arXiv:2303.00595v2 [cs.AI] 8 Aug 2023

# 知识图谱通用问答平台

Reham Omar*、Ishika Dhall*、Panos Kalnis§、Essam Mansour*
*加拿大康考迪亚大学 §沙特阿拉伯 KAUST
{fname}.{lname}@concordia.ca panos.kalnis@kaust.edu.sa

## 摘要

来自不同应用领域的知识被组织为知识图谱（KG），并存储在 RDF 引擎中，可通过 SPARQL 端点在 Web 上访问。要编写格式正确的 SPARQL 查询，需要了解图的结构及其各组成部分的确切 URI，这对普通用户而言并不实际。问答系统通过将自然语言问题转换为 SPARQL 来提供帮助。现有问答系统通常依赖针对特定应用人工整理的规则，或要求针对每个目标知识图谱提供先验信息、进行昂贵的预处理和模型适配。因此，它们难以推广到广泛的应用和知识图谱。本文提出 KGQAn，这是一种通用问答系统，无需针对每个目标知识图谱进行定制。KGQAn 不采用人工整理的规则，而是将问题理解创新性地形式化为文本生成问题，通过神经序列到序列模型将问题转换为中间抽象表示。我们还开发了一种即时链接器，在查询时利用 RDF 存储公开可访问的 API 和现有索引，将抽象表示映射为特定知识图谱的 SPARQL 查询，且无需任何预处理。我们针对多个真实知识图谱开展实验，结果表明 KGQAn 易于部署，且在答案质量和处理时间方面大幅超越最先进水平，尤其适用于训练期间未见过的任意知识图谱。

## 关键词

自然语言问答、知识图谱、RDF、序列到序列模型、即时实体与关系链接

**ACM 引用格式：**
Reham Omar*, Ishika Dhall*, Panos Kalnis§, Essam Mansour*. 2023. 面向知识图谱的通用问答平台。载于*2023* 年国际数据管理会议论文集（*SIGMOD '23*）。ACM，纽约州纽约市，美国，15 页。
https://doi.org/10.1145/3588911

## 1 引言

知识图谱（知识图谱）在各个应用领域中的可用性大幅提升。例如，有关科学出版物的知识图谱，如 Microsoft Academic Graph1，DBLP[2] 和 OpenCitations[3]；通用事实知识图谱，例如 DBpedia4、维基数据[5] 和 YAGO[6]；与生物科学相关的知识（例如 Bio2RDF[7]）；或政治知识（例如英国议会[8] 知识图谱）。由于 RDF 模型简单、SPARQL 查询语言功能强大，并且支持推理和推理扩展 [5, 24], ，许多知识图谱都存储在 RDF 数据管理系统 [1, 2] 中，可通过网络上的 SPARQL 端点远程访问。

考虑 $q^E$ = “丹麦海峡流入哪片海，且加里宁格勒是其岸边城市之一？ ”，这是热门 LC-QuAD 1.0 [52]基准测试中的一道问题。直观来看，$q^E$ 对应一个由两个三元组构成的合取式 SPARQL 查询：?sea，流入，丹麦海峡 和 ?sea，岸边城市，加里宁格勒。实际上，假设端点是 DBpedia，该查询应按图 1 所示的方式表达。请注意， $q^E$ 中的元素（例如“岸边城市”）与实际 SPARQL 组件（即 <http://dbpedia.org/ontology/最近城市>）之间的对应关系并不直观。

普通用户很难了解知识图谱的模式，也难以掌握实体和谓词的确切 URI，从而编写恰当的 SPARQL 查询。因此，近期研究 [27, 28, 31, 64]探讨了将自然语言问题自动转换为 SPARQL；这一问题称为问答（问答）。现有问答系统采用基于语义解析和人工整理规则的问题理解方法，并针对特定知识图谱量身定制；因此，它们难以推广到更广泛的应用领域。此外，现有问答系统依赖每个目标知识图谱的先验知识，将问题中的短语关联到知识图谱顶点和谓词。例如，gAnswer [27, 64] 和 EDGQA [28] 会执行开销高昂的预处理，为每个知识图谱生成索引，而 NSQA [31] 则在目标知识图谱上训练专用的深度神经网络。这带来了若干缺点：（*i*）预处理的计算成本很高，尤其是对于大型知识图谱；（*ii*）只有其所有者选择安装并持续维护问答系统的 SPARQL 端点才提供问答

```
1 前缀 dbv:<http://dbpedia.org/resource/>2 选择 ?海
WHERE { 3 ?海 <http://dbpedia.org/property/流出>
dbv:丹麦_海峡 . 4 ?海 <http://dbpedia.org/ontology/最
近城市>dbv:加里宁格勒 . }
```

**图 1：问题 $q^E$的 SPARQL 查询。“岸边城市”映射到 <http://dbpedia.org/ontology/nearestCity>。**

[1]https://makg.org/

[2]https://dblp.org/β
https://opencitations.net/sparql[4]
https://dbpedia.org/sparql[5]
https://query.wikidata.org/[6]
https://yago-knowledge.org/sparql[7]
https://bio2rdf.org/sparql[8]
https://api.parliament.uk/sparql

授予许可，允许为个人或课堂使用而免费制作本作品的全部或部分数字版或纸质副本，但前提是不得为营利或商业利益制作或分发副本，且副本首页须包含此声明和完整引文。对于本作品中由 ACM 以外的其他方拥有版权的内容，必须尊重其版权。允许在注明出处的情况下摘录。若要以其他方式复制、再版、发布到服务器或向列表重新分发，须事先获得具体许可和/或支付费用。请通过 permissions@acm.org 申请许可。*SIGMOD '23*，*2023* 年 *6* 月 *18* 日至 *23* 日，西雅图，华盛顿州© 2023 计算机协会。ACM ISBN 978-1-4503-XXXX-X/18/06...$15.00 https://doi.org/10.1145/3588911

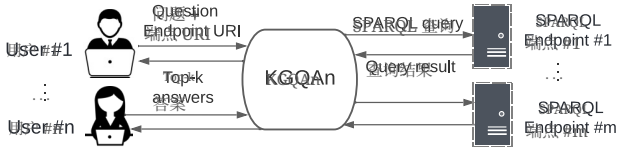



**图 2：用户可随时与 KGQAn 交互，针对任意知识图谱提交自然语言问题。问题理解独立于知识图谱。实际 SPARQL 查询的构建以即时方式，通过目标知识图谱公开访问的 API 完成。**

；以及（*iii*）难以推广到任意知识图谱。

与现有系统不同，我们受到网络搜索引擎的启发，这类搜索引擎能够独立于任何特定网站解答用户的问题。我们提出了一种名为 KGQAn[9] 的新方法，不再需要针对每个知识图谱单独定制问答系统。如图 2 所示，KGQAn 位于用户与任何可用知识图谱之间。KGQAn 为问题理解、链接以及最终答案的过滤 引入了新方法。我们将问题理解形式化为一项文本生成任务，从自然语言问题中提取抽象三元组模式。在部署之前，我们使用多样化的问题集，通过序列到序列（Seq2Seq） [50] 深度神经网络训练 KGQAn。由此得到的模型以问题为输入，提取一组三元组模式，用以表示实体之间的关系；这些实体要么在问题中明确提及，要么是未知项。利用这些三元组模式，我们构建所谓的短语图模式（*PGP*），它是系统对问题理解的正式抽象表示，且独立于任何知识图谱。

我们还提出了一种即时（即时）链接器，它会与目标知识图谱协商，确定 PGP 抽象元素与适当 URI 之间的映射，从而实例化格式正确的 SPARQL 查询。KGQAn 通过公开可访问的 SPARQL 端点 API 提交请求，并利用所有常见 RDF 存储库中都存在的内置索引 [44]（例如 Virtuoso、Stardog 和 Apache Jena）。链接过程通过基于通用英语文本训练的语义亲和模型透明地完成，无需知识图谱所有者进行任何特定配置或操作。SPARQL 查询会提交至知识图谱，KGQAn 收集并过滤答案后，再将其返回给用户。过滤依据包括预期的领域或答案的数据类型等约束；这些约束在构建 PGP 时确定。与现有问答系统不同，我们的过滤方法不需要关于知识图谱的任何先验知识。

我们在多种真实知识图谱上评估了 KGQAn。除了两个热门问答基准 LC- QuAD 1.0 [52] 和 QALD-9 [54], 所使用的 DBpedia 之外，我们还在 YAGO、Microsoft AcademicGraph 和 DBLP 上进行了实验。在答案质量（即宏平均 F1 分数）和问题处理时间方面，实验结果证实，当查询在训练期间未见过的任意知识图谱上执行时，KGQAn 的表现远胜现有问答系统。有趣的是，尽管 KGQAn 具有通用性，并不针对任何特定知识图谱，但它的表现几乎与当前最先进水平（最先进水平）相当，有时甚至更胜一筹；这里的基准测试是 LC-QuAD 1.0 和 QALD-9。需要注意的是，当前最先进水平的系统是专门针对这些基准测试进行优化的。

总而言之，我们的贡献如下：

- KGQAn 是首个通用问答系统，能够针对任意此前未见过的知识图谱处理即时提出的问题。
- 我们将问题理解新颖地形式化为文本生成问题。我们训练序列到序列神经网络，使其学会从自然语言问题中提取正式的抽象表示，且不受应用领域限制，也无需任何人工整理的规则。
- 一种新颖的即时链接与过滤方法，利用通用语义亲和模型和 SPARQL 端点的公共 API。我们的方法无需关于知识图谱的先验知识，旨在先最大限度地提高 SPARQL 查询的召回率，再通过过滤提升精确率。
- 我们使用来自不同领域的四个真实知识图谱进行了全面评估。对于已知基准测试，KGQAn 的表现与最先进水平相近；对于此前未见过的知识图谱，其表现则大幅超越最先进水平。

本文结构如下：第 2 节阐明必要的背景知识以及现有系统的局限性。第 3、4、5 和 6 节介绍 KGQAn 的架构，以及我们的问题理解、链接和过滤方法。第 7 节包含实验评估。第 8 节讨论相关工作，第 9 节总结全文。

## 2 背景与局限

问答系统通常将问答过程分为三个步骤：(*i*) 问题理解：从问题中提取实体和关系，并生成抽象表示；(*ii*)链接：将抽象表示映射到目标知识图谱中对应的顶点和谓词，以构造格式正确的 SPARQL 查询；以及 (*iii*) 过滤：根据某些约束条件选择最相关的答案。本节将讨论三个现有的问答系统：gAnswer [27, 64], 、EDGQA [28] 和 NSQA [31], , 它们总结于表 1。这些系统代表了最成功的方法，在 LC-QuAD 1.0 和 QALD-9 基准测试上的准确率（即 F1 分数）方面达到了最先进水平。

### 2.1 问题理解

问题理解步骤会识别自然语言问题中提及的实体、未知项和关系，同时生成它们相互依赖关系的抽象形式化表示（通常为图）。图 3 展示了运行示例 $q^E$的预期结果。

**gAnswer** 使用斯坦福依存句法分析器 [12] 将问题映射为句法依存树。随后遍历该树，生成查询的抽象图表示，称为语义查询图。此过程通过一组静态启发式语言规则实现，这些规则是根据 QALD-9 训练问题集整理而成的；例如，假定以“wh*”开头的问题词（如“what”）表示未知项。gAnswer 还依赖一组预定义的同义词 [41]。对于我们的运行示例，gAnswer 生成了许多不必要且错误的三元组，例如 加里宁格勒，城市所属，城市。

[9] KGQAn: 知识图谱问答平台。

**表 1：在 QALD-9 和/或 LC-QuAD 1.0 中取得最高 F1 分数的基于 SPARQL 的问答系统对比分析**

| | 问题理解 | 链接 | 过滤 |
|---|---|---|---|
| gAnswer [27, 64] | 依存句法分析 | 索引查找 | n/a |
| EDGQA [28] | 成分句法分析 | Dexter、EARL、Falcon | 按索引类型 |
| NSQA [31] | AMR解析，基于BERT | BLINK、SEM-Rel | n/a |
| **KGQAn（我们的方法）** | Seq2Seq文本生成 | RDF 引擎 + 语义相似度 | 按预测答案类型 |

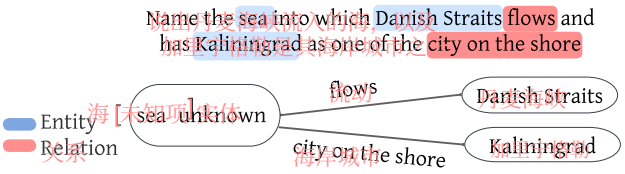



**图 3：问题理解 $q^E$ 将丹麦海峡和加里宁格勒识别为提及的实体；将海识别为未知项，将流动和海岸城市识别为关系，最终得到一个包含两个三元组模式的图。**

**EDGQA** 使用斯坦福核心自然语言处理解析器 [10] 构建问题的成分解析树。然后，通过迭代分解解析树，生成一个有根无环图，称为实体描述图。该过程基于针对 LC-QuAD 1.0 和 QALD-9 基准测试量身定制的规则。对于我们的运行示例，EDGQA 错误地将 [20] 丹麦海峡识别为关系，将流量识别为实体，并生成了以下三元组模式：未知项，丹麦海峡，流量。

