

East - 基于 LALR、链表和 tf-idf 的单词搜索

周泓余 *

2020 年 6 月 25 日

目录

1 版本	2
2 部署与使用	2
2.1 部署	2
2.2 入门	2
2.2.1 主要命令	2
2.2.2 检索语言	3
3 设计与框架	4
3.1 源代码组织	4
3.2 整体逻辑组织	5
4 示例	5
5 TODO、版本历史等辅助信息	7
5.1 ToDO	7
5.2 版本历史	7

* 邮箱: <peterlitszo@outlook.com>

1 版本

当前版本为:

```
version 0.4.2
```

2 部署与使用

2.1 部署

请按照:

```
$ go get -u -v github.com/PeterlitsZo/argparse
$ go build -o main ./src
```

或者: (如果在 Linux 环境下推荐使用)

```
$ make init
$ make
```

进行编译。因为本软件是基于 Goolge 开发的开源语言 GoLang 编写而成, 使用前需要安装 GoLang 的编译器。GoLang 语言是一门比较小众的语言, 所以如果没有该编译器的话, 还请进入 <https://golang.org/dl/> 下载。

输入命令 `./main version` 来检查是否部署成功。如果部署成功, 则会出现如下的信息:

```
$ ./main version
East version 0.4.x
$
```

2.2 入门

2.2.1 主要命令

编译完成后, 输入下命令可以得到主要用法:

```
$ ./main --help
usage: East <Command> [-h|--help]
description:
    sreach engine on file system
commands:
    run          the command to get the ID list (see README.pdf)
    mkindex      use this flag to make index named 'index.dict'
    interactive  make Self under the Interactive mode
    version      Show East's Version
arguments:
    -h, --help  Print help information
$
```

通过 `mkindex` 命令可以为 `dirpath` 文件夹（默认值为 “input”，但可以根据 `--dirpath` 来进行修改）生成索引，然后通过 `--useindex` 来让其它命令使用索引文件。如果没有 `--useindex` 命令，则会动态遍历文件夹，分词，以哈希表加链表作为数据结构（当然，使用索引文件的话，虽然不需要遍历分词，但是还是需要抽象成数据结构）。

最后使用 `run` 命令或者进入 `interactive` 模式来完成主要目标：检索。语法见后文。

2.2.2 检索语言

这一次选择使用 `goyacc` 工具来生成相应的 LALR 解析器源码，我首先实现了一个分词器，然后编写 BNF 式的语法规则，构建了解析器 `parse.y`。

语法规则如下：

```
<ast>          ::= SREACH <sreach_word>
                  | LIST
<sreach_word>  ::= <expr> OR <sreach_word>
                  | <expr>
<expr>         ::= <atom> AND <expr>
                  | <atom>
<atom>         ::= NOT STR
                  | STR
                  | NOT '(' <sreach_word> ')'
                  | '(' <sreach_word> ')'
```

实例：

```
1. sreach 'in' AND NOT 'in' OR 'her'
2. sreach "" or not ""
3. sreach 'outside' && !'inside'
4. sreach (('that' or !'that') and 'that')
```

如实例所述，STR 字符串是由单引号或者双引号包裹而成，支持为单引号或者双引号转义（其实在命令行结构下本身就需要对引号进行转义，所以难免会发生二次转义，比如 `--command="'\\\\"` 这样比较丑陋的语法）。

不过现在甚至可以在交互模式下使用了。使用 `-interactive` 进入交互模式，然后在交互下输入命令，就算是同时使用单引号和双引号也轻轻松松、再也不用担心二次转义啦（看到就是赚到）。

而 AND，OR，NOT 不仅仅支持单词（忽略大小写）样式，还支持 C 式的逻辑运算符，也就是说 `&&`，`||` 和 `!` 这三个传统命令。

现在还支持括号来实现更加高级的操作了。

3 设计与框架

3.1 源代码组织

总体来说，所有源代码都在 `src` 文件夹下，为：

```
./src
|-- argparse
|   |-- argparse.go
|-- list
|   |-- list.go
|-- logic
|   |-- logic.go
|-- main.go
|-- parse
|   |-- parse.go
|   |-- parse.y
|-- units
|   |-- file.go
|   |-- split.go
|   |-- version.go
|   |-- wordmap.go

5 directories, 10 files
```

