Contents

1 Hybrid Model: Two-Stage Classification

1.1 Overview

The **hybrid model** combines logistic regression and RNN in a two-stage pipeline to leverage the strengths of both approaches:

1. **Stage 1 (Logistic Regression)**: Identifies Nomenclature lines with high confidence using TF-IDF features
2. **Stage 2 (RNN/BiLSTM)**: Handles all other predictions (Description and Misc) using sequential context

This approach addresses a key challenge: logistic regression excels at identifying Nomenclature (italicized species names, author citations) but struggles with fragmented Description lines. Meanwhile,

RNN models handle sequential context well but can miss distinctive Nomenclature patterns.

1.2 When to Use Hybrid Model

Use the hybrid model when: 1. **Logistic regression performs well on Nomenclature** (F1 > 0.5) 2. **RNN performs well on Description** (F1 > 0.6) 3. **Neither model alone achieves good overall performance** 4. **You want to combine complementary strengths**

1.3 Basic Usage

1.3.1 Simple Configuration

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

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

# Create hybrid classifier with default settings
classifier = SkolClassifierV2(
    spark=spark,
    input_source='files',
    file_paths=['data/annotated/*.ann'],
    model_type='hybrid',  # Use two-stage pipeline

    # Hybrid-specific parameter
    nomenclature_threshold=0.6,  # Confidence threshold for logistic Nomenclatur

    # RNN parameters (applied to RNN stage)
    window_size=35,
    hidden_size=256,
    num_layers=3,
    epochs=15,
    batch_size=4000,

    verbosity=1
)

# Train both models
results = classifier.fit()

# Evaluate
print(f"Nomenclature F1: {results['test_stats']['Nomenclature_f1']:.4f}")
print(f"Description F1: {results['test_stats']['Description_f1']:.4f}")
```

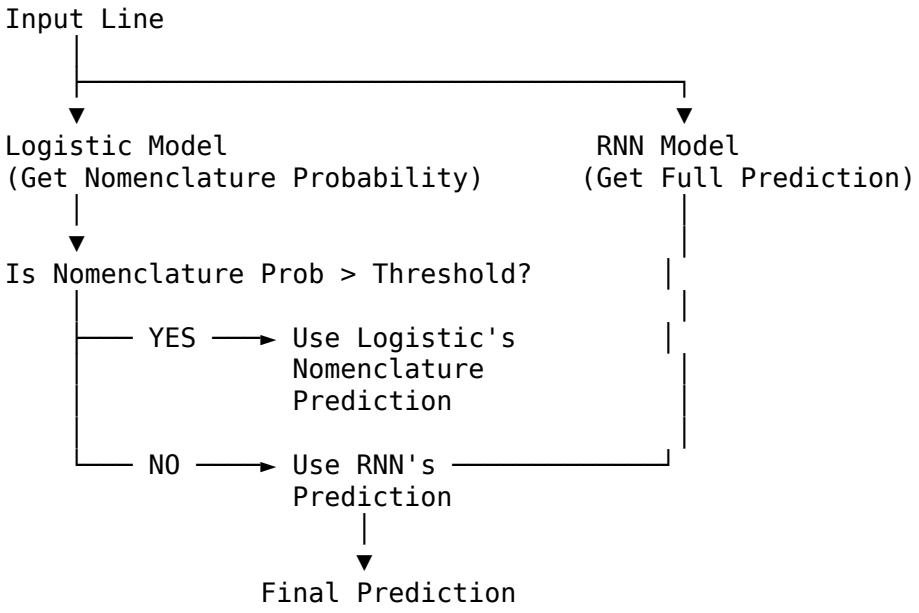
```
print(f"Misc F1: {results['test_stats']['Misc-exposition_f1']:.4f}")
```

1.3.2 Advanced Configuration with Separate Model Parameters

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

    # Threshold for using logistic's Nomenclature prediction
    nomenclature_threshold=0.7,  # Higher = more conservative

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

    # RNN parameters
    rnn_params={
        'window_size': 35,
        'hidden_size': 256,
        'num_layers': 3,
        'dropout': 0.4,
        'epochs': 15,
        'batch_size': 4000,
        'prediction_stride': 1,
        'class_weights': {
            'Nomenclature': 20.0,   # Lower weight than logistic
            'Description': 10.0,    # Focus on Description
            'Misc-exposition': 0.2
        }
    },

    verbosity=1
)

results = classifier.fit()
```

