

Evaluating Fine-Tuning for Information Extraction from SEEK Job Advertisements

Eamonn Lee (z5358883)

Rubik Liu (z5564105)

Yiming Zhou (z5462488)

Eddie Zhang (z5481990)

April 28, 2025

1 Introduction

The objective of this project is to explore the effectiveness of fine-tuning when using large language models (LLMs) for tasks. In this project two information extraction tasks were considered: salary extraction and work arrangement classification. These tasks were conducted on job advertisements provided by SEEK, an Australian job advertisement marketplace website. Specifically the project aims to determine whether performance is uplifted via fine-tuning with consideration of potential increased computational complexity and other variables.

The datasets which these models were trained on were provided by SEEK and include both labeled and unlabeled job advertisements provided under non-disclosure agreements. Two labeled datasets were used for each task - a development set and a test set. The unlabeled set was reserved for optional exploration.

The project structure involves building a baseline solution, a non-finetuned LLM solution and a fine-tuned LLM solution to compare the worth of finetuning with regards to certain evaluation metrics: precision, recall, F1 score and accuracy. Multiple models were investigated including RoBERTa, T5, mT5, OPT-350m, BART and Longformer.

Through experimentation and detailed quantitative evaluation, this report aims to deliver insights into the worth of finetuning in performing practical, real-world tasks.

2 Project Scope

Part	Points and Details
PART A: Problem Definition (Minimum: 10 credits)	<p>NLP Problem: 5 credits per problem</p> <p>Details: The project addresses the research question "Is fine-tuning worth it?". The project compared non-finetuned LLMs with fine-tuned LLMs for job advertisement information extraction tasks, evaluating whether fine-tuning improved performance metrics (precision, recall, F1 score, and accuracy). Two out of the three tasks — salary and work arrangements — were addressed. Obtaining the salary information was treated as a question-answering task, and obtaining the work arrangements information was treated as a textual classification task.</p> <p>Text Source/Domain: 5 credits per source/domain</p> <p>The project's text source was job advertisements obtained directly from SEEK, Australia's leading employment marketplace website. The datasets included labelled and unlabelled job advertisements specifically prepared for the project under NDA agreements, covering information extraction tasks regarding salary, work arrangements, and seniority.</p>
PART B: Dataset Selection (Minimum: 20 credits)	<p>Use two existing datasets: 10 credits</p> <p>Two provided datasets from SEEK were used: the labelled development and test sets for the information extractions of salary and work arrangements. These datasets were directly supplied by SEEK under NDA and were used to train and validate the models produced in the project.</p> <p>Create your own labelled dataset: 20 credits</p> <p>Two labelled datasets were created: 1900_labelled.json was created for RoBERTa salary model solutions, err_salary_development.csv was created for the mT5 salary model solutions</p>

Part	Points and Details
PART C: Modelling (Minimum: 30 credits)	<p>Baseline model: Essential Baseline approach models were implemented for both salary and work arrangements.</p> <p>Use a pre-trained/fine-tuned model: 5 credits per model (compare performance of multiple models) Multiple models were directly used, including:</p> <ul style="list-style-type: none"> • mT5 for the Salary dataset • T5 for the Salary dataset • RoBERTa for the Salary dataset • Facebook/OPT-350m for the Work Arrangements dataset • Facebook/bart-large-mnli for the Work Arrangements dataset <p>Fine-tune a model: 20 credits (e.g., fine-tuned BERT, prefix-tuned LLaMA) Multiple models were finetuned, including:</p> <ul style="list-style-type: none"> • mT5 for the Salary dataset • T5 for the Salary dataset • RoBERTa for the Salary dataset • Facebook/OPT-350m for the Work arrangements dataset with LORA • allenai/longformer-base-4096 for the Work arrangements dataset
PART D: Evaluation (Minimum: 20 credits)	<p>Quantitative Evaluation: 10 credits A set of standard metrics were used to evaluate the performance of solutions in both Salary and Work Arrangement Datasets. These metrics included precision, accuracy, F1 score and recall.</p> <p>Demo: 10 credits (simple demo via Gradio or equivalent) Gradio demo's were integrated into the Work Arrangements Zero-shot Text Classification Approach(<i>wa_tc_0shot.ipynb</i>). And, Gradio demo for Salary on link: https://huggingface.co/spaces/Garry0Zhou/Ad_toSalary_using_mt5</p>

3 Salary

3.1 Baseline Model

The baseline extractor begins by stripping all HTML tags to yield plain text, then tokenizes on whitespace and scans for salary cues (e.g., “salary,” “\$,” “¥”). Whenever a cue is found, it captures the next six tokens as a context window and applies a regex to extract one or two numeric values—rounding “k”-suffixes into full amounts and verifying that the lower bound does not exceed the upper.

