一种基于扩展概念图的词义识别算法

李 虹 李 磊
(广州中山大学软件研究所 广州510275)

摘 要 本文提出了一种基于扩展概念图的词义识别算法。该算法通过搜索概念图,寻找待识别词的两两词义之间的祖先分叉点和分叉路径,从而找到词义之间的相对差异路径,即决定路径。结合上下文词语的出现频率,该算法可以计算出上下文词语对各决定路径的支持度。而词义之间的相对决定路径的支持度的差别,正好反映了词义对待识别词的相对适合程度。本文提出的算法就是通过计算和比较这种差别,最终选出最适合待识别词的词义。为了对所提出的算法进行评估和比较,我们借助 WordNet 1.6和 SemCor 进行测试。测试结果表明,该算法具有较高的词义识别效率和准确度。

关键词 词义识别,概念图,最大祖先决定点,决定路径,WordNet,SemCor

A Word Sense Disambiguation Method Based on Extended Conceptual Ontology Graph

LI Hong LI Lei
(Institute of Softwore Reaserch, Zhongshan University, Guangzhou 510275)

Abstract This paper presents a word sense disambiguation algorithm based on extended conceptual ontology graph. The algorithm tries to find the branches of the paths converging to the same common ancestor nodes from two senses of the word to be disambiguated, by searching the crotch of the ancestor nodes of the two senses. The branches of the paths here are called the decisive paths for one sense against another. Then, with the help of the context words in the running text, it calculates each decisive path's support, compares each two senses and finally chooses the sense with top support of the decisive paths as the best suited sense. For evaluation and comparison, we test our algorithm on SemCor according with WordNet release 1.6. The result shows that our algorithm can disambiguate word sense with quite efficient speed and promising precision.

Keywords Word sense disambiguation, Conceptual ontology graph, Greatest decisive ancestor node, Decisive path, WordNet, SemCor

1 引言

词义识别在计算机语言学领域是一个长期存在的、至关重要的问题。多年以来,人们提出了各种各样的解决办法,包括:基于机器可读的词典的词义识别方法[1,2],使用训练文集或相关词组的统计识别方法[3,4]以及使用概念之间的关系作为识别基础的词义识别方法[5~9]等。

随着对词义识别方法的不断探索,人们累积了越来越多宝贵的词汇资源,为大家能够使用已有的资源来探索出更优的词义识别方法奠定了基础。公共领域词汇知识库 WordNet[10]就是当中最重要的词汇资源之一,因此近年来,人们所研究的词义识别算法也主要围绕 WordNet 展开。一般来说,利用 WordNet 辅助词义识别的算法思想是:寻找待识别词的各词义与上下文词语的各词义的共性,最终选取与上下文词语共性最大的词义作为待识别词的词义[9];或者是:利用 WordNet 提供的词义关系,选取在一定几何形状内获得上下文词语支持最大的词义作为待识别词的词义[8]。对于前一类算法,假设待识别词具有 n 个词义,共有 m 个上下文词语,每个平均具有 t 个词义,则寻找共性时路径搜索的次数至少为 $(n+m*t)$,路径搜索时每扩展一个结点,相应的词义结点需要与 $m*t$ 个上下文词义结点进行祖先结点的判定,因而效率并不是十分理想;对于后一类算法,基本固定的几何形状限制了算法的适应性,使它只能局限地工作在 WordNet 的框架内。事实上,WordNet 所提供的词义和词义之间的关系已经构成了一个具体的概念图样例,基于 WordNet 的词义识别算法隐含了利用概念图辅助词义识别的思想。但是,由于 WordNet 只是一个具体的概念图样例,并不具备抽象性和代表性,同时,WordNet 提供的语种也很有限,不包括中文,因此基于 WordNet 的词义识别算法也就缺乏一定的通用性。

针对以上几点,本文提出了基于扩展概念图的词义识别算法。该算法在识别词义时,先将待识别词的词义两两分为一组,然后搜索概念图,为每组词义寻找组内词义结点的公共祖先,从而找到它们之间的祖先分叉点,即祖先决定点,进而找到它们的分叉路径,即决定路径。接着,通过计算上下文词语对决定路径的支持度,我们得出一个词义相对于另一个词义在上下文中对待识别词的适合度。通过两两词义的比较,最终识别出一个在上下文中最适合待识别词的词义。

