Contents

1 Coalesce During Save Fix

1.1 Issue

Problem: When `coalesce_labels=True`, the coalescing was happening during `predict()`, which meant: 1. Users couldn't inspect individual line predictions 2. The notebook couldn't show `predicted_label` for sample predictions 3. The output DataFrame had a completely different structure that broke inspection code

User Report: "In the same code we now get an error about missing predicted_label. When rows are coalesced, they should all have the same predicted_label and it should be preserved."

1.2 Root Cause

The `predict()` method was calling `_format_predictions()`, which in turn called `YeddaFormatter.format()` with `coalesce_labels=True`. This meant:

```python
# In predict()
predictions_df = self._format_predictions(predictions_df)
# Returns coalesced format with no predicted_label column
```

When coalescing happens, the output structure changes from: - **Line-level**: `doc_id`, `attachment_name`, `value`, `line_number`, `predicted_label`, `annotated_value`

To: - **Coalesced**: `doc_id`, `attachment_name`, `coalesced_annotations` (array of strings)

This broke notebook code that expected to inspect `predicted_label`.

1.3 Solution

Design Decision: Keep coalescing as a **save-time operation** only, not a prediction-time operation.

1.3.1 Key Changes

1. **`predict()` always returns line-level data** - Even when `coalesce_labels=True`, users can inspect individual predictions
2. **Coalescing happens only in `save_annotated()`** - The output formatters apply coalescing when saving
3. **Better user experience** - Users can inspect predictions before saving, regardless of coalesce setting

1.3.2 Implementation

Changed: `_format_as_annotated()` to never coalesce

```
# Before - coalescing happened during predict()
def _format_as_annotated(self, predictions_df: DataFrame) -> DataFrame:
    formatter = YeddaFormatter(
        coalesce_labels=self.coalesce_labels,  # ❌ Applied coalescing
        line_level=self.line_level
    )
    return formatter.format(predictions_df)

# After - coalescing deferred to save time
def _format_as_annotated(self, predictions_df: DataFrame) -> DataFrame:
    """
    Format predictions as YEDDA-style annotated blocks.

    Note: This does NOT apply coalescing. Coalescing is only applied
    during save_annotated() to preserve line-level data for inspection.
    """
    formatter = YeddaFormatter(
        coalesce_labels=False,  # ✅ Never coalesce in predict()
        line_level=self.line_level
    )
    return formatter.format(predictions_df)
```

Changed: `_save_to_couchdb()` to use `CouchDBOutputWriter`

```
# Before - used non-existent save_predictions method
def _save_to_couchdb(self, predictions: DataFrame) -> None:
    conn = CouchDBConnection(...)
    conn.save_predictions(  # ❌ This method doesn't exist
        predictions,
        suffix=self.output_couchdb_suffix,
        coalesce_labels=self.coalesce_labels
    )

# After - uses proper output writer that handles coalescing
def _save_to_couchdb(self, predictions: DataFrame) -> None:
    from .output_formatters import CouchDBOutputWriter

    writer = CouchDBOutputWriter(
        couchdb_url=self.couchdb_url,
        database=self.couchdb_database,
        username=self.couchdb_username,
        password=self.couchdb_password
    )

    writer.save_annotated(  # ✅ Applies coalescing here
        predictions,
        suffix=self.output_couchdb_suffix,
        coalesce_labels=self.coalesce_labels,
        line_level=self.line_level
    )
```

1.4 Files Modified

1.4.1 1. skol_classifier/classifier_v2.py

Method: `_format_as_annotated`

Changes: - Line 565: Changed `coalesce_labels=self.coalesce_labels` to `coalesce_labels=False` - Lines 559-562: Added docstring explaining that coalescing is deferred

1.4.2 2. skol_classifier/classifier_v2.py

Method: `_save_to_couchdb`

Changes: - Replaced `CouchDBConnection.save_predictions()`

(which doesn't exist) - With `CouchDBOutputWriter.save_annotated()` (which handles coalescing properly) - Passes `coalesce_labels` and `line_level` to the writer

