Computer Science and Application 计算机科学与应用, 2021, 11(5), 1538-1547
Published Online May 2021 in Hans. http://www.hanspub.org/journal/csa
https://doi.org/10.12677/csa.2021.115158

Hans汉斯

基于FP-Tree算法的汉语复句关系词依存关系规则的自动挖掘

涂馨丹

武汉设计工程学院，湖北 武汉
Email: 594160587@qq.com

收稿日期：2021年4月25日；录用日期：2021年5月20日；发布日期：2021年5月27日

摘 要

目前关系词识别规则库中共有规则734条，主要是基于字面特征的规则，仍需补充基于依存关系的规则。本文在依存语法的基础上，运用挖掘频繁项集的FP-tree算法对复句中依存规则进行自动挖掘。首先对语料进行预处理，为避免每次重复扫描数据库，先根据关系词对复句进行分类；同时排除数据集过小的分类结果，以保证挖掘规则的质量；然后利用特征分析器分析预处理后的语料，并对分析结果进行形式化表示得到复句的依存特征集合；接着用FP-tree算法对实验语料进行规则挖掘，共挖掘规则84条。实验结果表明，FP-tree算法对依存规则进行自动挖掘的可行性和有效性。

关键词

关系词，依存关系，规则挖掘，FP-Tree

Automatic Mining of the Dependency Relation Rule of Relational Word in Chinese Compound Sentences Based on FP-Tree Algorithm

Xindan Tu

Wuhan Institute of Design and Sciences, Wuhan Hubei
Email: 594160587@qq.com

Received: Apr. 25th, 2021; accepted: May 20th, 2021; published: May 27th, 2021

文章引用: 涂馨丹. 基于 FP-Tree 算法的汉语复句关系词依存关系规则的自动挖掘[J]. 计算机科学与应用, 2021, 11(5): 1538-1547. DOI: 10.12677/csa.2021.115158

Abstract

The relation word recognition rule base has 734 rules, which are mainly based on the characteristics of literal, and the rules based on dependencies still need to supplement. On the basis of dependency syntax, this paper uses the FP-tree algorithm of mining frequent item sets to automatically mine the dependency rules in complex sentences. First of all, the language material is pre-processed, in order to avoid each repeated scan of the database, first according to the relationship word to classify the complex sentences, at the same time, the small classification results of data sets are excluded to ensure the quality of mining rules, then, the pre-processed language material is analyzed by the feature analyzer, and the analysis results are formalized to represent the set of dependent features of the complex sentence, then, mining the experimental material by FP-tree algorithm, and a total of 84 rules are mined. The experimental results show that this algorithm is feasible and effective in automatic mining dependency rule.

Keywords

Relational Words, Dependency Relation, Rule Mining, FP-Tree

1. 引言

汉语复句关系词的识别对汉语复句语义的识别至关重要，关系词在句中出现形式灵活、词性多变及其搭配多样，有些词在某类复句中充当关系词，在另一类复句中又不是关系词，给关系词的自动识别带来了一定的难度[1] [2]。

目前对关系词进行自动识别的主要方法是使用基于规则的方法，用是否满足规则中的约束条件来识别关系词。已知的字面特征为 12 种，另外，根据依存树库中关系词在复句中的依存关系提取出 7 条约束条件，在关系词字面特征的约束条件基础上加入关系词依存约束条件，共 19 种约束条件用于识别汉语复句关系词[3]。

关联规则挖掘包括单层关联规则挖掘和多层关联规则挖掘，单层规则挖掘算法包括 Apriori、AprioriTid、AIS、SETM 以及在 Apriori 算法基础上进行改进的 FP-tree 算法[4]，就这些算法的运行速度来说，对于大的数据集，SETM 运行时间要比 Apriori 慢一个数量级；对于不大的数据集，Apriori 的性能要比 AIS 好；对于小的数据集，AprioriTid 和 Apriori 的性能差不多；但对于大的数据集，AprioriTid 要比 Apriori 慢 2 倍左右；无论数据集的大小如何，相对于需要多次扫描数据集的 Apriori 来说，只需要扫描两次数据集的 FP-tree 算法性能更好[5]。

