Journal of Information Systems Engineering and Management
2025, 10(48s)
e-ISSN: 2468-4376
https://www.jisem-journal.com/

Research Article

Fine-Tuning Foundation Models for Domain-Specific Test Case Generation

Twinkle Joshi[1], Dishant Gala[2]
[1]*IQGeo, Canada, Email: twinklejjoshi@gmail.com*
[2]*Marsh and Mclennan, USA, Email: gala.dishant1992@gmail.com*

ARTICLE INFO

Received: 20 Feb 2025
Revised: 01 May 2025
Accepted: 18 May 2025

ABSTRACT

Domain-specific fine-tuning has become a cornerstone technique for adapting pre-trained models to high-stakes and specialized tasks across sectors such as healthcare, law, finance, and software engineering. This process customizes foundational AI models by exposing them to domain-relevant data, optimizing their performance for use cases that require nuanced understanding, strict compliance, or specialized syntax. In this article, we examine the theoretical foundation of domain-specific fine-tuning, reinforced by real-world case studies across medical imaging, legal NLP, financial sentiment analysis, and code generation. Additionally, we investigate its emerging role in automated test case generation using large language models (LLMs), demonstrating how fine-tuned models can enhance software quality by producing context-aware, risk-prioritized, and regulation-compliant test cases. By analyzing different fine-tuning techniques, we identify critical considerations including data representativeness, risk of overfitting, and continuous learning. The results underscore the transformative potential of fine-tuning in making AI models both reliable and valuable for domain-specific deployment.

Keywords: Foundation Models, Domain-Specific Fine-Tuning, Test Case Generation, Large Language Models (LLMs), PLM (Pre-trained Language Model), FLM (Fine-tuning Language Model), Fine-Tuning Techniques, Transfer Learning, Domain Adaptation, PEFT.

1. Introduction

Large-scale pre-trained models have demonstrated extraordinary capabilities across a range of general tasks, from language translation and image recognition to code completion and question answering. However, when applied to specialized fields like clinical diagnosis, legal contract analysis, financial forecasting, or software verification, these general-purpose models often fall short. The primary reason lies in their lack of exposure to the nuanced terminology, syntactic patterns, and contextual cues that characterize specialized domains. Domain-specific fine-tuning addresses this gap by adapting general models to the intricacies of a particular field, significantly enhancing their accuracy and applicability.

Fine-tuning involves modifying the parameters of a pre-trained model using curated datasets that are relevant to a specific domain. This process enables the model to internalize field-specific vocabulary, structures, and tasks, such as identifying indemnity clauses in legal documents or detecting tumors in radiological images. In doing so, it transforms the model into a powerful domain expert capable of supporting high-precision tasks.

This article explores both the conceptual framework and implementation strategies of domain-specific fine-tuning. Through a series of case studies—from medical imaging to legal NLP and sentiment analysis in finance—we illustrate the performance gains and practical benefits achieved through this approach. We also delve into the application of fine-tuning in the rapidly evolving field of software testing, where domain-adapted models are reshaping test case generation by incorporating historical defects, domain rules, and code structure awareness.

2. Understanding Domain-Specific Fine-Tuning of models

Domain-specific model fine-tuning involves customizing a general AI model—usually one that's been pre-trained on a broad and varied dataset—to perform well in a particular field like healthcare, law, finance, or engineering. This process adjusts the model's internal parameters so it better understands the specialized language, data patterns, and

Journal of Information Systems Engineering and Management
2025, 10(48s)
e-ISSN: 2468-4376
https://www.jisem-journal.com/

Research Article

requirements of the targeted domain. For example, a general language model can be fine-tuned using medical texts, such as clinical reports or scientific studies, to improve its accuracy in medical tasks like diagnosing conditions or retrieving healthcare information. By doing so, organizations can develop AI solutions that are both precise and aligned with the specific demands and regulations of their industry.

The value of fine-tuning for specific domains is significant. General-purpose models, while flexible, often fall short in delivering the accuracy needed for niche applications due to differences in terminology, context, and data formats. Fine-tuning addresses this shortfall by exposing the model to domain-relevant data, enabling it to learn specialized concepts and workflows. This ensures that AI systems can meet high standards for accuracy and reliability—an essential requirement in fields where mistakes can have serious consequences, such as medicine or financial services.

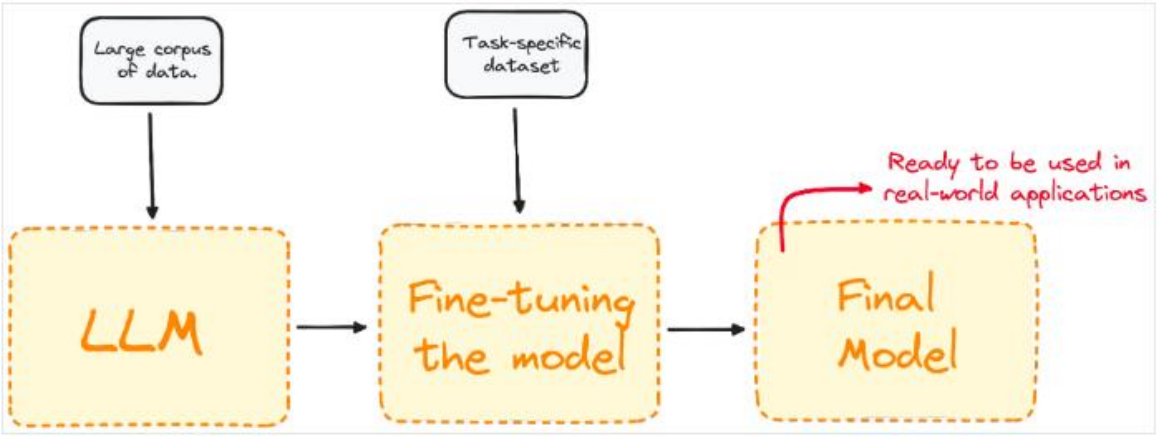

Figure 1: *Visualizing the Fine-Tuning process.*
Reference: https://www.datacamp.com/tutorial/fine-tuning-large-language-models

