Structured Information Extraction from Job Descriptions using Transformers

Xinyu Lan
College of Computing: Computer Science
Illinois Institute of Technology
Chicago, USA
xlan7@hawk.iit.edu

Pranav Kuchibhotla
College of Computing: Artificial Intelligence
Illinois Institute of Technology
Chicago, USA
pkuchibhotla@hawk.iit.edu

Krystian Szczepankiewicz
College of Computing: Computer Science
Illinois Institute of Technology
Chicago, USA
kszczepankiewicz@hawk.iit.edu

Mingkai Hsu
College of Computing: Computer Science
Illinois Institute of Technology
Chicago, USA
mkai@hawk.iit.edu

Melissa Mey Y. Laiz
College of Computing: Computer Science
Chicago, USA
mlaiz@hawk.iit.edu

***Abstract*—We scraped and auto-labeled ≈11 k job postings, then trained three memory-aware Transformer setups : FLAN-T5-Large (full fine-tune), FLAN-T5-XL + LoRA, and Mistral-7B + LoRA, each further compressed with 8-bit quantization. Evaluation on JSON parse rate and field-level F1 shows FLAN-T5-Large delivers the best overall extraction fidelity, while Mistral-7B edges ahead on semantically tricky fields such as salary normalization. The study confirms that even single GPU hardware can support accurate, large-scale job-data structuring, provided one balances model size, LoRA adapters, and quantization overhead.

Keywords—job postings, information extraction, transformer models, text structuring, FLAN-T5, Mistral, fine-tuning

I. Introduction And Project Goals

Job hunting can be stressful nowadays. Even with platforms like LinkedIn and Indeed, it's hard to find roles that are a strong match. A big part of this problem lies in the way job descriptions are written. Most postings are just big blocks of text with no standard format, which makes it tough for search engines to understand what the job is really about. Ultimately, candidates see postings that don't really match their background or expertise which leads to wasted time and missed opportunities.

We wanted to explore how artificial intelligence, particularly natural language processing, could be useful in this case. Our idea was to build a system that could read unstructured job descriptions and pull out key information such as job title, required skills, experience, location, main responsibilities and then convert it into a structured format. It'll make it much easier to filter, search and rank job postings in a way that's actually useful to someone looking for a job. To do this, we used Transformer-based models which is an advanced class of NLP models that have been very successful in understanding the structure and meaning of human language. These models are good at picking up context, which is important when working with job descriptions.

The goals of our project was fairly straightforward but required a lot of experimentation to get it right:

- Build a model that can take raw job descriptions and return structured data for key job fields.
- Make the extraction process as accurate and generalizable as possible, even across different industries or writing styles.
- Demonstrate how structured job data can improve job searching—for example, by enabling smarter search filters or personalized job recommendations.

We want to show that it is possible to improve the job search experience using modern machine learning tools. This project is a step toward making job platforms more helpful and user-focused, making the job hunt a little bit easier.

II. Related Work

In the past, traditional models like CRF were used to convert unstructured data to structured data. Nowadays, transformer-based models have become more popular.

There are several studies related to information extraction. Townsend et al.[12] proposed DOC2DICT, which can generate the structured data in JSON format. Paolini et al. [13] proposed the TANL framework, treating predicting structured output as a translation task. Lu et al. [14] proposed Text2Event, which can directly extract events from the text in an end-to-end manner. Later, the same authors[15] proposed UIE, which can universally model different information extraction tasks, adaptively generate targeted structures.

Transformer-based models such as T5[16] and Mistral[17] have been widely used for the generation task. T5 uses an encoder-decoder structure, while Mistral is a decoder-only model. In this project, we focus on comparing those 2 kinds of structures.

When training LLM, GPU resource limitation is a common problem. To train these LLMs on limited hardware, many studies now use parameter-efficient fine-tuning methods like LoRA[9], which only a small portion of model weights.