**NSQA** 将问题解析为一个有向有根无环图，称为抽象意义表示。解析基于深度神经符号框架。该神经网络使用从 Propbank 收集的面向动词的关系进行训练。训练集还包含一组针对不同问题整理的树。这些树的多样性决定了模型的准确率。目前尚不清楚[10] 该模型能否仅根据名词短语识别关系，例如“海岸城市”。生成的图与 SPARQL 的粒度不同；因此，必须对模型进行调整 [31]，使其适用于新的领域。

## 2.2 实体与关系链接

在问答系统中，链接是一项具有挑战性的任务 [33]，旨在将从问题中提取的实体和关系映射到知识图谱中相应的顶点和谓词。以 DBpedia 上的运行示例$q^E$ 为例，预期输出为：

| 短语 | URI |
|---|---|
| 丹麦海峡 | <http://dbpedia.org/resource/Danish_海峡> |
| 加里宁格勒 | <http://dbpedia.org/resource/Kaliningrad> |
| 流动 | <http://dbpedia.org/property/流出> |
| 海岸城市 | <http://dbpedia.org/ontology/nearestCity> |

现有系统将链接形式化为：(*i*) 在倒排索引中查找，该索引将同义词映射到目标知识图谱的相关顶点和谓词；或 (*ii*) 在由带标签文档构成的数据库中进行关键词搜索，这些文档由目标知识图谱的顶点和谓词构建；或 (*iii*) 基于深度学习的转换，该转换定义在目标知识图谱的向量空间上。这些方法因知识图谱而异，且需要针对每个目标知识图谱进行成本高昂的预处理。

**gAnswer** 通过从网络爬取构建的 crossWikis 词典 [49]，链接实体。词典中的每个实体都映射到目标知识图谱中的相应顶点，并附有置信度。对于关系链接，gAnswer 会构建一个词典，将关系提及映射到目标知识图谱中相应的谓词。例如，在某个特定知识图谱中，关系“由……主演”会映射到谓词 starring。该映射通过通用的自然语言处理 [35]方法生成，或通过使用 n 元语法 [21]生成。

**EDGQA**利用三个索引系统，即 Falcon [47]、EARL [18] 和 Dexter [6]，，对目标知识图谱的顶点和谓词进行索引。EDGQA 综合这三个系统的结果，将实体链接到知识图谱顶点；然后利用这些结果缩小关系链接 [40]的搜索空间；最后使用基于BERT的 [13] 模型对谓词列表进行排序。上述索引系统通过提取顶点及其相关描述来处理目标知识图谱。例如，Falcon 使用标准谓词 #label 和 #altLabel 来收集描述和同义词。随后，Falcon 执行词性标注和 n 元语法提取，生成带标签文档，以便在链接过程中进行关键词搜索。如果知识图谱不包含 # label 谓词，则必须为每种顶点类型手动选择合适的索引谓词，例如，可以选择 authorName 对作者类型的顶点进行索引。

**NSQA** 略有不同，因为它在链接过程中识别实体和关系。首先，它使用 Blink [61] 在向量空间中生成目标知识图谱顶点的嵌入。然后，它接收上一阶段生成的抽象图，并通过一个基于 BERT 的神经网络模型检测实体。该模型使用 LC-QuAD 1.0 中的 3,651 个问题进行训练，这些问题带有人工标注的提及。检测到的实体被嵌入到与知识图谱相同的空间中，并链接到其最近邻顶点。对于关系链接，NSQA 使用 SemRel [36]，，这是一种基于 Transformer 的神经网络模型，经过训练可预测关系。该模型会从知识图谱中的关系集合中返回一系列候选关系，并为它们排序。在 NSQA 中，构建向量空间，也就是为目标知识图谱顶点和谓词生成嵌入，是预处理阶段的主要成本所在。

## 2.3 过滤

一些问答系统（如 EDGQA）包含一个由 RDF 引擎过滤潜在答案的步骤。对于过滤步骤，现有系统会在预处理阶段，为知识图谱中的所有节点类型/类构建先验知识。过滤的做法是将未知项（例如 ?sea）链接到目标知识图谱中的某个顶点类型，例如 ?sea, a, dbo:Sea，前提是存在这样的类型。SPARQL 查询

[10]NSQA 是 IBM 的专有项目，我们无法使用它来复现结果或测试运行示例 $q^E$；我们的讨论基于来自 [31] 的信息。

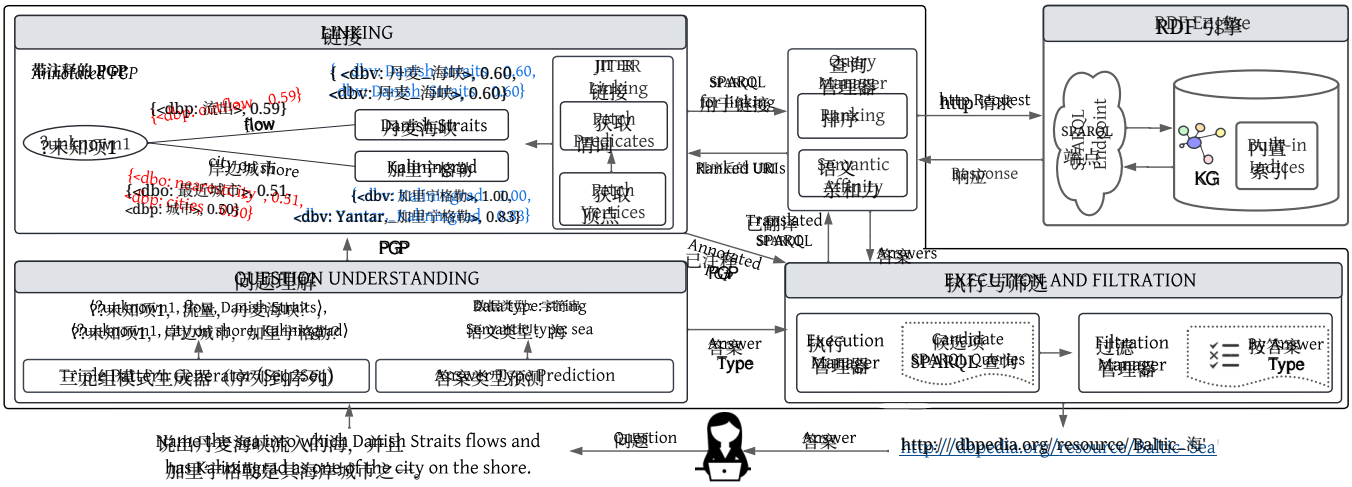



**图4：KGQAn 的架构包含三个阶段：a）问题理解：使用我们的三元组模式生成器（序列到序列模型）构建 PGP；b）将 PGP 链接到知识图谱：即时实体解析链接模块使用知识图谱中的相关顶点和谓词标注 PGP；c）执行和过滤：创建并执行 SPARQL 查询，然后对查询结果进行后置过滤。**

经过类型约束修订后，会被发送执行；因此，收到的答案已预先过滤。

## 3 KGQAN 架构

与现有方法类似，KGQAn 实现了三个不同的阶段：问题理解、链接以及执行与过滤，如图4所示。然而，与现有方法不同，后者需要先验知识，并且必须针对每个目标知识图谱进行昂贵的预处理；KGQAn 则是一种通用解决方案，适用于来自各种应用领域的任意知识图谱。

**问题理解**。 KGQAn 将问题理解形式化为一项文本生成任务。我们采用预训练的序列到序列语言模型，例如 BART [32]或 GPT-3 [4]。原始模型能够理解通用英语文本；我们通过一组典型问题及其对应的抽象三元组模式翻译结果，对其进行补充训练。训练过程独立于任何知识图谱。因此，该模型能够接收通用英语问题，并生成一系列抽象三元组模式。图4展示了，对于我们的运行示例 $q^E$，序列到序列模型生成了两个三元组：?unknown1，流量，丹麦海峡 和 ?unknown1，岸边城市，加里宁格勒。所有三元组构成一个抽象图表示，称为短语图模式（*PGP*）。

KGQAn 还会预测答案的预期数据类型和语义类型。我们训练了一个三层神经网络，其输入是英语问题，输出是答案的预期数据类型，即日期、数值型、布尔型和字符串。对于字符串这种情况，我们还会根据上下文预测语义类型。对于示例问题，预测的数据类型是字符串，而预测的语义类型是“海”。

**链接**。 直观而言，链接对应于抽象 PGP 图中的元素与目标知识图谱中的实际数据之间的语义匹配。KGQAn 不掌握目标知识图谱的任何先验信息，因此必须在问答过程中以即时（JIT）方式执行匹配。这样，KGQAn 就能支持任意知识图谱。由于大多数 RDF 引擎不支持语义搜索，我们将这一过程拆分到 KGQAn 和 RDF 引擎之间。KGQAn 通过标准 SPARQL API，为 PGP 中每个实体关联的短语提交一组文本包含查询；这些查询由 RDF 引擎的内置索引 [44] 处理。KGQAn 根据现有的通用词嵌入模型，为查询结果（即与 PGP 短语可能存在语义匹配的 URI）赋予语义亲和度分数。随后，KGQAn 使用排名前 $k$ 的匹配项，通过标准 SPARQL 查询再次查询目标知识图谱，这一次是为了获取与 PGP 谓词可能存在语义匹配的项。该过程的结果是一个带注释的 PGP。在图4的运行示例中，实体加里宁格勒有两个匹配项：dbv:Kaliningrad，分数为 1。00；以及 dbv:Yantar,_ Kaliningrad，分数为 0。83；而关系流出有一个匹配项 dbp:outflow，分数为 0。59。

**执行与筛选**。KGQAn 遍历带注释的 PGP，为目标知识图谱生成所有格式正确的候选 SPARQL 查询组合，这些查询在语义上可能与用户问题等价。根据语义亲和度分数，最有希望的前 $k$ 个查询会被发送到 RDF 引擎。KGQAn 收集答案，并根据前文所述的预期数据类型和语义类型对其进行筛选。请注意，执行多个候选查询虽然能提高召回率，但筛选步骤能显著提高精确率。与现有系统将特定于知识图谱的过滤器嵌入 SPARQL 查询不同，我们的过滤方法在后处理阶段应用，且独立于知识图谱；因此，该方法适用于任意知识图谱。在我们的运行示例中，用户收到 http://dbpedia.org/resource/Baltic_ Sea 作为最终答案。

## 4 KGQAN问题理解

本节介绍我们将问题理解问题形式化为文本生成任务的新颖方法。给定一个 $q$英语文本中的问题，我们生成一系列三元组模式实体$_i^a$，关系$_i$、实体$_i^b$，用于表示我们对 $q$的形式化理解；这是通过序列到序列深度神经网络实现的。三元组的所有组成部分要么是 $q$中的短语，要么是未知项（即变量）；因此，问题理解独立于任何特定的

领域或知识图谱。未知项尤其是与实体同等重要的一等公民；我们检测它们与提及的实体之间的语义关系，并预测其预期数据和语义类型。需要注意的是，我们的工作与关系三元组抽取 [9, 58] 相关，后者指的是从长文本中构建知识图谱；然而，未知项的存在使我们的任务更具挑战性。

## 4.1 短语三元组模式抽取

英语文本中的问题 $q$由一个词序列（$q_1$、 $q_2$、 $q_3$、……）组成，其中 $q_i$ 是第 $i$个词。我们的任务是生成能够捕捉 $q$ 语义的三元组序列。形式化表示如下：

**定义 4.1**（短语三元组模式抽取）：设 $q$ =（$q_1$、 $q_2$、 $q_3$、……）是一个问题。生成三元组序列 $TP$（$q$） =（$tp_1$、 $tp_2$、……）。每个 $tp_i \in TP$（$q$）的形式为 $tp_i$ =$\mathrm{e}_i^a$、$\mathrm{r}_i$、$\mathrm{e}_i^b$，其中 $e_i^a$和 $e_i^b$要么是 $q$中的实体短语，要么是未知项；$r_i$是 $q$中的关系短语。

*4.1.1* 通过 *Seq2Seq Transformer* 进行三元组抽取。 我们使用Seq2Seq预训练语言模型（*PLM*）对文本生成任务进行建模。输入查询 $q$ 是纯文本。为了与序列到序列模型兼容，输出 $TP$（$q$）也应表示为文本 [50]。设 $y = F_{txt}$（$TP$（$q$））为文本表示。 $F_{txt}$通过将 $tp_i \in TP$（$q$）标注为“EntityA”“EntityB”和“Relation”，分别把每个 $e_i^a$、 $e_i^b$、 $r_i$转换为英语文本。我们的示例问题 $q^E$生成了两个三元组： $TP$（$q^E$） =（?unknown1、flow、丹麦海峡、?unknown1、岸边城市、加里宁格勒），它们对应于以下三元组模式：

