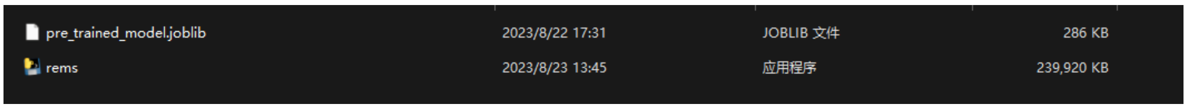
How to use REMS tool?

Welcome to the README guide for the **REMS** tool. This guide will walk you through the process of using the tool for both individual refactoring predictions and bulk testing based on the methods outlined in our research paper.

Part 1: Using the Out-of-the-Box Tool

Our package includes a pre-built command-line executable that you can use out of the box to perform refactoring predictions. This tool is designed for ease of use and allows you to test individual class methods for refactoring. Here's a step-by-step example:

1.**Download the Tool:** Download our rems executable program, please ensure that the program(`rems.exe`) and pre-trained model (`pre_trained_model.joblib`) are in the same directory.



```
pre_trained_model.joblib    2023/8/22 17:31    JOBLIB 文件    286 KB
rems                        2023/8/23 13:45    应用程序       239,920 KB
```

2.**Execute Refactoring Prediction:** Run the executable program with the appropriate arguments:

```
// source method
public String find() throws Exception {
    logger.info( "Starting find()" ); //f:log
    Session sess = HibernateUtil.getSessionFactory().openSession();
//f:hibernate
    Transaction t = sess.beginTransaction(); //f:hibernate

    Criteria criteria = sess.createCriteria( Customer.class ); //f:hibernate

    criteria.add( Example.create( this.customer
).excludeZeroes().ignoreCase().enableLike( MatchMode.ANYWHERE ) ); //f:hibernate
    if ( this.customer.getId() != null ) { //f:hibernate
        criteria.add( Restrictions.idEq( this.customer.getId() ) );
//f:hibernate
    } //f:hibernate

    @SuppressWarnings("unchecked")
    List<Customer> l = (List<Customer>) criteria.list(); //f:hibernate
    request.put( "list", l );
    t.commit(); //f:hibernate
    sess.close(); //f:hibernate

    this.task = SystemConstants.CR_MODE;
    logger.info( "Finishing find()" ); //f:log
    return INPUT;
}
```

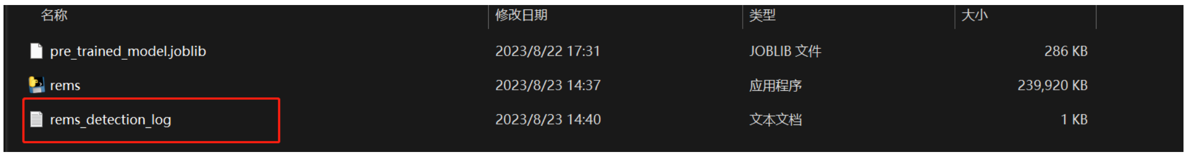
```
----------------------------------------------------------------------------

Start To Load Pretrain Model ... ...
Finish Loading Pretrain Model ! !
```

