Coding Exercise for PMU-B Coding AI - Code Clone Detector

In this coding exercise, you will try using code2vec to generate code vectors from code snippets and find similar code snippets by using cosine similarity. Please follow the steps below.

1. Move the main working folder.

```
cd /content
```

```
/content
```

2. Clone the code2vec project to this working folder.

```
!git clone https://github.com/tech-srl/code2vec.git
```

```
Cloning into 'code2vec'...
remote: Enumerating objects: 718, done.
remote: Counting objects: 100% (14/14), done.
remote: Compressing objects: 100% (13/13), done.
remote: Total 718 (delta 2), reused 8 (delta 0), pack-reused 704 (from 1)
Receiving objects: 100% (718/718), 5.14 MiB | 11.04 MiB/s, done.
Resolving deltas: 100% (418/418), done.
```

3. Clone the test data to this working folder.

```
!git clone https://github.com/cragkhit/PMUB-CodingAI-CloneData.git
```

```
Cloning into 'PMUB-CodingAI-CloneData'...
remote: Enumerating objects: 43, done.
remote: Counting objects: 100% (43/43), done.
remote: Compressing objects: 100% (31/31), done.
remote: Total 43 (delta 12), reused 39 (delta 11), pack-reused 0 (from 0)
Receiving objects: 100% (43/43), 5.91 KiB | 5.91 MiB/s, done.
Resolving deltas: 100% (12/12), done.
```

3. Download the pre-trained code2vec model.

```
!wget https://s3.amazonaws.com/code2vec/model/java14m_model.tar.gz
!tar -xvzf java14m_model.tar.gz
```

```
--2024-12-15 03:39:45--  https://s3.amazonaws.com/code2vec/model/java14m_model.tar.gz
Resolving s3.amazonaws.com (s3.amazonaws.com)... 16.15.192.66, 16.182.65.104, 52.217.231.232, ...
Connecting to s3.amazonaws.com (s3.amazonaws.com)|16.15.192.66|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 1440921240 (1.3G) [application/x-tar]
Saving to: ‘java14m_model.tar.gz’

java14m_model.tar.g 100%[===================>]   1.34G  54.2MB/s    in 30s

2024-12-15 03:40:16 (46.0 MB/s) - ‘java14m_model.tar.gz’ saved [1440921240/1440921240]

models/java14_model/saved_model_iter8.release.data-00000-of-00001
models/java14_model/saved_model_iter8.release.index
models/java14_model/saved_model_iter8.release.meta
models/java14_model/dictionaries.bin
```

