第 5 章

CHAPTER 5

依存句法分析

句法分析的基本任务是确定句子的句法结构或者句子中词汇之间的依存关系。主要包括两方面内容：一是确定语言的语法体系，即对语言中合法句子的语法结构给予形式化的定义；另一方面是句法分析技术，即根据给定的语法体系，自动推导出句子的句法结构，分析句子所包含的句法单位和这些句法单位之间的关系。

5.1 依存句法分析原理[12]

句法分析是自然语言处理领域的一个关键问题，如能将其有效解决，一方面可对相应树库构建体系的正确性和完善性进行验证；另一方面也可直接服务于各种上层应用，比如搜索引擎用户日志分析和关键词识别、信息提取、自动问答、机器翻译等其他自然语言处理相关的任务。

依存句法是由法国语言学家 L. Tesniere 最先提出，它将句子分析成一棵依存句法树，描述出各个词语之间的依存关系。依存句法指出了词语之间在句法上的搭配关系，这种搭配关系是和语义相关联的。例如，句子“会议宣布了首批资深院士名单。”的依存句法树如图 5.1 所示。

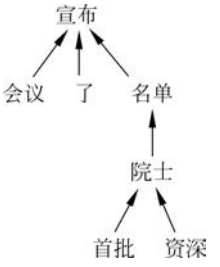

图 5.1 依存句法树（图片来源于搜狐）

从图 5.1 可以看出，词“宣布”支配“会议”“了”“名单”，故可以将这些支配词作为“宣布”的搭配词。

1. 句法分析是什么

句法分析（syntactic parsing）是自然语言处理中的关键技术之一，它是对输入的文本句子进行分析以得到句子的句法结构的处理过程。对句法结构进行分析，一方面是语言理解的自身需求，句法分析是语言理解的重要一环；另一方面也为其他自然语言处理任务提供支持。例如，句法驱动的统计机器翻译需要对源语言或目标语言（或者同时两种语言）进行句法分析。

语义分析通常以句法分析的输出结果作为输入以便获得更多的指示信息。根据句法结构的表示形式不同，最常见的句法分析任务可以分为以下 3 种：

（1）句法结构分析（syntactic structure parsing），又称短语结构分析（phrase structure parsing），也叫成分句法分析（constituent syntactic parsing）。作用是识别出句子中的短语结构以及短语之间的层次句法关系。

（2）依存关系分析，又称依存句法分析（dependency syntactic parsing），简称依存分析，作用是识别句子中词汇与词汇之间的相互依存关系。

(3) 深层文法句法分析，即利用深层文法，例如，词汇化树邻接文法(lexicalized tree adjoining grammar，LTAG)、词汇功能文法(lexical functional grammar，LFG)、组合范畴文法(combinatory categorial grammar，CCG)等，对句子进行深层的句法以及语义分析。

在自然语言处理中，用词与词之间的依存关系来描述语言结构的框架称为依存句法(dependence grammar)，又称从属关系语法。利用依存句法进行句法分析是自然语言理解的重要技术之一。

2. 重要概念

依存句法认为“谓语”中的动词是一个句子的中心，其他成分与动词直接或间接地产生联系。在依存句法理论中，“依存”指词与词之间支配与被支配的关系，这种关系不是对等的，这种关系具有方向。确切地说，处于支配地位的成分称之为支配者(governor、regent、head)，而处于被支配地位的成分称之为从属者(modifier、subordinate、dependency)。

依存句法本身没有规定要对依存关系进行分类，但为了丰富依存结构传达的句法信息，在实际应用中，一般会给依存树的边加上不同的标记。依存句法存在一个共同的基本假设：句法结构本质上包含词和词之间的依存(修饰)关系。一个依存关系连接两个词，分别是核心词(head)和依存词(dependent)。依存关系可以细分为不同的类型，表示两个词之间的具体句法关系。

3. 常见方法

(1) 基于规则的方法。早期基于依存句法的句法分析方法主要包括类似 CYK 的动态规划算法、基于约束满足的方法和确定性分析策略等。

(2) 基于统计的方法。统计自然语言处理领域也涌现出了一大批优秀的研究成果，包括生成式依存分析方法、判别式依存分析方法和确定性依存分析方法，这几类方法是数据驱动的统计依存分析中最具代表性的方法。

(3) 基于深度学习的方法。近年来，深度学习在句法分析课题上逐渐成为研究热点，主要研究工作集中在特征表示方面。传统的特征表示方法主要采用人工定义原子特征和特征组合，而深度学习则把原子特征(词、词性、类别标签)进行向量化，再利用多层神经元网络提取特征。