2.1. Domain-Specific Fine-Tuning in Real-World Applications

Case Study — Medical Imaging for Tumor Detection

In the healthcare domain, machine learning is frequently used to analyze diagnostic images for detecting tumors or other abnormalities. Fine-tuning convolutional neural networks (CNNs) on medical imaging data has shown promising results in this area.

- **Challenge:** Medical images often contain subtle and complex features that general models, trained on broad datasets, fail to detect. These models may lack the sensitivity required to identify small yet critical signs of disease.
- **Techniques Applied:** Researchers fine-tuned pre-trained CNNs using domain-specific datasets labeled for tumor presence. Transfer learning techniques were employed, with lower layers of the model frozen to retain basic visual recognition skills, while upper layers were fine-tuned to identify tumor-specific patterns.
- **Results Achieved:** The fine-tuned models achieved significantly improved sensitivity and accuracy, especially in detecting early-stage tumors. False negatives were reduced, which is essential in preventing missed diagnoses that could have life-threatening consequences.
- **Practical Benefits:** Radiologists gained a reliable decision-support tool that could function as a secondary reviewer, enhancing diagnostic accuracy and trust in AI-assisted medical workflows.
- **Potential Pitfalls:** With limited medical datasets, there's a high risk of overfitting—where the model performs well on the training data but struggles with unseen data. Researchers mitigated this with cross-validation and regularization techniques to maintain generalizability.

Case Study — NLP Models for Analyzing Legal Contracts

In the legal sector, AI is increasingly used to analyze contracts, but standard NLP models often struggle with legal terminology and complex structures.

- **Challenge:** Legal documents contain specialized jargon, long sentences, and precise clauses that general NLP models fail to interpret accurately, often missing key terms like indemnities or obligations.

Journal of Information Systems Engineering and Management
2025, 10(48s)
e-ISSN: 2468-4376
https://www.jisem-journal.com/

Research Article

- **Techniques Applied:** A transformer-based model, such as BERT, was fine-tuned on a corpus of annotated legal contracts. Techniques included data augmentation using paraphrasing and entity replacement to diversify the training data and improve robustness. Additional task-specific pretraining aligned the model with legal text structures.
- **Results Achieved:** The fine-tuned model significantly improved in identifying contract types, extracting key clauses, and tagging legal entities. Accuracy in recognizing obligations and indemnity clauses increased by 20%, and the misinterpretation of key phrases dropped substantially.
- **Practical Benefits:** This led to faster contract review and greater consistency, reducing the workload for legal professionals and improving operational efficiency.
- **Potential Pitfalls:** Legal terminology varies across jurisdictions and case contexts, so a model fine-tuned on one dataset may not perform well in another legal environment. Ongoing retraining with updated and regionally diverse data is necessary to maintain model effectiveness.

Case Study — Fine-Tuning Code Generation Models for Specific Programming Languages

In software development, code generation tools have become powerful aids for developers, especially when fine-tuned for specific languages and frameworks.

- **Challenge:** General models trained on broad code repositories may produce invalid syntax or fail to align with language-specific best practices, especially in complex or less common languages.
- **Techniques Applied:** Language models, including those based on GPT architectures, were fine-tuned on curated, language-specific datasets from platforms like GitHub. Tokenization was customized to recognize syntax and symbols unique to each language. Transfer learning was also used to retain core language modeling capabilities.
- **Results Achieved:** The fine-tuned models delivered more accurate, context-aware code suggestions, improved multi-line completions, and reduced syntax errors. Usability and developer satisfaction increased, especially for Python, Java, and C++.
- **Practical Benefits:** Developers saved time by receiving more relevant code completions and fewer errors, ultimately improving coding efficiency and reducing debugging workload.
- **Potential Pitfalls:** Programming languages evolve quickly, introducing new syntax or libraries. Without regular updates, models may generate outdated or incorrect code. Continuous retraining is essential to keep tools aligned with current development practices.

Case Study — Sentiment Analysis of Financial News & Reports

In finance, sentiment analysis helps organizations monitor market trends and investor sentiment by analyzing financial texts like news articles, reports, and press releases.

- **Challenge:** Generic sentiment models often misinterpret financial terminology. Phrases like “bearish forecast” or “liquidity crunch” can be misunderstood, leading to inaccurate sentiment ratings.
- **Techniques Applied:** Fine-tuning was conducted on a domain-specific corpus that included financial news, earnings calls, and investment reports. Techniques such as entity masking were used to direct model attention toward relevant financial terms. Financial dictionaries helped reinforce the correct interpretation of sentiment-laden phrases.
- **Results Achieved:** Accuracy in sentiment classification rose from 70% to over 85%, with significant improvements in identifying negative sentiment—an important factor in risk evaluation.
- **Practical Benefits:** Financial analysts could make faster, more precise assessments of market conditions, enhancing the speed and accuracy of investment decision-making.
- **Potential Pitfalls:** The financial domain changes rapidly. New phrases or market expressions can alter sentiment implications, making periodic retraining on updated data critical to maintaining accuracy.

