Contents

1 PDF Section Extractor - Figure Caption Extraction

1.1 Overview

The `PDFSectionExtractor` now automatically detects and extracts figure captions from PDF documents. Figure captions are stored separately and excluded from the main sections DataFrame.

1.2 Feature Description

1.2.1 What Gets Extracted

Figure captions matching these patterns are automatically detected: - **"Fig. 1."** - Standard figure caption - **"Figure 1:"** - Full word form - **"Fig 1A."** - With sub-figure labels - **"FIG. 1."** - Uppercase variant

1.2.2 Where Captions Are Stored

Figure captions are stored in a separate data structure accessible via `get_figure_captions()`: - Stored in `self.figure_captions` (list of dictionaries) - Cleared and repopulated on each `extract_from_document()` call - Excluded from the main sections DataFrame

1.3 Usage

1.3.1 Basic Extraction

```python
from pyspark.sql import SparkSession
from pdf_section_extractor import PDFSectionExtractor

# Initialize
spark = SparkSession.builder.appName("PDFExtractor").getOrCreate()
extractor = PDFSectionExtractor(spark=spark)

# Extract sections (also extracts figure captions internally)
sections_df = extractor.extract_from_document(
    database='skol_dev',
    doc_id='document-id'
)

# Access figure captions
captions = extractor.get_figure_captions()

print(f"Found {len(captions)} figure captions")
```

```
for caption in captions:
    print(f"Figure {caption['figure_number']}: {caption['caption'][:80]}...")
```

1.3.2 Caption Data Structure

Each caption is a dictionary with the following fields:

```
{
    'figure_number': str,           # e.g., "1", "2A", "3B"
    'caption': str,                 # Full caption text
    'doc_id': str,                  # CouchDB document ID
    'attachment_name': str,         # PDF filename
    'line_number': int,             # Line number in extracted text
    'page_number': int,             # PDF page number
    'empirical_page_number': int,   # Document page number (nullable)
    'section_name': str             # Section name (nullable)
}
```

1.3.3 Real-World Example

```
# Extract from academic paper
sections_df = extractor.extract_from_document(
    database='skol_dev',
    doc_id='00df9554e9834283b5e844c7a994ba5f'
)

# Get figure captions
captions = extractor.get_figure_captions()

# Output:
# [{
#     'figure_number': '1',
#     'caption': 'Fig. 1. Arachnopeziza hiemalis: A. An ascus. B. Apothecia. ...
#     'doc_id': '00df9554e9834283b5e844c7a994ba5f',
#     'attachment_name': 'article.pdf',
#     'line_number': 41,
#     'page_number': 2,
#     'empirical_page_number': 486,
#     'section_name': 'Holotype'
# }]
```

1.4 Working with Figure Captions

1.4.1 Iterate Over Captions

```
captions = extractor.get_figure_captions()

for i, caption in enumerate(captions, 1):
    print(f"\n--- Figure {i} ---")
    print(f"Number: {caption['figure_number']}")
    print(f"Page: {caption['empirical_page_number']}")
    print(f"Section: {caption['section_name']}")
    print(f"Caption: {caption['caption']}")
```

1.4.2 Export to JSON

```
import json

captions = extractor.get_figure_captions()
with open('figure_captions.json', 'w') as f:
    json.dump(captions, f, indent=2)
```

1.4.3 Create DataFrame from Captions

```
from pyspark.sql.types import StructType, StructField, StringType, IntegerType

# Get captions
captions = extractor.get_figure_captions()

# Define schema
caption_schema = StructType([
    StructField("figure_number", StringType(), True),
    StructField("caption", StringType(), False),
    StructField("doc_id", StringType(), False),
    StructField("attachment_name", StringType(), False),
    StructField("line_number", IntegerType(), False),
    StructField("page_number", IntegerType(), False),
    StructField("empirical_page_number", IntegerType(), True),
    StructField("section_name", StringType(), True)
])

# Create DataFrame
captions_df = spark.createDataFrame(captions, schema=caption_schema)
captions_df.show()
```

1.4.4 Filter by Figure Number

```
captions = extractor.get_figure_captions()

# Get specific figure
fig1 = [c for c in captions if c['figure_number'] == '1']
if fig1:
    print(f"Figure 1 caption: {fig1[0]['caption']}")

# Get figures with sub-labels (e.g., "1A", "1B")
fig1_subfigs = [c for c in captions if c['figure_number'].startswith('1')]
```

1.4.5 Group by Section

```
from collections import defaultdict

captions = extractor.get_figure_captions()

# Group captions by section
captions_by_section = defaultdict(list)
for caption in captions:
    section = caption['section_name'] or 'Unknown'
    captions_by_section[section].append(caption)

for section, caps in captions_by_section.items():
    print(f"\n{section} ({len(caps)} figures):")
    for cap in caps:
        print(f"  - Figure {cap['figure_number']}")
```

1.5 Pattern Detection

1.5.1 Supported Patterns

The `_is_figure_caption()` method detects:

```
# Pattern: (Fig|Figure|FIG) + number + optional letter + separator
r'^(Fig\.?|Figure|FIG\.?)\s*\d+[A-Za-z]?[\.:,\s]'
```

Matches: - "Fig. 1." - "Fig 1A." - "Figure 1:" - "Figure 2B." - "FIG. 3." - "Fig. 1A,B."

