Domain Name Generation with QLoRA Fine-tuning

This is report on the implementation of a domain name generation system that uses QLoRA fine-tuning on TinyLlama-1.1B to create relevant, brandable domain suggestions for business descriptions.

1. Methodology and Initial Result

1.1 Dataset Creation

The project began with the synthetic generation of a training dataset for the fine tuning of a selected llm.

- **Dataset Generation (`DomainDatasetGenerator`):** A generator was created to produce diverse business descriptions by randomly combining:
  - **Business Types:** 20 types (e.g., "restaurant", "tech startup", "consulting firm").
  - **Adjectives:** 21 descriptors (e.g., "modern", "premium", "sustainable").
  - **Locations:** 22 cities (e.g., "new york", "london", "tokyo").

  Following template is used to create business description:

```
templates = [
f"{adjective} {business_type} in {location}",
f"{business_type} specializing in {adjective} services in {location}",
f"{location}'s premier {adjective} {business_type}",
f"{adjective} {business_type} for {location} residents",
f"{business_type} offering {adjective} products in {location}"
]
```

- **Domain Suggestion Logic:** For each description, three domain names were generated using varied strategies: keyword combination, acronyms, hyphenation, and common affixes. This ensured the model was trained on multiple creative approaches, not just literal translations.

1.2 Dataset Versions

Three versions of the dataset have been created. A base model will be trained on each version, and their performance will be evaluated to determine the best-performing model.

Dataset Version	Description	**Generation Method**
domain_dataset_v1.jsonl	Synthetic dataset with **100 samples**	Generated using `DomainDatasetGenerator`
domain_dataset_v2.jsonl	Synthetic dataset with **500 samples**	Generated using `DomainDatasetGenerator`
domain_dataset_v3.jsonl	LLM-generated dataset with **290 samples**	Generated using **Claude Sonnet LLM**

1.3 Rationale for multiple data versions

The datasets were created with varying sizes and generation methods to systematically evaluate trade-offs:

- **Dataset size:** Smaller datasets (`v1`) allow for quick experimentation, while larger ones (`v2`) provide more training data and potentially better generalisation.
- **Generation method**: Synthetic datasets (`v1`, `v2`) ensure controlled and consistent data creation, where as the LLM-generated dataset (`v3`) introduces more natural and diverse examples, which may better reflect real-world scenarios.
- **Comparison goal:** By training the base model on all three versions, we can identify whether data quantity or data quality/diversity has a greater impact on performance.

1.4 Baseline Model Selection

- **Baseline Model** : **TinyLlama-1.1B-Chat-v1.0** was selected as the base model. This choice due to limitation to lack of gpu for training 7B+ models. Model training was done in Google Colab.
  Baseline Model has advantages and disadvantages.
  - **Advantages:** It is a small, efficient model ideal for rapid iteration and prototyping. Its chat-oriented fine-tuning made it a suitable foundation for instruction-following tasks like domain generation. It is ideal for deployment on edge devices.
  - **Disadvantages:** As a 1.1B parameter model, its creative and reasoning capabilities are inherently limited compared to larger models.

1.5 Model Fine tuning

Implemented QLoRA approach for the model fine tuning. Following is the quantisation and LoRA configurations.

```
bnb_config = BitsAndBytesConfig(
load_in_4bit=True,
bnb_4bit_quant_type="nf4",
bnb_4bit_compute_dtype=torch.bfloat16,
bnb_4bit_use_double_quant=True,
)
```

```
lora_config = LoraConfig(
    r=16,
    lora_alpha=32,
    target_modules=["q_proj", "v_proj"],
    lora_dropout=0.05,
    bias="none",
    task_type="CAUSAL_LM"
)
```

1.6 Initial Model Performance and Evaluation Metrics

The model was fine-tuned on the synthetic dataset using QLoRA (Quantized Low-Rank Adaptation), an efficient fine-tuning technique that reduces memory usage.

- **Model Confidence**: A custom heuristic metric was implemented (`calculate_confidence`) to score generated domains (0-1 scale) based on:
  1. **Relevance** (50% weight): Word overlap between the domain and business description.
  2. **Length Score** (30% weight): Preference for domains between 6-15 characters.
  3. **Complexity Penalty:** Deductions for hyphens, numbers, and mixed case.