It then attempts to identify explicit period indicators (“hourly,” “monthly,” etc.) in the same window; if none are present, it converts the average salary into an AUD-normalized figure via predefined exchange rates and normalization factors, using simple thresholds to infer whether the rate is hourly, daily, weekly, or monthly.

When multiple candidate ranges emerge, the method prioritizes explicitly flagged ranges and otherwise defaults to the first match; if no numbers are detected, it outputs “0-0-None-None.” This heuristic achieves the following results:

Dataset	Precision	Recall	F1 Score	Accuracy
Development	0.7811	0.9370	0.8519	0.8489
Test	0.7412	0.9206	0.8212	0.8219

Table 2: Model performance on development and test datasets.

3.2 Non-Finetuned RoBERTa Model

3.2.1 Architecture & Model

To extract the salary information from the job advertisements, we used a pretrained RoBERTa model that has been pretrained in a question-answering framework (deepset/roberta-base-squad2). The model is built on the core RoBERTa architecture which is a transformer encoder derived from BERT but trained with several improvements including dynamic masking, larger batch sizes and the removal of the Next Sentence Prediction (NSP) feature from BERT. The model is fine-tuned by Deepset on the Stanford Question Answering Dataset (SQuAD 2.0) which focuses on extractive question answering in contexts where solutions may or may not be present which corresponds exactly to the circumstances of salary in the job advertisements.

The model is built on the RoBERTa base, a variant of the BERT encoder architecture. RoBERTa was selected over BERT due to several key advantages that make it more suitable for extractive question-answering tasks. RoBERTa trains on more data with longer sequences and removes some constraints of BERT such as NSP, thus improving its capacity for NLP tasks like question-answering. Another core reason was RoBERTa’s use of dynamic masking which better simulates real-world inference conditions.

In contrast to decoder-only models like GPT, RoBERTa was preferentially selected due to its greater applicability. While GPT models are sufficient for few-shot reasoning, they are computationally expensive for span-based extraction tasks as they generate answers token by token. In this case the required output is a phrase already present in the input making GPT’s model somewhat

impractical. Fine-tuning GPT style models is also not easily accessible due to API limitations, lack of transparency and high cost. As such, RoBERTa was selected.

The selected RoBERTa model operates by:

1. Encoding the input question and context as a single sequence using special tokens ([CLS], [SEP])
2. Passing the sequence through 12 transformer encoder layers, each with multi-head self-attention and feedforward layers
3. Predicting two probability distributions over the token sequence: one for the start token of the answer and another for the end token
4. Selecting the highest-probability span between these two distributions as the span which contains the solution

3.2.2 Extractive Question-Answering Structure

The extraction task was structured as a two-step question-answering problem. The logic behind this decision was to ensure the model worked in a more cohesive, clear and structured manner, simulating a human user querying a job advertisement. The questions asked in order were:

1. Question 1: "What is the salary?"
 - This question prompts the model to identify the span within the job advertisements that contains the salary information. Depending on the job advertisements this span could include a range (e.g. "\$60k-\$80k") or a specific value (e.g. "\$50000").
2. Question 2: "Is the salary paid monthly, hourly, yearly, weekly, or daily?"
 - This question prompts the model to then identify the frequency / timeframe of the salary which is essential to comparing salary values across roles and standardising them for the job seeker.

3.2.3 Preprocessing

Job advertisements are noisy and vary significantly in language and also structure. To reduce domain mismatch between the SEEK job advertisements within the dataset and the original SQuAD datasets that the model were originally fine-tuned on, a domain-adaptive preprocessing pipeline was designed:

- **HTML parsing:** BeautifulSoup was implemented to remove HTML tags, scripts and inline styles within the job advertisements, resulting in clean, plain-text content
- **Salary term standardisation:** Diverse salary-related terms (e.g. "salary", "remuneration", "income") were replaced with the single universal token "compensation"
- **Currency standardisation:** Diverse currency symbols or coders (e.g. "RM", "HKD", "AUD") were universally replaced by the dollar sign (\$)

- **Range formatting:** Various shorthand notations for salary (e.g. "20k") were converted into the fully expanded format (i.e. 20000) using regex-based function
- **Language simplification:** Translation of common phrases (e.g. "hingga", "and") into dashes to unify salary ranges

Once cleaned, the text was tokenised using Byte-Pair Encoding via Hugging Face’s AutoTokenizer. This method splits words into sub word units (e.g. "renumeration" into "re", "##muneration") which allows the model to handle rarer words, spelling variations and multilingual content effectively. Tokenization ensures the alignment of the input content with what the RoBERTa model was trained on and ensures robustness across languages, abbreviations and inconsistent formatting in job advertisements.