III. Data Gathering and Structuring

To train an efficient transformer model skilled at extracting structured data from unstructured job postings, we then needed to produce a dataset marked by diversity and high quality. Our approach combined real-world job data harvested from online platforms with publicly available datasets. Thus, this approach allowed us to introduce the requisite variation and volume to ensure that the model would be effective at generalizing across industries, functions, and conventions of formatting.

A. Web Scraping and Dataset Collection

At first, we collected more than 7,000 job postings by utilizing a combination of web scraping methods and existing datasets:

- **Unofficial LinkedIn API** [1], this permitted the scraping of LinkedIn job postings with keyword searches such as “software engineer.”
- **LinkedIn Scraper** [2], this tool allowed us to gather more job metadata like full descriptions, corporate classifications and listed competencies.
- **Public Kaggle Dataset** [3], served as a useful reference point for checking our extraction processes and assessing model performance.

Therefore, to ensure that the model would be applicable across many different fields, it then configured the LinkedIn scrapers to pull listings from several different industries, but we specifically focused on tech. Due to the presence of rate limits and anti-bot protection on second-verification sites like LinkedIn, they developed a mechanism that increases the throughput while minimizing API timeouts. Thus, doing so, it was able to get around the said limitations and was able to at least gather 11,067 unique job listings as of the last phase. These listings constituted the base dataset for the training and testing component of the model

B. Storage Architecture and Data Structuring

All data, when gathered, was kept in a MongoDB database. Then MongoDB because it accommodated JSON-like structures and had seamless compatibility with Python, which was utilized in our pipeline end-to-end. Then organized the database into three tiers to keep the workflow tidy and modular:

- **Raw Data Layer**: Contains unprocessed job entries, including IDs, description titles, and associated metadata.
- **Cleaned Data Layer**: Contains pre-processed entries, optimized for token efficiency through filtering redundant fields and noisy data prior to labeling.
- **Labeled Data Layer**: This stores structured JSON outputs generated by the DeepSeek API, ready for ingestion by models like FLAN-T5 and Mistral-7B [4].

The layered architecture can trace the flow of the data from a raw input to a processed and annotated output, while also aiding the debugging and versioning processes. Overall, the combination of scraping, public datasets, and organized storage served as a solid foundation for model training.

IV. Dataset Refinement and Structuring

Prior to training the transformer model, it had to convert dirty job descriptions into clean, formatted data. Job listings are all over the map in terms of style and content. To make the data usable, then creating an efficient pipeline for cleaning, labeling, and formatting.

A. Cleaning Raw Job Data

The initial step involved cleaning of the raw job descriptions collected from LinkedIn and other sources. Thus, using MongoDB pipelines as well as Python scripts [7]:

- Remove HTML tags, emojis, and special characters
- Stripped redundant fields and reduced token count
- Normalized whitespace and punctuation for consistency

Overall, these steps helped reduce the noise and ensured that all input was focused on significant job-related data like responsibilities, qualifications, and skills

B. Labeling with DeepSeek API

DeepSeek-V3 API enabled auto-label job posts. Instead of manually tagging a multitude of postings, composing a detailed prompt [5]. These custom prompts instructing the DeepSeek transformer to extract job-relevant fields, such as:

- job_title
- location
- experience
- skills
- responsibilities

An example prompt looked like this:

You are a helpful AI trained to label job posting data. Label the following job posting with only JSON format output as per the example. If no information is found, leave the field blank. Do not add explanation.

Raw job data: {job_description}

C. Preprocessing for Model Input

To prepare the data for the transformer model FLAN-T5-XL from Hugging Face, formatted each sample as [8]:

{ "source": "raw_description", "target": "structured_json}

This format allowed direct input to the sequence-to-sequence pipeline of the model. For better training quality and to prevent runtime errors, thus incorporating a few more preprocessing steps:

- Validate the JSON structure
- Manage the input length to stay within the token limits
- Applied light stopword removal for efficiency

These steps made the dataset structured and clean, but also tuned in the model's architecture and constraints – laying the perfect groundwork for quality fine-tuning and robust structured extraction.

