Contents

1 Model Persistence Fix for SkolClassifierV2

1.1 Issue

Error: `TypeError: cannot pickle '_thread.RLock' object`

Location: skol_classifier/classifier_v2.py:633

Cause: The code was trying to use Python's `pickle` module to serialize PySpark ML models directly. PySpark models contain Java objects (including thread locks) that cannot be pickled.

1.2 Solution

Changed from using `pickle` to using PySpark's native model persistence methods, matching the approach used in the original SkolClassifier.

1.2.1 Key Changes

1. **Disk Storage**: Use PySpark's `.save()` method to save models to a directory structure

2. **Redis Storage**: Create a tar.gz archive of the saved models and store in Redis
3. **Metadata**: Store configuration and label mappings as JSON alongside models

1.3 Implementation Details

1.3.1 Disk Save (classifier_v2.py:574-618)

```
def _save_model_to_disk(self) -> None:
    """Save model to disk using PySpark's native save."""
    # Create directory structure
    model_dir = model_path.parent / model_path.stem

    # Save feature pipeline using PySpark
    self._feature_pipeline.save(str(pipeline_path))

    # Save classifier model using PySpark
    classifier_model.save(str(classifier_path))

    # Save metadata as JSON
    json.dump(metadata, f, indent=2)
```

Directory Structure:

```
model_name/
├── feature_pipeline/        # PySpark pipeline model
├── classifier_model/        # PySpark classifier model
└── metadata.json           # Configuration and labels
```

1.3.2 Disk Load (classifier_v2.py:620-655)

```
def _load_model_from_disk(self) -> None:
    """Load model from disk using PySpark's native load."""
    # Load using PySpark
    self._feature_pipeline = PipelineModel.load(str(pipeline_path))
    classifier_model = PipelineModel.load(str(classifier_path))

    # Load metadata
    metadata = json.load(f)

    # Recreate SkolModel wrapper
    self._model = SkolModel(...)
    self._model.set_model(classifier_model)
```

1.3.3 Redis Save (classifier_v2.py:657-713)

```python
def _save_model_to_redis(self) -> None:
    """Save model to Redis using tar archive."""
    # Save models to temp directory using PySpark
    self._feature_pipeline.save(str(pipeline_path))
    classifier_model.save(str(classifier_path))

    # Save metadata
    json.dump(metadata, f)

    # Create tar.gz archive
    with tarfile.open(fileobj=archive_buffer, mode='w:gz') as tar:
        tar.add(temp_path, arcname='.')

    # Save to Redis
    self.redis_client.set(self.redis_key, archive_data)
```

1.3.4 Redis Load (classifier_v2.py:715-768)

```python
def _load_model_from_redis(self) -> None:
    """Load model from Redis tar archive."""
    # Get from Redis
    serialized = self.redis_client.get(self.redis_key)

    # Extract tar archive to temp directory
    with tarfile.open(fileobj=archive_buffer, mode='r:gz') as tar:
        tar.extractall(temp_path)

    # Load models using PySpark
    self._feature_pipeline = PipelineModel.load(str(pipeline_path))
    classifier_model = PipelineModel.load(str(classifier_path))

    # Load metadata and recreate wrapper
    self._model = SkolModel(...)
    self._model.set_model(classifier_model)
```

1.4 Metadata Format

The `metadata.json` file stores:

```json
{
  "label_mapping": {
    "Label1": 0,
    "Label2": 1
  },
  "config": {
    "line_level": true,
    "use_suffixes": true,
    "min_doc_freq": 2,
    "model_type": "logistic",
    "model_params": {
      "maxIter": 10,
      "regParam": 0.01
    }
  },
  "version": "2.0"
}
```

1.5 Benefits

1. **Compatibility**: Uses PySpark's native serialization which properly handles Java objects
2. **Portability**: Saved models can be loaded on different machines with the same PySpark version
3. **Flexibility**: Models stored in Redis are in the same format, just compressed
4. **Consistency**: Matches the proven approach used in SkolClassifier V1

1.6 Testing

To test model persistence:

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

spark = SparkSession.builder.getOrCreate()

# Train and save to disk
classifier = SkolClassifierV2(
    spark=spark,
    input_source='files',
    file_paths=['data/*.ann'],
    model_storage='disk',
    model_path='models/my_model.pkl',  # Creates models/my_model/ directory
    line_level=True,
    use_suffixes=True,
    model_type='logistic'
)
results = classifier.fit()  # Automatically saves to disk
```

```
# Load from disk
classifier2 = SkolClassifierV2(
    spark=spark,
    input_source='files',
    file_paths=['data/*.txt'],
    model_storage='disk',
    model_path='models/my_model.pkl',
    auto_load_model=True,  # Automatically loads on init
    line_level=True
)
predictions = classifier2.predict()
```

1.7 Files Modified

1. skol_classifier/classifier_v2.py
   - `_save_model_to_disk()` - Lines 574-618
   - `_load_model_from_disk()` - Lines 620-655
   - `_save_model_to_redis()` - Lines 657-713
   - `_load_model_from_redis()` - Lines 715-768

1.8 Related Issues Fixed

This fix also resolves: - ✅ Cannot serialize PySpark models - ✅ Thread lock pickling errors - ✅ Java object serialization issues - ✅ Model portability across machines

1.9 Comparison with Original Approach

1.9.1 Before (Broken):

```
# ❌ Tried to pickle PySpark models directly
model_data = {
    'model': self._model,  # Contains unpicklable Java objects
    'feature_pipeline': self._feature_pipeline
}
pickle.dump(model_data, f)  # TypeError: cannot pickle '_thread.RLock'
```

1.9.2 After (Fixed):

```
# ✅ Use PySpark's native save
self._feature_pipeline.save(str(pipeline_path))  # Works!
classifier_model.save(str(classifier_path))       # Works!
json.dump(metadata, f)  # Python-only data
```

1.10 Notes

- Model directories are created with the same base name as the model_path (e.g., `models/my_model.pkl` → `models/my_model/`)
- Temporary directories are properly cleaned up after Redis operations
- The `_feature_extractor` must be available when loading to determine the correct features column name