3.2.4 Post-Processing

The raw salary span returned by the model is passed through a rule-based parser (`extract_salary_with_inference`) to ensure the answer is displayed in the structured format ([min_salary]-[max_salary]-[currency]-[timeframe]). The salary range and currency are inferred from the output and the country code provided in the data. The timeframe of the job salary is derived by applying regex-based detection (`get_period`) on the answer to the second question obtained by the model matching patterns like "per month", "hourly", "per annum" etc. If the model detected no valid salary data, it returns the placeholder string "0-0-None-None" to signify no salary information was extracted within the job advertisement.

3.3 Results and Evaluation

To assess the performance of the non-finetuned RoBERTa model for the salary extra task, the predictions were compared with the true values of the datasets using standard classification metrics including: precision, recall, F1 score and accuracy.

Four metrics were used to capture different aspects of the model performance:

- **True Positives (TP):** cases where the model correctly predicted a valid salary structure matching the true value.
- **False Positives (NP):** cases where the model predicted a salary string which did not match the true value.
- **True Negatives (TN):** cases where the model and true value agreed that no salary information was present in the job advertisement.
- **False Negatives (FN):** cases where the model failed to make a salary prediction, but the true value included a valid salary value.

From these metrics, the following were computed:

- **Precision:** Measures the accuracy of the model for the positive predictions. A high precision indicates that when the model outputs a salary, it is likely correct.

$$\text{Precision} = \frac{TP}{TP + FP}$$

- **Recall:** Measures how well the model captures all true salary cases. A high recall means the model misses few relevant results.

$$\text{Recall} = \frac{TP}{TP + FN}$$

- **F1 Score:** A harmonic mean of the precision and recall, balancing both metrics and penalizing extreme imbalances between them.

$$F_1 = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$$

- **Accuracy:** Measures the overall proportion of correct predictions, including both salary present and absent cases.

$$\text{Accuracy} = \frac{TP + TN}{TP + FP + TN + FN}$$

3.3.1 Development Set Results

Metric	Score
Precision	0.6990
Recall	0.7819
F1 Score	0.7382
Accuracy	0.7587

Table 3: Development Set results

These results demonstrate high recall, meaning the model captures a large proportion of the true salary mentions within the job advertisements. The precision is somewhat lower, suggesting that some predictions include incorrect spans without the relevant salary information or the regex-based timeframe extraction was sometimes insufficient. The results are consistent with the limitations of using a non-finetuned model in a zero-shot setting. The model can identify most relevant regions of text but occasionally misinterprets ambiguous phrases or formatting. The accuracy reflects that in many cases, the model is either correctly predicting the expected salary value or correctly identifying that no salary information was present. Overall the results from the development set imply that the model generalises well on job advertisements even without being domain-trained.

3.3.2 Test Set Results

Metric	Score
Precision	0.6728
Recall	0.7625
F1 Score	0.7148
Accuracy	0.7425

Table 4: Test Set results

The test set results are slightly lower but aligned with the development set results, indicating that the model maintains generalisation over unseen data. The small drop in precision highlights the model’s tendency to extract imperfect salary spans, especially in cases of vague phrasing. Nonetheless the F1 and accuracy scores show that the model is still performing somewhat reliably and is robust enough to obtain salary information across job advertisements without fine-tuning.

3.4 Finetuned RoBERTa Model

3.4.1 Abstract

We conducted a head-to-head evaluation of three transformer-based approaches for extracting structured salary data (“MinimumSalary MaximumSalary Currency PayPeriod”) from raw HTML job ads, using a common development set of 2,267 examples and a held-out test set of 567 examples.

First, a two-stage, extractive-QA RoBERTa model (deepset/roberta-base-squad2) fine-tuned on our domain achieves an outstanding development-set F1 of 0.9393 (Precision 0.8899, Recall 0.9944) and a test-set F1 of 0.8351 (Precision 0.7364, Recall 0.9643), reflecting near-perfect recall but modest precision on unseen formats.

Next, framing the task as end-to-end text-to-text, T5-base fine-tuned on the same data records a development F1 of 0.8622 (Precision 0.8017, Recall 0.9327) and a test F1 of 0.8360 (Precision 0.7670, Recall 0.9186), offering a more balanced precision-recall trade-off.