V. Implementation

To transform unstructured job postings into a structured JSON schema, model selection balanced architectural suitability with hardware constraints (primarily **single NVIDIA RTX 6000 GPUs with 24GB VRAM**). After extensive evaluation, we selected transformer architectures based on their suitability for structured text generation:

A. Encoder-Decoder Models

- FLAN-T5-Large (783M parameters): Fully fine-tuned end-to-end, leveraging effective copy mechanisms particularly beneficial for extracting exact spans like "programming languages"and "skills".
- FLAN-T5-XL (2.85B parameters): Implemented with parameter-efficient fine-tuning to fit within GPU constraints while accessing potentially richer semantic representations

B. Decoder-Only Model

Mistral-7B (7.25B parameters)[8]: Selected as a comparative approach for its reasoning capabilities and instruction-following abilities

Each model had their own advantages. The T5 models work well at structured text generation tasks thanks to their encoder-decoder setup, while Mistral's 7B parameters seems better at semantic understanding for fields where we need interpretation rather than just copying text chunks verbatim from job descriptions

The training methodology followed a consistent approach across all models, using identical data splits from approximately 11,000 labeled job postings (90% training, 10% validation) .For single-GPU training, we adopted gradient accumulation, extending up to 16 steps to simulate larger batch sizes, whereas multi-node experiments with Mistral relied on a hostfile-based setup using pdsh in conjunction with DeepSpeed for efficient distributed training. Cross-entropy loss was used, alongside debugging measures like loss monitoring and gradient norm tracking.

C. Training Procedures and Implementation Details

Given the limitations of GPU memory, optimizing both efficiency and scalability became a core priority throughout our training pipeline. To address this, we used several strategies to reduce memory usage. Moreover, we made some adjustments for specific models and improved the system setup to better support training.

Memory Optimization Strategies:

- **8-bit quantization**: Used INT8 precision (threshold=6.0) to reduce the overall memory footprint. This enabled larger model architectures to run within tight GPU constraints
- **Low-Rank Adaptation (LoRA)[9]**: Implemented with r=8 and alpha=16, targeting query and value projection matrices. This approach reduced trainable parameters to <1% while preserving performance capabilities. FLAN-T5-XL with LoRA used only 47% GPU memory at batch size=2, compared to 97% for the smaller T5-Large without LoRA
- **Mixed precision training**: Chose BF16 for T5-XL and Mistral models.
- **Memory-efficient optimizers**: Used Paged AdamW [10] 8-bit for LoRA models and Adafactor [11] for fully fine-tuned T5-Large

VI. Evaluation Framework And Metrics

To assess how well our models extract structured data from unstructured job postings, we designed a layered evaluation approach. It focuses on two key aspects: how accurate the model outputs are, and how consistently the models try to fill in every required field.

A. Evaluation Dataset and Setup

We used a held-out test set of 1,000 job descriptions, which were labeled using the DeepSeek-V3 API. These outputs served as the ground truth references for comparing model predictions. To ensure consistency, we used the same evaluation dataset across all model runs.

Before evaluating the models, we ensured each output was a valid JSON. If a model produced an invalid JSON, our cleanup function corrected it automatically. This helped us avoid parse errors and ensured fair and consistent evaluation. This step ensured consistency in scoring and prevented parse errors from misrepresenting metrics.

