

频繁子树模式在中心词识别中的应用研究

田卫东, 黄 勇

(合肥工业大学 计算机与信息学院, 安徽 合肥 230009)

摘 要: 中文问句中心词识别领域中, 现有方法未能有效利用依存句法中的深层统计关系. 为解决此问题并探究中心词在词的多维属性上的统计关系, 首次提出多维树概念, 给出多维频繁模式挖掘方案并应用于中文问句中心词识别中. 针对此应用给出频繁子树模式精简及规则冲突解决方案, 训练出一个中文中心词识别模型. 此方法是典型的客观方法, 实验表明, 此方法有较好的稳定性、适应性与鲁棒性, 且较条件随机场模型在准确率上有进一步提高.

关键词: 条件随机场; 依存关系树; 频繁子树模式; 模式精简; 规则冲突; 中心词

中图分类号: TP391

文献标识码: A

文章编号: 1000-7180(2015)11-0027-06

Study on the Application of Frequent Sub-tree Patterns in Focus Words Recognition

TIAN Wei-dong, HUANG Yong

(School of Computer & Information, Hefei University of Technology, Hefei 230009, China)

Abstract: In the field of Chinese Focus-words Recognition, current studies don't take full advantages of some deep statistical relationships in dependency syntax. To solve this problem and explore statistical relationships between Chinese focus words and the multiple properties of words, a new concept called Multi-Dimensional Tree (MDT) and a solution of mining frequent MDT pattern are proposed and applied. Solutions of condensing those frequent patterns and dealing with pattern conflicts are given, a Chinese focus words recognizer is trained. The method is a kind of typical objective method, the empirical results show that this method is good at stability, adaptability and robustness and can reach higher recognition accuracy rate than CRF model.

Key words: conditional random field; dependency relationship tree; frequent sub-tree pattern; condensing pattern; rule conflict; focus words

1 引言

与文本分类类似, 问题分类的目标就是将问句文本准确地划分到预先定义好的语义类别中^[1]. 通过语料整理发现, 在特征集合中, 问句中疑问词和中心词保留了相对较多的问题信息, 对问题分类有着举足轻重的作用^[2]. 对于大多数问题, 仅通过这两个特征便可将问题准确分类^[3]. 问题分类对 QA 系统研究和网络舆情分析等自然语言相关领域有着举足轻重的地位与作用.

例如问句 Q1“DNA 是什么意思?”, 其所寻求的

答案是 DNA 的意思, 故其应该归属到语义类别 DES_MEANING 中, 而其中心词“意思/n”则能够直接将该句划分到该类别中.

目前, 中文问句中中心词的抽取主要运用启发式规则. 启发式规则多为专家设计的规则库, 具有强烈的主观性及规则覆盖问题. 例如, 孙景广等人将中心词定义为“疑问意向词”^[4], 其选取疑问词周围的名词作为中心词并且优先选取疑问词右侧的词. 田卫东等人通过对中心词与疑问词位置关系的统计总结出两条启发式规则^[2].

因此, 统计规则方式越发获得学者的重视. 其

收稿日期: 2015-01-23; 修回日期: 2015-03-12

基金项目: 国家“八六三”高技术研究发展计划(2012AA011005); 国家自然科学基金(61273292)

中,公认较好的是条件随机场(CRF)^[5]模型.在数据挖掘思想的指导下,田卫东等利用 CRF 与错误驱动相结合的方式,也取得了不错的效果^[1].

然而,现有方法一般均未能有效利用依存句法中的深层统计关系.本文首次提出多维频繁子树的概念,探究此深层关系和中心词在词的多维属性上的统计关系.同时,提出得当的规则精简及冲突解决策略,训练出中文中心词识别模型.

2 相关概念

中心词最初被定义为“The most important word for comprehending a sentence”^[6].随着对中心词研究的深入及其在问题分类中越发重要的作用,中心词的概念及作用也越发清晰.Li 和 Roth 首次提出“Head Chunks”^[7]的概念,为疑问词后第一个动词或名词块并被看作问题分类的重要特征,并在 TREC10 中对自由格式的问题分类取得良好的分类效果.

本文所述中心词为中文问句的中心词,是问句中最能描述其所寻求答案的内容、类别的词或词组.中心词不仅能揭示答案的特征,更能为问题分类提供必要、有效的支撑,能为答案抽取及合成提供方法策略.