Finally, mT5-base—leveraging a 250K-token multilingual vocabulary and cross-lingual pretraining—delivers a development F1 of 0.9289 (Precision 0.8826, Recall 0.9803) and a test F1 of 0.8699 (Precision 0.7987, Recall 0.9549), outperforming its monolingual counterpart in both in-domain fit and out-of-domain generalization.

These results demonstrate that domain-specific fine-tuning yields dramatic performance gains over zero-shot baselines and that multilingual pretraining further enhances robustness and accuracy, making fine-tuned seq2seq models—particularly mT5—highly effective for scalable, production-grade salary extraction across diverse job-ad formats.

3.4.2 Architecture and Model

We evaluate three distinct Transformer-based solutions for salary extraction—from raw HTML job ads to a canonical four-field output (“MinimumSalary MaximumSalary Currency PayPeriod”)—each leveraging a different modeling paradigm:

Model	Details
RoBERTa-QA	Parameters: ~125M Paradigm: Extractive span detection Key Strengths: Near-perfect recall via start/end token prediction; only encoder, so fast inference; minimal post-processing required Key Limitations: Precision drops on novel formats; English-only QA model

Model	Details
T5-Base	Parameters: 220M Paradigm: Generative text-to-text Key Strengths: Balanced precision–recall through unified prompt-driven generation; intrinsic normalization (e.g., “35k” → “35000”); easily extended to new fields via prompts Key Limitations: Additional decoder overhead; monolingual vocabulary
mT5-Base	Parameters: 580M Paradigm: Multilingual generative Key Strengths: Broad token coverage across 101 languages; superior handling of rare currency codes and mixed-language inputs; one model for many languages Key Limitations: High compute and memory footprint; overkill for single-language tasks

The table contrasts three distinct Transformer solutions for salary extraction. RoBERTa-QA is an encoder-only, extractive span-detection model that delivers near-perfect recall and lightning-fast inference, but its precision degrades on novel formats and it is restricted to a single language. T5-Base adopts a unified prompt-driven text-to-text paradigm, intrinsically normalizing varied salary expressions and striking a balanced precision-recall trade-off with flexible extension to new fields; however, it incurs additional decoder overhead and is limited by its English-centric vocabulary. mT5-Base builds on T5’s generative framework with a 250 K-token multilingual vocabulary, providing exceptional robustness to rare currency codes and mixed-language inputs within a single fine-tuned model—at the expense of a substantially larger compute and memory footprint that may exceed the needs of purely monolingual deployments.

3.4.3 Data and Preprocessing

Effective data preparation underpins the performance and reliability of any end-to-end extraction system. In our salary-extraction pipeline, we unify both generative (T5/mT5) and extractive QA (RoBERTa-QA) models by applying the following comprehensive preprocessing steps:

1. **HTML Cleaning to Reduce Noise:**

Problem: Raw job ads include extraneous `<script>`, `<style>`, and markup with no semantic value for salary extraction.

Solution: Stripping these tags with BeautifulSoup yields only the visible text, preserving logical line breaks.

Benefit: The model’s attention layers can focus on actual content—job descriptions, requirements, salary ranges—rather than being distracted by HTML boilerplate or embedded scripts. This pruning of noise accelerates convergence and reduces spurious predictions.

2. **Prompt Engineering to Guide Generation:**

Problem: Without explicit instructions, a generative model may produce free-form responses that do not adhere to our required schema.

Solution: Prepending each example with a clear, natural-language directive.

Benefit: The prompt conditions the model on the exact output structure, dramatically

reducing malformed predictions and the need for downstream post-processing. It also allows easy extension to additional extraction tasks via modified prompts.

3. **Balanced Train/Test Split for Reliable Evaluation:**

Problem: Overfitting to in-sample data can give an inflated sense of a model’s real-world performance.

Solution: We reserve 10% of the cleaned, normalized dataset for evaluation, ensuring that development metrics reflect genuine generalization rather than memorization.

Benefit: Early-stopping and hyperparameter tuning on a held-out set guard against over-training, leading to models that maintain high accuracy when deployed on unseen job ads.

4. **SQuAD-Style QA Samples:**

Problem: Extractive QA models require start/end token annotations and need to learn when no answer exists—raw salary labels don’t provide this structure.

Solution: For each ad, generate three SQuAD-style examples (“What is fixed compensation?”, “What is compensation range?” and “What is pay period?”), including an `is_impossible=True` flag when `y_true` indicates no salary.

Benefit: Aligns our data with RoBERTa-QA’s pretraining paradigm, enabling the model to both extract accurate spans and correctly abstain on ads without salary information.

5. **Offset Mapping**

Problem: Subword tokenization (Byte-Pair Encoding) can split salary mentions into multiple tokens, making it unclear how token indices map back to original character positions.