- **Initial Performance:**
  - The average confidence across all test cases was very low at 0.303, indicating poor model output.
  - An external `LLMJudgeEvaluator` provided a assessment on five criteria: Relevance, Memorability, Technical Quality, and Creativity. The **average score was a 5.00/10.**
  - Output Formatting: The test results expose a failure in instruction following. The model frequently output invalid JSON, code blocks, comments (e.g., `#Domain Name:`), and even Python dictionaries instead of a simple list of domain names.

The initial model failed to learn the core task reliably.

2. Test Framework and Edge Cases definitions

2.1 Discovery Process: How Edge Cases Were Found

A dedicated `TestFramework` class was implemented to identify model weaknesses. It generates a suite of test cases across multiple categories beyond standard examples.

Failure Taxonomy: Categories of Failures with Examples

Category	Description	Example Input	Example Model Output (Failure)
Short Descriptions	Lack of contextual keywords leads to generic or irrelevant outputs.	`"bakery"`	`getbakery.com`, `thehub.io` (Too generic)
Long/Complex Descriptions	The model struggles to identify important key words from a long description.	`"a comprehensive digital marketing agency that specializes in social media management, SEO..."`	`comprehensive.com` (Picks a single, non-core word)
Ambiguous/Slogan-like	Vague descriptions without concrete nouns are challenging to map to a domain.	`"helping people feel better"`	`helpingpeople.com` (Lengthy, non-brandable)
Unusual Business Types	Lack of similar examples in training data causes poor performance.	`"medieval armor manufacturing workshop"`	`manufacturing.net` (Ignores unique keywords)
Technical/Specialized	Jargon and complex terms are often ignored or misused.	`"quantum computing research laboratory"`	`computing.org` (Drops the key differentiator "quantum")
Multilingual Terms	Non-English words are sometimes poorly integrated or spelled incorrectly.	`"german bakery traditional bretzel"`	`germanbakery.com` (Safe but uncreative, misses "bretzel")

3. Iterative Improvement

Three model versions are created applying same QLoRA three different versions of data.

- **Model 1** is the baseline model fine tuned using domain_dataset_v1.jsonl
- **Model 2** is baseline model fine tuned using domain_dataset_v2.jsonl
- **Model 3** is baseline model fine tuned using domain_dataset_v3.jsonl

3.1 Improvement Strategies

- **Model 1 to Model 2:** Focused on fixing output formatting issues including malformed JSON responses, inconsistent domain structures, and parsing errors that affected reliability. This is achieved by adding more training data.
- **Model 2 to Model 3:** Implemented quality enhancement strategies to improve domain name creativity, relevance, and overall scoring while maintaining the formatting fixes from the previous iteration.

3.2 LLM Judge Validation

To move beyond heuristic scoring, an `LLMJudgeEvaluator` was implemented using the llama-3.3-70b-versatile model via the Groq API.

- **Process:** For each test case, the top generated domains were sent to the LLM judge for evaluation on five criteria: Relevance, Memorability, Brandability, Technical Quality, and Creativity.
- **Validation:** The LLM judge scores were used as the ground truth for measuring the impact of iterative improvements, confirming that increases in the custom confidence score generally correlated with higher LLM judge scores.

3.3 Safety Filter Implementation

- **Keyword-Based Filtering**: Uses a blacklist of 21 inappropriate keywords covering adult content, gambling, illegal activities, violence, and fraud.
- **Immediate Blocking**: Stops processing instantly when prohibited terms are detected, preventing domain generation for harmful content.
- **Detailed Feedback**: Returns specific blocked keyword and rejection reason for transparency and debugging.