根据算法思想,对于具有 n 个词义的待识别词,本算法只需进行 n 次路径搜索,搜索过程中,每扩展一个结点,相应的词义结点只需和其余 $n-1$ 个词义结点进行祖先结点的判定,相对于 $(n+m*t)$ 次路径搜索和每个扩展结点的 $m*t$ 次判定,它具有较高的识别效率。此外,由于本算法的识别准确度主要依赖于分布在决定路径上的上下文词语,决定路径没有固定的几何形状,能够随着概念图的改变而改变,因此它具有更强的适应性。更重要的是,本算法所基于的扩展概念图,它既可以是确切的概念树,又可以是模糊的概念图,所以该算法具有一定的通用性和可扩展性。而该算法对概念图的唯一要求是:概念图中必须包含概念之间的隶属关系。

本文第2节将介绍相关的预备知识和概念;第3节给出本文所提出的词义识别算法;第4节给出相关的测试数据和结果;最后是小结。

2 预备知识和概念

引用1[12] 令 C 是一组概念,按照概念的外延把 C 划分

为不同的层次，构成一个有向的、带权值的、非循环的、无重边的多分图，称为模糊概念图。其中每个结点表示一个概念，最上层的结点代表最一般的概念，最下层的结点代表最具体的概念，结点 p 所在的概念层记作 $level(p)$。弧反映两个相邻层次上结点之间的隶属关系，权值表示隶属度。一般地，模糊概念图 $G=\{V,E\}$ 可以表示为

(1) $V(G)=\{p|p\in C\}$；

(2) $E(G)=\{\langle p,q\rangle|0<weight(p,q)\leqslant 1, level(q)=level(p)-1\}$；

(3) $\forall\ p\in V(G), \sum_{\substack{level(q)=level(p)-1\\ \langle p,q\rangle\in E(G)}} weight(p,q)\leqslant 1$

概念在含义上的复杂性可能造成概念的层次不清晰，一个概念可能直接隶属于不同层次上的多个概念。为此，用超弧来表示两个或以上层次的两个概念之间的关系，记作 $E'(G)=\{\langle p,q\rangle|weight(p,q)\in[0,1], level(q)<level(p)-1\}$，我们称之为扩展模糊概念图，简称扩展概念图。如图1，左图是扩展概念图的样例，描绘了来自词典 *thesaurus*1的词义和词义之间的关系。

定义1 已知扩展概念图 $G(V,E)$，$G'(V',E')$，$p(n_0,\cdots,n_x,\cdots,n_y)$，$p'(n_0,\cdots,n_x,\cdots,n_y)$ 分别是 G 和 G' 中的两条路径，则 $p=p'$，当且仅当：(1) $|p|=|p'|$，即两条路径的长度相同；(2) $\forall\ i\leqslant|p|$，对于 p 中第 i 个位置的结点 n_i，设 $C(n_i)$ 表示 n_i 的概念含义，则对于 p' 中同一位置的结点 n_i' 必须有 $C(n_i')=C(n_i)$；(3) $\forall\ i<|p|$，对于 p 中第 i，$i+1$ 个位置的结点 n_i,n_{i+1}，p' 中第 i，$i+1$ 个位置的结点 n_i',n_{i+1}'，必须有 $weight(n_i,n_{i+1})=weight(n_i',n_{i+1}')$。

定义2 已知扩展概念图 $G(V,E)$，$G'(V',E')$，则 G' 是 G 的等价单亲概念图，当且仅当：(1) $\forall\ v'\in V'$，v' 最多只有一个父结点；(2)对于 G 中的任意路径 p，在 G' 中找到且仅找到一条路径 p'，使得 $p=p'$ (3)对于 G' 中的任意路径 p'，在 G 中找到且仅找到一条路径 p，使得 $p=p'$。

