Contents

1 Line Number Tracking in Section Extraction Mode **1**
1.1 Overview 1
1.2 Implementation Details 1
1.2.1 1. PDFSectionExtractor Output Schema 1
1.2.2 2. Classifier Integration (Fixed) 2
1.2.3 3. Feature Pipeline Preservation 2
1.2.4 4. Output Sorting 2
1.3 Usage Example 3
1.4 Verification 3
1.5 Files Modified 3
1.6 Related Documentation 4

1 Line Number Tracking in Section Extraction Mode

1.1 Overview

Line numbers are now fully tracked and preserved through the entire prediction pipeline when using `extraction_mode='section'`. This enables proper sorting of extracted sections within each document.

1.2 Implementation Details

1.2.1 1. PDFSectionExtractor Output Schema

The PDFSectionExtractor has always included `line_number` in its output schema:

```
schema = StructType([
    StructField("value", StringType(), False),
    StructField("doc_id", StringType(), False),
    StructField("attachment_name", StringType(), False),
    StructField("paragraph_number", IntegerType(), False),
    StructField("line_number", IntegerType(), False),  # ← Line number field
    StructField("page_number", IntegerType(), False),
    StructField("empirical_page_number", IntegerType(), True),
    StructField("section_name", StringType(), True)
])
```

The `line_number` field tracks the line number of the first line of each section/paragraph in the original document.

1.2.2 2. Classifier Integration (Fixed)

Problem: The classifier's `_load_annotated_from_couchdb()` method was using traditional text loading for all extraction modes, which didn't preserve `line_number` for section mode.

Solution: Updated classifier_v2.py:888-927 to use section extraction when in section mode:

```
def _load_annotated_from_couchdb(self) -> DataFrame:
    """Load annotated data from CouchDB."""
    database = self.couchdb_training_database or self.couchdb_database

    # For 'section' extraction mode, use PDFSectionExtractor
    # which preserves line_number and other metadata
    if self.extraction_mode == 'section':
        return self._load_sections_from_couchdb(database=database)

    # For 'line' and 'paragraph' modes, use traditional text loading
    # ... (rest of method)
```

1.2.3 3. Feature Pipeline Preservation

PySpark ML pipelines automatically preserve all input columns during transformation. The `line_number` column flows through:

1. Tokenization
2. TF-IDF vectorization (words, suffixes, section names)
3. Feature assembly
4. Label indexing

All intermediate transformations preserve the `line_number` column.

1.2.4 4. Output Sorting

The CouchDBOutputWriter already has logic to sort predictions by `line_number` when the column is present:

```
if "line_number" in predictions.columns:
    predictions = (
        predictions.groupBy(groupby_col, attachment_col)
        .agg(
            expr("sort_array(collect_list(struct(line_number, annotated_value)))
        )
        .withColumn("annotated_value_ordered", expr("transform(sorted_list, x ->
        .withColumn("final_aggregated_pg", expr("array_join(annotated_value_orde
        .select(groupby_col, attachment_col, "final_aggregated_pg")
    )
```

This ensures that when predictions are aggregated and saved back to CouchDB, they maintain the original document order.

1.3 Usage Example

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

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

# Train classifier with section extraction mode
clf = SkolClassifierV2(
    spark=spark,
    extraction_mode='section',  # Enable section extraction
    input_source='couchdb',
    couchdb_database='skol_training',
    couchdb_doc_ids=['doc1', 'doc2', 'doc3'],
    output_dest='couchdb',
    verbosity=1
)

# Train model
stats = clf.fit()

# Make predictions (line_number preserved throughout)
predictions = clf.predict()

# Save to CouchDB (sorted by line_number automatically)
clf.save_annotated(predictions)
```

1.4 Verification

The complete pipeline has been verified to preserve `line_number`:

1. ✅ PDFSectionExtractor outputs `line_number`
2. ✅ Classifier loads training data with `line_number` (after fix)
3. ✅ Feature pipeline preserves `line_number`
4. ✅ Model predictions include `line_number`
5. ✅ Output formatter sorts by `line_number`

1.5 Files Modified

- skol_classifier/classifier_v2.py

  - Line 888-927: Updated `_load_annotated_from_couchdb()` to use section extraction for section mode
  - Line 990-1075: Updated `_load_sections_from_couchdb()` to accept optional database parameter

1.6 Related Documentation

- PDF Section Extractor
- Text Attachment Implementation
- Extraction Mode Migration