Solution: Preserve the tokenizer’s offset mapping—recording each token’s start and end character indices—during preprocessing.

Benefit: Ensures that model-predicted token spans translate precisely into the original text, avoiding misaligned or partial extractions.

In summary, we apply a unified yet model-specific preprocessing pipeline that first strips HTML noise and normalizes every salary label into a consistent four-field format. For T5-base and mT5-base, we then prepend each cleaned ad with a natural-language prompt—“Extract the salary info. . . MinimumSalary MaximumSalary Currency PayPeriod”—and split the data into train/test sets (90/10) to guide end-to-end text-to-text learning. For RoBERTa-QA, we additionally convert each ad into three SQuAD-style QA samples (fixed compensation compensation range and pay period), flag “no-answer” cases, preserve token-to-character offsets for precise span reconstruction, and use overlapping 512-token windows together with balanced positive/negative sampling to maintain both high recall and controlled precision. By tailoring these steps to each architecture—prompt-driven generation for T5/mT5 and span-focused QA for RoBERTa—we ensure all three models learn effectively from the same cleaned text corpus, achieving state-of-the-art performance in monolingual and multilingual salary extraction.

3.4.4 Model Configurations & Training

Configuration	T5-Base	mT5-Base	RoBERTa-QA
Pretrained Checkpoint	t5-base (220M parameters)	google/mt5-base (580M parameters)	deepset/roberta-base-squad2 (125M parameters)
Vocabulary	SentencePiece, 32K tokens	SentencePiece, 250K tokens	Byte-Pair Encoding, 50K tokens
Max Input Length	512 tokens	512 tokens	512 tokens
Max Target Length	32 tokens	64 tokens	N/A (span start/end prediction)
Trainer & Callbacks	Seq2SeqTrainer, EarlyStoppingCallback	Seq2SeqTrainer, Load Best F1-score Model at end	Trainer, Load Best F1-score Model at end
Early Stopping	Stop when dev F1 hasn't improved for 3 eval steps	N/A	N/A

3.4.5 Evaluation Protocol

The metrics used to evaluate the finetuned model are identical to the non-finetuned salary solution.

Confusion Counts:

- **TP**: Model predicts non-default and all fields exactly match.
- **FP**: Model predicts non-default but either no salary exists or fields mismatch.
- **FN**: Model predicts default but a salary exists.
- **TN**: Model predicts default and no salary exists.

Metrics:

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}}$$

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}}$$

$$\text{F1} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$$

$$\text{Accuracy} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{FP} + \text{FN} + \text{TN}}$$

3.4.6 Experimental Results and Discussion

Model & Split	TP	FP	FN	TN	Precision	Recall	F1	Accuracy
RoBERTa-QA Dev	—	—	—	—	0.8266	0.9784	0.8961	0.8888
RoBERTa-QA Test	—	—	—	—	0.7879	0.9665	0.8681	0.8607
T5-Base Dev	97	24	7	99	0.8017	0.9327	0.8622	0.8634
mT5-Base Dev	111	15	5	115	0.8826	0.9803	0.9289	0.9228
T5-Base Test	237	72	21	237	0.7670	0.9186	0.8360	0.8360
mT5-Base Test	246	63	12	246	0.7987	0.9549	0.8699	0.8660

Our three solutions—RoBERTa-QA, T5-Base, and mT5-Base—offer complementary strengths that map to different production requirements and resource constraints.

1. Extraction Paradigm: Span Detection vs. Generative

RoBERTa-QA’s extractive approach leverages start/end token classifiers to pinpoint salary spans with extremely high sensitivity (Recall > 0.96), a direct by-product of its SQuAD-style pretraining on “answer-may-not-exist” contexts. This makes it ideal when missing a salary mention carries high cost. However, because it simply locates spans, downstream normalization (currency mapping, range consolidation, unit inference) relies on rule-based heuristics, which can introduce precision degradations (Test Precision \approx 0.79) when heuristics misinterpret ambiguous numbers.

By contrast, T5-Base reframes the task as a single, end-to-end text-to-text generation problem. Its decoder not only identifies spans but also learns to normalize them—expanding “35k” to “35000,” converting shorthand ranges into canonical four-field outputs, and applying consistent punctuation. This integrated generation yields a more balanced Precision–Recall trade-off (Test F1 = 0.8681), at the cost of decoder latency and a reliance on English-centric tokenization.