4. Model Comparison And Performance Analysis

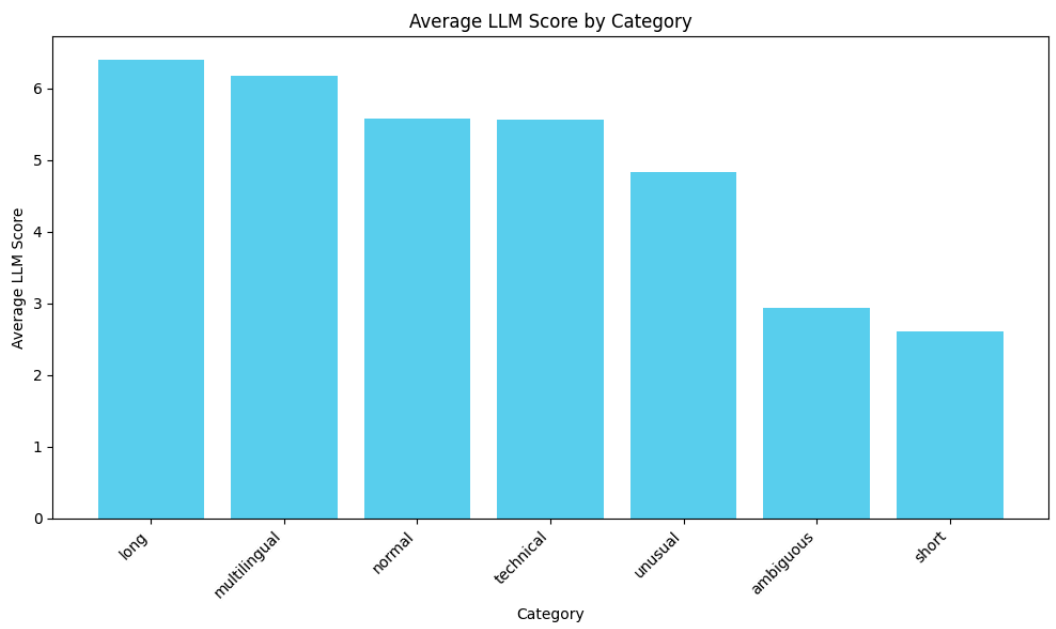
Metric	Model 1	Model 2	Model 3	Improvement (M1→M3)
Average LLM Score	5.00/10	5.16/10	6.21/10	+24.2%
Model Confidence	0.303	0.474	0.437	+44.2%
Best Individual Score	6.7/10	6.6/10	7.0/10	+4.5%
Output Quality Issues	High	Low	Minimal	Significant

- **Average LLM Score**: Model 1 (5.00/10) → Model 2 (5.16/10, +3.2%) → Model 3 (6.21/10, +24.2% total improvement)
- **Model Confidence**: Model 1 (0.303) → Model 2 (0.474, +56.4%) → Model 3 (0.437, +44.2% total improvement)

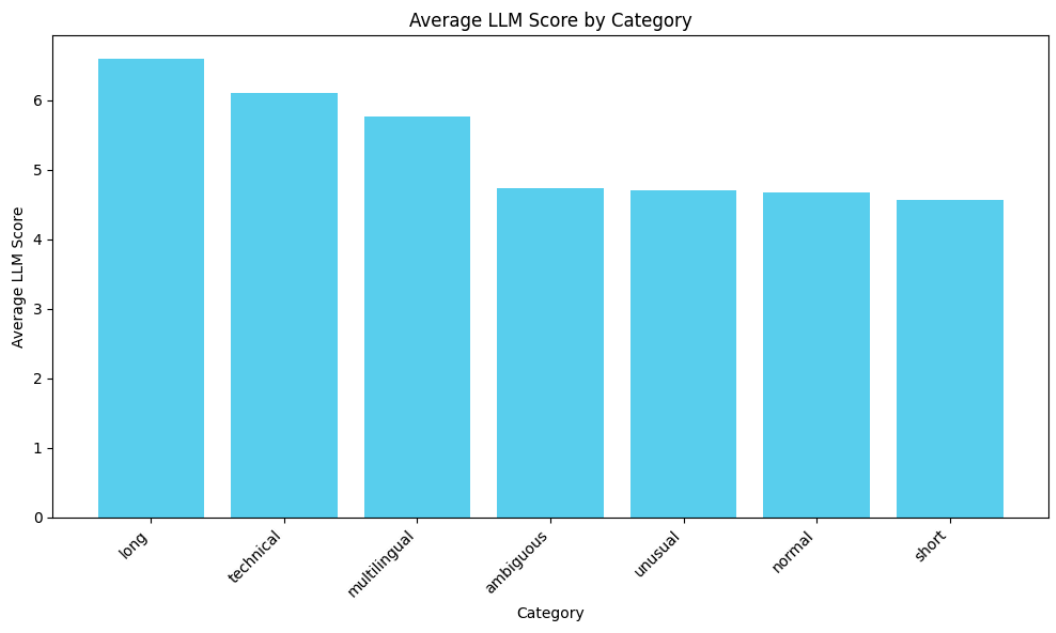
- **Output Quality**: Model 1 had significant formatting errors → Model 2 achieved consistent formatting → Model 3 maintained formatting while improving content quality

4.1 Edge Case Performance Analysis

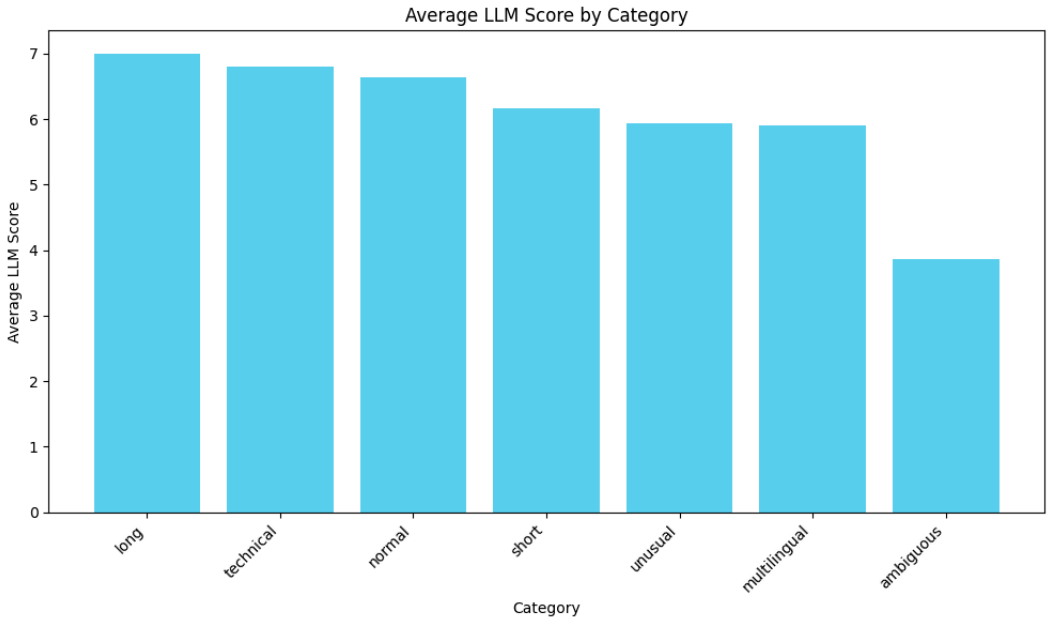
Model 1 performance:



Model 2 Performance:



Model 3 performance :



Edge case improvements from model 1 to model 3:

- **Short Prompts**: Improved from 2.6/10 to 3.87/10 (+49% enhancement in handling minimal input scenarios)
- **Ambiguous Descriptions**: Enhanced from 2.9/10 to 3.87/10 (+33% improvement in interpreting vague requirements)
- **Technical Domains**: Significant jump from 5.27/10 to 6.87/10 (+30% increase in specialized terminology handling)

5. Production Readiness

- Model 3 should be deployed for production use due to its performance across all key metrics (6.21/10 LLM score vs 5.00/10 for Model 1, 24% improvement) .
- Model 3 also resolved formatting issues compared to other version.
- The current QLoRA implementation proved effective for rapid prototyping, successfully reducing memory usage while enabling fine-tuning on limited hardware (Google Colab).
- The LoRA configuration (r=16, alpha=32, targeting q_proj and v_proj layers) achieved reasonable performance gains with minimal computational overhead.

Limitations:

- TinyLlama-1.1B base model lacks creative capabilities needed for high-quality domain generation.
- Custom confidence scoring sometimes shows poor correlation with actual quality.
- the system struggles with edge cases like ambiguous descriptions (3.87/10) and short prompts.

6. Future Improvements:

6.1 QLoRA/LoRA Fine-tuning Improvements

- **Increase LoRA Rank**: Upgrade from r=16 to r=32-64 for more expressive parameter updates.
- **Base Model Upgrade**: Move to 7B+ parameter models with enhanced QLoRA configurations for improved creative output
- **Multi-Adapter Strategy**: Train specialized LoRA adapters for different business categories (technical, creative, traditional)

6.2 Training Data and Evaluation Improvements

- **Training Data**: Replace synthetic data with real-world domain examples and successful branding cases

- **Confidence Scoring**: Develop sophisticated metrics incorporating brandability, market availability, and memorability
- **Evaluation Framework**: Implement A/B testing against user satisfaction metrics rather than automated scores
- **LLM based Evaluation:** Use Latest OpenAI or Claude models as judges. And can combine evaluation from multiple LLM and take average
- **Prompt Engineering**: Develop robust templates specifically designed for edge case handling

7. Github Repository:

Full code can be found here: https://github.com/jithinrajr98/domain-name-generator/tree/main