2.2. Common Lessons Learned from These Case Studies

From across all the case studies, several important lessons emerge about domain-specific fine-tuning:

- **Enhanced Accuracy:** Fine-tuning significantly improves a model’s ability to interpret domain-specific language, syntax, and visual features, resulting in better performance.

Journal of Information Systems Engineering and Management
2025, 10(48s)
e-ISSN: 2468-4376
https://www.jisem-journal.com/

Research Article

- **Improved Real-World Value:** Custom models produce more reliable and applicable outputs, making them suitable for use in high-stakes industries such as healthcare, law, finance, and software.
- **Continuous Learning Is Essential:** Fields like finance and tech evolve quickly. To stay effective, fine-tuned models must be retrained regularly to incorporate new terminology, practices, or regulatory requirements.
- **Representative Data Is Crucial:** Using a diverse and comprehensive dataset within the domain is vital to building models that generalize well. Including various data sources helps increase resilience and prevent narrow specialization.
- **Risk of Overfitting:** Without careful fine-tuning and validation, models may become too tailored to their training set and fail to perform well on new data. Techniques like cross-validation and diversified datasets help prevent this issue.

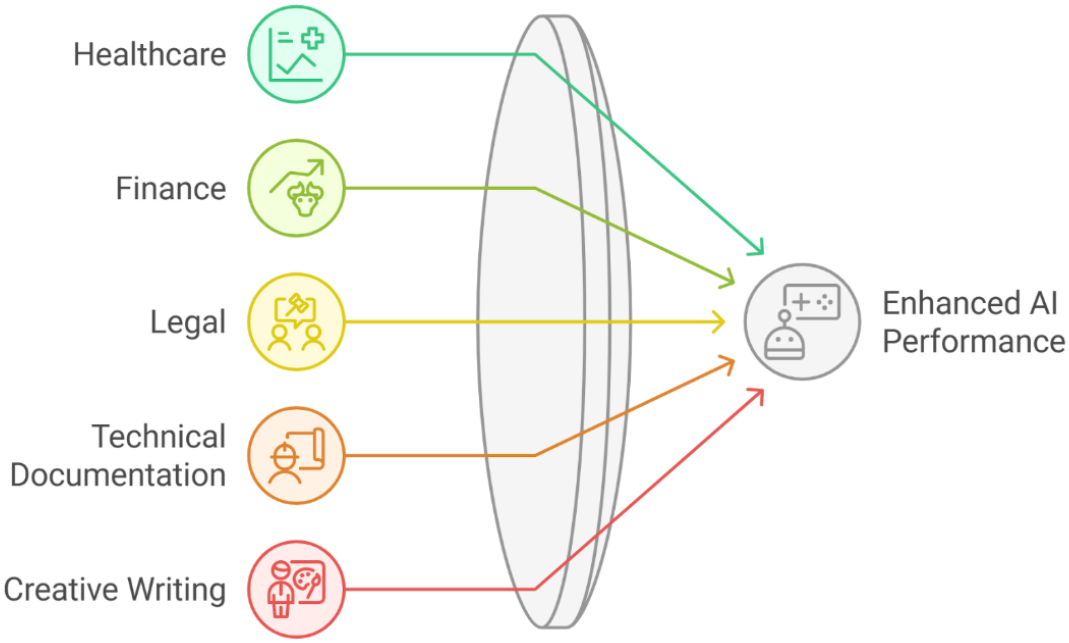

Figure 2: *Domain-Specific Fine-tuning*
Reference: https://www.rapidinnovation.io/post/fine-tuning-large-language-models-llms

3. Domain-Specific Fine-Tuning for test case generation

In the modern era of software development, Large Language Models (LLMs) are reshaping how developers test and validate code. By offering an intelligent, AI-driven approach to test case generation, can help teams automate and accelerate their quality assurance processes. However, while generic LLMs are powerful, their full potential is only realized through domain-specific fine-tuning—a strategy that ensures generated test cases are relevant, accurate, and context-aware.

3.1. Why Generic LLMs Fall Short in Specialized Testing

Generic LLMs are trained on diverse datasets encompassing vast codebases, documentation, and forums. While this breadth of knowledge is valuable, it lacks the specificity required for nuanced testing in certain domains like healthcare, finance, embedded systems, or telecom.
Challenges with using generalized models include:

- Misunderstanding of domain-specific terminology and patterns
- Inability to capture unique error conditions or regulatory constraints
- Generation of irrelevant or low-value test cases
- Inaccurate edge case or boundary testing for critical systems

In short, a one-size-fits-all model may fail to meet the stringent requirements of specialized applications—especially those with complex business logic, safety implications, or high compliance burdens.

Journal of Information Systems Engineering and Management
2025, 10(48s)
e-ISSN: 2468-4376
https://www.jisem-journal.com/

Research Article

3.2. What Is Domain-Specific Fine-Tuning in terms of Testing?

Domain-specific fine-tuning refers to the process of adapting a pre-trained LLM to a particular application area by exposing it to curated, domain-relevant datasets. These datasets typically include:

- Historical bug reports and test logs
- Project-specific unit and integration tests
- Industry-standard compliance rules
- Domain-specific APIs and function signatures

Fine-tuned models learn from this focused dataset, allowing them to generate test cases that are contextually accurate and practically useful.

