arXiv:2303.07146v1 [cs.PL] 13 Mar 2023

NeuroQL：用于主体间推理的神经符号语言与数据集

Nick Papoulias OrgD. Labs 研究总监
https://orgdlabs.comnpapoylias@orgdlabs.com

摘要

我们提出了一项新的人工智能任务及其基线解决方案，用于主体间推理。我们将主体间信息定义为一种可能由不同参与方共享的客观信息与主观信息的混合体。例如，商品及其由信息检索系统报告的客观属性，可能需要与在线论坛中的主观用户评论交叉参照。要使人工智能系统能够成功地对两者进行推理，就必须将对客观事实的符号推理与主观用户评论中形成的共同共识相结合。为此，我们推出了 NeuroQL 数据集和领域专用语言，作为解决这一问题的基线方案。NeuroQL 是一种神经符号语言，它将神经原语融入逻辑合一，以实现抽取和检索。它可以作为自动翻译的目标，将以自然语言提出的主体间问题转换为能够回答这些问题的神经符号代码。

1 引言

我们的数字世界由客观性和主观性程度各异的信息源构成。存储在数据库或其他信息检索系统中的结构化信息（例如 例如，产品的物理属性和价格）会作为客观事实呈现给潜在客户，而与之相伴的自由文本评论，按定义则属于这一光谱中的主观部分。然而，作为这些系统的用户，我们关注的是同时涉及这两个方面的问题。这类问题既涉及产品权威性的结构化数据，也涉及公众的非结构化意见。例如，考虑以下搜索查询：“对于价格约*30*美元、评论不少于*1.4*万条且未停产的耳机，其低音表现如何？”我们在图 1中展示了这一查询：

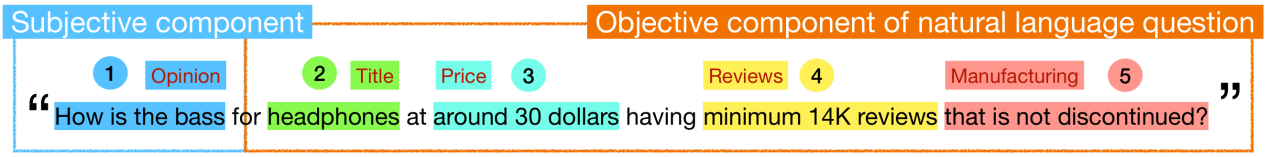


图1：一个主体间问题：“对于价格约*30*美元、评论不少于*1.4*万条且未停产的耳机，其低音表现如何？”。该问题的主观和客观成分分别包括：（即，观点、标题、价格、评论、制造等）。

我们将这类问题称为主观间问题，因为它们涉及主观信息与客观信息的混合，而这些信息可能由不同参与方共享。目前，信息系统通过半人工的两阶段流程处理此类查询。第一步，用户

需要通过仅涉及产品客观属性和描述的查询，锁定一款感兴趣的产品（例如涉及标题、价格、评论数量等的查询等等，如上例所示）。然后在第二步，用户需要手动浏览该产品的评论，或使用相似度搜索，以找到与产品质量相关的主观观点（例如上例中关于低音音质的观点）。随着深度学习[24, 17] 在自然语言处理领域 [48]，尤其是深度学习问答（问题与答案）模型 [46], 的发展，如今可以通过神经检索和神经理解来辅助这第二步的评论处理。

在这项工作中，我们探讨了自动化并合并这两个阶段的可能性，这两个阶段既涉及符号推理（针对结构化事实信息），也涉及神经元推理（针对非结构化主观来源）。在人工智能文献中，如何衔接符号推理与神经元推理，是神经符号系统所涉及的一个已知开放问题 [8, 32]。我们的贡献是为主体间查询与推理提出一项新的人工智能任务和基线解决方案，正如我们上文所见。

为此，我们提出 NeuroQL 数据集和领域专用语言 [14, 34]），作为解决这一问题的基线方案。我们的数据集扩展了以往针对问答系统的研究，这些研究关注元数据 [31] 和主观 [4]信息，并纳入主体间问题及其向神经符号查询的转换。我们的查询使用 NeuroQL 表达。NeuroQL 是一种神经符号语言，我们将其嵌入 Python [47]运行时环境中。这个嵌入式领域专用语言实现并扩展了逻辑合一 [41, 40]，并加入用于抽取和检索的神经原语。如图2所示，NeuroQL 可以作为自动翻译的目标，将以自然语言提出的主体间问题转换为能够回答这些问题的神经符号代码：

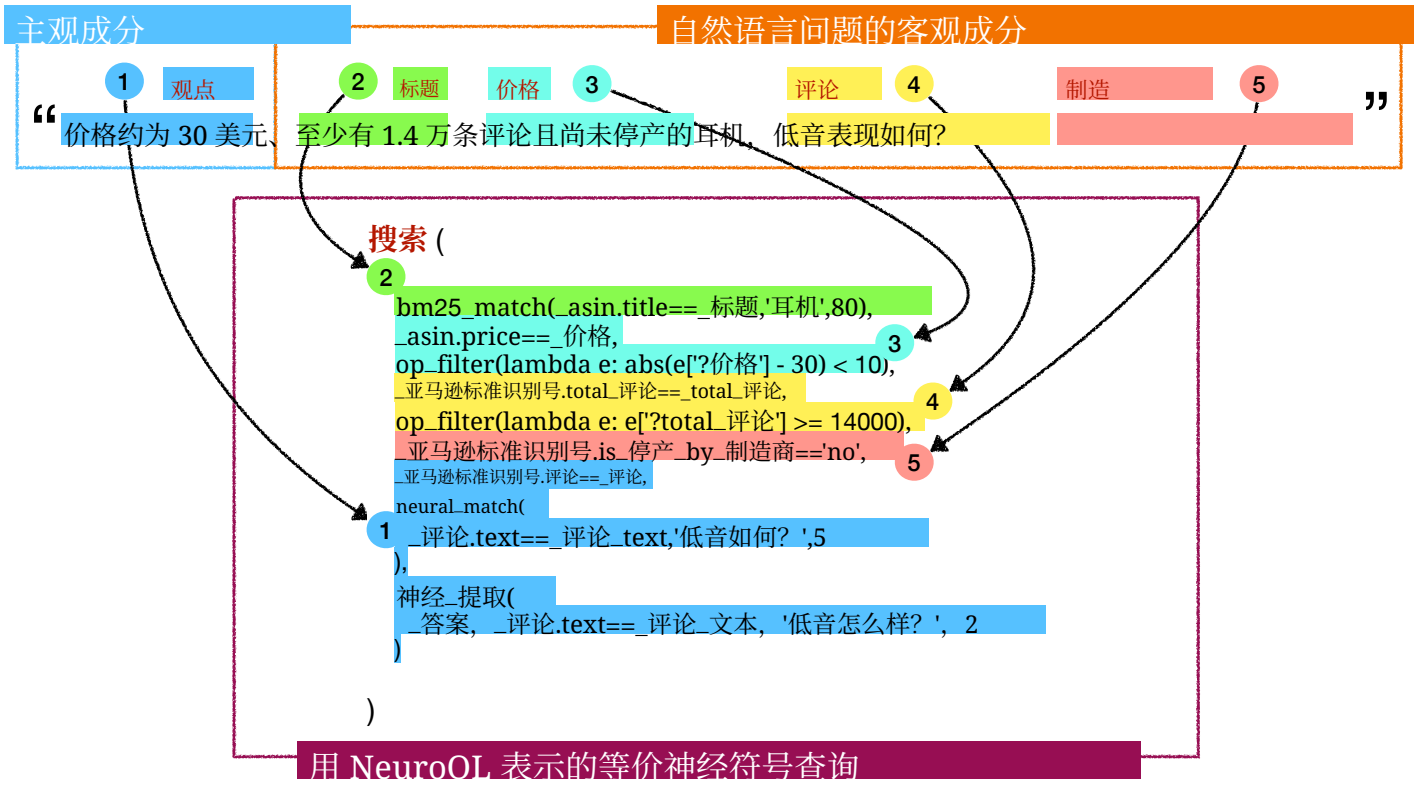


图 2：主体间问题“这款耳机的低音表现如何？价格约为 *30* 美元、评论数至少有 *1.4*万条且尚未停产？”及其转换为 NeuroQL 神经符号查询后的等价关系。从上方开始，我们展示了问题中的主观成分和客观成分及其对应类别（即 观点、标题、价格、评论、制造等）。在底部，我们可以看到 NeuroQL 中相应的搜索表达式，其中所有子查询均已突出显示，以便与自然语言中的对应类别相匹配。

In t本图展示了我们关于 NeuroQL 的目标和假设。首先，我们假设：

> (***H1***)：可以将主体间问题（如图2顶部所示）——这些问题包含不同的主观和客观成分——转换为以 *NeuroQL* 表示的神经符号查询。

为检验（***H1***），我们尝试采用针对该领域微调的神经网络翻译方案，不仅要区分客观成分和主观成分，还要区分不同的子查询类别（即：观点、标题、价格、评论、制造等）。这些子查询类别在 NeuroQL 中有不同的译法（如图2底部所示），并通过高亮标示，与自然语言中的对应类别相匹配。
此外，我们提出以下假设：

> (***H2***)：通过合一进行符号推理（涵盖结构化客观事实），类似于 *Prolog* *[9, 11]*和 *Datalog* *[6]*系统中的推理，可以有条理地结合神经元推理（用于处理非结构化主观数据），从而为主体间查询生成令人满意的答案。

这意味着，神经符号综合能够以简洁的句法和语义形式呈现给 NeuroQL 用户，同时仍能令人满意地回答主体间查询。为检验（***H2***），我们衡量了针对此前未见主体间问题的 NeuroQL 查询翻译结果的召回率、精确匹配和 *F1* 分数 [46]。

本文其余部分安排如下：第2 节介绍我们的研究成果，包括：NeuroQL 的实践演示（第2.1小节）、NeuroQL 推理管道的描述（第2.2小节）、NeuroQL 数据集的描述（第2.3小节）、我们的翻译、召回率、精确匹配和 F1 分数实验（第2.4小节），以及对相关工作的进一步评论（第2.5小节）。第3 节讨论我们这项贡献的意义、可能的效度威胁以及未来展望。最后，第4 节以相关的方法论评论总结全文。
本文中的所有数据和示例均可在以下网址获取：https://orgdlabs.com/neuroQL .

2 结果

2.1 NeuroQL 示例

您可以在 Python 环境中加载 NeuroQL 语言，例如我们为读者提供的环境，网址如下：https://orgdlabs.com/neuroql_例如，运行以下导入语句即可（见代码清单1）：

```
1来自 NeuroQL  导入 *2NeuroQL.加载('asin_key_properties.csv','asin_
                                                                    s
```

然后，如代码清单 1 第 2 行所示，您可以加载数据（例如，从 CSV 文件中加载），以填充知识库。本例中，我们在第 2 行加载产品属性和评论，其一般格式如下：

```
1 ... 2 B00001P4ZH . 标题 . Koss PortaPro 头戴式耳机（带收纳盒）
3 ]
H
量
号
```

清单 2：产品属性摘录

这些数据由一个产品标识符（如 B00001P4ZH）以及其对应的属性和值组成。在亚马逊网站产品目录中，这种标识符称为亚马逊标准识别号（asin） [2]。标识符后面是属性名称（如标题或价格）以及该属性的值（例如，上文第 3 行价格属性的值 39.36）。NeuroQL 能够加载任意长度、任意嵌套层级的 n元组；不过，为了简单起见，我们在此采用一种扁平表示，形式为：宾语.属性 = 值。这种表示为面向对象范式的程序员以及使用基于三元组的知识系统（主语、谓语、宾语） [3]的工程师所熟悉。如果需要显式加载新项目（而不是从预先保存的格式加载），可以按如下方式调用 NeuroQL 的事实原语：fact((’B00001P4ZH’,’price’,39.36))。或者，我们可以声明该标识符：B00001P4ZH = ids()，然后只需写出 B00001P4ZH.price == 39.36。

加载数据后，我们可以使用 NeuroQL 执行第一个简单查询，如下所示：

```
1 B00001P4ZH = 标识符()2 _属性，_值 = 变量()3 搜索(B00001P4ZH .
_
```

此处，在列表 3 中，我们在第 1 行定义了感兴趣的标识符，并在第 2 行声明了感兴趣的变量。在本例中，我们希望检索与特定标识符相关的所有可用属性及其值。为此，我们在第 3 行调用 搜索 NeuroQL 原语，它以一个 NeuroQL 表达式作为输入，并从知识库中返回满足该输入表达式的所有可能值组合。

因此，我们的搜索结果始终由一系列映射（例如字典）组成，这些映射将未知变量（见列表 4）绑定到满足表达式的值：

```
1 { ’? property ’: ’ 星级 ’ , ’? value ’: 4.7} , 2 { ’? property ’: ’ item_weight ’ , ’? value ’:
2.
va ’
评 ’?
pr
pr
va ’}
, 1
```

列表 4：按 ID 检索产品属性

这种绑定方式通过符号统一实现 [41, 40],，稍后我们将通过一组神经元原语对其进行扩展。

继续看我们的示例，NeuroQL 还允许从较简单的查询创建更复杂的查询，并提供了额外的组合方式，如列表5所示：

```
1 _亚马逊标准识别号，_总计_评论，_价格，_标题，2 _评论，_评论_文本，
_
题
( ]
3(
```

在第 1 和第 2 行，我们定义了一组变量，并希望找到它们的绑定（本例及后续示例均如此）。接着在第 4 至第 8 行，我们查询价格约为 30 美元的耳机的标识符。要用 NeuroQL 表达此查询，我们将多个表达式合取起来（将每个表达式作为单独的参数传递给我们的 搜索 原语）。第一个

这样的表达式会执行 *bm25* [39] 匹配所有产品标题，并返回排名前 80 的结果（第 5 行）。类似地，在第 6 行，我们查询将_价格 变量绑定到迄今为止找到的产品的价格。最后，在第 7 行，我们会筛选结果，只保留价格与目标价格相差不超过 10 美元的产品。

请注意，这里的 id（也就是 _亚马逊标准识别号）、产品_标题以及产品_价格，都是变量。这些变量是我们要求 NeuroQL 为我们绑定的。每个绑定都对应知识库中的特定值，这些值共同满足我们的表达式，从而得到以下结果（列出6）：

1 ... 2 { ' ?asin ': ' B00001P4ZH ' , 3 '? title ': ' Koss Porta Pro 头戴式耳机（带收纳盒） ', 4 '?
pı
机

列表 6：简单查询的输出示例（显示总计 10 个结果中的 2 个样例）

根据图中的完整示例逐步构建 1 和 2 ，现在我们可以提出以下查询（列表 7）：搜索价格约为 *30* 美元、至少有 *14K* 条评论且未被制造商停产的耳机：

1 搜索 (2 bm 25 匹配 (_亚马逊标准识别号 . 标题 == 标题 . ' 耳机 ' .8 0) .3 亚
马
,5
总
商

这是通过在我们先前的查询（列出5的查询）中添加第 5 至第 7 行来实现的，如查询列表7所示。描述一个_总计_条评论变量，以便在第 5 行将其绑定到（我们已检索到的）产品。然后在第 6 行根据评论总数进一步筛选结果。最后，在第 7 行添加额外限制，要求我们查找的产品不得已停产。这个扩展查询的结果见查询列表8：

1 { ' ?asin ': ' B00001P4ZH ' , 2 '? 标题 ': ' Koss PortaPro 头戴式耳机（带收纳盒） ', 3 '?
价
HI
BC
_ 计

2.1.1 神经符号组合

到目前为止，我们已经看到 NeuroQL 如何处理自然语言问题中的客观成分。现在，我们将扩展之前的示例（清单 7），以处理一个完整的

主体间问题，如图1 和 2中所述。具体来说，“一款售价约 *30* 美元、至少有 *1.4*万条评论且未停产的耳机，低音表现如何？”。为此，我们从扩展清单 9第 8 行中的示例开始，在那里我们将一个_评论变量绑定到迄今为止找到的产品的评论上。

然后，在第 9 行，我们调用神经原语 `neural_match`，并按以下算法步骤扩展经典合一：

> *1. neural_match* 将接收子查询 *_review.text==_review_text* 作为第一个参数。它将使用此子查询创建一组 *(id,* 文档*)* 配对（在本例中为*_review,_review_text* 配对），并在此过程中绑定变量 *_review_text*。*2.* 接着，它将尝试将 *_review_text* 的绑定与初始查询的主观成分进行匹配，在本例中即：“低音如何？”，这是我们传递给 *neural_match* 原语的第二个参数。在默认设置下，此匹配将通过 *DPR*（*Dense Passage Retriever*，密集段落检索器）*[21]*,完成，所用的问题与上下文编码器由 *Natural Questions* 数据集训练而成 *[25, 23]*。*3.* 最后，*neural_match* 将筛选迄今为止的所有查询绑定（截至列表 *9* 第 *11* 行），仅保留 *DPR _review_text* 匹配的前 *5* 个结果（使用第 *10* 行的第三个参数）。

1 搜索 (2 bm 25 匹配 (_亚马逊标准识别号 . 标题 == 标题 . ’ 耳机 ’ .8 0) .3 亚马

5

评

8

文

本

List第 9 行：通过筛选和扫描评论创建主体间查询，以回答一个特定的 r que问题

至此，我们已经找到了与问题匹配的前 5 条评论，这些评论来自满足客观约束的确切产品子集（截至第 11 行）。现在，我们可以使用神经_提取原语（清单9的第 12 至 14 行），从中提取与主体间问题相关的意见，具体如下：

> 1. 神经_提取首先会接收一个新变量的名称，以便将提取出的、符合目标问题的文本绑定到该变量（本例中该变量为_答案）。2. 然后，它会使用子查询_评论 . *text*==_评论_文本作为第二个参数传入，以创建一组与该子查询匹配的新（标识符、文档），如前所述。3. 接着，它会使用第三个参数（主观子组件“低音如何？”），尝试从_评论_文本文档中提取这个问题的答案。它将使用阅读器模型（例如 *MiniLM*）来完成此操作。该模型 *[49]* 最初基于 *SQuAD 2.0* 数据集进行训练*[37]*，之后又在 *NeuroQL* 训练集上进行微调，以提升性能（详见第*2.4*节）。4. 最后，神经_提取会筛选所有查询绑定，仅保留阅读器提取出的前 *2* 个结果（使用第 *13* 行的第四个参数）。

主体间查询的最终结果见列表10，其中展示了最终变量绑定的一个示例。这些结果现在同时包括客观（标题、价格、总_评论、制造）和主观（评论、答案）部分，它们共同满足了查询的约束。例如，对于产品B000AJIF4E（满足我们关于标题、价格、

总_评论、制造等。）最相关的观点是“低音很棒”，而对于产品B00001P4ZH，则是“低音果然很弱”。

1 { ’ ?亚马逊标准识别号 ’: ’ B000AJIF4E ’ , 2 ’? 标题 ’: ’ 索尼 MDR7506 专业大振膜耳机 ’ ,
3 ’
2 :
识
格
f f
询

我们通过展示如何在 NeuroQL 中定义和使用推理规则 [1]来完成这些示例。清单11 的第 1 至 6 行使用原语 `rule` 定义了一条推理规则。第 1 行的第一个参数是子查询（ `_asin.well_ranked == True`），它定义了当其余参数均满足时该规则能够推导出的结论。其余参数本身也是子查询，并构成一个合取。

每当该规则的头部与我们当前要绑定的子查询统一（例如我们搜索原语第 11 行中的子查询）时，就会测试这条规则。这意味着系统会进行嵌套搜索，尝试在当前绑定的基础上满足该规则中的合取，并在需要时返回新的绑定。

1 rule (_asin . well_ranked == True ,2 _asin . total_评论 == _total_
评
as

7 search (8 bm 25 匹配 (_亚马逊标准识别号 . 标题 == 标题 . ’ 耳机 ’ .8 0) .9
亚)
,11
停
15
18

列表 11：在搜索过程中加入一条用于推理排名靠前的产品的规则

用通俗易懂的话来说，这条规则指出，如果一款产品的评论数至少为 20K，且评分至少为 4.0 星级，那么它就属于排名靠前的产品。在列表 11 的第 11 行应用这条规则，而不是采用我们之前不那么严格的标准（位于列表9的第 5 和第 6 行），会得到更受约束的结果（见列表 12）。

1 . . .

[1]关于 NeuroQL 中推理规则语法和语义的形式化模型，将在后续论文中介绍。

2 { ’?亚马逊标准识别号’: ’ B000AJIF4E ’ , 3 ’? 标题 ’: ’ 索尼 MDR7506 专业大振膜耳机 ’ , 4 ’? 价
8
前

2.2 NeuroQL 架构

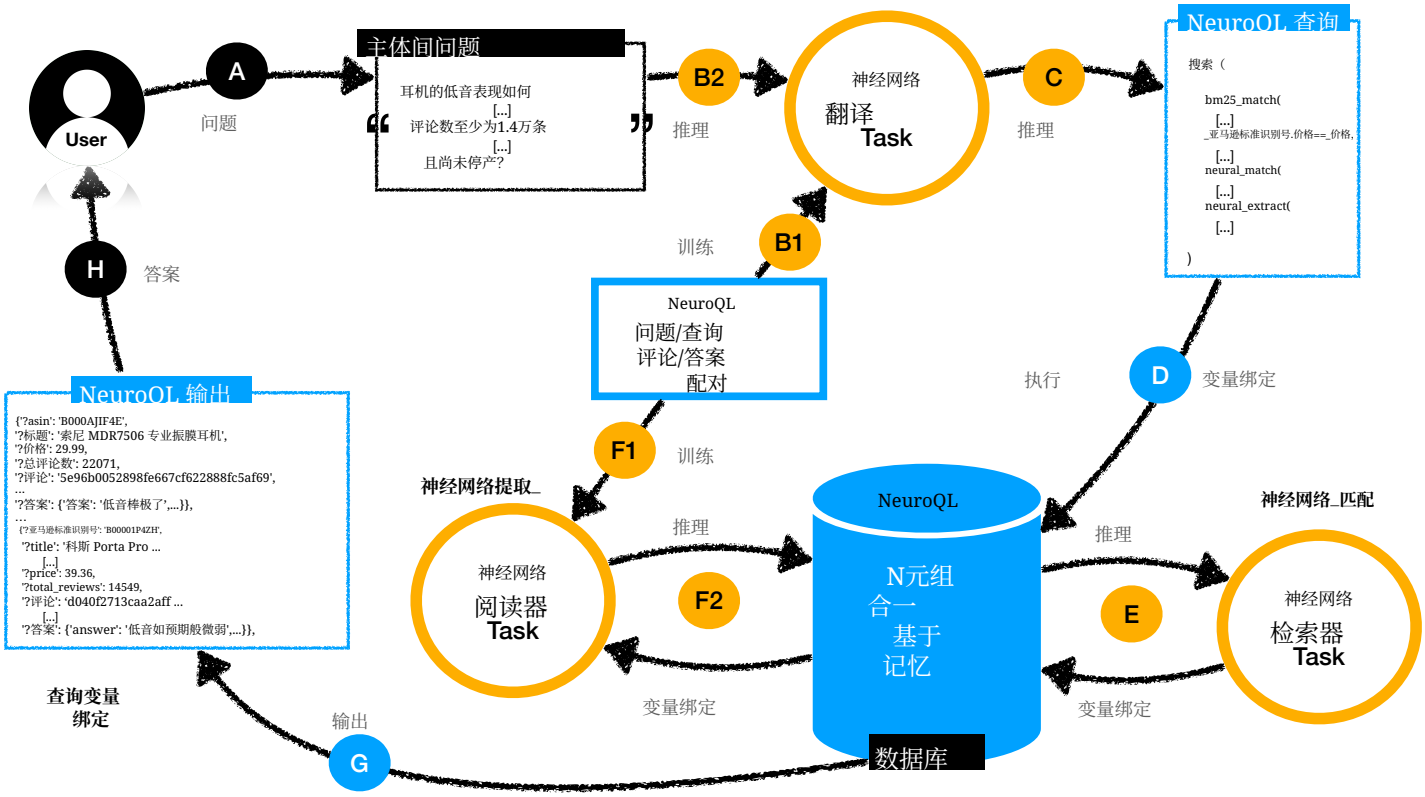


图 3：NeuroQL 架构：从一个主体间问题出发，NeuroQL 架构首先使用针对该任务微调的神经翻译网络，推断出等效的 NeuroQL 查询。随后，系统会对 NeuroQL 数据库执行该查询。该数据库结合符号方法（通过合一）和神经推理（通过检索器和阅读器网络），逐步绑定查询变量，并向用户返回答案。

我们的两个主要假设（详见第 1 节）是：（***H1***）可以将主体间问题从自然语言自动翻译为 NeuroQL；以及（***H2***）NeuroQL 可以通过神经推理简洁地扩展合一，从而为主体间问题生成令人满意的答案。为了检验（***H1***）和（***H2***），我们设计了以下架构（见图 3），作为主体间问答任务的基线解决方案。从图 3的左上角开始，我们看到用户以自然语言提交了一个主体间问题 。在推理过程中，该问题 将被传递给翻译模型 研究中使用的是经过微调的 eT5 模型 [51]），该模型将尝试把问题翻译为 NeuroQL 查询 。翻译模型针对这一下游任务，使用 NeuroQL 问题/查询数据集进行训练 （详见第 2.3

和 2.4 节）。随后，我们的架构将尝试执行该查询 ；正如第 2.1 节所 这意味着在 NeuroQL 数据库中找出满足查询主合取的所有可能绑定。NeuroQL 数据库本身是一个内存中的 n 元组存储，其上应用了我们扩展的合一算法。这些扩展包括 神经_匹配 以及 神经_提取 基本 作。这些基本操作基于一个检索器模型（在我们的例子中是一个 DPR [21] 使用自然问题数据集训练 [25, 23]）和一个阅读器模型（一个 MiniLM [49] 最初在 SQuAD 2.0 数据集 [37]上训练）。使用这些模型，我们的合一引擎将尝试进一步绑定并筛选逻辑合取中的神经子查询（正如我们在第 2.1.1 节中所见）。在训练期间，我们使用 NeuroQL 训练集 用我们的评论/答案配对）对阅读器进行进一步微调，以提升其性能（详见第 节）。最后，当找到所有可能的绑定后，系统将向用户返回一个答案。

清单13 和 14 逐步展示了如何先将自然语言问题转换为 NeuroQL（清单 13 的第 1 至 3 行），调用我们的基本操作NeuroQL。translate。接着，展示如何动态评估生成的查询（清单 14 的第 1 行）以获得结果（使用 NeuroQL。eval ）。两个示例的匹配结果都与清单 9 和 10 中的先前结果一致。最后，清单 15展示了如何通过调用我们的基本操作NeuroQL。answer，在单个步骤中完成转换和评估（清单 15 的第 1 行）。

1 query =NeuroQL。translate (2 ’ 耳机的低音怎么样，价格在[...] 左右？ ’ 3) 4
pr
# #
pr
as
ws
no
=
re
# #
转

1 NeuroQL . eval (查询) 2 3 输出 : # # #

```
14 '?价格 ': 39.36 ,
15 '? 总计_条评论': 14549,
16 '? 评论 ': 'd040f2713caa2aff0ce95affb40e12c2', 17 ...18 '? 答案
':
##
```

1 NeuroQL . 答案 (2 '对于价格约为[...] 的耳机，低音表现如何? ' 3) 4 5 输出：# # # #
♯ ':
' B
评
12

```
14 {'?亚马逊标准识别号 ': 'B00001P4ZH',
15 '? 标题 ': ' Koss PortaPro 头戴式耳机（带收纳盒） ', 16 '?价格 ': 39.36 , 17 '?
总.
19
# ♯
#
```

清单 15：在单个 ca 中结合主体间问题的翻译与评估 ll

2.3 NeuroQL 数据集

NeuroQL 数据集扩展了以往专注于元数据 [31] 和主观 [4] 信息的问答系统研究，将内容扩展至*1505* 个主体间问题，以及将这些问题转换为神经符号查询（例如列表16）。这些（问题、查询）配对与一个包含*4250* 个属性的详细知识库相结合（例如列表2），涵盖*500* 种不同产品，包括*1583*条评论以及*1627* 条用于问答抽取的真实标注。

1 B00001P4ZH，问题，0514 ee34 ...
2 0514 ee34 ...，文本，耳机型号 6303157 ...3 0514 ee34 ...，查询，" 搜索(4 bm25_
匹
酝

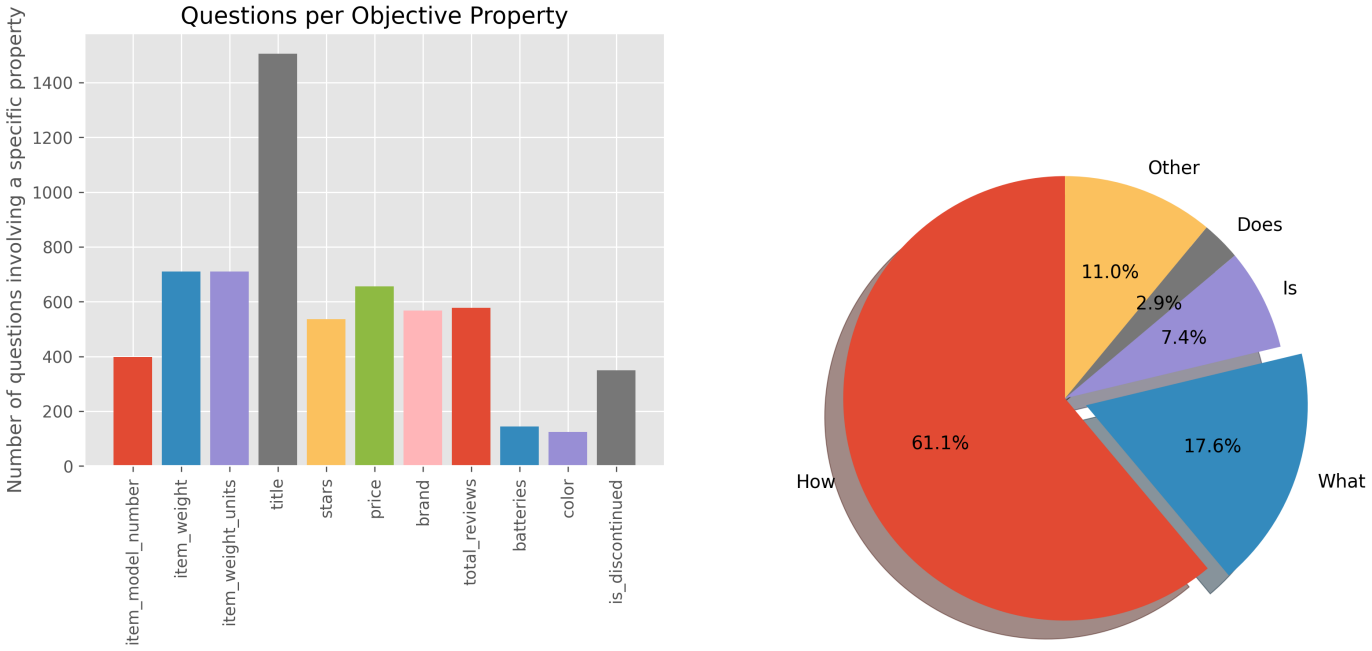
表1 列出了我们纳入数据集的所有客观属性类别，并为每个类别提供了简短描述。每个类别后面都附有其取值范围或一个示例值，供参考。最后，图4 展示了数据集中涉及特定客观属性的问题分布（图4左侧）。右侧（与 [46]类似）则给出了主体间问题中主观成分首词的统计数据。

