arXiv:2304.10893v1 [cs.CV] 21 Apr 2023

FindVehicleand VehicleFinder：用于基于自然语言的车辆检索的命名实体识别数据集，以及基于关键词的跨模态车辆检索系统

Runwei Guan[1,2,3,4†], Ka LokMan[2†], FeifanChen[2†], Shanliang Yao[1,2,3,4],Rongsheng Hu[6],XiaohuiZhu[2], Jeremy Smith[1], Eng Gee Lim[2] 和 Yutao Yue[3,4,5*1]利物浦大学电气工程与电子学系，英国利物浦，L69 3BX。[2]西交利物浦大学先进技术学院，中国苏州，215123。[3]西交利物浦大学—江苏省产业技术研究院技术学院，西交利物浦大学，中国苏州，215123。[4*]深度感知技术研究所，江苏省产业技术研究院，无锡，214000，中国。[5]利物浦大学数学科学系，英国利物浦，L69 7ZX。[6]昆明理工大学信息工程与自动化学院，中国昆明，650500。

*通讯作者。电子邮箱：yueyutao@idpt.org；贡献作者：Runwei.Guan@liverpool.ac.uk；Ka.Man@xjtlu.edu.cn；sgfchen5@liverpool.ac.uk；shanliang.yao@liverpool.ac.uk；1033170432@stu.jiangnan.edu.cn；Xiaohui.Zhu@xjtlu.edu.cn；J.S.Smith@liverpool.ac.uk；enggee.lim@xjtlu.edu.cn；

†这些作者对本研究的贡献相同。

摘要

基于自然语言（NL）的车辆检索任务旨在从所有候选车辆中，检索出与给定自然语言查询最一致的车辆。由于自然语言查询易于获取，此类任务在构建交互式智能交通系统（ITS）方面前景广阔。目前的解决方案主要致力于提取文本和图像特征，并将它们映射到同一潜在空间中，以比较相似度。然而，现有方法通常使用依存句法分析或语义角色标注技术来查找与车辆属性相关的关键词。这些技术可能需要大量预处理和后处理工作，而且当自然语言查询较为复杂时，还可能提取出错误的关键词。为解决这些问题并简化流程，我们借鉴命名实体识别（NER）的思路，构建了交通领域的命名实体识别数据集 FindVehicle。该数据集包含 42.3k 条经过标注的车辆轨迹自然语言描述，涵盖车辆的位置、朝向、类型和颜色等信息。FindVehicle 还采用了重叠实体和细粒度实体，以满足更进一步的需求。为验证其有效性，我们提出了一个基于自然语言的车辆检索基线模型 VehicleFinder。实验表明，使用基于 FindVehicle 预训练的文本编码器后，VehicleFinder 在我们基于 UA-DETRAC 构建的自制数据集上，通过文本指令检索目标车辆时，精确率达到 87.7%，召回率达到 89.4% [1]。从将指令输入 VehicleFinder 到判断目标车辆是否与指令一致，在一颗 ARM v8.2 CPU 上耗时 279.35 ms，在一块 RTX A4000 GPU 上耗时 93.72 ms，远快于基于 Transformer 的系统。该数据集已通过 https://github.com/GuanRunwei/FindVehicle 开源，代码实现可通过 https://github.com/GuanRunwei/VehicleFinder-CTIM 获取。

关键词： 跨模态学习、命名实体识别、智能交通系统、车辆检索、人机交互、目标检测

1 引言

车辆检索是一项任务，旨在根据查询图像从大型图像库中找到目标车辆。这是一种图像到图像的匹配技术，也称为车辆再识别 [2][3][4][5]。这项技术在构建面向智慧城市的智能交通系统 [6][7][8][9] [10]方面前景广阔。然而，基于图像的车辆检索系统在实际应用中也存在缺陷。例如，此类系统需要图像来提供目标车辆的特征，而在现实世界中，获取这样的图像并非易事。此外，由于只能通过一种模态提供时空信息，基于图像的车辆检索系统的性能也可能受到限制。

