TuneKit - Model Recommendations by Task

Overview

This document defines the **exact models** TuneKit will recommend for each task type, along with the appropriate fine-tuning method.

Task 1: Text Classification

Use cases: Sentiment analysis, topic classification, spam detection, intent classification

Model Options

Model	Size	Speed	Quality	When to Use	Fine-tuning Method
distilbert-base-uncased	66M	Fast	Good	<1000 rows OR user wants speed	**Full**
bert-base-uncased	110M	Medium	Very Good	1000-5000 rows	**Full**
roberta-base	125M	Medium	Excellent	>5000 rows	**Full**
microsoft/deberta-v3-base	184M	Slower	Best	>10k rows, need max accuracy	**Full**

Decision Logic

```
def get_classification_model(num_rows: int, user_description: str) -> dict:
    """
    Pick best model for text classification
    """
    wants_speed = any(w in user_description.lower()
                      for w in ["fast", "quick", "speed", "efficient"])

    # Speed priority
    if wants_speed and num_rows < 2000:
        return {
            "model": "distilbert-base-uncased",
```

```
            "method": "full",
            "reasoning": "Fast model for speed priority"
        }

    # Small dataset
    if num_rows < 1000:
        return {
            "model": "distilbert-base-uncased",
            "method": "full",
            "reasoning": "Small dataset - lightweight model to avoid overfitting"
        }

    # Medium dataset
    elif num_rows < 5000:
        return {
            "model": "bert-base-uncased",
            "method": "full",
            "reasoning": "Medium dataset - balanced BERT model"
        }

    # Large dataset
    elif num_rows < 10000:
        return {
            "model": "roberta-base",
            "method": "full",
            "reasoning": "Large dataset - RoBERTa for best quality"
        }

    # Very large dataset
    else:
        return {
            "model": "microsoft/deberta-v3-base",
            "method": "full",
            "reasoning": "Very large dataset - state-of-art DeBERTa model"
        }
```

Special Cases

Long texts (>512 tokens):

- Use `allenai/longformer-base-4096` (supports 4096 tokens)
- Method: **Full fine-tuning**

Multilingual:

- Use `xlm-roberta-base` (supports 100+ languages)
- Method: **Full fine-tuning**

Task 2: Named Entity Recognition (NER)

Use cases: Extract names, locations, organizations, dates, medical entities

Model Options

Model	Size	Speed	Quality	When to Use	Fine-tuning Method
distilbert-base-uncased	66M	Fast	Good	<1000 rows OR user wants speed	**Full**
bert-base-cased	110M	Medium	Very Good	1000-5000 rows	**Full**
roberta-base	125M	Medium	Excellent	>5000 rows	**Full**

Decision Logic

```
def get_ner_model(num_rows: int, user_description: str) -> dict:
    """
    Pick best model for NER

    Note: NER typically uses cased models (capitalization matters)
    """
    wants_speed = any(w in user_description.lower()
                      for w in ["fast", "quick", "speed", "efficient"])

    # Speed priority
    if wants_speed and num_rows < 2000:
        return {
            "model": "distilbert-base-cased",
            "method": "full",
            "reasoning": "Fast cased model for NER with speed priority"
        }

    # Small dataset
    if num_rows < 1000:
        return {
            "model": "distilbert-base-cased",
            "method": "full",
```

```
            "reasoning": "Small dataset - lightweight cased model"
        }

    # Medium dataset
    elif num_rows < 5000:
        return {
            "model": "bert-base-cased",
            "method": "full",
            "reasoning": "Medium dataset - BERT-cased for entity recognition"
        }

    # Large dataset
    else:
        return {
            "model": "roberta-base",
            "method": "full",
            "reasoning": "Large dataset - RoBERTa for best NER quality"
        }
```

Special Cases

Biomedical/Medical NER:

- Use `dmis-lab/biobert-base-cased-v1.1` (trained on PubMed)
- Method: **Full fine-tuning**

Clinical NER:

- Use `emilyalsentzer/Bio_ClinicalBERT` (trained on clinical notes)
- Method: **Full fine-tuning**

Multilingual NER:

- Use `xlm-roberta-base` (supports 100+ languages)
- Method: **Full fine-tuning**

Task 3: Instruction Fine-Tuning

Use cases: Chatbots, question answering, instruction following, conversational AI

Model Options

Model	Size	Speed	Quality	When to Use	Fine-tuning Method
TinyLlama/TinyLlama-1.1B-Chat-v1.0	1.1B	Fast	Good	All cases (default)	**QLoRA**
meta-llama/Llama-2-7b-chat-hf	7B	Slower	Excellent	>5000 rows, need quality	**QLoRA**
mistralai/Mistral-7B-Instruct-v0.2	7B	Slower	Excellent	Alternative to Llama-2	**QLoRA**