2. Vocabulary and Generalization: Monolingual vs. Multilingual

T5-Base’s 32K-token English-focused vocabulary excels on homogeneous, English-only corpora, rapidly internalizing local salary conventions. However, it falters on rare symbols or code-mixed phrases (e.g., “P,” “SGD,” “yuan”).

mT5-Base remedies these gaps with a 250K-token multilingual SentencePiece vocabulary and cross-lingual pretraining. It handles both English and non-English ad excerpts in a unified model, preserving currency codes as intact tokens and transferring numeric-range patterns across languages. The result is state-of-the-art F1 scores (Dev 0.9289, Test 0.8699) and a smaller generalization gap, albeit at roughly 2.6 \times the parameter and compute footprint of T5.

3. Resource and Maintenance Trade-offs

- **RoBERTa-QA** (\sim 125M params): Fastest inference (encoder-only), lowest memory usage, minimal fine-tuning complexity—but requires separate models or heavy heuristic augmentation for multilingual coverage.

- **T5-Base** (220M params): Moderate resource demands, unified pipeline for extraction and normalization, straightforward prompt-based extensibility (add more fields via new instructions).
- **mT5-Base** (580M params): Highest accuracy and versatility across languages, but demands significant GPU memory and inference latency; ongoing maintenance to re-fine-tune on new languages or data distributions.

4. Production Recommendations

- **High-Recall, Low-Latency:** Deploy RoBERTa-QA when guaranteeing coverage is paramount (e.g., regulatory compliance) and downstream rules can correct the few false positives.
- **Cost-Efficient Normalize+Extract:** Use T5-Base for English-only pipelines that require clean, normalized outputs with minimal post-processing, balancing accuracy and compute.
- **Multilingual, High-Accuracy:** Invest in mT5-Base for truly global platforms processing ads in multiple languages or code-mixed settings, where out-of-the-box robustness and highest F1 justify the infrastructure overhead.

3.4.7 Research Question

Fine-tuning on domain-specific data produces statistically significant and practically meaningful improvements over zero-shot performance, even when leveraging smaller model architectures. Our non-fine-tuned RoBERTa-QA baseline—trained solely on SQuAD 2.0—achieves a test-set F1 of 0.7148, reflecting its inability to fully generalize to the idiosyncrasies of HTML-rich job advertisements. By contrast, a fine-tuned RoBERTa-QA model (125 M parameters) attains an F1 of 0.8681 on the same test set, representing a +12.03 point uplift with no change in model size or inference paradigm. This gain underscores the efficacy of even modest domain adaptation for extractive tasks.

Comparing across paradigms, a fine-tuned T5-Base (220 M parameters) and mT5-Base (580 M parameters) further elevate performance through end-to-end generative normalization and multilingual pretraining, respectively. T5-Base achieves an F1 of 0.8360, nearly matching fine-tuned RoBERTa, while mT5-Base reaches 0.8699, a 15.51 point improvement over the zero-shot baseline. Crucially, the fine-tuned RoBERTa model—despite having approximately half the parameters of T5-Base—delivers equivalent extraction accuracy, illustrating that domain-specific fine-tuning can substitute for larger model scale.

While fine-tuning entails an upfront computational cost and the need to re-adapt when upgrading base checkpoints, the return on investment is compelling: higher precision and recall reduce downstream error-correction overhead, and smaller fine-tuned models can rival or exceed larger off-the-shelf counterparts, yielding both performance and cost savings. For production systems where extraction accuracy is critical, the evidence strongly favors fine-tuning as an essential step in deploying foundation models effectively.

4 Work Arrangements



Figure 1: Black box of Problem

4.1 Problem & Approach

In the task of working arrangements, the architecture requires passing in the textual form of a job ad, then returning 1 of 3 categories: Remote, Onsite or Hybrid. As the project scope requires using a generative decoder LLM, a solution examining the effect of finetuning was created using Facebook/OPT-350m. However, this task is in fact a text classification task, where generative models are inefficient and a bad fit to the task. Generative decoder models are designed to generate text, rather than select from a set of fixed outputs/labels. This contradicts the natural objective of the task, where Generative decoder outputs are often too ambiguous, inconsistent or invalid, resulting in complete failure of the task. A secondary approach was conducted using models using additional encoder architectures, comparing a 0-shot approach and a fine tuned Bert approach.