按照模块化结构，所有的源代码都分工合作，各司其职。分别对他们进行短暂描述有：

argparse 解析命令行命令，提供 usage 提示。

list 提供有序链表的接口。

logic 实现了底层，如根据抽象出的语法树处理业务进行搜索处理。

parse 讲前端的命令处理为抽象语法树。

units 杂项，其他。包含：

file 提供处理文件的接口。

spilt 提供文本分词的接口，用来见文件转换为数据结构，或者说是 index 文件的第一步。

version 提供版本信息。

wordmap 抽象出的数据结构。

3.2 整体逻辑组织

整体上来说逻辑分层，分别分为 I/O 层，Mid 层¹，Low 层² 和 Sys 层³。大体来说，整体的逻辑组织结构如图 1。

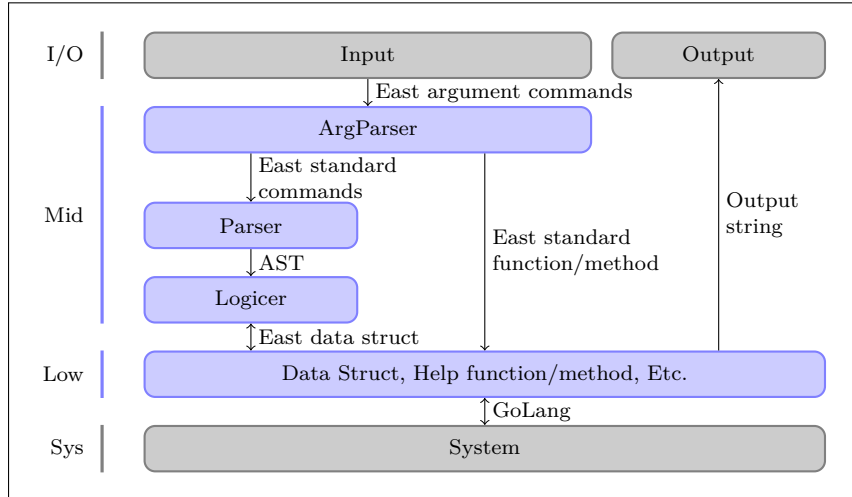


图 1: the struct of East

4 示例

[需要更新]

```

$ # ---[ build itself
$ ]-----
$
$ go build -o main ./src
$ ./main --help
usage: East [-h|--help] [-d|--dirpath "<value>"] [-c|--command
    "<value>"]
    [-m|--mkindex] [-u|--useindex] [-i|--interactive]

    sreach engine on file system

Arguments:

    -h --help          Print help information
    -d --dirpath       the input files' folder path. Default:
                        input
  
```

¹用来处理大部分逻辑，包括，处理从 I/O 中传入的数据，使用 Low 层作为工具进行逻辑处理，返回 Low 层数据结构

²定义和抽象绝大多数的数据结构。

³如内存管理等等。使用 GoLang 语言操控，并没有实际接触。

```

-c --command      the command to get the ID list (see README
.pdf). Default:
-m --mkindex      use this flag to make index named 'index.
dict'
-u --useindex      use file 'index.dict' to find result
-i --interactive   make self under the interactive mode
$
$ # ---[ a little test using `--command`
]-----
$
$ ./main --command="'in' || 'not'"
result: [ d01.txt, d10.txt, d02.txt, d03.txt, d04.txt, d06.txt,
d07.txt, d08.txt, d09.txt ]
$
$ # -[ use flag `--mkindex` and `--useindex` to hold the result
]---
$
$ ls
README.md
README.pdf
input
main
make.sh
src
$
$ ./main --mkindex
$
$ ls # it will make a new file named index.dict
README.md
README.pdf
index.dict
input
main
make.sh
src
$
$ ./main --useindex --command="not 'in'"
result: [ d5.txt ]
$
$ ---[ the interactive mode
]-----
$
$ ./main --useindex --interactive
Enter `quit` for quit
copyleft (C) Peterlits Zo <peterlitszo@outlook.com>
Github: github.com/PeterlitsZo

Command > "that" and 'peter'
result: [ ]
Command > 'that' and ('peter' or '!peter')
result: [ d06.txt, d07.txt ]
Command > quit
$

```

5 TODO、版本历史等辅助信息

5.1 ToDO

- ☐ 有什么办法可以来一键安装 GoLang的依赖吗?
- ☐ 更加详细的，函数式的结构。

注: ☐ 未完成, ☒ 正在完成中, ☒ 已经完成。

5.2 版本历史

0.4.2: 全面更新了文档。使用 L^AT_EX 而非 Markdown来编写文档。

0.4.1: 重构，并让 `interactive`成为默认选项。

0.4.0: 使用 `sub-command`。

0.3.0: 使用前置命令以支持更多的操作，目前支持命令 `sreach`。

0.2.4: 使用 `Peterlits/argparse`替代原作者的库，以获得更好的 `usage` 输出（不过如果原作者如果接受了我的 `pull request` 的话，那么其实可能又会换回来）。

before: 未记录。