本文在依存树库与现有规则库的基础上，使用 FP-tree 算法对复句关系词的依存关系规则进行自动挖掘，旨在挖掘更多潜在规则。

2. 研究现状

目前用于识别汉语复句关系词的约束条件共 19 种，其中基于字面特征的约束条件 12 种，基于依存关系的约束条件 7 种，编号从 13~19。本文使用 FP-tree 算法对依存关系规则进行自动挖掘，对这 7 个约束条件在挖掘语料库中的表示形式规定如表 1 所示：

Table 1. Representation of dependency constraints
表 1. 依存关系约束条件表示形式

Id	type	dpConstrains	remark
13	支配词词性	parentPos(A)	关系词 A 的支配词词性
14	关系词与支配词之间的依存关系	Relation(A)	关系词 A 与支配它的词二者之间的依存关系
15	关系词对的依存方向	relationDirection(A,B)	关系词 A、B 的依存方向，与关系词库依存方向一致为 1，否则为 0
16	是否支配其他词	isParent(A)	关系词 A 是否支配其他词，支配其他词为 1，否则为 0
17	支配词是否相同	ispEqual(A,B)	关系词 A、B 的支配词是否相同，相同为 1，不相同为 0
18	关系词的层次距离	Distance(A,B)	A、B 两个关系词的层次距离
19	关系词的支配词之间的关系	Gwdp(A,B)	A、B 两个关系词的支配词之间的依存关系

表 1 中 expression 一列即是依存关系在挖掘语料中的表示形式，在 expression 中关系词的依存关系类型和依存关系之间采用：type-content，用 i (依次取1,2,3,⋯)表示复句中的第 i 个关系词，不同约束类型之间用“，”隔开。

以例句(1)为例说明 expression 中的表示形式：

例句(1)高国胜想，既然能赚钱又不麻烦，就答应下来。《人民日报》1995 年 06 月 02 日 09 版次

其依存分析如图 1 所示：

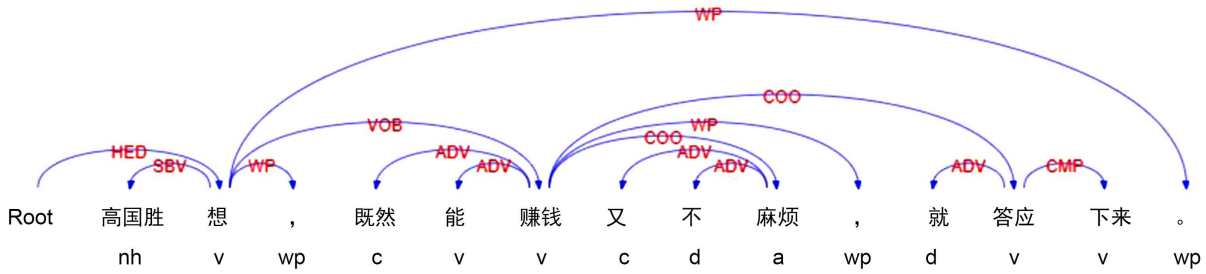

Figure 1. Dependency syntax analysis diagram of example 1
图 1. 例句 1 的依存句法分析图

根据表 1 中 expression 中规定的表达形式，其依存关系可以表示为：131-v，132-a，141-ADV，142-ADV，15-1，161-0，162-0，17-0，18-1，19-COO。

FP-tree 算法是在经典规则挖掘算法 Apriori 的基础上改进得到的[6]，Apriori 算法最大的缺点是每计算一次候选项集就要扫描一次事物数据库，FP-tree 则只需要扫描两次事物数据库，在不生成候选项的情况下，把事务数据库压缩到一棵只存储频繁项的树结构中[7]，从而实现 Apriori 算法的功能。

3. FP-Tree 算法自动挖掘事务数据库中规则

本文的研究基于 CCCS (CCCS 语料库由华中师范大学语言研究所创建，包含 65 万余条汉语复句语料)汉语复句语料库，利用哈尔滨工业大学的 LTP 平台对语料库中的复句进行依存句法分析，构建依存树库，提取依存树中关系词的依存关系，并形式化表示为 7 种约束条件，在此约束条件基础上本文利用 FP-tree 算法挖掘语料库中频繁出现的依存关系规则。