为缓解这些问题，近年来，自然语言（NL）作为现实世界中的另一种重要模态，受到了研究人员越来越多的关注。基于自然语言的车辆检索系统旨在根据自然语言描述识别目标车辆。此类系统具有更广泛的应用场景，例如，旁观者仅提供非正式描述时，仍可据此查找车辆。目前大多数基于自然语言的车辆检索实现都会构建文本编码器和视觉编码器，从两种数据中提取特征。随后，它们将提取到的文本和视觉嵌入投影到同一潜在空间，以比较两者的相似度。此外，这些方法还会对视觉数据和自然语言数据进行细致处理，以获得更有效的表征。例如，将车辆轨迹图像裁剪后生成全局运动图像 [11][12][13][14]。对于自然语言，一些与车辆属性相关的关键词（例如颜色、车辆类型和朝向）会从给定的自然语言查询中提取出来 [11][12][14][15]。尽管这些工作在 CityFlow-NL[16] 基准数据集上取得了可接受的性能，但仍有改进空间，尤其是在自然语言处理方面。我们发现，在实现关键词提取时，现有方法通常依赖依存句法分析（例如使用 NLTK 软件包）或语义角色标注技术，来判断一个词是否为关键词。这些技术只为句子中的词语标注词性。这意味着还需要预设规则和后处理，将提取出的关键词划分到相应的车辆属性中，使整个流程变得复杂 [15][17]。如果自然语言描述较为复杂，此类方法还可能提取出错误的关键词，进而导致后续模块中的错误传播，降低车型性能。

事实上，关键词提取已经是自然语言处理（NLP）中的一项成熟技术，也称为命名实体识别（NER）。阻碍我们应用最先进的命名实体识别模型来解决上述问题的主要障碍，是缺乏具有高质量标注的领域特定语料库。因此，为了缓解这一问题，我们提出了一个专注于交通领域、带有命名实体标注的自然语言数据集，称为 FindVehicle。它由从城市交通监控摄像头视角描述车辆的文本组成。数据集中的一些示例描述如下所示。我们还将其与从其他使用自然语言的交通领域数据集中选取的实例进行比较，即 Talk2Car [18] 和 CityFlow-NL [16]。所有细节见表 1。我们精心构建车辆描述，使其符合真实交通场景，并补充有关目标车辆的更详细信息。我们的数据集包含八类车辆特征，即车辆位置、朝向、品牌、车型、类型、颜色、与交通监控摄像头的距离以及速度。相比之下，Talk2Car[18] 仅记录车辆类型，而 CityFlow-NL [16] 仅包含四类信息，即车辆颜色、类型、动作和场景。描述文本中包含的车辆信息越多，数据就越能准确反映现实生活中的交通场景，同时降低基于自然语言的

自然语言的歧义给车辆检索任务带来了挑战。FindVehicle 和 CityFlow-NL [16]都包含对车辆与其他车辆（周边车辆）之间关系的描述。因此，我们不将周边车辆视为独立特征。此外，FindVehicle 标注了多粒度的命名实体标签，以满足未来的更多需求。

为验证所提出数据集的有效性，我们构建了一个简单且高效的跨模态车辆检索系统，称为 VehicleFinder。与参数量庞大、推理时间较长的现有基于Transformer的模型 [19][20], 不同，VehicleFinder 仅有 881 万个参数。这意味着它能够在实际场景中实现实时性能，并且更适用于边缘设备。VehicleFinder 使用我们基于 UA-DETRAC [1]训练集构建的自有图文数据集 Vehicle-TI 进行训练和测试。输入 VehicleFinder 的关键词由在 FindVehicle 上预训练的命名实体识别模型提取。实验结果表明，VehicleFinder 在车辆检测任务中的精确率为 87.7%，召回率为 89.4%。它在一颗 8 核 ARM v8.2 CPU 上的延迟为 279.35 毫秒。