3.3. Key Benefits of Domain-Specific Fine-Tuning for Test Generation

- **Automation**: Automatically create test cases from specifications, user stories, or source code.
- **Greater Coverage**: Detects edge conditions and complex test scenarios that manual testing might overlook.
- **Accelerated Testing**: Rapidly produce and run test cases with the help of AI-driven tools.
- **Error Minimization**: Lower the likelihood of mistakes in test design and implementation.
- **Cost Reduction**: Streamline resource usage during the testing phase.
- **Adaptive Learning**: Improve test quality over time by learning from historical test outcomes.
- **Full-Spectrum Test Generation**: Create tests for functional behavior, edge conditions, and boundary limits.
- **Realistic Scenario Modeling**: Develop scenarios that reflect both standard and rare user behaviors.
- **Code-Level Analysis**: Evaluate source code to uncover potential vulnerabilities and produce aligned test cases.
- **Quick Turnaround**: Decrease time spent on test creation by automating the process.
- **Coverage Gap Detection**: Identify untested code areas and recommend additional tests.
- **Risk-Based Prioritization**: Rank test cases based on historical issues and system risk levels.
- **Progressive Optimization**: Continuously refine tests using feedback from previous testing rounds.
- **Elimination of Redundancy**: Reduce repetition by generating only unique and relevant test cases.
- **Realistic Test Data**: Generate meaningful data inputs to simulate real-world usage.
- **CI/CD Integration**: Seamlessly embed LLM-powered testing into continuous integration workflows.

3.4. Types of Test Cases That Can be Generated

- **Functional Test Cases**
  - Validate features against expected specifications
  - Ensure the application behaves as intended
- **Regression Test Cases**
  - Re-test existing functionality after code updates
  - Detect bugs that may have been reintroduced
- **Edge Test Cases**
  - Test boundary values and extreme inputs
  - Identify failures under rare or limit conditions
- **Negative Test Cases**
  - Use invalid or unexpected inputs
  - Validate the system's error handling and robustness
- **Performance Test Cases**
  - Measure system response and behavior under load
  - Assess speed, scalability, and stability
- **Security Test Cases**
  - Simulate attacks and misuse scenarios
  - Identify vulnerabilities and improve application security
- **Integration Test Cases**
  - Test interactions between different system modules

Journal of Information Systems Engineering and Management
2025, 10(48s)
e-ISSN: 2468-4376
https://www.jisem-journal.com/

Research Article

- Ensure smooth and consistent data flow
- **Exploratory Test Cases**
  - Conduct creative, unscripted testing
  - Discover unexpected or hidden issues

3.5. Case study: Domain Adaptation for Deep Unit Test Case Generation

Overview:
This study explores the fine-tuning of the CodeT5 model—a transformer-based code model—for unit test case generation. The researchers first fine-tuned CodeT5 on a general test generation task using the Methods2Test dataset. Subsequently, they applied project-level domain adaptation by further fine-tuning the model on specific projects within the Defects4j dataset.
Key Findings:

- The domain-adapted model demonstrated significant improvements in test generation metrics:
  - Line coverage increased by up to 18.62%.
  - Mutation scores improved by up to 16.45%.
  - Enhancements were also observed in BLEU and CodeBLEU scores, indicating better syntactic and semantic quality of the generated tests.

- When combined with search-based tools like EvoSuite, the domain-adapted model contributed to an average increase of 34.42% in line coverage and 6.8% in mutation scores, showcasing its complementary benefits.

Implications:
This case study underscores the effectiveness of domain adaptation in enhancing the performance of foundation models for specific software projects, leading to more comprehensive and effective test suites.

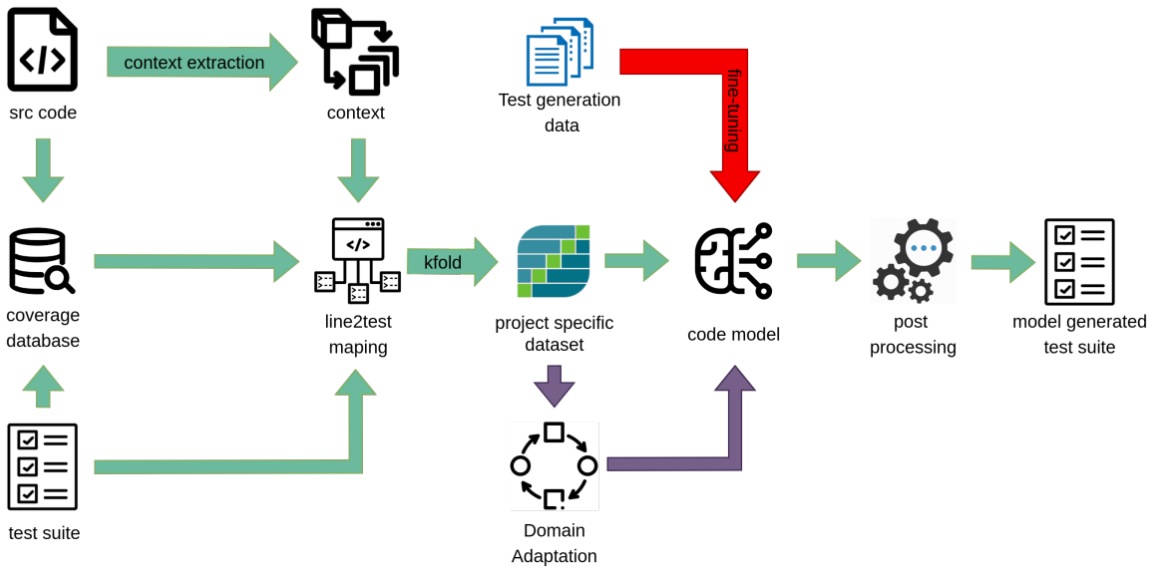

Figure 3: *Test generation framework using CodeT5 and project-level domain adaptation. Red indicates fine-tuning the code model on the test-generation task. Purple indicates project-level domain adaptation. Green indicates inference dataflow.-Specific Fine-tuning*
Reference: https://arxiv.org/pdf/2308.08033v2