[关系(标签="流量"), 实体A(标签="名称", 类别=变量, 变量ID=1), 实体B(标签="丹麦海峡", 类别=实体)] ∧关系(标签="岸边城市"), 实体A(标签="名称", 类别=变量, 变量ID=1), 实体B(标签="加里宁格勒", 类别=实体)]

我们考虑两类预训练语言模型：（$i$）编码器-解码器，例如 BART [32]；以及（$ii$）仅解码器，例如 GPT-3 [4]。这些预训练语言模型支持动态长度序列生成 [53]。图5展示了我们使用编码器-解码器预训练语言模型进行训练的过程。编码器接收 $q$作为输入，并应用自注意力机制，生成 $q$的向量表示（嵌入）。编码器嵌入 $F^e$（$q$）封装了 $q$的各种特征，并突出其最重要的词元。解码器学习将三元组模式生成为词元序列，其中对词元 $i$的预测基于 $F^e$（$q$）以及所有先前的词元。解码器使用两种注意力机制：（$i$）对其输入 $y$应用自注意力，以及（$ii$）在 $F^e(q)$ 与 $y$ [56] 之间应用交叉注意力。如果使用仅解码器预训练语言模型，则将 $q$直接作为输入提供给解码器，并通过仅对 $q$ [4]应用自注意力机制来训练解码器生成$y$。

*4.1.2* 用于三元组模式抽取的训练数据集。 序列到序列模型可以是 BART 或 GPT-3，预先使用通用英语文本进行训练。为了将其用于问题理解，我们必须针对三元组抽取进行额外训练，如图5所示。为此，我们使用从 LC- QuAD 1.0 [52] 和 QALD-9 [54]的训练数据集中收集的 1,752 个问题，准备了一个经过人工标注的训练数据集。每个问题 $q$均标注了

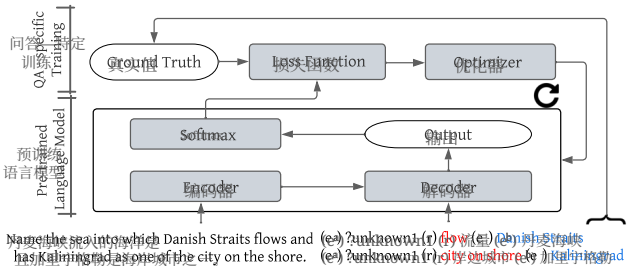



**图5：以问题和三元组模式序列分别作为编码器和解码器的输入，训练我们的序列到序列模型。训练基于1,752个手动标注的问题。**

适当的三元组集合 $TP(q)$。对于每个 $q$，标注过程的主要步骤如下：

- 识别短语三元组模式的数量 $|TP(q)|$。
- 检测共享同一未知项的三元组模式。
- 为每个未知项分配唯一标识符。
- 对于每个 $\mathrm{e}_i^a$， $\mathrm{r}_i$, $\mathrm{e}_i^b$， $\in TP(q)$，从 $q$中分别提取与 $e_i^a$、$e_i^b$和 $r_i$对应的实体短语和关系短语。

我们设计训练数据集，使其包含具有不同数量未知项和三元组模式的问题。在标注中，我们假设存在一个主要未知项[11]（即意图），需要为其提供答案。也可能存在其他未知项，但它们被视为中间变量。区分主要未知项和中间未知项有助于 KGQAn 在执行与过滤阶段进行处理。例如，中间变量可以用于生成的 SPARQL 查询中的 WHERE 子句，但不能用于 SELECT 子句。

我们考虑实体和关系在句法和语义表达上的各种情况。例如，实体可以是命名实体，如“丹麦海峡”；也可以是实体提及，如“首都地区”。关系可以是动词，如“流动”；可以是带副词的动词，如“解决”；也可以是名词短语，如“岸边城市”。我们的数据集使序列到序列模型能够学习如何区分实体短语和关系短语，而无需进行额外的语义解析或词性标注。

未来可以通过补充更多具有代表性的案例来扩展我们的数据集。不过，它已经足够通用，能够让序列到序列模型从各种领域中提取三元组。例如，尽管我们的训练使用了来自 DBpedia 的一般事实，且没有任何关于出版物的问题，但如实验评估所示，该模型仍能理解与 DBLP 相关的问题。还需注意，标注数据集不包含实际的 SPARQL 查询，因为这取决于目标知识图谱，而目标知识图谱在训练期间尚不可知。

## 4.2 短语图模式（PGP）

设 $q$为一个问题， $TP(q)$为相应的三元组模式集合。我们将 $q$的问题短语图模式定义为：

[11]QALD-9 和 LC-QuAD 1.0 基准测试都不包含具有两个意图的问题。未来，我们计划扩展 KGQAn，使其能够支持具有两个意图的问题，例如：新冠肺炎何时、何地开始传播?

**定义 4.2**（短语图模式（PGP））。设 E、R 分别为节点和边的集合。$q$ 的短语图模式是一个无向图 $PGP(q) = (\mathrm{E}, \mathrm{R})$，满足：对于每个三元组 实体$_i^a$、关系$_i$、实体$_i^b \in TP(q)$，存在一个节点 $e_i^a \in \mathrm{E}$，其标签为实体$_i^a$；存在一个节点 $e_i^b \in \mathrm{E}$，其标签为实体$_i^b$；并且存在一条边 $(e_i^a, e_i^b) \in \mathrm{R}$，其标签为关系$_i$。

直观而言，$PGP(q)$ 将 $TP(q)$ 中的三元组模式集合连接成一个图，该图表示对问题 $q$ 的形式化理解，如图4所示。请注意，尽管 RDF 数据构成有向图，$PGP(q)$ 却是无向的。原因在于 PGP 并不了解目标知识图谱。因此，此时我们还不知道目标知识图谱中出现了哪些谓词，也不知道它们的方向。根据问题的不同，构建出的 PGP 可能呈现不同形状，例如星形或路径查询图。按照问题的不同类别，[17],KGQAn 的当前实现能够支持多种问题，包括：单事实、带类型的单事实、多事实和布尔问题。

## 4.3 数据与语义类型预测

如上所述，生成的 PGP 包含一个主要未知项。KGQAn 会预测该未知项预期的数据类型和语义类型；这些信息将在查询执行后用于提高答案的精确率。KGQAn 仅使用用户问题 $q$ 来预测数据类型和语义类型，而不考虑目标知识图谱。

预期的数据类型可以是日期、数值型、布尔型或字符串。我们将数据类型预测定义为一个分类任务。我们使用 QALD-9 基准测试中的训练数据集训练深度神经网络，因为其中的问题已经标注了数据类型。如果预期的数据类型是字符串，我们还会预测语义类型（例如人物、书籍等）。我们采用以下启发式方法：问题中的第一个名词就是语义类型。我们使用 AllenNLP 成分句法分析器 [26] 获取问题中的词性标签，并提取标签为名词的第一个短语。该启发式方法只会影响 KGQAn 后置过滤步骤的准确率，不会影响问题理解和链接。对于运行示例 $q^E$，KGQAn 预测的数据类型和语义类型分别为字符串和“海”。

# 5 KGQAN 链接

对于问题 $q$，链接阶段接收问题理解阶段输出的 $PGP$（$q$），并生成图的带注释版本 $AGP$（$q$）。$AGP$（$q$）中的顶点和边会标注目标知识图谱中与 $q$含义在语义上最接近的 URI。由于目标知识图谱无法预先确定，KGQAn 会在查询时以即时方式实现链接，并使用一组轻量级 SPARQL 请求。这样，KGQAn 无需进行任何预处理即可适用于任意知识图谱。

## 5.1 实体链接

令 $l_n$ 为 $PGP(q)$ 中节点 $n$ 的标签。令 $d_v$ 为目标知识图谱 $KG$ 中顶点 $v$ 的描述（即类型为字符串的字面量）。令 $S(l_n, d_v)$ 为 $l_n$ 与 $d_v$ 之间的语义亲和度分数；其定义请参见第 5.4 节 $S(\circ, \circ)$。直观来说，对于每个节点 $n$，我们需要遍历 $KG$ 中的所有顶点 $v$；计算它们与 $n$ 的语义亲和度；并选出排名前 $k$ 的匹配项。这些是添加为标注图 $AGP(q)$ 中 $n$ 上的标注的相关顶点。

这里有两个实际问题：首先，如何提取 $d_v$ $v$ 节点在 $KG$中的描述。有些知识图谱使用人类可读的 URI，并通过 rdf:label 谓词明确标记描述。例如，DBpedia 中戴安娜王妃的条目包含三元组 http://dbpedia.org/resource/Diana,_戴安娜，威尔士王妃__, rdf:label，“Diana, Princess of Wales”。我们可以通过 SPARQL ASK 查询检查 $KG$ 是否包含 rdf:label 谓词；如果包含，就可以直接检索描述。然而，其他知识图谱可能更加晦涩。例如，Microsoft Academic Graph 中 Jim Gray 的条目是 mag:2279569217。幸运的是，其中有一个三元组 https://makg.org/entity/2279569217, foaf:name，“Jim Gray”，我们可以据此检索描述。不过，foaf:name 是一个任意谓词，对于其他实体可能有所不同。因此，通常我们会检索通过任意谓词将 $KG$ 顶点连接到字符串类型字面量的三元组。

第二个问题是，典型的 RDF 引擎不支持语义搜索。因此，函数 $S(\circ, \circ)$ 必须在远程的 KGQAn 站点计算。显然，我们不想将大量（可能不相关的）数据从 $KG$传输到 KGQAn。我们采用以下启发式方法：标签 $l_n$ 由一个词序列组成（例如“丹麦海峡”）。我们仅从 $KG$ 请求那些通过任意谓词 $p$与字面量 $d_v$相连的顶点$v$，且 $d_v$ 包含 $l_n$中任意组合的词。该启发式方法可转换为以下 SPARQL 查询，其中 $Q(l_n)$表示 $l_n$中所有词的析取布尔表达式：

```
查询：potentialRelevantVertices(l_n,
maxVR) 1 SELECTDISTINCT?v?d_v 2
WHERE { 3 ?v ?p?d_v . 4 ?d_v  <bif:contains>
Q(l_n) . } 5 LIMIT maxVR
```

所有现代 RDF 引擎（如 Virtuoso、Stardog 和 Apache Jena）默认都会构建全文索引，以支持文本搜索 [44]；因此，可以高效地执行 potentialRelevantVertices。由于查询仍可能返回大量匹配项，我们采用启发式方法将结果数量限制为 $maxVR$；在实验中，我们将 $maxVR = 400$ 设为相应值。该查询假定使用 Virtuoso 作为 RDF 引擎。其他引擎可能提供略有不同的 API；例如，对于 Stardog，我们将 <bif:contains> 替换为 <stardog:textMatch>。我们将相关顶点集合正式定义如下：

定义 5.1（相关顶点 RV）。给定一个问题 $q$ 和一个目标知识图谱 $KG = (V, P)$，并将其视为三元组集合。设 $n$为 $PGP(q)$ 中一个标签为 $l_n$ 的节点。设 $T_d = \{\mathrm{v}, \mathrm{p}\ d_v : v \in V, p \in P\}$ 为来自 $KG$ 的三元组子集，其中描述 $d_v$是字符串类型的字面量，且 $l_n$全部或部分包含于 $d_v$ 中。设 $T_v = \{v, S(l_n, d_v) : \mathrm{v}, \mathrm{p}\ d_v \in T_d\}$，其中 $S(\circ, \circ)$ 表示语义亲和度分数。对于 $KG$ 中的节点 $n$，其相关顶点集合 RV $(n, KG)$ 是 $T_v$ 的子集，其中包含亲和度分数排名前 $k$ 的所有配对。

请注意，在目标知识图谱上执行的 potentialRelevantVertices 查询是一种启发式方法，能够快速检索出一组规模相对较大但可能不准确的

**算法 1**KGQAn实体链接

**输入：** $n$：$PGP(q)$中的一个节点，$KG$：目标知识图谱，$maxVR$：最大获取顶点数 ，$k$：顶点数
**输出：** $n$，并标注相关的 k 个顶点 $\mathrm{R}\mathcal{V}(n, KG)$

1：**如果** $n.type$ 为“未知项”**则** ▷ $n$ 是一个变量
2: **返回** $n$.RV $\leftarrow \emptyset$
3: **结束 if**
4: $T_d \leftarrow$ potentialRelevantVertices($n.l_n$, $maxVR$) ▷ 将 SPARQL 转换为 $KG$
5: $T_v \leftarrow \emptyset$
6: **对于** 每个 $v$, $d_p \in T_d$**执行**
7: $T_v \leftarrow T_v \cup v, S(n.l_n, d_v)$ ▷ 计算语义亲和度
8: **结束循环**
9: **返回** $n.\mathrm{R}\mathcal{V}$ $\leftarrow$ 中亲和度分数最高的所有配对

顶点。在我们的运行示例中，对于 PGP 节点加里宁格勒 $q_E$，它同时返回了 dbv:Kaliningrad 和 dbv:Yantar，_加里宁格勒（见图4）。在 KGQAn 站点计算的语义亲和度分数将 dbv:Kaliningrad 识别为最佳匹配。