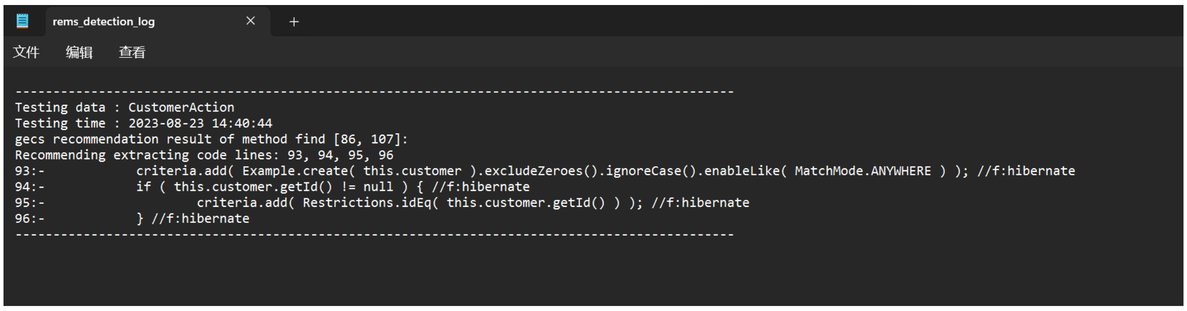
```
----------------------------------------------------------------------------
The Java source file's absolute path for refactoring prediction:
D:\dataset\REMS\TEST\10000\CustomerAction.java
source code preprocessing...
method name: setRequest, line range: (46, 48)
method name: setParameters, line range: (51, 54)
method name: getCustomer, line range: (56, 58)
method name: setCustomer, line range: (60, 62)
method name: getTask, line range: (64, 66)
method name: setTask, line range: (68, 70)
method name: execute, line range: (73, 75)
method name: input, line range: (78, 84)
method name: find, line range: (86, 107)
method name: save, line range: (109, 124)
method name: update, line range: (126, 141)
method name: delete, line range: (143, 158)
method name: edit, line range: (160, 177)
----------------------------------------------------------------------------
Please select a method:find
embedding extraction...
model detection...
applicable analysis...
Recommending extracting code lines:  [93, 94, 95, 96]
Relevant information is stored in rems_detection_log.txt
----------------------------------------------------------------------------
```

Replace `D:\dataset\REMS\TEST\10000\CustomerAction.java` and `find` with the desired class path and method names.

3.**Detection Log:** We have provided a log file to record all the information regarding your refactoring predictions using GECS. The file is located in the same directory as the GECS executable program.



名称	修改日期	类型	大小
pre_trained_model.joblib	2023/8/22 17:31	JOBLIB 文件	286 KB
rems	2023/8/23 14:37	应用程序	239,920 KB
rems_detection_log	2023/8/23 14:40	文本文档	1 KB



```
--------------------------------------------------------------------------------------------
Testing data : CustomerAction
Testing time : 2023-08-23 14:40:44
gecs recommendation result of method find [86, 107]:
Recommending extracting code lines: 93, 94, 95, 96
93:-            criteria.add( Example.create( this.customer ).excludeZeroes().ignoreCase().enableLike( MatchMode.ANYWHERE ) ); //f:hibernate
94:-            if ( this.customer.getId() != null ) { //f:hibernate
95:-                    criteria.add( Restrictions.idEq( this.customer.getId() ) ); //f:hibernate
96:-            } //f:hibernate
--------------------------------------------------------------------------------------------
```

Part 2: Reproducing Research Methodology

If you find that you need to perform refactoring predictions in bulk and require greater efficiency, you can replicate our research methodology using the files provided in our package. Our package covers every step from dataset preparation to classifier-based bulk testing. Here's how you can proceed:

1. **Refer to Documentation and Code:** Inside the package, you'll find comprehensive documentation and code examples that guide you through the entire process.

2. **Batch Dataset Preparation:** Follow the guidelines (`src/utils/get_dataset/README`) and code (`src\utils\get_dataset\`) provided in the documentation to create batch datasets for refactoring prediction.
3. **Generating Batch Embeddings:** Use the code snippets provided in the package (`src/FlowEmbedding` and `src/TreeEmbedding`) to generate embeddings for your batch data.
4. **Combining Batch Embeddings:** Learn how to merge batch embeddings using the reference code (`src\utils\Embedding_fusion.py` and `src\utils\compact_bilinear_pooling\`).
5. **Performing Bulk Testing:** Utilize the provided classifier (`src\utils\classifier`) to perform bulk testing. The package offers code to calculate precision, recall, and F1-measure values.
6. **Environment Setup:** For each step, the README documentation details the environment setup required to successfully replicate the methodology.

Wishing you a smooth experiment!