通常使用的指标包括无标记依存正确率(unlabeled attachment score，UAS)、带标记依存正确率(labeled attachment score，LAS)、依存正确率(dependency accuracy，DA)、根正确率(root accuracy，RA)、完全匹配率(complete match，CM)等。这些指标的具体意思如下：

(1) 无标记依存正确率(UAS)——测试集中找到其正确支配词的词(包括没有标注支配词的根节点)所占总词数的百分比。

(2) 带标记依存正确率(LAS)——测试集中找到其正确支配词的词，并且依存关系类型也标注正确的词(包括没有标注支配词的根节点)占总词数的百分比。

(3) 依存正确率(DA)——测试集中找到正确支配词非根节点词占所有非根节点词总数的百分比。

(4) 根正确率(RA)——有两种定义：一种是测试集中正确根节点的个数与句子个数的百分比；另一种是指测试集中找到正确根节点的句子数所占句子总数的百分比。

(5) 完全匹配率(CM)——测试集中无标记依存结构完全正确的句子占句子总数的百分比。

4. 数据集

Penn Treebank 是一个项目的名称，该项目的目的是对语料进行标注，标注内容包括词性

标注以及句法分析。

CoNLL 经常开放句法分析的学术评测。

5. 相关开源工具介绍

下面介绍几个相关开源工具。

1) StanfordCoreNLP

该工具由斯坦福大学开发,提供依存句法分析功能。

Github 地址: https://github.com/Lynten/stanford-corenlp。

官网: https://stanfordnlp.github.io/CoreNLP/。

安装: pip install stanfordcorenlp。

国内源安装: pip install stanfordcorenlp -i https://pypi.tuna.tsinghua.edu.cn/simple。

使用 StanfordCoreNLP 进行依存句法分析:

先下载模型,下载地址: https://nlp.stanford.edu/software/corenlp-backup-download.html。

再使用例子进行依存句法分析,代码如下:

```
from stanfordcorenlp import StanfordCoreNLP
# 对中文进行依存句法分析
zh_model = StanfordCoreNLP(r'stanford-corenlp-full-2018-02-27', lang= 'zh')
s_zh = '我爱自然语言处理技术!'
dep_zh = zh_model.dependency_parse(s_zh)
print(dep_zh)
[('ROOT', 0, 4), ('nsubj', 4, 1), ('advmod', 4, 2), ('nsubj', 4, 3), ('dobj', 4, 5), ('punct', 4, 6)]
# 对英文进行依存句法分析
eng_model = StanfordCoreNLP( r'stanford-corenlp-full-2018-02-27')
s_eng = 'I love natural language processing technology!'
dep_eng = eng_model.dependency_parse(s_eng)
print(dep_eng)
[('ROOT', 0, 2), ('nsubj', 2, 1), ('amod', 6, 3), ('compound', 6, 4), ('compound', 6, 5),
('dobj', 2, 6), ('punct', 2, 7)]
```

2) HanLP

HanLP 是一系列模型与算法组成的自然语言处理工具包,它提供了中文依存句法分析功能。

Github 地址: https://github.com/hankcs/pyhanlp。

官网: http://hanlp.linrunsoft.com/。

安装: pip install pyhanlp。

国内源安装: pip install pyhanlp -i https://pypi.tuna.tsinghua.edu.cn/simple。

使用该工具对示例进行依存句法分析的代码如下:

```
from pyhanlp import *
s_zh = '我爱自然语言处理技术!'
dep_zh = HanLP.parseDependency(s_zh)
print(dep_zh)
1 我 我 r r _ 2 主谓关系 _ _
2 爱 爱 v v _ 0 核心关系 _ _
3 自然语言处理 自然语言处理 v v _ 4 定中关系 _ _
4 技术 技术 n n _ 2 动宾关系 _ _
5 ! ! wp w _ 2 标点符号 _ _
```

3) SpaCy

该工具为工业级的自然语言处理工具,遗憾的是目前不支持中文。

4）FudanNLP

该工具为复旦大学自然语言处理实验室开发的中文自然语言处理工具包，包含信息检索（文本分类、新闻聚类）、中文处理（中文分词、词性标注、实体名识别、关键词提取、依存句法分析、时间短语识别）、结构化学习（在线学习、层次分类、聚类）。

