Software Design

Embedded Intent Recognizer

Revision 1.0

Table of Contents

1. Introduction....1
1.1 Purpose....1
1.2 System Overview....1
2. Design Considerations....1
2.1 Assumptions....1
2.2 Constraints....1
2.3 System Environment....1
2.4 Risks and Volatile Areas....1
3. Architecture....2
3.1 Overview....2
3.2 System operation....2
4. Intent Classification Data....3
4.1 File format....3
Table 2: intent.csv columns description....3
4.2 Data extensibility....4
5. Software Design....5
5.1 Input Sentence....5
5.1.1 Preprocessing....5
5.1.2 Extract keywords....6
5.2 CsvRow....6
5.2.1 Attributes....6
5.2.2 Methods....6
5.3 CsvFile Parser....7
5.3.1 Attributes....7
5.3.2 Methods....7
6. Directory structure....8
7. Test....9
7.1 Run “demo” binary....9
7.2 Run “unit_tests” binary....10
8. Discussion on Deep learning method....10

Revision History

Version	Name	Reason For Changes	Date
1.0	*Thu-Thuy Do*	*Initial Revision*	*2022/03/08*

1. Introduction

1.1 Purpose

This design will detail the implementation of the Embedded Intent Recognizer.

1.2 System Overview

This project provides the basic functionality of an Embedded Intent Recognizer command line tool. In this system, user inputs a question or a request and the output is the intent category (intent class) of the user input. The intent classes and the user input keywords in this system can be flexibly extensible.

2. Design Considerations

2.1 Assumptions

The intent data is extensible with the predefined format. The detailed description of the intent data will be explained in section 3.

2.2 Constraints

The system can handle difference semantic variation of the inputs with some constraints as follows:

- The name entities (private person name, locations, etc.) must start with uppercase characters. For example: New York, Celine Dion.
- Categories such as book names, song names must be put in double-quote marks. For example: “Heal the World”.
- The system cannot recognize the misspelling words.
- The system ignores the special characters such as “#$%#”

2.3 System Environment

The system environment is as follows:

- OS: Ubuntu 18.04
- Cmake 3.16.3
- C++17
- GoogleTest

2.4 Risks and Volatile Areas

None have been identified.

3. Architecture

In this part, the top level design view of the Embedded Intent Recognizer is provided

3.1 Overview

The system can be decomposed into the following components: The input sentence parser, the intent data model and the intent data storage.

3.2 System operation

Figure 1 presents the basic system operation as follows:

- The IntentData is stored in a csv format. It will be explained in detailed in Section 4.
- When the system starts, the intent data is loaded to Intent Data Model. The intent categories is encoded with the one-hot-encoding technique, which is popular in Natural Language Processing.
- User input a sentence. The input sentence should follows the constraints presented in Section 2.
- The InputSentenceParser does the preprocessing for the input sentence before searching for its intent category. The pre-processing steps include: removing special characters, extracting the quoted phrases, extracting the name entities and detecting the remaining keywords.
- Keywords are searched, if found, the intent data model returns its encoded intent category and finally match the encoded intent to the intent string.
- The Intent string has the information which user needs.

