School of Information Technology and Electrical Engineering
INFS3208 – Cloud Computing

Programming Assignment Task III (10 Marks)

Task description:

In this assignment, you are asked to write a piece of Spark code to **count occurrences of verbs in the UN debates and find the most similar debate contents. The returned result should be the top 10 verbs that are most frequently used in all debates and the debate that is most similar to the one we provide.** This assignment is to test your ability to use transformation and action operations in Spark RDD programming and your understanding of Vector Database. You will be given three files, including a UN General Debates dataset (`un-general-debates.csv`), a verb list (`all_verbs.txt`) and a verb dictionary file (`verb_dict.txt`). These source files are expected to be stored in a HDFS. You can choose either Scala or Python to complete this assignment in the Jupyter Notebook. There are some technical requirements in your code submission as follows:

Objectives:

1. **Read Source Files from HDFS and Create RDDs (1.5 marks):**
    - Read the UN General Debates dataset (`un-general-debates.csv`) from HDFS and convert only the "text" column into an RDD. Details of `un-general-debates.csv` are provided in the Preparation section below (1 mark).
    - Read the verb list file (`all_verbs.txt`) and verb dictionary file (`verb_dict.txt`) from HDFS and load them into separate RDDs (0.5 marks).
    - Note: If you failed to read files from HDFS, you can still read them from the local file system in `work/nbs/` and complete the following tasks.
2. **Use Learned RDD Operations to Preprocess the Debate Texts (3 marks):**
    - Remove empty lines (0.5 marks).
    - Remove punctuations that could attach to the verbs (0.5 marks).
        - E.g., "work," and "work" will be counted differently, if you DO NOT remove the punctuation.
    - Change the capitalization or case of text (0.5 marks).
        - E.g., "WORK", "Work" and "work" will be counted as three different verbs, if you DO NOT make all of them in lower-case.
    - Find all verbs in the RDD by matching the words in the given verb list (`all_verbs.txt`) (0.5 mark).
    - Convert all verbs in different tenses into the simple present tense by looking up the verbs in the verb dictionary list (`verb_dict.txt`) (1 mark).
        - E.g., regular verb: "work" - works", "worked", and "working".
        - E.g., irregular verb: "begin" - "begins", "began", and "begun".

- E.g., linking verb “be” and its various forms, including “is”, “am”, “are”, “was”, “were”, “being” and “been”.
- E.g., (work, 100), (works,50), (working,150) should be counted as (work, 300).

3. **Use learned RDD Operations to Count Verb Frequency (3 marks):**
   - Count the **top 10** frequently used verbs in UN debates (2 marks).
   - Display the results in the format (“verb1”, count1), (“verb2”, count2), … and in a descending order of the counts (1 marks).
4. **Use Vector Database (Faiss) to Find the Most Similar Debate (2.5 marks):**
   - Convert the original debates into vectors and store them in a proper Index (1.5 mark).
   - Search the debate content that has the most similar idea to “Global climate change is both a serious threat to our planet and survival.” (1 mark)

Preparation:

In this individual coding assignment, you will apply your knowledge of Vector Database, Spark, Spark RDD Programming and HDFS (in Lectures 7-10). Firstly, you should read **Task Description** to understand what the task is and what the technical requirements include. Secondly, you should review the creation and usage of Faiss, transformations and actions in Spark, and usage of HDFS in Lectures and Practicals 7-10. In the Appendix, there are some transformation and action operations you could use in this assignment. Lastly, you need to write the code (Scala or Python) in the Jupyter Notebook. **All technical requirements need to be fully met to achieve full marks.** You can either practice on the GCP’s VM or your local machine with Oracle Virtualbox if you are unable to access GCP. Please read the **Example of writing Spark code** below to have more details.

Assignment Submission:

- You need to compress only the Jupyter Notebook (.ipynb) file.
- The name of the compressed file should be named “FirstName_LastName_StudentNo.zip”.
- You must make an online submission to Blackboard **before 3:00 PM on Friday, 17/10/2025**
- Only one extension application could be approved due to medical conditions.

Main Steps:

Step 1:

Log in your VM instance and change to your home directory. We recommend using a VM instance with at least 4 vCPUs, 8G memory and 20GB free disk space.

Step 2:

```
git clone https://github.com/csenw/cca3.git && cd cca3
```

Run these commands to download the required docker-compose.yml file and configuration files.