总而言之，本文的主要贡献包括：

1. 我们提出了交通领域首个名为 FindVehicle 的命名实体识别数据集（基准数据集），其中包含 4.23 万个句子、136.1 万个词元、20.25 万个实体和 21 种实体类型。FindVehicle 不仅同时包含扁平实体和重叠实体，还涵盖粗粒度和细粒度实体类型。2. 我们提出了名为 VehicleFinder 的文本-图像跨模态车辆检索系统，以验证所提出的命名实体识别数据集的有效性。VehicleFinder 是一种效率极高、性能优良的车型，能够实现实时运行，并可部署于边缘设备。3. 在实验过程中，我们构建了名为 Vehicle-TI 的文本图像车辆匹配数据集。Vehicle-TI 包含 335,040 个训练样本、179,520 个测试样本和 83,776 个验证样本。

本文其余部分组织如下：第 2 节介绍本文的相关工作；第3 节介绍 FindVehicle 的关键信息及其构建方法；第 4节介绍 FindVehicle 的统计详情；第 5 节介绍我们的文本-图像跨模态车辆检索系统 VehicleFinder；第 6 节介绍 FindVehicle 的基线；第 7节介绍 VehicleFinder 的实验详情；第 8 节介绍本文的结论和未来工作；第9 节介绍 FindVehicle 面临的一些挑战。

2 相关工作

2.1 命名实体识别

命名实体识别（NER）是自然语言处理（NLP）中的一项经典序列标注任务，旨在定位并分类文本中具有特定类型的词语或句子。

表 1 车辆检索数据集

数据集	数据样本	信息性[1]	数量	含有命名实体识别
Talk2Car[2]	我的朋友正要下汽车。停一下，也让我下车! 是的，那应该是我儿子站在公交车旁边的楼梯上。 我的妈妈在右边！Park在她附近，她可能想搭便车。	Low	11959	No
CityFlow-NL[3]	一辆货运卡车驶过一条有许多小型汽车的十字路口。 那辆大型绿色平板半挂卡车正在直行。 一辆绿色卡车驶过十字路口，后面跟着一辆轿车。	中等	9374	No
FindVehicle	一辆 [灰色] [轿车] 以 [58公里每小时]的速度向 [右] 行驶。 也许是一辆 [[福特] [蒙迪欧]][4]。 一辆 [沃尔沃] [卡车] 停在路边，后面是一辆 [[法拉利] [458]]。 这辆 [蓝色] [[宝马] [320]]正 [驶离] ，距离我们 [150 米] 远。	High	42341	Yes

[1] 根据文本描述中关键车辆特征的集中程度计算得出。
[2] [18]
[3] [16]
[4] 指分配给某种实体类型的文本片段。例如，在短语 [[福特] 中 [蒙迪欧]]，[福特] 是一个 *vehicle brand* 实体，而 [蒙迪欧] 是一个 *vehicle model* 实体。 [[福特] [蒙迪欧]] 是一个 *vehicle type-sedan*实体。

命名实体识别模型的输入是带有词性标注的序列，如公式 1,

$$WT = (w_1, t_1), (w_2, t_2) \dots (w_i, t_i) \dots (w_n, t_n) \tag{1}$$

其中， n 表示由分词程序分出的词语数量。 t_i 是词语 w_i 的词性。

<escape>基于分词和词性标注的命名实体识别过程分为拆分、合并（确定实体边界）以及对部分词语重新分类（确定实体类别）。其输出是一个最优序列 WC^*， TC^*其格式为一对 *(*词类别（*WC*），标注类别（*TC*）*)*，如公式2所示。</escape>

$$WC^*, TC^* = (wc_1, tc_1), (wc_2, tc_2), (wc_i, tc_i), \dots, (wc_m, tc_m) \tag{2}$$