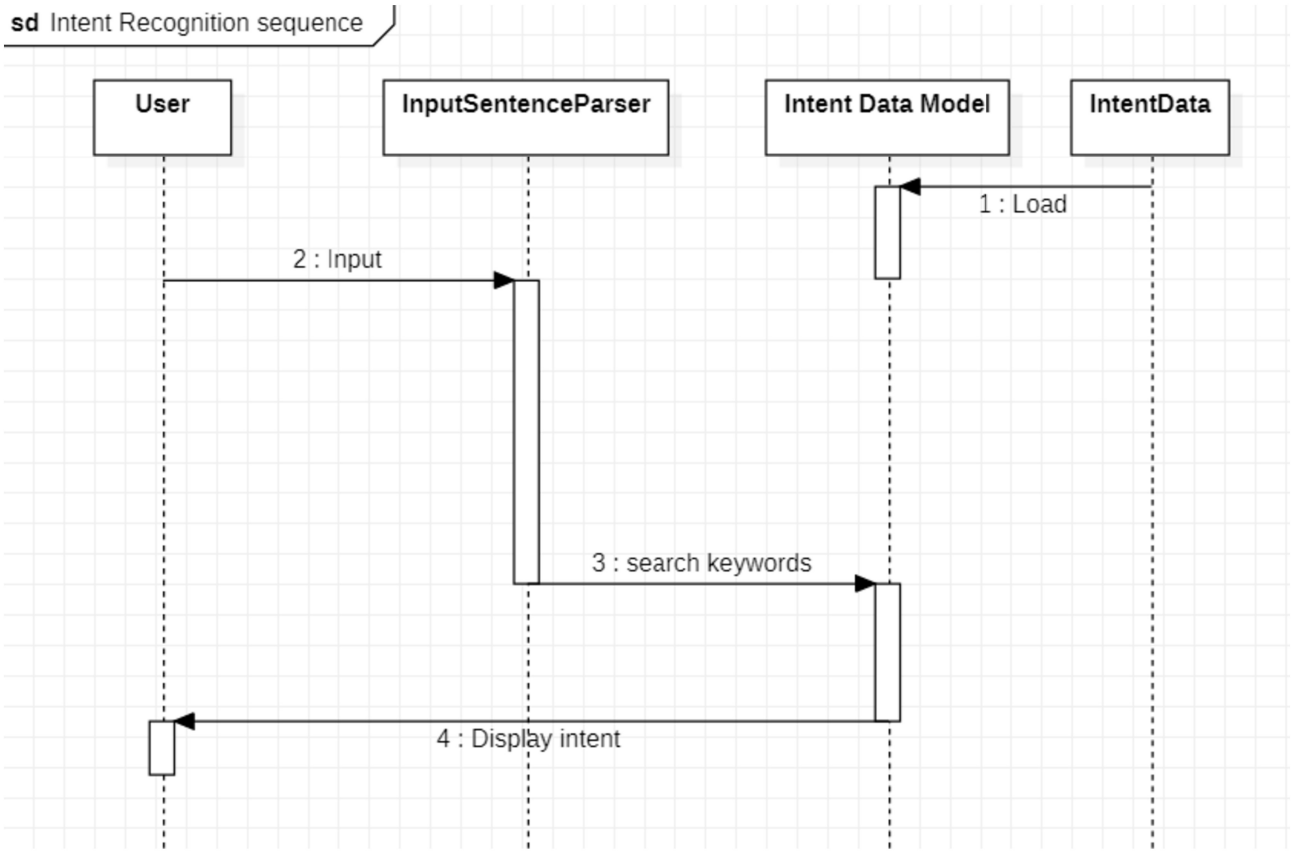


***Figure 1**: Sequence of Intent Recognizer system*

4. Intent Classification Data

4.1 File format

The intent classification is stored in a csv file. The format is showed in Table 1

word	topic	In city	people	time	book	weather
weather	0	0	0	0	0	1
fact	1	0	0	0	0	0
movie	1	0	0	0	0	0
news	1	0	0	0	0	0
kpop	1	0	0	0	0	0
war	1	0	0	0	0	0
London	0	1	0	0	0	0
Paris	0	1	0	0	0	0
Seoul	0	1	0	0	0	0
Berlin	0	1	0	0	0	0
New York	0	1	0	0	0	0
Michael Jackson	0	0	1	0	0	0
Celine Dion	0	0	1	0	0	0
tomorrow	0	0	0	1	0	0
today	0	0	0	1	0	0
yesterday	0	0	0	1	0	0
Moscow	0	1	0	0	0	0
Atomic Habits	0	0	0	0	1	0
sunny	0	0	0	0	0	1
windy	0	0	0	0	0	1

Table 1: *intent.csv file*

Columns in this file are explained in Table 2

Column Name	Data type	Description
word	String	Key word in the input sentence, which will be mapped to a specific intent class
topic	Integer (0 or 1)	The intent about some topic. Keyword in this intent type will be displayed in the result
In city	Integer (0 or 1)	The intent about some cities. The city name must be started with uppercase character. If the input sentence has the keyword in this intent class, the result string will show “In city” in its content
people	Integer (0 or 1)	The intent about some people. The people’s name must be started with uppercase character. If the input sentence has the keyword in this intent class, the result string will show “people” in its content
time	Integer (0 or 1)	The intent about time. If the input sentence has the keyword in this intent class, the result string will show “time” in its content
book	Integer (0 or 1)	The intent about books’ name. The book’s name must be put in the double quoted marks. If the input sentence has the keyword in this intent class, the result string will show “book” in its content
weather	Integer (0 or 1)	The intent about weather. If the input sentence has the keyword in this intent class, the result string will show “weather” in its content

