Cws 文档

庄国华
上海埃帕信息科技有限公司
guohua.zhuang@ape-tech.com

2010 年 07 月 19 日

摘要

中文分词旨在对中文句子的切分以及对切分后的词语进行词性标注。cws分词切割采用一般随机过程与最短路径算法相结合的思路，在词法标注方面采用隐马尔可夫模型，利用其Viterbi算法求出最优词性标注方案，并在此基础上设计多层隐马尔可夫模型进行命名实体的识别，并引入CRF条件随机场增强对中文人名识别的效果。

目录

0．安装与配置

Firstly，you should install CRF++.

```
        % wget http://sourceforge.net/projects/crfpp/files/crfpp/0.54/CRF%2B%2B-0.54.tar.gz
        % tar -zvxf CRF++-0.54.tar.gz
        % cd CRF++-0.54/
        % ./configure
        % make
        % su
        # make install
```

Now，you can install cws project.

```
        % export PKG_CONFIG_PATH="/usr/local/lib/pkgconfig/"
        % ./configure
        % make
        % su
        # make install
```

If it comes out some error like ":malloc" errors，try those operators:

```
        % ac_cv_func_malloc_0_nonnull=yes ac_cv_func_realloc_0_nonnull=yes ./configure --with-
gnu-ld
        % make
        % su
        % make install
```

1. cws 命令行操作

```
用法：cws [选项]... [输入文件] [输出文件]
将<输入文件>中的内容作为输入源，把结果保存到<输出文件>。
(支持输入文件或者键盘输入、输出文件或屏幕打印)

长选项必须用的参数在使用短选项时也是必需的。
        -a,     --addword               添加新词及其词性、词频
        -s,     --segmentation-pos      中文分词及标注词性(默认选项)
        -q,     --queryword             查询字典中词的信息
        -t,     --type=(PEKING|PENN)    词性标注类型(默认为 PEKING，仅配合-a、-s 使用)
        -c,     --crfpp=(yes|no)        是否使用 CRF 命名实体识别(默认为 yes，仅配合-s 使用)
        -d,     --dict=字典编号              选择要查询的字典(默认为 1，仅配合-q 使用)
                                                1. coreDict;
                                                2. BigramDict;
                                                3. nr;
                                                4. ns;
                                                5. tr;
                                                6. coreDict_Penn.dct;
                                                7. BigramDict_Penn.dct;
                --help                  显示此帮助信息并离开
                --version               显示版本自信息并离开
```

示例 1. 中文分词及标注词性(默认选项)

```
$ cws [回车]
他说的确实在理
他/r  说/v  的/u  确实/ad  在理/a

$ cws source.txt [回车]
他/r  说/v  的/u  确实/ad  在理/a

$ cws source.txt result.txt [回车]
```

示例 2. 添加新词及其词性、词频

```
$ cws -a -t=PENN[回车]
埃帕 NR 1[回车]
Congratulation! Add word "埃帕"(length = 6, handle = 22, nPos = NR, nFrequency = 1)
successfule!
酷灵 NR 1[回车]
Congratulation! Add word "酷灵"(length = 6, handle = 22, nPos = NR, nFrequency = 1)
successfule!
a a a[回车][命令行操作最后一定要输入一组词频为非法数字的行以结束程序，文件则不需要]
Congratulation! Save the dictionary file "../Data/coreDict_Penn.dct" successful!
You have to do "make install" and then the dictionary will take effect.
```

```
$ cat new_words.txt [回车]
埃帕 nt 1
酷灵 nz 1
$ cws -a --type=PEKING new_words.txt [回车]
Congratulation! Add word "埃帕"(length = 6, handle = 28276, nPos = nt, nFrequency = 1)
successfule!
Congratulation! Add word "酷灵"(length = 6, handle = 28282, nPos = nz, nFrequency = 1)
successfule!
Congratulation! Save the dictionary file "../Data/coreDict_Unicode.dct" successful!
You have to do "make install" and then the dictionary will take effect.
```

示例3. 查询字典中词的信息

```
$ cws -q -d=3
李
Block = 26446
Count = 5
        word = 0
                nFrequency = 1097
                nWordLen = 0
                nHandle = 1
        word = 1
                nFrequency = 2
                nWordLen = 0
                nHandle = 2
        word = 2
                nFrequency = 2
                nWordLen = 3
                nHandle = 23
                sWord = 子
        word = 3
                nFrequency = 2
                nWordLen = 3
                nHandle = 1
                sWord = 逵
        word = 4
                nFrequency = 216
                nWordLen = 3
                nHandle = 24
```

