```
#Running on Colab
!pip install pyspark
!pip install -U -q PyDrive
!apt install openjdk-8-jdk-headless -qq
import os
os.environ['JAVA_HOME'] = '/usr/lib/jvm/java-8-openjdk-amd64'
```

```
Collecting pyspark
  Downloading pyspark-3.5.1.tar.gz (317.0 MB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 317.0/317.0 MB 2.3 MB/s eta 0:00:00
  Preparing metadata (setup.py) ... done
Requirement already satisfied: py4j==0.10.9.7 in /usr/local/lib/python3.10/dist-packages (from pyspark) (0.10.9.7)
Building wheels for collected packages: pyspark
  Building wheel for pyspark (setup.py) ... done
  Created wheel for pyspark: filename=pyspark-3.5.1-py2.py3-none-any.whl size=317488491 sha256=ecbb498bb31b8c35009f4ac8283187283f908bf4d9d837aa09aca2b6898363f9
  Stored in directory: /root/.cache/pip/wheels/80/1d/60/2c256ed38dddce2fdd93be545214a63e02fbd8d74fb0b7f3a6
Successfully built pyspark
Installing collected packages: pyspark
Successfully installed pyspark-3.5.1
The following additional packages will be installed:
  libxtst6 openjdk-8-jre-headless
Suggested packages:
  openjdk-8-demo openjdk-8-source libnss-mdns fonts-dejavu-extra fonts-nanum fonts-ipafont-gothic
  fonts-ipafont-mincho fonts-wqy-microhei fonts-wqy-zenhei fonts-indic
The following NEW packages will be installed:
  libxtst6 openjdk-8-jdk-headless openjdk-8-jre-headless
0 upgraded, 3 newly installed, 0 to remove and 45 not upgraded.
Need to get 39.7 MB of archives.
After this operation, 144 MB of additional disk space will be used.
Selecting previously unselected package libxtst6:amd64.
(Reading database ... 121918 files and directories currently installed.)
Preparing to unpack .../libxtst6_2%3a1.2.3-1build4_amd64.deb ...
Unpacking libxtst6:amd64 (2:1.2.3-1build4) ...
Selecting previously unselected package openjdk-8-jre-headless:amd64.
Preparing to unpack .../openjdk-8-jre-headless_8u402-ga-2ubuntu1~22.04_amd64.deb ...
Unpacking openjdk-8-jre-headless:amd64 (8u402-ga-2ubuntu1~22.04) ...
Selecting previously unselected package openjdk-8-jdk-headless:amd64.
Preparing to unpack .../openjdk-8-jdk-headless_8u402-ga-2ubuntu1~22.04_amd64.deb ...
Unpacking openjdk-8-jdk-headless:amd64 (8u402-ga-2ubuntu1~22.04) ...
Setting up libxtst6:amd64 (2:1.2.3-1build4) ...
Setting up openjdk-8-jre-headless:amd64 (8u402-ga-2ubuntu1~22.04) ...
update-alternatives: using /usr/lib/jvm/java-8-openjdk-amd64/jre/bin/orbd to provide /usr/bin/orbd (orbd) in auto mode
update-alternatives: using /usr/lib/jvm/java-8-openjdk-amd64/jre/bin/servertool to provide /usr/bin/servertool (servertool) in auto mode
update-alternatives: using /usr/lib/jvm/java-8-openjdk-amd64/jre/bin/tnameserv to provide /usr/bin/tnameserv (tnameserv) in auto mode
Setting up openjdk-8-jdk-headless:amd64 (8u402-ga-2ubuntu1~22.04) ...
update-alternatives: using /usr/lib/jvm/java-8-openjdk-amd64/bin/clhsdb to provide /usr/bin/clhsdb (clhsdb) in auto mode
update-alternatives: using /usr/lib/jvm/java-8-openjdk-amd64/bin/extcheck to provide /usr/bin/extcheck (extcheck) in auto mode
update-alternatives: using /usr/lib/jvm/java-8-openjdk-amd64/bin/hsdb to provide /usr/bin/hsdb (hsdb) in auto mode
update-alternatives: using /usr/lib/jvm/java-8-openjdk-amd64/bin/idlj to provide /usr/bin/idlj (idlj) in auto mode
update-alternatives: using /usr/lib/jvm/java-8-openjdk-amd64/bin/javah to provide /usr/bin/javah (javah) in auto mode
update-alternatives: using /usr/lib/jvm/java-8-openjdk-amd64/bin/jhat to provide /usr/bin/jhat (jhat) in auto mode
update-alternatives: using /usr/lib/jvm/java-8-openjdk-amd64/bin/jsadebugd to provide /usr/bin/jsadebugd (jsadebugd) in auto mode
update-alternatives: using /usr/lib/jvm/java-8-openjdk-amd64/bin/native2ascii to provide /usr/bin/native2ascii (native2ascii) in auto mode
update-alternatives: using /usr/lib/jvm/java-8-openjdk-amd64/bin/schemagen to provide /usr/bin/schemagen (schemagen) in auto mode
update-alternatives: using /usr/lib/jvm/java-8-openjdk-amd64/bin/wsgen to provide /usr/bin/wsgen (wsgen) in auto mode
update-alternatives: using /usr/lib/jvm/java-8-openjdk-amd64/bin/wsimport to provide /usr/bin/wsimport (wsimport) in auto mode
update-alternatives: using /usr/lib/jvm/java-8-openjdk-amd64/bin/xjc to provide /usr/bin/xjc (xjc) in auto mode
Processing triggers for libc-bin (2.35-0ubuntu3.4) ...
/sbin/ldconfig.real: /usr/local/lib/libtbb.so.12 is not a symbolic link

/sbin/ldconfig.real: /usr/local/lib/libtbbbind_2_0.so.3 is not a symbolic link

/sbin/ldconfig.real: /usr/local/lib/libtbbmalloc.so.2 is not a symbolic link

/sbin/ldconfig.real: /usr/local/lib/libtbbbind.so.3 is not a symbolic link
```

```
#  Importing Required Libraries
import pyspark
from pyspark.sql import *
from pyspark.sql.functions import *
from pyspark import SparkContext, SparkConf

# Create Spark session and ContextRun PySpark.
# create the session
conf = SparkConf().set("spark.ui.port","4050")
# create the context
sc = pyspark.SparkContext(conf=conf)
spark = SparkSession.builder.appName("DataFrame").config('spark.ui.port', '4050').getOrCreate()
spark
```

