databricks Programming Assignment 1

(https://databricks.com)

1

```
from pyspark.sql.functions import lower, regexp_replace, split, explode, col, udf
from pyspark.sql.types import ArrayType, StringType
```

2

```
# Step 1: Load the Data
fp = 'dbfs:/FileStore/tables/programming_assignment_1/Class_6___Text_File_for_Letter_Pairs.txt'
df = spark.read.text(fp)
```

3

```
# Verify the text was loaded one line at a time into a dataframe
display(df.head(20))
```

Table

	value
1	
2	Data have become a torrent flowing into every area of the global economy.1 Companies churn out a
3	burgeoning volume of transactional data, capturing trillions of bytes of information about their
4	customers, suppliers, and operations. millions of networked sensors are being embedded in the
5	physical world in devices such as mobile phones, smart energy meters, automobiles, and industrial
6	machines that sense, create, and communicate data in the age of the Internet of Things.2 Indeed, as
7	companies and organizations go about their business and interact with individuals, they are
8	generating a tremendous amount of digital “exhaust data,” i.e., data that
9	are created as a by-product of other activities. Social media sites, smartphones, and other
10	consumer devices including PCs and laptops have allowed billions of individuals around the world to
11	contribute to the amount of big data available. And the growing volume of multimedia content has
12	played a major role in the exponential
13	growth in the amount of big data (see Box 1, “What do we mean by ‘big data’?”). Each second of
14	high-definition video, for example, generates more than 2,000 times as many bytes as required to
15	store a single page of text. In a digitized world, consumers going about their day—communicating,

20 rows

4

```
# Step 2: Preprocess the Text
df = df.withColumn('text_lower', lower(col('value')))
df = df.withColumn('text_letters_only', regexp_replace(col('text_lower'), '[^a-z]', ' '))
df = df.withColumn('words', split(col('text_letters_only'), ' '))
words_df = df.select(explode(col('words')).alias('word'))
words_df = words_df.filter(col('word') != '')
```

5

```
# Verify the words have been properly preprocessed
display(words_df.head(20))
```

Table

	word

1	data
2	have
3	become
4	a
5	torrent
6	flowing
7	into
8	every
9	area
10	of
11	the
12	global
13	economy
14	companies
15	churn

20 rows

6

```
# Step 3: Extract Bi-grams
def get_bigrams(word):
    word = word.strip()
    bigrams = []
    if len(word) >= 2:
        for i in range(len(word) - 1):
            bigram = word[i:i+2]
            bigrams.append(bigram)
    return bigrams

get_bigrams_udf = udf(get_bigrams, ArrayType(StringType()))
bigrams_df = words_df.withColumn('bigrams', get_bigrams_udf(col('word')))
bigrams_exploded_df = bigrams_df.select(explode(col('bigrams')).alias('bigram'))
```

7

```
# Verify that each word was broken into its bigrams properly
display(bigrams_df.head(20))
```

Table

	word	bigrams
1	data	["da","at","ta"]
2	have	["ha","av","ve"]
3	become	["be","ec","co","om","me"]
4	a	[]
5	torrent	["to","or","rr","re","en","nt"]
6	flowing	["fl","lo","ow","wi","in","ng"]
7	into	["in","nt","to"]
8	every	["ev","ve","er","ry"]
9	area	["ar","re","ea"]
10	of	["of"]
11	the	["th","he"]
12	global	["gl","lo","ob","ba","al"]
13	economy	["ec","co","on","no","om","my"]
14	companies	["co","om","mp","pa","an","ni","ie","es"]
15	churn	["ch","hu","ur","rn"]

20 rows

8

```
# Verify that the bigrams_exploded_df was properly mapped from the bigrams_df
display(bigrams_exploded_df.head(20))
```

Table

	bigram
1	da
2	at
3	ta
4	ha
5	av
6	ve
7	be
8	ec
9	co
10	om
11	me
12	to
13	or
14	rr
15	re

20 rows

9

```
# Step 4: Count Bi-gram Frequencies
bigram_counts = bigrams_exploded_df.groupBy('bigram').count()
```

10

```
# Verify the counts are being done properly
display(bigram_counts.head(20))
```

Table

	bigram	count
1	ct	30
2	ld	16
3	ey	7
4	rf	1
5	en	73
6	du	13
7	ye	5
8	ss	14
9	pi	3
10	pu	12
11	cc	3
12	cr	16
13	ei	13
14	pr	15
15	us	28

20 rows

11

```
# Step 5: Find the Top 5 Most and Least Frequent Bi-grams
# Top 5 Most Frequent Bi-grams
top5 = bigram_counts.orderBy(col('count').desc()).limit(5)
print("Top 5 Most Frequent Bi-grams:")
top5.show(truncate=False)
```

```
Top 5 Most Frequent Bi-grams:
+------+-----+
|bigram|count|
+------+-----+
|th    |146  |
|in    |128  |
|an    |127  |
|at    |126  |
|re    |102  |
+------+-----+
```

12

```
# Top 5 Least Frequent Bi-grams
bottom5 = bigram_counts.orderBy(col('count').asc()).limit(5)
print("Top 5 Least Frequent Bi-grams:")
bottom5.show(truncate=False)
```

```
Top 5 Least Frequent Bi-grams:
+------+-----+
|bigram|count|
+------+-----+
|uf    |1    |
|ju    |1    |
|lw    |1    |
|ft    |1    |
|cs    |1    |
+------+-----+
```

13