4. Key Considerations in Fine-Tuning

4.1. Techniques

Fine-tuning methods help adjust model parameters to meet specific requirements. These techniques generally fall into two categories: supervised fine-tuning and reinforcement learning from human feedback (RLHF).

Journal of Information Systems Engineering and Management
2025, 10(48s)
e-ISSN: 2468-4376
https://www.jisem-journal.com/

Research Article

- **Supervised Fine-Tuning**

This approach trains the model on a labeled dataset where each input is associated with a correct answer. The model refines its parameters to improve accuracy, leveraging its pre-existing knowledge from pre-training. Supervised fine-tuning enhances task-specific performance efficiently.

Common Supervised Fine-Tuning Techniques:

1. **Basic Hyperparameter Tuning** – Manually adjusting parameters like learning rate, batch size, and epochs to optimize model learning while preventing overfitting.
2. **Transfer Learning** – Using a pre-trained model and fine-tuning it on task-specific data, reducing training time and improving performance.
3. **Multi-Task Learning** – Training the model on multiple related tasks simultaneously to enhance generalization and robustness.
4. **Few-Shot Learning** – Enabling the model to adapt to a new task with minimal labeled data by leveraging pre-existing knowledge.
5. **Task-Specific Fine-Tuning** – Optimizing the model for a single, well-defined task to improve precision and relevance.

- **Reinforcement Learning from Human Feedback (RLHF)**

RLHF enhances model performance by incorporating human feedback into the training process. This iterative approach refines responses based on real-world evaluations.

Common RLHF Techniques:

1. **Reward Modeling** – Human evaluators rank model outputs, guiding the model to optimize predictions based on these rankings.
2. **Proximal Policy Optimization (PPO)** – An iterative algorithm that updates the model's policy while maintaining stability by preventing drastic changes.
3. **Comparative Ranking** – Learning from relative rankings of multiple outputs to improve nuanced understanding.
4. **Preference Learning** – Training models based on human preferences between different outputs rather than numerical rewards.
5. **Parameter-Efficient Fine-Tuning (PEFT)** – Modifying only a small subset of parameters to optimize performance while reducing computational costs.

These techniques help tailor LLMs to specific domains, improving accuracy, efficiency, and adaptability.

4.2. Best Practices for Effective Fine-Tuning

- **High-Quality and Sufficient Data**
  The effectiveness of fine-tuning heavily depends on the quality and quantity of the training data. Clean, relevant, and adequately sized datasets are essential—otherwise, poor data input will lead to poor model output ("garbage in, garbage out").
- **Hyperparameter Optimization**
  Fine-tuning involves trial and error. It's important to experiment with different learning rates, batch sizes, and training epochs to find the ideal configuration. Proper tuning ensures the model learns efficiently and generalizes well without overfitting.
- **Ongoing Evaluation**
  Continuously monitor the model's performance using a separate validation set during training. This helps track progress, detect overfitting, and guide necessary adjustments to improve model performance.

4.3. Common Pitfalls to Avoid in Fine-Tuning

- **Overfitting**
  Occurs when the model is trained too long or on too little data, resulting in strong performance on training data but poor generalization to new, unseen inputs.

Journal of Information Systems Engineering and Management
2025, 10(48s)
e-ISSN: 2468-4376
https://www.jisem-journal.com/

Research Article

- **Underfitting**
  Happens when the model doesn't train long enough or uses overly conservative settings, failing to learn the task properly.
- **Catastrophic Forgetting**
  During task-specific fine-tuning, models can lose the general knowledge acquired during pretraining, reducing their ability to handle a wide range of language tasks.
- **Data Leakage**
  Ensuring complete separation between training and validation data is crucial. Overlapping datasets can falsely inflate evaluation scores and misrepresent the model's true performance.

5. Data Preparation Strategies for Domain-Specific Models

5.1. The Role of Domain-Specific Terminology

One of the primary challenges in fine-tuning AI models for specialized tasks is effectively handling domain-specific terminology. General-purpose language models, while broadly capable, often lack the contextual understanding needed to interpret industry-specific jargon, acronyms, or technical language. For example, terms like "EKG," "MRI," and "LDL" in medicine, or "precedent," "statutory," and "appellant" in law, require precise contextual knowledge to be interpreted correctly—something general models are not equipped to handle without further training.
Fine-tuning helps bridge this gap by exposing the model to terminology and language patterns unique to a given domain. This exposure improves the model's ability to accurately recognize and respond to specialized language during tasks like question answering, classification, or summarization. However, the process of fine-tuning with domain-specific terms presents its own set of difficulties. High-quality, domain-relevant data must be curated, annotated correctly, and processed in a way that preserves the nuances of the language used in that field.
Additionally, tokenization and vocabulary design may need to be adapted so that the model can correctly interpret compound terms or abbreviations. In fields like healthcare or finance, where privacy laws and data regulations limit access to annotated datasets, this becomes even more complex. Without proper handling of domain-specific terminology, AI systems risk generating misleading or inaccurate outputs, undermining their usefulness in professional applications.

