Contents

1 Extract Taxa Schema Type Mismatch Fixes

1.1 Problem

The `extract_and_save_taxa_pipeline()` function was reporting failures (success=False) when trying to save taxa to CouchDB: - **First issue**: 2763 failures due to incorrect integer field types - **Second issue**: 5239 failures (after first fix) due to None values in source map

All extractions were failing, suggesting issues in the extraction stage, not the save stage.

1.2 Root Causes

Multiple schema type mismatches between the Spark DataFrame schema and the actual data types returned by `Taxon.as_row()`:

1.2.1 What `Taxon.as_row()` Returns

From taxon.py:60-85:

```
def as_row(self) -> Dict[str, None | str | int | Dict[str, None | str | int]]:
    '''Convert this Taxon to a dictionary suitable for output.'''

    # ... code ...
    retval: Dict[str, None | str | int | Dict[str, None | str | int]] = {
        'taxon': "\n".join((str(pp) for pp in self._nomenclatures)),
        'description': "\n".join((str(pp) for pp in self._descriptions)),
        'source': {
            'doc_id': source_doc_id,
            'url': source_url,
            'db_name': source_db_name,
        },
        'line_number': line_number,           # INTEGER (or None)
        'paragraph_number': pp.paragraph_number,  # INTEGER (or None)
        'page_number': pp.page_number,            # INTEGER (or None)
        'empirical_page_number': pp.empirical_page_number,  # INTEGER (or None)
    }
    return retval
```

The method returns: - **integers** (or None) for `line_number`, `paragraph_number`, `page_number` - **string** (or None) for `empirical_page_number` (can be Roman numerals like "xvii") - **dict with None values** for `source` - specifically, `source['url']` can be None

1.2.2 What the Schema Declared

From extract_taxa_to_couchdb.py:290-298 (before fix):

```
extract_schema = StructType([
    StructField("taxon", StringType(), False),
    StructField("description", StringType(), False),
    StructField("source", MapType(StringType(), StringType()), False),  # WRONG:
    StructField("line_number", StringType(), False),        # WRONG: Should be I
    StructField("paragraph_number", StringType(), False),   # WRONG: Should be I
    StructField("page_number", StringType(), False),        # WRONG: Should be I
    StructField("empirical_page_number", StringType(), True),  # Correct type, b
])
```

The schema had multiple issues: 1. **Integer fields declared as StringType**: The numeric fields were declared as strings 2. **MapType doesn't allow None values**: The source dict can have None for the `url` key, but `MapType(StringType(), StringType())` doesn't allow null values 3. **Non-nullable integer fields**: The integer fields should be nullable since they can be None

1.2.3 Why This Caused Failures

When Spark tried to create the DataFrame at line 307:

```
taxa_df = spark.createDataFrame(taxa_rdd, extract_schema)
```

It encountered type mismatches and failed to create the DataFrame, causing all rows to fail. This failure occurred **before** any data reached the `save_taxa_to_couchdb_partition` function, which is why all 2763 records showed `success=False`.

1.3 The Fixes

1.3.1 Fix 1: Integer Field Types (First Issue - 2763 Failures)

Updated the schema to use `IntegerType` for numeric fields and make them nullable.

1.3.2 Fix 2: Allow None in Source Map (Second Issue - 5239 Failures)

Updated the MapType to allow None values using `valueContainsNull=True`.

1.3.3 Final Schema

```
from pyspark.sql.types import MapType, IntegerType

extract_schema = StructType([
    StructField("taxon", StringType(), False),
    StructField("description", StringType(), False),
    StructField("source", MapType(StringType(), StringType(), valueContainsNull=
    StructField("line_number", IntegerType(), True),        # FIXED: IntegerType
    StructField("paragraph_number", IntegerType(), True),   # FIXED: IntegerType
    StructField("page_number", IntegerType(), True),        # FIXED: IntegerType
    StructField("empirical_page_number", StringType(), True),  # Correct: String
])
```

Key changes: 1. **Import IntegerType**: Added to imports at line 288 2. **Changed integer field types**: Numeric fields now use `IntegerType` instead of `StringType` 3. **Made fields nullable**: Changed `nullable=False` to `nullable=True` for integer fields since they can be None 4. **Allow None in map values**: Added `valueContainsNull=True` to MapType for the source field, allowing `source['url']` to be None 5. **Keep empirical_page_number as StringType**: Since it can contain Roman numerals like “xvii”

1.4 Testing

After applying the fix, re-run the pipeline in the notebook:

```
taxa_df = extract_and_save_taxa_pipeline(
    spark=spark,
    ingest_couchdb_url=ingest_couchdb_url,
    ingest_db_name=ingest_db_name,
    taxon_couchdb_url=taxon_couchdb_url,
    taxon_db_name=taxon_db_name,
    ingest_username=ingest_username,
    ingest_password=ingest_password,
    pattern=pattern
)

# Check for successes
successful = taxa_df.filter("success = true").count()
failed = taxa_df.filter("success = false").count()

print(f"Successful: {successful}")
print(f"Failed: {failed}")

# If there are still failures, check the error messages
if failed > 0:
    taxa_df.filter("success = false").select("error_message").distinct().show(tr
```

1.5 Related Issue

This same file also needs to be added to `spark.submit.pyFiles` in the notebook to avoid the `ModuleNotFoundError: No module named 'extract_taxa_to_couchdb'` error when Spark workers try to deserialize UDFs.

Add this line to the Spark configuration in the notebook:

```
.config("spark.submit.pyFiles",
        f'{parent_path / "line.py"},{parent_path / "fileobj.py"},'
        f'{parent_path / "couchdb_file.py"},{parent_path / "finder.py"},'
        f'{parent_path / "taxon.py"},{parent_path / "paragraph.py"},'
        f'{parent_path / "label.py"},{parent_path / "file.py"},'
        f'{parent_path / "extract_taxa_to_couchdb.py"}'  # ADD THIS LINE
       )
```

1.6 Files Modified

- extract_taxa_to_couchdb.py: Fixed schema definition

1.7 Additional Notes

- The type mismatch would have manifested as an error during DataFrame creation, not during the save operation
- All 2763 failures were due to this single schema issue

- The fix ensures the DataFrame schema matches the actual data structure returned by `Taxon.as_row()`