Step 3:

```
sudo chmod -R 777 nbs/
docker-compose up -d
```

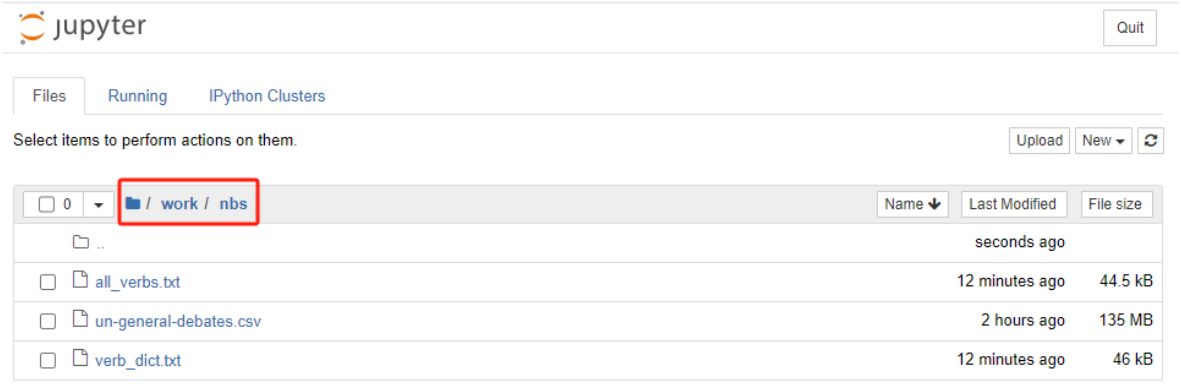
Run all the containers using docker-compose

```
Creating namenode ... done
Creating datanode1    ... done
Creating datanode2    ... done
Creating datanode3    ... done
Creating spark-master ... done
Creating spark-worker-2  ... done
Creating spark-worker-1  ... done
Creating jupyternb       ... done
Creating resourcemanager ... done
Creating nodemanager     ... done
Creating historyserver   ... done
```

```
CONTAINER ID   IMAGE                                                COMMAND                  CREATE
D          STATUS                  PORTS
                           NAMES
1685ccb20b75   bde2020/hadoop-nodemanager:2.0.0-hadoop3.2.1-java8    "/entrypoint.sh /run…"   45 sec
onds ago   Up 42 seconds (healthy)   8042/tcp
                           nodemanager
370ee611e81a   bde2020/hadoop-historyserver:2.0.0-hadoop3.2.1-java8   "/entrypoint.sh /run…"   45 sec
onds ago   Up 42 seconds (healthy)   8188/tcp
                           historyserver
5856b4cd6473   jupyter/all-spark-notebook:42f4c82a07ff               "tini -g -- bash -c …"   45 sec
onds ago   Up 42 seconds            0.0.0.0:8888->8888/tcp, :::8888->8888/tcp
                           jupyternb
b3440c5ee53f   bde2020/spark-worker:3.0.0-hadoop3.2                  "/bin/bash /worker.sh"   45 sec
onds ago   Up 42 seconds            0.0.0.0:8081->8081/tcp, :::8081->8081/tcp
                           spark-worker-1
953c22a46997   bde2020/spark-worker:3.0.0-hadoop3.2                  "/bin/bash /worker.sh"   45 sec
onds ago   Up 43 seconds            8081/tcp, 0.0.0.0:8082->8082/tcp, :::8082->8082/tcp
                           spark-worker-2
d4ccf41e661e   bde2020/spark-master:3.0.0-hadoop3.2                  "/bin/bash /master.sh"   46 sec
onds ago   Up 45 seconds            0.0.0.0:7077->7077/tcp, :::7077->7077/tcp, 6066/tcp, 0.0.0.0:808
0->8080/tcp, :::8080->8080/tcp   spark-master
a550e5ed40ca   bde2020/hadoop-datanode:2.0.0-hadoop3.2.1-java8       "/entrypoint.sh /run…"   46 sec
onds ago   Up 44 seconds (healthy)   9864/tcp
                           datanode3
ea90917bce4a   bde2020/hadoop-datanode:2.0.0-hadoop3.2.1-java8       "/entrypoint.sh /run…"   46 sec
onds ago   Up 45 seconds (healthy)   9864/tcp
                           datanode1
f1c2b0adea8f   bde2020/hadoop-datanode:2.0.0-hadoop3.2.1-java8       "/entrypoint.sh /run…"   46 sec
onds ago   Up 44 seconds (healthy)   9864/tcp
                           datanode2
386be288f029   bde2020/hadoop-namenode:2.0.0-hadoop3.2.1-java8       "/entrypoint.sh /run…"   51 sec
onds ago   Up 45 seconds (healthy)   0.0.0.0:9000->9000/tcp, :::9000->9000/tcp, 0.0.0.0:80->9870/tcp,
 :::80->9870/tcp               namenode
```

Step 4:

Open the Jupyter Notebook (`http://external_IP:8888`) and you can find all the files under the `work/nbs/` folder. This is also the folder where you should write the notebook (.ipynb) file.