5.2. Collecting Domain-Specific Data

The first and most crucial step in building a specialized dataset for fine-tuning models is collecting high-quality, domain-specific data. The challenge lies in gathering sufficient data that captures the unique characteristics of the domain, be it legal, medical, or technical. Effective data collection strategies include:

- **Public Databases & Repositories**: Many fields have open-source datasets available. For example, PubMed Central provides medical datasets, while Kaggle and GitHub host various legal, financial, and coding data, all of which are useful for initial fine-tuning.
- **Web Scraping**: In cases where data isn't readily available, web scraping can be a useful approach. By scraping resources like online law libraries or medical journals, one can build a relevant corpus for specific domains. Tools like Scrapy and BeautifulSoup are commonly used for this purpose.
- **Synthetic Data Generation**: When real data is scarce or sensitive, synthetic data can fill gaps. This is especially useful in fields such as healthcare or law, where privacy is a concern, and helps simulate realistic data points for model training.
- **Crowdsourcing & Expert Annotations**: For tasks requiring specialized knowledge, such as labeling medical images or legal documents, crowdsourcing platforms like Amazon Mechanical Turk or tools like Prodigy are effective for gathering accurate, domain-specific annotations from experts.

Each strategy should be selected based on the domain's data availability, accessibility, and quality requirements to ensure the dataset accurately reflects the domain's specific nuances.

5.3. Assembling the Right Team

Building domain-specific datasets requires a cross-functional team to ensure quality and relevance:

Journal of Information Systems Engineering and Management
2025, 10(48s)
e-ISSN: 2468-4376
https://www.jisem-journal.com/

Research Article

- **Domain Experts**: Provide expertise on terminology, patterns, and data quality, ensuring the dataset accurately reflects the domain and is free from irrelevant noise.
- **Data Scientists & Machine Learning Engineers**: Design preprocessing workflows, automate data cleaning and tokenization, and manage large datasets.
- **Compliance Officers**: In regulated sectors, these individuals oversee legal and ethical compliance, ensuring that data use, model behavior, and deployment meet industry regulations.
- **Annotation Specialists**: Ensure accuracy in labeling and annotations, essential for the domain specific tasks..
- **IT and DevOps Teams**: Responsible for operationalizing the model, including deployment, monitoring, and ongoing maintenance. They may also set up automation for model updates and retraining.
- **Business Leaders**: Executives and managers who evaluate the model's strategic value, weighing its impact on efficiency, customer outcomes, and risk management.

5.4. Curating & Preparing the Data

Data curation is a crucial process to ensure only high-quality, relevant data is included in the dataset. In domain-specific data curation, the following steps are key:

- **Removing Irrelevant or Outdated Data**: Outdated information can cause inaccuracies, especially in fast-evolving fields like medicine. Eliminating obsolete data helps keep the dataset up-to-date and accurate.
- **Balancing Data Classes**: In domains with imbalanced classes (e.g., rare diseases in healthcare), it's vital to ensure balance to prevent bias in the model. Techniques like synthetic oversampling (e.g., SMOTE) can help balance underrepresented data.
- **Annotation & Labeling**: Precise annotation is essential for tasks like text classification or entity recognition. Involving domain experts ensures accurate labeling of legal clauses, medical diagnoses, or code errors, improving the model's learning.
- **Data Segmentation**: Segmenting the data based on relevant categories (e.g., separating medical data by disease type) helps structure the dataset and allows the model to focus on distinct aspects of the domain, improving learning effectiveness.

5.5. Data Preprocessing

Once curated, the data must be cleaned and tokenized for training. Pre-processing involves:

- **Handling Domain-Specific Abbreviations and Terms**: It's important to standardize or expand abbreviations and technical terms to avoid ambiguity, especially in fields like medicine (e.g., "BP" for blood pressure) and coding (e.g., "arg" for argument).
- **Noise Reduction**: Domain-specific data often includes irrelevant information such as redundant legal phrases ("plaintiff argued") or comments in code. Removing this noise ensures cleaner, more focused data for fine-tuning.
- **Ensuring Consistency**: Consistent formatting is crucial for effective model learning. Variations in abbreviations (e.g., "HBP" vs. "hypertension") can hinder training, so maintaining consistent terminology is key.
- **Maintaining Data Privacy & Compliance**: In sensitive domains (e.g., healthcare, finance), preprocessing must include data anonymization to protect private information and comply with regulations like HIPAA or GDPR.

6. Step-by-Step Guide to Fine-Tuning Models for Test Case Generation

Fine-tuning large language models (LLMs) for domain-specific test case generation has the potential to significantly improve the efficiency and coverage of your software testing process. This guide walks through the key steps to fine-tune an LLM and integrate it into your testing pipeline. By automating test creation, you reduce manual effort, increase test coverage, and enhance the quality of your software products.

Step 1: Define Your Objectives

The first step in implementing LLM-powered test case generation is to clearly define the objectives of your project. This includes:

- **The target domain** (e.g., finance, healthcare, geospatial, etc.).

Journal of Information Systems Engineering and Management
2025, 10(48s)
e-ISSN: 2468-4376
https://www.jisem-journal.com/

Research Article

- **Test Type**: Identify the type of test cases you need, such as functional, regression, performance, or security test cases.
- **Test Coverage**: Determine the areas of the application that need coverage (e.g., APIs, user interfaces, business logic).
- **Outcome Expectations**: Clarify the expected output from the LLM—do you need detailed test scenarios, edge cases, or boundary tests?

