第 3 章

基于众包的自然语言查询中语义关系识别规则挖掘

知识图谱上的大多数自然语言查询系统都需要依赖查询中的语义关系，是因为知识图谱中的三元组<subject，predicate，object>与语义关系<arg1，rel，arg2>具有相同的形式。然而到目前为止，现有方法中的语义关系识别规则都来自人为假设，具有较强的主观性和较低的识别率。因此，本章将从数据集中挖掘语义关系识别的规则，由此可以替代人为假设的规则，更准确地识别自然语言查询中的语义关系。第一，提出了新的众包模型——带反馈的并行式迭代模型，用于生成语义关系数据集。该模型不仅继承了迭代模型的完整性和并行模型的准确性，而且还节省了人力资源。第二，从依赖结构和语义关系的关联中挖掘了语义关系识别的规则，其中依赖结构数据集由 Stanford Parser 生成，语义关系数据集由带反馈的并行式迭代模型生成。第三，基于挖掘出的语义关系识别规则，提出了一种针对知识图谱的自然语言查询的语义关系识别算法，且实验结果表明该算法能够识别比现有方法更多的语义关系。

3.1 问题描述及创新点

知识图谱的自然语言查询易于使用且具有强大的表达能力，但受限于语义关系的识别能力。大多数自然语言查询系统包含三个阶段：从自然语言查询中识别语义关系，为语义关系产生候选映射，将语义关系映射的组合转换为 SPARQL 语句或从知识图谱中搜索包含语义关系映射组合的子图。很显然，语义关系在知识图谱的自然语言查询中有着非常重要的作用。然而，现有的最顶尖的自然语言查询系统，如 DEANNA[38]和图数据驱动方法[46]，对知识图谱的自然语言查询具有巨大的推动作用，但是受限于自然语言查询中的大多数语义关系不能完全被识别，导致查询的回答率都没有太大的提高。

现有的很多自然语言查询方法[36~39,46,47]可以通过人工假设的规则来识别自然语言查询中的一些语义关系。Yahya[36~39]和 Zou[46]等人依靠自然语言查询中的动词短语和解释词典 D 来识别语义关系，其中，解释词典记录了动词短语和谓词之间的语义对等关系。方法的基本思路是找到自然语言查询和解释词典 D 共有的所有动词短语，然后分别根据知识图谱

[36,37,38,39]或启发式语言规则[46]从自然语言查询中找到动词短语的两个关联参数。最后，动词短语与两个关联参数形成一个语义关系<arg1，rel，arg2>。然而，上述方法存在两个主要缺点：如果解释词典 D 不包含自然语言查询中的动词短语，或者语义关系中的关系短语 rel 不是动词短语，则现有的查询系统无法识别语义关系。例如，对于查询“How many books by[1] Kerouac were published by Viking Press?”，动词短语“published”很有可能出现在解释词典 D 中，而非动词短语“by[1]”则不会出现在解释词典中。因此，现有方法可以识别查询中的语义关系<Kerouac，published，Viking_Press>，却不能识别语义关系<book，by[1]，Kerouac>。此外，Liu 等人[47]使用了一些依赖结构来识别自然语言查询中的语义关系，比如“nsubj”“nmod:of”“amod”等，对于自然语言查询中的语义关系识别有一定的改善，但是仅使用这些依赖结构还是远远不够的。

因此，本章期望提出一种方法，能够从数据中挖掘出一些规则，然后用这些规则识别自然语言查询中的大部分语义关系，而不是依靠人为假设的规则，如动词短语等。基于数据挖掘中关联分析的核心思想，从依赖结构和语义关系的关联中挖掘语义关系识别规则。此外，为了实现这个目标，必须获得依赖结构数据集和语义关系数据集，其中依赖结构数据集可以由 Stanford Parser 生成，而语义关系数据集则需要手动收集。众包是帮助解决计算机难以处理但是人类可以轻松处理的各种各样问题的人类组织模型，因此非常适合用于生成语义关系数据集。然而，目前的众包模型还不够完善，只能在较少的错误识别（但不完整）或更好的完整性（但更多的错误识别）两种众包模型（迭代模型/并行模型）之间进行选择。由于数据挖掘算法要想获得高质量的结果，必须要有高质量的数据作为基础，因此本章还提出了新的众包模型，该模型具有较少的错误识别（即准确性）和良好的完整性，以此保证挖掘得到的规则的可靠性。

为了解决上述问题，本章的贡献主要有以下 3 点。

（1）提出了一种新的众包模型（即带反馈的并行式迭代模型），用于生成语义关系数据集。该众包模型分别从迭代模型和并行模型中继承了完整性和准确性，而且在新模型中两个子模型可以相互抑制对方的缺点，从而提高结果的质量。此外，由于众包工作者对自己的错误总是比其他众包工作者更敏感，因此让同一组众包工作者（而不是多组众包工作者）在模型中迭代执行任务，既可以改善结果质量又节省人力资源。

（2）从依赖结构数据集和语义关系数据集中挖掘依赖结构与语义关系之间的关联，即语义关联规则，包含四种重要规则：subject-like，object-like，triple-like 及依赖结构组合。依赖结构数据集和语义关系数据集分别是由 Stanford Parser 和最新提出的众包模型（即带反馈的并行式迭代模型）从自然语言查询中获得。

（3）基于挖掘得到的语义关联规则，提出了针对知识图谱的自然语言查询的语义关系识别算法，且实验结果表明该算法能够在自然语言查询中识别出比现有的方法更多的语义关系。

3.2 众包模型

在本节中，首先介绍两种常见的众包模型：迭代模型和并行模型。然后，基于上述模型提出了两种混合模型：迭代式并行模型和并行式迭代模型。最后，基于这两种混合模型，

提出了一种高质量、低人力的众包模型：带反馈的并行式迭代模型。

3.2.1　迭代模型和并行模型

作为常见的众包模型，迭代模型[213~217]和并行模型[218~222]可以弱化不诚实或恶意众包工作者对众包结果的影响。

迭代模型：n 个众包工作者逐一执行相同的任务，也就是说，第 $i+1$ 个众包工作者在第 i 个众包工作者的结果上执行此任务，直到所有众包工作者都完成任务。最后一个众包工作者的结果被认为是众包模型的最终结果。迭代模型如图 3-1 所示。

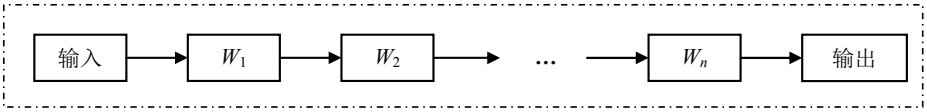


图 3-1　迭代模型

并行模型：n 个众包工作者独立完成相同的任务。因此，在所有工作者完成任务后有 n 个结果，通过将 n 个结果聚合得到众包模型的最终结果。在聚合过程中，被 k（k 由专家指定）个或更多众包工作者认可的部分才能作为最终结果。并行模型如图 3-2 所示。

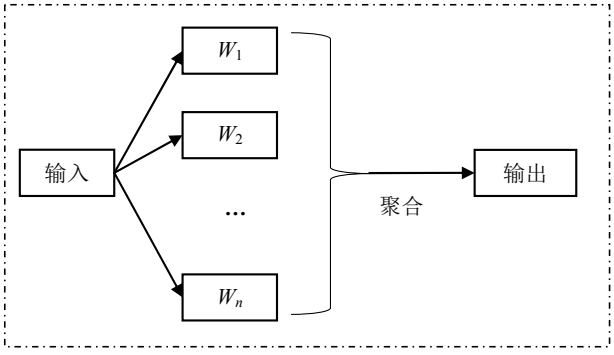


图 3-2　并行模型

3.2.2　迭代式并行模型和并行式迭代模型

迭代模型和并行模型各有优劣，迭代模型可以保证结果的完整性，即可以识别更多的对象，而并行模型的结果是众包工作者的共识，即可以保证识别结果的准确性。通过考虑两个模型的整体优势，本章提出了如图 3-3 所示的迭代式并行模型和如图 3-4 所示的并行式迭代模型（$W_{i\text{-}j}$ 表示第 i 个工作组中的第 j 个众包工作者，n 为工作组的个数，m 为每组众包工作者人数，矩形代表众包工作者正在执行众包任务）。

