LLM Resume Parser

Piyawut Pattamanon

Short-term Strategy

Constraints

- Only 5 hours to prepare
- Should process millions of files per month
    - avg 4 resume per second

Strategy

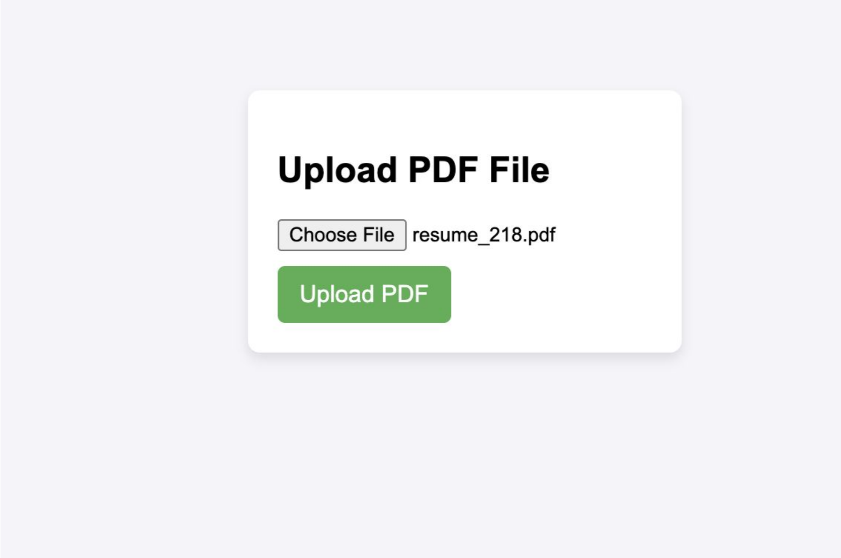
- Avoid big dataset procurement
- Minimize custom model training

Tactic

- Use ready-to-use foundation LLM
- Some prompt refinement
- Asynchronous + Parallel Processing for high throughput

Demo

The PDF Upload Form



The API Result

NLP Technologies

Pre Deep Learning Techniques
- RegEx, TF/IDF, Rule based

Pre Embedding Deep Learning
- LSTM, CNN

Word Vector (Last 8 years)

Transformer (Last 5 years)

LLM (Last 12 months) <-- State of The Art

<u>Using LLM is NOT simple</u>
- Slow
- Expensive
- Doesn't follow instructions
- Indeterministic

Yeah, not simple but can be managed

Measure of Success

Content Correctness
- Exact Matching
    - Recall, Precision, F-Measure
    - Separate by columns (e.g. job title, school, company name)
- Fuzzy Matching
    - Embedding Cosine Similarity