下面用一个简单的例子阐明算法中涉及到的概念：

设事务数据库如表 2 所示：

Table 2. Transaction database
表 2. 事务数据库

a e f g
a f g
a b e f g
e f g
e b f h

项目：如表 2 中的“a”、“e”、“f”等表示一个项目；
事务：表 2 中每行表示一个事务，事务由若干个互不相同的项目构成，表中共五个事务；
模式：任意几个项目的组合称为一个模式，用{}包含一个模式中的项目；
支持数：指事务数据库中某个模式的个数，表 2 中模式{a, f, g}的支持数为 3；
支持度：模式的支持数占数据库中事务的比例，计算公式如下：

$$\text{support} = \frac{\text{NUM}(\text{sameConstraints})}{\|D\|} \quad \text{(公式 2.1)}$$

$\|D\|$ 表示事务数据库中所有模式的和，也即是挖掘语料中复句的总和，NUM(sameConstrains)表示模式的支持数[8]，也即是挖掘语料中包含某种频繁模式的复句总和，如模式{a, f, g}的支持度为 3/5；

最小支持度(minSupport)：提前设置好的，根据各项目在事务数据库中出现的次数设置最佳的 minSupport，本文设置最小支持度为 $\frac{\|D\|}{2}+1$，$\|D\|$ 表示挖掘语料中复句的总和；

频繁模式：支持数大于阈值 minSupport 的模式称为频繁模式；

置信度：计算公式如下为

$$\text{Confidence} = \frac{\text{NUM}(\text{sameConstraintsAndsameResult})}{\text{NUM}(\text{sameConstraints})} \times 100\% \quad \text{(公式 2.2)}$$

在上述公式中若 NUM(sameConstrains)表示某模式 α 的支持数，即挖掘语料中包含某种频繁模式的复句总和，那么 NUM(sameConstraintsAndsameResult)就表示包含模式 α 的 β 模式的支持数，即包含上述频繁模式的另一频繁模式的复句总和。例如模式{f, g}的支持数为 4，支持度为 4/5，{a}的支持数为 3，支持度为 3/5，那么{f, g} => {a}的置信度为，{a, f, g}的支持数除以{f, g}的支持数，结果为 3/4，本文设置置信度为 $\frac{1}{2}+0.01$，即 0.51。

FP-tree：首先建立一棵以 NULL 为根节点的树，然后统计事务数据库中各项的支持度，并将其按照降序排列，再依次插入到这棵树中，同时将每个项的支持度和树中相应的结点链接起来[9]，即可得到一棵 FP-tree。

条件模式基(CPB)：是指在一棵 FP-tree 树中出现在某一频繁项(结点)之前的所有祖先结点的集合[10]。比如 a 在事务中一共出现了 3 次，那么 a 的条件模式基就是其祖先路径{f，g：3(频度为 3)}。

条件树(PostModel)：根据上述 FP-Tree 的形成步骤，把条件模式基构造成的一棵新的 FP-tree，初始值为空。

FP-tree 算法的四个关键步骤如下：

1) 构造项头表：扫描事务数据库，在表中记录频繁项集合 F，将 F 根据支持度递减在表中排序[11]，

得到项头表 L。

2) 构造原始 FP-tree：根据频繁项在项头表中的排序，重新排序事务数据库中每条事物的频繁项。并构造以 NULL 为根节点的 FP-tree，将重排后的事务数据库中的每个事物的每个频繁项插入树中[12]。插入频繁项时应遵循两个原则：① 若该频繁项节点在 FP-tree 中已存在，则将其节点的支持度加 1；② 若该节点不存在，则创建支持度为 1 的节点，并把该节点链接到项头表中[13]。

3) 调用 FP-growth(Tree, null)进行挖掘。伪代码如下：

```
Procedure FP_growth(Tree,PostModel)
  If Tree 含单个路径 P then{
      For 路径 P 中结点的每个组合(记做 b)产生模式 bUPostModel，其支持度
      support=b 中结点的最小支持度；
}
else{
  for each 在 Tree 的头部(按照支持度由低到高顺序进行扫描){
      产生一个模式 b=UPostModel，其支持度 support=.support;
      构造 b 的条件模式基，然后构造 b 的条件数 tree-b;
      If tree-b 不为空 then
          调用 FP_growth(tree-b,b);
}
}
```