Table 2: *intent.csv columns description*

4.2 Data extensibility

The intent classes can be extended by adding new columns to this table. The intent is defined by the column name. The keyword can be extended by adding new rows to this table. A keyword is classified to only 1 intent class.
The keywords in the input sentence can be stored by its sequence in the sentence. The keywords in the intent classification data table can be stored similar to the concept of “Bag of Words” in Natural Language Processing. The system can handle difference semantic variations of the input.
The intent classes is encoded with the “one-hot encoding” technique which is also used in Natural Language Processing, the number of intent classes can be extendable, the intent string in the output result get the information from the column header. There is no hard coding in the source code.

5. Software Design

This section provides low-level design descriptions that directly support construction of modules. The UML class diagram is presented in Figure 2:

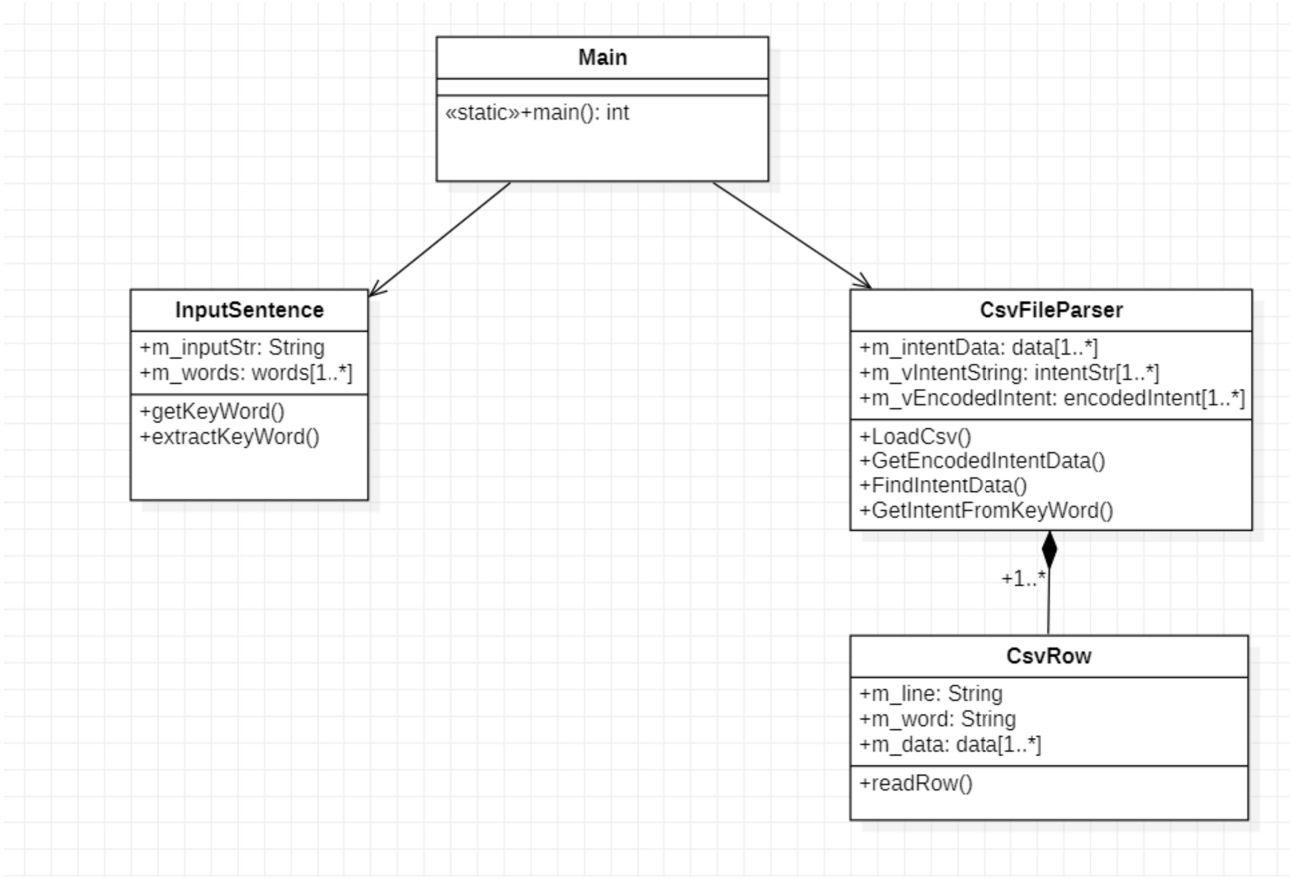


***Figure 2**: UML class diagram*

5.1 Input Sentence

5.1.1 Preprocessing

When user presses enter, the input sentence is read. The preprocessing steps include:
- Extract pattern in double-quoted marks
- Remove special characters

The API to do the preprocessing steps are described as follows:

`void extractPattern(std::string &s, std::vector<std::string> &regWords)`	
Input	Input string s
Output	Vector of string extracted from double-quoted marks

Description	Method to extract strings following a predefined regex expression (in this case: string in double quoted marks)

`void removeSpecialChars(std::string& s)`	
Input	Input string s
Output	Output string s: special characters are removed in place
Description	Remove special characters in a string

5.1.2 Extract keywords

After the input sentence is preprocessed, the keywords are extracted to be ready for the next steps to recognize the intent of the sentence.

`void extractKeywords()`	
Input	Void
Output	Void
Description	This method reads the string to extract keywords which is stored as a private member of InputSentence class and the output is stored as a vector of string and also is a private member of InputSentence class.

5.2 CsvRow

This class contains the attributes and methods for a row in the csv file.