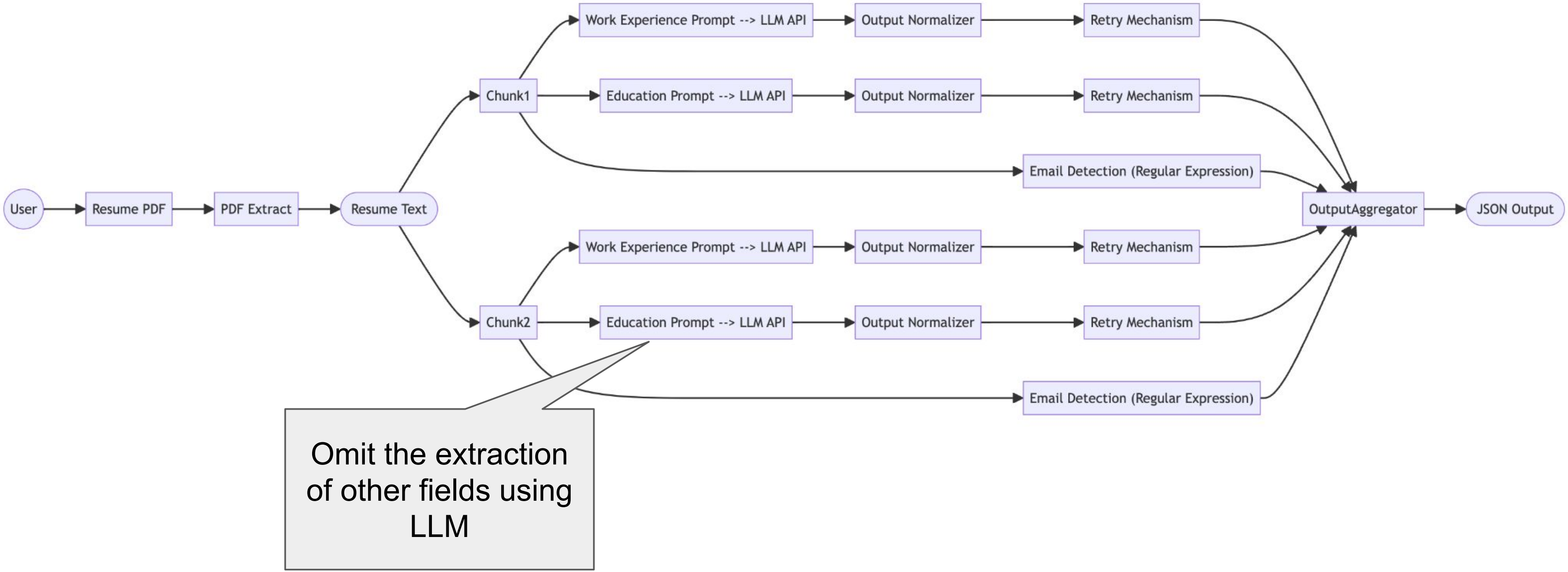
Answer Reliability
- Error Rate
- Retry Count

Schema Correctness
- Accuracy

System
- Throughput (requests / second)
- Latency
- Cost
    - Token Count
    - CPU/GPU utitilization
    - Memory consumption

User
- Net Promoter Score (NPS)

Single Prediction Pipeline



Pipeline Design Principles

Avoid LLM Cognitive Overload

- By splitting into sub-tasks

Handle LLM Indeterminism

- By Retry Mechanism

Parallel Processing When Possible

Handle limited LLM input size

- By splitting into chunks then aggregate

Adapt Task Complexity vs Model Complexity

- Use simple models for simple tasks
    - e.g. just RegEx for e-mail detection

Dataset

<u>Now</u>

- Ready-to-use public dataset
- Kaggle's Resume Entities for NER
- Source: Indeed.com in India
- Fully annotated (good)
- 220 resumes (small -- not good)

<u>Long Term</u>

- Manatal strength? Large amount of resumes
- Ask for consent before collection
- Mask Personal Data
- Annotation Process
    - LLM
    - Human confirm

Prompt Fine Tuning ex. 1 (AFTER)

Prompt (AFTER)

you are resume parser.
extract his job positions and these attributes related to the position:
job title, company, end_date, start_date, city, country_code.

no more attributes other than this.
output in json.

example output

```
[
    {
      "city": "",
      "job_title": "NCAT ASEC",
      "country": "",
      "company": "Hickman Property Holdings LLC",
      "end_date": "2023-",
      "start_date": "2022-2",
      "country_code": "IN"
    }
]
```

if there is no job position in his resume. output a blank array like this
[ ]

Extracted Result (AFTER)

```
[
    {
        "city": "Hyderabad",
        "country_code": "IN",
        "company": "Infosys",
        "end_date": "",
        "job_title": "Security Analyst",
        "start_date": "2015-03"
    }
]
```

Prompting Strategy
- Give context
- Give examples (Few-shot learning)
- Reduce ambiguity in wording
- Add negative command if bad things happen

Prompt Fine Tuning ex. 1 (BEFORE)

Prompt (BEFORE)
you are resume parser.
extract his job positions and information related to each position.
output in json

example output

```
[
    {
      "city": "",
      "title": "NCAT ASEC",
      "country": "",
      "employer": "Hickman Property Holdings LLC",
      "end_date": "2023-",
      "start_date": "2022-2",
      "country_code": ""
    }
]
```

if there is no job position in his resume. output a blank array like this
[ ]

Extracted Result (BEFORE)

```
[
    {
        "city": "Hyderabad",
        "country_code": "IN",
        "title": "Security Analyst in Infosys",
        "employer": "",
        "start_date": "March 2015",
        "end_date": "",
        "skills": [...],
        "responsibility":[...],
    }
]
```

