

敏感词过滤

gaccob

2012 年 11 月 25 日

一个简单的敏感词过滤的设计:

- 最基本的思路是建立hash表来做查询.
- 敏感词的长度一般不会太长, 一般也就2个汉字(4个字节)左右, 可以再一份敏感词长度的索引, 这样子在处理时能提高效率.
- 编码问题: 词库和源因为以往的历史原因, 是gbk编码, 而非utf.
- 目前是一个英文忽略大小写的设计.
- 代码在gbase中, 关键部分的代码如下所示:

```
1 // dirty word
2 typedef struct dirty_t
3 {
4     char word[MAX_DIRTY_WORDS_LEN];
5     int prev;
6     int next;
7 } dirty_t;
8
9 // dirty context
10 typedef struct dirty_ctx_t
11 {
12     int table_size;
13     dirty_t table[MAX_DIRTY_WORDS_COUNT];
14     int hash[MAX_DIRTY_WORDS_HASH_COUNT];
15     int index[256][256];
16 } dirty_ctx_t;
17
18 // if > 0x80, means double bytes
19 // else, single byte
20 const uint8_t GB_SPEC_CHAR = (uint8_t)(0x80);
```

```
21
22 // dirty words check
23 int dirty_check(dirty_ctx_t* ctx, const char* src, int
    len)
24 {
25     static char lowercase[MAX_SOURCE_WORDS_LEN];
26     int i, k, step, key;
27     const uint8_t* from;
28
29     if (!ctx || !src || len > MAX_SOURCE_WORDS_LEN) {
30         return -3;
31     }
32     for (i = 0; i < len; i++) {
33         lowercase[i] = tolower(src[i]);
34     }
35
36     for (i = 0, step = 0; i < len; i += step) {
37         key = 0;
38         from = (const uint8_t*)&lowercase[i];
39         if (from[0] < GB_SPEC_CHAR) {
40             key = ctx->index[0][from[0]];
41             step = 1;
42         } else if (i + 1 < len) {
43             key = ctx->index[from[0]][from[1]];
44             step = 2;
45         } else {
46             printf("source_code_error\n");
47             return -2;
48         }
49         // no index, go ahead
50         if (0 == key) {
51             continue;
52         }
53         // found key
54         for (k = 1; k < MAX_DIRTY_WORDS_LEN; k++) {
55             // exceed len
56             if (i + k > len) {
57                 break;
58             }
59             // no dirty
60             if (0 == CHECK_DIRTY_FLAG(key, k)) {
61                 continue;
62             }
63             if (0 == dirty_hash_find(ctx, (const char*)
                from, k)) {
64                 return -1;
65             }
66             // no need to loop all
67             RESET_DIRTY_FLAG(key, k);
```

```
68         if (0 == key) {  
69             break;  
70         }  
71     }  
72 }  
73 return 0;  
74 }
```