迭代式并行模型：众包工作者分为 n 组，每组众包工作者按照迭代模型的方式执行众包任务，由此分别得到 n 组的识别结果。最终以并行模型的结果聚合方式生成模型的最终结果。

并行式迭代模型：众包工作者分为 n 组，组内以并行模型的方式执行众包任务，同一组众包工作者被认为是迭代模型中的一次迭代，即组与组之间按照迭代模型的方式执行众包任务，最后一组的结果经过聚合之后得到模型的最终结果。

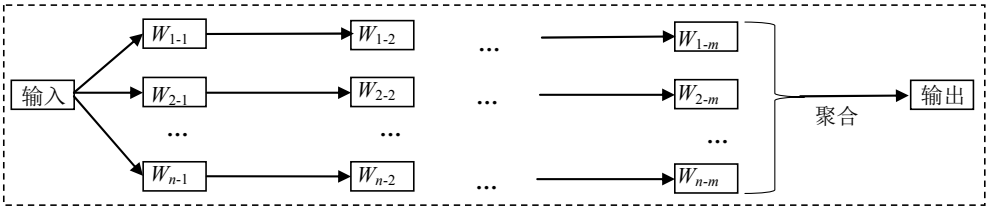


图 3-3　迭代式并行模型

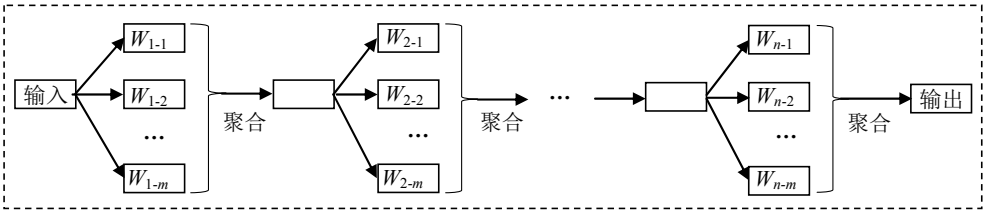


图 3-4　并行式迭代模型

3.2.3　带反馈的并行式迭代模型

理论和后续的实验都表明并行式迭代模型优于迭代式并行模型。迭代式并行模型和并行式迭代模型都融合了迭代模型的完整性和并行模型的准确性。在迭代式并行模型中，组与组之间是并行的，因此信息不共享，每个众包工作者只能在本组中之前众包工作者的基础上执行众包任务。然而，在并行式迭代模型中，组与组之间是迭代的，每个众包工作者是在本组之前的所有众包工作者的基础上执行众包任务。因此，并行式迭代模型中的众包工作者比迭代式并行模型中的众包工作者具有更好的任务基础（即任务结果具有更好的广度和深度），由此并行式迭代模型中的众包工作者可以继续不断扩展任务结果的广度和深度，任务也将被快速推进，在众包工作者人数相同时，并行式迭代模型将获得更多更好的结果。本章后续的实验也验证了这一点。

综上所述，并行式迭代模型的性能优于迭代式并行模型的性能，因此在并行式迭代模型的基础上，提出了带反馈的并行式迭代模型。该模型不仅继承了并行式迭代模型的优势，还考虑了众包工作者总是对自己的错误比对其他众包工作者的错误更敏感的特性，因此让同一组众包工作者迭代执行众包任务，即在每次迭代中都是同一组众包工作者执行众包任务，带反馈的并行式迭代模型如图 3-5 所示，不仅提高了结果的质量，而且节省了人力。

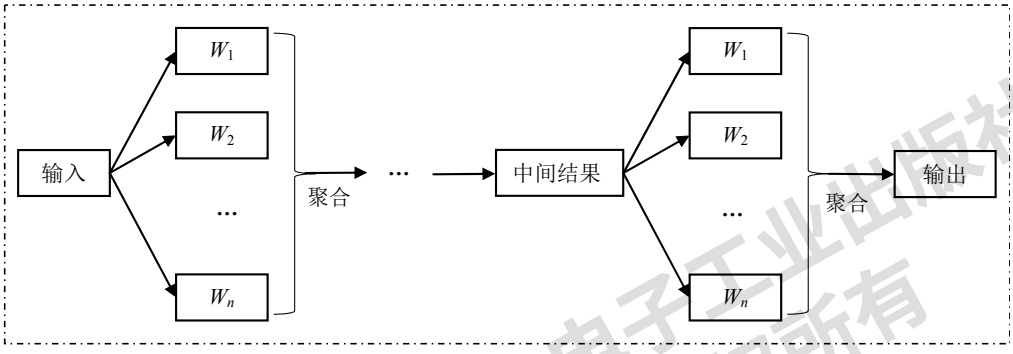


图 3-5　带反馈的并行式迭代模型

3.3　生成语义关系数据集和依赖结构数据集

3.3.1　众包模型标记语义关系

众包平台如亚马逊 AMT（Amazon Mechanical Turk）和猪八戒网等，可以轻松招募很多众包工作者完成微任务（即人类智能任务/Human Intelligence Task，简称 HIT）。为了让众包工作者以众包模型（即带反馈的并行式迭代模型）的组织方式标记自然语言查询中的语义关系，本章创建了如图 3-6 所示的标记语义关系的 HIT，并在亚马逊 AMT 招募了众包工作者。在 HIT 中，众包工作者在查询中发现语义关系时，需要写下语义关系（即主语，关系短语，宾语），然后通过按钮“Next Query”处理下一个查询，或通过按钮“Previous Query”修订之前的结果。由此可以获得所有自然语言查询中的语义关系数据集。

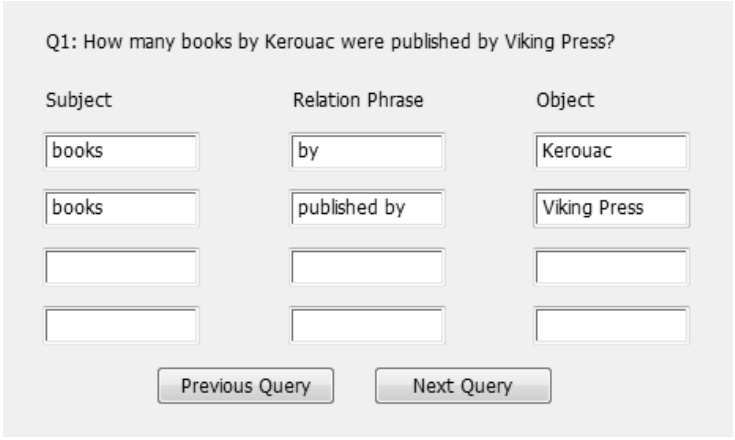


图 3-6　标记语义关系的 HIT

定义 3.1　（语义关系，R）一个语义关系是一个三元组，代表了用户查询的部分意图，表示为 R<arg1, rel, arg2>，其中 rel 是关系短语，arg1 和 arg2 是两个关联参数。

例 3.1　对于查询“How many books by[1] Kerouac were published by Viking Press?”，<books, published, Viking_Press> 是一个语义关系，其中“published”是关系短语 rel，“books”和“Viking_Press”分别是关系短语“published”的两个关联参数即 arg1 和 arg2。此外，还可以在查询中找到另一个语义关系<books, by[1], Kerouac>。

3.3.2　Stanford Parser 生成依赖结构

自然语言处理的文献[190]表示依赖结构更适合关系抽取，且 Stanford Parser（http://nlp.stanford.edu:8080/ parser /, 访问日期：2018 年 11 月）是生成依赖结构的最好的工具。因此，本章用 Stanford Parser 从自然语言查询中获得依赖结构。图 3-7 为“How many books by[1] Kerouac were published by Viking Press?”查询示例的依赖结构。

advmod(many-2, How-1)
amod(books-3, many-2)
nsubjpass(published-7, books-3)
case(Kerouac-5, by-4)
nmod:by(books-3, Kerouac-5)
auxpass(published-7, were-6)
root(ROOT-0, published-7)
case(Press-10, by-8)
compound(Press-10, Viking-9)
nmod:by(published-7, Press-10)