算法 1 概述了针对单个节点 $n$ 的实体链接过程；必须对每个节点 $n \in PGP(q)$ 调用该算法。在第 1 行，我们检查 $n$ 是否为未知项（即变量）；若是，则此阶段不存在相关顶点（第 2 行）。第 4 行在目标 RDF 引擎上执行 potentialRelevantVertices SPARQL 查询。第 5 至 8 行计算每个返回顶点与 $n$ 的语义亲和度，并将所得的 $vertex$，$score$ 配对存储在 $T_v$ 中，如定义 5.1 所述。最后，第 9 行从 $T_v$ 中选出分数最高的配对，以构建相关顶点集合 $\mathrm{R}\mathcal{V}(n, KG)$。

*5.1.1* **复杂度。** 算法 1 执行 $|\mathrm{E}|$ 次，其中 E 是 $PGP(q)$ 中节点的集合；参见定义 5.3。其开销主要由第 4 行的 SPAQRL 查询决定。设 $c_{rdf}$ 为在 RDF 引擎中执行关键词搜索的开销。$c_{rdf}$ 由特定引擎实现的索引和方法决定。例如，Apache Jena 通过 Lucene [30], 或 Elastic search 实现全文搜索。第 6 行的 for 循环执行固定次数，因为它受 SPARQL 查询中的参数 $maxVR$ 限制。第 7 行语义亲和度计算的开销取决于 $|l_n| \cdot |d_v|$。不过，在实际应用中，这两个集合都只包含少量单词，因此其开销可视为常数。第 9 行的 top-$k$ 计算同样受 $maxVR$ 限制，因此也是常数。算法 1 的最终复杂度为 $O(c_{rdf}|\mathrm{E}|)$。

## 5.2 关系链接

令 $p$ 为目标知识图谱 $KG$中的一个谓词。由于 $KG$是有向的，因此 $p$ 必须与某个顶点 $v \in KG$ 相连，形成出边或入边。如果将 $KG$ 表示为三元组，则分别存在三元组 $v, p,$ ?obj 或 ?sub $p, v$。令 $d_p$ 为 $p$的描述（即人类可读的字符串）。在某些知识图谱中，谓词的 URI 包含人类可读文本，例如 DBpedia 中的 dbo:spouse。在这种情况下，我们假设 URI 就是谓词的描述，即$d_p = p$。我们通过两种不同的 SPARQL 查询来获取出边和入边谓词（相对于顶点 $v$）。用于检索出边谓词 $p$ 的 $d_p$ 的 SPARQL 查询如下：

```
查询：outgoingPredicate(v)
1 SELECTDISTINCT?p 2
WHERE{ 3 v ?p ?obj.}
```

查询 incomingPredicate(v) 完全相同，只有第 3 行不同，变为 ?sub ?p v。大多数 RDF 引擎都能高效执行出向和入向查询，因为它们会以六种方式为传统查找建立所有三元组的索引 [59, 63] 。然而，在某些知识图谱中，谓词 URI 可能是任意的内部标识符[12]。例如，URI 为 $p$ = wdg:P227 的谓词，其描述也可以通过向知识图谱 wdg 发送 SPARQL 查询来检索。收到查询结果后，我们会进行检查：如果 ?$p$ 是任意内部标识符，那么我们会再发送一个查询以获取该谓词的描述。

设 $r$ $PGP$ $q$ $n^a$ $n^b$ $\in$ $PGP$ 是（ ）中的一个关系，连接两个节点，($q$)。这些节点已分别链接到相关顶点 $\mathrm{R}\mathcal{V}(n^a, KG)$ 和 $\mathrm{R}\mathcal{V}(n^b, KG)$。我们的目标是链接 $r$。我们方法的直觉是，PGP 中任何成功链接的顶点对 $node$，$relation$ 都必须至少在目标知识图谱中出现一次。因此，我们不再盲目地在 $KG$中搜索谓词，而是将范围限制为那些连接到相关顶点 $\mathrm{R}\mathcal{V}(n^a, KG) \cup \mathrm{R}\mathcal{V}(n^b, KG)$ 的谓词。对于每个这样的顶点 $v$，我们使用 SPARQL 查询 outgoingPredicate(v) 和 incomingPredicate(v) 检索其所有相连的谓词。注意，我们必须检查两个方向，因为 $PGP(q)$ 是无向的。

对于每个检索到的谓词 $p$ 及其对应的描述$d_p$，我们使用公式 1 计算亲和度分数 $S(l_r, d_p)$，其中 $l_r$是 $r$的标签（见定义 5.2）。语义匹配使我们能够将标签“wife”映射到例如 DBpedia 谓词 dbo:spouse。我们根据亲和度分数选取得分最高的前-$k$ 个谓词，作为 $r$的相关谓词集合。形式化地：

定义 5.2（相关谓词 RP）。给定一个问题 $q$和一个目标知识图谱 $KG = (V, P)$，该图谱被视为一个三元组集合。设 $r$是 $PGP(q)$中的一个关系，其标签为 $l_r$；设 $n^a, n^b \in PGP(q)$是与 $r$相连的节点。令$T_{rv} = \left\{v : v, s_v \in \mathrm{RV}(n^a, KG) \bigcup \mathrm{RV}(n^b, KG)\right\}$为 $n^a$和 $n^b$的相关顶点之并。与相关顶点相连的所有出边和入边谓词的集合为 $T_{pd} = \{p, d_p, v, o : v \in T_{vr}, v, p,$ ?obj $\in KG \vee$ ?sub $p, v \in KG\}$；若 $v$是三元组中的宾语，则标志$o$为 $True$；标签 $d_p$按上述方法获取。令 $T_p = \{p, S(l_r, d_p)\ v, o : p, d_p, v, o \in T_{pd}\}$，其中 $S(\circ, \circ)$为语义亲和度分数。关系 $r$在目标 $KG$中的相关谓词 RP $(r, KG)$集合，是$T_p$的子集，其中包含亲和度分数最高的前-$k$ 个谓词对。

对于我们的运行示例 $q^E$，假设目标知识图谱为 DBpedia，则关系“岸边城市”的前 5 个谓词是：dbp:city、dbp:locationCity、dbo:nearestCity、dbp:cities 和 dbp:country。

[12]维基数据采用这套命名系统，并提供了一种查询描述的方法，例如：https://www.wikidata.org/wiki/Wikidata:SPARQL_查询_服务/查询#为_属性_添加_标签

**算法 2**KGQAnRelationLink

**输入：** $r$：$PGP(q)$ 中的一个关系，$KG$：目标知识图谱，$k$：谓词数
**输出：** $r$，并标注相关谓词 $\mathrm{R}\mathcal{P}(r,\ KG)$

1: $n^a$ $n^b$ 与 $r$ 相连的 $\leftarrow PGP(q)$ 个节点
2: $T_{rv} \leftarrow$ 所有 $v \in \mathrm{R}\mathcal{V}(n^a,\ KG) \cup \mathrm{R}\mathcal{V}(n^b,\ KG)$ 的集合
3: $T_{pd} \leftarrow \varnothing$ ▷ 相关顶点的谓词
4: **对**每个 $v \in T_{rv}$ **执行**
5: $T_{pd} \leftarrow T_{pd} \cup$ 出向谓词$(v)$ ▷ SPARQL 到 $KG$
6: $T_{pd} \leftarrow T_{pd} \cup$ 入向谓词$(v)$
7: **结束循环**
8: $T_p \leftarrow \varnothing$
9: **对于**每个 $p,\ d_p,\ v,\ o \in T_{pd}$ **do**
10: **如果**not isHumanReadable$(p)$ **则**
11: $d_p \leftarrow$获取谓词描述$(p)$
12: **结束 if**
13: $T_p \leftarrow T_p \cup p,\ S(l_r,\ d_p)\ v,\ o$ ▷计算语义亲和度
14: **结束 for**
15: **返回** $r.\mathrm{R}\mathcal{P} \leftarrow$个来自 $T_p$ 且具有最高-$k$ 亲和度分数的所有配对

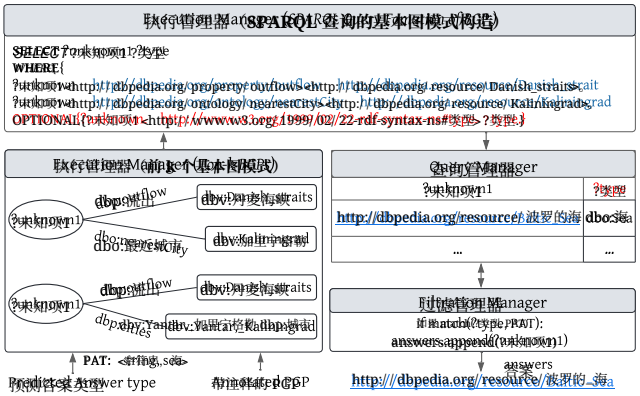



**图6：KGQAn 的执行与过滤阶段使用预测答案类型和带注释的 PGP。KGQAn 生成一组 BGP集合，并对其进行排序，以创建最多 $k$个查询。KGQAn 使用可选的三元组模式调整这些查询，以返回主要未知项的类型供过滤使用。**

算法 2 总结了针对单个关系 $r$ 的关系链接过程；对于每个关系 $r \in PGP(q)$，都必须调用该算法。第 2 行构造集合 $T_{rv}$，其中包含 $n^a$ 和 $n^b$ 的相关顶点的并集；$n^a$ 和 $n^b$ 是与 $r$ 相连的 PGP 节点。在第 3–7 行中，对于 $T_{rv}$ 中的每个相关顶点 $v$，我们在目标知识图谱上执行 SPARQL 查询 outgoingPredicate(v) 和 incomingPredicate(v)，以检索与 $v$ 相连的所有谓词及其描述；并将结果存入集合 $T_{pd}$。接着，第 8–14 行计算 $T_{pd}$ 中每个谓词与 $r$ 的语义关联度，并将得到的 $predicate, score$ 对存入 $T_p$，详见定义 5.2。在第 10–12 行中，如果 $?p$ 不是人类可读的（例如由随机字符和数字组成的序列），我们就发起查询以获取 $?p$ 的描述。最后，第 15 行从 $T_p$ 中选取得分排名前 $k$ 的对，以构造相关谓词集合 $\mathrm{R}\mathcal{P}(r, KG)$。

*5.2.1* **复杂度。** 算法 2 执行 $|\mathrm{R}|$ 次，其中 R 是 $PGP(q)$ 中关系的集合；参见定义 5.3。开销最大的是执行 SPARQL 查询。由于 RDF 引擎的索引 [59, 63] 中有查找查询，每个查询都可以在常数时间 $c_{lk}$内执行。然而，这类查询共有 $2 \cdot |T_{vr}|$ 个，其中 $|T_{vr}| \leq 2 \cdot maxVR$，因为有两个节点连接到 $r$；参见 potentialRelevantVertices 查询的定义。实践中，我们只使用 $k < maxVR$ 个顶点，因此第 4–7 行的开销变为 $O(k \cdot c_{lk})$。第 9 行 for 循环的开销取决于 $T_{pd}$，其中可能包含目标知识图谱中的所有谓词。不过，实践中我们会限制每个顶点检索到的谓词数量，因此开销为 $O(k)$；第 15 行同理。算法 2 的总体复杂度为 $O(k \cdot c_{lk} \cdot |\mathrm{R}|)$

## 5.3 带注释图模式（AGP）

设 $q$ 为一个问题，其对应的图 $PGP(q)$ 是在问题理解过程中构建的。给定一个知识图谱 $KG$，算法 1 和 2 以即时方式生成标注图 $AGP(q,\ KG)$，该图将 $PGP(q)$ 的抽象组件关联到 $KG$ 中的实际顶点和谓词，如图4所示。形式化地：

**定义 5.3** （带注释图模式（AGP））。设 $PGP(q)$ =(E,R) 为问题 $q$ 的短语图模式，$KG$ 为目标知识图谱。相应的带注释图模式 $AGP(q, KG)$ 是一个由节点 E 和关系 R 组成的无向图，其中每个 $n \in$ E 都标注有相关顶点 $\mathrm{R}\mathcal{V}(n, KG)$，每个 $r \in$R 都标注有相关谓词 $\mathrm{R}\mathcal{P}(r, KG)$。