其中 $m \leq n$， $wc_i = [w_j, \dots, w_{j+k}]$， $tc_i = [t_j, \dots, t_{j+k}]$， $1 \leq k$， $j + k \leq n$。简而言之，命名实体识别模态可写为式3 所示，

$$(WC^*, TC^*) = argmax_{(WC,TC)} P(WC, TC|W, T) \tag{3}$$

其中， W 是词序列，而 T 是标注序列。 $P(\cdot)$ 是一个条件概率模型。

隐马尔可夫模型 [21] 和条件随机场 [22] 是两种典型的命名实体识别机器学习模型。卷积神经网络 [23],循环神经网络 [24], Transformer [25], 和图神经网络 [26], 这些深度学习模型在命名实体识别任务中均取得了最先进的结果。

此外，过去几年中提出了许多命名实体识别数据集。这些 [27][28][29][30][31] 是广为人知的命名实体识别数据集（基准数据集）。在这些数据集中，命名实体主要分为三类：平坦实体、重叠实体和非连续实体。 [32] 提出了一种统一神经网络框架，可同时解决这三个命名实体识别问题。

2.2 文本-图像车辆检索

基于测试集图像的跨模态学习进行车辆检索近年来备受关注 [11][33][13][34][35][14][36][15][37]。车型可以根据文本描述找出匹配度最高的车辆。根据架构，主要有两种形式。第一种是基于早期检索的端到端神经网络，在早期阶段融合图像和文本特征。第二种是基于后期检索的非端到端系统，分别提取图像和文本特征，并将其输入决策模块。

2.3 对比语言图像预训练

对比语言图像预训练（CLIP）将语言和图像两种模态结合在一个神经网络中，主要用于基于自然语言和计算机视觉的多模态任务。在此之前，大多数计算机视觉研究都是基于预定义标签进行训练的，而这种监督方式限制了神经网络的泛化能力和实用性。在自然语言处理（NLP）领域，已有大量研究利用海量语料库数据进行自监督学习，且这些车型的效果已经超过了人工标注的数据集 [38][39]。在计算机视觉（CV）领域，目前的主流方法仍然是使用带有标注信息的大规模数据集进行预训练 [40]。原始 CLIP [19] 创造性地使用文本作为监督信号来训练视觉车型，并在 ImageNet [40]上取得了显著成果。此外，原始 CLIP [19] 在零样本任务中也表现出色。 [20] 提出了一种名为 DenseCLIP 的 CLIP 框架，该框架擅长处理密集预测任务，例如语义分割和密集目标检测。 [41] 提出了一种新的对比损失函数，用于对语义空间中图像和文本特征的位置及几何信息进行归一化。

3 FindVehicle 的构建

3.1 简介

FindVehicle 是交通领域的首个命名实体识别数据集。它基于 UA-DETRAC [1] 的图像样本构建。FindVehicle 包含从交通监控摄像机视角对道路交通参与者（主要是车辆）的各种描述。一条描述包含一辆或多辆车辆的多个属性。这些属性均可由交通传感器检测到，例如监控摄像机、激光雷达和雷达。此外，FindVehicle 还融入了大量现实世界的先验知识，例如车辆品牌和车型。更进一步，FindVehicle 同时包含粗粒度实体和细粒度实体。实体既包括扁平实体，也包括重叠实体。

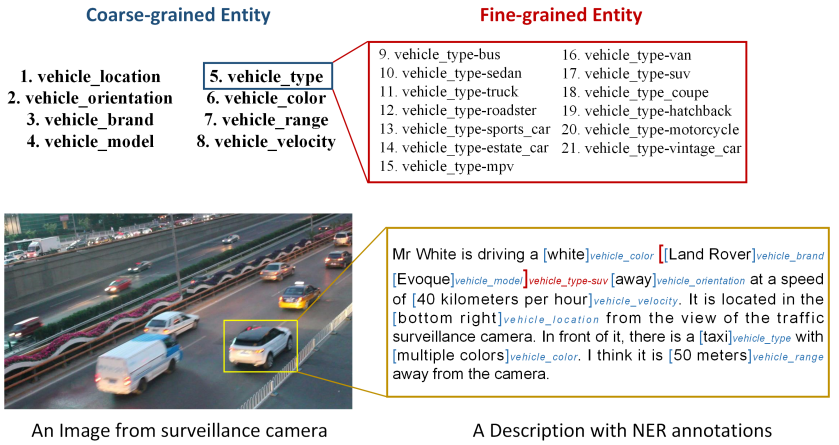


图 1 FindVehicle 的实体类型及标注示例。图像来源于 UA-DETRAC [1]。

3.2 实体类型

如图 1 所示，FindVehicle 中有 21 种实体类型，包括 8 种粗粒度实体和 13 种细粒度实体。这些实体都是车辆属性，并且均遵循现实世界中的分布。此外，FindVehicle 还同时包含扁平实体和重叠实体。