Does NOT match: - "See Fig. 1" (not at start) - "The figure shows…" (wrong keyword) - "Fig A" (no number)

1.5.2 Figure Number Extraction

The `_extract_figure_number()` method extracts:

```
# Pattern: (Fig|Figure|FIG) + (number + optional letter)
r'^(?:Fig\.?|Figure|FIG\.?)\s*(\d+[A-Za-z]?)'
```

Examples: - “Fig. 1. Description” → “1” - “Figure 2A: Details” → “2A” - “Fig 3B.” → “3B”

1.6 Implementation Details

1.6.1 Detection Logic

1. **Pattern Matching**: Checks if paragraph starts with figure caption pattern
2. **Metadata Capture**: Stores all context (page, section, line number)
3. **Exclusion**: Skips adding to main DataFrame records
4. **Storage**: Appends to `self.figure_captions` list

1.6.2 Processing Flow

```python
# During parse_text_to_sections():

for paragraph in paragraphs:
    para_text = ' '.join(paragraph).strip()

    if self._is_figure_caption(para_text):
        # Extract figure number
        figure_num = self._extract_figure_number(para_text)

        # Store separately
        self.figure_captions.append({
            'figure_number': figure_num,
            'caption': para_text,
            # ... other metadata
        })
    else:
        # Add to regular sections
        records.append({
            'value': para_text,
            # ... other fields
        })
```

1.6.3 Multiple Document Handling

When processing multiple documents, captions are cleared and repopulated for each document:

```python
# extract_from_document() clears figure_captions at the start
self.figure_captions = []

# After processing each document, captions contain only that document's figures
captions = extractor.get_figure_captions()  # Only from last document
```

Note: For multiple documents, extract captions after each document or store them separately.

1.7 Benefits

1.7.1 1. Cleaner Text Analysis

- Figure captions don't interfere with section text analysis
- Easier to process pure narrative content
- No mixed content types in main DataFrame

1.7.2 2. Structured Figure Data

- Easy access to all figures in a document
- Figure numbers extracted automatically
- Full context preserved (page, section, line)

1.7.3 3. Better Document Understanding

- Know which sections contain figures
- Track figure distribution across pages
- Link figures to specific sections

1.7.4 4. Flexible Access

- Separate accessor method
- Can be converted to DataFrame
- Easy to export or process independently

1.8 Examples

1.8.1 Example 1: Count Figures per Section

```python
from collections import Counter

captions = extractor.get_figure_captions()
section_counts = Counter(c['section_name'] or 'Unknown' for c in captions)

print("Figures per section:")
```

```python
for section, count in section_counts.most_common():
    print(f"  {section}: {count}")
```

1.8.2 Example 2: Create Figure Index

```python
captions = extractor.get_figure_captions()

print("FIGURE INDEX")
print("=" * 60)
for caption in sorted(captions, key=lambda x: x['figure_number']):
    page = caption['empirical_page_number'] or caption['page_number']
    print(f"Figure {caption['figure_number']:3s} - Page {page:3d}")
    print(f"  {caption['caption'][:70]}...")
    print()
```

1.8.3 Example 3: Extract Figure References

```python
# Get all figures
captions = extractor.get_figure_captions()
figure_numbers = [c['figure_number'] for c in captions]

# Find references in text
sections_df = extractor.extract_from_document('db', 'doc_id')

for fig_num in figure_numbers:
    # Find mentions of this figure
    mentions = sections_df.filter(
        sections_df.value.contains(f"Fig. {fig_num}") |
        sections_df.value.contains(f"Figure {fig_num}")
    )

    print(f"\nFigure {fig_num} mentioned in {mentions.count()} sections")
```

1.9 Verification

1.9.1 Check Exclusion from DataFrame

```python
sections_df = extractor.extract_from_document('db', 'doc_id')
captions = extractor.get_figure_captions()

if captions:
    # Try to find first caption in DataFrame
    caption_text = captions[0]['caption'][:50]
    matches = sections_df.filter(sections_df.value.contains(caption_text[:30]))
    if matches.count() == 0:
        print("✓ Figure captions correctly excluded from DataFrame")
    else:
        print("✗ ERROR: Figure caption found in DataFrame")
```

1.9.2 Verbose Output

With `verbosity=2`:

```
Extracted 7926 characters from PDF
Parsed 26 sections/paragraphs
Extracted 1 figure captions
Extracted empirical page numbers: {1: 108, 2: 486, 3: 487, 4: 488, 5: 489}
```

1.10 See Also

- PDF_SECTION_EXTRACTOR_SUMMARY.md - Complete feature summary
- PDF_SECTION_METADATA_ENHANCEMENT.md - Metadata features
- PDF_DATAFRAME_OUTPUT.md - DataFrame features
- pdf_section_extractor.py - Implementation

Update Date: 2025-12-22 **Status**: ✅ Complete and tested **Breaking Changes**: None (backward compatible) **New Features**: Figure caption detection, extraction, and separate storage