5.2.1 Attributes

Name	*Type*	*Description*
m_line	String	Contains 1 line read from csv file
m_data	Vector<Int>	Contains the encoded intent vector for the keyword defined in the csv file
m_word	String	Contains the keyword

5.2.2 Methods

`void readRow(std::istream& str, std::string &word, std::vector<int> &data)`	
Input	Input istream str
Output	Return the string type keyword `word` and a vector of integer `data` for the encoded intent
Description	This method reads the string from input istream and extract the keyword

	and the corresponding encoded intent vector of that keyword

5.3 CsvFile Parser

This class provides methods for parsing csv file to get the predefined keywords and its encoded intent vector and the methods to find a keyword's encoded intent vector. The mapping data is stored in an unordered_map data structure to enable quick searching. Methods to find the intent string from keywords are also implemented in this class.
To make the system extendable, the intent string is extracted from the header of the rows in csv file. This class provides the methods and attributes to parse the header line and stores the intent string. The attributes and methods description are present in the next sections below.

5.3.1 Attributes

Name	*Type*	*Description*
m_path	Filesystem::path	Object of type path, which represents the path of csv file on filesystem.
m_vIntentStr	Vector<String>	The vector for the intent string
m_vEncodedIntent	Vector<Vector<int>>	The vector of encoded intent vector
m_intentData	Unordered_map<String, Vector<int>>	Map of keyword and its encoded intent vector

5.3.2 Methods

`void LoadCsvFile()`	
Input	Void
Output	Void: the parsing data is stored in the private members of CsvFileParser class
Description	This method loads the intent.csv file from the data folder and parses it. It prepares the data for getting the intent from keywords in the input sentence.

`void GetEncodedIntentDataImpl(std::string& header, std::vector<std::string> &vIntentStr, std::vector<std::vector<int>> &vEncodedIntent)`	
Input	header string
Output	Vector of intent string and vector of encoded vectors defined in csv file

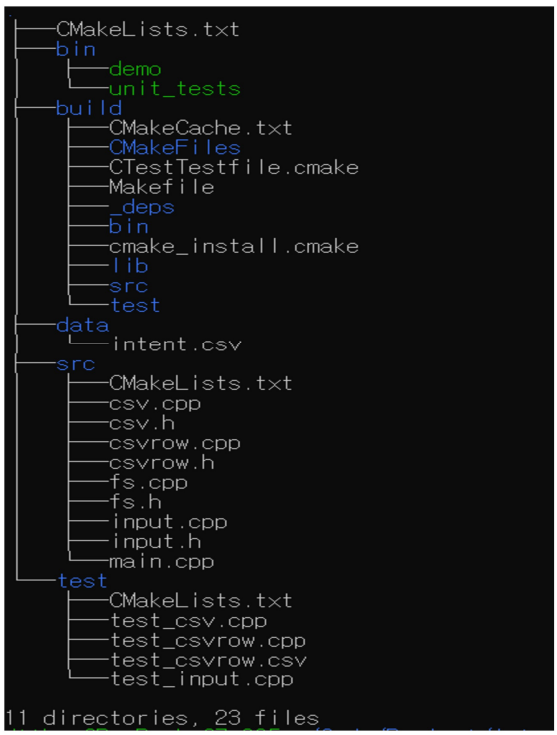
Description	This method parses the header of csv file to get the intent strings and the corresponding encoded intent vectors of these intent strings

