**Embedded ASR engine test specification**

**introduction**

This document is the test specification for embedded ASR engine. The engine is implemented with C++. In this test, some test cases are used to find out what the engine can do, and detect hidden restrict and unexpected case.

**documentation testing**

There is some ambiguity in the document.

1. The document does not mention environment variables should be set.

occasion:

After compiling the engine, go to directory enginelib/, and run

./decoder-scp config wav.scp

detail:

enginelib/decoder-scp, enginelib/config, and enginelib/wav.scp are provided by the engine as demo.

error:

error while loading shared libraries

1. The document does not mention the device on which to use the engine should be consistent.

occasion:

Install OpenBLAS and compile the engine in tiger01. Then switch to wolf02, go to directory enginelib/, set environment variables and run

 ./decoder-scp config wav.scp

detail:

enginelib/decoder-scp, enginelib/config, and enginelib/wav.scp are provided by

the engine as demo.

error message:

core dumped

1. The document does not mention the input variable limitation of a provided function.

occasion:

After compiling the engine, go to directory enginelib/, and run

 ./decoder-scp\_chunk\_1 config wav.scp

or run

 ./decoder-scp\_chunk\_3 config wav.scp

detail:

a. enginelib/config and enginelib/wav.scp are provided by the engine as demo.

b. decoder-scp\_chunk\_1.cc and decoder-scp\_chunk\_3.cc are modified from enginelib/decoder-scp.cc which is provided by the engine. They are saved at

/work6/weiy/work/gitlab/fnasr/enginelib/. Compared with decoder-scp.cc, they both change the value of chunk\_len which is used as input of provided function decoder\_put\_data().

error:

For decoder-scp\_chunk\_1.cc, it is core dumped. For decoder-scp\_chunk\_3.cc, the result is random

**functional testing**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| No | operation | expected output | actual output | result |
| 1 | let mode=1, chunk\_len=8000, input empty wav file | Alert | core dumped | Fail |
| 2 | let mode=1, chunk\_len=8000, input illegal wav file | Alert | core dumped | Fail |
| 3 | let mode=1, chunk\_len=8000, input extra long wav file | get certain result | get certain result | Pass |
| 4 | let mode=1, chunk\_len=8000, input normal wav file | get certain result | get certain result | Pass |
| 5 | let mode=1, chunk\_len=8000, input short wav file | get certain result | get certain result | Pass |
| 6 | let mode=1, chunk\_len=8000, input silence wav file | get certain result | get certain result | Pass |
| 7 | let mode=1, chunk\_len=8000, input instument wav file | get certain result | get certain result | Pass |
| 8 | let mode=1, chunk\_len=80, input normal wav file | get same result to case 4 | get same result to case 4 | Pass |
| 9 | let mode=1, chunk\_len changes each time, input normal wav file | get same result to case 4 | get same result to case 4 | Pass |
| 10 | let mode=0, input extra long wav file | get same result to case 3 | get different result with case 3 | Fail |
| 11 | let mode=0, input normal wav file | get same result to case 4 | get same result to case 4 | Pass |
| 12 | let mode=0, input short wav file | get same result to case 5 | get same result to case 5 | Pass |
| 13 | let mode=0, input silence wav file | get same result to case 6 | get same result to case 6 | Pass |
| 14 | let mode=0, input instrument wav file | get same result to case 7 | get same result to case 7 | Pass |
| 15 | let mode=1, chunk\_len=8000, change AM and LM, input normal wav file | get certain result | get certain result | Pass |

**performance testing**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| No | operation | expected output | actual output | result |
| 16 | let mode=1, chunk\_len=8000, input normal wav file circularly | get certain result, no memory leak | get random results, no memory leak in 48h | Fail |
| 17 | let mode=0, input normal wav file circularly | get certain result, no memory leak | get random results, no memory leak in 48h | Fail |
| 18 | let mode=1, chunk\_len=8000, input extra long wav file circularly | get certain result, no memory leak | get random results, no memory leak in 48h | Fail |
| 19 | let mode=0, input extra long wav file circularly | get certain result, no memory leak | get random results, no memory leak in 48h | Fail |

**note**

1. mode and chunk\_len are variables in provided code enginelib/decoder-scp.cc. mode is used to switch reading mode of input audio, and it equals to 0 in default. chunk\_len is valid when mode=1, which decides chunk length each time, and it equals to 8000 in default.
2. Empty wav file is a wav file whose size is 0 byte. It is generated by renaming an empty txt file.
3. Illegal wav file is generated by renaming a nonempty txt file.
4. Extra long wav file is a one hour long file. It is composed of some songs, speeches and instrumental music.
5. Normal wav file are 100 sentences provided by the engine.
6. Short wav file are some samples cut from normal wav file.
7. Silence wav file are some wav files which have no sound.
8. Instrument wav file are some instrumental music wav files.
9. The default AM and LM are set in enginelib/decoder.cc. In Test Case 15, AM and LM are changed to the alternative ones provided in enginelib/decoder.cc.
10. More detail about each test case can be found in test case script /work6/weiy/work/gitlab/fnasr/enginelib/testcase.sh.