5.2　HanLP 基于神经网络依存句法分析器[13]

HanLP 本身提供了基于神经网络的依存句法分析器，除此之外，还提供了很多句法分析器，其结构如图 5.2 所示。

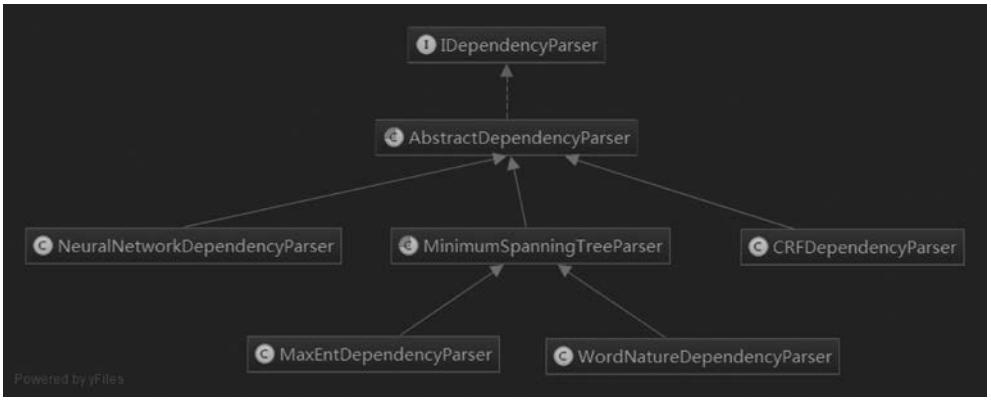

图 5.2　句法分析器接口（图片来源于 CSDN）

句法分析器接口代码如下所示：

```
/**
 * 依存句法分析器接口
 *
 * @author hankcs
 */
public interface IDependencyParser
{
    /**
     * 分析句子的依存句法
     *
     * @param termList 句子，可以是任何具有词性标注功能的分词器的分词结果
     * @return CoNLL 格式的依存句法树
     */
    CoNLLSentence parse(List<Term> termList);

    /**
     * 分析句子的依存句法
     *
     * @param sentence 句子
     * @return CoNLL 格式的依存句法树
     */
    CoNLLSentence parse(String sentence);

    /**
     * 获取 Parser 使用的分词器
     *
     * @return
     */
```

```
    Segment getSegment();

    /**
     * 设置 Parser 使用的分词器
     *
     * @param segment
     */
    IDependencyParser setSegment(Segment segment);

    /**
     * 获取依存关系映射表
     *
     * @return
     */
    Map<String, String> getDeprelTranslator();

    /**
     * 设置依存关系映射表
     *
     * @param deprelTranslator
     */
    IDependencyParser setDeprelTranslator(Map<String, String> deprelTranslator);

    /**
     * 依存关系自动转换开关
     * @param enable
     */
    IDependencyParser enableDeprelTranslator(boolean enable);
}
```

简单示例代码如下：

```
/**
 * 依存句法分析(神经网络句法模型需要 -Xms1g -Xmx1g -Xmn512m)
 * @author hankcs
 */
public class DemoDependencyParser
{
    public static void main(String[] args)
    {
        CoNLLSentence sentence = HanLP.parseDependency("徐先生还具体帮助他确定了把画雄鹰、
松鼠和麻雀作为主攻目标。");
        System.out.println(sentence);
        // 可以方便地遍历它
        for (CoNLLWord word : sentence)
        {
            System.out.printf("%s --(%s)-->%s\n", word.LEMMA, word.DEPREL, word.HEAD
.LEMMA);
        }
        // 也可以直接拿到数组,任意顺序或逆序遍历
        CoNLLWord[] wordArray = sentence.getWordArray();
        for (int i = wordArray.length - 1; i >= 0; i--)
        {
            CoNLLWord word = wordArray[i];
            System.out.printf("%s --(%s)-->%s\n", word.LEMMA, word.DEPREL, word.HEAD
.LEMMA);
        }
        // 还可以直接遍历子树,从某棵子树的某个节点一路遍历到虚根
        CoNLLWord head = wordArray[12];
        while ((head = head.HEAD) != null)
```

```
        {
            if (head == CoNLLWord.ROOT) System.out.println(head.LEMMA);
            else System.out.printf("%s --(%s)--> ", head.LEMMA, head.DEPREL);
        }
    }
}
```