## 5.4 语义亲和度计算

本节讨论如何计算语义亲和度分数 $S(\circ,\ \circ)$，该分数用于实体与关系链接。设 $l = (l_1, l_2, ...)$ 是由一系列词组成的字符串。对于每个词$l_i \in l$，我们使用在大量英语词汇上预训练的 FastText 模型 [3]，生成词嵌入 $E_w(l_i)$。其基本思想是，如果两个词在语义上相似，它们在嵌入的向量空间中就会彼此接近。如果 FastText 无法识别 $l_i$，我们则使用 chars2vec [7, 8] 预训练模型生成字符嵌入 $E_c(l_i)$，该模型能够捕捉词语拼写的相似性。设 $l^X$ 为一个字符串，$X = (x_1, x_2, ..., x_{|l^X|})$ 为一个嵌入数组，其中，如果 $l_i^X$ 出现在 FastText 中，则 $x_i = E_w(l_i^X)$；否则为 $x_i = E_c(l_i^X)$。再设 $l^Y$ 为一个字符串，其对应的嵌入数组为 $Y$。$l^X$ 与 $l^Y$ 之间的语义亲和度 [22] 定义如下：

$$S(l^X, l^Y) = \frac{\sum_{x_i \in X, y_j \in Y} sim(x_i, y_j)}{|X| \cdot |Y|} \quad (1)$$

其中 $sim(\circ,\ \circ)$ 表示余弦相似度。公式 1 考虑了所有 $(x_i,\ y_j)$ 对。其中一些配对可能包含来自不同模型的嵌入，例如 $\left(E_w(l_i^X)\ E_c(l_j^Y)\right)$；在这种情况下，我们定义 $sim(x_i,\ y_j) = 0$。

**算法3** 从 $AGP$生成 Top-$k$ SPARQL 查询

**输入：** $AGP(q,\ KG)$：一个带注释图模式
**输出：** $BGP_{SQ}$：Top-k SPARQL查询

```
1: BGP_all ← getBGPs(AGP(q, KG))        ▷ 所有可能的组合
2: BGP_S ← calculateScores(BGP_all)      ▷ 使用公式 2
3: BGP_R ← rank(BGP_S)
4: BGP_SQ ← getSPARQL(BGP_R, k)          ▷ 带有可选类型子句
```

虽然上述方法是我们默认的语义亲和度计算方法，但我们也使用基于句子的嵌入模型进行了实验。我们使用 GPT-3 [4] 预训练Transformer 为整个字符串生成单个嵌入。在这种情况下，公式 1 可简化为：$S(l^X,\ l^Y) = sim\left(E_{GPT}(l^X)\ E_{GPT}(l^Y)\right)$。

## 6 KGQAn执行与过滤

我们采用即时方法，将过滤过程从 RDF 引擎转移到 KGQAn，并针对给定的问题和任意知识图谱，在运行时为其制定计划。这一转变使 KGQAn 无需预处理即可运行。图 6 展示了 KGQAn 中的执行与过滤过程。KGQAn 的后过滤方法会汇总通过针对 $KG$ 执行语义等价查询（BGP集合）所收集的答案；$KG$ 是一组三元组 sub $p$, obj。在 $KG$中，obj 可以是顶点 sub 的某个类类型，而 $p$ 将是一个 rdf:type[13] 谓词。如果存在主要未知项的类类型，KGQAn 会通过在选定的顶层 BGP 中添加一个可选三元组模式来获取该类类型；$unknown1$，rdf:type, ?$c$。随后，KGQAn 根据这组经过调整的 BGP 生成 SPARQL 查询，将其发送至 RDF 引擎执行，并维护集合 {$a$，?$c$}，其中 $a$ 是收到的答案，?$c$ 是其关联的类类型。最后，KGQAn 遍历此集合，过滤掉每个 $a$，?$c$，其中 ?$c$ 与以下预测不匹配：(*i*) 日期或数值型情况下预测的数据类型；或 (*ii*) 预测的语义类型（若预测的数据类型为字符串）。

定义6.1（基本图模式（BGP））。设 $AGP(q, KG)$ =(E, R) 为问题 $q$和目标知识图谱 $KG$的带注释图模式。 $AGP(q, KG)$ 中三元组模式的集合为 $TP = \{n^a, r, n^b : n^a, r, n^b \in AGP(q, KG)\}$。对于 $TP$中的每个三元组，赋值 $v^a \rightarrow n^a$, $v^b \rightarrow n^b$ 和 $p \rightarrow r$，使得$v^a \in n^a$.RV $v^b \in n^b$.RV 和 $p \in r$.RP。所得三元组的方向 $v^a, p, v^b$ 或 $v^b, p, v^a$ 取决于 $p$的标志 $o$（见定义 5.2）。

算法3说明了 KGQAn 如何从带注释图模式 $AGP$生成排序的SPARQL查询列表。第1行创建 $BGP_{all}$，即使用 $AGP$中相关顶点和谓词的所有有效组合生成的 BGP集合；请注意，定义6.1仅描述了一个 这样的 BGP。第2行计算每个 $BGP \in BGP_{all}$的分数。 $BGP$ 的分数定义如下：

$$score(BGP) = \frac{1}{|TP|} \sum_{\langle v^a, p, v^b \rangle \in TP} \left(s_{v^a} + s_p + s_{v^b}\right) \qquad (2)$$

其中，$s_{v^a}$、$s_{v^b}$和$s_p$是相关顶点和谓词的得分；请参阅定义5.1和定义5.2。第3行根据计算出的得分对 $BGP_S$进行排序。在第4行，我们最终将排名前$k$的 BGP集合转换为 $BGP_R$ 对应的 SPARQL 查询。每个查询后都附加一个可选子句，其中包含三元组模式$unknown1$，rdf:type, ?$c$，用于获取与主要未知项关联的类型或类（如果存在）。

[13]https://www.w3.org/1999/02/22-rdf-syntax-ns#type

## 7 实验评估

### 7.1 评估设置

*7.1.1* 对比系统。 我们将 KGQAn 与 gAn- swer [27, 64], EDGQA [28], 和 NSQA [31]进行比较。EDGQA 在 QALD-9 和 LC-QuAD 1.0 中均处于最先进水平。gAnswer 在 QALD-9 挑战赛中排名第一 [54]。NSQA 使用深度学习模型来支持问题理解和链接，并在 QALD-9 中超越了 gAnswer。gAnswer [25] 和 EDGQA [19] 的代码均已公开。我们复现了 gAnswer 和 EDGQA 的结果。NSQA 使用逻辑神经网络 [45], 和专门的数据集来训练 AMR模型；二者均不可用。因此，我们仅报告 [31]中提供的结果。

*7.1.2* 四个不同的真实知识图谱。为在多样化的应用领域中评估这些系统，我们使用了四个真实知识图谱，即 DBpe- dia、YAGO [16, 51], DBLP [11] 以及 Microsoft Academic Graph（*MAG*） [34]。DBpedia 和 YAGO 均为通用事实知识图谱，其中大多数实体与地点和人物有关。DBLP 和 MAG 面向科学出版物、引文、作者和机构，其中一些实体通过较长的短语来标识，例如论文标题和会议名称。此外，这四个知识图谱具有不同的图规模，以研究图规模对这些系统所需预处理的影响，如表2所示。

*7.1.3* 基准测试与问题集。问答基准包括英语问题、对应的 SPARQL 查询标注，以及来自特定版本知识图谱的正确答案集合。QALD-9 [54] 和 LC-QuAD 1.0 [52] 被广泛用于评估问答系统在不同版本 DBpedia 上的表现。QALD-9 包含408个问题作为训练集，以及150个问题作为测试集。QALD-9 中问题的平均长度为7.5个词。LC-QuAD 1.0 包含由4000个问题构成的训练集和由1000个问题构成的测试集。这些问题基于不同模板创建。YAGO、DBLP 和 MAG 均没有黄金标准。因此，我们请一组具有计算机科学背景的学生，以大多数问答系统能够解决的 QALD-9 问题为参考，提出用于在 YAGO、DBLP 和 MAG 中查找事实的类似问题。我们为每个知识图谱收集了100个问题，并为其标注了正确的 SPARQL 查询和答案。这些新基准旨在评估问答系统在未见知识图谱和领域中解决与 QALD-9 类似问题的能力。表2总结了我们使用的五个基准测试。我们采用以下评估指标：精确率（P）、召回率（R）和宏观F1（F1），并使用 QALD-9自动评估工具 [42] 计算这些指标。

*7.1.4 SPARQL* 端点的 *RDF* 引擎。我们使用 Virtuoso 7.2.5.2 作为 SPARQL 端点，因为它被广泛用作大型知识图谱（如 DBpedia 和 MAG）的端点。我们为每个基准测试准备了五个不同的 Virtuoso 端点。实验中，所有问答系统都使用在这些端点上运行的 Virtuoso 引擎标准未修改安装版本。EDGQA链接方法（Falcon）使用 Elasticsearch。我们使用 Elasticsearch 7.10.2 为知识图谱建立索引，以便 EDGQA 使用 Falcon 链接方法。

**表 2：所用的基准测试、各知识图谱的规模，以及问答系统对知识图谱进行预处理（即索引）所需的时间。**

| 基准测试统计 | | | | EDGQA - 由 Falcon 进行索引 | | gAnswer | |
|---|---|---|---|---|---|---|---|
| 基准测试 | #问题数 | 知识图谱名称 | 三元组数（百万） | 索引时间（小时） | 索引大小（GB） | 索引时间（小时） | 索引大小（GB） |
| QALD-9 | 150 | DBpedia-10 | 194 | 6.51 | 1.80 | 2.86 | 8.60 |
| LC-QuAD 1.0 | 1000 | DBpedia-04 | 140 | 6.23 | 1.70 | 2.28 | 6.60 |
| YAGO-Bench | 100 | YAGO-4 | 145 | 6.88 | 2.00 | 1.81 | 4.10 |
| DBLP-Bench | 100 | DBLP | 136 | 4.83 | 1.60 | 1.91 | 5.20 |
| MAG-Bench | 100 | MAG | 13000 | 103.22 | 92.00 | 37.40 | 319.00 |

*7.1.5* 计算基础设施。我们将每个受评估的问答系统和知识图谱部署在同一台本地机器上运行的 Virtuoso SPARQL 端点上。实验采用两种不同的配置。第一种配置使用两台 Linux 机器，每台配备 16 个核心和 500GB 随机存取存储器。除 MAG 相关实验外，所有实验均使用这两台机器。第二种配置使用五台配备*32* 个核心和*3TB* 随机存取存储器的 Linux 机器。EDGQA 和 gAnswer 使用这五台机器对 MAG 进行预处理，并生成其方法运行所需的索引。

*7.1.6 KGQAn* 实现与设置。 KGQAn 使用[14] Python 3.7 实现，所有模型均使用 PyTorch 1.11.0 训练。我们的实现支持通过多线程并行执行和筛选。由于 gAnswer 和 EDGQA 不支持并行处理，我们没有为执行与筛选启用多线程。我们使用 BART [32] 构建 QU 序列到序列模型。语义亲和模型使用 FastText 的 wiki-news-300d-1M 模型。我们还在第 7.3 小节展示了使用 GPT-3 [4] 构建 QU 和语义亲和模型的实验。KGQAn 需要四个参数：(*i*) 最大获取顶点数，用于决定针对某个实体可以调查多少个顶点；(*ii*)顶点数，用于决定可以用多少个顶点为 PGP 中的每个节点添加注释；(*iii*) 谓词数，用于决定可以用多少个谓词为 PGP 中的每条边添加注释；以及 (*iv*) 最大查询数，用于决定针对该问题生成的等价 SPARQL 查询数量。我们的实验分别采用以下数值：*400*、*1*、*20* 和 *40*。我们根据每个顶点的谓词平均数量来确定谓词数。我们调整这些参数，使其适用于不同的知识图谱，而非针对某一个知识图谱追求最佳表现。

## 7.2 在真实知识图谱上的实验

gAnswer、EDGQA 和 NSQA 使用 QALD-9 和 LC-QuAD 1.0 进行训练和评估。我们在五个基准测试上，使用多种真实知识图谱将 KGQAn 与这些系统进行比较。我们执行了 gAnswer 和 EDGQA 所需的预处理。表2总结了各系统预处理所耗费的时间和存储空间。我们使用已见基准（即 QALD-9 和 LC-QuAD 1.0）以及未见基准，在 YAGO、DBLP 和 MAG 上评估了这些系统。表3展示了所有系统在五个基准测试上的结果。

*7.2.1* 预处理成本。EDGQA 集成了三种不同的链接方法，即 Falcon [47]、Dexter [6], 和 EARL[18]，用于实体与关系链接。我们使用 Falcon 为所有知识图谱建立索引，使 EDGQA 能够运行。为了复现 EDGQA 在 QALD-9 和 LC-QuAD 1.0 上的结果，我们使用 EDGQA 仓库提供的 Dexter 和 EARL 索引。对于 MAG，我们根据每种实体类型对应的谓词定制 Falcon 代码，例如论文的标题和作者的姓名。gAnswer 提供了预处理所需的索引机制和关系提及文件。我们使用配备 *3TB* 随机存取存储器的机器运行 gAnswer 索引脚本。如表 2 所示，对于 EDGQA，Falcon 为知识图谱建立索引所需的时间比 gAnswer 的索引方法更长。

