import random

import string

from pyspark.sql import SparkSession

from pyspark.sql.functions import udf, col, when, row\_number

from pyspark.sql.window import Window

from pyspark.sql.types import StringType

# Initialize Spark session

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

# Sample Data

data = [("OTHER",), ("SAMPLE",), ("TEST",), ("DATA",), ("EXAMPLE",), ("HELLO",), ("WORLD",)]

df = spark.createDataFrame(data, ["cleaned\_name"])

# Function to introduce minor typos or completely change a word

def introduce\_typos(word, change\_type):

"""Introduce minor typos for 'Similar' or completely change for 'Dissimilar'."""

if not word or len(word) < 2:

return word

word = list(word)

if change\_type == "same":

return "".join(word) # No change

elif change\_type == "similar":

# Introduce minor typo (swap two adjacent letters)

idx = random.randint(0, len(word) - 2)

word[idx], word[idx + 1] = word[idx + 1], word[idx] # Swap adjacent letters

return "".join(word)

elif change\_type == "dissimilar":

# Completely change the word but keep the same length

return "".join(random.choices(string.ascii\_uppercase, k=len(word)))

return "".join(word)

# Register as PySpark UDF

introduce\_typos\_udf = udf(introduce\_typos, StringType())

# Assign a row number to distribute changes equally

window\_spec = Window.orderBy("cleaned\_name") # Ensures fair distribution

df = df.withColumn("row\_number", row\_number().over(window\_spec))

# Apply transformations to create modified names

df = df.withColumn(

"cleaned\_name\_modified",

when((col("row\_number") % 3 == 0), introduce\_typos\_udf(col("cleaned\_name"), "same")) # No change

.when((col("row\_number") % 3 == 1), introduce\_typos\_udf(col("cleaned\_name"), "similar")) # Minor typo

.otherwise(introduce\_typos\_udf(col("cleaned\_name"), "dissimilar")) # Completely change

)

# Assign classification labels based on changes

df = df.withColumn(

"classification\_final",

when(col("cleaned\_name\_modified") == col("cleaned\_name"), "Same")

.when(col("cleaned\_name\_modified") != col("cleaned\_name"), "Similar")

.when(col("cleaned\_name\_modified").rlike("^[A-Z]+$"), "Dissimilar") # If word is fully randomized

.otherwise("Unknown")

)

# Drop temporary row\_number column

df = df.drop("row\_number")

# Show results

df.select("cleaned\_name", "cleaned\_name\_modified", "classification\_final").show(truncate=False)