3.2.1 粗粒度实体

共有 8 种粗粒度实体，包括车辆位置、车辆朝向、车辆品牌、车辆型号、车辆类型、车辆颜色、车辆范围和车辆速度。

车辆位置 表示交通监控摄像头视角下车辆所在的位置，例如右下方、左上方等。

车辆朝向 表示交通监控摄像头视角下车辆车头的朝向，例如朝向镜头、背向镜头等。

车辆品牌 表示车辆的品牌。FindVehicle 收录了全球 65 个车辆品牌。

车辆型号 表示各车辆品牌的车型。FindVehicle 收录了不同车辆品牌的 4793 种车型。例如，Q7 是奥迪的一款车型。

车辆类型 表示车辆的类型，例如轿车、SUV 等。

车辆颜色 表示车辆的颜色，例如银灰色、玫瑰红等。

车辆范围表示车辆与交通监控摄像头之间的距离，例如 18 米、123 米等。

车辆 速度 表示车辆在道路上行驶的速度，例如每小时 50 公里、120 km/h 等。

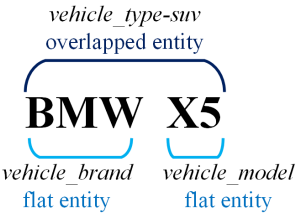


图 2 扁平实体与重叠实体示例。

3.2.2 细粒度实体

如图所示，1，在 FindVehicle 中，有 13 种细粒度实体，它们属于粗粒度实体 车辆类型，例如，宝马X5是 车辆类型-*SUV*中的一种细粒度实体。细粒度实体包含了有关汽车的人类先验知识。

3.2.3 扁平实体与重叠实体

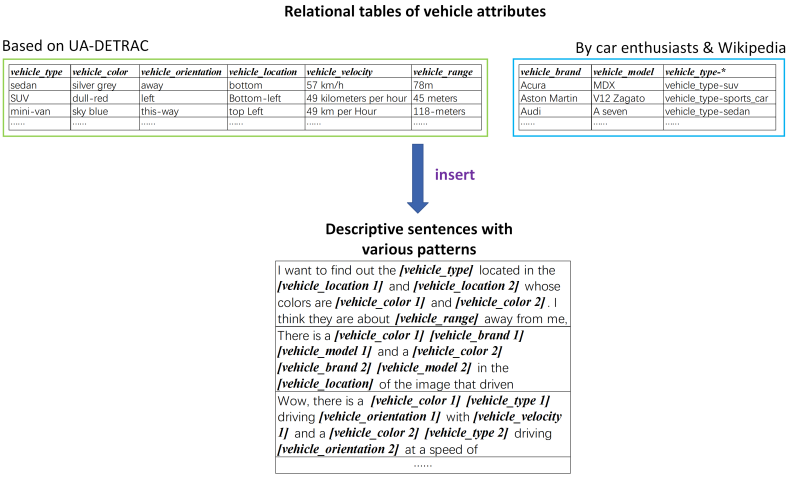
重叠实体存在于粗粒度实体中，包括 车辆品牌、车辆型号，也存在于细粒度实体中，包括 车辆类型-*。例如，如图 2 所示，宝马 的标签是 车辆品牌，而 X5 的标签是 车辆型号。对于汽车爱好者来说，宝马X5 的标签是 车辆类型-*suv*。