*7.2.2* 已见基准。对于 LC-QuAD 1.0 基准测试，如表3所示，KGQAn 的 F1 分数几乎与 EDGQA 相同，精确率则提高了 15%。由于积极采用了三个不同的链接系统进行索引，并使用人工整理规则来理解 LC-QuAD 1.0 中的问题，EDGQA 的召回率比 KGQAn 高出近 19%。这些问题是使用模板自动生成的，也就是说，针对每个模板整理规则，有助于在训练问题和测试问题上取得良好表现。与 LC-QuAD 1.0 不同，QALD-9 的问题由人工创建，复杂程度各不相同。因此，QALD-9 更具挑战性，例如，人工整理的规则可能适用于训练问题，却无法处理测试问题。这解释了 EDGQA 在 LC-QuAD 1.0 和 QALD-9 上的 F1 分数为何差异巨大。如表3所示，KGQAn 在 QALD-9 上优于现有问答系统，其精确率和 F1 分数分别达到 49.81 和 43.99，表现出色。

*7.2.3 YAGO*、*DBLP* 和 *MAG* 上的未见基准测试。为了评估各系统针对任意知识图谱处理问题的通用性，我们采用了三个所有系统（包括 KGQAn ）都未见过的基准测试。其中一个基准测试包含与 QALD-9 中的问题相似的问题，目标知识图谱为 YAGO；YAGO 与 DBpedia 类似。另外两个基准测试面向与 DBpedia 不同的领域，问题涉及 DBLP 和 MAG 中与论文及作者相关的事实。有趣的是，gAnswer 在 YAGO 基准测试中的表现优于 LC-QuAD 1.0 基准测试，因为 YAGO 中的问题与 QALD-9 中的问题相似。在 MAG 和 DBLP 的大多数情况下，顶点的 URI 以代码结尾，例如，“Jim Gray”的 URI 是 mag*e*:2279569217，参见第 5.1 小节。gAnswer 的倒排索引基于 URI。因此，gAnswer 的链接功能无法找到“Jim Gray”。所以，由于其问题理解和链接能力，gAnswer 在 DBLP 中回答了两个问题，在 MAG 中则一个也没答对。在 DBLP 和 MAG 上，EDGQA 未能回答大多数问题，因为它无法理解涉及名称较长的实体的问题。KGQAn 的问题理解模型在提取较长名称的实体方面具有更好的泛化能力。此外，KGQAn 的语义亲和模型基于通用英语文本训练。这使我们的即时链接能够在不同领域的知识图谱上良好运行，并利用 RDF 引擎的内置索引。

[14]https://github .com/CoDS -GCS/KGQAn

**表3：五个基准测试的结果。KGQAn 在不同领域知识图谱上的召回率和精确率表明，得益于我们的序列到序列模型和即时链接（无需针对特定知识图谱进行预处理），KGQAn 的表现优于其他系统。**

| 系统 | QALD-9-DBpedia | | | LC-QuAD 1.0-DBpedia | | | YAGO | | | DBLP | | | MAG | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | P | R | F1 | P | R | F1 | P | R | F1 | P | R | F1 | P | R | F1 |
| NSQA | 31.89 | 32.05 | 31.26 | 44.76 | 45.82 | 44.45 | - | - | - | - | - | - | - | - | - |
| gAnswer | 29.34 | 32.68 | 29.81 | 82.21 | 4.31 | 8.18 | **58.49** | 34.05 | 43.04 | **78.00** | 2.00 | 3.90 | 0.0 | 0.0 | 0.0 |
| EDGQA | 31.30 | **40.30** | 32.00 | 50.50 | **56.00** | **53.10** | 41.90 | 40.80 | 41.40 | 8.00 | 8.00 | 8.00 | 4.00 | 4.00 | 4.00 |
| **KGQAn** | **51.13** | 38.72 | **44.07** | **58.71** | 46.11 | 51.65 | 48.48 | **65.22** | **55.62** | 57.87 | **52.02** | **54.79** | **55.43** | **45.61** | **50.05** |

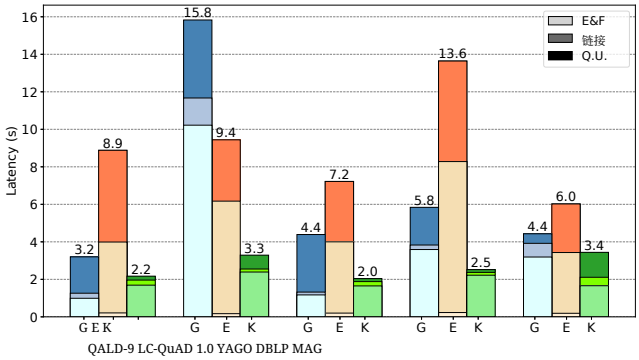



**图 7：gAnswer (G)、EDGQA (E) 和 KGQAn (K) 的响应时间。每个柱形自下而上显示按问题理解、链接和执行/过滤 (E&F) 分类的平均响应时间。**

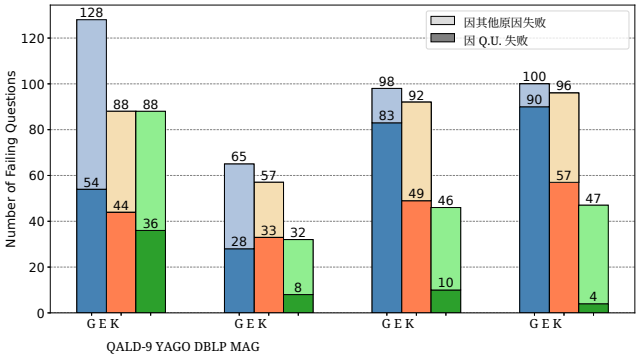



**图8：每个基准测试中 R=0 和 F1=0 失败的问题数量。每个柱形表示自下而上分类为因问题理解或其他原因失败的问题总数。在多种基准测试中，KGQAn 失败的问题数量最少。**

*7.2.4* 响应时间。本实验从响应时间角度分析问答系统。对于每个系统，我们计算其回答每组问题的平均响应时间。我们采用本地部署方式，将问答系统和基准测试的 SPARQL 端点部署在同一台机器上，以消除网络延迟或知识图谱（如 MAG 或 DBpedia ）的实际端点负载所造成的波动。我们报告各步骤的平均耗时，包括问题理解（问题理解）、链接以及执行/过滤，如图 7 所示。KGQAn 的问题理解模型占用了大部分响应时间。KGQAn 的链接通常耗时最短。最后，KGQAn 的执行与筛选比链接耗时更长，具体取决于生成的 SPARQL 查询的复杂程度以及查询结果的规模。

在 gAnswer 和 EDGQA 中，总响应时间主要取决于问题理解的准确性，具体而言：（*i*）提取出的实体和关系会影响链接时间；（*ii*）提取出的三元组数量会影响生成的 SPARQL 查询数量，也就是执行时间。总响应时间更多地受整个问答流程的复杂度影响，而非图规模。例如，KGQAn 在 LC-QuAD 1.0 和 MAG 上耗时相近。总体而言，EDGQA 在链接阶段耗时更长，因为它采用的链接方法使用基于磁盘的索引。相比之下，gAnswer 会在处理任何问题之前将索引加载到内存中。

## 7.3 分析 KGQAn

*7.3.1* 问题理解。本实验从回答问题时完全失败的角度分析问答系统，即召回率。
且 F1值为零。这种失败可能是由问题理解或其他原因造成的，例如链接或过滤。无法理解问题意味着未能正确检测出任何实体或关系短语。我们手动统计了各基准测试中因问题理解而失败的案例，汇总结果如图8所示。在不同领域的基准测试中，KGQAn 对大多数问题的理解能力均优于 gAnswer 和 EDGQA；无论是总体失败问题数，还是每项基准测试中因问题理解而失败的问题数，KGQAn 都是最少的。对于未见过的领域（即 DBLP），KGQAn 理解问题的能力也优于 gAnswer 和 EDGQA。

*7.3.2* 即时链接与后置过滤。 [18]提供了用于 LC-QuAD 1.0 实体与关系链接任务的标注数据集。在本实验中，我们使用该数据集分析 gAnswer 和 EDGQA 所采用链接方法的性能，如图9所示。我们提出的即时链接方法通过将部分链接任务转化为由 RDF 引擎执行的查询，并在 KGQAn 中进行语义排序，从而无需预处理阶段。EDGQA 使用了三种不同的链接系统，因此在该任务中表现出色。gAnswer 的问题理解仅基于 QALD-9 进行训练。因此，它在大多数情况下都无法提取正确的实体和关系，因而在 LC-QuAD 1.0 的链接任务中表现不佳。我们的方法旨在最大限度地提高召回率，以找到正确的顶点。KGQAn 会对无关答案进行后置过滤，以提高精确率。因此，KGQAn 的最终 F1 分数几乎与实体链接取得的最高 F1 分数相同。相比之下，EDGQA 的整体处理流程对三种链接系统所取得的 F1 分数产生了负面影响。

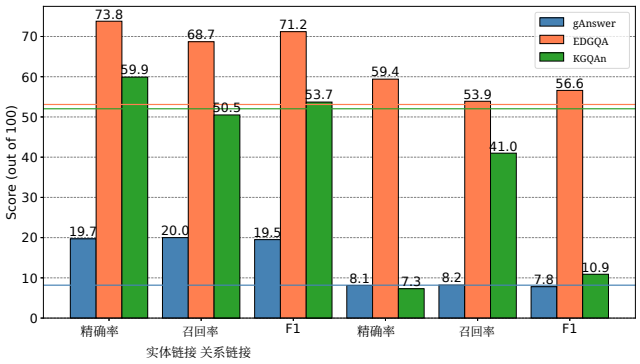



**图9：使用 LC-QuAD1.0 进行实体与关系链接的结果表明，EDGQA 能够从结合三种链接方法中获益。水平线表示各系统的最终 F1 分数。与 EDGQA 不同，KGQAn 的最终 F1 分数几乎与其实体链接阶段的最高 F1 值相同。**

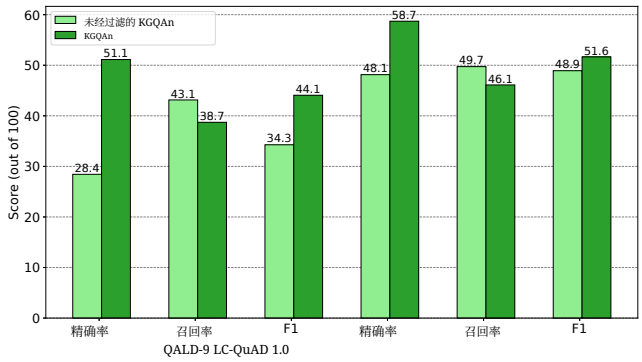



**图 10：使用 QALD-9 和 LC-QuAD 1.0 分析 KGQAn 在过滤和不进行过滤情况下的性能。过滤提高了精确率，但略微降低了召回率。对于两个基准测试，最终的 F1 分数均有所提高。**

**表4：KGQAn 使用不同预训练模型训练问题理解（QU）和语义亲和度（SA）模型时的性能。在大多数情况下，我们采用 BART 和细粒度（FG）亲和度的默认设置，其 F1 分数优于在问题理解和粗粒度（CG）亲和度中使用 GPT-3 的变体。**

| 基准测试 | 问题理解：BART<br>SA：细粒度 | 问题理解：GPT-3<br>SA：细粒度 | 问题理解：BART<br>SA：GPT-3 |
|---|---|---|---|
| QALD-9 | **44.07** | 42.12 | 42.60 |
| LC-QuAD 1.0 | 51.65 | **52.87** | 50.86 |
| YAGO | **55.62** | 54.94 | 55.02 |
| DBLP | **54.79** | 54.42 | 41.72 |
| MAG | **50.05** | 49.26 | 37.64 |

*7.3.3* 过滤效果。本实验分析 KGQAn 在使用和不使用过滤时的性能。由于篇幅有限，我们仅展示在 QALD-9 和 LC-QuAD 1.0 上的结果，如图10所示。KGQAn 会预测答案数据类型。我们的模型在根据日期、数值型或布尔型数据类型过滤答案方面表现出色。由于歧义较大，使用语义类型过滤答案的准确度不如使用日期、数值型或布尔型数据类型。总体而言，我们设计过滤方法时尽量避免大幅降低召回率。QALD-9 中答案为日期、数值型或布尔型的问题占比高于 LC-QuAD 1.0。因此，使用过滤的 KGQAn 在 QALD-9 上的表现优于在 LC-QuAD 1.0 上的表现。

*7.3.4 KGQAn*和不同的预训练语言模型。本实验分析了不同预训练语言模型（PLM）在问题理解和语义亲和度方面对 KGQAn 性能的影响，这两方面均用于链接和过滤。在问题理解方面，我们使用 BART [32] 和 GPT-3 [4] 来建模三元组模式提取任务。对于 BART，我们使用 Huggingface API [29] 进行训练，因为它允许我们灵活调整训练参数。GPT-3 模型只能通过 OpenAI API [37]进行微调。因此，我们使用 GPT-3 优化模型时的控制能力较弱。

我们的语义亲和模型依赖于两组词的表示（嵌入）。我们采用细粒度方法，以词对为粒度估算亲和度。在这种方法中，我们使用 FastText 和 chars2vec 模型。作为粗粒度方法，我们使用 GPT-3 为每组词获取一个向量嵌入。

