Installation

The main github repository for the project is https://github.com/deept368/cq-cpp
All the additional files are saved in the google drive folder
https://drive.google.com/drive/u/0/folders/1_E67QWO3nG-UAWtwqNbWi81k5otsrcLx
Please get appropriate access permission for the group.

A C++ version of Contextual Quantizer.

Implementation of Paper Compact Token Representations with Contextual Quantization for Efficient Document Re-ranking.

Models
All the pretrained models are kept in /model folder. We are primarily using .pt files and loading them as models in the code. For BERT, we are using model.proto as pretrained weights. Along with this vocabulary and codebook files are also present in the /model folder.

Dependencies

Intel MKL

Download the appropriate version of intel oneapi that contains the mkl library from the link given below:
https://www.intel.com/content/www/us/en/developer/tools/oneapi/onemkl-download.html?operatingsystem=linux&distributions=online

Follow the instructions on this page to set up the library. Once done, note the installation path for the mkl library. Usually, the path would be '/opt/intel/oneapi/mkl/latest'. Later, we'll need this path to indicate the mkl library path in the CMakeFile of our project.

Install librprotobuf-dev

Bert uses protobuf to convert pytorch pretrained model in protobuf (.proto) file and load it for C++

Install this library using the command on the linux terminal.

Unset
```
sudo apt-get install libprotobuf-dev
```

Install latest version of cmake

We will install cmake directly from source instead of using apt-get as some linux distributions might not have the latest version of cmake. Remove existing version of cmake on your machine.

Steps for installing latest version of cmake

1. Remove existing version of cmake

Unset
```
sudo apt remove --purge --auto-remove cmake
```

2. Visit https://cmake.org/download/ and download the latest bash script. (In my case cmake-3.27.7.tar.gz is sufficient). On a remote machine you can use wget to download the script.

Unset
```
wget
https://github.com/Kitware/CMake/releases/download/v3.27.7/cmake-
3.27.7.tar.gz
```

3. Make the script executable

Unset
```
chmod +x /opt/cmake-3.*your_version*.sh
```

4. Change to desired installation directory (to /opt/ for example)

Unset
```
mv cmake-3.*your_version*.sh /opt/
```

5. Run the installation script (You will need to press y twice during installation)

```
Unset
sudo bash /opt/cmake-3.*your_version*.sh
```

6. The script installs the binary to /opt/cmake-3.*your_version* so in order to get the cmake command, make a symbolic link

```
Unset
sudo ln -s /opt/cmake-3.*your_version*/bin/* /usr/local/bin
```

7. Test your results

```
Unset
cmake --version
```

Install Protobuf compiler

This project requires a protobuf-compiler having versions between 3.11.0 and 3.11.4. Hence, we will install directly from source to get the correct required version.

Protobuf download - make from source - version 3.11.0

1. Download appropriate tar.gz version file from the link: https://github.com/protocolbuffers/protobuf/releases/tag/v3.11.0. On a remote machine, you can download it using wget.

```
Unset
wget
https://github.com/protocolbuffers/protobuf/releases/download/v3.
11.0/protobuf-all-3.11.0.tar.gz
```

2. Extract the contents and change in the directory
1. ./configure

2. make
3. make check
4. sudo make install
5. sudo ldconfig # refresh shared library cache.

Install boost library

```
Unset
sudo apt-get install libboost-all-dev
```

Install or Build 'libutf8proc.a':

If you don't have 'libutf8proc.a' installed on your system, you need to either install it using your package manager or build it from source. To build it from source, you can typically follow these steps:

1. Download the source code

```
Unset
git clone https://github.com/JuliaStrings/utf8proc.git
cd utf8proc
```

2. Build the library

```
Unset
make
```

3. Install it

```
Unset
sudo make install
```

Note the installation path for the file libutf8proc.a as we'll need it during our project setup to modify the CMakeLists file. The path will be where you install using the above commands.

We are using PyTorch C++ to carry out various neural netowrk operations. Install the stable version of libtorch for C++ from here:. This file is also available in the shared drive folder with the name: libtorch should be present in /cq-cpp.

Setup

We are using PyTorch C++ to carry out various neural network operations. Install the stable version of libtorch for C++ from here:
https://download.pytorch.org/libtorch/cpu/libtorch-cxx11-abi-shared-with-deps-1.13.1%2Bcpu.zip
Libtorch should be present in /cq-cpp. Place the downloaded zip file in the base project directory.

First, clone the github repository for the project into an appropriate directory on your machine.
https://github.com/deept368/cq-cpp
From the drive
(https://drive.google.com/drive/u/0/folders/1_E67QWO3nG-UAWtwqNbWi81k5otsrcLx),
download the following files/folders and place them inside the project folder in the relative locations shown below.

1. model.proto - inside cq-cpp/model/
2. output_result.zip - Unzip this file and place it inside cq-cpp/data/

In the CMakeLists.txt file (present in the base folder of the project), change the following 2 variable paths to point to the intel mkl library and the utf8proc library

1. set(MKL_ROOT /opt/intel/oneapi/mkl/latest) - change this line to point to the appropriate location for your installed intel mkl library.
2. set(utf8proc_LIBRARY /home/ajit/utf8proc/libutf8proc.a) -> change this line to point to the appropriate location for your installed utf8proc library.

Running the project

bert_sample.cpp is the entry point for our code.
From the cq-cpp folder (base project folder), run the following commands.

```
Unset
mkdir build
cd build
cmake .. -DCMAKE_MODULE_PATH=/path/to/cq-cpp
make -j4
```

```
./bert-sample
```

Results:
A trec file is generated which contains the results of re-ranking. Sample trec file can be found in /output folder.

Testing

The dataset used is MS MARCO passage dataset containing 8.8 million passages. In src/config.h, we have two configs, QUERY_FILE and RESULTS_FILE where we put the path of the file that has all the queries of MS MARCO corresponding to their ids and path of the file which contains results obtained from topK retrieval respectively. Sample data is present in /data folder of the remote box

All the config variables are explained below and can be changed to achieve desired modifications.
Model and dataset parameters:
HIDDEN_SIZE
QUERY_MAXLEN
DOC_MAXLEN
DIMENSION_SIZE
VOCAB_SIZE
PAD_TOKEN_ID
CODEBOOK_COUNT
CODES_COUNT
TORCH_DTYPE
SIMILARITY_METRIC
CODEBOOK_DIM

Input data parameters:
BASE_STORE_FILE -> Location for the compressed embedding store (provide a base file value)
STORE_SIZE -> Number of files for the embeddings store
IN_MEMORY_CODES -> Flag that if set to true will first load the embeddings into memory and then use them. If set to false, the embeddings will be read from disk as required
QUERY_FILE -> File containing the input queries
RESULTS_FILE -> File to save the query results
PRE_BATCH_SIZE -> Batch size for processing the queries