quality, these pre-labels are then subjected to a rigorous manual correction phase by human annotators, correcting any errors made by the initial automated pass.

The fully annotated dataset are sorted by the length of the resume text. The longest 2,500 resumes were allocated to the training set, with the remaining 494 forming the test set. This entire process yielded a high-quality, and structurally diverse dataset ideal for

**Prompt for Work Experience Extraction**

Extract the following work experiences into a JSON array. If any field does not exist, output an empty string "". If no work experience is mentioned, return an empty array [].

```
{
  "workExperience": [
    {
      "company": "",        // Company name, e.g., "Alibaba Group"
      "position": "",       // Job title or role, e.g., "Backend Engineer"
      "startDate": "",      // Start date in "YYYY-MM" format, e.g., "2020-07"
      "endDate": "",        // End date in "YYYY-MM" format; use "present" if currently employed
      "location": "",       // City or region of the job location, e.g., "Hangzhou"
      "description": ""     // Description of responsibilities and achievements; preserve original wording
    }
  ]
}
```

**Figure 6: Prompt for Extracting Work Experience in JSON Format.**

training and evaluating layout-aware information extraction models.

## D Details of Baselines

To comprehensively evaluate our proposed framework, we compare it against a wide range of baseline systems, as well as benchmark different LLMs in our pipeline.

- **Bello** [5]. A widely deployed, commercial resume analysis service in HR automation. It processes resumes across varying formats and layouts, applies bilingual parsing, knowledge graph enhanced field extraction, and document structure understanding at scale. We treat Bello's Intelligent Parser as a strong industrial baseline in our work.
- **PaddleNLP** [17]. An open-source NLP library developed by Baidu based on the PaddlePaddle deep learning framework. It provides pre-trained models and end-to-end pipelines for a wide range of NLP tasks. In our experiments, we adopt its IE workflow for resume field extraction.
- **Claude-4** [4]. The latest large language model from Anthropic, designed for high-performance reasoning and long-context understanding.
- **Gemini-2.5flash** [9]. A lightweight version of Google's Gemini 2.5 model, optimized for fast inference and reduced latency, while maintaining reasonable performance for common LLM tasks.
- **GPT-4o** [15]. OpenAI's flagship multimodal model released in 2024, capable of handling text, audio, and image inputs natively.
- **Deepseek-v3** [11]. An open-source, multilingual LLM developed by DeepSeek that achieves competitive performance across various reasoning and language tasks.
- **Qwen-max** [1]. A state-of-the-art, multilingual large language model with over 100 billion parameters, released as part of the Qwen model series by Alibaba. It is designed for general-purpose reasoning and excels in multi-step, instruction-following tasks.
- **Qwen3 Series** [3]. Qwen3 series includes models of various sizes designed for efficiency and controllability. We compare Qwen3-14B, Qwen3-4B, and Qwen3-0.6B to evaluate the performance of our pipeline with different complexity of LLM models.

## E Details of Metrics

In our experiments, we adapt three standard evaluation metrics Precision, Recall and F1-score [8] to resume entity extraction task, and introduce a novel metric Accuracy to specifically evaluate our multi-strategy field matching logic. Let $E_{gt}$ be the set of ground-truth entity fields, $E_{pred}$ be the set of fields predicted by our model, $E_{align}$ be the set of aligned fields via Hungarian algorithm, and $E_{correct}$ be the set of correct matches through multi-strategy field matching. An entity field $e \in E_{pred}$ is considered correct and included in $E_{correct}$ if and only if it is first successfully aligned with a ground-truth entity via the Hungarian algorithm (*i.e.*, in $E_{align}$) and subsequently passes our multi-strategy field matching criteria.

Based on these definitions, we compute the following metrics:

- **Precision** measures the proportion of correctly extracted entity fields among all fields predicted by the model.

$$\text{Precision} = \frac{|E_{correct}|}{|E_{pred}|} \tag{1}$$

- **Recall** quantifies the proportion of ground-truth entity fields that are correctly extracted by the model.

$$\text{Recall} = \frac{|E_{correct}|}{|E_{gt}|} \tag{2}$$

- **F1-Score** is the harmonic mean of Precision and Recall, providing a single, balanced measure of overall performance.

$$\text{F1-Score} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \tag{3}$$

- **Accuracy** measures the fraction of correctly extracted entity fields among all aligned fields. This metric helps us distinguish errors caused by the alignment algorithm from those caused by