4) 递归调用 FP-growth(新的 CPB，新的 PostModel)，直到发现新的 CPB 为空时退出。

4. 数据预处理

本文在依存树库与现有规则库的基础上，使用 FP-tree 算法对关系词的依存规则进行自动挖掘，规则挖掘的内容包括两部分:一是挖掘规则库中不存在的关系词搭配；二是挖掘规则库中已有的关系词搭配的其他潜在规则，根据这两个挖掘需求对依存树库进行数据预处理，得到挖掘规则所需要的原始数据库。

4.1. 规则库中不存在的关系词搭配

首先介绍数据预处理用到的两个表的结构，分别是用于存储挖掘的新规则的表 mining 和用来存储规则库中不存在的关系词搭配的原始数据库 result-1，result-1 的结构和图 2 中依存树库的结构相同，表 mining 的结构如表 3。

id	sentence	xml	json	source	senmatch
1	他们是忠厚的，甚至不愿踩死一只小蚂蚁，然而他	<?xml version="	[[[{ "id	《长江日报》1983年06月11	然而
2	10多天来，张群的父母一面忙着护理住院的女儿，	<?xml version="	[[[{ "id	《长江日报》1989年05月17	一面，一面
3	10分钟时，武钢队左前锋12号孙庆率先发难，他[	<?xml version="	[[[{ "id	《长江日报》1988年10月25	虽然，然而；虽然；然而
4	10年的经验告诉我们，能够给全国各族人民带来巨	<?xml version="	[[[{ "id	《长江日报》1988年12月22	既，也
5	10年的实践已经可以使我们得出这个认识，因而[	<?xml version="	[[[{ "id	《长江日报》1989年11月02	因而
6	10年的实践证明，这一方针是我们的立国之本和[	<?xml version="	[[[{ "id	《长江日报》1989年07月17	不仅，也；不仅，也

Figure 2. Dependency tree library
图 2. 依存树库

Table 3. Structure of table mining
表 3. 表 mining 的结构

ID	Keymarks	dpType	dpconstrains	result	remarks

表 3 中 ID 表示新的关系词搭配在表中的序号，根据写入表中的先后顺序对 ID 进行编号，从 1 开始，Keymarks 表示新的关系词搭配，和原来的规则库中表示方法相同，dpType 和 dpconstrains 分别表示挖掘出的新规则的依存约束类型和依存约束条件的形式化表示，result 是关系词的判别结果，remarks 里存储备注内容。

依存树库中关系词搭配的表示形式形如“一面，一面”，关系词搭配之间用“，”间隔，关系词搭配在规则库中的表示形式如“一面/一面”，关系词搭配之间用“/”隔开，对其中一种表示形式做适当变化，针对依存树库中的每一条数据都进行 senmatch 一列与规则库中关系词搭配一列进行比较，若 senmatch 中的关系词搭配在规则库中找不到相同的关系词搭配，则把新的关系词搭配存入表 mining 中 Keymarks 一列，并把依存树库中对应的一条数据存入到 result-1 中，得到的数据库及其格式如图 3 所示：

id	sentence	xml	json	source	senmatch
21	11—12月，每百公	<?xml version="1.	[[[{ "id": 0,	《长江日报》1989年01月	虽然，但因
94	1982年9月被上海(	<?xml version="1.	[[[{ "id": 0,	《长江日报》1987年01月	虽然，但因
766	“既然快餐食品即	<?xml version="1.	[[[{ "id": 0,	《长江日报》1985年04月	既然，又
774	“既然日本对华侵	<?xml version="1.	[[[{ "id": 0,	《长江日报》1982年08月	既然，又
775	“既然如此，那金	<?xml version="1.	[[[{ "id": 0,	《长江日报》1985年09月	既然，又
778	“既然如此，你们	<?xml version="1.	[[[{ "id": 0,	《长江日报》1982年03月	既然，又
1777	“我入党时既然宣	<?xml version="1.	[[[{ "id": 0,	《长江日报》1986年12月	既然，又

