

 **WORKSHOP** 研讨/分享/感悟

仓颉社区中的灵感碰撞 社区先行者的干货分享

仓颉代码转译工具code_tools

主讲人：杨海龙 (博士在读)

- code_tools
- code generation
- antlr4cj
- based on AI



江南大学
JIANGNAN UNIVERSITY



人工智能与计算机学院
School of Artificial Intelligence and Computer Science

WORKSHOP

目录

- 仓颉转译工具code_tools的介绍
- 基于code_tools的社区实践项目antlr4cj
- 基于人工智能的code_tools学术探索

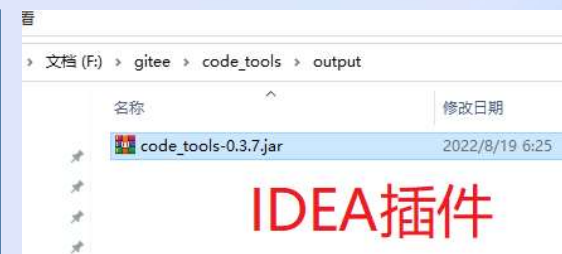
仓颉转译工具code_tools的介绍

仓颉转译工具code_tools是通过源源转译的方式为仓颉生态自动生成代码的工具。code_tools主要包括如下工具：

- j2cj: java语言代码转译成仓颉代码。
- c2cj: c语言代码转译成仓颉代码。
- g2cj: go语言代码转译成仓颉代码。（设计中）
- 注释: 自动为仓颉编程语言添加注释
- 用例: 自动为仓颉编程语言生成LLT和UT用例。

code_tools的意义将其他成熟工业语言（java/c/c++/go...）生态的优秀项目引入到仓颉生态，辅助仓颉生态构筑领域框架和领域库等基础设施。

code_tools相关项目的gitee地址：
https://gitee.com/HW-PLLab/code_tools
<https://gitee.com/HW-PLLab/c2cj>
<https://gitee.com/HW-PLLab/j2cj>



WORKSHOP

仓颉转译工具code_tools的j2cj

j2cj
Java to cangjie

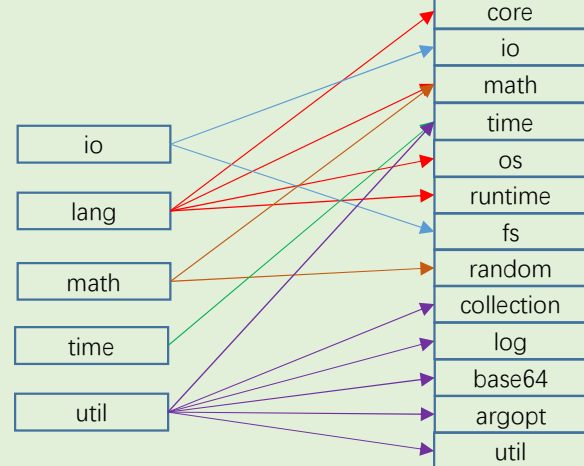
代码骨架的转译

1

函数body里逻辑代码的转译

2

Java的base库映射到仓颉标准库:



java语法映射到仓颉语法:变量声明和赋值、基本数据类型、操作符、流程控制、数组、函数、枚举、类、接口、lambda...

Java源码

仓颉源码

```
Java源码:
package org.antr.v4.runtime.misc;
import java.io.File;
import java.io.FileInputStream;
import java.io.FileOutputStream;
import java.io.IOException;
import java.io.InputStreamReader;
import java.io.OutputStreamWriter;
import java.util.Arrays;
import java.util.BitSet;
import java.util.Collection;
import java.util.HashMap;
import java.util.Iterator;
import java.util.Map;
public class Utils {
    // Seriously: why isn't this built in to java? ugh!
    public static <T> String join(Iterator<T> iter, String separator) {
        StringBuilder buf = new StringBuilder();
        while ( iter.hasNext() ) {
            buf.append( iter.next() );
            if ( iter.hasNext() ) {
                buf.append( separator );
            }
        }
        return buf.toString();
    }
    public static <T> String join(T[] array, String separator) {
        StringBuilder builder = new StringBuilder();
        for (int i = 0; i < array.length; i++) {
            builder.append(array[i]);
            if (i < array.length - 1) {
                builder.append(separator);
            }
        }
        return builder.toString();
    }
}

仓颉源码:
package default
from std import io.*
from std import collection.*
// import java.util.Collection;
// import java.util.Iterator;
open public class Utils {
    public static func join<T> (iter: Option<Iterator<T>>, separator: S
    // -- StringBuilder buf = new StringBuilder();
    // -- while (iter.hasNext()) {
    // --     buf.append(iter.next());
    // --     if (iter.hasNext()) {
    // --         buf.append(separator);
    // --     }
    // -- return buf.toString();
    }
    return ""
}
    public static func join<T> (array: Array<T>, separator: String): S
    // -- StringBuilder builder = new StringBuilder();
    // -- for (int i = 0; i < array.length; i++) {
    // --     builder.append(array[i]);
    // --     if (i < array.length - 1) {
    // --         builder.append(separator);
    // --     }
    // -- return builder.toString();
    }
    return ""
}
    public static func numNonNull (data: Array<Object>): Int32 {
    // -- int n = 0;
    // -- if (data == null) return n;
    // -- for (Object o : data) {
    // --     if (o != null) n++;
    // -- }
    // -- return n;
    }
    return 0
}
```