我们比较了 KGQAn 使用这些不同模型组合时的性能。KGQAn 默认采用我们的基于 BART 的序列到序列模型和细粒度语义亲和度（FG）模型。我们创建了 KGQAn 的两个不同变体：在其中一个变体中，我们将基于 BART 的序列到序列模型替换为使用 GPT-3 进行问题理解训练的模型。在第二个变体中，我们使用基于 BART 的序列到序列模型，以及基于 GPT-3 的粗粒度语义亲和度（CG）模型。表 4 从左到右展示了每个基准测试中，KGQAn 在默认设置和两个变体下得到的最终 F1值。我们的默认设置在大多数情况下优于这两个变体。

## 7.4 基准测试问题的分类体系

我们构建了一套分类体系，根据不同特征研究基准测试问题的复杂度，并依据该分类体系评估 KGQAn、gAnswer 和 EDGQA 的性能。如表 5 所示，我们的分类体系根据（*i*）候选 SPARQL 查询的形状和（*ii*）语言复杂度对问题的复杂度进行分类。SPARQL 查询可分为星形查询或路径查询 [39]。在我们的分类体系中，星型查询由一个或多个共享同一主语的三元组模式组成。路径查询至少包含两个三元组模式，其中一个三元组模式的宾语是另一个模式的主语，即呈线性形状。依据 LC-QuAD 2.0 [17], 提供的语言分类，QALD-9 中的问题和未见基准测试中的问题被分为单事实、带类型的单事实、多事实和布尔问题。在不同基准测试的大多数情况下，KGQAn 的表现均优于 gAnswer 和 EDGQA。

**表 5：根据 LC-QuAD 2.0 中问题的语言复杂度以及 SPARQL 查询的形状，对基准测试问题进行分类后，三个系统分别解决的问题数量。0 [17]**

| 查询类型 | SPARQL 形状 | | | | | | | | LC-QuAD 2.0 分类体系 | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | Star | | | | Path | | | | 单一事实 | | | | 带类型的事实 | | | | 多重事实 | | | | 布尔型 | | | |
| 基准测试 | # queries | **KGQAn** | EDGQA | gAnswer | # queries | **KGQAn** | EDGQA | gAnswer | # queries | **KGQAn** | EDGQA | gAnswer | # queries | **KGQAn** | EDGQA | gAnswer | # queries | **KGQAn** | EDGQA | gAnswer | # queries | **KGQAn** | EDGQA | gAnswer |
| QALD-9 | 131 | **60** | 56 | 21 | 19 | 2 | **5** | 0 | 81 | **46** | 41 | 16 | 28 | 7 | **8** | 3 | 37 | 9 | 9 | 2 | 4 | 0 | **3** | 0 |
| YAGO-B | 92 | **63** | 39 | 32 | 8 | **5** | 4 | 3 | 87 | **61** | 40 | 33 | 6 | **5** | 3 | 2 | 6 | **2** | 0 | 0 | 1 | 0 | 0 | 0 |
| DBLP-B | 92 | **46** | 8 | 2 | 8 | **8** | 0 | 0 | 85 | **49** | 8 | 1 | 11 | **4** | 0 | 1 | 4 | **1** | 0 | 0 | 0 | - | - | - |
| MAG-B | 77 | **44** | 4 | 0 | 23 | **9** | 0 | 0 | 75 | **40** | 4 | 0 | 7 | **2** | 0 | 0 | 16 | **9** | 0 | 0 | 2 | **2** | 0 | 0 |

## 8 相关工作

人们正投入越来越多的努力，运用不同的问题理解和链接技术开发问答系统 [38, 43]。问答系统可分为基于 SPARQL 和非基于 SPARQL 的系统。非基于 SPARQL 的系统包括 QAmp [55], 、Treo [23], 等 [60, 62]。这些系统不会将问题转换为 SPARQL 查询，而是实现专有的查询引擎；因此，它们无法用于任意 SPARQL 端点。

基于 SPARQL 的系统运用各种技术，将自然语言问题转换为 SPARQL 查询，包括：（*i*）问题理解，以生成中间抽象表示；以及（*ii*）将抽象表示链接到知识图谱的顶点和谓词。例如 gAnswer [27, 64], EDGQA [28], NSQA [31], WDAqua-core 1 [15], Bio-SODA [48], 和 QAnswerKG [14]。

问题理解技术分为以下几类：（*i*）基于规则的方法，例如 gAnswer [27, 64], EDGQA [28], 和 WDAqua-core1 [15]。这些方法根据词性（POS）标注整理规则，以提取实体和关系短语。与 KGQAn 不同，基于人工整理规则的系统难以推广到广泛的应用场景和多元化的用户群体。（*ii*）基于深度学习的方法，例如 NSQA [31], 。该方法整理大量问题的树表示，并使用深度学习训练一个名为 AMR 的语义解析器。需要注意的是，AMR 的粒度与 SPARQL 不同，因此需要调整 NSQA 才能适用于新领域 [31]。KGQAn 不存在这一问题，因为它通过遍历 PGP 生成 SPARQL 查询，而 PGP 的节点和边均标注了知识图谱中的相关顶点和谓词。（*iii*）基于词元的方法 [14, 48] ，从问题中提取与这些系统构建的倒排索引中的条目匹配的最长关键词序列。基于词元的系统依赖于特定知识图谱。

KGQAn 使用我们的即时方法对生成的 PGP 进行标注，该方法无需对目标知识图谱进行任何预处理。与 KGQAn 不同，现有系统会预先为目标知识图谱建立索引，例如 EDGQA [28] 和 gAnswer [27, 64], ；或者构建包含知识图谱所有顶点和谓词的向量空间，利用深度学习模型预测链接，例如 NSQA [31]。在这些系统中，预处理成本与知识图谱的规模成正比。例如，在 gAnswer 中，预处理复杂度是知识图谱顶点数量的多项式函数 [27, 64]。EDGQA 使用三个不同的系统为目标知识图谱建立索引，每个系统为知识图谱建立索引的方式各不相同。

KGQAn 从问题中抽取一系列三元组模式；实体可以是未知项（即变量）。与我们的问题不同，现有的关系三元组抽取技术 [9, 58] 不处理未知项；此外，关系是从预先定义的列表中选取的，也就是说，它们并非从给定文本中抽取。因此，关系三元组抽取模型及其训练数据集无法用于我们的任务。

我们使用 BART [32]将任务训练为序列到序列模型。现有系统也使用序列到序列模型。例如，文献 [46] 利用提取的知识图谱特定信息进行训练，以生成问题中关系对应的谓词序列。此外，文献 [57] 接收一个经过标注的问题，并生成一个经过标注的 SQL 查询。QAmp [55]也使用序列到序列模型，但并非将其用于生成任务，而是用于将问题中的序列标记为实体、谓词或类。与这些模型不同，KGQAn 的序列到序列模型不依赖查询语言、特定知识图谱或特定领域。因此，我们的模型能够更灵活地理解不同领域中的问题，无需领域专家准备训练集。

## 9 结论

现有问答系统针对特定领域和知识图谱构建，且需要耗费大量成本进行预处理。因此，它们无法按需处理针对任意知识图谱的问题。本文介绍了 KGQAn，这是一种按需知识图谱问答平台，能够克服上述局限。KGQAn 提出了一种新颖的问题理解形式化方法：利用序列到序列神经网络对三元组模式进行提取。我们的模型能够泛化，从而理解不同领域的问题。此外，KGQAn 引入了即时链接与过滤方法，将实体与关系链接作为语义搜索查询执行，并将部分工作卸载至 RDF 引擎。我们使用广泛应用的基准测试和四个不同的真实场景知识图谱，对最先进水平的问答系统进行了评估。根据知识图谱的规模，KGQAn 可节省数小时至数天的预处理时间。在常用基准测试上，KGQAn 的 F1 分数与最先进水平相当；而对于此前未见的知识图谱，其表现则大幅超越最先进水平。在未来工作中，我们计划扩展问题理解训练，以涵盖更复杂的问题；也计划支持多意图问题。

**致谢**。感谢 EDGQA、gAnswer 和 Falcon 的作者协助我们复现其结果。

# 参考文献


[1] Ibrahim Abdelaziz、Essam Mansour、Mourad Ouzzani、Ashraf Aboulnaga 和 Panos Kalnis。2017。Lusail：一个可大规模查询关联数据的系统。VLDB基金会会议录（PVLDB）11，4（2017），485–498。http: //www.vldb.org/pvldb/vol11/p485-abdelaziz.pdf [2] Waqas Ali、Muhammad Saleem、Bin Yao、Aidan Hogan 和 Axel-Cyrille Ngonga Ngomo。2022。用于查询知识图谱的 RDF 存储库与 SPARQL引擎综述。VLDB Journal 31，3（2022），1–26。https://doi.org/10.1007/s00778-021- 00711-3[3] Piotr Bojanowski、Edouard Grave、Armand Joulin 和 Tomás Mikolov。2017。利用子词信息丰富词向量。《计算语言学协会汇刊》（ 2017 135 146 999 ），–。[4] https://transacl.org/ojs/index.php/tacl/article/view/ Tom B. Brown、Benjamin Mann、Nick Ryder、Melanie Subbiah、Jared Kaplan 等。2020。语言模型是少样本学习器。载于《神经信息处理系统进展》第33卷：2020年神经信息处理系统年度会议，NeurIPS。https://proceedings.neurips.cc/paper/2020/hash/1457c0d6bfcb4967418bfb8ac142f64a-Abstract.html[5] Damian Bursztyn、François Goasdoué、Ioana Manolescu 和 Alexandra Roatis。2015。网络数据推理：算法与性能。载于 IEEE国际数据工程会议（ICDE）论文集，第1272卷。1541–1544。https://doi.org/10.1109/ICDE.2015.7113422[6] Diego Ceccarelli、Claudio Lucchese、Salvatore Orlando、Raffaele Perego 和 Salvatore Trani。2014。Dexter 2.0：用于语义丰富数据的开源工具。载于国际语义网会议海报与演示分会场（ISWC）论文集，第1272卷。417–420。http://ceur-ws.org/Vol-1272/ paper_127.pdf [7] Chars2vec：基于循环神经网络的字符级词嵌入模型，文章。2019。https://hackernoon.com/chars2vec-character-based-language-model-for-handling-real-world-texts-with-spelling-errors-and-a3e4053a147d。[8] Chars2vec：基于循环神经网络的字符级词嵌入模型，GitHub。2019。https://github.com/IntuitionEngineeringTeam/chars2vec。[9] Yubo Chen、Yunqi Zhang 和 Yongfeng Huang。2022。通过文本与图的相互生成学习关系三元组抽取的推理模式。载于计算语言学协会（ACL）论文集精选。1638–1647 2022 129 。https://aclanthology.org/ .findings-acl.[10] 2022。CoreNLP。https://stanfordnlp.github.io/CoreNLP/。[11] DBLP 发布版。2022。https://dblp.org/rdf/release/dblp-2022-06-01.nt.gz。[12] Marie-Catherine De Marneffe 和 Christopher D Manning。2016。斯坦福类型化依存关系手册。技术报告。斯坦福大学。https://nlp.stanford. edu/software/dependencies_manual.pdf [13] Jacob Devlin、Ming-Wei Chang、Kenton Lee 和 Kristina Toutanova。2019。BERT：用于语言理解的预训练双向 Transformer。载于计算语言学协会北美分会：人类语言技术会议论文集（NAACL-HLT）。 4171 4186 10 18653 1 19 1423 –。https://doi.org/ . /v /n [14] - Dennis Diefenbach、José Giménez-García、Andreas Both、Kamal Singh 和 Pierre Maret。2020。QAnswer KG：设计一个基于 RDF 数据的可移植问答系统。载于国际语义网会议（ESWC）论文集，第12123 429 445 10 1007 978 3 030 49461 2_25 卷。–。https://doi.org/ . / - - - - Dennis Diefenbach、Kamal Deep Singh 和 Pierre Maret。[15] 2018 1 。WDAqua-core：面向 RDF知识库的问答服务。载于万维网大会（ACM）伴随论文集。1087–1091。https://doi.org/10.1145/3184558. 3191541[16] YAGO 下载。2022。https://yago-knowledge.org/downloads/yago-4。[17] Mohnish Dubey、Debayan Banerjee、Abdelrahman Abdelkawi 和 Jens Lehmann。2019。LC-QuAD 2.0：面向维基数据和 DBpedia 上复杂问答的大型数据集。载于《语义网》国际会议（ISWC）论文集，第11779卷。 69 78 10 1007 978 3 030 30796 7_5 –。https://doi.org/ . / - - - - Mohnish Dubey、Debayan Banerjee、Debanjan Chaudhuri 和 Jens Lehmann。2018。[18] EARL：面向知识图谱问答的实体与关系联合链接。载于国际语义网会议（ISWC）论文集，第 11136 108 126 10 1007 978 3 030 00671 6_7 卷。–。https://doi.org/ . / - - - - EDGQA 代码。https://github.com/HXX /EDGQA。[20] EDGQA 日志。2021。https://raw.githubusercontent.com/HXX97/EDGQA/main/ query_logs/query_log_test_lc-quad_2021_07_08%2023_48.txt。[21] Anthony Fader、Stephen Soderland 和 Oren Etzioni。2011。识别开放信息抽取中的关系。载于自然语言处理实证方法会议（EMNLP）论文集。1535–1545。https: //aclanthology.org/D11-1142/[22] Raul Fernandez、Essam Mansour、Abdulhakim Qahtan、Ahmed Elmagarmid、Ihab Ilyas 等。2018。语义渗透：利用词嵌入链接数据集以实现数据发现。载于国际数据工程会议（ICDE）论文集。989–1000。https://doi.org/10.1109/ICDE.2018.00093[23] André Freitas、João Gabriel Oliveira、Seán O'Riain、João Carlos Pereira da Silva 和 Edward Curry。2013。利用语义相关性查询关联数据图：一种独立于词汇表的方法。《数据与知识工程》（DKE）88（2013），126–141。https://doi.org/10.1016/j.datak.2013.08.003