Decision Logic

```
def get_instruction_model(num_rows: int, user_description: str) -> dict:
    """
    Pick best model for instruction fine-tuning

    Note: Always use QLoRA for LLMs (memory efficient)
    """
    wants_quality = any(w in user_description.lower()
                        for w in ["accurate", "best", "quality", "production"])

    # Default: TinyLlama (accessible, fast)
    if num_rows < 5000 and not wants_quality:
        return {
            "model": "TinyLlama/TinyLlama-1.1B-Chat-v1.0",
            "method": "qlora",
            "reasoning": "TinyLlama with QLoRA - efficient for instruction tuning"
        }

    # Large dataset or quality priority → Llama-2 7B
    else:
        return {
            "model": "meta-llama/Llama-2-7b-chat-hf",
            "method": "qlora",
            "reasoning": "Llama-2 7B with QLoRA - best quality for instruction
tuning"
        }
```

Why QLoRA for All LLMs?

Memory requirements without QLoRA:

- TinyLlama 1.1B: ~8-10GB VRAM

- Llama-2 7B: ~28-32GB VRAM (not feasible for most users)

With QLoRA:

- TinyLlama 1.1B: ~4-5GB VRAM ✅
- Llama-2 7B: ~8-12GB VRAM ✅

QLoRA makes 7B models accessible on consumer GPUs.

Complete Decision Tree

```
def get_model_recommendation(
    task_type: str,
    num_rows: int,
    user_description: str,
    dataset_stats: dict
) -> dict:
    """
    Master function: Returns model + method for any task
    """

    # Task 1: Text Classification
    if task_type == "classification":
        return get_classification_model(num_rows, user_description)

    # Task 2: NER
    elif task_type == "ner":
        return get_ner_model(num_rows, user_description)

    # Task 3: Instruction Tuning
    elif task_type == "instruction_tuning":
        return get_instruction_model(num_rows, user_description)

    else:
        raise ValueError(f"Unknown task type: {task_type}")
```

Summary Table

Default Recommendations (No Special Requirements)

Task	<1000 rows	1000-5000 rows	>5000 rows	Method
Classification	distilbert-base-uncased	bert-base-uncased	roberta-base	Full
NER	distilbert-base-cased	bert-base-cased	roberta-base	Full
Instruction	TinyLlama-1.1B	TinyLlama-1.1B	Llama-2-7B	QLoRA

Special Cases

Scenario	Model	Method
User wants speed	distilbert-base-*	Full
Long texts (>512 tokens)	longformer-base-4096	Full
Multilingual	xlm-roberta-base	Full
Medical/Clinical text	Bio_ClinicalBERT	Full
Biomedical NER	biobert-base-cased-v1.1	Full
Production chatbot	Llama-2-7B-chat	QLoRA

Key Insights

Why Full Fine-Tuning for BERT-sized Models?

1. **Small enough** - BERT models fit in consumer GPUs (4-6GB VRAM)
2. **Best practice** - Industry standard for encoder models
3. **Better quality** - Full fine-tuning gives 100% performance
4. **Simpler code** - No need for PEFT library

Why QLoRA for LLMs?

1. **Memory constraint** - 7B models need 28GB+ without quantization
2. **Accessibility** - QLoRA makes them work on 8-12GB GPUs
3. **Good quality** - Only 1-2% quality loss vs full fine-tuning
4. **Modern standard** - What most people use for LLM fine-tuning

Why These Specific Models?

- **DistilBERT**: Fastest, 66M params, good for small data

- **BERT**: Balanced, 110M params, industry standard
- **RoBERTa**: Best encoder, 125M params, improved BERT
- **DeBERTa**: State-of-art encoder, 184M params, for large data
- **TinyLlama**: Smallest good LLM, 1.1B params, accessible
- **Llama-2 7B**: Best open-source LLM, 7B params, production-ready

---

Implementation Priority

Phase 1 (MVP - Support These First)

- ✅ distilbert-base-uncased (classification)
- ✅ bert-base-uncased (classification)
- ✅ roberta-base (classification)
- ✅ distilbert-base-cased (NER)
- ✅ bert-base-cased (NER)
- ✅ TinyLlama-1.1B-Chat (instruction)

6 models total - covers 90% of use cases

Phase 2 (Extensions)

- xlm-roberta-base (multilingual)
- longformer-base-4096 (long texts)
- Bio_ClinicalBERT (medical)
- Llama-2-7B-chat (production chatbots)

4 additional models - covers special cases

Next Steps

1. Implement the decision logic for Phase 1 models
2. Test with sample datasets for each task type
3. Add Phase 2 models based on user demand
4. Consider adding model size detection (auto-decide full vs LoRA)