SparkSession - in-memory

SparkContext

Spark UI

Version
v3.5.1
Master
local[*]
AppName
pyspark-shell

STEP 1: Shingling: Convert documents to sets

```
import hashlib
from pyspark.sql import SparkSession

HugeDocuments = sc.parallelize([
    'D1,Pretend this is a very huge document about data science and machine learning',
    'D2,Pretend this is another very huge document about big data and its applications',
    'D3,Mining of Massive Datasets is a course focusing on techniques for handling large-scale data'
])

K = 5  # Shingle length (9 is Ideal for large documents)
buckets = 30 # Number of buckets for hashing (increased for better distribution)

# Function to hash a shingle
def hashIt(shingle):
    hashObj = hashlib.sha256(shingle.encode())
    hashed = int.from_bytes(hashObj.digest(), byteorder='big') % buckets
    return hashed

# Function to generate shingles from a document
def getShingles(line):
    global K
    documentNumber, text = line.split(',', 1) #Split the string only at the first comma hence the ,1 this is to avoid in text commas split
    text = text.lower()
    setOfShingles = set()
    for i in range(len(text) - K + 1):
        shingle = text[i:i + K]
        hashedShingle = hashIt(shingle)
        setOfShingles.add(hashedShingle)
    return documentNumber, setOfShingles

# Generating the shingles
shingles = HugeDocuments.map(lambda x: getShingles(x))

# Removing duplicate shingles
uniqueShingles = shingles.flatMap(lambda x: x[1]).distinct().collect()

# Dictionary to map each shingle to an index
shingleIndex = {shingle: i for i, shingle in enumerate(uniqueShingles)}

# Map documents to shingle indices
docShin = shingles.map(lambda x: (x[0], [shingleIndex[shingle] for shingle in x[1]]))

# Create a sparse matrix
sparseM = docShin.flatMapValues(lambda x: x).map(lambda x: (x, 1)).reduceByKey(lambda x, y: x).sortByKey()

# Convert sparse matrix to DataFrame
dfData = sparseM.map(lambda x: (x[0][0], x[0][1], x[1])).toDF(["Document", "Shingle", "Value"])

# Pivot the DataFrame to create the boolean matrix
pivotedDf = dfData.groupby("Document").pivot("Shingle").agg({"Value": "max"}).fillna(0)

# Show the boolean matrix
pivotedDf.show()
```

```
+--------+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
|Document|  0|  1|  2|  3|  4|  5|  6|  7|  8|  9| 10| 11| 12| 13| 14| 15| 16| 17| 18| 19| 20| 21| 22| 23| 24| 25| 26| 27| 28| 29|
+--------+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
|      D1|  1|  1|  1|  1|  1|  1|  1|  1|  1|  1|  1|  1|  1|  0|  1|  1|  1|  1|  1|  1|  1|  1|  1|  1|  1|  1|  1|  1|  1|  0|
|      D3|  1|  1|  1|  1|  0|  1|  1|  1|  1|  1|  1|  1|  1|  1|  1|  1|  1|  1|  1|  1|  1|  1|  1|  1|  1|  1|  1|  1|  1|  1|
|      D2|  1|  0|  1|  1|  1|  1|  1|  1|  1|  1|  1|  1|  1|  1|  0|  1|  1|  0|  1|  1|  1|  1|  1|  1|  1|  0|  1|  1|  1|  1|
+--------+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
```

