Contents

1 Column Name Fix: row_number → line_number

1.1 Issue

Error: `AnalysisException: [UNRESOLVED_COLUMN.WITH_SUGGESTION] A column or function parameter with name 'row_number' cannot be resolved. Did you mean one of the following? ['line_number', 'value', 'doc_id', 'word_tf', 'words']`

Location: skol_classifier/output_formatters.py:151

Cause: The output formatters were referencing `row_number` column, but the actual column name in the DataFrames is `line_number` (as created by the data loaders and feature extraction pipeline).

1.2 Solution

Changed all references from `row_number` to `line_number` throughout the output_formatters.py file to match the actual column name used in the data pipeline.

1.3 Files Modified

1.3.1 skol_classifier/output_formatters.py

Changed locations:

1. **Line 102**: Docstring for `coalesce_lines` UDF

- Before: `rows: List of (row_number, value, predicted_label) tuples`
- After: `rows: List of (line_number, value, predicted_label) tuples`

2. **Line 153**: `coalesce_consecutive_labels` method
   - Before: `expr("struct(row_number, value, predicted_label)")`
   - After: `expr("struct(line_number, value, predicted_label)")`
3. **Line 198-204**: `FileOutputWriter.save_annotated` method
   - Before: `if "row_number" in predictions.columns:`
   - After: `if "line_number" in predictions.columns:`
   - Before: `expr("sort_array(collect_list(struct(row_number, annotated_value))) AS sorted_list")`
   - After: `expr("sort_array(collect_list(struct(line_number, annotated_value))) AS sorted_list")`
4. **Line 319-325**: `CouchDBOutputWriter.save_annotated` method
   - Before: `if "row_number" in predictions.columns:`
   - After: `if "line_number" in predictions.columns:`
   - Before: `expr("sort_array(collect_list(struct(row_number, annotated_value))) AS sorted_list")`
   - After: `expr("sort_array(collect_list(struct(line_number, annotated_value))) AS sorted_list")`

1.4 Code Changes

1.4.1 YeddaFormatter.coalesce_consecutive_labels

Before:

```
return (
    predictions
    .groupBy(groupby_col)
    .agg(
        collect_list(
            expr("struct(row_number, value, predicted_label)")  # ❌ Wrong colu
        ).alias("rows")
    )
    .withColumn("coalesced_annotations", coalesce_udf(col("rows")))
    .select(groupby_col, "coalesced_annotations")
)
```

After:

```
return (
    predictions
    .groupBy(groupby_col)
    .agg(
```

```
        collect_list(
            expr("struct(line_number, value, predicted_label)")  # ✅ Correct co
        ).alias("rows")
    )
    .withColumn("coalesced_annotations", coalesce_udf(col("rows")))
    .select(groupby_col, "coalesced_annotations")
)
```

1.4.2 FileOutputWriter.save_annotated

Before:

```
# Check if we have row_number for ordering
if "row_number" in predictions.columns:  # ❌ Wrong column
    aggregated_df = (
        predictions.groupBy(groupby_col)
        .agg(
            expr("sort_array(collect_list(struct(row_number, annotated_value)))
        )
        ...
    )
```

After:

```
# Check if we have line_number for ordering
if "line_number" in predictions.columns:  # ✅ Correct column
    aggregated_df = (
        predictions.groupBy(groupby_col)
        .agg(
            expr("sort_array(collect_list(struct(line_number, annotated_value)))
        )
        ...
    )
```

1.4.3 CouchDBOutputWriter.save_annotated

Before:

```
# Check if we have row_number for ordering
if "row_number" in predictions.columns:  # ❌ Wrong column
    predictions = (
        predictions.groupBy(groupby_col, attachment_col)
        .agg(
            expr("sort_array(collect_list(struct(row_number, annotated_value)))
        )
        ...
    )
```

After:

```
# Check if we have line_number for ordering
if "line_number" in predictions.columns:  # ✅ Correct column
    predictions = (
        predictions.groupBy(groupby_col, attachment_col)
        .agg(
            expr("sort_array(collect_list(struct(line_number, annotated_value)))
        )
        ...
    )
```

1.5 Why This Matters

The `line_number` column is created by: 1. **AnnotatedTextLoader**: When loading annotated files with `line_level=True` 2. **RawTextLoader**: When loading raw text files with `line_level=True` 3. **FeatureExtractor**: Preserves the column throughout the pipeline

Using the correct column name ensures that: - Lines are ordered correctly in the output - Coalescing works properly for consecutive labels - Aggregation maintains the original line order from the source files

1.6 Testing

To verify the fix works:

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

spark = SparkSession.builder.getOrCreate()

# Test with line-level processing and coalescing
classifier = SkolClassifierV2(
    spark=spark,
    input_source='files',
    file_paths=['data/*.ann'],
    line_level=True,
    coalesce_labels=True,
    output_format='annotated',
    model_type='logistic'
)

results = classifier.fit()
predictions = classifier.predict()  # Should work now without column errors
```

1.7 Related Issues Fixed

This fix resolves: - ✅ Column resolution errors in output formatting - ✅ Proper line ordering in aggregated output - ✅ Coalescing functionality for consecutive labels - ✅ Consistency with data loader column naming

1.8 Notes

- The column is named `line_number` because it tracks line numbers in the source files
- This naming is consistent across all data loaders (files and CouchDB)
- The column is used for ordering when aggregating predictions back into documents
- For paragraph-level processing, the column still exists but represents paragraph order