Department of Computer Science and Engineering (Data Science)

Machine Learning – IV

Experiment 1

Name: Umang Kirit Lodaya **SAP ID: 60009200032**

Batch: D11

Aim:

To implement and understand the Word Count algorithm using the MapReduce programming model, a fundamental application for processing and analyzing large text datasets in a parallel and distributed manner.

Theory:

Introduction to MapReduce:

- MapReduce is a programming model designed for processing and generating large datasets in parallel across distributed clusters.
- It consists of two main phases: Map and Reduce.

Word Count Overview:

- **Map Step:**
  - Mapper Function:
    - For each input record (line of text), the mapper function emits key-value pairs.
    - The key is the word, and the value is the count (usually 1).
- **Shuffling and Sorting:**
  - The MapReduce framework automatically shuffles and sorts the emitted key-value pairs, grouping them by key.
- **Reduce Step:**
  - Reducer Function:
    - For each group of values with the same key, the reducer function sums the counts to get the total occurrences of each word.

Step-by-Step Implementation:

- **Step 1: Mapper Function**
  - Implement the mapper function to tokenize words and emit key-value pairs.
- **Step 2: Shuffling and Sorting**
  - Understand the automatic shuffling and sorting mechanism of the MapReduce framework.
- **Step 3: Reducer Function**
  - Implement the reducer function to aggregate word counts.
- **Step 4: Input Data Formatting**
  - Prepare input data in a format suitable for MapReduce processing (e.g., a collection of text documents).
- **Step 5: Experimentation with Input Sizes**
  - Experiment with different sizes of input text datasets to observe the scalability of the MapReduce approach.

Implementation Tips:

- Handle special cases like punctuation, case sensitivity, and stemming appropriately.
- Optimize the mapper and reducer functions for efficient word counting.

Lab Experiments to be Performed in This Session:

Execute the Word Count using Map Reduce on a dataset to gain insights into its functionality and operation.

NAME: UMANG KIRIT LODAYA

SAP ID: 60009200032

BATCH: K1 / D11

AIM: WORD COUNT USING MAP REDUCE

In [1]:
```
import os
```

In [2]:
```
PATH = '/content/drive/MyDrive/Colab Notebooks/ML - IV'
```

In [3]:
```
FILES = []
for x in os.listdir(PATH):
    if x.startswith("file"):
        FILES.append(x)
```

In [4]:
```
FILES
```

Out[4]:
```
['file.txt', 'file2.txt']
```

In [5]:
```
FILE = []
for fil in FILES:
    with open(PATH + f'/{fil}', 'r') as f:
        FILE.extend(f.readlines())
```

In [6]:
```
F = []
for i in range(len(FILE)):
    sent = FILE[i].lower().split()
    new = []
    for word in sent:
        n = ''
        for char in word:
            if char.isalnum():
                n += char

        new.append(n)

    F.extend(new)
```

In [7]:
```
with open(PATH + '/mapper.txt', 'w') as f:
    for word in F:
        f.write(f"{word},1\n")
```

SAMPLE OUTPUT OF MAPPER

mapper.txt

```
the,1
first,1
sentence,1
establishes,1
the,1
key,1
idea,1
of,1
balance,1
theory,1
the,1
next,1
sentence,1
begins,1
with,1
balance,1
theory,1
and,1
ends,1
with,1
social,1
ties,1
which,1
is,1
the,1
focus,1
of,1
the,1
```

In [8]:

```python
MAP_OUT = None
with open(PATH + '/mapper.txt', 'r') as f:
    MAP_OUT = f.readlines()

SHUFFLE = sorted(MAP_OUT)

with open(PATH + '/shuffle.txt', 'w') as f:
    for word in SHUFFLE:
        f.write(f"{word}")
```

SAMPLE OUTPUT OF SHUFFLER

mapper.txt shuffle.txt

```
a,1
a,1
a,1
a,1
a,1
a,1
about,1
and,1
and,1
and,1
and,1
awkward,1
balance,1
balance,1
balance,1
balance,1
be,1
before,1
before,1
begins,1
begins,1
benefits,1
but,1
but,1
choose,1
choppy,1
cliques,1
coherent,1
cohesion,1
comes,1
```

In [9]:

```python
SHUFFLE = None
with open(PATH + '/shuffle.txt', 'r') as f:
    SHUFFLE = f.readlines()

freq = {}

for word in SHUFFLE:
    word = word[:-1]
    word, count = word.split(',')
    count = int(count)
    freq[word] = freq.get(word, 0) + 1
```

In [10]:

```python
with open(PATH + '/reducer.txt', 'w') as f:
    for key in freq:
        f.write(f"{key},{freq[key]}\n")
```

SAMPLE OUTPUT OF REDUCER

mapper.txt shuffle.txt reducer.txt

```
a,6
about,1
and,4
awkward,1
balance,4
be,1
before,2
begins,2
benefits,1
but,2
choose,1
choppy,1
cliques,1
coherent,1
cohesion,1
comes,1
concept,3
concepts,1
connects,1
consider,1
consistent,1
contrast,1
could,1
daisychain,1
discuss,1
dont,1
each,1
ends,1
ensure,1
establishes,1
```