1.4 How It Works

1.4.1 Two-Stage Prediction Flow



1.4.2 Example Predictions

High Confidence Nomenclature (Logistic detects):

```
Line: "Xylaria hypoxylon (L.) Grev."
Logistic Probabilities: [0.05, 0.05, 0.90] ← 90% Nomenclature
Threshold: 0.6
Decision: Use Logistic → Nomenclature ✓
```

Low Confidence / Not Nomenclature (RNN handles):

```
Line: "with hyaline, septate"
Logistic Probabilities: [0.30, 0.40, 0.30] ← 30% Nomenclature
Threshold: 0.6
Decision: Use RNN → Description ✓
```

1.5 Tuning the Threshold

The `nomenclature_threshold` parameter controls how conservative the logistic model is:

1.5.1 Conservative (High Threshold = 0.7-0.8)

- **Precision-focused**: Only very confident Nomenclature predictions used
- **Use when**: Logistic has high precision but lower recall
- **Effect**: Fewer false positive Nomenclature predictions

```
classifier = SkolClassifierV2(
    ...,
    model_type='hybrid',
    nomenclature_threshold=0.75  # Conservative
)
```

1.5.2 Balanced (Medium Threshold = 0.5-0.7)

- **Recommended starting point**
- **Use when**: Logistic has balanced precision/recall
- **Effect**: Good mix of logistic and RNN predictions

```
classifier = SkolClassifierV2(
    ...,
    model_type='hybrid',
    nomenclature_threshold=0.6  # Default, balanced
)
```

1.5.3 Aggressive (Low Threshold = 0.3-0.5)

- **Recall-focused**: More Nomenclature predictions from logistic
- **Use when**: Logistic has high recall but lower precision
- **Effect**: More Nomenclature coverage, some false positives

```
classifier = SkolClassifierV2(
    ...,
    model_type='hybrid',
    nomenclature_threshold=0.4  # Aggressive
)
```

1.5.4 Finding the Optimal Threshold

Try different thresholds and evaluate:

```
thresholds = [0.4, 0.5, 0.6, 0.7, 0.8]

for threshold in thresholds:
    classifier = SkolClassifierV2(
        spark=spark,
        model_type='hybrid',
```

```
        nomenclature_threshold=threshold,
        # ... other params
        verbosity=1
    )

    results = classifier.fit()

    print(f"\nThreshold: {threshold}")
    print(f"  Nomenclature F1: {results['test_stats']['Nomenclature_f1']:.4f}")
    print(f"  Description F1: {results['test_stats']['Description_f1']:.4f}")
    print(f"  Overall F1: {results['test_stats']['f1_score']:.4f}")
```

1.6 Complete Example

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

# Initialize
spark = SparkSession.builder.appName("Hybrid Model Example").getOrCreate()
redis_client = redis.Redis(host='localhost', port=6379, decode_responses=False)

# Configure hybrid model
config = {
    'spark': spark,
    'input_source': 'files',
    'file_paths': ['data/annotated/*.ann'],
    'model_type': 'hybrid',
    'auto_load_model': False,

    # Hybrid configuration
    'nomenclature_threshold': 0.6,

    # Logistic parameters (for Nomenclature detection)
    'logistic_params': {
        'maxIter': 20,
        'regParam': 0.01
    },

    # RNN parameters (for Description/Misc)
    'rnn_params': {
        'window_size': 35,
        'hidden_size': 256,
        'num_layers': 3,
        'dropout': 0.4,
```

```
        'epochs': 15,
        'batch_size': 4000,
        'prediction_stride': 1,
        'class_weights': {
            'Nomenclature': 20.0,
            'Description': 8.0,
            'Misc-exposition': 0.2
        }
    },

    # Storage
    'model_storage': 'redis',
    'redis_client': redis_client,
    'redis_key': 'hybrid_model_v1',

    'verbosity': 1
}

# Train
print("Training hybrid model...")
classifier = SkolClassifierV2(**config)
results = classifier.fit()

# Evaluate
print("\n" + "="*70)
print("Hybrid Model Results")
print("="*70)
print(f"Nomenclature F1: {results['test_stats']['Nomenclature_f1']:.4f}")
print(f"Description F1:  {results['test_stats']['Description_f1']:.4f}")
print(f"Misc F1:         {results['test_stats']['Misc-exposition_f1']:.4f}")
print(f"Overall F1:      {results['test_stats']['f1_score']:.4f}")
print(f"Overall Accuracy: {results['test_stats']['accuracy']:.4f}")

