Contents

1 Hybrid Model Quick Start 1
1.1 What Was Implemented 1
1.2 Files Created 1
1.3 Files Modified 1
1.4 Quick Usage 2
1.4.1 Basic Example 2
1.4.2 Advanced Example (Separate Model Configs) 2
1.5 Running the Example 3
1.6 How It Works 3
1.7 Key Parameters 4
1.8 Tuning the Threshold 4
1.9 Expected Performance 4
1.10 Next Steps 5
1.11 Monitoring 5
1.12 Troubleshooting 5
1.12.1 Problem: Worse than RNN alone 5
1.12.2 Problem: Very few logistic predictions (<1%) 5
1.12.3 Problem: Training too slow 5
1.13 Documentation 6

1 Hybrid Model Quick Start

1.1 What Was Implemented

A **two-stage hybrid model** that combines: 1. **Logistic Regression**: Detects Nomenclature lines with high confidence 2. **RNN/BiLSTM**: Handles Description and Misc predictions

This leverages the best of both: - Logistic excels at TF-IDF patterns (italicized species names, author citations) - RNN handles sequential context (fragmented description lines)

1.2 Files Created

1. **skol_classifier/hybrid_model.py**: Core implementation
2. **docs/hybrid_model_usage.md**: Full documentation
3. **examples/train_hybrid_model.py**: Usage examples

1.3 Files Modified

1. **skol_classifier/model.py**: Added `model_type='hybrid'` support to `create_model()`

1.4 Quick Usage

1.4.1 Basic Example

```python
from pyspark.sql import SparkSession
from skol_classifier.classifier_v2 import SkolClassifierV2

spark = SparkSession.builder.appName("Hybrid").getOrCreate()

classifier = SkolClassifierV2(
    spark=spark,
    input_source='files',
    file_paths=['data/annotated/*.ann'],
    model_type='hybrid',           # Two-stage pipeline
    nomenclature_threshold=0.6,    # Confidence threshold

    # RNN configuration (for Description/Misc)
    window_size=35,
    hidden_size=256,
    num_layers=3,
    epochs=15,
    batch_size=4000,
    class_weights={
        'Nomenclature': 80.0,
        'Description': 8.0,
        'Misc-exposition': 0.2
    },

    verbosity=1
)

results = classifier.fit()
```

1.4.2 Advanced Example (Separate Model Configs)

```python
classifier = SkolClassifierV2(
    spark=spark,
    input_source='files',
    file_paths=['data/annotated/*.ann'],
    model_type='hybrid',
    nomenclature_threshold=0.6,

    # Logistic parameters (for Nomenclature)
    logistic_params={
        'maxIter': 20,
        'regParam': 0.01,
        'class_weights': {
            'Nomenclature': 50.0,  # Focus on Nomenclature
            'Description': 1.0,
            'Misc-exposition': 1.0
        }
    },

    # RNN parameters (for Description/Misc)
    rnn_params={
        'window_size': 35,
        'hidden_size': 256,
        'num_layers': 3,
        'dropout': 0.4,
        'epochs': 15,
        'batch_size': 4000,
        'class_weights': {
            'Nomenclature': 20.0,
            'Description': 10.0,   # Focus on Description
            'Misc-exposition': 0.2
        }
    },

    verbosity=1
)

results = classifier.fit()
```

1.5 Running the Example

```
cd /data/piggy/src/github.com/piggyatbaqaqi/skol
python examples/train_hybrid_model.py
```

This will: 1. Train a hybrid model with default settings 2. Print performance metrics 3. Save the model to Redis

1.6 How It Works

```
Input Line
    ↓
Logistic Model: Get Nomenclature confidence
    ↓
Is confidence > threshold?
    ├─ YES → Use Logistic's Nomenclature prediction
    └─ NO  → Use RNN's prediction
```

Example: - Line: “Xylaria hypoxylon (L.) Grev.” - Logistic Nomenclature confidence: 0.92 - Threshold: 0.6 - **Decision**: 0.92 > 0.6 → Use Logistic → **Nomenclature** ✓

1.7 Key Parameters

Parameter	Description	Default	Tuning
`nomenclature_threshold`	Confidence required for logistic Nomenclature	0.6	Lower = more Nomenclature predictions
`logistic_params`	Config for logistic model	{}	Set maxIter, regParam, class_weights
`rnn_params`	Config for RNN model	{}	Set window_size, epochs, class_weights

1.8 Tuning the Threshold

Try different values to find the best balance:

```
for threshold in [0.4, 0.5, 0.6, 0.7, 0.8]:
    classifier = SkolClassifierV2(
        ...,
        nomenclature_threshold=threshold
    )
    results = classifier.fit()
    print(f"Threshold {threshold}: F1 = {results['test_stats']['f1_score']:.4f}"
```

Rule of thumb: - **0.4-0.5**: Aggressive Nomenclature detection (high recall, some false positives) - **0.6-0.7**: Balanced (recommended starting point) - **0.7-0.8**: Conservative (high precision, may miss some Nomenclature)

1.9 Expected Performance

Based on your earlier results:

Model	Nomenclature F1	Description F1	Overall F1
Logistic	0.55-0.70	0.30-0.45	0.50-0.60
RNN	0.05-0.20	0.50-0.65	0.45-0.55

Model	Nomenclature F1	Description F1	Overall F1
Hybrid	**0.50-0.65**	**0.55-0.70**	**0.60-0.70**

The hybrid should achieve: - **Nomenclature**: Near logistic performance (logistic's strength) - **Description**: Near RNN performance (RNN's strength) - **Overall**: Better than either model alone

1.10 Next Steps

1. **Try the basic example** with your data
2. **Tune the threshold** to optimize for your use case
3. **Compare against individual models** to verify improvement
4. **Adjust class weights** if needed for each model

1.11 Monitoring

During prediction, you'll see:

```
[Hybrid Predict] Prediction sources:
[Hybrid Predict]   Logistic (Nomenclature): 245 (3.1%)
[Hybrid Predict]   RNN (Description/Misc): 7675 (96.9%)
```

This shows: - 3.1% of predictions came from logistic (high-confidence Nomenclature) - 96.9% of predictions came from RNN (everything else)

1.12 Troubleshooting

1.12.1 Problem: Worse than RNN alone

- **Cause**: Logistic isn't trained well or threshold is wrong
- **Fix**: Check logistic performance alone, adjust threshold

1.12.2 Problem: Very few logistic predictions (<1%)

- **Cause**: Threshold too high or few Nomenclature lines in data
- **Fix**: Lower threshold to 0.4-0.5

1.12.3 Problem: Training too slow

- **Cause**: Training both models takes ~2x time
- **Fix**: Reduce RNN epochs or use smaller batch_size for speed

1.13 Documentation

- **Full guide**: docs/hybrid_model_usage.md
- **Implementation**: skol_classifier/hybrid_model.py
- **Example script**: examples/train_hybrid_model.py