3.3 语料库收集

如图 4所示，语料库收集包括两部分：包含简单上下文的语料库和包含复杂上下文的语料库。

包含简单上下文的语料库指的是短句，这些短句列于表 1的“数据样本”列中。如图 3 所示，首先，我们从 UA- DETRAC [1]中抽取一些带有边界框和标签的目标车辆。基于这些样本，我们创建了一个关系表，用于保存相应车辆的属性。表中的每一项代表一辆具有多个属性的车辆。此外，为了增加数据集的复杂性，我们将一些正式的短语型和词型实体替换为符合非正式表达习惯的说法，并添加一些 UA-DETRAC [1]中不存在的罕见实体。此外，对于以下三个实体的生成： 车辆品牌、车辆型号 和 车辆类型-*，我们邀请了三位汽车爱好者，凭借其丰富的汽车知识并参考维基百科的搜索结果，收集并整理相关数据。他们以不同的表达方式撰写数据，最终整理出 65 个车辆品牌、4793 个车辆型号和 13 种车辆类型。其次，我们招募了四位志愿者，请他们按照各自的语气和表达习惯，写出多种句式的描述性句子。所有志愿者都受过良好教育，具备足够的英语语言知识。第三，我们通过句子自动生成框架，将目标车辆及其属性插入这些句式中。

如图所示1 ，具有复杂上下文的语料库包含带有人物主观情感和想象力的叙事长句或段落



vehicle_type	vehicle_color	vehicle_orientation	vehicle_location	vehicle_velocity	vehicle_range
sedan	silver grey	away	bottom	57 km/h	78m
SUV	dull-red	left	Bottom-left	49 kilometers per hour	45 meters
mini-van	sky blue	this-way	top Left	49 km per Hour	118-meters
.....	.....	.....	.....	.....	.....

vehicle_brand	vehicle_model	vehicle_type-*
Acura	MDX	vehicle_type-suv
Aston Martin	V12 Zagato	vehicle_type-sports_car
Audi	A seven	vehicle_type-sedan
.....	.....	.....

图 3 简单上下文语料库的生成。

。不同于结合人工与计算机生成简单上下文语料库的做法，复杂上下文语料库完全由人类编写。我们团队的四名成员通过观察 UA-DETRAC 中的图像，按照各自的写作习惯并发挥想象力，写下相应的句子和段落 [1]。

3.4 命名实体识别标注

如图 4 所示，在我们的命名实体识别标注框架中，分别针对具有简单上下文和复杂上下文的语料库，包含两个处理流程。

具有简单上下文的语料库的标注，是由我们的标注自动生成框架与句子自动生成同步完成的。之后，自动生成校正框架会自动识别自动生成框架生成的命名实体识别标注是否存在错误。如果数据存在错误，标注流程将被中断，并报告错误位置，然后我们会检查并修正错误。如果不存在错误，带有标注的语料库将直接加载到数据集中。

具有复杂上下文的语料库标注完全由人工完成。标注依据标注人员的常识和知识。标注人员均为熟悉车辆且擅长叙事写作的志愿者。

如图 5 所示，我们以两种格式组织数据：JSON 格式和 CoNLL 格式 [27]。键 *ner label* 的值为标注的命名实体。每个元素中 *ner label* 的值为 [实体类型、字符跨度的起始索引、字符跨度的结束索引、词元跨度的起始索引、词元跨度的结束索引]。我们的标注同时考虑字符级和词元级，以满足命名实体识别模型的不同需求。键 *re label* 表示 *ner label* 各值的索引，这些值在句子上下文中指向同一目标。