Step 2: Choose the Right LLM

- Choose a large language model (LLM) that best fits your project's needs.
  - **Text-generation models**: GPT-4, GPT-3.5, LLaMA 2, Falcon, Mistral
  - **Code-oriented models**: CodeLlama, StarCoder, Codex, WizardCoder
- These models offer strong capabilities, with the right choice depending on factors like cost, scalability, and output quality. After narrowing down the models, test them with some sample data to evaluate performance. Make sure the model is capable of generating the specific type of test cases required by your project.

LLM	Best For
GPT-4	Complex applications & detailed testing
Gemini Pro	Adaptive & real-time test generation
LLaMA 2	Large-scale enterprise applications
Falcon	Security & performance testing
Mistral	Exploratory & edge case testing

Figure 4: *Choosing the Right LLM for Test Case Generation*

Reference: https://www.frugaltesting.com/blog/llm-powered-test-case-generation-enhancing-coverage-and-efficiency

Step 3: Prepare Your Data

Effective fine-tuning requires high-quality, relevant data. Follow these steps to prepare the data:

- **Data Collection**: Gather domain-specific information like:
  - API documentation
  - Functional requirements
  - User stories and acceptance criteria
  - Any existing test cases
- **Data Formatting**: Organize your data into a structured format, such as:
  - JSON, CSV, or plain text files for easy integration with the LLM.
- **Data Quality**: Clean the data by removing inconsistencies, duplicates, or irrelevant information. For privacy reasons, anonymize sensitive data where necessary.
- **Data Annotation**: If required, label the data with specific tags (e.g., positive, negative, boundary cases) to guide the model in generating diverse test cases.

Journal of Information Systems Engineering and Management
2025, 10(48s)
e-ISSN: 2468-4376
https://www.jisem-journal.com/

Research Article

Step 4: Set Up Your Environment

Ensure your development environment is ready for fine-tuning and LLM integration:

- **Install Dependencies**: Install libraries such as transformers from Hugging Face, or LangChain for managing chains of LLM operations.
  - pip install transformers langchain
- **API Keys**: If you plan to use cloud-based models like GPT-4 or Gemini Pro, make sure to set up your API keys for access.
- **Environment Setup**: Set up a secure environment for managing sensitive data and models, especially when working with external APIs.

Step 5: Fine-Tune the LLM

Fine-tuning involves updating the pre-trained model with domain-specific data to improve its accuracy in test case generation.

- **Transfer Learning**: Start by loading a pre-trained model, then adjust it to the specific vocabulary and patterns of your domain. This process is known as **transfer learning**.
- **Regular Validation**: Continuously evaluate the model's accuracy during training using a validation dataset to ensure it generates relevant and high-quality test cases.

Step 6: Generate Test Cases

- Craft precise prompts that incorporate relevant context and desired results.
- Leverage Fine-tuned model to create diverse test scenarios, including positive, negative, edge, and boundary cases.
- Apply prompt engineering strategies to enhance the quality and effectiveness of test case generation.

Step 7: Validate and Optimize

After generating test cases, validate them to ensure accuracy and relevance:

- **Manual Review**: Have QA engineers review the test cases to check for alignment with functional requirements.
- **Automated Validation**: Use test case validation frameworks to automatically compare the generated test cases against expected outputs.
- **Optimization**: Continuously refine your model and prompts based on feedback and observed results.

Step 8: Integrate with Test Management System

Automate test case execution by integrating with popular test management tools:

- **Jira, TestRail, Zephyr**: Export test cases to these tools for easy tracking and management.
- **Selenium, Playwright, Postman**: Automate test execution by integrating with popular automation frameworks.
- Make sure to monitor and analyze the test results for any discrepancies or insights that can be used to improve the testing pipeline.

Step 9: Monitor and Maintain

Monitor the performance of the model over time:

1. **Model Monitoring**: Track the model's performance to detect any issues with the quality of generated test cases.
2. **Regular Updates**: Retrain the model periodically using new data to maintain accuracy and relevance.
3. **Feedback Loops**: Implement feedback loops to continuously improve the test case generation process based on QA feedback and system performance.

Step 10: Measure Success

Evaluate the success of the LLM-powered test case generation with key metrics:

- **Test Coverage**: Measure how well the generated test cases cover the entire application.
- **Defect Detection**: Compare the defect detection rate between manually created and AI-generated test cases.
- **Execution Time**: Track the time spent on test creation and execution to measure efficiency improvements.

Journal of Information Systems Engineering and Management
2025, 10(48s)
e-ISSN: 2468-4376
https://www.jisem-journal.com/

Research Article

- **Stakeholder Feedback**: Gather feedback from developers and QA engineers to assess the value and quality of the generated test cases.