2. 函数接口

2.1 中文分词及标注词性

对 CwsRun 方法的中文版说明：

CwsRun 的使用分为两类：

第一类以返回 string 类型的结果，其调用形式为“string CwsRun(const string&, char, int)”。

如果第二个参数 char 传入 ID_STRING_PEKING，则表示使用北京大学的中文词性标注。

如果第二个参数 char 传入 ID_STRING_PENN，则表示使用宾夕法尼亚州中文词性标注。

第二类以返回包含词及其对应词性数组类型 vector< pair< string, string> >的结果，其调用形式为“VecStrStr CwsRun(const string&, int, int)”。

如果第二个参数 int 传入 ID_ARRAY_PEKING，则表示使用北京大学的中文词性标注。

如果第二个参数 int 传入 ID_ARRAY_PENN，则表示使用宾夕法尼亚州中文词性标注。

如果不带第二个参数，默认返回第一类中北京大学中文词性标注的 string 结果。

第三个参数表示是否使用 CRF 条件随机场增进命名实体的识别(默认参数为 ID_CRFPP_NO，不使用，要使用传入 ID_CRFPP_YES)：

优点是能提高对人名识别的准确率。

缺点是速度会比原来慢一个数量级。

示例程序 1：使用北京大学的中文词性标注并返回 string 类型的结果

```
#include <cws/Cws.h>

using namespace namespace_cws;

int main()
{
        Cws* pcws = new Cws();

        // Saving the tagging results.
        string s_ret;
        // Input the string.
        string s_str;

        while(getline(cin, s_str))
        {
                s_ret = pcws->CwsRun(s_str, ID_STRING_PEKING);
                cout << s_ret << endl;
        }

        CWS_DELETE(pcws);
        return 0;
}
```

示例程序 2：使用宾夕法尼亚州中文词性标注并返回数组结果

```
#include <cws/Cws.h>

using namespace namespace_cws;

int main()
{
        Cws cws;
        VecStrStr s_ret;
        string s_str;
        IVecStrStr it_ret;
        while(getline(cin, s_str))
        {
                s_ret = cws.CwsRun(s_str, ID_ARRAY_PENN);
                for(it_ret = s_ret.begin(); it_ret != s_ret.end(); it_ret++)
                {
                        cout << it_ret->first << '/' << it_ret->second << "  ";
                }
                cout << '\n';
        }
        return 0;
}
```

2.2 添加新词及词性词频

示例程序1：向北大核心词库添加新词及其词性词频

```
#include <cws/Cws.h>

using namespace namespace_cws;

int main()
{
        int i = 0;
        Cws* pcws = new Cws();

        string s_str;
        string s_POS;
        int n_frequency = 0;
        while(cin >> s_str, cin >> s_POS, cin >> n_frequency)
        {
                if(!pcws->m_coreDict.AddWord(s_str.c_str(), (int)s_str.length(), pcws-
>m_ctx.m_pSymbolTable[pcws->m_POS_map[s_POS]], n_frequency))
                {
                        cout << __FILE__ << ": " << __LINE__ << ": Somthing error in function
AddWord() with add word \"" << s_str << "\".\n";
                        CWS_DELETE(pcws);
                        exit(-1);
                }
                else
                {
                        cout << "Congratulation! Add word \"" << s_str <<\
                         "\"(length = " << s_str.length() << ", handle = " << pcws-
>m_ctx.m_pSymbolTable[pcws->m_POS_map[s_POS]] <<\
                         ", nPos = " << s_POS << ", nFrequency = " << n_frequency << ")
successfule!" << endl;
```

```
                }
        }

        // The dictionary path.
        string s_path = "../Data/coreDict_Unicode.dct";
        if(!pcws->m_coreDict.SaveDictionaryUTF16(s_path.c_str()))
        {
                cout << __FILE__ << ": " << __LINE__ << ": Something error in function 
SaveDictionary().\n";
                CWS_DELETE(pcws);
                exit(-1);
        }
        else
        {
                cout << "Congratulation! Save the dictionary file \"" << s_path << "\" 
successful!\n";
                cout << "You have to do \"make install\" and then the dictionary will take 
effect.\n";
        }
        CWS_DELETE(pcws);

        return 0;
}
```

示例程序 2：向宾夕法尼亚州核心词库添加新词及其词性词频

```
#include <cws/Cws.h>

using namespace namespace_cws;

int main()
{
        int i = 0;
        Cws* pcws = new Cws();

        string s_str;
        string s_POS;
        int n_frequency = 0;
        while(cin >> s_str, cin >> s_POS, cin >> n_frequency)
        {
                if(!pcws->m_coreDict_Penn.AddWord(s_str.c_str(), (int)s_str.length(), pcws-
>m_ctx_Penn.m_pSymbolTable[pcws->m_PennPOS_map[s_POS]], n_frequency))
                {
                        cout << __FILE__ << ": " << __LINE__ << ": Somthing error in function 
AddWord() with add word \"" << s_str << "\".\n";
                        CWS_DELETE(pcws);
                        exit(-1);
                }
                else
                {
                        cout << "Congratulation! Add word \"" << s_str <<\
                         "\"(length = " << s_str.length() << ", handle = " << pcws-
>m_ctx_Penn.m_pSymbolTable[pcws->m_PennPOS_map[s_POS]] <<\
                         ", nPos = " << s_POS << ", nFrequency = " << n_frequency << ")
successfule!" << endl;
                }
        }

        // The dictionary path.
        string s_path = "../Data/coreDict_Penn.dct";
        if(!pcws->m_coreDict_Penn.SaveDictionaryUTF16(s_path.c_str()))
        {
                cout << __FILE__ << ": " << __LINE__ << ": Something error in function
SaveDictionary().\n";
                CWS_DELETE(pcws);
                exit(-1);
        }
        else
        {
                cout << "Congratulation! Save the dictionary file \"" << s_path << "\"
successful!\n";
                cout << "You have to do \"make install\" and then the dictionary will take
effect.\n";
        }
        CWS_DELETE(pcws);

        return 0;
}
```

2.3 字典查询

示例程序：根据用户输入的查询要求返回字典中相应词的信息

```
#include <cws/Cws.h>

using namespace namespace_cws;

/
 Show the dict's i content.
 @param dict the dictionary be display.
 @param i the index of the character in dictionary.
 @return no return.
 /
void ShowDictContent(Dictionary& dict, int i);

/
 Show all the dict's content.
 @param dict the dictionary be display.
 @return no return.
 /
void TemporaryOutput(Dictionary& dict);

/
 Show the special binary grammer dictionary.
 @param dict the dictionary be display.
 @param ctx the context.
 @return no return.
 /
void TemporaryOutputBigrammer(Dictionary& dict, Context& ctx);

int main()
{
        Cws* pcws = new Cws();
        int i = 0;
        int select = 0;
        string s_input;
        cout << "1. coreDict; 2. BigramDict; 3. nr; 4. ns; 5. tr; 6. coreDict_Penn.dct; 7.
BigramDict_Penn.dct;\n";
        cout << "Please select the dictionary you are interested in: ";
        cin >> select;
        getchar();
        while(cout << "Please input the Chinese character: ", getline(cin, s_input))
        {
                int n_str_first_len = GetFirstCharacterLengthUTF8(s_input.c_str());
                // Convert the character to the order.
                i = UTF16_EncodingtoOrder(g_CharacterObjective.UTF8toUTF16(s_input.c_str(),
n_str_first_len, true));
                switch(select)
                {
                        case 1:
                        {
                                ShowDictContent(pcws->m_coreDict, i);
                                break;
                        }
                        case 2:
                        {
                                ShowDictContent(pcws->m_bigramDict, i);
                                break;
                        }
                        case 3:
                        {
                                ShowDictContent(pcws->m_nrDict, i);
                                break;
                        }
                        case 4:
                        {
                                ShowDictContent(pcws->m_nsDict, i);
                                break;
                        }
                        case 5:
                        {
                                ShowDictContent(pcws->m_trDict, i);
                                break;
                        }
                        case 6:
                        {
                                ShowDictContent(pcws->m_coreDict_Penn, i);
                                break;
                        }
                        case 7:
                        {
                                ShowDictContent(pcws->m_bigramDict_Penn, i);
                                break;
                        }
                        case 16:
                        {
                                TemporaryOutput(pcws->m_coreDict_Penn);
                                break;
                        }
                        case 17:
                        {
                                TemporaryOutputBigrammer(pcws->m_bigramDict_Penn, pcws-
>m_ctx_Penn);
                                break;
                        }
                        default:
                        {
                                cout << __FILE__ << ": " << __LINE__ << ": You are select the
wrong dictionary." << endl;
                                CWS_DELETE(pcws);
                                exit(-1);
                        }
                }
        }
        CWS_DELETE(pcws);
        return 0;
}

void ShowDictContent(Dictionary& dict, int i)
{
        int j = 0;
        cout << "Block = " << i << endl;
        cout << "Count = " << dict.m_blockTableUTF16[i].nCount << endl;
        if(dict.m_blockTableUTF16[i].nCount > 0)
        {
                // Read word items.
                for(j = 0; j < dict.m_blockTableUTF16[i].nCount; j++)
                {
                        cout << "\tword = " << j << endl;
                        cout << "\t\tnFrequency = " <<
dict.m_blockTableUTF16[i].pWordItemHead[j].nFrequency << endl;
                        cout << "\t\tnWordLen = " <<
dict.m_blockTableUTF16[i].pWordItemHead[j].nWordLen << endl;
                        cout << "\t\tnHandle = " <<
dict.m_blockTableUTF16[i].pWordItemHead[j].nHandle << endl;
                        if(dict.m_blockTableUTF16[i].pWordItemHead[j].nWordLen > 0)
                        {
                                cout << "\t\tsWord = " <<
dict.m_blockTableUTF16[i].pWordItemHead[j].sWord << endl;
```

```
                        }
                }
        }
}

void TemporaryOutput(Dictionary& dict)
{
        int i = 0;
        int j = 0;
        char h[10] = {0};
        string s_first;

        for(i = 0; i < UNICODE_COUNT; i++)
        {
                s_first = g_CharacterObjective.UTF16toUTF8(UTF16_OrdertoEncoding(i, h), 2,
true);
                if(dict.m_blockTableUTF16[i].nCount > 0)
                {
                        // Read word items.
                        for(j = 0; j < dict.m_blockTableUTF16[i].nCount; j++)
                        {
                                cout << s_first;
                                if(dict.m_blockTableUTF16[i].pWordItemHead[j].nWordLen > 0)
                                {
                                        cout <<
dict.m_blockTableUTF16[i].pWordItemHead[j].sWord;
                                }
                                cout << '\t';
                                cout <<
g_PennPOS[dict.m_blockTableUTF16[i].pWordItemHead[j].nHandle] << '\t';
                                cout << dict.m_blockTableUTF16[i].pWordItemHead[j].nFrequency
<< '\n';
                        }
                }
        }
}

void TemporaryOutputBigrammer(Dictionary& dict, Context& ctx)
{
        int i = 0;
        int j = 0;
        char h[10] = {0};
        string s_first;

        for(i = 0; i < UNICODE_COUNT; i++)
        {
                s_first = g_CharacterObjective.UTF16toUTF8(UTF16_OrdertoEncoding(i, h), 2,
true);
                if(dict.m_blockTableUTF16[i].nCount > 0)
                {
                        // Read word items.
                        for(j = 0; j < dict.m_blockTableUTF16[i].nCount; j++)
                        {
                                cout << s_first;
                                if(dict.m_blockTableUTF16[i].pWordItemHead[j].nWordLen > 0)
                                {
                                        cout << 
dict.m_blockTableUTF16[i].pWordItemHead[j].sWord;
                                }
                                cout << '\t';
                                cout << dict.m_blockTableUTF16[i].pWordItemHead[j].nFrequency 
<< '\t';
                                cout << 
g_PennPOS[dict.m_blockTableUTF16[i].pWordItemHead[j].nHandle/100] << '\t';
                                cout << 
g_PennPOS[dict.m_blockTableUTF16[i].pWordItemHead[j].nHandle%100] << '\t';
                                cout << ctx.m_pContext-
>aContextArray[dict.m_blockTableUTF16[i].pWordItemHead[j].nHandle/100] \
                                        [dict.m_blockTableUTF16[i].pWordItemHead[j].nHandle
%100] << '\n';
                        }
                }
        }
}
```

3. 扩展插件

cwsplugin 命令行操作

```
用法：cwsplugin [选项]...
(支持输入文件或者键盘输入、输出文件或屏幕打印)

长选项必须用的参数在使用短选项时也是必需的。
        -t,     --type=(Cn|Cy|Cws|C2|En)        分词器种类(默认为 Cws)
        -d,     --detail=(yes|no)               是否显示分词偏移量等详细信息(默认为 no)
        -h,     --help                          显示此帮助信息并离开
                --help-seg                      显示分词方式并离开
        -v,     --version                       显示版本自信息并离开
```

插件功能

```
There are some word segmentation method provide by scholar.
And you should note that punctuation would not be result.


1. Cn: Single-word word (单字分词)
        Example: "这是测试 test 句子。"
        Result: "这/是/测/试/test/句/子"

2. Cy: According to the symbol of the word (符号分词)
        Example: "这是测试 test 句子。"
        Result: "这是测试/test/句子"

3. Cws: Chinese word segmentation (中文分词)
        Example: "这是测试 test 句子。"
        Result: "这/是/测试/test/句子"

4. C2: Tow or more word segmentation (n 元分词) (You can change "for circle" n in
ImC2LexAnalyzer.c:155)
        Example: "这是测试 test 句子。"
        Result: "       这
                        这是
                        这是测
                        这是测试
                        这是测试 t
                        这是测试 te
                        这是测试 tes
                        这是测试 test
                        这是测试 test 句
                        这是测试 test 句子
                        是
```

```
                                是测
                                是测试
                                是测试t
                                是测试te
                                是测试tes
                                是测试test
                                是测试test句
                                是测试test句子
                                测
                                测试
                                测试t
                                测试te
                                测试tes
                                测试test
                                测试test句
                                测试test句子
                                试
                                试t
                                试te
                                试tes
                                试test
                                试test句
                                试test句子
                                t
                                te
                                tes
                                test
                                test句
                                test句子
                                e
                                es
                                est
                                est句
                                est句子
                                s
                                st
                                st句
                                st句子
                                t
                                t句
                                t句子
                                句
                                句子
                                子", God, There are so many word, are they?

5. En: English word segmentation (英文分词)
        Example: "This are some tests."
        Result: "this/are/some/test/."
```

配置文件

```
#分词器名称
ImEnLexAnalyzer
```

```
ImCnLexAnalyzer
ImCyLexAnalyzer
ImC2LexAnalyzer
ImCwsLexAnalyzer
```

调用参数

```
m_pAnalyzer = (ImLexAnalyzer*)ImXXLexAnalyzer_new();
ImXXLexAnalyzer_tokenize(m_pAnalyzer, lpsz_input, &p_tokens)
ImXXLexAnalyzer_delete(m_pAnalyzer);
```

调用示例

```
        // initialize the lexical analysis.
        m_pAnalyzer = (ImLexAnalyzer*)ImXXLexAnalyzer_new();

        // ...

        // input string to be analysis.
        const char* lpsz_input = "This is the content you input here.";

        // prepare a result list.
        ImTokenList* p_tokens = NULL;

        // tokenize.
        if(ImXXLexAnalyzer_tokenize(m_pAnalyzer, lpsz_input, &p_tokens)) return;        //
Segmentation.

        // print the token's elements.
        GList* p_list_point = p_tokens->m_pTokens;
        while(p_list_point)
        {
                // get one token from list.
                ImToken* p_token = (ImToken*)(p_list_point->data);

                // print the word content.
                char* lpsz_data = NULL; ImToken_get_data(p_token, &lpsz_data);
                printf("\nword = [%s]\n", lpsz_data);
                if(NULL != lpsz_data) { free(lpsz_data); lpsz_data = NULL; }

                // print the word's length.
                printf("word's length = [%d]\n", (int)ImToken_get_dataLength(p_token));

                // print the word's offset count.
                printf("offset number = [%d]\n", (int)ImToken_get_offsetNumber(p_token));

                // get the offset array.
```

```
            char* p_char_t = NULL; ImToken_get_offsets(p_token, &p_char_t);
            unsigned short* pOffsets = (unsigned short*)p_char_t;

            // printf the specific offset number.
            int n_offset_num = (int)ImToken_get_offsetNumber(p_token), j = 0;
            for( ; j < n_offset_num; j++)
                    printf("\toffset[%d] = [%u]\n", j, (unsigned short)pOffsets[j]);

            // point to the next token.
            p_list_point = g_list_next(p_list_point);
    }

    // free the result token list.
    ImTokenList_delete(p_tokens);

    // ...

    // free the lexical analysis.
    ImXXLexAnalyzer_delete(m_pAnalyzer);
```

4. Demo

常规中文简体分词

demo/main.cpp

添加词

demo/addWord.cpp
demo/addWord_Penn.cpp

字典查询

demo/dict.cpp