如图1所示，右图是左图的等价单亲概念图。

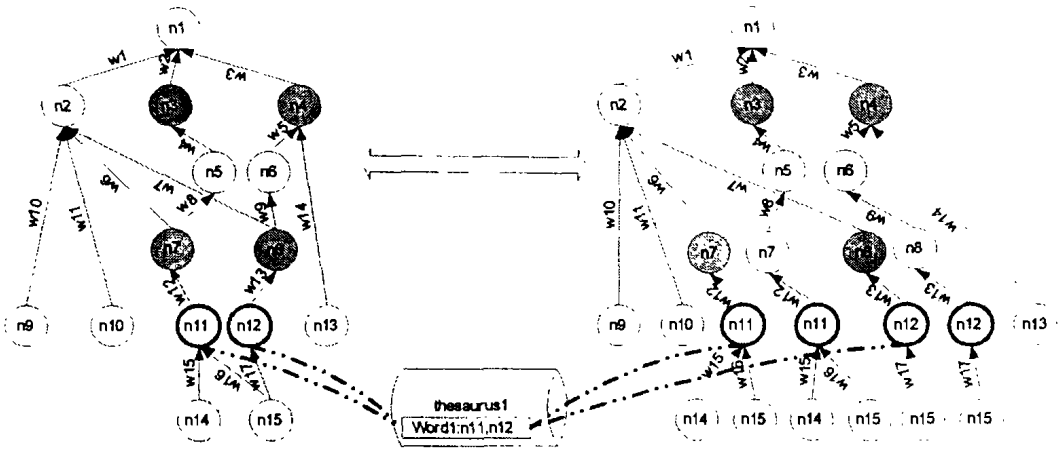

图1 扩展概念图样例：右图是与左图等价的单亲概念图。图中被小黑点填充的灰色圈代表 $n11$ 相对 $n12$ 或 $n12$ 相对 $n11$ 的最大祖先决定点。

定义3 已知待识别词 word 的词义集合 S，扩展概念图 $G(V,E)$，$n_y,n_z\in S\subseteq V$，则 n_y 相对于 n_z 的祖先决定点集合的定义如下：(1)若 n_y 和 n_z 不存在祖先和后代的关系：$AncestorDecisiveNodeSet(n_y,n_z)=\{n_x|n_x\in V, n_x\neq n_z$，并且 n_x 是 n_y 的祖先，n_x 不是 n_z 的祖先$\}$；(2)若 n_y 是 n_z 的祖先或后代：$AncestorDecisiveNodeSet(n_y,n_z)=\{\}$，那么，$n_x$ 是 n_y 相对于 n_z 的祖先决定点，当且仅当 $n_x\in AncestorDecisiveNodeSet(n_y,n_z)$。

如图1的左图，$AncestorDecisiveNodeSet(n11,n12)=\{n3,n5,n7\}$。

定义4 已知待识别词 word 的词义集合 S，扩展概念图 $G(V,E)$，$n_y,n_z\in S\subseteq V$，$n_u,n_v\in AncestorDecisiveNodeSet(n_y,n_z)$，则 $>_d(n_u,n_v,G)$ 为真，当且仅当在 G 中 n_u 是 n_v 的祖先；否则，$>_d(n_u,n_v,G)$ 为假。

定义5 已知待识别词 word 的词义集合 S，扩展概念图 $G(V,E)$，与 G 等价的单亲概念图 $G'(V',E')$，$n_y,n_z\in S\subseteq V$，n_y 相对于 n_z 的祖先决定点集合 D，则 n_y 相对于 n_z 的最大祖先决定点集合的定义如下：$GAncestorDecisiveNodeSet(n_y,n_z)=\{n_x|n_x\in D$，并且，对于$\forall\ v\in D, v\neq n_x$，满足 $>_d(v,n_x,G')$ 为假$\}$，那么，n_x 是 n_y 相对于 n_z 的最大祖先决定点，当且仅当 $n_x\in GAncestorDecisiveNodeSet(n_y,n_z)$。

如图1，$GAncestorDecisiveNodeSet(n11,n12)=\{n3$，以 $n2$ 为父结点的 $n7$ 分结点$\}$，简记为 $GAncestorDecisiveNodeSet(n11,n12)=\{n3,n7\}$。