Example Instruction-Based SFT Sample for Basic Information Extraction

**Instruction:** You are a professional resume analysis assistant. Your task is to convert the given resume text into the JSON format specified below. (If both Chinese and English resumes appear, only extract from the Chinese one.)
Extract the following information into a JSON object. If any field does not exist, output an empty string "".

```
{
  "basicInfo": {
    "name": "",            # Name, e.g.: Zhang San
    "personalEmail": "",   # Email, e.g.: 610730297@qq.com
    "phoneNumber": "",     # Phone/Mobile number, preserve original format, e.g., "+1 (201) 706 1136"
    "age": "",             # Current age
    "born": "",            # Birth year (and month if available), e.g.: 1996-11
    "gender": "",          # Male/Female. Leave empty if not present.
    "desiredLocation": ["city name", ...],
                           # Expected job location(s), e.g., ["Beijing", "Shanghai"]
                           # Must be explicitly mentioned. If not, set to [].
    "currentLocation": "", # Current city of residence. Must be explicitly mentioned.
    "placeOfOrigin": ""    # Place of origin. Do not confuse with current location.
  }
}
```

/no_think

**Input (Indexed Resume Text):**
[0]: Gu Dabai [1]: Phone: 13987898888 [2]: Email: 123245677@123.com [3]: Job Objective: New Media Operations [4]: Education [5]: 2012.09—2016.06 Hebei University of Science and Technology, School of Business, Major: Information Management and Information Systems, Bachelor's degree [6]: Core Courses: Principles of Management, Macroeconomics, Data Structures, Database Systems, Intro to Software Engineering, CAD [7]: Internet Business Intelligence [8]: Work Experience [9]: 2016.07—Present Rainbow Network Co., New Media Operations Director [10]: Responsible for managing multiple social media accounts (Weibo, WeChat, Toutiao) [11]: Led online/offline brand campaigns and media exposure strategies [12]: Organized celebrity livestreams, reaching 2+ billion views; [13]: Gained 500K new followers annually per account [14]: 2013.09—2015.06 Haichao Consulting Co., Senior Consultant [15]: Assisted in public relations and brand image development [16]: Participated in project management and business analysis; [17]: Built client profiles and data processing tools [18]: Skills & Certifications [19]: TEM-8 (Advanced English), experience with Facebook, LinkedIn [20]: National Level 5 JAVA Certificate [21]: Self-Evaluation [22]: Creative, passionate, and curious thinker [23]: Keen on digital and online marketing trends [24]: Highly organized and responsible [25]: Excellent in teamwork and independent execution

**Output:**

```
{
  "basicInfo": {
    "name": "Gu Dabai",
    "phoneNumber": "13987898888",
    "personalEmail": "123245677@123.com",
    "age": "",
    "born": "",
    "gender": "",
    "desiredLocation": [],
    "currentLocation": "",
    "placeOfOrigin": ""
  }
}
```

**Figure 7: An Instruction-based SFT Sample for Extracting Basic Personal Information.**

our field-matching rules.

$$\text{Accuracy} = \frac{|E_{correct}|}{|E_{align}|} \tag{4}$$

## F Finer-Grained Comparison of Different Models

In addition to the overall performance averaged across all resume fields, we also conduct finer-grained accuracy comparisons across different field groups in SynthResume and RealResume datasets. The complete results are illustrated in Table 5 and Table 6 respectively.