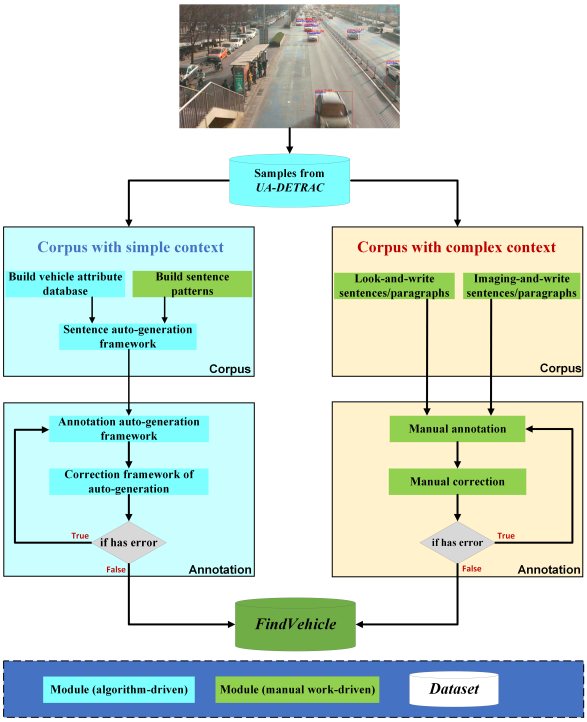


图 4 FindVehicle 的语料库收集与标注框架。

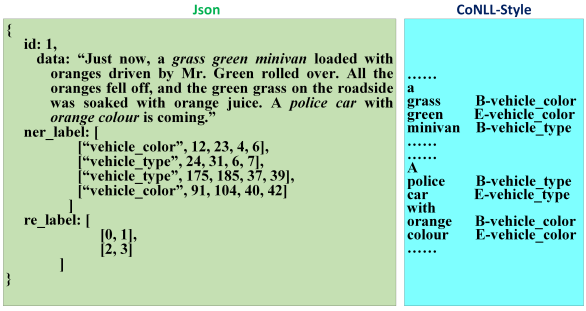


图 5 FindVehicle 的两种标注格式。

4 FindVehicle 的数据统计

4.1 FindVehicle 的规模与分布

FindVehicle 是交通领域首个同时包含自动标注和人工标注的命名实体识别数据集。如表 2 所示，我们展示了 FindVehicle 与其他广泛使用的知名命名实体识别数据集的统计信息，包括 CoNLL'03 [27],WikiGold [28], WNUT'17 [29],I2B2 [42]和 OntoNotes [30]。FindVehicle 包含 4.23 万个句子，共 136.1 万

表 2 FindVehicle 和其他知名命名实体识别数据集的统计信息

数据集	句子	词元	实体	实体类别	领域
WikiGold [28]	1.7k	39k	3.6k	4	通用
WNUT'17 [29]	4.7k	86.1k	3.1k	6	社交媒体
CoNLL'03 [27]	22.1k	301.4k	35.1k	4	新闻稿
I2B2 [42]	107.9k	805.1k	28.9k	23	医学
OntoNotes [30]	103.8k	2067k	161.8k	18	通用
FindVehicle（本文方法）	**42.3k**	**1361.1k**	**202.5k**		**交通**

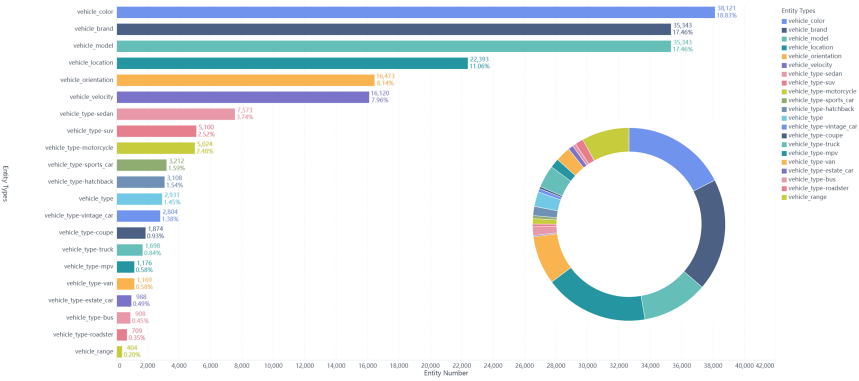


图6 FindVehicle 中实体的统计数据。

表3 FindVehicle的数据划分

数据集	粗粒度实体	细粒度实体	扁平实体	重叠实体
训练集	84.6千	1.82万	84.2k	18.6k
测试集	82.5k	1.74万	8.27万	1.72万