值得注意的是，在寻找最大祖先决定点时，概念图之间的等价转换只是思维上的一种虚转换，可以充分利用有向边的权值来绕过实图转换。

定义6 已知待识别词 word 的词义集合 S，扩展概念图 $G(V,E)$，与 G 等价的单亲概念图 $G'(V',E')$，$n_y,n_z\in S\subseteq V$，n_y 相对于 n_z 的最大祖先决定点集合 GD，则 $p(n_1,\cdots,n_u,\cdots,n_v,n_t)$ 是 n_y 相对于 n_z 的决定路径，当且仅当以下三个条件之一成立：

(1) n_y 和 n_z 之间不存在祖先和后代的关系，并且，$n_1=n_y$ 或 n_1 是 n_y 的后代，$n_u=n_y,n_v\in GD,n_t$ 是 n_y 和 n_z 的公共祖先；

(2) n_y 是 n_z 的祖先，并且，$n_t=n_y$，对于 $\forall\ v\in p, v\neq n_t$，必须满足 $v\neq n_z, v$ 是 n_y 的后代，v 的分结点在 G' 中不是 n_z 的后代；

(3) n_y 是 n_z 的后代，并且，n_1 是 n_y 的后代，$n_t=n_y$。

若 $p(n_1,\cdots,n_u,\cdots,n_v,n_t)$ 是 n_y 相对于 n_z 的决定路径，则 $DPath(p,n_y,n_z)$ 为真。

那么，n_y 相对于 n_z 的决定路径集合 $DPathSet(n_y,n_z)=\{p|DPath(p,n_y,n_z)$ 为真$\}$。

如图1，Word1的词义 $n11$ 相对于 $n12$ 共有四条决定路径，分别是 $(n14,n11,n7,n2)$，$(n15,n11,n7,n2)$，$(n14,n11,n7,n5,n3,n1)$ 和 $(n15,n11,n7,n5,n3,n1)$。如果将 *Word*1的词义

改为 $n2$ 和 $n7$，则 $n2$ 相对于 $n7$ 的决定路径是（$n15$，$n12$，$n8$，$n2$）；而 $n7$ 相对于 $n2$ 的决定路径是（$n14$，$n11$，$n7$）和（$n15$，$n11$，$n7$）。

定义7 已知待识别词 word 的词义集合 S，扩展概念图 $G(V,E)$，$n_x,n_y\in S\subseteq V$，$p(n_i,n_k,\cdots,n_t)\in DPathSet(n_x,n_y)$，则上下文词语对 $p(n_i,n_k,\cdots,n_t)$ 的支持度为：

$$Support_{p(n_i,n_k,\cdots,n_t)}=\sum_{n_z\text{is a node of the path},n_z=n_t}^{\text{all }n_z} belong\_to(n_z,n_t)*freq(n_z)*\alpha_{distance(n_y,n_x)} \quad (1)$$

在式(1)中，$belong\_to(n_z,n_t)$ 是 n_z 对 n_t 的隶属度，其定义依据文[12]中对 *belong* 的定义；$freq(n_z)$ 指的是具有 n_z 词义的上下文词语的出现频率；$\alpha_{distance(n_y,n_x)}$ 是调整参数，功能是：根据决定路径上结点与被支持的词义 n_x 的层次距离来调整该结点词频对决定路径的支持度的影响。通常来说，路径上与被支持的词义 n_x 的层次距离越远的结点，它对决定路径的支持度的影响就越弱，所以 $\alpha_{distance(n_y,n_x)}$ 一般具有 $1/x^n$ 的形式，其中 $x\geqslant 1$，是一个可选参数；n 为路径结点与被支持的词义结点之间的层次距离。例如，在图1中，$n3$ 与 $n11$ 的层次距离为3，如果 x 取值为2，则 $\alpha_{distance}(n_3,n_{11})=0.125$。

3 算法

3.1 简介

本文提出的词义识别算法，通过将待识别词的词义两两分为一组，为每组词义中一方词义相对于另一方词义分别寻找其最大祖先决定点，从而找到它们的相对决定路径，即找到每组词义的相对词义差别。结合上下文词语的出现频率，根据式(1)，可以计算出上下文词语对每组词义中彼此之间的决定路径的支持度，从而计算出每组词义中各词义对待识别词的适合程度。上下文词语对每组词义中双方的支持度的反差越大，则组内词义的差别就越大，可区分度也就越大。通过对每组词义逐一进行比较，最终找到在上下文中最适合待识别词的词义。