Figure 3. Database and its format
图 3. 数据库及其格式

接着根据表 mining 中 Keymarks 一列中的关系词搭配对 Result-1 中的原始数据进行分类，得到各个关系词搭配的原始依存树库，如图 4 是新的关系词搭配“虽然/但因”的原始依存树库：

id	sentence	xml	json	source	senmatch
21	11—12月，每百	<?xml versic	[[[{ "id"	《长江日报》1989	虽然，但因
94	1982年9月被上海	<?xml versic	[[[{ "id"	《长江日报》1987	虽然，但因
6364	从而使许多同一地	<?xml versic	[[[{ "id"	《长江日报》1987	虽然，但因
9438	对手虽然相对较弱	<?xml versic	[[[{ "id"	《长江日报》1983	虽然，但因
9555	对有些虽然未到离	<?xml versic	[[[{ "id"	《长江日报》1982	虽然，但因
10903	该市凌岳商场今年	<?xml versic	[[[{ "id"	《长江日报》1986	虽然，但因

Figure 4. The original dependency tree library of “although/but because”
图 4. “虽然/但因”的原始依存树库

得到每个关系词搭配的原始依存树库之后，通过运用复句特征分析器对复句中的依存关系进行分析，主要分析表 1 中的 7 个依存约束类型，规定分析之后的表示形式如表 1 中 expression 中所述，同时将 senmatch 列替换为规则库中关系词的判别结果 result 的表达形式，并将各个关系词的判别结果用“^”符号连接起来，则得到新的依存树库，依存树库的格式如表 4 所示：

Table 4. New dependency tree format
表 4. 新的依存树库的格式

id	sentence	xml	json	expression	source	result

表 4 中 expression 是经过依存分析得到的依存约束条件的形式化表示，以关系词搭配“既然/又”为例，如图 5 即是得到的新的依存树库：

id	sentence	xml	json	expression	source	result
766	“既然快餐食品[	<?xml versi	[[{ "id'	131-a，132-a，1	《长江日报》	R(既然)=true^R(又)=ture
774	“既然日本对华侵	<?xml versi	[[{ "id'	131-v，132-v，1	《长江日报》	R(既然)=true^R(又)=ture
775	“既然如此，那金	<?xml versi	[[{ "id'	131-v，132-v，1	《长江日报》	R(既然)=true^R(又)=ture
777	“既然如此，那又	<?xml versi	[[{ "id'	131-v，132-a，1	《长江日报》	R(既然)=true^R(又)=ture
778	“既然如此，你们	<?xml versi	[[{ "id'	131-v，132-v，1	《长江日报》	R(既然)=true^R(又)=ture
1777	“我入党时既然宣	<?xml versi	[[{ "id'	131-v，132-v，1	《长江日报》	R(既然)=true^R(又)=ture

Figure 5. New dependency tree library
图 5. 新的依存树库

4.2. 规则库中已有的关系词搭配

挖掘规则库中已有的关系词搭配的潜在规则，是在之前的规则库对复句关系词的识别基础上，挖掘那些规则库不能识别的复句语料，把不能识别的复句按照关系词搭配分类，得到规则库中规则不能识别的已有关系词搭配的原始语料库，如图 6 所示是规则库中关于“不仅/还”关系词搭配所不能识别的原始依存树库：

id	sentence	xml	json	source	senmatch
435	“不仅要说服自己	<?xml version="1	[[{ "id":	《长江日报》198	不仅，还
523	“当前，我国经济、	<?xml version="1	[[{ "id":	《长江日报》198	不仅，还
558	“电热褥”不仅能弱	<?xml version="1	[[{ "id":	《长江日报》198	不仅，还
898	“考试”结束后，-	<?xml version="1	[[{ "id":	《长江日报》198	不仅，还
1540	“台步”，不仅是[	<?xml version="1	[[{ "id":	《长江日报》198	不仅，还

Figure 6. Original dependency tree library of relational word collocation “not only/also”
图 6. 关系词搭配“不仅/还”的原始依存树库

对图 6 中所得到的原始依存树库运用特征分析器分析其依存关系，并存入表 4 所示的 expression 列中，得到实验所需的挖掘语料，如图 7 即是规则库中已有关系词搭配“不仅/还”的挖掘语料：

id	sentence	xml	json	expression	source	result
435	“不仅要说服自己	<?xml versic	[[{ "id	131-v，132-v	《长江日报》198	R(不仅)=true^R(还)=ture
523	“当前，我国经济、	<?xml versic	[[{ "id	131-v，132-v	《长江日报》198	R(不仅)=true^R(还)=ture
558	“电热褥”不仅能弱	<?xml versic	[[{ "id	131-v，132-v	《长江日报》198	R(不仅)=true^R(还)=ture
898	“考试”结束后，-	<?xml versic	[[{ "id	131-v，132-v	《长江日报》198	R(不仅)=true^R(还)=ture
1540	“台步”，不仅是[	<?xml versic	[[{ "id	131-v，132-v	《长江日报》198	R(不仅)=true^R(还)=ture

Figure 7. Mining corpus for the collocation of relative words “not only/also”
图 7. 关系词搭配“不仅/还”的挖掘语料

5. 基于 FP-tree 算法自动挖掘语料库中关系词依存关系规则

在本文的第 2 节中已经介绍了 FP-tree 算法如何自动挖掘规则，FP-tree 算法能够挖掘出数据中的频繁模式，本节利用 FP-tree 算法挖掘语料库中频繁出现的规则。

如图 5 和图 7 是 FP-tree 算法进行挖掘所需的熟语料，主要针对熟语料库中的两列进行规则自动挖掘，分别是 expression 和 result，expression 列是关系词搭配的依存约束条件，result 列是关系词搭配的判别结果，利用 FP-tree 算法挖掘出关系词搭配的依存约束条件与判别结果之间的频繁模式，算法的输入是经过特征分析器分析后的语料库，如针对关系词搭配“既然/又”进行规则挖掘的语料库格式如图 8 所示：

expression	result
131-a , 132-a , 141-ADV , 142-ADV , 15-1 , 161-0 , 162-0 , 17-1 , 18-0 , 19-1	R(既然)=true^R(又)=ture
131-v , 132-v , 141-ADV , 142-ADV , 15-1 , 161-0 , 162-0 , 17-0 , 18-1 , 19-COO	R(既然)=true^R(又)=ture
131-v , 132-v , 141-ADV , 142-ADV , 15-1 , 161-1 , 162-0 , 17-0 , 18-1 , 19-COO	R(既然)=true^R(又)=ture
131-v , 132-a , 141-ADV , 142-ADV , 15-1 , 161-0 , 162-0 , 17-0 , 18-3 , 19-0	R(既然)=true^R(又)=ture
131-v , 132-v , 141-ADV , 142-ADV , 15-1 , 161-0 , 162-0 , 17-0 , 18-1 , 19-COO	R(既然)=true^R(又)=ture
131-v , 132-v , 141-ADV , 142-ADV , 15-1 , 161-0 , 162-0 , 17-0 , 18-1 , 19-COO	R(既然)=true^R(又)=ture
131-v , 132-v , 141-ADV , 142-ADV , 15-1 , 161-0 , 162-0 , 17-1 , 18-0 , 19-1	R(既然)=true^R(又)=ture

Figure 8. Corpus for rule mining with relational words collocation “since/again”
图 8. 关系词搭配“既然/又”进行规则挖掘的语料库

图 8 中的每一条记录就是一条事务，读取事务时先读取 expression 列中的项然后读取 result 列中的项，两列之间的内容用“，”隔开，所有事务的集合即是要处理的数据集合，语料库中关系词搭配是“既然/又”的复句共 522 条，所以供挖掘的数据集合中包含 522 条数据，即 $\|D\|$ = 522，为保证挖掘规则的实用性，本文对最小支持度和置信度的设置采取过半原则，即只挖掘超过挖掘数据集一半的规则，因此本次挖掘设置最小支持度 minSupport = $\frac{\|D\|}{2}+1$，即 minSupport = 262，置信度 Confident = 0.51。

算法描述如下：

输入：语料库中的数据集合 List<List<String>> transactions；最小支持度 minSupport；置信度 Confident

输出：依存规则与关系词判别结果的频繁模式集合 Map<List<String>,Integer>；频繁模式的频数 FrequentPattens

初始化 PostModel=[]，CPB=transactions

voidFPGrowth(List<List<String>>CPB,List<String>PostModel){

If CPB 为空{

return；

}

else{

统计 CPB 中每一个项目的计数，把计数小于最小支持数 minSupport 的删除掉，对于 CPB 中的每一条事务按项目计数降序排列。

由 CPB 构建 FP-Tree，FP-Tree 中包含了表头项 headers，每一个 header 都指向了一链表 HeaderLinkList，链表中的每个元素都是 FP-Tree 上的一个节点，且节点名称 header.name 相同。

}

For header in headers{

newPostModel=header.name+PostModel

把<newPostModel, header.count>加到 FrequentPattens 中。

newCPB=[]

Or TreeNode in HeaderLinkList;

得到从 FP-Tree 的根节点到 TreeNode 的全路径 path，把 path 作为一个事务添加到 newCPB 中，要重复添加 TreeNode.count 次。

}

FPGrowth(newCPB,newPostModel);

}

根据上述 FP-tree 算法对“既然/又”的关系词搭配进行自动挖掘的结果如图 9 所示：

```
buildFPTree use time 9
模式  频数
[R(既然)=true^R(又)=true，131-v,132-v,141-ADV,142-ADV,15-1,161-0,162-0,17-0,18-1,19-COO]   407
[R(既然)=true^R(又)=true]       522
[131-v,132-v,141-ADV,142-ADV,15-1,161-0,162-0,17-0,18-1,19-COO]  408

条件 结果  支持度 置信度
[131-v,132-v,141-ADV,142-ADV,15-1,161-0,162-0,17-0,18-1,19-COO]-> R(既然)=true^R(又)=true  407
```

Figure 9. Automatic mining results of the relational word collocation of “since/again”
图 9. “既然/又”的关系词搭配自动挖掘结果

从实验结果可知，由依存约束条件 131-v，132-v，141-ADV，142-ADV，15-1，161-0，162-0，17-0，18-1，19-COO 推出关系词搭配判别结果 R (既然) = true^R (又) = true 的支持度为 407，置信度为 1。

从上述自动挖掘结果可得一条关于关系词搭配“既然/又”的依存规则，解析：

[131-v，132-v，141-ADV，142-ADV，15-1，161-0，162-0，17-0，18-1，19-COO]->R(既然)=true^R(又)=true

可以得到“既然/又”的依存约束规则如表 5 所示：

Table 5. Dependency constraint rules of “Since/again”
表 5. “既然/又”的依存约束规则

Keymarks	dpType	dpconstrains	result
既然/又	13 + 14 + 15 + 16 + 17 + 18	parentPos (既然) = v^parentPos (又) = v + Relation (既然) = ADV^Relation (又) = ADV + relationDirection (既然，又) = 1 + isParent (既然) = 0^isParent (又) = 0 + ispEqual (既然，又) = 0 + Distance (既然，又) = 1 + Gwdp (既然，又) = COO	R (既然) = true，R (又) = true

本节主要对复句关系词搭配的依存关系进行自动挖掘，共挖掘出 87 条规则，其中包括普通规则 32 条，连用规则 20 条，句式规则 22 条，连用句式规则 13 条，如图 10 是自动挖掘得到的普通规则 ordinary-mining 的部分截图：

ID	Keymarks	dpType	dpconstraints	result
1	除了/还有	14+15+16	Relation(除了)=ADV+isParent(	R(除了)=true,R(还有)=true
2	虽说/但是	14+16	Relation(虽说)=ADV^Relation(	R(虽说)=true,R(但是)=true
3	不仅仅/而且	14+15+16	Relation(不仅仅)=ADV^Relatio	R(不仅仅)=true,R(而且)=true
4	既然/又	13+14+15+	parentPos(既然)=v^parentPos	R(既然)=true,R(又)=true
5	或则/或则	14+15	Relation(或则)=ADV^Relation(	R(或则)=true,R(或则)=true

Figure 10. Some screenshots of ordinary-mining
图 10. 普通规则 ordinary-mining 的部分截图

将挖掘得到的规则并入到原来的规则库中，由于原来的规则库中规则包括两部分，字面特征规则和依存约束规则，本节主要对依存约束规则进行挖掘，因此，字面特征约束类型 constraintType 和约束条件 constraints 两列的值默认为空即可。