运行结果如下：

```
1.  1 徐先生  徐先生  nh  nr  _  4  主谓关系 _  _
2.  2 还  还  d  d  _  4  状中结构  _  _
3.  3 具体  具体 a  a  _  4  状中结构  _  _
4.  4 帮助  帮助 v  v  _  0  核心关系  _  _
5.  5 他  他  r  rr  _  4  兼语  _  _
6.  6 确定  确定 v  v  _  4  动宾关系  _  _
7.  7 了  了  u  ule  _  6  右附加关系  _  _
8.  8 把  把  p  pba  _  15  状中结构  _  _
9.  9 画  画  v  v  _  8  介宾关系  _  _
10. 10  雄鹰 雄鹰 n  n  _  9  动宾关系  _  _
11. 11  、  、  wp  w  _  12  标点符号  _  _
12. 12  松鼠 松鼠 n  n  _  10  并列关系  _  _
13. 13  和  和  c  cc  _  14  左附加关系  _  _
14. 14  麻雀 麻雀 n  n  _  10  并列关系  _  _
15. 15  作为 作为 p  p  _  6  动宾关系  _  _
16. 16  主攻 主攻 v  v  _  17  定中关系  _  _
17. 17  目标 目标 n  n  _  15  动宾关系  _  _
18. 18  。  。  wp  w  _  4  标点符号  _  _
19.
20. 徐先生 --(主谓关系)--> 帮助
21. 还 --(状中结构)--> 帮助
22. 具体 --(状中结构)--> 帮助
23. 帮助 --(核心关系)--> ##核心##
24. 他 --(兼语)--> 帮助
25. 确定 --(动宾关系)--> 帮助
26. 了 --(右附加关系)--> 确定
27. 把 --(状中结构)--> 作为
28. 画 --(介宾关系)--> 把
29. 雄鹰 --(动宾关系)--> 画
30. 、--(标点符号)--> 松鼠
31. 松鼠 --(并列关系)--> 雄鹰
32. 和 --(左附加关系)--> 麻雀
33. 麻雀 --(并列关系)--> 雄鹰
34. 作为 --(动宾关系)--> 确定
35. 主攻 --(定中关系)--> 目标
36. 目标 --(动宾关系)--> 作为
37. 。--(标点符号)--> 帮助
38.
39. 。--(标点符号)--> 帮助
40. 目标 --(动宾关系)--> 作为
41. 主攻 --(定中关系)--> 目标
42. 作为 --(动宾关系)--> 确定
43. 麻雀 --(并列关系)--> 雄鹰
44. 和 --(左附加关系)--> 麻雀
45. 松鼠 --(并列关系)--> 雄鹰
46. 、--(标点符号)--> 松鼠
47. 雄鹰 --(动宾关系)--> 画
48. 画 --(介宾关系)--> 把
49. 把 --(状中结构)--> 作为
50. 了 --(右附加关系)--> 确定
51. 确定 --(动宾关系)--> 帮助
```

```
52.     他 --(兼语)--> 帮助
53.     帮助 --(核心关系)--> ##核心##
54.     具体 --(状中结构)--> 帮助
55.     还 --(状中结构)--> 帮助
56.     徐先生 --(主谓关系)--> 帮助
57.
58.     麻雀 --(并列关系)--> 雄鹰 --(动宾关系)--> 画 --(介宾关系)--> 把 --(状中结构)
    --> 作为 --(动宾关系)--> 确定
```

各类关系解释如下：

(1) 主谓关系(subject-verb,SBV)：我送她一束花（我 <-- 送)。

(2) 动宾关系(verb-object,VOB)：我送她一束花（送 --> 花)。

(3) 间宾关系(indirect-object,IOB)：我送她一束花（送 --> 她)。

(4) 前置宾语(fronting-object,FOB)：他什么书都读（书 <-- 读)。

(5) 兼语(double,DBL)：他请我吃饭（请 --> 我)。

(6) 定中关系(attribute,ATT)：红苹果（红 <-- 苹果)。

(7) 状中结构(adverbial,ADV)：非常美丽（非常 <-- 美丽)。

(8) 动补结构(complement,CMP)：做完了作业（做 --> 完)。

(9) 并列关系(coordinate,COO)：大山和大海（大山 --> 大海)。

(10) 介宾关系(preposition-object,POB)：在贸易区内（在 --> 内)。

(11) 左附加关系(left adjunct,LAD)：大山和大海（和 <-- 大海)。

(12) 右附加关系(right adjunct,RAD)：孩子们（孩子 --> 们)。

(13) 独立结构(independent structure,IS)：两个单句在结构上彼此独立。

(14) 核心关系(head,HED)：指整个句子的核心。

可以使用 DependencyViewer 进行可视化，如图 5.3 所示。

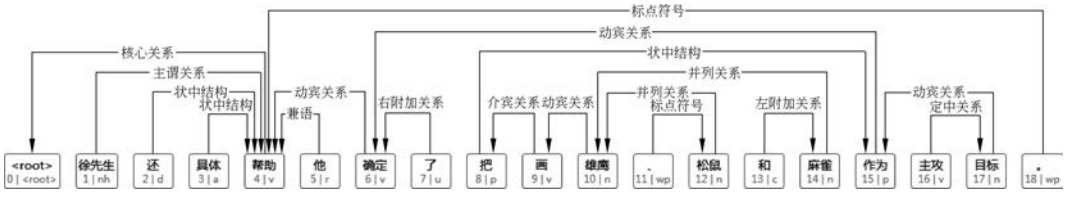

图 5.3　依存句法关系(图片来源于 CSDN)