2.4 实验与验证

NeuroQL 数据集支持两种不同的实验任务。第一种是*NeuroQL* 翻译任务，目标是将以自然

产品属性		
Name	描述	示例值
标题	产品在网上显示的标题	例如。 *Apple Magic Mouse*
品牌	产品品牌	例如。铁三角
商品型号_ _	制造商序列号或其他 ID	例如。*6229a003aa*
价格	产品在线上公布的价格	范围为 $[1.95, 999.0]$
星级	产品的平均评分	范围为 $[2.8, 5]$
评论总数_	产品的评论总数	范围为 $[2, 134717]$
颜色	产品在网上标注的颜色	例如 紫水晶灰
商品_重量	在线报告的产品价格	范围为 $[0.01, 73.2]$
商品_重量_单位	商品_重量的单位	$\in \{'kilograms', 'ounces', 'pounds'\}$
电池	电池数量	在范围 $[1, 3]$ 内
已停产_	持续生产	$\in \{'no', 'yes'\}$

表 1：NeuroQL 数据集：客观属性的类型、描述和示例值



（a）每项客观属性的问题数量　　(b) 主体间问题的主观成分统计

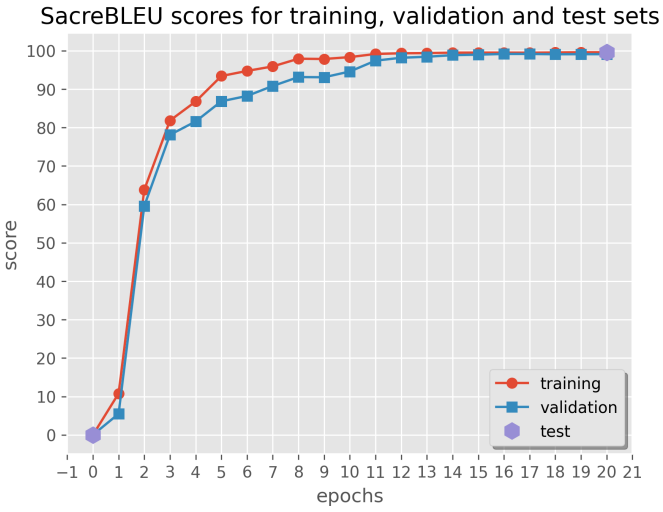
图 4：NeuroQL 数据集中属性和主观成分的细分情况

语言提出的主体间问题翻译为能够回答这些问题的神经符号代码。该任务考虑（问题、查询）配对，并可使用*sacreBleu* [36]以及机器翻译任务中常用的四元组或三元组精确率等指标进行评估 [46]。

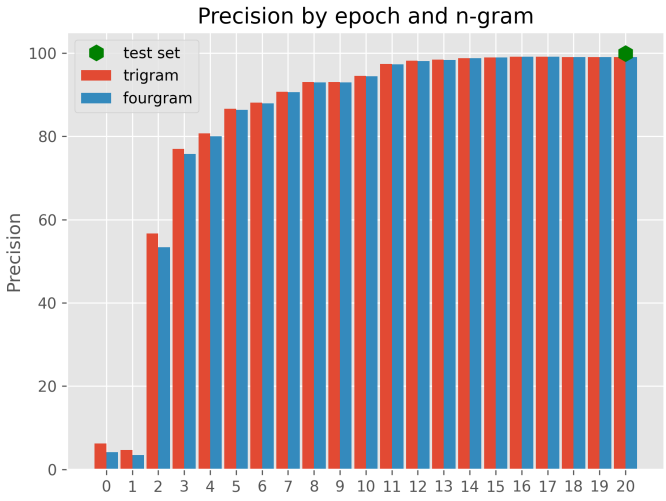
第二项任务是*NeuroQL* 查询任务，其目标是微调神经原语，例如神经_抽取子查询，以尽可能生成准确的结果。该任务会考虑（查询，答案）配对，并可使用召回率、精确匹配和 *F1* 分数等问答抽取任务中常用的指标进行评估 [46]。

对于我们的*NeuroQL* 翻译任务，我们对 CodeT5 模型进行微调 [51]，训练*20* 个训练轮次，旨在将主体间问题转换为 NeuroQL 代码。我们将最大输入/输出长度设为*512* 个词元，以适配我们的（问题，查询）配对；批次大小为*64*，*1e-4* 学习率，采用线性调度器和*AdamW* 优化器 [28]。如图5左侧所示，模型在第 11 个训练轮次之前显著改善（且未出现过拟合）；此后

收益逐渐递减，到第 20 个训练轮次时，训练集和验证集的 sacreBleu 分数都接近完美。第 20 个训练轮次后的测试集评估证实了模型的性能。我们将数据集划分为训练集（*80%*）、验证集（*10%*）和测试集（*10%*），每个子集均使用唯一的产品 ID，以确保不存在数据泄漏。图5 右侧报告的四元组和三元组精确率与我们观察到的 sacreBleu 结果一致。鉴于这些结果，我们关于将主体间问题转换为神经符号查询可行性的初始假设（***H1***）已得到验证。



（a）将以自然语言提出的主体间问题转换为 NeuroQL 代码时的 sacreBleu 分数（训练集、验证集和测试集）。



（b）将主体间问题转换为 NeuroQL 时，我们的验证集上每个训练轮次的四元组精确率和三元组精确率。右上角还报告了训练后测试集的四元组精确率。

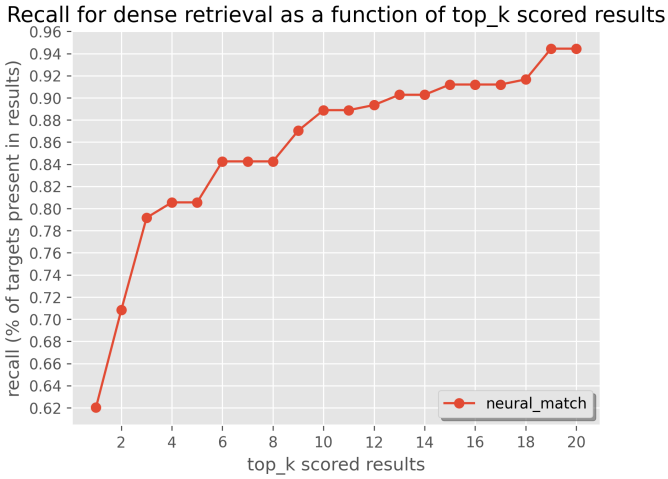
F图 5：NeuroQL 翻译任务的实验结果，包括 sacreBleu、四元组精确率和三元组精确率指标。

对于 NeuroQL 查询任务，我们首先针对检索器返回的不同 top_k 结果数，测量使用 Natural Questions 数据集 [25, 23]）训练的 DPR [21] 模型对测试集的影响。随后，我们对阅读器模型进行微调（该模型为 MiniLM [49]，最初使用 SQuAD 2.0 数据集 [37]训练），以提升其在阅读器返回不同 top_k 结果数时的表现。从图 6 左侧可以看到，即使只返回 top_*2* 结果，我们的 DPR 模型在召回率达到 70% 时表现也相当不错。不过，直到返回 top_*13*结果后，召回率才超过 90%；在 top_*20*时，召回率达到 94%。利用检索器的这一最高召回率，我们随后评估了微调后的阅读器模型。正如预期，随着 top_k 增大，精确匹配和 F1 分数均有所提升。不过，观察到的最佳值分别为精确匹配 0.20 和 F1 分数 0.33，二者均出现在返回 top_*8* 结果时，与 [46], 使用相同模型报告的用户评论抽取式问答分数相当。

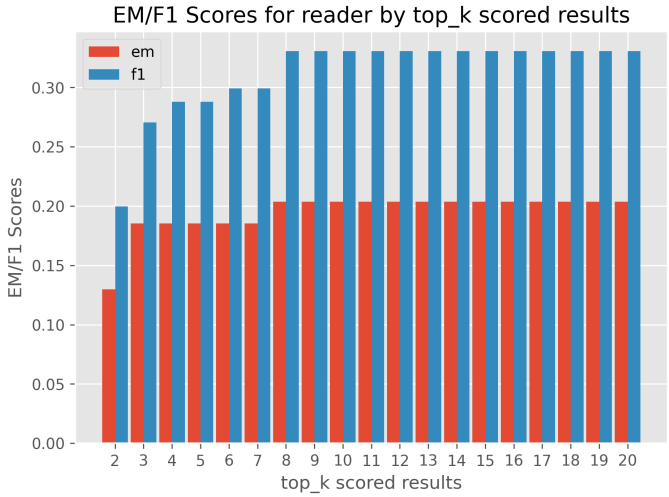
在进行此次评估之前，我们对 MiniLM 进行了微调 [49] 阅读器模型，持续*3* 个训练轮次，以提高其抽取准确率。我们使用了*384*词元的序列长度，采用*128*词元的文档步长，批次大小为*16*；模型在预测结果时也可以返回无答案。我们将学习率设为*1e-5*，并设置*0.2* 的预热，使用与之前相同的数据集划分。鉴于上述结果，我们关于通过神经原语扩展合一来回答主体间问题是否可行的初始假设（***H2***）仅得到部分验证。本文所述的神经符号合一确实可用于执行主体间神经符号查询。然而，这些结果仅构成一个初步基线。要提高抽取答案的准确率，仍需开展更多工作，尤其是改进神经_抽取基本操作。

2.5 相关工作

有大量研究致力于将自然语言问题转换为成熟的符号语言（如 SQL）中的查询 [22, 5, 53, 52]和 SPARQL [54, 42]。这些解决方案



（a）不同 top_k 个评分结果下稠密检索器（即神经_匹配）的召回率分数。



（b）不同 top_k 个评分结果下阅读器（即神经_抽取）的精确匹配和 F1 分数。

图6：我们在 NeuroQL 查询任务上的实验结果，包括 DPR 和阅读器模型的召回率、精确匹配和 F1 分数。

只有一个用于翻译的神经元组件，其流程是：神经网络翻译 ▷符号代码▷ 符号推理。我们的贡献更进一步，将自然语言转换为神经符号代码，打通符号推理与神经元推理之间的闭环：神经网络翻译▷神经符号代码 ▷ 神经符号推理。

Deep-prolog [30] 和 TensorLog [10] 是另外两种神经符号和逻辑导向的编程语言，旨在整合神经元推理与符号推理。两者都依赖概率逻辑，因此尝试将神经原语作为带概率的谓词加以整合，而这些概率可由神经网络参数化。同样，Scallop [19] 针对知识图谱采用加权图方法，其中实体之间的权重是学习得到的。最后，SQLFlow [50] 关注的不是执行，而是使用类 SQL 语言为机器学习工作负载编排任务（借助 Kubernetes）。在我们的工作中，神经元推理与符号推理的整合旨在实现执行，其语义层级高于概率方法。在 NeuroQL 中，推理任务（例如神经元抽取和匹配）被表示为用于合一的筛选和绑定操作 [41, 40]。这使我们能够创建一种简洁的神经符号查询语言，可作为自然语言问题自动翻译的目标语言。

许多用于神经符号编程的领域特定语言 [8] 面向特定领域，例如符号回归 [12], 行为分析 [45] 和程序合成 [13]。我们是首个使用神经翻译解决主体间推理问题的工作，拓展了关于元数据 [31]和主观性的先前研究 [4]。

最后，我们的工作还与更广泛的领域相关，即检索增强 [27, 44, 16] 和工具增强[43, 7, 26] 的答案抽取与生成。不过，在我们的案例中，我们为任务提出了一种丰富的中间表示，其形式为神经符号代码。这使我们能够以可解释且明确的方式整合神经推理与符号推理。

3 讨论

我们认为，将自然语言转换为神经符号代码，而非仅仅转换为符号代码，是朝着打通神经元推理与符号推理之间闭环迈出的一步。我们在本工作中提出的主体间推理任务、数据集和领域专用语言，提供了一个贴近现实需求且具有鲜明神经符号特征的案例研究。因此，我们认为它可作为评估未来工作的基线。

我们关于假设（***H1***）的研究结果*NeuroQL* 翻译任务令人意外。对于这项任务，网络不仅需要学会区分客观成分和主观成分，还需要区分不同的子查询类别（即，观点、标题、价格、评论、制造等），如图2所示。鉴于这项任务的 sacreBleu 分数近乎完美，我们认为，未来版本的数据集需要放宽封闭世界假设（即所有查询都涉及产品及其客观或主观属性这一前提）。例如，我们可以考虑扩展数据集，纳入差异足够大的探究领域，例如新闻文章及其评论。此外，自动改写 [15]我们的问题，以创建规模更大、更多样化的数据集，也是未来工作的一个有前景方向。

关于 ***H2*** 假设的结果，对于*NeuroQL* 查询任务而言，与我们的预期更为接近，因为这些结果与 [46] 报告的、针对用户评论的较简单抽取式问答任务的结果相当。尽管这些结果可作为基线，但显然仍需做更多工作来提高准确率。
关于可能的效度威胁，我们注意到，问题中部分措辞直接取自真实场景用法，未经编辑。乍看之下，这是数据集的一项优势（也就是说，模型需要学会适应真实场景噪声）。然而，存在句法或语义错误的示例（例如，非母语者所写的内容）也会影响标注质量，使有效答案更难确定。在这方面，自动释义改写 [15]和自动纠错方法或许也能帮助我们控制并衡量语言连贯性对数据集的影响。

4 方法

领域专用语言： 我们将 NeuroQL 实现为一种领域专用语言 [14, 34] ，使用反射和元编程 [29, 35], ，使我们能够仅针对 NeuroQL 表达式更改 Python 的常规语义（运行时的其余部分保持不变）。这些更改包括*(i)* 重新定义变量声明（使用 `vars()` 和 `ids()` 基本操作时）*(ii)* 重新定义 NeuroQL 变量和 NeuroQL 标识符的点运算符、赋值运算符和相等运算符。这些更改使 NeuroQL 成为一种嵌入式混合语言，符合 [38] 中提供的领域专用语言分类。

合一： 我们的基础合一算法 [41, 40] 是对模式匹配 [1, 18], 的泛化实现，其中模式和目标都可以包含变量。该泛化算法接收一组绑定作为输入（通常称为框架），并返回可能的扩充，其中包含尚未解析变量的匹配值。通过允许联合或并行匹配子查询，我们可以实现合取和析取等逻辑运算。随后，可以通过嵌套模式匹配实现基于推理规则的合一：在当前绑定的基础上，尝试满足规则体。

BM25： 在我们的工作中，我们对 BM25（最佳匹配25）算法稍作调整，以解决原始算法 [20] 中最常见的“完全重叠”问题。BM25 本身是对词频-逆文档频率（TF-IDF）的改进 [20], ，这两种算法都用作稀疏检索器的排序方法。在我们的方案中，我们会对待匹配的文档和查询进行预处理，包括分词和词干提取。由于在这种情况下可以使用准确度更高的 DPR 基元（见下文），我们没有针对 BM25 中的同义词问题提供进一步的解决方案。这两种方法（BM25 和 DPR）在速度和准确率之间为用户提供了权衡：BM25 速度最快，而 DPR 准确率最高。

稠密段落检索：DPR（稠密段落检索）是一种使用稠密文本嵌入来表示待匹配的查询和文档的方法。它基于 BERT 双编码器架构，通过计算文档与查询之间的点积相似度来进行匹配。我们的 DPR 基于 [21], 使用以 Natural Questions 数据集训练的问题与上下文编码器 [25, 23]。随后，我们将此 DPR 用作 神经_匹配原语的后端。神经_匹配原语接收一个子查询，并利用其结果创建一组(标识符, 文档)配对。随后，它使用我们的 DPR 将这些文档与目标查询文本进行匹配，并返回排名最高的_个匹配结果（完整示例见神经_匹配一节：2.1.1）。

阅读器模型： 阅读器或阅读理解模型是一种能够通过返回文档中相关文本区间来执行抽取式问答的神经网络。在我们的工作中，我们使用最初在 SQuAD 2.0 数据集 [37] 上训练、随后在 NeuroQL

训练集上进一步微调以提升性能的模型（详见第 2.4 节）。随后，该阅读器被用作我们 神经_抽取 原语的后端。该 神经_抽取 原语接收一个新变量的名称，用于绑定抽取出的答案；接收一个查询，用于创建 *(id, document)* 配对；最后接收目标查询文本。然后，它使用阅读器从我们的文档中抽取相关文本区间，并返回找到的前_个结果。我们对阅读器模型进行了 *3* 个训练轮次的微调，使用 *384* 个词元的序列长度，文档步长为 *128* 个词元。批次大小设为 *16*，学习率为 *1e-5*，并采用*0.2* 的预热；同时支持在预测结果时返回无答案的情况。

翻译模型： 翻译模型是一种序列到序列神经网络，通过输入序列与目标序列的配对进行训练。在我们的研究中，我们使用 NeuroQL 问题/查询数据集对 CodeT5 模型进行微调 [51]），以便将用自然语言提出的主体间问题转换为能够回答这些问题的 NeuroQL 查询（详见第 2.3 节和第 2.4节）。随后，该翻译模型将作为我们 NeuroQL。translate和 NeuroQL。answer基本操作的后端。顾名思义，这些基本操作能够将问题转换为 NeuroQL 代码，并尝试回答这些问题。我们的翻译模型经过了 *20* 个训练轮次 的微调，输入/输出的最大长度为 *512* 个词元，批次大小为 *64* ，学习率为 *1e-4* ，使用 线性 调度器和 *AdamW* 优化器 [28]。

SacreBleu、召回率、精确匹配和 ***F1*** 分数： 在机器翻译任务中，***sacreBleu*** [36] 分数用于评估生成的译文，重点关注所报告结果的可复现性和可比性。它本身是对 Bleu [33] 分数的标准化，该分数通过比较生成序列与参考译文中的 n 元语法来进行评估。在我们的工作中，我们使用 *sacreBleu* 评估翻译模型的质量（详见第 2.4节）。 召回率： 这是一项用于评估检索方法的指标，例如我们采用 DPR 模型的 神经_匹配基本操作。召回率表示检索器返回的前_k 个结果中相关文档所占的比例（详见第 2.4节）。 精确匹配和 ***F1*** 分数： 用于评估问答抽取方法，例如我们采用阅读器模型的 神经_抽取 基本操作。精确匹配表示精确抽取匹配所占的比例，而 F1 分数则衡量精确率与召回率的调和平均数（详见第2.4节）。

代码与数据可用性

本文中的所有数据和示例均可在此获取：https://orgdlabs.com/neuroQL

参考文献

[1] H. Abelson 和 G. J. Sussman。计算机程序的构造和解释。麻省理工学院出版社，1996年。[2] P. Bausch。*Amazon Hacks*。O'Reilly Media, Inc.，2003年。[3] T. Berners-Lee、D. Connolly、L. Kagal、Y. Scharf 和 J. Hendler。N3logic：万维网的逻辑框架。逻辑编程理论与实践，8(3)：249–269，2008年。[4] J. Bjerva、N. Bhutani、B. Golshan、W.-C. Tan 和 I. Augenstein。Subjqa：一个用于主观性和评论理解的数据集。*arXiv* 预印本 *arXiv:2004.14283*，2020年。[5] U. Brunner 和 K. Stockinger。Valuenet：一种从数据库信息中学习的自然语言转 SQL 系统。载于*2021*年 *IEEE* 第 *37* 届数据工程国际会议（*ICDE*），第 2177–2182 页。IEEE，2021年。[6] S. Ceri、G. Gottlob、L. Tanca 等。关于 Datalog，你一直想知道却从未敢问的一切。*IEEE* 知识与数据工程汇刊，1(1)：146–166，1989年。[7] H. Chase。Langchain 文档，2023年。网址：https://langchain.readthedocs.io/en/latest/。[8] S. Chaudhuri、K. Ellis、O. Polozov、R. Singh、A. Solar-Lezama、Y. Yue 等。神经符号编程。《编程语言基础与趋势》，7(3)：158–243，2021年。

[9] W. F. Clocksin 和 C. S. Mellish。Prolog 编程。Springer Science & Business Media，2003年。[10] W. Cohen、F. Yang 和 K. R. Mazaitis。TensorLog：一种使用深度学习基础设施实现的概率数据库。《人工智能研究杂志》，67：285–325，2020年。[11] A. Colmerauer。Prolog III 简介。《ACM 通讯》，33(7)：69–90，1990年。[12] M. Cranmer、A. Sanchez Gonzalez、P. Battaglia、R. Xu、K. Cranmer、D. Spergel 和 S. Ho。利用归纳偏置从深度学习中发现符号模型。《神经信息处理系统进展》，33：17429–17442，2020年。[13] K. Ellis、C. Wong、M. Nye、M. Sable-Meyer、L. Cary、L. Morales、L. Hewitt、A. Solar-Lezama 和 J. B. Tenenbaum。Dreamcoder：通过醒睡式贝叶斯程序学习，积累可泛化、可解释的知识。arXiv 预印本 arXiv:2006.08381，2020年。[14] M. Fowler。领域专用语言。培生教育，2010年。[15] Y. Fu、Y. Feng 和 J. P. Cunningham。利用潜在词袋生成释义。《神经信息处理系统进展》，32，2019年。[16] D. GmbH。Haystack 文档，2023年。网址 https://docs.haystack.deepset.ai/ docs。[17] I. Goodfellow、Y. Bengio 和 A. Courville。深度学习。麻省理工学院出版社，2016年。[18] C. Hanson 和 G. J. Sussman。灵活性软件设计：如何避免把自己逼入编程死角。麻省理工学院出版社，2021年。[19] J. Huang、Z. Li、B. Chen、K. Samel、M. Naik、L. Song 和 X. Si。Scallop：从概率演绎数据库到可扩展的可微推理。《神经信息处理系统进展》，34：25134–25145，2021年。[20] D. Jurafsky。语音与语言处理，第 3 版（草稿）。https://web.stanford.edu/ juraf- sky/slp3/，2023年。[21] V. Karpukhin、B. Oˇguz、S. Min、P. Lewis、L. Wu、S. Edunov、D. Chen 和 W.-t. Yih。用于开放域问答的稠密段落检索。arXiv 预印本 arXiv:2004.04906，2020年。[22] H. Kim、B.-H. So、W.-S. Han 和 H. Lee。自然语言转 SQL：我们目前进展如何？《VLDB 汇刊》，13(10)：1737–1750，2020年。[23] T. Kwiatkowski、J. Palomaki、O. Redfield、M. Collins、A. Parikh、C. Alberti、D. Epstein、I. Polosukhin、J. Devlin、K. Lee、K. Toutanova、L. Jones、M. Kelcey、M.-W. Chang、A. M. Dai、J. Uszkoreit、Q. Le 和 S. Petrov。自然问题：问答研究的一个基准测试。《计算语言学协会汇刊》，7：452–466，2019年。doi：10.1162/tacl_a_00276。网址 https://aclanthology.org/Q19-1026。[24] Y. LeCun、Y. Bengio 和 G. Hinton。深度学习。《自然》，521(7553)：436–444，2015年。[25] K. Lee、M.-W. Chang 和 K. Toutanova。用于弱监督开放域问答的潜在检索。载于第 57 届计算语言学协会年会论文集，第 6086–6096 页，意大利佛罗伦萨，2019年7月。计算语言学协会。doi：10.18653/v1/P19-1612。网址 https://aclanthology.org/P19-1612。[26] Y. Levine、I. Dalmedigos、O. Ram、Y. Zeldes、D. Jannai、D. Muhlgay、Y. Osin、O. Lieber、B. Lenz、S. Shalev-Shwartz 等。站在巨型冻结语言模型的肩膀上。arXiv 预印本 arXiv:2204.10019，2022年。[27] P. Lewis、E. Perez、A. Piktus、F. Petroni、V. Karpukhin、N. Goyal、H. Küttler、M. Lewis、W.-t. Yih、T. Rocktäschel 等。面向知识密集型自然语言处理任务的检索增强生成。《神经信息处理系统进展》，33：9459–9474，2020年。[28]I. Loshchilov 和 F. Hutter。解耦权重衰减正则化。arXiv 预印本 arXiv:1711.05101，2017年。

[29] P. Maes。计算反射中的概念与实验。*ACMSigplanNotices*，22(12)：147–155，1987年。[30] R. Manhaeve、S. Dumancic、A. Kimmig、T. Demeester 和 L. De Raedt。DeepProbLog：神经概率逻辑编程。神经信息处理系统进展，31，2018年。[31] J. Ni、J. Li 和 J. McAuley。利用远程标注评论和细粒度方面为推荐提供依据。载于*2019*年自然语言处理实证方法会议暨第九届自然语言处理国际联合会议（*EMNLP-IJCNLP*）论文集，第188–197页，2019年。[32] M. Nye、M. Tessler、J. Tenenbaum 和 B. M. Lake。通过双系统神经符号推理提升神经序列模型的连贯性与一致性。神经信息处理系统进展，34：25192–25204，2021年。[33] K. Papineni、S. Roukos、T. Ward 和 W.-J. Zhu。BLEU：一种机器翻译自动评估方法。载于计算语言学协会第*40*届年会论文集，第311–318页，2002年。[34] N. Papoulias。LANDS 平台：语言与领域语法。载于第*12*届 *ACM SIGPLAN* 软件语言工程国际会议论文集，第66–70页，2019年。[35] N. Papoulias、M. Denker、S. Ducasse 和 L. Fabresse。元控制的终端用户抽象：将反射图具象化。计算机程序科学，140：2–16，2017年。[36] M. Post。呼吁清晰报告 BLEU 分数。*arXiv* 预印本 *arXiv:1804.08771*，2018年。[37] P. Rajpurkar、R. Jia 和 P. Liang。了解自己不知道什么：SQuAD 中的不可回答问题。*arXiv* 预印本 *arXiv:1806.03822*，2018年。[38] L. Renggli。统一工具支持下的动态语言嵌入。Lukas Renggli，2010年。[39] S. Robertson、H. Zaragoza 等。概率相关性框架：BM25 及其进展。信息检索基础与趋势，3(4)：333–389，2009年。[40] A. J. Robinson 和 A. Voronkov。自动推理手册，第1卷。爱思唯尔，2001年。[41] J. A. Robinson。一种基于归结原理的面向机器的逻辑。*ACM* 学报（*JACM*），12(1)：23–41，1965年。[42] I. Saparina 和 A. Osokin。根据中间问题分解生成 SPARQL 数据库查询。*arXiv* 预印本 *arXiv:2109.06162*，2021年。[43] T. Schick、J. Dwivedi-Yu、R. Dessì、R. Raileanu、M. Lomeli、L. Zettlemoyer、N. Cancedda 和 T. Scialom。Toolformer：语言模型可以自学使用工具。*arXiv* 预印本 *arXiv:2302.04761*，2023年。[44] D. Singh、S. Reddy、W. Hamilton、C. Dyer 和 D. Yogatama。面向开放域问答的多文档阅读器与检索器端到端训练。神经信息处理系统进展，34：25968–25981，2021年。[45] J. J. Sun、A. Kennedy、E. Zhan、D. J. Anderson、Y. Yue 和 P. Perona。任务编程：学习数据高效的行为表征。载于*IEEE/CVF* 计算机视觉与模式识别会议论文集，第2876–2885页，2021年。[46] L. Tunstall、L. Von Werra 和 T. Wolf。使用 *Transformer* 进行自然语言处理。O’Reilly Media, Inc.，2022年。[47] G. Van Rossum。*Python* 语言参考。Python 软件基金会，2023年。[48] A. Vaswani、N. Shazeer、N. Parmar、J. Uszkoreit、L. Jones、A. N. Gomez、Ł. Kaiser 和 I. Polosukhin。注意力就是你所需要的一切。神经信息处理系统进展，30，2017年。

[49] W. Wang、F. Wei、L. Dong、H. Bao、N. Yang 和 M. Zhou。MiniLM：面向任务无关压缩的预训练 Transformer 深度自注意力蒸馏。神经信息处理系统进展，33：5776–5788，2020年。[50] Y. Wang、Y. Yang、W. Zhu、Y. Wu、X. Yan、Y. Liu、Y. Wang、L. Xie、Z. Gao、W. Zhu 等。SQLFlow：连接 SQL 与机器学习的桥梁。*arXiv* 预印本 *arXiv:2001.06846*，2020年。[51] Y. Wang、W. Wang、S. Joty 和 S. C. Hoi。CodeT5：面向代码理解与生成的标识符感知统一预训练编码器—解码器模型。*arXiv* 预印本 *arXiv:2109.00859*，2021年。[52] K. Wu、L. Wang、Z. Li、A. Zhang、X. Xiao、H. Wu、M. Zhang 和 H. Wang。通过分层 SQL 到问题生成进行数据增强，以实现跨领域文本到 SQL 解析。*arXiv* 预印本 *arXiv:2103.02227*，2021年。[53] K. Xuan、Y. Wang、Y. Wang、Z. Wen 和 Y. Dong。Sead：通过模式感知去噪实现端到端文本到 SQL 生成。*arXiv* 预印本 *arXiv:2105.07911*，2021年。[54] X. Yin、D. Gromann 和 S. Rudolph。将自然语言神经机器翻译为 SPARQL。未来一代计算机系统，117：510–519，2021年。