Step 5:

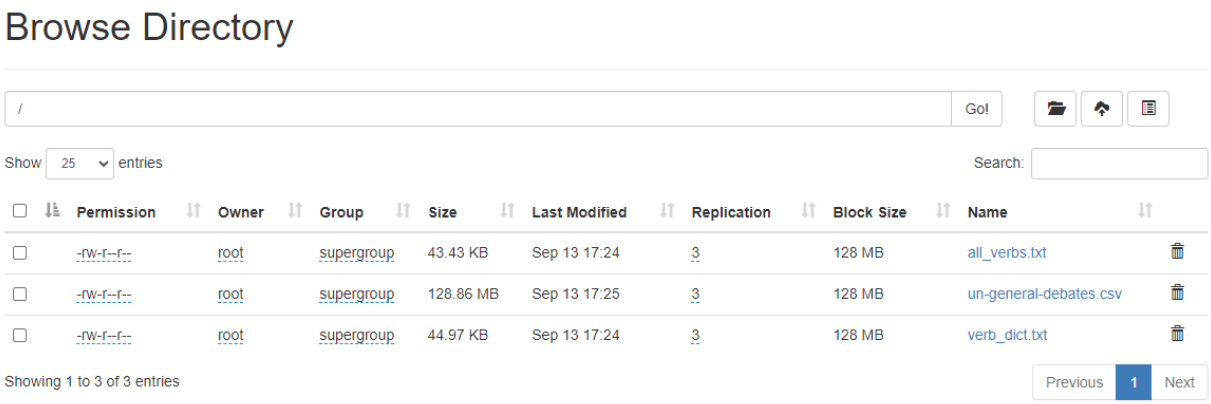
```
docker ps

docker exec <container_id> hdfs dfs -put /home/nbs/all_verbs.txt /all_verbs.txt

docker exec <container_id> hdfs dfs -put /home/nbs/verb_dict.txt /verb_dict.txt

docker exec <container_id> hdfs dfs -put /home/nbs/un-general-debates.csv /un-general-debates.csv
```

Run the above commands to put the three source files into HDFS. Substitute `<container_id>` with your `namenode` container ID. After that, you should see the three files from HDFS web interface at `http://external_IP/explorer.html`



Step 6:

The `un-general-debates.csv` is a dataset that includes the text of each country's statement from the general debate, separated by "country", "session", "year" and "text". This dataset includes over forty years of data from different countries, which allows for the exploration of differences between countries and over time [1,2]. It is organized in the following format:

	session	year	country	text
0	44	1989	MDV	It is indeed a pleasure for me and the member...
1	44	1989	FIN	\nMay I begin by congratulating you. Sir, on ...
2	44	1989	NER	\nMr. President, it is a particular pleasure ...
3	44	1989	URY	\nDuring the debate at the fortieth session o...
4	44	1989	ZWE	I should like at the outset to express my del...

In this assignment, we only consider the "text" column.

The `verb_dict.txt` file contains different tenses of each verb, separated by commas. The first word is the simple present tense of the verb.

```
abash,abash,abashed,abashed,abashes,abashing
abate,abate,abated,abated,abates,abating
abide,abide,abode,abode,abides,abiding
absorb,absorb,absorbed,absorbed,absorbs,absorbing
accept,accept,accepted,accepted,accepts,accepting
accompany,accompany,accompanied,accompanied,accompanies,accompanying
ache,ache,ached,ached,aches,aching
achieve,achieve,achieved,achieved,achieves,achieving
acquire,acquire,acquired,acquired,acquires,acquiring
```

The `all_verbs.txt` file contains all the verbs.

```
abash
abashed
abashed
abashes
abashing
abate
abated
abated
abates
abating
abide
abode
abode
abides
abiding
absorb
absorbed
absorbed
absorbs
```

Step 7:

Create a Jupyter Notebook to complete the programming objectives.

We provide some intermediate output samples below. Please note that these outputs are NOT answers and may vary from your outputs due to different implementations and different Spark behaviors.

