FNASR API 2.0 Design Specification

|  |  |  |  |
| --- | --- | --- | --- |
| **Version** | **Author** | **Reviewer** | **Comments** |
| 1.0 | WD | TZY | Initial proposal for new API |
| 1.5 | WD |  | Add implementation details |

# Introduction

## Current API 1.0

FNASR (Blaster) is a standalone engine that can provide ASR service with a set of neat API functions. The present APIs involve basically 5 steps:

1. decoder\_init(): Initial ASR engine, which essentially allocates an asrDecoder object. More specifically, it reads in the basic FST.
2. decoder\_create(): Create a SingularDecoder object, which involves an NN decodable, and an isolated FST instance using the basic FST in the asrDecoder object.
3. decoder\_put\_data(): Put data chunks, one by one, with a chunk\_no associated for each chunk.
4. decoder\_get\_result(): Retrieve results. This can be done at any time.
5. decoder\_finish(): Release the decoder object, free NN and FST instances.
6. decoder\_release(): Release the asrDecoder object, free the FST.

To improve the online performance, we MUST keep the SigularDecoder objects residing in the memory and handle decoding tasks one by one, instead creating a new SigularDecoder object for each task. To meet this request, the present API will reset this object whenever a chunk\_no is 1 or -1.

## 1.2 How the NN decodable deal with chunk\_no

The essential goal for using chunk\_no is to add left or right context. This is well defined in the NN decodable. In the decodable, there is a variable chunk\_num\_ , it will be reset to zero when the decodable obj is created or the decodable function reset() is called. Then the logic of the NN decodable is clear:

* chunk\_num=0||chunk\_no=0: the first block in stream model, add left context.
* chunk\_num=0&&chunk\_no=-1: the first block in the batch mode, add both left and right context.
* chunk\_num>0 && chunk\_no>0: the successive blocks in stream mode.
* Chunk\_num>0 && chunk\_no=-1: the last block in the stream mode, add right context.

## 1.3 How the engine deals with chunk\_no

SingularDecoder check the chunk\_no and data length as follows:

* If len==0: simply return wave error[bug!].
* If chunk\_no ==-1 && present feat length <=0: return feat error.

## 1.4 How the API deals with chunk\_no

* API defines the chunk\_no different from that in the SigularDecoder, it starts from 1 rather than 0. The logic is as follows:
* If abs(chunk\_no)==1, then reset the decoder first[bug!].
* Perform decoding as usual, but change the chunk\_no to the convention used by the decoder, i.e., starting from 0, rather than 1.

# Issues to address

* The present put\_data() API tries to use a chunk\_no=-1 to trigger both the end of utterance (so add right context), and reset the nnet. This will cause the function of the API over-loaded. Ideally, we hope the users tell the engine to finish the decoding, and then the user will get the result, before any reset.
* Using chunk\_no=-1 to indicate the end of utterance looks not user friendly, we define a macro to do so.
* The data length=0 should be supported when chunk\_no=-1.

# Proposal

## 3.1 Overview

We will design a new API for FNASR (Blaster), defined by FNASR API 2.0. The new API will have the following properties:

1. Add a new function decoder\_reset() in the API, so that users can reset the decoder explicitly, rather than relying on the implicit way by setting chunk\_no=-1 or chunk\_no=1.
2. Define END\_OF\_UTT to indicate the end of utterance.
3. When processing the last data block (indicated by block\_no=END\_OF\_UTT), support an empty block.

## Design details

### 3.2.1. END\_OF\_UTT

* END\_OF\_UTT defined as a macro in enginelib/decoder.h.

#define END\_OF\_UTT -1

* Note defined yet in JNI.

## 3.2.2. Last empty data block

SingularDecoder::Decode():

if (len == 0){

if (chunk\_no < 0){

feats\_.Resize(0,0);

}else{

return ERR\_WAVE;

}

}else{ //data available

…

}

Create a pseudo feature block for the last empty block, and put it to the decoder. There might be some remainder features remained in the decodable, which will be augmented with the right context (chunk\_no=-1 will be transferred in to the decodable).

### 3.2.3 decoder\_reset()

#### Enginelib/engine.cc

int decoder\_reset(const struct decoder\_obj \*obj) {

try {

obj->singular\_decoder->Reset();

return 0;

} catch (const std::exception &e) {

return -1;

}

}

#### jni/engine.cc

JNIEXPORT jint JNICALL Java\_com\_freeneb\_fnasr\_decoder\_Engine\_decoder\_1reset

(JNIEnv \*env, jobject, jlong obj){

SingularDecoder \*ptr = (SingularDecoder\*)obj;

try {

ptr->Reset();

return 0;

} catch (const std::exception &e) {

return -1;

}

}

##### Whenever start a new decoding task, decoder\_reset() should be invoked.

### 3.2.4 First chunk reset

* The first chunk (chunk\_no=1) will cause the decoder to reset(). This is just for security reason, to avoid old API users to forget to reset().
* Using reset() explicitly is the recommended way.

### 3.2.5 Reset in construction

The singularDecoder and StreamNnet3Decodable does not do explicit reset() when the obj is initialized. This has caused some bugs. The reset() function has been explicitly added into the initialization step.

### 3.2.6 Modified files

# modified: online\_api.txt

# modified: ../src/engine/singular-decoder.cc

# modified: ../src/engine/stream-nnet-decodable.cc

# modified: ../src/enginelib/decoder-scp.cc

# modified: ../src/enginelib/decoder.cc

# modified: ../src/enginelib/decoder.h

# modified: ../src/jni/com/freeneb/fnasr/decoder/Engine.java

# modified: ../src/jni/com\_freeneb\_fnasr\_decoder\_Engine.h

# modified: ../src/jni/engine.cc

# Impact and risk

## 4.1 Back compatibility

* For stream mode, users can use the new API without any change in their application, due to the first chunk reset design.
* For the batch mode, users have to call the reset() function before or after each decoding task, since the users did not set any chunk\_no=1.

## 4.2 Impact on performance

No.

## 4.3 Impact on accuracy

No.

## 4.4 Impact on other modules or products

* All the new version of the downstream demos should be changed according to the new API.
* Online-stream applications that used API 1.0 can use API 2.0 without any change in their code.
* Move doc/online\_api.txt to doc/api.txt

## 4.5 Impact on test procedure

1. All the test cases should be changed to use the new API decoder\_reset() to reset the decoding process.
2. All the test cases should be changed to use the new macro END\_OF\_UTT to notify the end of utterance, rather than -1.

## 4.6 Impact on customer usage

1. The FNASR customer doc should be amended to use the END\_OF\_UTT macro.
2. The FNASR customer doc should be updated to use the decoder\_reset() function to reset the recognition thread.
3. The example programs delivered to users should be changed accordingly.

## 4.7 Other risk

No