树 $T = (V, E)$ 是一种有根结点的有向无环连通图,其中 V 为树的结点集, E 为树的边集.树中,有唯一结点 $r \in V$ 称为树根.

依存语法^[8]是一种使用非常广泛的语法形式,由法国语言学家 L. Tesiniere 提出.依存关系树完整地保留了句子的语义信息,显式地将句子的结构特征等隐含信息表示出来,因此,可将对应原句子处理转为对相应依存关系树处理.

3 多维树

3.1 多维树

定义 1 (多维树) 给定树 $T = (V, E)$, 其中 $V = \{\text{props} \mid \text{props} \neq \emptyset, \text{props} \subset \text{VEC}\}$ 为树 T 的结点集, $E = \{(x, y) \mid x, y \in V\}$ 为树的边集, 称树 T 为多维树. VEC 为结点的属性列表, 亦称树的维列表.

性质 1 当给定多维树 T , 有且仅有一个维列表与之对应并也随之确定.

由定义可知, 相较于传统树, 多维树的最大特点是其每个结点都由至少一个属性值组成, 即多维树的每个结点值都是一个属性集合.

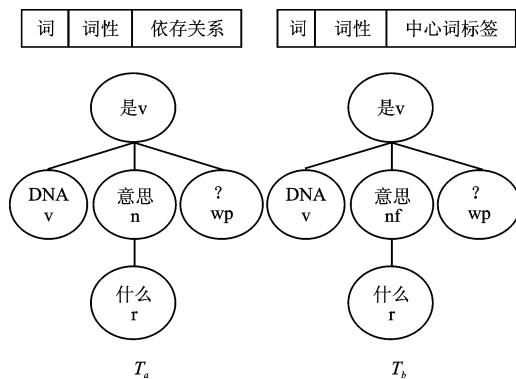


图 1 多维树示例

图 1 所示为中文问句 Q1:“DNA 是什么意思?”的依存关系树及对应的维列表.

定义 2 (维) 给定多维树 T , 对于任一结点 $n \in T$, 函数 $\dim_T(n) = |n|$ 称为结点 n 的维, 其中 $| \cdot |$ 表示多维树结点 n 的属性集合的势. 称 $\dim(T) = \arg\max \dim_T(n)$ 为多维树 T 的维.

性质 2 对于任意给定的多维树 T , 其结点维的值为 $[1, \dim(T)]$.

给定树 $T, \exists m, n \in T$ 使 $\dim_T(m) \neq \dim_T(n)$ 成立, 称树 T 为固定多维树; 反之为可变多维树.

称维度为 k 的树为 k 维树. 可知图 1 中, 树 T_a 是一棵固定多维树, 而 T_b 是一棵可变多维树.

3.2 频繁多维子树

由于多维树的结点值是属性集合, 故多维子树不仅包括传统意义上的子树, 即结构意义上的子树, 还包括集合意义上的子树及二者混合产生的子树. 令 $S \leq T$ 表示 S 是 T 的子树, 则三种子树定义如下所示.

定义 3 (结构多维子树) 给定多维树 $S = (V_s, E_s), T = (V_t, E_t)$, 如果 $S \leq T$, 且对于任意 $x \in V_s, \varphi(x) \in V_t$, 有 $\dim_s(x) = \dim_t(\varphi(x))$ 成立, 则称树 S 是树 T 的结构多维子树. 结构多维子树分为诱导结构多维子树及嵌入式结构多维子树.

性质 3 一棵有 n 个结点的多维树 T 的结构多维子树(森林)个数 st_s 有

$$st_s(T) = 2^n - 1.$$

定义 4 (嵌入式子树的覆盖) 给定树 T 及其嵌入式子树 S_e , 在 T 中必存在一棵诱导子树 S_i , 使得: (1) $S_e \leq S_i$; (2) $\text{root}(S_e) = \text{root}(S_i)$; (3) S_e 中所有的叶结点均为 S_i 中的叶结点. 称 S_i 为 S_e 在 T 上的覆盖, 记为 $\text{cover}(T, S_e) = S_i$.