4. Go inside the code2vec project.

```
cd code2vec/
```

```
/content/code2vec
```

5. Try running code2vec to generate a code vector from the given Input.java file.

```
!python3 code2vec.py --load /content/models/java14_model/saved_model_iter8.release --predict --export_code_vectors
```

```
2024-12-15 03:40:41,594 INFO     RELEASE                                         False
2024-12-15 03:40:41,594 INFO     SAVE_EVERY_EPOCHS                               1
2024-12-15 03:40:41,594 INFO     SAVE_T2V                                        None
2024-12-15 03:40:41,595 INFO     SAVE_W2V                                        None
2024-12-15 03:40:41,595 INFO     SEPARATE_OOV_AND_PAD                            False
2024-12-15 03:40:41,595 INFO     SHUFFLE_BUFFER_SIZE                             10000
2024-12-15 03:40:41,595 INFO     TARGET_EMBEDDINGS_SIZE                          384
2024-12-15 03:40:41,595 INFO     TEST_BATCH_SIZE                                 1024
2024-12-15 03:40:41,595 INFO     TEST_DATA_PATH
2024-12-15 03:40:41,595 INFO     TOKEN_EMBEDDINGS_SIZE                           128
2024-12-15 03:40:41,595 INFO     TOP_K_WORDS_CONSIDERED_DURING_PREDICTION        10
2024-12-15 03:40:41,595 INFO     TRAIN_BATCH_SIZE                                1024
2024-12-15 03:40:41,595 INFO     TRAIN_DATA_PATH_PREFIX                          None
2024-12-15 03:40:41,595 INFO     USE_TENSORBOARD                                 False
2024-12-15 03:40:41,595 INFO     VERBOSE_MODE                                    1
2024-12-15 03:40:41,595 INFO     _Config__logger                                 <Logger code2vec (INFO)>
2024-12-15 03:40:41,595 INFO     context_vector_size                             384
2024-12-15 03:40:41,595 INFO     entire_model_load_path                          /content/models/java14_model/saved_model_iter8.release
2024-12-15 03:40:41,595 INFO     entire_model_save_path                          None
2024-12-15 03:40:41,595 INFO     is_loading                                      True
2024-12-15 03:40:41,595 INFO     is_saving                                       False
2024-12-15 03:40:41,595 INFO     is_testing                                      False
2024-12-15 03:40:41,595 INFO     is_training                                     False
2024-12-15 03:40:41,595 INFO     model_load_dir                                  /content/models/java14_model
2024-12-15 03:40:41,596 INFO     model_weights_load_path                         /content/models/java14_model/saved_model_iter8.release
2024-12-15 03:40:41,596 INFO     model_weights_save_path                         None
2024-12-15 03:40:41,596 INFO     test_steps                                      0
2024-12-15 03:40:41,596 INFO     train_data_path                                 None
2024-12-15 03:40:41,596 INFO     train_steps_per_epoch                           0
2024-12-15 03:40:41,596 INFO     word_freq_dict_path                             None
2024-12-15 03:40:41,596 INFO     ------------------------------------------------------------------------
2024-12-15 03:40:41,596 INFO     Loading model vocabularies from: `/content/models/java14_model/dictionaries.bin` ...
2024-12-15 03:40:43,631 INFO     Done loading model vocabularies.
2024-12-15 03:40:44,309 INFO     Done creating code2vec model
2024-12-15 03:40:55.489428: I tensorflow/compiler/mlir/mlir_graph_optimization_pass.cc:388] MLIR V1 optimization pass is not enab
2024-12-15 03:41:16.499135: W external/local_tsl/tsl/framework/cpu_allocator_impl.cc:83] Allocation of 328766976 exceeds 10% of f
2024-12-15 03:41:18.052281: W external/local_tsl/tsl/framework/cpu_allocator_impl.cc:83] Allocation of 666036736 exceeds 10% of f
2024-12-15 03:41:18.442087: W external/local_tsl/tsl/framework/cpu_allocator_impl.cc:83] Allocation of 465772032 exceeds 10% of f
2024-12-15 03:41:20,665 INFO     Initalized variables
2024-12-15 03:41:20,675 INFO     Loading model weights from: /content/models/java14_model/saved_model_iter8.release
2024-12-15 03:41:22.777786: W external/local_tsl/tsl/framework/cpu_allocator_impl.cc:83] Allocation of 328766976 exceeds 10% of f
2024-12-15 03:41:22.777847: W external/local_tsl/tsl/framework/cpu_allocator_impl.cc:83] Allocation of 666036736 exceeds 10% of f
2024-12-15 03:41:24,379 INFO     Done loading model weights
Starting interactive prediction...
Modify the file: "Input.java" and press any key when ready, or "q" / "quit" / "exit" to exit
Traceback (most recent call last):
  File "/content/code2vec/code2vec.py", line 37, in <module>
    predictor.predict()
  File "/content/code2vec/interactive_predict.py", line 34, in predict
    user_input = input()
KeyboardInterrupt
Exception ignored in atexit callback: <function load_source.<locals>.<lambda> at 0x7968bd714700>
Traceback (most recent call last):
  File "/usr/local/lib/python3.10/dist-packages/tensorflow/python/autograph/pyct/loader.py", line 57, in <lambda>
```

6. We'll start creating code vectors of multiple code snippets.

```
cv1 = '0.2585543 0.018499821 0.6259956 -0.91153747 0.28625304 -0.20867313 -0.6456262 -0.5417256 0.40780866 0.6872428 0.48593658 0.804606
cv2 = '0.2585543 0.018499821 0.6259956 -0.91153747 0.28625304 -0.20867313 -0.6456262 -0.5417256 0.40780866 0.6872428 0.48593658 0.804606
cv3 = '0.64085674 0.26927942 -0.35969922 -0.91668624 0.40962797 -0.28096685 -0.37260604 -0.19097356 -0.42043567 -0.7046634 0.8543717 0.8
cv4 = '0.5734073 -0.61398137 0.42359218 -0.50067383 -0.31744143 0.28601596 0.47893968 0.33539 -0.5896994 -0.31442124 -0.045689236 0.1951
```