**Table 5: Fine-grained Performance Comparison of Different model on SynthResume Dataset.**

| | Period | | | | Named Entity | | | | Long Text | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **Model** | **Acc.** | **Prec.** | **Recall** | **F1** | **Acc.** | **Prec.** | **Recall** | **F1** | **Acc.** | **Prec.** | **Recall** | **F1** |
| *Non-LLM Baselines* | | | | | | | | | | | | |
| Bello | 0.852 | 0.922 | 0.852 | 0.885 | 0.841 | 0.781 | 0.720 | 0.749 | 0.469 | 0.246 | 0.273 | 0.259 |
| PaddleNLP | 0.433 | 0.723 | 0.407 | 0.511 | 0.818 | 0.727 | 0.679 | 0.699 | 0.272 | – | – | – |
| *OCR + LLM* | | | | | | | | | | | | |
| Claude-4 | 0.977 | 0.984 | 0.975 | 0.980 | 0.939 | 0.955 | 0.951 | 0.951 | 0.731 | 0.634 | 0.741 | 0.684 |
| *Our Pipeline (LLM)* | | | | | | | | | | | | |
| Claude-4 | 0.960 | 0.983 | 0.963 | 0.973 | 0.963 | 0.964 | 0.949 | 0.956 | 0.825 | 0.786 | 0.816 | 0.801 |
| Gemini2.5-flash | 0.963 | 0.983 | 0.967 | 0.975 | 0.959 | 0.972 | 0.944 | 0.957 | 0.835 | 0.823 | 0.842 | 0.831 |
| GPT-4o | 0.960 | 0.987 | 0.961 | 0.974 | 0.966 | 0.968 | 0.955 | 0.961 | 0.840 | 0.819 | 0.840 | 0.829 |
| Deepseek-v3 | 0.961 | 0.978 | 0.965 | 0.971 | 0.968 | 0.976 | 0.951 | 0.962 | 0.829 | 0.834 | 0.794 | 0.813 |
| Qwen-max | 0.942 | 0.948 | 0.943 | 0.945 | 0.979 | 0.969 | 0.970 | 0.970 | 0.824 | 0.798 | 0.826 | 0.811 |
| Qwen3-14B | 0.888 | 0.892 | 0.913 | 0.902 | 0.961 | 0.955 | 0.954 | 0.954 | 0.698 | 0.664 | 0.665 | 0.664 |
| Qwen3-4B | 0.893 | 0.872 | 0.907 | 0.889 | 0.955 | 0.949 | 0.956 | 0.953 | 0.513 | 0.503 | 0.554 | 0.527 |
| Qwen3-0.6B | 0.619 | 0.682 | 0.660 | 0.668 | 0.637 | 0.749 | 0.734 | 0.691 | 0.256 | 0.162 | 0.226 | 0.184 |
| *Our Pipeline (SFT)* | | | | | | | | | | | | |
| Qwen3-0.6B-sft | 0.897 | 0.908 | 0.907 | 0.907 | 0.951 | 0.941 | 0.936 | 0.938 | 0.858 | 0.760 | 0.777 | 0.767 |

**Table 6: Fine-grained Performance comparison of Different model on RealResume Dataset.**

| | Period | | | | Named Entity | | | | Long Text | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **Model** | **Acc.** | **Prec.** | **Recall** | **F1** | **Acc.** | **Prec.** | **Recall** | **F1** | **Acc.** | **Prec.** | **Recall** | **F1** |
| *Non-LLM Baselines* | | | | | | | | | | | | |
| Bello | 0.921 | 0.968 | 0.813 | 0.879 | 0.885 | 0.801 | 0.740 | 0.769 | 0.540 | 0.553 | 0.459 | 0.500 |
| PaddleNLP | 0.387 | 0.587 | 0.381 | 0.451 | 0.722 | 0.653 | 0.597 | 0.622 | – | – | – | – |
| *OCR + LLM* | | | | | | | | | | | | |
| Claude-4 | 0.979 | 0.987 | 0.984 | 0.986 | 0.937 | 0.958 | 0.960 | 0.959 | 0.582 | 0.512 | 0.598 | 0.548 |
| *Our Pipeline (LLM)* | | | | | | | | | | | | |
| Claude-4 | 0.963 | 0.985 | 0.960 | 0.972 | 0.964 | 0.924 | 0.980 | 0.949 | 0.869 | 0.819 | 0.899 | 0.854 |
| Gemini2.5-flash | 0.970 | 0.984 | 0.973 | 0.978 | 0.945 | 0.898 | 0.974 | 0.931 | 0.888 | 0.851 | 0.880 | 0.865 |
| GPT-4o | 0.963 | 0.982 | 0.962 | 0.972 | 0.950 | 0.914 | 0.973 | 0.941 | 0.880 | 0.848 | 0.899 | 0.870 |
| Deepsek-v3 | 0.960 | 0.987 | 0.957 | 0.971 | 0.939 | 0.903 | 0.940 | 0.918 | 0.867 | 0.842 | 0.865 | 0.852 |
| Qwen-max | 0.971 | 0.989 | 0.968 | 0.978 | 0.936 | 0.895 | 0.941 | 0.915 | 0.827 | 0.810 | 0.832 | 0.820 |
| Qwen3-14B | 0.963 | 0.982 | 0.963 | 0.972 | 0.939 | 0.871 | 0.935 | 0.899 | 0.724 | 0.699 | 0.717 | 0.707 |
| Qwen3-4B | 0.901 | 0.902 | 0.963 | 0.930 | 0.921 | 0.849 | 0.936 | 0.886 | 0.579 | 0.533 | 0.612 | 0.567 |
| Qwen3-0.6B | 0.680 | 0.750 | 0.724 | 0.734 | 0.647 | 0.683 | 0.697 | 0.671 | 0.120 | 0.126 | 0.182 | 0.136 |
| *Our Pipeline (SFT)* | | | | | | | | | | | | |
| Qwen3-0.6B-sft | 0.956 | 0.976 | 0.951 | 0.963 | 0.953 | 0.909 | 0.962 | 0.932 | 0.866 | 0.807 | 0.874 | 0.838 |