`size_t FindIntentData(std::vector<int>& vData, std::string& str) const`	
Input	Input string to find its encoded intent category.
Output	Encoded data vector. The method returns the size of encoded vector data. If it is not found, the return size is 0.
Description	This method finds the encoded data vector given an input string, which represents a keyword

`std::string GetIntentFromKeyWords(std::string keyword)`	
Input	Input string to find its intent string.
Output	Intent string
Description	This method finds the intent string given an input string, which represents a keyword

6. Directory structure

This section provides the description for software directory structure:



Folder	Description
bin	Binary files include: demo: binary of project unit_tests: binary of unit test
src	Source code of projects
test	Unit test source code (use GoogleTest framework)
data	Data folder contains “intent.csv” file

Table 3: *Directory structure*

7. Test

This section provides the screen captures for several tests and unit_tests

7.1 Run “demo” binary

```
dtthuy@ProBook-G7-635:~/Code/Project/IntentRecognizer0308/bin$ ./demo
Enter input:Tell me an interesting fact?????????????
Intent: Get fact
dtthuy@ProBook-G7-635:~/Code/Project/IntentRecognizer0308/bin$ ./demo
Enter input:Who is Celine Dion?
Intent: Get people
dtthuy@ProBook-G7-635:~/Code/Project/IntentRecognizer0308/bin$ ./demo
Enter input:It's windy in Seoul !!!
Intent: Get weather In city
dtthuy@ProBook-G7-635:~/Code/Project/IntentRecognizer0308/bin$ ./demo
Enter input:What is the weather here?
Intent: Get weather
dtthuy@ProBook-G7-635:~/Code/Project/IntentRecognizer0308/bin$ ./demo
Enter input:What is the weather in New York?
Intent: Get weather In city
dtthuy@ProBook-G7-635:~/Code/Project/IntentRecognizer0308/bin$ ./demo
Enter input:Tell me about kpop
Intent: Get kpop
dtthuy@ProBook-G7-635:~/Code/Project/IntentRecognizer0308/bin$ ./demo
Enter input:who is the author of "Atomic Habits"?
Intent: Get book
dtthuy@ProBook-G7-635:~/Code/Project/IntentRecognizer0308/bin$ ./demo
Enter input:Tell me the news ..........................................................###
Intent: Get news
dtthuy@ProBook-G7-635:~/Code/Project/IntentRecognizer0308/bin$
```

7.2 Run “unit_tests” binary

```
dtthuy@ProBook-G7-635:~/Code/Project/IntentRecognizer0308/bin$ ./unit_tests
Running main() from /home/dtthuy/Code/Project/IntentRecognizer0308/build/_deps/gtest-src/googletest/src/gtest_main.cc
[==========] Running 8 tests from 3 test suites.
[----------] Global test environment set-up.
[----------] 1 test from csvrow
[ RUN      ] csvrow.readRowTest
[       OK ] csvrow.readRowTest (1 ms)
[----------] 1 test from csvrow (2 ms total)

[----------] 3 tests from csv
[ RUN      ] csv.CsvFileErrorTest
[       OK ] csv.CsvFileErrorTest (0 ms)
[ RUN      ] csv.GetEncodedIntentDataTest
[       OK ] csv.GetEncodedIntentDataTest (0 ms)
[ RUN      ] csv.GetIntentFromKeywordTest
[       OK ] csv.GetIntentFromKeywordTest (1 ms)
[----------] 3 tests from csv (6 ms total)

[----------] 4 tests from input
[ RUN      ] input.InputSentenceTest
[       OK ] input.InputSentenceTest (1 ms)
[ RUN      ] input.GetKeyWordsTest
[       OK ] input.GetKeyWordsTest (1 ms)
[ RUN      ] input.GetKeyWordsInDoubleQuoteTest
[       OK ] input.GetKeyWordsInDoubleQuoteTest (1 ms)
[ RUN      ] input.GetKeyWordsInputStringWithSpecialCharsTest
[       OK ] input.GetKeyWordsInputStringWithSpecialCharsTest (1 ms)
[----------] 4 tests from input (13 ms total)

[----------] Global test environment tear-down
[==========] 8 tests from 3 test suites ran. (69 ms total)
[  PASSED  ] 8 tests.
```