在本算法中，对最大祖先决定点以及决定路径的搜索是自底向上，逐层展开的。虽然需要对每两个词义寻找彼此之间的相对决定路径，但算法不需要对每组词义都重复一次搜索过程。事实上，在路径目标结点的有效性数组 *Validity*[*i*]（*Validity*[*i*]的含义：就当前目标结点来说，它所支持的词义的当前搜索路径，相对于另一个词义 *i* 来说，是否已是决定路径或有可能继续被扩展成决定路径，*true* 表示"是"，*false* 表示"否"）的辅助下，算法只需对待识别词的每一种词义进行一次路径搜索，并在搜索过程中，进行相应的最大祖先决定点和决定路径的判定即可。

本算法的依赖基础是扩展概念图。根据算法思想，除了要求包含概念和概念之间的隶属关系以外，算法对概念图没有其他约束要求。但是，考虑到算法的实现技术，我们额外要求为概念图中的每个结点分配一个唯一的 id，并且 id 是按照中序遍历，即从上往下，从左往右的顺序增量编号的。此外，我们还假设每个结点都携带其相应的层次信息，并且，所有结点都有一个共同的祖先，即概念图的根结点。这三个额外的要求可以在概念图被装载到内存后，通过预备工作来满足，所以它们实际上并没有增加算法对概念图的约束要求。

3.2 具体算法

/ * 伪码内部的关键变量 * /

SenseNum：用以表示待识别词的词义数目；
CNode[]：用以存储词义结点的结点入口；
DecisvePath[]：用以存储每个词义相应的搜索路径，初始化为空；
Layer[]：用以存储每个词义结点所位于的层次信息；
Ancestor_Descendant[][]：用以存储每组词义的关系，即是否存在祖先和后代的关系，初始化为 *false*；
RelationEdge[][]：当某组词义存在祖先和后代的关系时，用以存储连接后代词义和祖先词义的路径的关键有向边，即该路径上，祖先词义与其儿子的有向边，初始化为空。

/ * 算法伪码 * /

(1)按照各词义的层次以及词义结点 id 从大到小的顺序排序 *Layer*[]和 *CNode*[]
(2)搜索各词义相对于另一个词义的部分决定路径，即从词义结点出发，经过最大祖先决定点，到达这两个词义的公共祖先的路径，伪码如下：

```
LayerInProcess = Layer[0]+1;
Finished = false; //标记搜索尚未结束
while(Finished == false){
 LayerInProcess--; //当前处理层次-1
 //逐一进行最大祖先决点判定
 for(ConceptNo = 0; ConceptNo < SenseNum; ConceptNo++)
   for(CompareNo = 0; CompareNo < SenseNum; CompareNo++)
     if((CompareNo == ConceptNo)||(Layer[CompareNo] < LayerInProcess)){
       continue;
     }else if(Layer[CompareNo] == LayerInProcess){
       //进行词义之间的祖先/后代关系判断
       for each path1 in DecisvePath[ConceptNo]
         if(path1.destinenode == CNode[CompareNo])
           {
           path1.destinenode.Validity[CompareNo] = false;
           Ancestor_Descendant[CompareNo][ConceptNo] = true;
           将 path1中指向 path1.destinenode 的边添加到
           RelationEdge[CompareNo][ConceptNo]
         }
     }else{
       //通过寻找公共祖先，找到最大祖先决定点
       for each path1 in DecisvePath[ConceptNo]
         for each path2 in DecisvePath[CompareNo]
           //找到公共祖先
           if ((path1.destinenode == path2.destinenode) &&
             (path1.destinenode.Validity[CompareNo] == true)){
             添加 path1中指向 path1.destinenode 的源结点到 GAncestorDecisiveNode(ConceptNo,CompareNo)，并标记 path1.destinenode 为 p 的目标结点之一，p∈DPathSet(ConceptNo,ComapreNo)
             path1.desnode.Validity[CompareNo] = false
           }
     }
 //扩展每一个词义结点的每一条路径
 Finished = true;
 for(ConceptNo = 0; ConceptNo < SenseNum; ConceptNo++){
   for each path1 in DecisivePath[ConceptNo]{
     如果 path1可继续扩展，则扩展 path1的目标结点到新目标结点 node2，并重新计算 path1中每个结点到 Node2的 belong 值
   }
   合并 DecisivePath[ConceptNo]中所有目标结点相同的路径
   将 CNode[j]中的所有路径按照目标结点的 id 从大到小的顺序排序
   for each path1 in DecisivePath[ConceptNo]
     for(CompareNo = 0; CompareNo < SenseNum; CompareNo++)
       if(path1.destinenode.Validity[CompareNo] == true){ //需要继续比较
         Finished = false;
         Break;
       }
     if(Finished == false) break;
 }
}
```