7. Next, we will calculate the cosine similarity of two code vectors.

```
# Code adapted from https://www.geeksforgeeks.org/how-to-calculate-cosine-similarity-in-python/
# import required libraries
import numpy as np
from numpy.linalg import norm

# convert the codevectors to numpy arrays
A = np.array(cv1, dtype=float)
B = np.array(cv1, dtype=float)

# compute cosine similarity -- compare the same vector of the same code snippet
cosine = np.dot(A,B)/(norm(A)*norm(B))
print("Cosine Similarity (A-A):", cosine)

# convert the codevectors to numpy arrays
A = np.array(cv1, dtype=float)
B = np.array(cv2, dtype=float)

# compute cosine similarity -- compare between two code snippets
```

```
cosine = np.dot(A,B)/(norm(A)*norm(B))
print("Cosine Similarity (A-B):", cosine)

# convert the codevectors to numpy arrays
A = np.array(cv1, dtype=float)
B = np.array(cv3, dtype=float)

# compute cosine similarity -- compare between two code snippets
cosine = np.dot(A,B)/(norm(A)*norm(B))
print("Cosine Similarity (A-B):", cosine)
```

```
Cosine Similarity (A-A): 1.0
Cosine Similarity (A-B): 1.0
Cosine Similarity (A-B): 0.592724290020246
```

8. Write code to create a list of the 4 code vectors and compare all of them.

```
# Fill in this part
import numpy as np
from numpy.linalg import norm

A = np.array(cv1, dtype=float)
B = np.array(cv1, dtype=float)
cosine = np.dot(A, B) / (norm(A) * norm(B))
print("Cosine Similarity (1-1):", cosine)

B = np.array(cv2, dtype=float)
cosine = np.dot(A, B) / (norm(A) * norm(B))
print("Cosine Similarity (1-2):", cosine)

B = np.array(cv3, dtype=float)
cosine = np.dot(A, B) / (norm(A) * norm(B))
print("Cosine Similarity (1-3):", cosine)

B = np.array(cv4, dtype=float)
cosine = np.dot(A, B) / (norm(A) * norm(B))
print("Cosine Similarity (1-4):", cosine)

A = np.array(cv2, dtype=float)
B = np.array(cv1, dtype=float)
cosine = np.dot(A, B) / (norm(A) * norm(B))
print("Cosine Similarity (2-1):", cosine)

B = np.array(cv2, dtype=float)
cosine = np.dot(A, B) / (norm(A) * norm(B))
print("Cosine Similarity (2-2):", cosine)

B = np.array(cv3, dtype=float)
cosine = np.dot(A, B) / (norm(A) * norm(B))
print("Cosine Similarity (2-3):", cosine)

B = np.array(cv4, dtype=float)
cosine = np.dot(A, B) / (norm(A) * norm(B))
print("Cosine Similarity (2-4):", cosine)

A = np.array(cv3, dtype=float)
B = np.array(cv1, dtype=float)
cosine = np.dot(A, B) / (norm(A) * norm(B))
print("Cosine Similarity (3-1):", cosine)

B = np.array(cv2, dtype=float)
cosine = np.dot(A, B) / (norm(A) * norm(B))
print("Cosine Similarity (3-2):", cosine)

B = np.array(cv3, dtype=float)
cosine = np.dot(A, B) / (norm(A) * norm(B))
print("Cosine Similarity (3-3):", cosine)

B = np.array(cv4, dtype=float)
cosine = np.dot(A, B) / (norm(A) * norm(B))
print("Cosine Similarity (3-4):", cosine)

A = np.array(cv4, dtype=float)
B = np.array(cv1, dtype=float)
cosine = np.dot(A, B) / (norm(A) * norm(B))
print("Cosine Similarity (4-1):", cosine)

B = np.array(cv2, dtype=float)
cosine = np.dot(A, B) / (norm(A) * norm(B))
print("Cosine Similarity (4-2):", cosine)
```

```
B = np.array(cv3, dtype=float)
cosine = np.dot(A, B) / (norm(A) * norm(B))
print("Cosine Similarity (4-3):", cosine)

B = np.array(cv4, dtype=float)
cosine = np.dot(A, B) / (norm(A) * norm(B))
print("Cosine Similarity (4-4):", cosine)
```

```
Cosine Similarity (1-1): 1.0
Cosine Similarity (1-2): 1.0
Cosine Similarity (1-3): 0.592724290020246
Cosine Similarity (1-4): 0.4986763756688598
Cosine Similarity (2-1): 1.0
Cosine Similarity (2-2): 1.0
Cosine Similarity (2-3): 0.592724290020246
Cosine Similarity (2-4): 0.4986763756688598
Cosine Similarity (3-1): 0.592724290020246
Cosine Similarity (3-2): 0.592724290020246
Cosine Similarity (3-3): 1.0
Cosine Similarity (3-4): 0.5198205202061463
Cosine Similarity (4-1): 0.4986763756688598
Cosine Similarity (4-2): 0.4986763756688598
Cosine Similarity (4-3): 0.5198205202061463
Cosine Similarity (4-4): 0.9999999999999998
```