6. 总结

汉语复句关系词的依存关系是指根据依存语法研究汉语依存树库中众多树形图节点之间的支配与被支配的关系，即分析经过分词后的复句中关系词之间的支配关系，把线性的复句转化为一颗依存结构树，对树形结构我们都不陌生，依存树中同样存在根节点、中间节点和叶子节点，只是在依存树中占据每个节点的是词[14]。

关系词的依存关系自动识别是理解复句语义层次的关键，目前对关系词自动识别的研究主要集中在字面特征和语法角度理解关系词，本文在汉语复句依存句法分析基础上，根据已有的 7 种依存约束条件，运用频繁项集挖掘算法 FP-tree 对依存树库中的隐藏规则进行自动挖掘，完善现有依存规则。自动挖掘规则的内容包括两部分，首先在依存树库中挖掘规则库中不存在的关系词搭配，然后，挖掘在规则库中已经存在的关系词搭配的其他规则，根据这两个挖掘需求，对实验语料进行依存特征分析，并将实验语料所需的依存约束条件进行形式化表示。实验共挖掘出 84 条依存规则。

但对于关系词自动识别的研究并不止于依存句法分析，在规则库中加入关系词所在分句的语义特征分析，根据语义相似度及相关度是关系词自动识别的下一步研究方向。

参考文献

[1] 邢福义. 汉语复句研究[M]. 北京: 商务印书馆, 2003.

[2] 姚双云. 复句关系标记的搭配研究[M]. 武汉: 华中师范大学出版社, 2008.

[3] 杨进才, 涂馨丹, 胡金柱, 等. 基于依存关系规则的汉语复句关系词自动识别[J]. 计算机应用研究, 2018, 35(6): 1756-1760.

[4] Houtsma, M. and Swami, A. (1995) Set-Oriented Mining for Association Rules in Relational Databases. *Proceedings of the* 11*th IEEE International Conference on Data Engineering*, Taipei, 6-10 March 1995, 25-34. https://doi.org/10.1109/ICDE.1995.380413

[5] Ganter, B. and Wille, R. (1999) Formal Concept Analysis: Mathematical Foundations. Springer, Berlin, 131-139. https://doi.org/10.1007/978-3-642-59830-2

[6] 况莉莉. Apriori 算法和 FP-tree 算法的探讨[J]. 淮北煤炭师范学院学报, 2010, 31(2): 44-49.

[7] 马丽生, 姚光顺, 杨传健. 基于改进 FP-tree 的最大频繁项目集挖掘算法[J]. 计算机应用, 2012, 32(2): 326-329. https://doi.org/10.3724/SP.J.1087.2012.00326

[8] 王中华. 汉语复句关系词自动标识中规则自动生成方法研究[D]: [硕士学位论文]. 武汉: 华中师范大学, 2013.

[9] 赵鹏. 海量高维数据下的频繁项目集挖掘算法研究[J]. 计算机应用与软件, 2012, 29(7): 150-153.

[10] 纪勇. 基于频繁模式的 KPI 异常检测研究[J]. 无线互联科技, 2016(15): 115-118.

[11] 袁文群. 基于子图关联规则的链接预测研究[D]: [硕士学位论文]. 重庆: 重庆大学, 2012.

[12] Agrawal. R, Imielinski. T, Swami. A. (1993) Mining Associations between Sets of Items in Massive Databases. *Proceedings of the* 1993 *ACM SIGMOD International Conference on Management of Data*, Washington DC, June 1993, 207-216. https://doi.org/10.1145/170035.170072

[13] Pei, J., Han, J., Mortazavi-Asl, B., *et al.* (2001) PrefixSpan: Mining Sequential Patterns Efficiently by Prefix-Projected Pattern Growth. *Proceedings* 17*th International Conference on Data Engineering*, Heidelberg, 2-6 April 2001, 215-224.

[14] Marcus, M., Santorini, B., *et al.* (1993) Building a Large Annotated Corpus of English: The Penn Treebank. *Computational Linguistics*, **19**, 313-330. https://doi.org/10.21236/ADA273556