(3)逆向搜索各词义相对于另一个词义的剩余部分决定路径，即从词义结点的后代出发，到达词义结点的路径。

(4)将(3)搜索到的部分决定路径合并，得到完整的决定路径。

(5)根据式(1)，结合上下文（上下文词语的选取与文[8]一样，采用窗口来筛选）词频，计算每个词义相对于另一个词义所获得的支持度，两两比较后，取相对支持度最大者（即相对于其他任何词义，其支持度最大）为待识别词的词义。

4 实验数据

虽然本算法的核心在于寻找待识别词的各词义相对于另一个词义的最大祖先决定点和决定路径，但是，最终确定待识别词的词义还要依靠上下文词频对各词义的影响。待识别词的词义受上下文词语的约束越大，则本算法识别词义的准确度就越高。如图2所示，对于具有词义7，8(结点编号)的待识别词"苹果"，算法找到词义7相对于词义8的决定路径为(7，4，2，1)，上下文对该路径的支持度为1(除了"苹果"自己，没有其他上下文词语支持)；词义8相对于词义7的决定路径为(8，5，3，1)，上下文对该路径的支持度为2(除了"苹果"自己，还获得上下文词语"计算机"的支持)，所以算法很准确地识别"苹果"的词义为词义8。

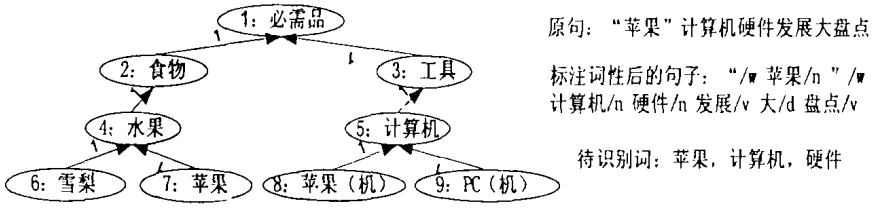

图2 例子：左图为简单的概念图，有向边上的数字为权值；右边为待识别词的出处。

为了更客观地测试和评价本算法对词义识别的效率和准确度，我们将采用公共领域词汇知识库 WordNet 1.6[10]以及已标注词义的 Brown 文集 SemCor[11]来进行测试。

WordNet 提供的词义和词义之间的 hyponyms(后代)、hypernyms(祖先)关系构成了公共领域中相当完整的一个概念图。它也正是本测试所依赖的信息基础。在本测试中，我们对于概念图中有向边的权值的规定如下：若一个词义结点只有唯一的父结点，则它与父结点之间的有向边的权值为1；否则，如果一个词义结点有 n 个父结点，则它与各父结点之间的有向边的权值分别为1/n。在测试时，由于内存限制，WordNet 概念图始终位于外存，路径搜索通过文件指针的移动，在外存中进行。因此，以下词义识别的耗时测试结果包含了文件的操作耗时。

SemCor 为我们的测试提供了有效的检验标准：从文集中选定一篇文档，通过将本算法的识别结果与 SemCor 标注的词义比较，即可检验算法对词义识别的正确性。与文[8]一样，我们随机选取了若干文档作为本测试的检验文档，并以 br-a01为代表，在表1和表2中给出相应的测试结果。br-a01共有1815个词语，其中482个是已标注词义的名词，341个是多义名词。