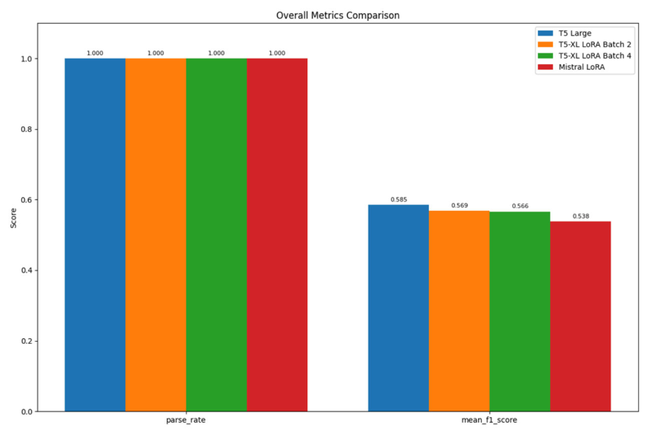

Figure 1: Overall parse rate and mean F1-score comparison across all evaluated models. All models achieved a perfect parse rate (1.0), with FLAN-T5-Large achieving the highest mean F1-score.

B. Metrics used

We used the following metrics to evaluate each model:

- **Parse Rate:** The percentage of outputs that were valid JSON. All evaluated models achieved a 100% parse rate, due to the effectiveness of the cleanup function.
- **Presence Rate:** For each field, we measured how often the model attempted to populate it. A high presence rate suggests that the model consistently recognized the field and made an extraction attempt, regardless of correctness.
- **Field-Level Accuracy:** For single-value fields like "experience_level", "work_location", "required_experience", and "salary_min", we used exact string match accuracy to check if the predicted value matched the ground truth.
- **List Field Metrics (Precision, Recall, F1 Score):** For list-style fields such as skills, industries, and job functions, we used token-level evaluation.
  - *Precision*: How many of the predicted items were correct?
  - *Recall*: How many correct items did the model successfully predict?
  - *F1 Score*: A balance between precision and recall.
- **Mean F1 Score:** We averaged the F1 scores across all list fields to obtain an overall performance score for each model.

All evaluation scripts were implemented in Python using sklearn and the evaluate library, ensuring reproducibility and consistency across experimental runs.

VII. Detailed Model Performance Results

We conducted a systematic comparison of four transformer model configurations: FLAN-T5-Large (783M parameters, full fine-tuning), FLAN-T5-XL with LoRA (batch sizes 2 and 4), and Mistral-7B with LoRA (7.25B parameters, decoder-only). The models were evaluated on their ability to extract structured JSON representations from job descriptions, under a consistent setup.

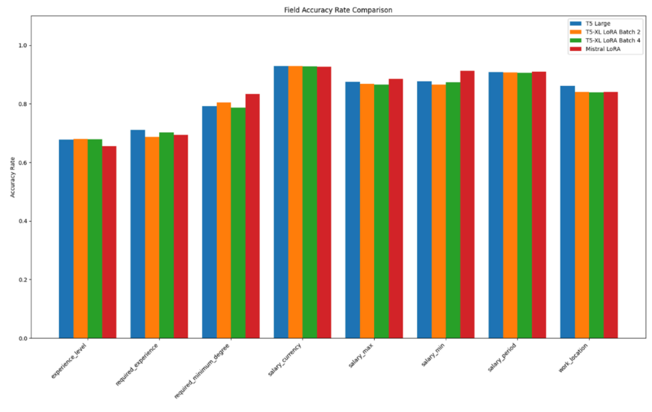

Figure 2: Field-level accuracy comparison on key single-value attributes such as "salary_min", "required_experience", and "work_location". Mistral LoRA shows stronger performance in semantically inferred fields, while T5-Large is better in direct extraction fields.

A. Summary of Results

Model	**Mean F1 Score**	**Parse Rate**	**Training Time**	**Notable Strengths**
FLAN-T5-Large	0.585	100%	7.3 hours	Highest overall F1, stable training
FLAN-T5-XL LoRA (batch=2)	0.569	100%	22.2 hours	Efficient GPU usage, good generalization
FLAN-T5-XL LoRA (batch=4)	0.566	100%	21.4 hours	Maintains performance at larger batch sizes
Mistral-7B LoRA	0.538	100%	56 hours	Strong in semantically

				y complex fields

B. Model Behaviour and Key Insights

FLAN-T5-Large consistently outperformed all other configurations in terms of mean F1 score and field-level accuracy. It was particularly strong results on structured fields like “skills_programming_languages” and “benefits”, where precise extraction of known entity types is important. Its strong performance is likely due to its encoder-decoder setup, which helps us with accurately copying spans.