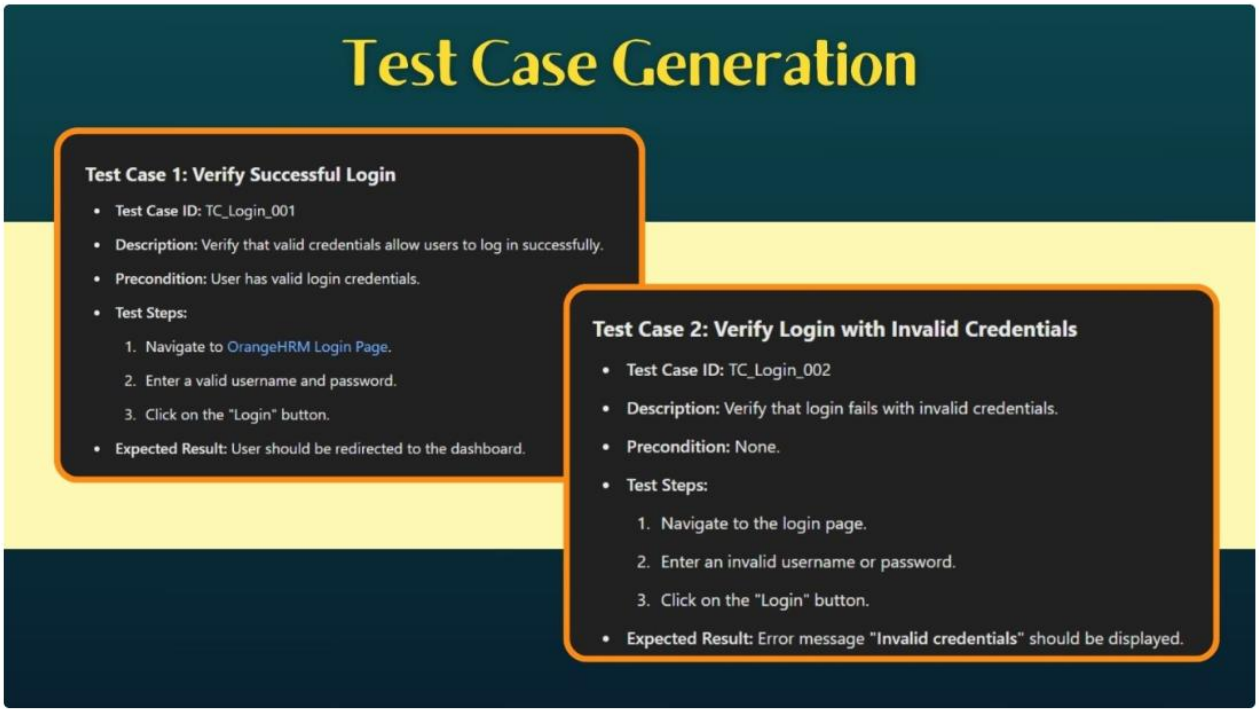

Figure 5: *Test Case Generation*

Reference: https://www.frugaltesting.com/blog/llm-powered-test-case-generation-enhancing-coverage-and-efficiency

7. Conclusion

Domain-specific fine-tuning is no longer a theoretical luxury—it is a practical necessity for deploying AI systems in sensitive and complex domains. As demonstrated across multiple industries—healthcare, law, finance, and software engineering—fine-tuned models outperform their generic counterparts by substantial margins, particularly in tasks requiring domain fluency, precision, and reliability. The case studies reveal how this technique bridges the gap between general AI and specialized applications, delivering value through increased accuracy, faster decision-making, and greater operational efficiency.

In software testing, fine-tuning large language models to domain-specific codebases and testing practices has ushered in a new era of intelligent automation. From improved line coverage to more accurate unit test suggestions, the benefits are clear. However, these advancements come with caveats. Risks like overfitting, data scarcity, and model drift must be mitigated through careful dataset selection, cross-validation, and periodic retraining. Continuous learning, diverse domain representation, and thoughtful adaptation techniques are critical to maintaining long-term effectiveness.

As AI continues to permeate every aspect of digital workflows, domain-specific fine-tuning will be a key enabler in transforming foundation models into trusted specialists. The future lies in hybrid systems where large general models serve as the backbone, but are surgically refined for specific missions—merging breadth with depth for real-world impact.

8. References

[1] N. Benji, "Combatting AI Improvement Slowdown, Part 3 — Enhanced Fine-Tuning Techniques For Specific Domain Performance," Medium, November 14, 2024. https://medium.com/@noel.benji/combatting-ai-improvement-slowdown-part-3-enhanced-fine-tuning-techniques-for-specific-domain-9a87ccdb13e1.

[2] DataCamp, "Fine-Tuning LLMs: A Guide With Examples," DataCamp, December 4, 2024. https://www.datacamp.com/tutorial/fine-tuning-large-language-models.

Research Article

[3] J. Cheonsu, "Fine-tuning and Utilization Methods of Domain-specific LLMs," arXiv, January 2024. https://arxiv.org/pdf/2401.02981.
[4] J. Shin, S. Hashtroudi, H. Hemmati, and S. Wang, "Domain Adaptation for Deep Unit Test Case Generation," arXiv, September. 2024. https://arxiv.org/pdf/2308.08033v2.
[5] FrugalTesting, "LLM-Powered Test Case Generation: Enhancing Coverage and Efficiency," FrugalTesting, April 10, 2025. https://www.frugaltesting.com/blog/llm-powered-test-case-generation-enhancing-coverage-and-efficiency.
[6] Katalon, "Autonomous Test Generation: Revolutionizing Software Testing," Katalon, April 10, 2025: https://katalon.com/resources-center/blog/autonomous-test-generation-revolutionizing-software-testing.
[7] Simform, "A Complete Guide to Fine-Tuning Large Language Models," Simform, July 3, 2023: https://www.simform.com/blog/completeguide-finetuning-llm/
[8] Rapid Innovation, "Ultimate Guide to LLM Fine-Tuning 2025," Rapid Innovation, 2025: https://www.rapidinnovation.io/post/fine-tuning-large-language-models-llms