定义 5 (维度多维子树) 给定多维树 $S = (V_s, E_s), T = (V_t, E_t)$, 如果 $E_s = E_t, |V_s| = |V_t|$, 且

存在 $x \in V_s, \varphi(x) \in V_t$, 有 $\dim_s(x) \neq \dim_t(\varphi(x))$ 成立, 则称树 S 是树 T 的维度多维子树。

性质 4 给定一棵多维树 T , 若 $\forall n \in T$, $\dim_T(n) = m$, 则 T 的非空维度多维子树个数 st_d 有 $st_d(T) = (2^m - 1)^n$ 。

定义 6 (混合多维子树) 给定多维树 $S = (V_s, E_s)$, $T = (V_t, E_t)$, 如果 $S \leq T$, $\exists x \in V_s, \varphi(x) \in V_t$, 且 $\dim_s(x) \neq \dim_t(\varphi(x))$ 成立, 则称树 S 是树 T 的混合多维子树。

本文将上述类型子树统一称为多维子树。

性质 5 给定一棵多维树 T , 若 $\forall n \in T$, $\dim_T(n) = m$, 则 T 的非空子树(森林) st 个数有

$$st(T) = \sum_{i=1}^n C_n^i (2^m - 1)^i.$$

定义 7 (支持数) 令 D 表示多维树数据库, T 为某一多维模式树, 称 D 中出现 T 的记录条数为 D 对 T 的支持数, 记为 $supp_D(T)$ 。

定义 8 (频繁多维子树) 在多维树数据库 D 中, 给定最小支持度阈值 $minsupp$, 对于某一多维模式树 T , 当且仅当 $supp_D(T) \geq minsupp$ 时, 称多维模式树 T 在 D 中是频繁的。

3.3 多维树存储

文中采用图 2 所示的形式存储及表示多维树。其中, 结点序号是指将树中结点按某一规则编号, 如先序深度遍历。根结点的父结点序号是一个不同于所有结点序号的值。这样的存储方式也能满足有序树的要求, 在向量中从头到尾查看, 某个结点的孩子顺序正好与向量顺序是一致的。

```
typedef struct _mdt_node {
    int id;           // 结点序号
    vector<string> props; // 词的属性集
    int parent;       // 父结点编号
} MDTN;
typedef vector<MDTN> MDT;
```

图 2 多维树结点结构

3.4 频繁多维子树挖掘

图 3 是本文多维频繁子树挖掘的总体框架图。

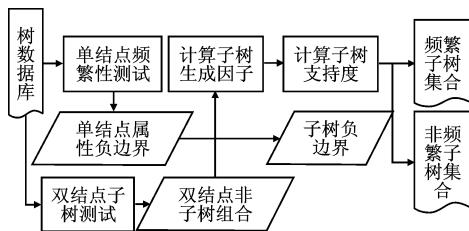


图 3 多维子树挖掘总体框架图

本文主要从三个方面对子树生成逻辑进行控制: 首先是对树中单结点进行频繁性测试, 计算结点内属性集的频繁性, 记录负边界; 第二, 在树中进行两两结点的子树测试, 记录所有由两个结点生成的子树森林; 最后, 对子树进行频繁度计算, 记录子树的负边界。具体逻辑如算法 1 所示。

算法 1: 频繁多维子树挖掘算法

输入: (1) T : 当前正在处理的树;

(2) $minsupp$: 最小支持度阈值;

(3) $indices$: 指定属性在维列表中的下标。

输出: (1) $freqST$: D 中所有的频繁子树;

```

01 function MDSTM(D, minsupp, indices);
02   freqST ← ∅;
03   infreqST ← ∅;
04   foreach tree t in D do begin
05     Initiate(selector, 0, t.size());
06     NBorderST ← ∅;
07     subforest ← SubForestTest(T);
08     NBorderNode ← NegativeBorder4NodeTest(T);
09     next ← false;
10     while not next do begin
11       next ← NextSelector(subforest, NBorderNode, NBorderST, selector);
12       if next then
13         break;
14       subtree ← GenerateSubtree(t, selector);
15       if subtree.empty() or processed(subtree) then
16         continue;
17       supp ← CountSupport(subtree);
18       if supp ≥ minsupp then
19         freqST ← freqST ∪ {subtree};
20       else begin
21         infreqST ← infreqST ∪ {subtree};
22         NBorder_ST ← NBorder_ST ∪ {subtree};
23       end.
24     end.
25   end.
```

4 规则生成

定义 9 (统计规则) 给定树库 D 及 $T_a, T_b \in D$ 且 $T_a \leq T_b$, 称 $T_a \Rightarrow T_b, cf_D$ 为统计规则。其中, cf 为树库 D 中该统计规则的置信度, 计算如下:

$$cf_D(T_a \Rightarrow T_b) = \frac{supp_D(T_b)}{supp_D(T_a)}$$

本文按图 4 所示的形式存储统计规则。

```
typedef struct _rule {
    MDT cond; // 规则前件
    MDT deci; // 规则后件
    int target; // 目标结点编号
    char label; // 目标结点标签
    double cf; // 规则置信度
} Rule;
```