FLAN-T5-XL with LoRA demonstrated very good performance while reducing memory usage by a lot. At a batch size of 2, it only consumed 47% of GPU memory, in contrast to the 97% used by FLAN-T5-Large. This efficiency comes from Low-Rank Adaptation, which significantly cuts down the number of trainable parameters while maintaining solid generalization. Although it was slightly lower in F1 score, FLAN-T5-XL matched or even exceeded T5-Large in specific multi-label fields like “skills_tools” and “job_functions”.

Mistral-7B with LoRA, the largest model tested, had lower overall F1 scores, but stood out in areas that required interpreting and restructuring vague information. It showed high accuracy in fields like “required_minimum_degree” and “salary_min”, where the model needed to normalize paraphrased phrases or numeric ranges (e.g., "Bachelor’s or Master’s in CS", "$250–350k"). This aligns with its decoder-only, instruction-tuned architecture, which is built to handle semantic reasoning and inference tasks well.

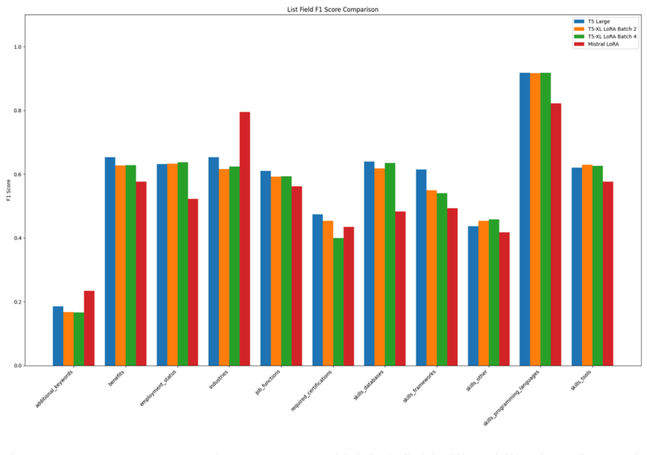

Figure 3: F1-score comparison across multi-label fields like skills, benefits, and industries. T5 models show consistent performance, while Mistral LoRA excels in complex semantic fields like industries.

C. Field Presence Rates and Scheman Alignment

We noticed that the T5-based models, especially FLAN-T5-Large and FLAN-T5-XL, consistently produced high presence rates (≥ 93%) across all key fields. This shows they are well-attuned to the structure of the data and reliably detect the fields they need to extract.

On the other hand, Mistral-7B showed lower presence rates for some fields like “experience_level” (84.1%) and “work_location” (88.1%). This suggests that it tends to skip uncertain fields unless it’s highly confident in its prediction. While this helps reduce the chance of incorrect entries, it may lead to missing data in cases where full coverage is needed.

D. Practical Implications and Conclusion

All models returned a 100% parse rate after our cleanup process, confirming the effectiveness of our JSON cleanup function. This is critical for maintaining pipeline stability and ensuring that downstream systems receive consistently formatted data.

Based on overall performance, FLAN-T5-Large appears to be the best option when high recall and structured output are priorities, especially when hardware resources are limited. It offers the most balanced combination of accuracy, efficiency, and stability. However, Mistral-7B demonstrates meaningful advantages in domains where inference, paraphrasing, or numeric translation is required.

VIII. Results and Conclusions

Under the constraint of a single RTX 6000 (24 GB VRAM), our comparative evaluation produced several findings. After uniform post‑processing with a bespoke JSON‑sanitization routine, every model consistently attained a 1.00 parse success rate, confirming baseline stability.