8. Discussion on Deep learning method

For intent recognition problem, we can use Deep learning method to train a set of sentences and use the trained model to predict the intent given by user. This section describes the basic steps to prepare the project and build the deep learning model for the intent recognition problem.

8.1 Prepare the dataset

The dataset is in the format of csv file (intent_data.csv) with 9947 sentences which are categorized into 5 classes of intention:
- BookRestaurant
- GetWeather
- PlayMusic
- RateBook
- GetFact

The intention classes are also encoded with one-hot encoding techniques.
The dataset is preprocessed with two steps:
- Fill the null data with valid values
- Remove stop words. The stop words collection is defined with English language.
- The input sentences are padded so that they have same length before input to model
- The input sentences are tokenized with a filter to remove special characters.

8.2 Build the model

Long short-term memory (LSTM) is a deep recurrent neural network architecture used for classification of sequence data. One issue of LSTM model is that it can easily be overfit therefore have low performance in predicting the real world data. To avoid this issue, we can use drop-out regularization method where input and recurrent connections to LSTM units are probabilistically excluded from activation and weight updates while training a network.

The model is briefly summarized as in Figure 3:

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
embedding (Embedding)        (None, 4000, 32)          307392
_________________________________________________________________
lstm (LSTM)                  (None, 10)                1720
_________________________________________________________________
dropout (Dropout)            (None, 10)                0
_________________________________________________________________
dense (Dense)                (None, 800)               8800
_________________________________________________________________
dropout_1 (Dropout)          (None, 800)               0
_________________________________________________________________
dense_1 (Dense)              (None, 200)               160200
_________________________________________________________________
dropout_2 (Dropout)          (None, 200)               0
_________________________________________________________________
dense_2 (Dense)              (None, 5)                 1005
=================================================================
Total params: 479,117
Trainable params: 479,117
Non-trainable params: 0
```

Figure 3: Model summary

The model is trained with batch_size = 64 and 5 epochs.

```
Epoch 1/5
112/112 [==============================] - 470s 4s/step - loss: 0.6022 - accuracy: 0.7561 - val_loss: 0.0163 - val_accuracy: 0.9987
Epoch 2/5
112/112 [==============================] - 453s 4s/step - loss: 0.1359 - accuracy: 0.9557 - val_loss: 0.0077 - val_accuracy: 0.9975
Epoch 3/5
112/112 [==============================] - 453s 4s/step - loss: 0.1010 - accuracy: 0.9636 - val_loss: 0.0069 - val_accuracy: 0.9987
Epoch 4/5
112/112 [==============================] - 456s 4s/step - loss: 0.0876 - accuracy: 0.9673 - val_loss: 0.0077 - val_accuracy: 0.9975
Epoch 5/5
112/112 [==============================] - 455s 4s/step - loss: 0.0752 - accuracy: 0.9719 - val_loss: 0.0084 - val_accuracy: 0.9975
```

After trained, the model is saved to h5 file format.

The performance is evaluated as follows:

```
63/63 [==============================] - 17s 264ms/step - loss: 0.0340 - accuracy: 0.9894
Test set
  Loss: 0.034
  Accuracy: 0.989
```

This project is also published on my Kaggle account:

https://www.kaggle.com/azaleado/intentclassifier

8.3 Test and deploy model

In this project, a test function named "`intention_predict_input`" is provided. In this function, the h5 model is loaded and predict the input sentence.
This model is developed using Keras library which is a high level library on top of Tensorflow. In the real world deployment for the embedded platform, we can convert the model using Tensorflow lite library and use it to predict the input sentence from user.
The test result is presented as follows: