//批量拆分单词

procedure AutoSplitWords;

var

sSqlStr : string;

ADataSet : TClientDataSet;

sWordName : string;

iWordTotal : Integer;

sCondition : string;

iSplitSuccessNum : Integer;

begin

//成功拆分的单词数

iSplitSuccessNum := 0;

//是否停止

iStopAutoSplitWords := 0;

ADataSet := TClientDataSet.Create(nil);

try

//循环没有单词结构的单词

if Length(sAutoSplitWordPart)= 1 then

begin

sCondition := ' and vc\_word\_name like ''' + sAutoSplitWordPart + '%''';

end

else if Length(sAutoSplitWordPart)> 1 then

begin

sCondition := ' and vc\_word\_name like ''%' + sAutoSplitWordPart + '%''';

end

else

begin

sCondition := '1 = 0';

end;

sSqlStr := 'select vc\_word\_name from twordinfo'

+ ' where i\_roots\_lock = 0'

+ sCondition

+ ' and vc\_word\_name not like ''%''''%'''

+ ' and vc\_word\_name not like ''% %''';

ADataSet.Data := OpenSQL(sSqlStr);

ADataSet.First;

iWordTotal := ADataSet.RecordCount;

while not ADataSet.Eof do

begin

//

if iStopAutoSplitWords = 1 then Break;

sWordName := ADataSet.FieldByName('vc\_word\_name').Value;

frmServerMain.AddServerTaskLog(IntToStr(ADataSet.RecNo) + ': ' + IntToStr(iWordTotal) + ' 拆分： ' + sWordName);

//如果单词中含有特殊字符，那么就去掉

//如果单词中不包含引号，那么就拆分掉

if not ContainsStr(sWordName, '''') then

begin

//从最长的词根处拆分

if not resolveWord(sWordName, 1) then

begin

//从次长的词根处拆分

if resolveWord(sWordName, 2) then

begin

iSplitSuccessNum := iSplitSuccessNum + 1;

end;

end

else

begin

iSplitSuccessNum := iSplitSuccessNum + 1;

end;

end;

ADataSet.Next;

end;

frmServerMain.AddServerTaskLog('单词拆分结束，成功拆分：' + IntToStr(iSplitSuccessNum) + ' 个');

finally

ADataSet.Close;

ADataSet.Free;

end;

end;

//分解单个单词

function resolveWord(sWordName : string; iSelectRecNo : Integer): Boolean;

var

sRootName, sUsage, sCnMeaning : string;

iSeq : Integer;

iRootType : Integer;

//后缀类型

iGeneralType : Integer;

iLen : Integer;

ADataSet : TClientDataSet;

iCurPos : Integer;

sWordStruct : string;

sModWordName : string;

sSign : string;

i : Integer;

sSqlStr : string;

AList : TStringList;

sWordCnMeaning :string;

//单个字母的位置

iMonoLetterPos : Integer;

begin

Result := False;

//得到该单词的所有词根

//循环所有的词根，得到

ADataSet := TClientDataSet.Create(nil);

AList := TStringList.Create;

try

sWordCnMeaning := getWordCnMeaning(sWordName);

iCurPos := 0;

sModWordName := sWordName;

ADataSet.Data := OpenSQL(getRootSqlStr(sWordName, ''));

//ADataSet.First;

//开始拆分的词根序号（开始拆分第iSelectRecNo个词根）

if ADataSet.RecordCount >= iSelectRecNo then

begin

ADataSet.RecNo := iSelectRecNo;

end

else

begin

Exit;

end;

while not ADataSet.Eof do

begin

//依次拆分单词

sRootName := ADataSet.fieldbyname('vc\_root\_name').Value;

sUsage := ADataSet.fieldbyname('vc\_usage').Value;

sCnMeaning := ADataSet.fieldbyname('vc\_cn\_meaning').Value;

iLen := ADataSet.fieldbyname('rootlen').Value;

iSeq := ADataSet.fieldbyname('i\_seq').Value;

iRootType := StrToInt(VarToStrDef(ADataSet.fieldbyname('i\_root\_type').Value, '0'));

iGeneralType := StrToInt(VarToStrDef(ADataSet.fieldbyname('i\_general\_type').Value, '0'));

//得到当前词根在单词中的位置

iCurPos := Pos(sRootName, sModWordName);

//根据该词根在单词中位置可以做好过滤

{

如果位置为0，那么该词根一定不是后缀

如果位置为最后，那么该词根一定不是前缀

得到该单词的词性

如果该单词为形容词，那么后缀就只能是形容词

}

//后缀不能在最前面

if (iRootType = 3) and (iCurPos = 1) then

begin

ADataSet.Next;

Continue;

end;

//前缀不能在最后面

if (iRootType = 1) and (Length(sWordName) = iCurPos + Length(sRootName) - 1) then

begin

ADataSet.Next;

Continue;

end;

//如果该词根在词尾 且是后缀 iGeneralType

if (Length(sWordName) = iCurPos + Length(sRootName) - 1) and (iGeneralType >=300) then

begin

//如果该单词可以判断词性

if ContainsStr(sWordCnMeaning, '.') then

begin

//不满足下列条件的，肯定不是复合条件的后缀

if

//如果词根是名词后缀，如果在此意思中找不到名词标志

(iGeneralType = 301) and

(not ContainsStr(sWordCnMeaning, 'n.')) or

//如果词根是形容词后缀，如果在此意思中找不到形容词标志

(iGeneralType = 303) and

(not (ContainsStr(sWordCnMeaning, 'adj.') or ContainsStr(sWordCnMeaning, 'a.'))) or

//如果词根是动词后缀，如果在此意思中找不到动词标志

(iGeneralType = 302) and

(not (ContainsStr(sWordCnMeaning, 'v.') or (ContainsStr(sWordCnMeaning, 'vt.')) and not ContainsStr(sWordCnMeaning, 'adv.')))

then

begin

ADataSet.Next;

Continue;

end;

end;

end;

//处于词根bi

if sRootName = 'bi' then

begin

if ContainsStr(sWordCnMeaning, '二') or

ContainsStr(sWordCnMeaning, '两') or

ContainsStr(sWordCnMeaning, '双') then

begin

//二，两

iSeq := 1;

end

else

begin

//生命

iSeq := 2;

end;

end;

//处于词根di

if sRootName = 'di' then

begin

if ContainsStr(sWordCnMeaning, '二') or

ContainsStr(sWordCnMeaning, '两') or

ContainsStr(sWordCnMeaning, '双') then

begin

//二，两

iSeq := 3;

end;

end;

//处于词根phas

if sRootName = 'phas' then

begin

if ContainsStr(sWordCnMeaning, '相') or

ContainsStr(sWordCnMeaning, '阶段') then

begin

//相位，状态

iSeq := 2;

end

else

begin

//说

iSeq := 1;

end;

end;

//处于词根ic

if sRootName = 'ic' then

begin

//如果出现在词尾 且该词是形容词

if (Length(sWordName) = iCurPos + Length(sRootName) - 1)

and (ContainsStr(sWordCnMeaning, 'adj.') or ContainsStr(sWordCnMeaning, 'a.'))

then

begin

//形容词后缀

iSeq := 1;

end;

end;

//处于词根i

if sRootName = 'i' then

begin

//如果出现在词尾

if (Length(sWordName) = iCurPos + Length(sRootName) - 1) then

begin

//形容词后缀

iSeq := 2;

end

else

begin

//音节过度

iSeq := 1;

end;

end;

//a,o,e,i,u出现在此种，那么为音节过度

//处于词根a

if sRootName = 'a' then

begin

//如果不在词尾也不在词首

if not ((Length(sWordName) = iCurPos + Length(sRootName) - 1) or (iCurPos=1)) then

begin

//音节过度

iSeq := 6;

end;

end;

//处于词根o

if sRootName = 'o' then

begin

//如果不在词尾也不在词首

if not ((Length(sWordName) = iCurPos + Length(sRootName) - 1) or (iCurPos=1)) then

begin

//音节过度

iSeq := 3;

end;

end;

//处于词根e

if sRootName = 'e' then

begin

//如果不在词尾也不在词首

if not ((Length(sWordName) = iCurPos + Length(sRootName) - 1) or (iCurPos=1)) then

begin

//音节过度

iSeq := 5;

end;

end;

//处于词根u

if sRootName = 'u' then

begin

//如果不在词尾也不在词首

if not ((Length(sWordName) = iCurPos + Length(sRootName) - 1) or (iCurPos=1)) then

begin

//音节过度

iSeq := 1;

end;

end;

//处于词根re

if sRootName = 're' then

begin

if ContainsStr(sWordCnMeaning, '回') or

ContainsStr(sWordCnMeaning, '后') then

begin

//回，向后

iSeq := 1;

end

else if ContainsStr(sWordCnMeaning, '再') or

ContainsStr(sWordCnMeaning, '重') then

begin

//再，重新

iSeq := 2;

end

else if ContainsStr(sWordCnMeaning, '反') then

begin

//相反，反对

iSeq := 3;

end

else

begin

//加强语气

iSeq := 4;

end;

end;

//处于词根post

if sRootName = 'post' then

begin

if ContainsStr(sWordCnMeaning, '后') then

begin

//后

iSeq := 1;

end

else

begin

//邮政

iSeq := 2;

end;

end;

//处于词根form

if sRootName = 'form' then

begin

if ContainsStr(sWordCnMeaning, '蚁') then

begin

//蚁

iSeq := 3;

end

else

begin

//形状

iSeq := 1;

end;

end;

//处于词根ward

if sRootName = 'ward' then

begin

if ContainsStr(sWordCnMeaning, '向') or

ContainsStr(sWordCnMeaning, '往')or

ContainsStr(sWordCnMeaning, '朝') then

begin

//向

iSeq := 2;

end

else

begin

//保卫，守卫

iSeq := 1;

end;

end;

//处于词根blast

if sRootName = 'blast' then

begin

if ContainsStr(sWordCnMeaning, '爆') then

begin

//爆

iSeq := 2;

end

else

begin

//细胞

iSeq := 1;

end;

end;

sSign := '';

for i := 0 to iLen - 1 do

begin

sSign := '&' + sSign;

end;

sModWordName := AnsiReplaceText(sModWordName, sRootName, sSign);

//生成单词结构（等号前面的部分将用于排序，iCurPos为该词根在单词中的位置）

AList.Add(RightStr('0000' + IntToStr(iCurPos), 3) + '=' + sRootName + '[r' + IntToStr(iSeq) + ']');

AList.Sort;

//AList.CustomSort(DescCompareInt);

sWordStruct := '';

for i := 0 to AList.Count - 1 do

begin

if sWordStruct = '' then

sWordStruct := AList.ValueFromIndex[i]

else

sWordStruct := sWordStruct + ' + ' + AList.ValueFromIndex[i];

{

//判断是否含有单个字母，如果有的话，那么就得到该字母的位置

//根据该位置计算出上一个位置的字母，然后比较是否一样

if ContainsStr(AList.ValueFromIndex[i], '+ b[r') then

begin

iMonoLetterPos := StrToInt(AList.Names[i]);

if not ((iMonoLetterPos > 2) and (Copy(sWordName, iMonoLetterPos-1, 1) = 'b')) then

begin

ADataSet.Next;

Continue;

end;

end;

}

end;

//单词被拆分准确了

if DecodeROOTs(sWordStruct) = sWordName then

begin

//对拆分结果进行分析

//词根结构中双写辅音b，但单词中却没有两个b相连接

sSqlStr := 'update twordinfo'

+ ' set vc\_word\_struct = ''' + sWordStruct + ''''

+ ' where vc\_word\_name = ''' + sWordName + '''';

ExecSQL(sSqlStr);

Result := True;

break;

end;

if Length(DecodeROOTs(sWordStruct)) > Length(sWordName) then

break;

//sCondition := sCondition + ' and vc\_root\_name not like ''%' + sRootName + '%''';

ADataSet.Data := OpenSQL(getRootSqlStr(sModWordName, ''));

end;

finally

ADataSet.Close;

ADataSet.Free;

AList.Clear;

AList.Free;

end;

end;