Contents

1 Column Name Standardization: annotated_pg → annotated_value

1.1 Issue

Problem: The codebase used inconsistent naming for the annotated prediction column: - Some code used `annotated_pg` (paragraph-level naming) - Other code used `annotated_value` (more generic and accurate)

User Report: “In the next block of the notebook, we fail trying to output annotated_pg. I think that is now annotated_value (which is a better name). Please fix all the references to annotate_pg in the notebook, and python files under the current directory.”

1.2 Root Cause

Historical naming inconsistency: - Original code was written for paragraph-level classification and used `annotated_pg` - V2 API supports both line-level and paragraph-level, making `annotated_pg` misleading - Output formatters use `annotated_value` which is more generic and accurate

1.3 Solution

Standardized all references to use `annotated_value` throughout the codebase.

1.3.1 Why `annotated_value` is Better

1. **Generic**: Works for both line-level and paragraph-level data
2. **Accurate**: The column contains the annotated text value, not specifically paragraphs
3. **Consistent**: Matches the V2 API naming conventions
4. **Clear**: Describes what the column contains (a value with annotations)

1.4 Files Modified

1.4.1 1. jupyter/ist769_skol.ipynb

Changed cells: - Cell id="73ecd132-4946-47cf-a67b-364335eb768b": Display predictions - Cell id="db747262-98fb-4166-abb0-e55966e538c9": Filter for "nov." in nomenclature

Before:

```
predictions.select("predicted_label", "annotated_pg").where('predicted_label = "
predictions.select("*").filter(col("annotated_pg").contains("nov.")).where("pred
```

After:

```
predictions.select("predicted_label", "annotated_value").where('predicted_label
predictions.select("*").filter(col("annotated_value").contains("nov.")).where("p
```

1.4.2 2. skol_classifier/classifier.py

Changed locations: - Line 332: Column name in paragraph extraction - Lines 360-363: Aggregation expressions - Line 428: Column name in annotation formatting - Lines 456-459: Aggregation expressions - Line 821: Column name in CouchDB save - Lines 933-936: Aggregation expressions

Before:

```
"annotated_pg",
...
expr("sort_array(collect_list(struct(row_number, annotated_pg))) AS sorted_list"
.withColumn("annotated_pg_ordered", expr("transform(sorted_list, x -> x.annotate
.withColumn("final_aggregated_pg", expr("array_join(annotated_pg_ordered, '\n')"
```

After:

```
"annotated_value",
...
expr("sort_array(collect_list(struct(row_number, annotated_value))) AS sorted_li
.withColumn("annotated_value_ordered", expr("transform(sorted_list, x -> x.annot
.withColumn("final_aggregated_pg", expr("array_join(annotated_value_ordered, '\n
```

Note: `final_aggregated_pg` name was kept for backward compatibility with CouchDB save methods.

1.4.3 3. COUCHDB_INTEGRATION.md

Changed documentation: - Line 131: Updated parameter description

Before:

```
- `predictions`: DataFrame with predictions (must include `annotated_pg` column)
```

After:

```
- `predictions`: DataFrame with predictions (must include `annotated_value` colu
```

1.5 Column Naming Summary

After this change, the column naming is:

Column Name	Purpose	Used In
`value`	Raw text content	Input data (lines or paragraphs)
`annotated_value`	YEDDA-formatted annotation	Output from formatters
`predicted_label`	Predicted class label	Model predictions
`final_aggregated_pg`	Aggregated annotations for save	CouchDB save operations

1.6 Impact

1.6.1 Before (Inconsistent)

```
# Some code expected annotated_pg
predictions.select("annotated_pg").show()  # ❌ Column doesn't exist

# Other code used annotated_value
formatter.format(predictions)  # Returns annotated_value column
```

1.6.2 After (Consistent)

```
# All code uses annotated_value
predictions.select("annotated_value").show()  # ✅ Works

formatter.format(predictions)  # Returns annotated_value column
```

1.7 Example Usage

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

spark = SparkSession.builder.getOrCreate()

classifier = SkolClassifierV2(
    spark=spark,
    input_source='files',
    file_paths=['data/*.ann'],
    line_level=True,
    output_format='annotated',
    model_type='logistic'
)

results = classifier.fit()
predictions = classifier.predict()

# Show annotated values
predictions.select("value", "predicted_label", "annotated_value").show(10)

# Filter by content in annotations
nomenclature = predictions.filter(
    col("annotated_value").contains("nov.")
).where("predicted_label = 'Nomenclature'")

nomenclature.count()
```

1.8 Related Issues Fixed

This fix resolves: - ✅ Consistent column naming across codebase - ✅ Notebook cells that reference annotated predictions - ✅ Documentation accuracy - ✅ Confusion between line-level and paragraph-level naming

1.9 Notes

- The `final_aggregated_pg` column name is kept for CouchDB operations for backward compatibility
- All user-facing code now uses `annotated_value`
- The change is purely cosmetic and doesn't affect functionality
- Old code referencing `annotated_pg` will need to be updated to use `annotated_value`

1.10 Migration Guide

If you have existing code that uses `annotated_pg`, update it:

Before:

```
# Old code
df.select("annotated_pg")
df.filter(col("annotated_pg").contains("text"))
df.withColumn("new_col", col("annotated_pg"))
```

After:

```
# New code
df.select("annotated_value")
df.filter(col("annotated_value").contains("text"))
df.withColumn("new_col", col("annotated_value"))
```