图 3-7　查询示例的依赖结构

3.4　挖掘语义关联规则

3.4.1　挖掘语义关联规则的算法

定义 3.2　（包含关系）如果依赖结构中的所有短语都存在于语义关系中，则依赖结构和语义关系满足包含关系。

定义 3.3　（候选项集）一个候选项集由一个或两个依赖结构和一个语义关系组成，每个依赖结构和语义关系满足包含关系。

例 3.2　对于查询“How many books by[1] Kerouac were published by Viking Press?”，“{nmod, *R*}”（即“{nmod:by(books, Kerouac), *R*<books, by[1], Kerouac>}”）是一个候选项集，而“{amod, *R*}”（即“{amod(books, many), *R*<books, by[1], Kerouac>}”）不是候选项集，因为后者不满足邻近关系，即短语“many”不存在于语义关系“*R*<books, by[1], Kerouac>”中。

定义 3.4　（参与率 PR）候选项集（表示为 c）中的第 i 个依赖结构（表示为 dep_i）的参与率 PR，被定义为候选项集 c 中依赖结构 dep_i 的出现次数与数据集中依赖结构 dep_i 的出现次数的比值，即：

$$\mathrm{PR}\left(\mathrm{dep}_i, c\right)=\frac{|\pi_{\mathrm{dep}_i}(c)|}{|\pi_{\mathrm{dep}_i}(U)|} \tag{3.1}$$

其中，π 是关系的投影操作，U 表示依赖结构全集。

定义 3.5　（参与度 PI）候选项集 c 的参与度 PI 是 c 中所有依赖结构的参与率 PR 的最小值，即：

$$\mathrm{PI}(c)=\min_{i=1}^{n}\mathrm{PR}(\mathrm{dep}_i, c) \tag{3.2}$$

定义 3.6　（频繁项集）c 是一个频繁项集，仅当 c 中任意依赖结构 dep_i 在数据集中的出现次数大于给定的出现次数阈值 min_count，且 c 的参与度 PI(c)大于参与度阈值 min_prev（其中 min_count 和 min_prev 都是由专家确定的）。

例 3.3　对于图 3-8 中的候选项集{dobj, *R*}，其中的依赖结构“dobj”的出现次数是 1（即{dobj(have-6, employees-3), *R*<google, have, employees>}），且依赖结构“dobj”在依赖结构数据集中的出现次数为 2（即“dobj(Give-1, websites-4)”和“dobj(have-6, employees-3)”），

可知候选项集 c={dobj,R}中的“dobj”的参与率为 0.5。由于 c 中只有一个依赖结构，所以 PI(c)=0.5。此外，如果出现次数阈值 min_count=1 且参与度阈值 min_prev=0.6，因为 c 的参与度小于参与度阈值，则候选项集 c={dobj, R}是非频繁项集。

定义 3.7　（语义关联规则）语义关联规则是一个表达式 $X \Rightarrow R$，其中 X 是由一个或两个依赖结构组成的项集，R 是自然语言查询中的语义关系，且{X,R}是频繁项集。

例 3.4　对于频繁项集“{nmod:by(books, Kerouac), R<books, by[1], Kerouac>}”和“{nsubjpass(published, books)，nmod:by(published, Press), R<books, published, Viking_Press>}”，可以分别简写为“{nmod, R}”和“{nsubjpass, nmod, R}”。由此，可以得到两个语义关联规则“{nmod}$\Rightarrow R$”和“{nsubjpass, nmod}$\Rightarrow R$”。此外，后者还包含了两个子规则，即“{nsubjpass}$\Rightarrow R$”和“{nmod}$\Rightarrow R$”。

本章的语义关联与传统的关联分析相似，但并不相同。相似点：传统的关联分析和本章的语义关联都是挖掘项集的频繁共现关系，即传统的关联分析中的“项集”频繁地出现在事务中，而本章的语义关联中的“项集”（由“依赖结构”和“语义关系”组成）频繁地来自相同的自然语言查询。不同点：①传统的关联分析中的项集来自事务，即项集共同出现在多个事务中，而本章中的语义关联的项集来自语义关系和依赖结构的包含关系，即项集中的语义关系包含了依赖结构中的短语；②传统的关联分析的模式阶数由项的个数决定，而本章的语义关联的模式阶数只有二阶和三阶；③挖掘传统的关联分析的频繁项集，可以通过低阶频繁模式不断连接和验证频繁度，得到高阶模式，而挖掘语义关联的频繁项集，不需要依靠连接得到高阶，只需要确认语义关系和依赖结构的包含关系及验证频繁度即可。

综上所述，传统的关联分析和本章的语义关联的本质都是挖掘项集的频繁共现关系，但是模式的构造和挖掘方法不同。因此，本章只使用传统的关联分析的核心思想并根据语义关联的特点设计算法。

本章提出了挖掘语义关联规则的算法，即算法 MSAR（Mining Semantic Association Rules），算法的思路如下：首先，分别用 Stanford Parser 和众包模型从自然语言查询中生成依赖结构数据集和语义关系数据集（step1～2）。其次，通过确认语义关系和依赖结构的包含关系，得到候选项集（step3）。然后，计算每个候选项集的参与率和参与度（step4）。再通过出现次数阈值 min_count 和参与度阈值 min_prev，生成频繁项集（step5）。最后将频繁项集转换为语义关联规则（step6）。

Algorithm MSAR (Mining Semantic Association Rules)

Require: **Input:** natural language query δ_Q, min_count, min_prev
Output: semantic association rules δ_{SAR}

step1: δ_{dep}=Stanford_Parser(δ_Q)
step2: δ_R=Crowdsourcing(δ_Q)
step3: δ_c=Produce_candicate_itemset(δ_{dep},δ_R)
step4: $\delta_{c\_with\_PI}$=Caculate_PR_and_PI(δ_c)
step5: $\delta_{frequent\_c}$=Produce_frequent_itemset($\delta_{c\_with\_PI}$, min_count, min_prev)
step6: δ_{SAR}=Produce_rules($\delta_{frequent\_c}$)

图 3-8 给出了语义关联规则挖掘的示例。对于查询“Query 1: Give me the websites of

companies with more than 500000 employees.”和“Query 2: How many employees does Google have?”，可以依靠 Stanford Parser 从两个查询中获得依赖结构（step1），图中省略了一些不能用于产生语义关系的依赖结构，然后通过包含关系产生候选项集（step2）。之后，候选项集将被简化，并且计算参与率和参与度（step3）。最后，根据给定的出现次数阈值 min_count 和参与度阈值 min_prev，得到频繁项集和语义关联规则（step4～5）。

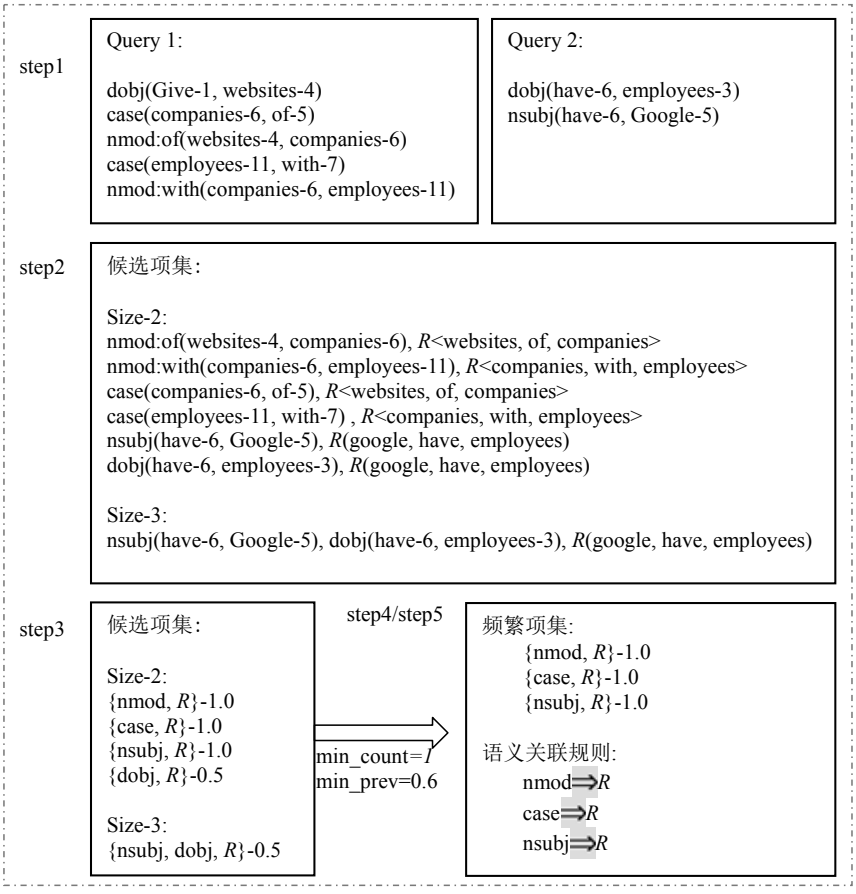


图 3-8　语义关联规则挖掘的示例

3.4.2　算法 MSAR 的复杂度

由于本章只使用传统的关联分析的核心思想，模式的最大阶数为三阶，因此本章的复杂度远低于传统的关联分析。为了分析复杂度，本章假设有 N 个自然语言查询，并且一个查询中的语义关系和依赖结构的个数最多为 R 和 D。

由文献[247,46]可知，Stanford Parser 对于每个查询的时间复杂度是 $O(R^3)$，所以 step1 中的时间复杂度为 $O(N \cdot R^3)$。在 step2 中，对于每个查询，四个众包工作者识别两次语义关系的时间复杂度为 $O(8 \cdot N)=O(N)$。在 step3 中，分析语义关系是否包含每个依赖结构的时间复杂度为 $O(N \cdot R \cdot D)$。在 step4 中，计算所有依赖结构的出现次数（时间复杂度为 $O(N \cdot D)$），以及所有候选项集的出现次数，因为每个语义关系都可以与三个候选项目集关

联，所以时间复杂度为 $O(N \cdot R \cdot 3)=O(N \cdot R)$，然后计算每个候选项集的 PR 和 PI（时间复杂度为 $O(N \cdot R)$）。在 step5 中，删除所有不频繁的候选项集的时间复杂度为 $O(N \cdot R)$。最后，step6 为所有频繁项集生成语义关联规则的时间复杂度为 $O(N \cdot R)$。综上所述，MSAR 算法的时间复杂度为 $O(N \cdot R^3)+O(N)+O(N \cdot R \cdot D)+O(N \cdot D)+O(N \cdot R)+O(N \cdot R)+O(N \cdot R)+O(N \cdot R) = O(N \cdot R^3+N \cdot R \cdot D)$。

存储自然语言查询、语义关系、依赖结构的空间复杂度分别为 $O(N)$、$O(R)$、$O(D)$。此外，由于每个语义关系可以与三个候选项集相关联，并且频繁项集的数量小于候选项集的数量，因此存储候选项集的 PR/PI 和存储频繁项集的语义关联规则的空间复杂度分别为 $O(3 \cdot R \cdot 2)=O(R)$和 $O(3 \cdot R)=O(R)$。综上所述，算法 MSAR 的空间复杂度为 $O(N)+O(R)+O(D)+O(R)+O(R)=O(N+R+D)$。

3.5　实验结果及分析——众包模型

为了避免数据量小会影响众包模型分析的效果，因此首先在数据量较大的震后遥感图像数据集上进行实验，用于分析和验证所有模型（迭代模型、并行模型、迭代式并行模型、并行式迭代模型、带反馈的并行式迭代模型），然后在数据量较小的自然语言查询数据集上进行实验，用于确认带反馈的并行式迭代模型也是有效的。

3.5.1　实验数据及评估标准

实验包括两个数据集，第一个数据集是 260 平方公里范围的双石镇和灵关镇的震后遥感图像，两个镇都邻近 2013 年雅安 7 级地震的震中心。本章把图像中的房屋作为震后搜救目标，震后遥感图像来源于国家基础地理信息中心（http://ngcc.sbsm.gov.cn/，访问日期：2018 年 11 月）。第二个数据集是 20 个自然语言查询，众包任务是从自然语言查询中识别出语义关系。实验的众包工作者来自众包平台亚马逊 AMT，每个模型中众包工作者人数为 8 名或 9 名。用户判断标记目标是否正确的参考数据由地震专家提供。

本章使用准确率、召回率、加权度量作为评估标准来评估众包模型的实验结果，并考虑认同比 Agreement 对模型的影响。对于众包模型执行结果，通常有两种常见错误：①众包工作者的结果可能不正确；②众包工作者可能会丢失一些结果。准确率用于评估目标的正确性，精度越高，错误结果就越少。召回率用于评估结果的完整度，召回率越高，缺失的结果越少。F-measure 是 Precision 和 Recall 的加权度量，当存在较少的不正确结果和较少的缺失结果时，F-measure 会很高，这意味着结果的质量越好。在汇总各种类型的并行结果时，认同比 Agreement 也是影响汇总结果质量的一个参数，本章也会分析 Agreement 如何影响实验结果的质量。

定义 3.8　（认同比，Agreement，简写为 A）当所有 n 个众包工作者中的 m 个众包工作者都认同了同一个结果，则这个结果被作为众包模型的识别结果，认同比表示为

$$A = m \tag{3.3}$$

定义 3.9　（准确率，Precision）在以众包模型的组织方式执行完众包任务后，返回结果中正确结果的个数与返回的结果个数之比，即：

$$\text{Precision} = \frac{|E \cap V|}{|V|} \tag{3.4}$$

其中，E 是专家认定的标准参考结果，V 是众包工作者返回的结果。

定义 3.10 （召回率，Recall）在以众包模型的组织方式执行完众包任务后，返回结果中正确结果的个数与标准参考结果的个数之比，即：

$$\text{Recall} = \frac{|E \cap V|}{|E|} \tag{3.5}$$

定义 3.11 （加权度量，F-measure）F-measure 使得评测结果可以在准确率 Precision 和召回率 Recall 之间进行权衡，即：

$$\text{F-measure} = \frac{2 \bullet \text{Precision} \bullet \text{Recall}}{\text{Precision} + \text{Recall}} \tag{3.6}$$

3.5.2 迭代模型和并行模型

在迭代模型中，招募了 9 名众包工作者执行震后救援目标识别，迭代模型的实验结果如图 3-9 所示（I 为迭代次数）。从图中可以看出，随着迭代次数的增加，Precision、Recall 和 F-measure 逐渐趋于稳定。因此，后续众包工作者不会显著改善以前的目标识别结果，并且当迭代次数达到 3 次时，迭代模型可以获得良好的结果。此外，由于不诚实或恶意众包工作者的出现，结果的质量在第 5 次迭代中有所下降，这是迭代模型的一个缺点。一旦这位众包工作者出现在最后一次迭代中，以前的所有努力都将徒劳无益。

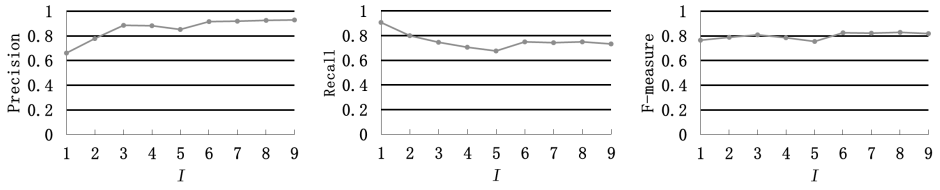


图 3-9　迭代模型的实验结果

在并行模型中，招募了 9 名众包工作者执行震后救援目标识别，并行模型实验结果如图 3-10 所示（A 为认同比）。从图中可以看出，当 A 从 1 到 8 的变化过程中，准确率 Precision 不断上升，召回率 Recall 不断下降。这是因为当准确率 Precision 上升时，每个目标都必须得到更多众包工作者的认同，目标更可能是正确的救援目标且目标数量逐渐减少。根据 F-measure 的变化，当 A 为 2、3、4 时，F-measure 达到最高点，模型表现最好。因此，在并行模型中，当 2 到 4 个众包工作者仔细工作时，并行模型可以获得良好的结果。

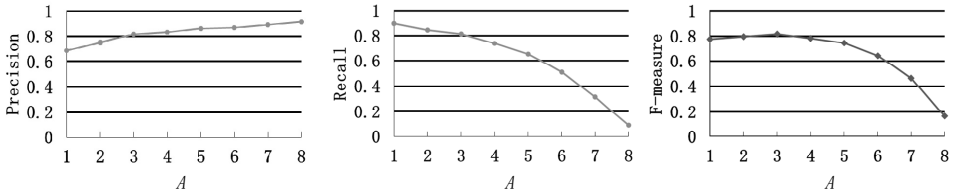


图 3-10　并行模型的实验结果

3.5.3　迭代式并行模型和并行式迭代模型

根据迭代模型和并行模型可知，为了获得好的结果，迭代模型需要 3 次迭代，并行模型中需要 2 到 4 名众包工作者。因此，设计了三种类型的迭代式并行模型实验：①2 个并行组，每组包括 4 次迭代（即图 3-3 中的 $n=2$ 和 $m=4$），迭代式并行模型结果如图 3-11 所示；②3 个并行组，每组包括 3 次迭代（即图 3-3 中的 $n=3$ 和 $m=3$），迭代式并行模型结果如图 3-12 所示；③4 个并行组，每组包括 2 次迭代（即图 3-3 中的 $n=4$ 和 $m=2$），迭代式并行模型结果如图 3-13 所示。在这三种类型的迭代式并行模型中，并行组的数量分别为 2、3、4，对应的迭代次数分别为 4、3、2，因此，所需众包工作者人数分别为 8、9、8（并行式迭代模型也有类似情况，且后续的实验表明一个众包工作者的差异对结果的影响可以忽略）。

迭代式并行模型主要由并行组主导，因此使用赞同比 A 作为 x 轴。当每个小组在模型中完成相应迭代次数之后，聚合所有结果。为了更好地了解众包工作者在目标识别过程中的进展，在每次迭代之后查看聚合的实验结果，并将其与专家的结果进行比较。由图 3-11、图 3-12、图 3-13 中的实验结果可知，①迭代式并行模型具有迭代模型的特点，即随着迭代次数的增加，准确率 Precision 会上升，Recall 会下降；②迭代式并行模型也具有并行模型的特点，即随着 A 的增加，准确率 Precision 会下降，Recall 会上升。当 A 为 2 时，迭代式并行模型的 F-measure 高于迭代模型和并行模型的 F-measure。

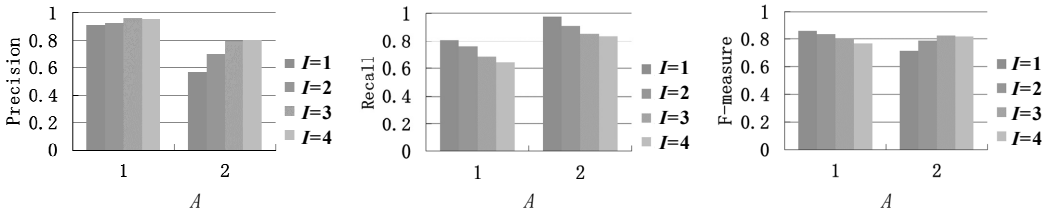


图 3-11　迭代式并行模型结果（2 个并行组）

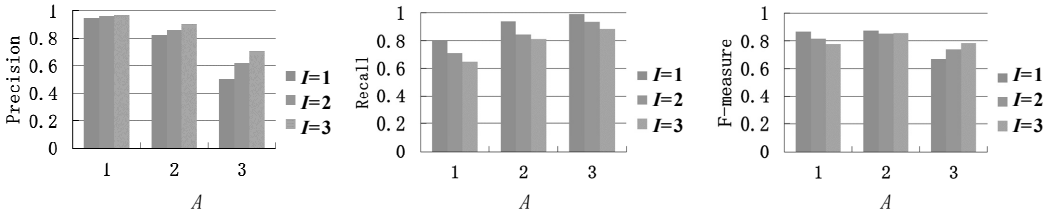


图 3-12　迭代式并行模型结果（3 个并行组）

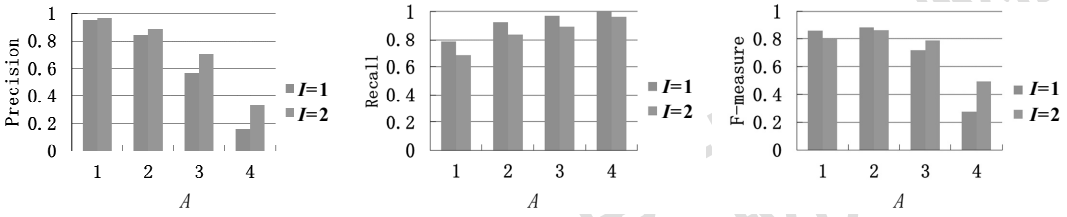


图 3-13　迭代式并行模型结果（4 个并行组）

同理，设计了三种类型的并行式迭代模型实验：①2 次迭代，每次迭代包括 4 名并行众包工作者（即图 3-4 中的 $n = 2$ 和 $m = 4$），并行式迭代模型结果如图 3-14 所示；②3 次迭代，每次迭代包括 3 名并行众包工作者（即图 3-4 中的 $n = 3$ 和 $m = 3$），并行式迭代模型结果如图 3-15 所示；③4 次迭代，每次迭代包括 2 名并行众包工作者（即图 3-4 中的 $n = 4$ 和 $m = 2$），并行式迭代模型结果如图 3-16 所示。

并行式迭代模型主要由迭代组主导，因此使用迭代次数 I 作为 x 轴。由图 3-14、图 3-15、图 3-16 中的实验结果可知，并行式迭代模型具有明显的迭代模型特性，即 Precision、Recall 和 F-measure 随着迭代次数的增加而逐渐稳定。而且，并行式迭代模型可以很好地弥补并行模型的缺陷（让众包工作者很好地对结果达成共识），即 A 只对第一次迭代的结果有重大影响，在之后的迭代中共识已经基本形成。

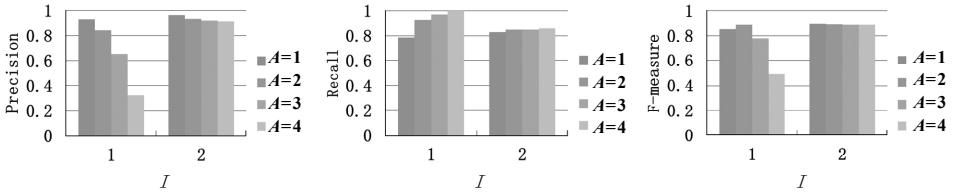


图 3-14　并行式迭代模型结果（2 次迭代）

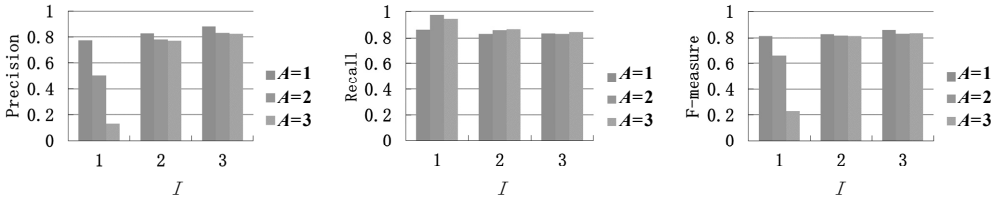


图 3-15　并行式迭代模型结果（3 次迭代）

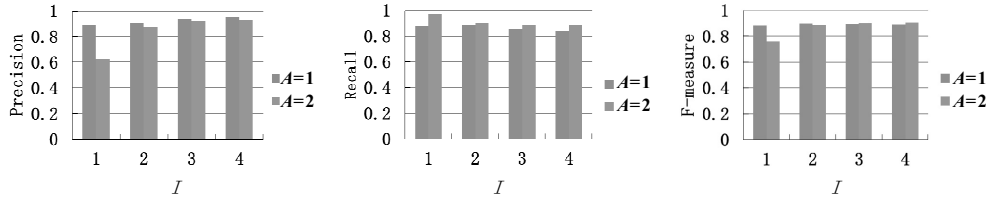


图 3-16　并行式迭代模型结果（4 次迭代）

此外，对比了三种迭代式并行模型（图 3-17）和三种并行式迭代模型（图 3-18）的最优结果，得出以下结论：①在三种迭代式并行模型中，3 个并行组和 4 个并行组的结果比 2 个并行组的结果好；②在三种并行式迭代模型中，2 次迭代和 4 次迭代的结果比 3 次迭代的结果好；③并行式迭代模型比迭代式并行模型的结果好。

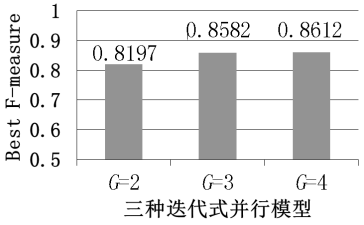


图 3-17　迭代式并行模型对比

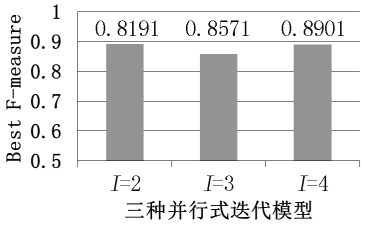


图 3-18　并行式迭代模型对比

3.5.4　带反馈的并行式迭代模型

在比较了三种迭代式并行模型（图 3-17）和三种并行式迭代模型（图 3-18）的结果之后，得出结论：并行式迭代模型比迭代式并行模型更好。因此，设计了带反馈的并行式迭代模型，即下一次迭代中的众包工作者与前一次迭代中的众包工作者相同。为了更好地避免不诚实或恶意众包工作者的影响，招募了 4 名众包工作者在带反馈的并行式迭代模型中迭代两次（即图 3-5 中的 $n = 2$ 和 $m = 4$），结果如图 3-19 所示。一方面，4 名众包工作者比 2 名众包工作者更具稳定性，因为 2 名众包工作者中如果存在不诚实或恶意众包工作者，对结果影响较大，而 4 名众包工作者中出现不诚实或恶意众包工作者，对结果影响较小。另一方面，2 名众包工作者需要迭代 4 次，而 4 名众包工作者只需要迭代两次，迭代次数较多很容易导致众包工作者的不满。因此，4 名众包工作者迭代两次的方案较优。

带反馈的并行式迭代模型由迭代组主导，因此使用迭代次数 I 作为 x 轴。如图 3-19 中的实验结果所示，带反馈的并行式迭代模型继承和改进了并行式迭代模型的优点。如图 3-20 所示，带反馈的并行式迭代模型（M5）的 F-measure 为 0.92，而迭代模型（M1）、并行模型（M2）、迭代式并行模型（M3）、并行式迭代模型（M4）的 F-measures 分别为 0.8，0.83，0.86 和 0.89。因此，带反馈的并行式迭代模型比以上所有模型的执行结果更好。不仅提高了结果质量，而且减少了完成任务的人力资源数量（仅四名众包工作者）。

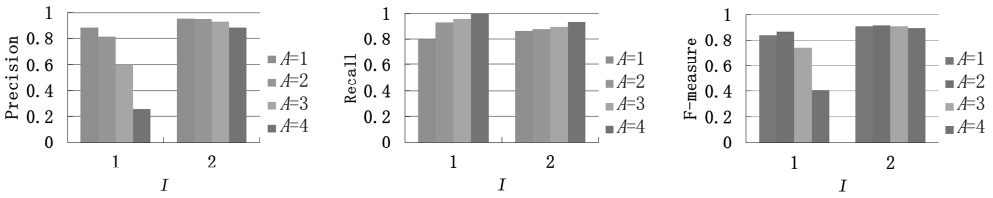


图 3-19　带反馈的并行式迭代模型结果

此外，在自然语言查询数据集上也进行了少量实验，对比了迭代模型（M1）、并行模型（M2）、带反馈的并行式迭代模型（M5）的结果，自然语言查询数据集的实验结果如图 3-21 所示。对于 20 个自然语言查询，并行模型可以正确识别其中 15 个查询的语义关系，迭代模型可以正确识别其中 17 个查询的语义关系，而带反馈的并行式迭代模型可以正确识别所有查询（即 20 个）的语义关系。很明显，并行模型中众包工作者是独立的，迭代模型中众包工作者可以改进之前的结果，但也容易陷入局部最优，带反馈的并行式迭代模型可

以通过别人的结果不断改进众包工作者自己的识别方法和结果。

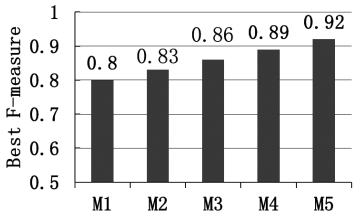


图 3-20　所有模型的结果对比

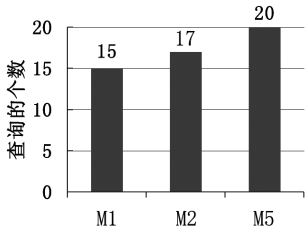


图 3-21　自然语言查询数据集的实验结果

3.6　实验结果及分析——语义关联规则

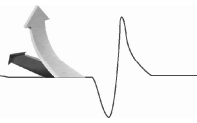
定义 3.12　（subject-like, object-like, triple-like）有三种依赖结构可以用于生成语义关系。如果依赖结构中的两个短语分别是语义关系中的 arg1 和 rel (或 rel 和 arg2)，则此依赖结构属于 subject-like (或 object-like)。类似地，如果依赖结构中的两个短语是语义关系中的 arg1 和 arg2，则此依赖结构属于 triple-like。

例 3.5　对于语义关联规则“{nsubjpass(published, books)，nmod:by(published, Press)} ⇒ *R*<books, published, Viking_Press>”，依赖结构“nsubjpass”和“nmod”分别属于 subject-like 和 object-like。对于语义关联规则“{nmod:by(books, Kerouac)}⇒*R*<books, by, Kerouac>”，依赖结构“nmod”属于 triple-like。

除了具有明确分类的的依赖结构，还发现了一些新的依赖结构。在挖掘语义关联规则之前，很多依赖结构都有明确分类，如“subj, nsubj, nsubjpass, csubj, csubjpass, xsubj”和“obj, pobj, dobj, iobj”分别属于 subject-like 和 object-like。在挖掘语义关联规则之后，发现了一些新的依赖结构如表 3-1 所示，依赖结构“acl”属于 subject-like，依赖结构“case”和“dep”属于 object-like，依赖结构“nmod”既属于 object-like 又属于 triple-like。此外，在表 3-1 中，还展示了每个依赖结构的参与度 PI 和参与率 PR，由于表中的每个项集只有一个依赖结构，所以参与度 PI 和参与率 PR 是相等的。例如，“nsubjpass-0.97(31/32)”表示在依赖结构数据集中“nsubjpass”有 32 个实例，其中 31 个属于 subject-like，因此依赖结构“nsubjpass”在项集 *c*={nsubjpass, R}中的参与率及项集 *c* 的参与度为 0.97。

表 3-1　一些新的依赖结构

是否频繁	subject-like	PI(PR)	object-like	PI(PR)	triple-like	PI(PR)
频繁的 (min_prev=0.2 且 min_count>15)	nsubjpass nsubj acl	0.97(31/32) 0.73(99/136) 0.89(17/19)	case dobj dep nmod advmod	0.57(95/166) 0.56(61/108) 0.38(6/16) 0.36(62/172) 0.21(9/42)	nmod	0.52(89/172)
非频繁的 (min_prev=0.2 且 min_count>15)	expl dep case cop dobj nmod	0.33(1/3) 0.19(3/16) 0.03(5/166) 0.07(3/43) 0.01(1/108) 0.01(1/172)	expl xcomp cop acl iobj	0.33(1/3) 0.33(2/6) 0.12(5/43) 0.05(1/19) 0.02(1/42)	nsubj	0.06(8/136)



语义关联规则挖掘算法还发现了一些可用于生成语义关系的依赖结构组合，如表 3-2 所示。由于分析这些组合的目的是了解从依赖结构到语义关系的组合规则，所以未设置 min_prev，以及依赖结构组合是否频繁取决于每个依赖结构是否频繁。表 3-2 中还展示了每个依赖结构组合的参与度 PI 和参与率 PR，例如“nsubj,dobj-0.41(56/136|56/108)”意味着依赖结构组合“nsubj,dobj”的 56 个实例可以生成语义关系，在依赖结构数据集中“nsubj”和“dobj”分别有 136 个和 108 个实例，因此依赖结构“nsubj”和“dobj”在项集 c={nsubj, dobj, R}中的参与率 PR 分别是“56/136”和“56/108”，并且项集 c 的参与度 PI 是 0.41（即最小的参与率 56/136）。

表 3-2　依赖结构组合

是否频繁	依赖结构组合	PI(PR)
频繁的	nsubj, dobj nsubj, nmod nsubjpass, nmod nsubjpass, advmod acl, nmod nsubj, dep nsubj, advmod nsubjpass, dep acl, dobj nsubj, iobj nsubjpass, dobj	0.41(56/136|56/108) 0.16(28/136|28/172) 0.13(22/32|22/172) 0.10(4/32|4/42) 0.08(14/19|14/172) 0.04(5/136|4/16) 0.04(5/136|5/42) 0.03(1/32|1/16) 0.02(2/19|2/108) 0.01(1/136|1/42) 0.01(1/32|1/108)
非频繁的	case, case nsubjpass, acl nsubjpass, xcomp dep, nmod nmod, dobj dep, dobj nsubj, expl nsubj, xcomp	0.03(5/166|5/166) 0.03(1/32|1/39) 0.03(1/32|1/16) 0.01(2/16|2/172) 0.01(1/172|1/108) 0.01(1/16|1/108) 0.01(1/136|1/3) 0.01(1/136|1/6)

对于表 3-1 中的依赖结构“case”，尽管属于 object-like，但是大多数实例不能生成语义关系，且其他能够生成语义关系的实例都可以被依赖结构“nmod”替代。①“case”有 156 个实例属于 object-like，但是所有这些实例都不能生成语义关系，也就是说没有另一个依赖结构可以和这些实例进行组合，并且这些实例可以被“nmod”替代，而“nmod”可以生成语义关系，如表 3-3 依赖结构“case”的实例分布情况中的语义关系“R<cities, in, New_Jersey>”可以由依赖结构“nmod:in(cities, New_Jersey)”生成，而不能由依赖结构“case(New_Jersey, in)”和其他的依赖结构通过组合生成。②虽然“case”有 5 个实例不能被依赖结构“nmod”所替代，但是这 5 个实例也都不能生成语义关系，如表 3-3 中的依赖结构“case(GMT, by)”，虽然也属于 object-like，却不能生成语义关系<games, by, GMT>。③如表 3-2 所示，有 5 个语义关联规则“case, case⇒R”的实例，但是所有这些实例都可以被依赖结构“nmod”替代，例如表 3-3 中的语义关系“R<cinemas, in, Netherlands>”既可以通过依赖结构组合“case(cinemas, in) & case(Netherlands, in)”生成，也可以通过依赖结构“nmod:in(cinemas, Netherlands)”生成。

对于表 3-1 中的依赖结构“iobj”，只有一个实例属于“iobj”，纯属偶然。这是因为有 41 个查询都是类似于“Give me…”，使得大部分实例都是“iobj(Give, me)”。换句话说，

除了实例“iobj(Give, me)”“iobj”的所有其他实例都属于 object-like。

表 3-1 中的依赖结构“nmod”是语义关系识别最显著的发现。一方面，如表 3-1 中 triple-like 所示，89 个语义关系可以直接通过依赖结构“nmod”生成。另一方面，如表 3-2 所示，50 个语义关系可以通过组合依赖结构“nmod”和其他的依赖结构如“nsubj,nsubjpass,acl”等组合得到。

表 3-3　依赖结构“case”的实例分布情况

	case	nmod	语义关系	个数
object-like	case(New_Jersey, in)	nmod:in(cities, New_Jersey)	<cities, in, New_Jersey>	156
	case(GMT, by)	nothing	<games, by, GMT>	5
case, case⇒R	case(cinemas, in)& case(Netherlands, in)	nmod:in(cinemas, Netherlands)	<cinemas, in, Netherlands>	5

3.7　语义关系识别

3.7.1　语义关系识别的算法

语义关联规则只表达了依赖结构和语义关系之间的关联，依赖结构的组合方式和顺序还需要进一步确定，才能生成语义关系。第一，依赖结构“subj, nsubj, nsubjpass, csubj, csubjpass, xsubj”和“obj, pobj, dobj, iobj”分别属于 subject-like 和 object-like。第二，从表 3-1 的结果可知，依赖结构“acl”和“dep/advmod”分别属于 subject-like 和 object-like。第三，依赖结构“nmod”不仅属于 object-like 而且属于 triple-like。因此，将所有的依赖结构分为三类，三类依赖结构集如表 3-4 所示。

表 3-4　三类依赖结构集

分类	依赖结构集
$\delta_{\text{subject-like}}$	subj, nsubj, nsubjpass, csubj, csubjpass, xsubj, acl
$\delta_{\text{object-like}}$	obj, pobj, dobj, iobj, dep, advmod
$\delta_{\text{triple-like}}$	nmod

基于上述三类依赖结构集，确定了生成语义关系时依赖结构的组合顺序。首先，组合 $\delta_{\text{subject-like}}$ 和 $\delta_{\text{object-like}}$ 中的依赖结构。然后，组合 $\delta_{\text{subject-like}}$ 和 $\delta_{\text{object-like}}$（即“nmod”）中的依赖结构，因为依赖结构“nmod”的一些实例属于 object-like。最后，将 $\delta_{\text{triple-like}}$ 中依赖结构“nmod”剩余的实例生成语义关系。详细的组合规则如下：

$$R(s,p,o)= f(\delta_{\text{subject-like}} \wedge \delta_{\text{object-like}}) \tag{3.7}$$

$$R(s,p,o)= f(\delta_{\text{subject-like}} \wedge \delta_{\text{triple-like}}) \tag{3.8}$$

$$R(s,p,o)= f(\delta_{\text{triple-like}}) \tag{3.9}$$

此外，语义关系中的一个短语可能包含不仅一个单词，例如实体“Viking Press”和关

系短语“official_langauge”，需要通过依赖结构“compound”将多个单词组合一起形成一个短语。综上所述，提出了语义关系识别的算法，即算法 SRR（Semantic Relation Recognition）。算法思路如下：首先，使用 Stanford Parser 从自然语言查询中生成依赖结构集（step1）。其次，通过依赖结构“compound”组合包含多个单词的实体、关系短语、文本，并更新依赖结构集（step2～4）。然后，从依赖结构集中获得用于生成语义关系的表 3-4 中的三类依赖结构集（step5～7）。最后，依据语义关系的组合规则生成自然语言查询的语义关系（step8～10）。

Algorithm SRR(Semantic Relation Recognition)

Require: **Input**: Natural language question N
Output: the set of semantic relations δ_R

step1: δ_{dep}=Stanford_Parser(N)
step2: $\delta_{compound}$= Get_dependency_structures(δ_{dep})
step3: $\delta_{compound\_phrase}$=Combine_words($\delta_{compound}$)
step4: δ_{dep}=Update_dependency_structures($\delta_{compound\_phrase}$)
step5: $\delta_{subject\text{-}like}$= Get_dependency_structures(δ_{dep})
step6: $\delta_{object\text{-}like}$= Get_dependency_structures(δ_{dep})
step7: $\delta_{triple\text{-}like}$= Get_dependency_structures(δ_{dep})
step8: δ_R=Combine($\delta_{subject\text{-}like}$, $\delta_{oject\text{-}like}$)
step9: $\delta_R = \delta_R$ +Combine($\delta_{subject\text{-}like}$ +$\delta_{triple\text{-}like}$)
step10: $\delta_R = \delta_R$ +Combine($\delta_{triple\text{-}like}$)

3.7.2　算法 SRR 的复杂度

为了分析复杂度，本小节假设一个查询中的语义关系和依赖结构个数最多为 R 和 D。

由文献[247,46]可知，Stanford Parser 从一个查询中生成依赖结构的时间复杂度为 $O(R^3)$。在 step2～4 中，依赖结构的获取和更新需要对所有依赖结构进行两次扫描，且需要组合的词组数量接近 3 个，则时间复杂度为 $O(3 \cdot D \cdot 2) = O(D)$。在 step5～7 中，依赖结构的分类只需要扫描一次所有的依赖结构，则时间复杂度为 $O(D)$。在 step8～10 中，在最坏的情况下，在 step8 和 step9 中所有依赖结构两两组合生成语义关系，时间复杂度为 $O(D \cdot D \cdot 2)=O(D^2)$，在 step10 中所有依赖结构生成语义关系时间复杂度为 $O(D)$。综上所述，算法 SRR 的时间复杂度为 $O(R^3)+O(D)+O(D)+O(D^2)+O(D)=O(R^3+D^2)$。

存储原始依赖结构、已分类的依赖结构、所有语义关系的空间复杂度分别为 $O(D)$、$O(D)$、$O(R)$。因此算法 SRR 的空间复杂度为 $O(D)+O(D)+O(R)=O(D+R)$。

3.7.3　实验结果及分析——语义关系识别

将本章算法与其他试图提高语义关系识别的算法进行比较，如表 3-5 所示，标准查询数据集来自 QALD-3 问题集（由于目前发表的最好的几个问答算法都在 QALD-3 问题集上执行，本章使用 QALD-3 问题集可以很好地与现有方法进行对比），总共包含 198 个自然语言查询。算法 SRR 可以识别 256 个语义关系中的 221 个语义关系，并且 176

个查询中的所有语义关系都可以被识别。很明显，算法 SRR 优于其他方法[36,37,38,39,46,47]，因为现有的方法依赖人为假设的规则，而算法 SRR 依赖于从数据中挖掘出来的规则。

表 3-5　语义关系识别算法对比

算法	语义关系（个）	查询（个）	核心思路
SRR	221(256)	176(198)	依靠从数据中挖掘得到的识别规则
DEANNA[38]	68	46	依靠动词短语和名词短语
Zou 等人[46]	109	69	依靠动词短语、名词短语、基于依赖树的算法
Liu 等人[47]	126	96	依靠特定依赖结构，如 nsubj、nmod:of、amod 等

此外，表 3-2 中有 139 个频繁的依赖结构组合，表 3-1 中有依赖结构“nmod”的 89 个实例属于 triple-like，似乎可以产生 228 个语义关系，但算法 SRR 只在表 5 中产生了 221 个语义关系，这是因为有些语义关系可以从两种方式生成。

例 3.6　对于查询“Give me all people that were born in Vienna and died in Berlin.”，存在依赖结构“nsubjpass(born-7, people-4)/acl:relcl(people-4, born-7)”和“nmod:in(born-7, Vienna-9)”。因此，两种依赖结构组合“nsubjpass, nmod”和“acl, nmod”都可以生成语义关系 *R*<people, born, Vienna> 如图 3-22 所示。同理，语义关系 *R*<people, died, Berlin> 也可以通过两种组合方式得到。

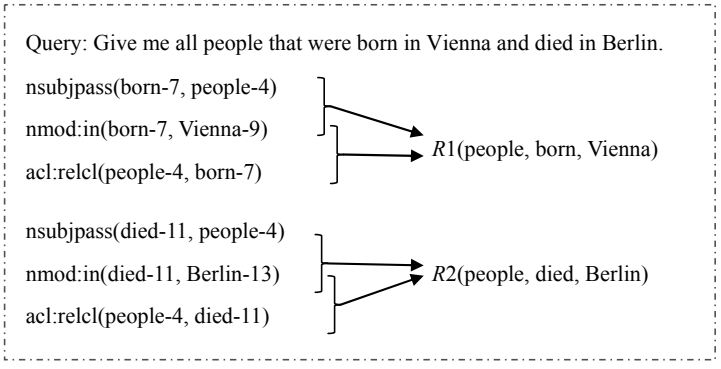
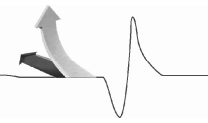


图 3-22　两种依赖结构组合生成语义关系

3.8　本章小结

新的众包模型（即带反馈的并行式迭代模型）继承了迭代模型的完整性和并行模型的准确性，提高了结果的质量并节省了人力资源。对于迭代模型，*n* 个众包工作者连续执行相同的任务，后面的众包工作者可以持续探索新的结果，因此迭代模型的结果具有很好的完整性。对于并行模型，*n* 个众包工作者独立完成同一个任务，众包工作者之间不会相互影响，最终的结果来源于众包工作者的共识，因此并行模型的结果具有良好的准确率。带反馈的并行式迭代模型优于现有的并行模型和迭代模型，有两方面的优势。一方面，新的众包模型是迭代模型和并行模型的融合，模型中的迭代部分可以让众包工作者不断探索新



的结果，保证了结果的完整性，并且模型中的并行部分可以形成大量的共识结果，保证了结果的准确性，因此可以生成更完整的解决方案并不断提高结果的准确性。另一方面，众包工作者总是对自己的错误更敏感，而不是对其他众包工作者的错误更敏感，因此唯一的一组众包工作者在带反馈的并行式迭代模型中迭代地执行相同的任务，不仅可以改善众包工作者的执行能力从而提高结果质量，而且可以节省人力资源。

本章提出了从依赖结构和语义关系之间的关联中挖掘语义关联规则的算法 MSAR。首先，依靠 Stanford Parser 从自然语言查询中生成依赖结构数据集，并依靠新的众包模型（即带反馈的并行式迭代模型）从自然语言查询中生成语义关系数据集。然后，根据关联分析的核心思想和语义关联规则的特点，提出了一种从依赖结构数据集和语义关系数据集中挖掘语义关联规则的算法。最后得到了许多有意义的语义关联规则，例如依赖结构和语义关系之间的关联规则（即 subject-like⇒R, object-like⇒R, triple-like⇒R），以及依赖结构组合和语义关系之间的关联规则，如“nsubj,nmod⇒R”等。

基于挖掘得到的语义关联规则，提出了一种面向知识图谱的自然语言查询的语义关系识别算法，实验结果表明，该方法优于现有方法。首先，该算法通过将 subject-like 中的依赖结构与 object-like 中的依赖结构组合得到语义关系。然后，通过将 subject-like（或 object-like）中的依赖结构和 triple-like 中的依赖结构（即依赖结构“nmod”）组合生成语义关系，因为“nmod”的一些实例属于 object-like。最后，将 triple-like 中剩余的依赖结构生成语义关系。由于语义关系识别的规则是从数据中挖掘出来的，而现有方法中的规则来自人为假设（具有很强的主观性），所以对于知识图谱的自然语言查询，算法 SRR 可以从自然语言查询中识别比现有方法更多的语义关系。

综上所述，本章提出了面向知识图谱的自然语言查询的基于众包的语义关系识别方法，该方法依靠 Stanford Parser 生成依赖结构数据集，依靠本章提出的众包模型（即带反馈的并行式迭代模型）生成语义关系数据集，然后从两个数据集中挖掘语义关联规则，最后依靠挖掘出的语义关联规则识别自然语言查询中的语义关系。与现有方法中来自人为假设的语义关系识别规则相比，算法 SRR 的规则来自数据，避免了人为假设的主观性，因此更加可靠，且可以识别自然语言查询中的大部分语义关系。