Java与仓颉语法比较: https://gitee.com/HW-PLLab/j2cj/blob/develop/doc/compare_Java9_CangJie.md#%E6%9D%A1%E4%BB%B6%E8%AF%AD%E5%8F%A5
Java与仓颉核心库比较: https://gitee.com/HW-PLLab/j2cj/tree/develop/doc/libs_compare



仓颉转译工具code_tools的c2cj

c2cj是生成对c语言库封装一个仓颉实现的native层。在C语言包括C++的生态语言库可以这个native层间接被仓颉调用。



C语言转仓颉 <https://gitee.com/HW-PLLab/c2cj>
<https://gitee.com/HW-PLLab/cangjie-example/tree/master/NativeInvoke>

```
/*
 * Copyright (c) Cangjie Library Team 2022-2022. All rights reserved.
 */

package demo
from std import ffi.c.*

foreign func cjNativePrintHello(): Int64
foreign func cjNativePrintSomething(str: CString): Int64
foreign func cjNativeSetVal(index: Int64, val: Int64): Int64
foreign func cjNativeGetLibName(index: Int64): CString

public func NativePrintHello(): Int64 {
    return unsafe { cjNativePrintHello() }
}

public func NativePrintSomething(str: String): Int64 {
    var cs= CString(str)
    let result = unsafe { cjNativePrintSomething(cs) }
    cs.free()
    return result
}

public func NativeSetVal(index: Int64, val: Int64): Int64 {
    return unsafe { cjNativeSetVal(index, val) }
}

public func NativeGetLibName(index: Int64): String {
    let lib = unsafe { cjNativeGetLibName(index) }
    var ret = lib.toString()
    return ret
}
```

仓颉Native层

引入C接口

封装的仓颉Native接口

仓颉转译工具code_tools的注释和用例自动生成

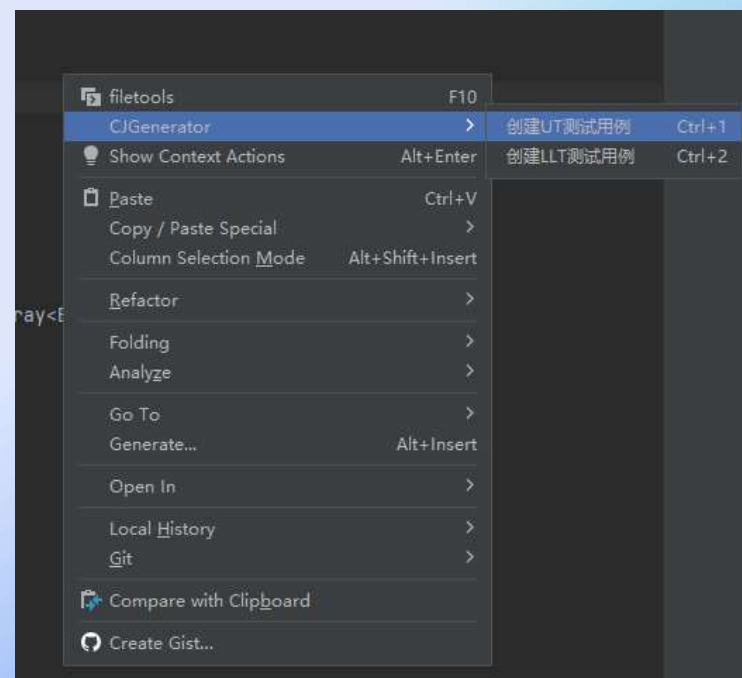
code_tools提供针对仓颉源码的注释和用例的自动生成。

注释主要是包括文件头、类注释、函数注释。

Code_tools可以一键生成仓颉源码的LLT和UT用例。

```
/*
 * The Function is compare
 *
 * @param aval of Array<Byte>
 * @param fromIndex of Int64
 * @param bval of Array<Byte>
 *
 * @return Type of Bool
 */
func compare(aval: Array<Byte>, fromIndex: Int64, bval: Array<Byte>): Bool {
    let a_size: Int64 = aval.size()
    let b_size: Int64 = bval.size()
    if(a_size - fromIndex < b_size) {
        return false
    }
    for(i in 0..a_size) {
        if(aval[i + fromIndex] != bval[i]) {
            return false
        }
    }
}
```

自动生成函数注释



基于code_tools的社区实践项目 antlr4cj



What is ANTLR?