# Save model
classifier.save_model()
print(f"\n✓ Model saved to Redis: {config['redis_key']}")

# Later: Load and predict
new_classifier = SkolClassifierV2(
    spark=spark,
    model_type='hybrid',
    model_storage='redis',
    redis_client=redis_client,
    redis_key='hybrid_model_v1',
    auto_load_model=True
)
```

```
# Make predictions
predictions = new_classifier.predict(new_data)
```

1.7 Expected Performance

With proper configuration, the hybrid model should achieve:

Class	Hybrid F1	Logistic F1	RNN F1	Improvement
Nomenclature	**0.50-0.65**	0.55-0.70	0.05-0.20	Uses logistic's strength
Description	**0.55-0.70**	0.30-0.45	0.50-0.65	Uses RNN's strength
Misc	**0.75-0.85**	0.75-0.85	0.70-0.80	Both models decent
Overall	**0.60-0.70**	0.50-0.60	0.45-0.55	**Best of both**

1.8 Monitoring Predictions

The hybrid model tracks which stage made each prediction:

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

If you see: - **High logistic % (>10%)**: Many Nomenclature lines detected by logistic - **Very low logistic % (<1%)**: Consider lowering threshold or checking logistic training - **Balanced distribution**: Normal for datasets with 3-5% Nomenclature

1.9 Advantages

1. **Leverages Complementary Strengths**:
   - Logistic: Strong TF-IDF pattern matching for Nomenclature
   - RNN: Sequential context for fragmented Descriptions
2. **Better than Ensemble Averaging**:
   - Ensemble: Mixes both models' errors
   - Hybrid: Routes to best model for each class
3. **Interpretable**:
   - Clear decision boundary (threshold)
   - Can inspect which model made each prediction
4. **Tunable**:
   - Single threshold parameter to optimize
   - Can adjust each model independently

1.10 Disadvantages

1. **More Complex**:
   - Trains two models instead of one
   - Longer training time (~2x)
   - Larger storage footprint
2. **Requires Both Models to Work**:
   - If logistic fails, Nomenclature suffers
   - If RNN fails, Description/Misc suffer
3. **Limited to Nomenclature Detection**:
   - Only uses logistic for one class
   - Doesn't leverage logistic for Description/Misc

1.11 Troubleshooting

1.11.1 Problem: Low Nomenclature F1 with Hybrid

Symptom: Hybrid Nomenclature F1 worse than standalone logistic

Cause: Threshold too high or logistic not well-trained

Solution:

```
# Check logistic performance alone
logistic_only = SkolClassifierV2(spark=spark, model_type='logistic', ...)
logistic_results = logistic_only.fit()
print(f"Logistic Nomenclature F1: {logistic_results['test_stats']['Nomenclature_

# If logistic is good, lower threshold
classifier = SkolClassifierV2(
    ...,
    nomenclature_threshold=0.5  # Lower from 0.6
)
```

1.11.2 Problem: Low Description F1 with Hybrid

Symptom: Hybrid Description F1 worse than standalone RNN

Cause: RNN not well-trained or weights too conservative

Solution:

```
# Increase RNN Description weight
classifier = SkolClassifierV2(
    ...,
    rnn_params={
        'class_weights': {
            'Description': 15.0,  # Increase from 8.0
            'Nomenclature': 20.0,
```

```
            'Misc-exposition': 0.2
        }
    }
)
```

1.11.3 Problem: Very Few Logistic Predictions (<1%)

Symptom: Prediction sources show <1% from logistic

Cause: Threshold too high, logistic not confident, or few Nomenclature lines

Solution:

```
# Lower threshold
classifier = SkolClassifierV2(
    ...,
    nomenclature_threshold=0.4  # More aggressive
)

# Or check logistic probability distribution
logistic_probs = logistic_model.predict_proba(test_data)
# Analyze Nomenclature confidence values
```

1.12 Comparison: Hybrid vs Single Models

1.12.1 When to Use Each

Use Logistic Alone: - Nomenclature detection is primary goal - Don't have GPU for RNN training - Want fast training and inference

Use RNN Alone: - Description/Misc are primary goals - Have GPU available - Sequential context is important

Use Hybrid (Recommended when): - Need good performance on ALL classes - Have resources to train both models - Logistic and RNN have complementary strengths

1.13 References

- Related: docs/class_weights_usage.md - Class weights for both models
- Related: docs/focal_f1_loss.md - Alternative RNN loss function
- Implementation: skol_classifier/hybrid_model.py