[24] Luis Antonio Galárraga、Christina Teflioudi、Katja Hose 和 Fabian M. Suchanek。2013 年。AMIE：不完整证据下本体知识库中的关联规则挖掘。载于国际万维网会议（WWW）论文集。 413 422 10 1145 2488388 2488425 –。https://doi.org/。[25] 2022 。GAnswer 代码。。https://github.com/pkumod/gAnswer。[26] Matt Gardner、Joel Grus、Mark Neumann、Oyvind Tafjord、Pradeep Dasigi 等。2018 年。AllenNLP：深层语义自然语言处理平台。载于 NLP 开源软件研讨会（NLP-OSS）论文集。1–6。https: //aclanthology.org/W 18 2501 - Sen Hu、Lei Zou、Jeffrey Yu、Haixun Wang 和 Dongyan Zhao。2018 年。通过知识图谱上的子图匹配回答自然语言问题。《IEEE Transactions on Knowledge and Data Engineering》（TKDE）30（2018），824–837。https://doi.org/[27] 10 1109 2017 2766634 。/TKDE。。Xixin Hu、Yiheng Shu、Xiang Huang 和 Yuzhong Qu。2021 年。基于 EDG 的问题分解：面向知识库的复杂问答。载于国际语义网会议（ISWC）论文集。128–145。https: //doi.org/10 1007 978 3 030 88361 4_8 。/---- Huggingface。。https://huggingface.co/。[28] 2022 。[29] Jena 文本搜索。2021 年。https://jena.apache.org/documentation/query/text-query。html。[30] Pavan Kapanipathi、Ibrahim Abdelaziz、Srinivas Ravishankar、Salim Roukos、Alexander G. Gray 等。2021 年。利用抽象意义表示进行知识库问答。载于计算语言学协会年会论文集：发现篇（ACL/IJCNLP）。3884–3894。https://doi.org/10.18653/v1/2021。findings-acl。[32] 339 Mike Lewis、Yinhan Liu、Naman Goyal、Marjan Ghazvininejad、Abdelrahman Mohamed 等。2020 年。BART：用于自然语言生成、翻译和理解的去噪序列到序列预训练。载于计算语言学协会年会（ACL）论文集。 7871 7880 10 18653 1 2020 703 –。https://doi.org/。/v/.acl-main。[33] Xueling Lin、Haoyang Li、Hao Xin、Zijian Li 和 Lei Chen。2020 年。KBPearl：由联合实体与关系链接支持的知识库构建系统。《PVLDB》13，7（2020），1035–1049。https: //doi.org/ 10 14778 3384345 3384352 。/。[34] 2022 MAG 记录。。https://zenodo.org/record/4617285#.YrNszNLMJhH。[35] Ndapandula Nakashole、Gerhard Weikum 和 Fabian M. Suchanek。2012 年。PATTY：具有语义类型的关系模式分类体系。载于自然语言处理实证方法与计算语言学习会议（EMNLP-CoNLL）论文集。1135–1145。https: //aclanthology.org/D12-1104/[36] Tahira Naseem、Srinivas Ravishankar、Nandana Mihindukulasooriya、Ibrahim Abdelaziz 和 Young-Suk Lee。2021 年。面向知识库问答的语义感知关系链接 Transformer 模型。载于计算语言学协会年会与自然语言处理国际联合会议（ACL/IJCNLP）论文集。256 262 10 18653 1 2021 34 –。https://doi.org/。/v/。acl-short。[37] OpenAI。2022 。。https://openai.com/。[38] Fatma Ozcan、Abdul Quamar、Jaydeep Sen、Chuan Lei 和 Vasilis Efthymiou。2020 年。数据自然语言接口的现状与开放性挑战。载于国际数据管理会议（SIGMOD）论文集。ACM，2629 2636 10 1145 3318464 3383128 –。https://doi.org/。/。[39] 2016 M. Tamer Özsu。。RDF 数据管理系统综述。载于《Frontiers of Computer Science》10，3（2016），418–432。https://doi.org/10.1007/s11704-016-5554-y [40] Jeff Pan、Mei Zhang、Kuldeep Singh、Frank Harmelen、Jinguang Gu 等。2019 年。实体增强的关系链接。载于国际语义网会议（ISWC）论文集，第 11778 卷。523–538。https://doi.org/10.1007/978-3-030-30793-6 30[41] 预定义同义词。2022 年。https://drive.google.com/file/d/1hmqaftrTo0_ qQNRApCuxFXaBx7SosNVy/view。[42] QALD 脚本。2022 年。https://github.com/ag-sc/QALD/blob/master/6/scripts/evaluation.rb。[43] Abdul Quamar、Vasilis Efthymiou、Chuan Lei 和 Fatma Özcan。2022 年。数据自然语言接口。《Foundations and Trends in Databases》11，4（2022），319–414。https://doi.org/10 1561 1900000078 。/ Hoan Nguyen Mau Quoc、M Serrano、HN Mau、JG Breslin 和 D Le-Phuoc。2019 年。面向关联传感器数据的 RDF 存储库性能研究。《Semantic Web Journal》1（2019），72。[44] http://www.semantic-web-journal.net/system/files/swj2249.pdf [45] Ryan Riegel、Alexander G. Gray、Francois P. S. Luus、Naweed Khan、Ndivhuwo Makondo 等。2020 年。逻辑神经网络。CoRR abs/2006.13155（2020 2006 13155 2006 13155 ）。arXiv：。https://arxiv.org/abs/。[46] Gaetano Rossiello、Nandana Mihindukulasooriya、Ibrahim Abdelaziz、Mihaela A. Bornea、Alfio Gliozzo 等。2021 年。用于知识库问答的生成式关系链接。载于语义网会议（ISWC）论文集（《计算机科学讲义》），第 12922 卷。Springer，321–337。https: //doi.org/ 10 1007 978 3 030 88361 4_19 。/----Ahmad Sakor、Isaiah Mulang、Kuldeep Singh、Saeedeh Shekarpour 和 Maria-Esther Vidal 等。2019 年。旧即是金：语言驱动的实体与[47]

短文本关系链接。载于北美计算语言学协会分会会议：人类语言技术（*NAACL-HLT*）论文集。2336–2346。https://doi.org/10.18653/v1/n19-1243[48] Ana Sima、Tarcisio Farias、Maria Anisimova、Christophe Dessimoz 和 Marc Rechavi 等。2021 年。Bio-SODA：无需训练数据即可在知识图谱上实现自然语言问答。载于科学与统计数据库管理国际会议（*SSDBM*）论文集。61 72 10 1145 3468791 3469119 –。https://doi.org/。/。Valentin I. [49] Spitkovsky 和 Angel X. Chang。。面向英文维基百科概念的跨语言词典。载于语言资源与评测国际会议（*LREC*）论文集。3168–3175。2012 http://www.lrec-conf.org/proceedings/lrec2012/summaries/266.html [50] Ilya Sutskever、Oriol Vinyals 和 Quoc Le。2014 年。使用神经网络进行序列到序列学习。载于神经信息处理系统进展会议（*NeurIPS*）论文集。3104–3112。h t t ps://proceedings.neurips.cc/paper/2014/hash/a14ac55a4f27472c5d894ec1c3c743d2-Abstract.html[51] Thomas Tanon、Gerhard Weikum 和 Fabian Suchanek。2020 年。YAGO 4：一个合理的知识库。载于欧洲语义网会议（*ESWC*）论文集，第 12123 583 596 10 1007 978 3 030 49461 2_34 卷。–。https://doi.org/。/---- [52] Priyansh Trivedi 、 Gaurav Maheshwari 、 Mohnish Dubey 和 Jens Lehmann。2017 年。LC-QuAD：面向知识图谱复杂问答的语料库。载于国际语义网会议（*ISWC*）论文集，第 10588 卷。210 218 10 1007 978 3 319 68204 4_22 –。https://doi.org/。/---- Immanuel [53] Trummer。。从 BERT 到 GPT-Codex：利用超大型语言模型在数据管理中的潜力。《*VLDB*基金会会刊》15，12（2022），3770–3773。https://www.vldb.org/pvldb/vol15/p3770-trummer.pdf [54] Ricardo Usbeck、Ria Gusmita、Axel-Cyrille Ngomo 和 Muhammad Saleem。2018 年。第 9 届链接数据问答挑战赛（QALD-9）。载于第 *4* 届语义深度学习研讨会（*SemDeep-4*）、*NLIWoD4*：数据网络自然语言接口（*NLIWOD-4*）研讨会与第 *9* 届链接数据问答挑战赛（*QALD-9*）联合论文集：该活动与第 *17* 届国际语义网会议（*ISWC*）同期举行，第 2241 卷。58–64。2022 3 http://ceur-ws.org/Vol-2241/paper-06.pdf[55] Svitlana Vakulenko、Javier Garcia、Axel Polleres、Maarten Rijke 和 Michael Cochez。2019 年。面向知识图谱复杂问答的消息传递。载于*ACM*信息与知识管理国际会议（*CIKM*）论文集。1431–1440。https://doi.org/10.1145/3357384。3358026[56] Ashish Vaswani、Noam Shazeer、Niki Parmar、Jakob Uszkoreit、Llion Jones 等。2017。Attention is All you Need。神经信息处理系统进展（*NeurIPS*）论文集，5998–6008。https://proceedings.neurips.cc/paper/2017/hash/3f5ee243547dee91fbd053c1c4a845aa-Abstract.html [57] Wenlu Wang、Yingtao Tian、Haixun Wang 和 Wei-Shinn Ku。2020。数据库的自然语言接口：运用对抗方法实现问题理解的迁移学习能力。载于*IEEE*国际数据工程会议（*ICDE*）论文集。IEEE，97–108。https: //doi.org/ 10 1109 48307 2020 00016 . /ICDE . . [58] Zhepei Wei、Jianlin Su、Yue Wang、Yuan Tian 和 Yi Chang。2020。一种用于关系三元组抽取的新型级联二元标注框架。载于第*58*届计算语言学协会年会论文集（*ACL*）。1476 1488 10 18653 1 2020 136 – . https://doi.org/ . /v / . [59] acl-main. Cathrin Weiss、Panagiotis Karras 和 Abraham Bernstein。2008。Hexastore：用于语义网数据管理的六元组索引。*VLDB*基金会会刊（*PVLDB*）1, 1 2008 10 14778 1453856 1453965 ( )。[60] https://doi.org/ . / . Dekun Wu、Nana Nosirova、Hui Jiang 和 Mingbin Xu。年份不详。一种通用 FOFE-net 框架，可用于知识库问答，简单而有效。计算研究知识库（*CoRR*）abs/1903.12356（2019）。arXiv:1903.12356 http://arxiv.org/abs/1903 12356 。Ledell Wu、Fabio Petroni、Martin Josifoski、[61] Sebastian Riedel 和 Luke Zettlemoyer。2020。通过稠密实体检索实现可扩展的零样本实体链接。载于自然语言处理实证方法会议（*EMNLP*）论文集。6397 6407 10 18653 1 2020 519 – . https://doi.org/ . / [62] v / .emnlp-main. Xuchen Yao 和 Benjamin Durme。年份不详。结构化数据上的信息抽取：使用 Freebase 进行问答。载于计算语言学协会年会论文集（*ACL*）。956–966。2014 https://doi.org/ 10 3115 1 14 1090 . /v /p - Pingpeng Yuan、Pu Liu、Buwen Wu、[63] Hai Jin、Wenya Zhang 和 Ling Liu。2013。TripleBit：一种用于大规模 RDF 数据的快速紧凑系统。*VLDB*基金会会刊（*PVLDB*）6，7（2013），517–528。https://doi.org/10.14778/ 2536349.2536352[64] Lei Zou、Ruizhe Huang、Haixun Wang、Jeffrey Yu、Wenqiang He 等。2014。RDF 上的自然语言问答：一种由图数据驱动的方法。载于国际数据管理会议（*SIGMOD*）论文集。313–324。https://doi.org/10.1145/2588555.2610525