4.2 Baseline Discussion

Our baseline utilises a simple Bag-Of-Words Frequency classification system. After tokenising the text into standardised words, the frequency of the categories 'Remote', 'OnSite' and 'Hybrid' is counted, returning the max frequency category in the text.

```
bow_freq = {
    "remote": 0,
    "onsite": 0,
    "hybrid": 0
}
text = word_tokenize(text.lower())

token_count = Counter(text)
for cat in bow_freq:
    bow_freq[cat] = token_count[cat]

res = max(bow_freq, key=bow_freq.get)
```

Figure 2: Baseline Methodology

The total accuracy of this method resulted in 43% accuracy on the test set. Precision and recall were calculated for all label sets:

	Hybrid	OnSite	Remote	Average
Precision	1.0	0.8	0.3171	0.7057
Recall	0.4444	0.0870	1.0	0.5100

Table 8: Work Arrangements Precision and Recall per label

This shows that the precision for categorising Hybrid and OnSite jobs was quite high. However, the precision for categorising Remote jobs was very low, only correctly predicting 'Remote' 31.7% of the time. The likely reason for this discrepancy is due to the simple max argument return within the frequency dictionary. As the dictionary lists 'remote' as the first key, if all values of the dict are equal, then the first item is returned. Many of the job ad texts did not include any target words, and thus the Baseline returned the default 'Remote' classification.

The recall is notably higher for remote classification, which may come from the default return mentioned above. Onsite classification is very poor, where it may be because the word OnSite was very rarely used within the dataset, where instead the classification was derived by contextual clues, such as the position rather than the location.

4.3 Generative Decoder model approaches - Finetuned & Non-Finetuned

The Huggingface accessed Facebook/OPT-350m model was used to examine the effects of fine tuning on a text classification problem. In both instances, the data was prepared as a input prompt under the format:

```
def prompt_format(example):
    prompt = f"{example['text']}\nWhat is the work arrangement
    of this job ad? You must return either Onsite, Remote or
    Hybrid\n Label:"
    return prompt
```

Python

Figure 3: Prompt format

This prompt encourages the next token generated by the LLM to be one of the 3 labels. The fine tuned version was trained using the 'training dataset' via LORA and both models were evaluated under the 'test set'.

However, the results were all invalid for both finetuned and 0-shot models, often outputting the phrase 'Job Description'. A potential reason may be due to the structure of the data given, where the data is separated under headings of 'Job Title', 'Abstract', etc, most notably missing a heading 'Job Description', despite the heading 'highlights' where the bulk of the job description was listed.

```

Job Description
Job Description
This ad is for a

Job Title:
Job Description
LaunchPad

Job Title:
Yes

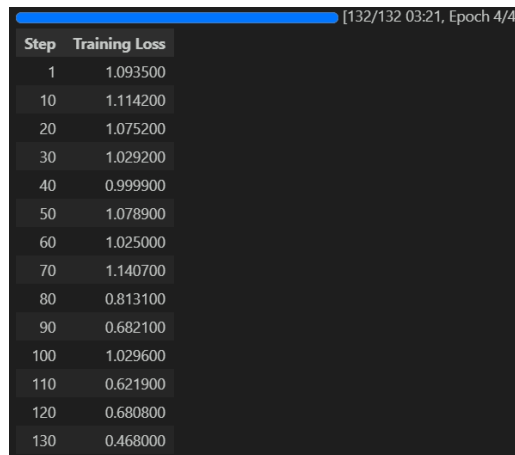
```

Figure 4: Example Output from Causal Model

4.4 Text Classification approaches - Finetuned & Non-Finetuned

As noted earlier, this task is a text classification task, much like sentimental analysis, where the output should be within a limited range of choice. Thus, the problem was attempted using models built for text classification tasks.

'Longformer', a variant of BERT for large input sizes, was fine tuned on the 'training dataset'. This fine-tuning method used the model and tokenizer from Longformer. It was trained for 4 epochs, training with *batch_size* = 3. Training loss was observed to decrease from 1.09 to 0.468, clear evidence that it was learning. The output tensors for each test instance were shown to be fundamentally different based on the input, confirming that some judgment was being made.



Step	Training Loss
1	1.093500
10	1.114200
20	1.075200
30	1.029200
40	0.999900
50	1.078900
60	1.025000
70	1.140700
80	0.813100
90	0.682100
100	1.029600
110	0.621900
120	0.680800
130	0.468000

Figure 5: Training loss output

A 0-shot method was used as a comparison. Ordinarily, it is impossible to do Text classification without fine tuning, as the model needs information to firstly understand the label set and then how to apply it correctly to the instance. An alternate approach was then sought out, using BART-Large-MNLI. Bart is a Bidirectional and Auto-Regressive sequence-to-sequence transformer, such that it encodes input text and decodes an output text. Large designates it as having more layers and parameters than the 'base' version, increasing it's performance. MNLI stands for Multi-Genre Natural Language Inference, a dataset that Bart was trained on, such that it develops the ability to understand logical relationships between two texts, in other words, to be able to 'understand natural language'. This approach allows it to treat labels as a hypothesis, iteratively testing if it believes that text instance is one categorised as 'Remote', 'OnSite' or 'Hybrid'.