图 4 统计规则的存储方式

算法 2: 规则生成算法

输入: (1) T : 频繁依存子树集 (2) D : 语料树库
(3) i : 针对 T 中第 i 条子树生成规则
(4) mincf: 最小置信度阈值
输出: (1) R : T 中第 i 条子树的直接嵌入式规则集

```
01 function GenerateEmbeddedRules( $T, D, i, \text{mincf}$ );
02   target  $\leftarrow \emptyset$ ;
03   for  $j \leftarrow 0$  to  $T[i].\text{size}()-1$  do
04     if  $T[i][j]$  is not a leaf node then
05       target  $\leftarrow \text{target} \cup \{j\}$ ;
06   for  $j \leftarrow 0$  to  $\text{target.size}()-1$  do begin
07     count  $\leftarrow \text{TargetCounter}(D, T, i, \text{target}[j])$ ;
08     cf  $\leftarrow \text{float}(\text{count}) / T[i].\text{back}()$ ;
09     if cf  $< \text{mincf}$  then
10       continue;
11     FillRule(rule,  $T[i]$ ,  $T[i]$ ,  $\text{target}[j]$ , cf);
12     rules  $\leftarrow \text{rules} \cup \{\text{rule}\}$ ;
13   end.
14   return rules;
```

由于嵌入型直接规则的目标结点只可能是非叶结点, 算法 2 首先将所有非叶结点标记为可能的目标(04~05 行), 再一一计算置信度, 将符合要求的规则纳入规则库。由于诱导规则生成逻辑与嵌入型规则类似, 本文不再列出。

5 规则精简及选取

5.1 规则精简

关联规则挖掘的一个普遍问题就是产生的关联规则数量庞大, 严重影响规则库的使用效率。识别并去除冗余规则也成为重要的研究热点。对关联规则库分析得知压缩规则库是可行的, 并且是十分有用的。

定义 10 (基规则) 给定规则库 RS 及其中某规则 $B: T_a \Rightarrow T_b$, 若 B 蕴含的信息不能由其他任何规则经过运算得到, 称 B 是 RS 的基规则。

定义 11 (冗余规则) 若规则 R 能由某些基规则经过运算得到, 则称规则 R 为冗余规则。

为识别冗余规则, 本文针对不同类别的冗余规则进行如下描述, 并以此为指导压缩规则库。

定义 12 (重复冗余规则) 给定规则 $R: T_a' \Rightarrow T_b$, 若有某基规则 $B: T_a \Rightarrow T_b$ 使得 $T_a \leq T_a'$, 且有 $\text{cf}(R) \leq \text{cf}(B)$ 成立, 则称 R 为 B 的重复冗余规则。

定义 13 (导出冗余规则) 给定规则 $R: T_a \Rightarrow T_c$, 若有基规则 $B_1: T_a \Rightarrow T_b$ 及 $B_2: T_b \Rightarrow T_c$, 则称规则 R 为导出冗余规则。

由于本文考虑嵌入式规则, 故重复冗余规则中有一类特殊的规则只有具体到某条问句时才能确定其是否冗余, 称这类规则为限定冗余规则。

定义 14 (限定冗余规则) 给定规则 $R: T_a' \Rightarrow T_b$ 及基规则 $B: T_a \Rightarrow T_b$, T_a 与 T_a' 没有子树关系。若在给定问句 Q 中, $\text{cover}(Q, T_a') \geq \text{cover}(Q, T_a)$, 称规则 R 为 B 在 Q 中的限定冗余规则。

可知, 只要全部保留了基规则, 就可以保证规则库的信息完整性。通常, 一个规则库中, 基规则只占其中不大的一部分, 因此, 删除其中大部分规则, 并不影响整个规则库的信息完整性。

定理 1 称规则 R 是冗余的, 则必定可以找到某些基规则 $B_i, i = 1, \dots, m$ 运算得到 R 中的信息证明如下所示。(1) 对于第二类规则, 假定有规则 R_2 及其冗余规则 R_1 : ① 若 R_2 是非冗余的且不是基规则, 与定义矛盾, 假设不成立; ② 若 R_2 是冗余的, 则可找到某规则 R_3 包含其所有信息。由 ① 知 R_3 必定为冗余的或基规则, 由归纳法可知最终必有某基规则 B 包含 R_1 的全部信息。(2) 同理可证第三类规则。

