

# East - 基于 LALR 和链表的单词搜索

版本

```
version 0.2.3
```

使用

请按照：

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

或者：

```
$ make init
$ make
```

进行编译。因为本软件是基于 Google 开发的开源语言 go 编写而成，使用前需要安装 go 的编译器。go 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。

编译完成后，输入下命令可以得到主要用法：

```
$ ./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
  -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
```

通过--mkindex命令可以为dirpath文件夹（默认为 input，但可以根据--dirpath来进行修改）生成索引，然后通过--useindex来让其使用索引文件。如果没有--mkindex和--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--command命令来完成主要目标：检索。语法见后文。

## command 实现与语法

这一次选择使用 go yacc 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      ::= expr OR ast
          |   expr
expr     ::= atom AND expr
          |   atom
atom     ::= NOT STR
          |   STR
          |   NOT '(' ast ')'
          |   '(' ast ')'
```

实例：

```
1. 'in' AND NOT 'in' OR 'her'
2. "" or not ""
3. 'outside' && !'inside'
4. (('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）。~~（但是我的高数作业还有很多，实在是没有时间来支持交互式了）~~

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 `&&`，`||`，`!`。

现在还支持括号来实现更加高级的操作了。~~（p.s. 如果有时间的话可能会支持括号）~~

示例

```
$ # ---[ 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
-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  
$
```

## 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。