1.5 Impact

1.5.1 Before (Broken)

```
classifier = SkolClassifierV2(
    spark=spark,
    input_source='couchdb',
    couchdb_database='mydb',
    line_level=True,
    coalesce_labels=True,  # Coalescing during predict()
    output_format='annotated'
)

predictions = classifier.predict()

# ❌ BROKEN: No predicted_label column!
predictions.select("doc_id", "attachment_name", "predicted_label").show()
# Error: Column 'predicted_label' does not exist

# Columns: doc_id, attachment_name, coalesced_annotations
```

1.5.2 After (Fixed)

```
classifier = SkolClassifierV2(
    spark=spark,
    input_source='couchdb',
    couchdb_database='mydb',
    line_level=True,
    coalesce_labels=True,  # Coalescing deferred to save()
    output_format='annotated'
)

predictions = classifier.predict()

# ✅ WORKS: predicted_label column exists!
predictions.select("doc_id", "attachment_name", "predicted_label").show()
# Shows individual line predictions

# Columns: doc_id, attachment_name, value, line_number,
#          predicted_label, annotated_value
# Coalescing happens here during save
classifier.save_annotated(predictions)
# Saved output has coalesced annotation blocks
```

1.6 Workflow

1.6.1 Complete Workflow with Coalescing

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

spark = SparkSession.builder.getOrCreate()

# Configure with coalescing enabled
classifier = SkolClassifierV2(
    spark=spark,
    input_source='couchdb',
    couchdb_url='http://localhost:5984',
    couchdb_database='mydb',
    couchdb_username='admin',
    couchdb_password='password',
    couchdb_pattern='*.txt',
    line_level=True,
    coalesce_labels=True,        # Enable coalescing
    output_format='annotated',
    output_dest='couchdb',
    output_couchdb_suffix='.ann',
    model_type='logistic'
)

# 1. Train (if needed)
results = classifier.fit()

# 2. Predict - always returns line-level data
predictions = classifier.predict()

# 3. Inspect predictions - works because data is line-level
print("Sample predictions:")
predictions.select(
    "doc_id",
    "attachment_name",
    "predicted_label"
).show(5)

# 4. Show label distribution
predictions.groupBy("predicted_label").count().show()
```

```python
# 5. Save - coalescing happens here
print("Saving with coalescing...")
classifier.save_annotated(predictions)
# Output in CouchDB has coalesced annotation blocks
```

1.7 Benefits

1. **Consistent API**: `predict()` always returns the same structure regardless of `coalesce_labels`
2. **Inspectable Results**: Users can always examine individual predictions
3. **Flexible Output**: Users can inspect before saving, even with coalescing enabled
4. **Separation of Concerns**: Prediction and output formatting are separate operations
5. **Backward Compatible**: Existing code that doesn't use coalescing works the same way

1.8 Related Issues Fixed

This fix resolves: - ✅ Missing `predicted_label` column in predictions - ✅ Inability to inspect predictions when coalescing enabled - ✅ Notebook code that expects line-level prediction data - ✅ Incorrect use of non-existent `CouchDBConnection.save_predictions()` method

1.9 Testing

To verify the fix:

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

spark = SparkSession.builder.getOrCreate()

# Test with coalescing enabled
classifier = SkolClassifierV2(
    spark=spark,
    input_source='couchdb',
    couchdb_url='http://localhost:5984',
    couchdb_database='mydb',
    couchdb_pattern='*.txt',
    line_level=True,
    coalesce_labels=True,  # Enable coalescing
    output_format='annotated',
    output_dest='couchdb',
    model_type='logistic'
)

# Predict
predictions = classifier.predict()

# Test 1: Verify line-level structure
assert "predicted_label" in predictions.columns
assert "attachment_name" in predictions.columns
assert "line_number" in predictions.columns
print("✓ Line-level structure preserved")

# Test 2: Verify can inspect predictions
predictions.select("doc_id", "attachment_name", "predicted_label").show(5)
print("✓ Can inspect predictions")

# Test 3: Verify can analyze results
label_dist = predictions.groupBy("predicted_label").count()
label_dist.show()
print("✓ Can analyze prediction distribution")

# Test 4: Verify save works (coalescing applied here)
classifier.save_annotated(predictions)
print("✓ Save with coalescing successful")
```

1.10 Notes

- `predict()` returns line-level data even when `coalesce_labels=True`
- Coalescing only happens during `save_annotated()`
- Users can call `predict()` multiple times without re-coalescing
- The `coalesce_labels` setting affects only the saved output format, not the prediction output