ANTLR (ANother Tool for Language Recognition) is a powerful parser generator for reading, processing, executing, or translating structured text or binary files. It's widely used to build languages, tools, and frameworks. From a grammar, ANTLR generates a parser that can build and walk parse trees.



Terence Parr is a tech lead at Google and until 2022 was a professor of data science / computer science at Univ. of San Francisco. He is the maniac behind ANTLR and has been working on language tools since 1989.

Check out Terence impersonating a machine learning droid: [explained.ai](#)

ANTLR（全名：ANother Tool for Language Recognition）是基于LL(*)算法实现的语法解析器生成器（parser generator），用Java语言编写，使用自上而下（top-down）的递归下降LL剖析器方法。由旧金山大学的Terence Parr博士等人于1989年开始发展。

ANTLR有什么用？

- **定制特定领域语言(DSL)**。类似hibernate中的HQL，用DSL来定义要执行操作的高层语法，这种语法接近人可理解的语言，由DSL到计算机语言的翻译则通过ANTLR来做，可在ANTLR的结构语言中定义DSL命令具体要执行何种操作；
- **文本解析**。可利用ANTLR解析JSON，HTML，XML，EDIFACT，或自定义的报文格式。解析出来的信息需要做什么处理也可以在结构文件中定义；
- **数学计算**。加减乘除，线性方程，几何运算，微积分等等；
- **IDE语法检查**。
-

WORKSHOP

基于code_tools的社区实践项目 antlr4cj

ANTLR支持的语言包括C#、c++、Dart、go、java、js、python、swift。Cangjie正在适配中。

antlr4cj项目是code_tools在仓颉社区第一个也是最大的实践项目，一共生成了174个.cj文件，合计1万6千多行代码。

antlr4cj项目中的仓颉代码大部分是code_tools生成的代码，需要大量人力来翻译code_tools暂时没法生成的代码。



目前antlr研究小组主要是我和汪鹏程在主导antlr4cj项目，有兴趣的同学可以加入我们。

相关项目的gitee地址：
<https://gitee.com/HW-PLLab/antlr4cj>



基于人工智能的code_tools学术探索

code_tools的核心任务是将其他语言的程序转译到仓颉编程语言。

也就是输入为其他语言的代码生成问题

代码生成模型的设计问题

Code Generation

94 papers with code • 12 benchmarks • 22 datasets

Code Generation is an important field to predict explicit code or program structure from multimodal data sources such as incomplete code, programs in another programming language, natural language descriptions or execution examples. Code Generation tools can assist the development of automatic programming tools to improve programming productivity.

Libraries

Use these libraries to find Code Generation models and implementations

DeepLearnXMU/CG-RL	3 papers	23 ★
codedotal/gpt-code-clippy	2 papers	1,978 ★
salesforce/coder1	2 papers	177 ★
arminmoin/ML-Quadrat	2 papers	21 ★

Datasets



See all 22 code generation datasets

idea1: 通过社区代码设计仓颉语言的数据集 (dataset)，经过训练后，允许多模态 (multi-modal) 数据作为输入 (而不是仅是其他高级语言代码)，生成仓颉代码。

idea2: 其他语言的编程知识 (knowledge) 迁移到仓颉代码生成模型中，特别是领域编程知识，这样让仓颉代码生成模型也有了其他领域代码生成能力。

idea3: 如何基于对抗学习，将仓颉编译器作为判决器，加上仓颉代码度量，来提升生成仓颉代码的质量。

WORKSHOP

code_tools的规划

基于IDEA插件版本会继续向前演进

- j2cj尝试函数体的转译以提高转译率;
- c2cj现存问题的解决以达到合入主干的要求;
- g2cj的设计方案完成以及实现;

基于人工智能版本的探索

- 尝试基于transformer以及ASN（抽象语法网络）设计仓颉语言的代码生成模型。

WORKSHOP 研讨/分享/感悟

仓颉社区中的灵感碰撞 社区先行者的干货分享

Thanks!



杨海龙(Eric)
江苏 无锡



扫一扫上面的二维码图案，加我为朋友。

邮箱: yanghailong@stu.jiangnan.edu.cn



江南大学
JIANGNAN UNIVERSITY



人工智能与计算机学院
School of Artificial Intelligence and Computer Science

WORKSHOP

仓颉生态之代码觉醒

下次workshop讲吧

WORKSHOP