实际上, 定理 1 揭示的是冗余规则的一个充分必要条件。

规则精简只能处理重复冗余规则与导出冗余规则, 而限定冗余规则只能在具体到某个问句时才能确定。压缩规则库算法如下所示。

算法 3: 规则压缩算法

输入: (1) Rules: 待压缩的规则库
输出: (1) BaseRules: 压缩后的规则库

```
01 function CondenseRules(Rules);
02   BaseRules  $\leftarrow \emptyset$ ;
03   foreach rule  $r$  in Rules do begin
04     if BaseRules is empty then begin
05       BaseRules  $\leftarrow \{r\}$ ;
```

```

06    continue;
07    end.
08    foreach rule br in BaseRules do begin
09        temp ← GetRedundant(r, br);
10        if temp is null then
11            BaseRules ← BaseRules ∪ {r};
12        else if temp is r then
13            break;
14        else if temp is br then
15            BaseRules ← BaseRules - {r};
16        end.
17    end.
18    return BaseRules;

```

5.2 规则选取

定义 15 (规则冲突) 对于某一问题,有不少于两个持不一致意见的规则可以适配该问题,称在这个问题上出现规则冲突。

事实证明,即使经过精简的规则库,在实际应用中发生规则冲突也几乎是必然的。规则冲突的一个最直接也最明显的影响是加大规则库应用的难度,影响规则库预测的准确率。

定义 16 (规则簇) 对于某一问题,持相同观点的若干规则构成一个规则集合,称该集合为规则簇,对应观点为规则簇的核心,简称簇核。

由此,可将规则选取演变为对规则簇的选取。本文假定簇间是相互独立的,定义簇的参考度作为簇间比较的衡量指标。

定义 17 (簇的参考度) 在适配某问题的 r 条规则中,有 m 条规则构成一个规则簇 C ,则 C 的参考度 ref 的计算公式为

$$\text{ref}(C) = k \cdot \frac{\sum_{j=1}^r \text{cf}_j \cdot \left(\sum_{i=1}^m \text{cf}_i - \sum_{i=1}^m \text{cf}_i / \sum_{j=1}^r \text{cf}_j \right)^2}{\sum_{i=1}^m \text{cf}_i},$$

式中, k 为簇内规则数的作用因子, $k \in (0, 1]$, 本文中,取 k 为簇内规则数的倒数。

假设有某一问题 Q ,有 r 条规则适配该问题,本文制定如下策略解决规则冲突问题:

- (1) 若所有 r 条规则构成一个规则簇,则选取置信度最大的规则;
- (2) 若 r 条规则构成若干个规则簇,则选取有最大参考度的簇并在其中按 1 方式选出规则;
- (3) 若还不能选取规则,则在 r 条规则中选取置信度最大的规则。

6 实验结果及分析

本文采用哈尔滨工业大学信息检索实验室的语言技术平台(LTP-Cloud)的标注体系及实验数据集。实验环境为 Windows 7 Pro. 32 bit,处理器为 Intel 2.8 GHz 处理器,安装物理内存为 2 GB。

为使实验客观准确,本文采用 10-fold 方式划分实验数据,将 CRF 方法作为对比方法,并将 CRF 的训练模板设置与本文实验一致。

本文用表 1 所列出的评价指标来评价中心词识别的准确率。

表 1 实验评价函数

FR	$\text{FR} = \frac{\text{被正确标注的中心词数}}{\text{所有中心词数}}$
SR	$\text{SR} = \frac{\text{句子标注正确数}}{\text{句子总数}}$
NFW	$\text{NFW} = \frac{\text{非中心词被标注数}}{\text{词总数}}$

其中,“句子标注准确”有两层意思:(1)若句子中有中心词,且中心词被正确识别,则句子标注正确;(2)若句子中没有中心词,且算法也没有将任何词标注为中心词,则句子标注正确。“非中心词被标注”是指将非中心词错误地识别成中心词。NFW 值越小,说明方法效果越好。

本文在 10-fold 实验中运用控制变量法观察 minsupp 及 mincf 对中心词识别的影响,确定二者分别为 2 及 0.3。此方法虽不能说明该取值是最优的,但为方便起见,本文后续实验均采用此阈值。