选定参数 $\alpha_{distance(n,sns\ node)}=1$，Depth=0，通过反复的测试，我们发现，上下文筛选窗口的大小对本算法词义识别的准确度的影响与文[8]中图7所示的准确度弧线相符。当窗口大小为15时，词义识别的准确度最高，因此，以下测试都是基于窗口大小为15而进行的。

表1

CPU	内存	本算法的词义识别耗时	以文[9]为代表的词义识别耗时
赛扬633	128 M (SDRAM)	11036 (ms)	122550(ms)

概念图在外存时，本算法和其他算法对 br-a01中已标注词义的名词的词义识别耗时：测试结果是在两种算法均不经任何预处理的前提下得出的，即在识别词义时，本算法实时寻找词义之间的相对决定路径；文[9]则实时寻找待识别词和上下文词语的共性 sim。

表2

	双词义	三词义	四词义或以上	平均值
本算法	91.49%	76.06%	48.86%	66.28%
以文[9]为代表的算法	92.55%	77.46%	53.98%	69.50%

窗口大小=15，本算法和其他算法对 br-a01中已标注词义的多义名词的词义识别准确度

表1和表2的结果表明，虽然与文[9]相比，本算法的词义识别准确度稍显逊色，但它却拥有相当高的词义识别效率，在没有任何预处理的情况下，词义识别的效率提高了10倍。

小结 在本文中，我们提出了一个基于扩展概念图的通用的词义识别算法。该算法通过寻找待识别词的词义分叉路径来找到词义之间的差异性，再通过上下文词语的支持来扩大这种差异性，从中选取获上下文支持最大的词义作为待识别词的词义，从而达到词义识别的目的。

参 考 文 献

1 Cowie J，Guthrie J，Guthrie L. Lexical Disambiguation using Simulated annealing. In：Proc. of DARPA WorkShop on Speech and Natural Language，New York，Feb. 1992. 238～242

2 Guthrie L，Guthrie J，Cowie J. Resolving Lexical Ambiguity，in Memoranda in Computer and Cognitive Sicence MCCS-93-260，Computing Research Laboratory，New Mexico State University. Las Cruces，New Mexico. 1993

3 Yarowsky D. Word-Sense Disambiguation Using Statistical Models of Roget's Categories Trained on Large Corpora. In：Proc. of the 15th Intl. Conf. on Computational Linguistics (Coling'92)，Nantes，France，1992

4 Wilks Y，Fass D，Guo C，McDonal J，Plate T，Slator B. Providing Machine Tractablle Dictionary Tools，in Semantics and the Lexicon (Pustejowsky J. ed.)，1993. 341～401

5 Sussna M. Word Sense Disambiguation for Free-text Indexing Using a Massive Semantic Network. In：Proc. of the Second Intl. Conf. on Information and knowledge Management. Arlington，Virginia USA，1993

6 Voorhees E. Using WordNet to Disambiguate Word Senses for Text Retrival. In：Proc. of the Sixteenth Annual Intl. ACM SIGIR Conf. on Research and Developement in Information Retrieval，PA，June 1993. 171～180

7 Richarson R，Smeaton A F，Murphy J. Using WordNet as a Konwledge Base for Measuring Semantic Similarity between Words，in Working Paper CA-1294，School of Computer Applications，Dublin City University. Dublin，Ireland，1994

8 Agirre E，Rigau G. A Proposal for Word Sense Disambiguation using Conceptual Distance. In：1st Intl. Conf. on Recent Advances in NLP. Bulgaria，1995

9 Resnik P. Disambiguating Noun Groupings with Respect to WordNet Senses. In：Proc. of the 3rd Workshop on Very Large Corpora，MIT，June 1995

10 WordNet，a lexical database for the English language，ftp://ftp.cogsci.princeton.edu/pub/ wordnet/wn16pc.exe

11 SemCor，semantically tagged Brown Corpus files，ftp://ftp.cogsci.princeton.edu/pub/ wordnet/semcor16.exe

12 陈宁，陈安，周龙骧，等. 基于模糊概念图的文档聚类及其在 Web 中的应用. 软件学报，2002，13(8)：1598～1605