Prompt Fine Tuning ex. 2

Text

* Sincere and Hardworking in nature
* Highly Dedicated towards work

* Efficient Individual and Team Player
* Goal Oriented & Self Motivated

IT Literacy

A lot missing

Prompt (BEFORE)

you are resume parser.
extract his skills as specified in his resume.
output in json

example output

```
[
    {"name": "Community Service"},
    {"name": "Concord"}
]
```

if there is no skills in his resume, output a blank array like this
[ ]

Extracted Result (BEFORE)

```
[
  {"name": "IT Literacy"}
]
```

Prompt (AFTER)

you are resume parser.
extract his skills as specified in his resume.
both hard and soft skills.
output in json

example output

```
[
    {"name": "Community Service"},
    {"name": "Concord"}
]
```

if there is no skills in his resume, output a blank array like this
[ ]

Extracted Result (AFTER)

More complete

```
[
  {"name": "IT Literacy"},
  {"name": "Efficient Individual and Team Player"},
  {"name": "Goal Oriented & Self Motivated"},
  {"name": "Sincere and Hardworking in nature"},
  {"name": "Highly Dedicated towards work"}
]
```

Speed

<u>Measured</u>

Response Time: 6-12 seconds

(Model: llama-3-sonar-large-32k-online)

<u>Note</u>

- Use evaluation metrics to guide what model we can use (accuracy vs speed trade-off)

<u>Info</u>

- Avg. Resume Size: 1000 tokens
- Avg. Prompt Size (Input+Output): 2000 tokens

<u>Proactive Estimation</u>

Model	Tokens per Second	Time to Process 2000 Tokens (s)
Claude 3 Opus	28	71.43
Claude 3 Sonnet	64	31.25
Claude 3 Haiku	21,000	0.095
GPT-3.5 Turbo	57.4	34.84
GPT-4 Turbo	18.1	110.50
LLAMA 3 7B (M1 Processor)	16.2	123.45

Speed-Accuracy Tradeoff

- Simpler model = faster = less accurate

- Find the balance

    - Use evaluation + system metrics to guide

    - Use simplest model with acceptable accuracy

<u>Alternatives</u>

- Smaller Pre-Trained LLM (low effort)

- Custom Smaller LLM (very high effort)

    - Distill Knowledge from Bigger LLM

    - Fine tuned for Resume Parsing task

- Custom Tensorflow Model + BERT Transfer Learning (high effort)

- RegEx (medium effort)

Scalability: Serving Millions per Month

Main Strategy: Horizontal Scaling

Use Case: Near Real Time API
Lambda / Kubernetes -> AWS Bedrock

Use Case: Batch Processing
SQS -> Lambda / Kubernetes -> AWS Bedrock

Lambda vs Kubernete?
- Lambda if team is small + no infra team support
- Kuberbetes if infra team is matured
    - EC2 discounts are available
    - Avoid vendor locking

Quality Assurance

- Automated Evaluation Dataset Quality
    - Completeness
    - Consistency
- Automated Model Evaluation
- Automated Unit/Functional Testing
- System Load Testing
- Code Quality
    - Code Review
    - Automated Syntactic Check (flake8)
    - Automated Semantic Check (SonarQube)

Further Improvement

LLM Prompt Continuous Improvement

-> Learn from bad predictions

-> Hypothesize

-> Apply Improvement

-> Evaluate Impact

-> Repeat

Might try **Automated** Prompt Finetuning technique by LLM

Experiment Different Foundation Models

- Different LLM

- Different non-LLM techniques

LLM becomes 100X Cheaper + Faster

- Paper from MicroSoft end of Feb 2024
- Quantize until 1-bit
- No more expensive matrix multiplication
    - Only cheap addition
- Might be usable in 1-2 years time

The Era of 1-bit LLMs:
All Large Language Models are in 1.58 Bits

Shuming Ma* Hongyu Wang* Lingxiao Ma Lei Wang Wenhui Wang
Shaohan Huang Li Dong Ruiping Wang Jilong Xue Furu Wei◇
https://aka.ms/GeneralAI