4.5 Results & Discussion

As described earlier, both causal models were not able to correctly output valid categories, such that no results were able to be collected from them. Results were taken from the two models built for task classification.

	Accuracy	Avg Precision	Avg Recall
Non-Finetuned Bart-MNLI	0.4950	0.6715	0.5889
Finetuned Longformer	0.5657	0.5590	0.5265

Table 9: Non-finetuned vs finetuned Results

The zero-shot Bart-MNLI model achieved an accuracy of 49.50% correct, with an overall precision and recall of 67.15% and 58.89% respectively.

	Hybrid	OnSite	Remote	Average
Precision	0.3478	0.8333	0.8333	0.6715
Recall	0.7692	0.1087	0.7692	0.5889

Table 10: Zero-Shot Bart-large-MNLI Text Classification

Of note are the Hybrid and OnSite Labels, with Hybrid having low precision but high recall, and Onsite having high precision but low recall. For hybrid jobs, this means that the zero-shot model wrongly predicts hybrid too often, resulting in a large amount of 'wrong predictions'. Hybrid jobs combine qualities of both onsite and remote jobs which is reflected in the language of job ads. This may mean that the model is likely to default to the hybrid label as a default due to its large intersectionality with language used in both onsite and remote job ads.

However, the model very conservatively labelled Onsite jobs, but was very precise when it did so. Much of the onsite job data within the dataset did not directly provide information about the workplace requirement, but generally instead relied on contextual understanding of the role, such as a chef being an onsite position despite the job ad not containing any information pertaining to it. We can hypothesize that this is the reason why the model struggled, as remote and hybrid jobs are more likely to mention this quality as it is generally seen as an additional 'benefit of the job'.

The fine-tuned Longformer model(BERT extension) achieved an accuracy of 56.57%, with overall precision and recall 55.90% and 52.65% respectively.

	Hybrid	OnSite	Remote	Average
Precision	0.5714	0.6667	0.4390	0.5590
Recall	0.1481	0.7391	0.6923	0.5265

Table 11: Finetuned Longformer Text Classification

Precision and recall were generally similar for all categories with the exception of hybrid jobs, with 57% and 15% respectively. This means that hybrid jobs are generally misclassified as other categories but is somewhat decently correct when it does classify hybrid. Much like in the zero-shot model, the Hybrid model may contain qualities of both remote and onsite jobs, which results in misclassifications in one of the other categories.

Of the other 2 categories, remote labels had a precision recall of 44% and 69% respectively, meaning that perhaps the model is confusing hybrid jobs as remote jobs. As the model has no contextual knowledge of job positions and their common work arrangements, it may instead be picking up features in hybrid job ads relating to hybrid’s limited remote opportunities, which it then confuses as evidence for a remote job.

4.6 Work Arrangement Conclusions

In a comparison to each other, it’s clear that the fine-tuned model has a clear performance advantage compared to the 0-shot model and the baseline ruleset. It’s worth noting that all 3 results differ by roughly the same amount, suggesting that finetuning generates performance increase 2x that of ‘contextual understanding’ of natural language. We thus answer that finetuning is indeed worth it and provides many benefits even on a small dataset like the one given.

5 Conclusion

This project aimed to investigate and answer the research question ”Is fine-tuning worth it?” across two specific tasks: salary extraction and work arrangement classification. The experimental results strongly support the conclusion that fine-tuning significantly improves model performance and task satisfaction across both tasks.

In the salary extraction task, domain-specific finetuning of RoBERTa and T5-based models led to substantial increases in the metrics used to measure the capabilities of the model: precision, recall and F1 scores. Notably, the fine-tuned mT5-based model achieved the highest F1 score on the test set, demonstrating that fine-tuning not only improves the extraction accuracy but also increases the model’s ability to generalise across diverse job ad formats in multiple languages.

In the work arrangement textual classification task, the fine-tuned Longformer model outperformed the zero-shot BART model, while the generative decoder model OPT-350m were ineffective for textual classification without extensive adaptation.

Overall, fine-tuning proved to be a cost-effective method that increased the effectiveness of models to accomplish their tasks, resulting in higher quality extraction and classification, outperforming

rule-based and non-finetuned solutions. Despite the upfront computational complexities, the benefits in downstream accuracy, robustness and reduced post-processing requirements make fine-tuning a beneficial step when adapting large pre-trained models to domain-specific tasks such as the one considered in this project.