project #2

The api will return in plain text the segmented chinese text. The Lac model is used as model, it's based on paddle for both training and deployment.

build:

> prerequisite
>
> - I recommand installation by source code which will better match the CUDA/CuDNN version depended by paddle inference.
>   - Otherwise, the prebuilt lib is on paddle inference lib. Please wisely choose the correct package with your CUDA/CuDNN installed, if no precise version matched, then you have to purge the existed CUDA/CuDNN and reinstall the existed combination.
> - All the CUDA packages can be downloaded here CUDA Archive. The install instruction is on the official guide
> - Same for CuDNN Archive and CuDNN guide
> - Download the lastest Lac segmentation model, unzip it locally.
> - Install Sougou workflow

Replace path below with path where the dependencies installed above, by default, becareful of the two ***/opt*** path which are customized.

```
"/opt/local_paddle"
"/usr/lib/x86_64-linux-gnu/"
"/usr/local/cuda/lib64"
"/usr/include/x86_64-linux-gnu"
"/usr/lib/x86_64-linux-gnu"
"/opt/sogou/lib/cmake/workflow/"
```

Change the path, run the cmake command below.

```
mkdir build && cd build
cmake \
    -DPADDLE_LIB=<PADDLE_LIB> \
    -DCUDNN_LIB=<CUDNN_LIB> \
    -DCUDA_LIB=<CUDA_LIB> \
    -DTENSORRT_ROOT=<TENSORRT_ROOT> \
    -DTENSORRT_ROOT_LIB=<TENSORRT_ROOT_LIB> \
    -DWORKFLOW_LIB=<WORKFLOW_LIB> \
    ..
make
```

Usage:

```
# launch the server
# model_path is the lac model previously downloaded in prerequisite
# custom_dict_path is the custom dict location
./build/server_seg <port> <thread_num> <model_path> <custom_dict_path>

# request server
curl -X POST --data-binary @<test_data> <ip>:<port>
```

Example

```
./build/server_seg 9090 4 $(pwd)/models_general/lac_model $(pwd)/custom_dict.txt

# request server
curl -X POST --data-binary @$(pwd)/test/traited_pullword.log 127.0.0.1:9090
```

How to add customized segmentation word:

See add customized word for instruction of custom dict file, and modify the ***custom_dict.txt*** whatever you want, test it as before.

Benchmark

```
# download test case
./benchmark.py download

# run all test
./benchmark.py run

# For a shorter test, download one log file, then stop.
# Use `head -n 1000 ...log > trated_...log` to make a
# short test case, then `./benchmark.py run`
```

Benchmark result

spec

Spec	Config
CPU	AMD 5800H 8 cores 16 thread
RAM	16GB
OS	Ubuntu 20.04 on WSL 2

result

Process num	Thread num	Time	Traitment speed
2	5	13.25s	142.34KWords/s
2	8	12.19s	154.73KWords/s
2	10	12.37s	152.50KWords/s
8	5	60.69s	124.33KWords/s
8	8	59.40s	127.02KWords/s
8	10	60.59s	124.53KWords/s