本文规则精简实验数据如图 5 所示。频繁多维子树规则压缩率一般都在 97.5% 之上,说明本文对规则库的压缩是明显的。

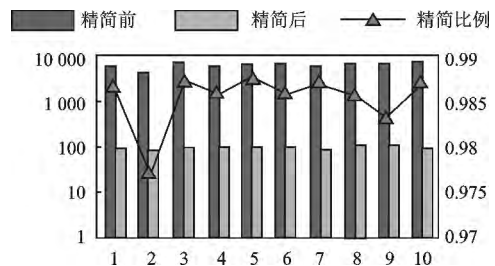


图 5 规则精简效率结果

如图 6 所示,本文方法在中心词识别准确率方面已经接近甚至超过 CRF。实验说明,本文方法在中心词识别方面是有效的。

- (1) 本文方法较 CRF 方法在 10 组实验中均有

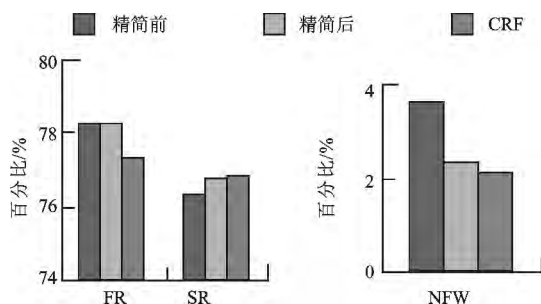


图6 10组实验平均结果

一定程度的提升,并且提升平均有1.05%。说明本文方法是有效的。

(2)原始规则库与压缩后的规则库在中文中心词识别准确率方面表现相当,说明本文规则精简策略是有效的。

(3)原始规则库识别中心词时,NFW水平较高,说明冗余规则对规则库在中心词识别任务中有一定的负面影响。

7 结束语

本文首次提出多维树的概念,弥补了树挖掘在多维树方面的不足,同时,比较全面地阐述了多维频繁子树的冗余特性,并提出一种多维子树模式的精简策略。实验说明,这种策略是行之有效的。

本文为中文问句中心词识别提供一种新的思路,实验表明,中心词在问句依存关系上的特征对中心词识别的帮助是显著的。与此同时,就中心词应用本身阐述了规则冲突的特点,并提出一种解决规则冲突的策略。

中心词识别一直以来都是一个困难的问题,以依存关系树来辅助识别中心词也有其不足之处。针对这些问题展开新的研究是本文下一步需要进行的工作;同时,规则冲突也是一个不可避免并亟待解决的问题,在此方面,本文也将尝试更合理的解决方

案,以进一步提高标注准确率。

参考文献:

- [1] Liu Xiaoming, Liu Li. Question classification based on focus[C]// 2012 International Conference on Communication Systems and Network Technologies. India, Chandigarh, IEEE, 2012: 512-516.
- [2] 田卫东,李亚娟. 基于CRF和错误驱动的中心词识别[J]. 计算机应用研究, 2013, 30(8): 2345-2348.
- [3] 田卫东,高艳影,祖永亮. 基于自学习规则和改进贝叶斯结合的问题分类[J]. 计算机应用研究, 2010, 27(8): 2869-2871.
- [4] 孙景广,蔡东风,吕德新,等. 基于知网的中文问题自动分类[J]. 中文信息学报, 2007, 21(1): 90-95.
- [5] Lafferty J, McCallum A, Pereira F C N. Conditional Random fields: Probabilistic models for segmenting and labeling sequence data[C]// 18th International Conference on Machine Learning 2001, ICML 2001. USA, Williamstown, 2001: 282-289.
- [6] Osaka M, Nishizaki Y, Komori M, et al. Effect of focus on verbal working memory: Critical role of the focus word in reading[J]. Memory & Cognition, 2002, 30(4): 562-571.
- [7] Li X, Roth D. Learning question classifiers[C]// Proceedings of the 19th international conference on Computational linguistics-Volume 1. Association for Computational Linguistics. Taipei, 2002: 1-7.
- [8] Tesnière L, Fourquet J. Eléments de syntaxe structurale[M]. Paris: Klincksieck, 1959.

作者简介:

田卫东 男,(1970-),副教授.研究方向为智能计算与数据挖掘。

黄勇(通讯作者) 男,(1990-),硕士研究生.研究方向为数据挖掘. E-mail: zhuixunxiyang@126.com