Among the candidates, FLAN‑T5‑Large, despite its modest 783 M parameters, demonstrated the highest mean F1‑score and the most favorable time‑to‑solution, advantages that we attribute to its encoder‑decoder architecture’s facility for direct span extraction. In contrast, Mistral‑7B, particularly when paired with parameter‑efficient LoRA adapters, excelled on semantically demanding tasks such as salary range normalization and degree‑level inference, indicating that its larger decoder capacity retains value where deeper reasoning is required. LoRA‑adapted T5‑XL offered an intermediate profile, lower memory usage than full fine‑tuning but without fully matching the precision of FLAN‑T5‑Large on routine fields.

Collectively, the results affirm that reliable, large‑scale structuring of job posting data is attainable on limited hardware, provided one balances model selection (e.g., FLAN‑T5‑Large for general extraction fidelity) with lightweight adaptation strategies, while reserving models such as Mistral‑7B for fields whose semantics demand greater interpretive depth.

Acknowledgment

The authors wish to thank Professor Shouvik Roy for his guidance and essential assistance with securing access to the Chameleon Cloud. We gratefully acknowledge the Chameleon Cloud project for providing the computational resources used in this research.

References

[1] Tom Quirk. *Unofficial LinkedIn API* [GitHub Repository]. https://github.com/tomquirk/linkedin-api

[2] Joeyism. *LinkedIn Scraper Tool* [GitHub Repository]. https://github.com/joeyism/linkedin_scraper

[3] Arshkon. *LinkedIn Job Postings Dataset* [Kaggle Dataset]. https://www.kaggle.com/datasets/arshkon/linkedin-job-postings

[4] Hugging Face. *FLAN-T5-XL Model* [Model Card]. https://huggingface.co/google/flan-t5-xl

[5] DeepSeek-V3 API. https://www.deepseek.com

[6] Hugging Face – FLAN-T5-XL. https://huggingface.co/google/flan-t5-xl

[7] MongoDB Aggregation Docs. https://www.mongodb.com/docs/manual/core/aggregation-pipeline/

[8] Hugging Face-Mistral-7B-v0.3 [Model Card] https://huggingface.co/mistralai/Mistral-7B-v0.3

[9] Edward J. Hu, et al. LoRA: Low-Rank Adaptation of Large Language Models [arXiv Paper]. https://arxiv.org/abs/2106.09685

[10] I. Loshchilov & F. Hutter. Decoupled Weight Decay Regularization [arXiv Paper] https://arxiv.org/abs/1711.05101

[11] N. Shazeer & M. Stern. Adafactor: Adaptive Learning Rates with Sublinear Memory Cost [arXiv Paper]. https://arxiv.org/abs/1804.04235

[12] Townsend, B., Jain, S., Wang, A., Wallace, E., & Gardner, M. (2021). *Doc2Dict: Information extraction as text generation* *https://arxiv.org/abs/2105.07510*

[13] Paolini, G., Zheng, Y., Xu, Y., McDonald, R., Glass, M., Stenetorp, P., & Riedel, S. (2021). *Structured prediction as translation between augmented natural languages.* *https://openreview.net/forum?id=US-TP-xnXI*

[14] Lu, Y., Shen, S., Tan, Z., Liu, P., Qiu, X., & Huang, X. (2021). *Text2Event: Controllable sequence-to-structure generation for end-to-end event extraction.* *https://aclanthology.org/2021.acl-long.217/*

[15] Lu, Y., Shen, S., Tan, Z., Qiu, X., & Huang, X. (2022). Unified structure generation for universal information extraction. *https://aclanthology.org/2022.acl-long.395/*

[16] Raffel, C., Shazeer, N., Roberts, A., Lee, K., Narang, S., Matena, M., Zhou, Y., Li, W., & Liu, P. J. (2020). Exploring the limits of transfer learning with a unified text-to-text transformer. Journal of Machine Learning Research, 21(140), 1–67. *http://jmlr.org/papers/v21/20-074.html*

[17] Mistral AI. (2023). Mistral 7B (arXiv:2310.06825). arXiv. *https://doi.org/10.48550/arXiv.2310.06825*