- Intermediate output sample 1, take only verbs:

```
verbs.take(10)
```

```
['is',
 'extend',
 'is',
 'am',
 'gains',
 'achieved',
 'expressing',
 'dedicated',
 'wishes',
 'note']
```

- Intermediate output sample 2, top 10 verb counts (without converting verb tenses):

```
# Collect and display the results
for word, count in sorted_word_count_rdd.collect():
    print(f"{word}: {count}")
```

```
is: 226436
be: 128314
are: 112198
have: 105463
has: 102665
been: 50225
states: 39149
was: 32069
support: 25203
developing: 22488
```

- Intermediate output sample 3, most similar debate:

```
print(df["text"][int(I[0])])
```

```
Today, we face the reality of
an international system that is becoming more complex
by the day. Globalization has created new
opportunities; it has also produced uncertainty and
insecurity. Sudden crises affect the fundamental
sources of our sustenance and progress: food, energy
and financial resources. Terrorism and nuclear
proliferation threaten our security. Climate change has
an impact on the future of our planet. Growing
instability characterizes the energy and financial
markets, widening the gap between rich and poor.
 Those global challenges need a timely political
response. A national response would be inadequate and
illusory, for the right response can only be global and
cooperative. A dramatic new vision of global
governance for the twenty-first century requires rules
that all nations can embrace – a new vision based on
three principles: inclusiveness, effectiveness and
shared responsibility.
```

You are free to use your own implementation. However, your result should reasonably reflect the top 10 verbs that are most frequently used in UN debates, and the most similar debate contents to the sentence “Global climate change is both a serious threat to our planet and survival.”

Reference:

[1] UN General Debates, https://www.kaggle.com/datasets/unitednations/un-general-debates.
[2] Alexander Baturo, Niheer Dasandi, and Slava Mikhaylov, "Understanding State Preferences With Text As Data: Introducing the UN General Debate Corpus". Research & Politics, 2017.

Appendix:

Transformations:

Transformation	Meaning
map(*func*)	Return a new distributed dataset formed by passing each element of the source through a function *func*.
filter(*func*)	Return a new dataset formed by selecting those elements of the source on which *func*returns true.
flatMap(*func*)	Similar to map, but each input item can be mapped to 0 or more output items (so *func*should return a Seq rather than a single item).
union(*otherDataset*)	Return a new dataset that contains the union of the elements in the source dataset and the argument.
intersection(*otherDataset*)	Return a new RDD that contains the intersection of elements in the source dataset and the argument.
distinct([*numPartitions*]))	Return a new dataset that contains the distinct elements of the source dataset.
groupByKey([numPartitions])	When called on a dataset of (K, V) pairs, returns a dataset of (K, Iterable<V>) pairs. Note: If you are grouping in order to perform an aggregation (such as a sum or average) over each key, using reduceByKey or aggregateByKey will yield much better performance. Note: By default, the level of parallelism in the output depends on the number of partitions of the parent RDD. You can pass an optional numPartitions argument to set a different number of tasks.
reduceByKey(func, [numPartitions])	When called on a dataset of (K, V) pairs, returns a dataset of (K, V) pairs where the values for each key are aggregated using the given reduce function func, which must be of type (V,V) => V. Like in groupByKey, the number of reduce tasks is configurable through an optional second argument.
sortByKey([ascending], [numPartitions])	When called on a dataset of (K, V) pairs where K implements Ordered, returns a dataset of (K, V) pairs sorted by keys in ascending or descending order, as specified in the boolean ascending argument.
join(otherDataset, [numPartitions])	When called on datasets of type (K, V) and (K, W), returns a dataset of (K, (V, W)) pairs with all pairs of elements for each key. Outer joins are supported through leftOuterJoin, rightOuterJoin, and fullOuterJoin.

Actions:

Action	Meaning
reduce(*func*)	Aggregate the elements of the dataset using a function *func* (which takes two arguments and returns one). The function should be commutative and associative so that it can be computed correctly in parallel.
collect()	Return all the elements of the dataset as an array at the driver program. This is usually useful after a filter or other operation that returns a sufficiently small subset of the data.
count()	Return the number of elements in the dataset.
first()	Return the first element of the dataset (similar to take(1)).
take(*n*)	Return an array with the first *n* elements of the dataset.
countByKey()	Only available on RDDs of type (K, V). Returns a hashmap of (K, Int) pairs with the count of each key.
foreach(func)	Run a function func on each element of the dataset. This is usually done for side effects such as updating an Accumulator or interacting with external storage systems. Note: modifying variables other than Accumulators outside of the foreach() may result in undefined behavior. See Understanding closures for more details.