Step 2: Min-Hashing: Convert large sets to short signatures

```
# Efficient MinHashing using multiple hash functions
numHashFunctions = 100

# Generate hash functions
def create_hash_functions(num_funcs, max_shingle):
    def make_hash_function(seed):
        random.seed(seed)
        a = random.randint(1, max_shingle)
        b = random.randint(0, max_shingle)
        return lambda x: (a * x + b) % max_shingle
    return [make_hash_function(i) for i in range(num_funcs)]

hash_functions = create_hash_functions(numHashFunctions, len(uniqueShingles))

# Function to update the signature matrix with min-hash values
def updateSignature(row, hash_funcs):
    document, shingleIndices = row
```

```
    document, shingleIndices = row
    updatedSig = [float('inf')] * len(hash_funcs)
    for shingleIndex in shingleIndices:
        for i, hash_func in enumerate(hash_funcs):
            hashed_val = hash_func(shingleIndex)
            if hashed_val < updatedSig[i]:
                updatedSig[i] = hashed_val
    return document, updatedSig

# Update signature matrix with min-hash
signatureMatrix = docShin.map(lambda x: updateSignature(x, hash_functions))
signatureDF = signatureMatrix.toDF(["Document", "Signature"])
signatureDF.show(truncate=False)
```

```
+--------+------------------------------------------------------------------------------------------------------------------------------------------------------------------
|Document|Signature
+--------+------------------------------------------------------------------------------------------------------------------------------------------------------------------
|D1      |[0, 3, 0, 0, 3, 8, 0, 0, 1, 4, 0, 12, 0, 0, 1, 0, 3, 0, 3, 1, 0, 1, 4, 24, 0, 0, 5, 0, 1, 2, 1, 0, 2, 0, 0, 4, 0, 1, 1, 0, 3, 0, 0, 1, 0, 1, 0, 2, 4, 2, 1, 0, 1, 7,
|D2      |[0, 3, 0, 0, 1, 8, 0, 0, 1, 4, 0, 12, 0, 0, 1, 0, 3, 1, 3, 1, 0, 1, 4, 24, 0, 0, 5, 0, 1, 2, 1, 0, 2, 2, 1, 4, 0, 1, 1, 0, 3, 0, 0, 1, 0, 1, 0, 2, 4, 2, 1, 0, 1, 7,
|D3      |[0, 3, 0, 0, 1, 8, 0, 0, 1, 4, 0, 12, 0, 0, 1, 0, 3, 0, 3, 1, 0, 1, 4, 24, 0, 0, 5, 0, 1, 2, 1, 0, 2, 0, 0, 4, 0, 1, 1, 0, 3, 0, 0, 1, 0, 1, 0, 2, 4, 2, 1, 0, 1, 7,
+--------+------------------------------------------------------------------------------------------------------------------------------------------------------------------
```

Step 3: Locality-Sensitive Hashing:

```
# Number of bands and rows per band
num_bands = 10
rows_per_band = len(hash_functions) // num_bands

# Function to hash a band
def hash_band(band):
    return hashlib.md5(band.encode()).hexdigest()

# Create candidate pairs using LSH
def lsh(signature, num_bands, rows_per_band):
    candidates = set()
    for band_idx in range(num_bands):
        band = signature[band_idx * rows_per_band: (band_idx + 1) * rows_per_band]
        band_str = ','.join(map(str, band))
        band_hash = hash_band(band_str)
        candidates.add((band_hash, band_idx))
    return candidates

# Apply LSH to the signature matrix
lsh_rdd = signatureMatrix.flatMapValues(lambda x: lsh(x, num_bands, rows_per_band))

# Group by band hash to find candidate pairs
candidate_pairs = lsh_rdd.map(lambda x: (x[1], x[0])).groupByKey().flatMap(lambda x: combinations(x[1], 2)).distinct()

# Collect candidate pairs
candidate_pairs_collected = candidate_pairs.collect()

# Calculate Jaccard similarity for candidate pairs
def jaccard_similarity(pair, signature_matrix):
    doc1, doc2 = pair
    sig1 = signature_matrix[doc1]
    sig2 = signature_matrix[doc2]
    intersection = 0
    for i in range(len(sig1)):
        if sig1[i] == sig2[i]:
            intersection += 1
    union = len(sig1) #length of either signature vector (since both vectors are of the same length).
    return intersection / union

# Convert signature matrix to a dictionary for fast access
signature_matrix_dict = {row[0]: row[1] for row in signatureMatrix.collect()}

# Calculate and filter pairs with high similarity
threshold = 0.95
similar_pairs = [(pair, jaccard_similarity(pair, signature_matrix_dict)) for pair in candidate_pairs_collected if jaccard_similarity(pair, signature_matrix_dict

# Display similar pairs
print("Documents with similarity greater than 95 percent:", similar_pairs)
```

```
Documents with similarity greater than 95 percent: [(('D1', 'D3'), 0.96), (('D1', 'D2'), 0.96)]
```