词元、20.25万实体和21个实体类别。如图6所示，实体类型呈长尾分布，以反映真实世界的交通场景。

4.2 数据集划分

FindVehicle 是一个混合命名实体识别数据集，包含扁平实体和重叠实体。我们将其划分为训练集和测试集。这两个数据集的详细信息如表 3 所示。对于训练集，其中包含 84.6k 个粗粒度实体和 18.2k 个细粒度实体。此外，还包含 84.2k 个扁平实体和 18.6k 个重叠实体。对于测试集，其中包含 82.5k 个粗粒度实体和 17.4k 个细粒度实体。此外，测试集中还包含 82.7k 个扁平实体和 17.2k 个重叠实体。

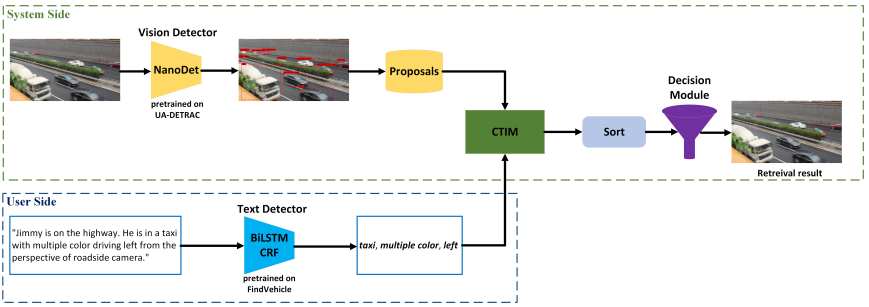


图 7 VehicleFinder 的架构。

5 VehicleFinder

VehicleFinder是一个轻量级文本-图像跨模态车辆检索系统。用户可以通过描述车辆的类型、颜色和朝向来找到目标车辆。如图 7 所示，VehicleFinder包含两个分支。其中一个分支通过视觉检测器提取候选项，另一个分支则通过文本检测器提取命名实体。我们采用NanoDet [43] 作为视觉检测器，并采用 BiLSTM-CRF [24]作为文本检测器。NanoDet [43] 在 UA- DETRAC 上进行了预训练， [1] 而 BiLSTM-CRF [24] 则在我们的 FindVehicle 上进行了预训练。候选项和命名实体将被加载到对比文本-图像模块（CTIM）中，以比较两种模态数据的语义相似度。

如图 8 所示，CTIM 中有两个编码器分支，分别用于对图像和文本模态的数据进行编码。CTIM 的输出是图像与文本之间的相似度，其取值范围为 0 到 1。输出低于 0.5 表示图像与文本无关，而输出高于 0.5 则表示二者相关。CTIM 是一个完整的卷积模块，其中的卷积操作均为深度可分离卷积 [44]，可显著减少参数数量，尤其是在神经网络的深层中。CTIM 可以作为即插即用模块应用于某些跨模态系统。

图像编码器分支中包含五个相同的编码器单元。编码器单元首先将输入特征图 $x_i \in R^{c\times h\times w}$ 分为三个分支，其中 c、 h、 w 分别表示特征图的通道数、高度和宽度。前三个分支使用不同大小的卷积核提取具有不同感受野的特征。输出特征图$\hat{x}_i \in R^{c\times h\times w}$ 经过 ReLU [45], 激活后，通过核大小为 3×3的深度可分离卷积操作增加通道数并减小空间尺寸。经过批归一化和 ReLU 激活后，输出特征图为 $x_{i+1} \in R^{2c\times \frac{h}{2}\times \frac{w}{2}}$。为缓解训练集阶段的梯度消失和梯度爆炸问题，将一条带有深度可分离卷积的长残差路径与输出特征图连接起来。编码器单元的最终输出为$\hat{x}_{i+1} \in R^{2c\times \frac{h}{2}\times \frac{w}{2}}$。整个过程如公式4所示。

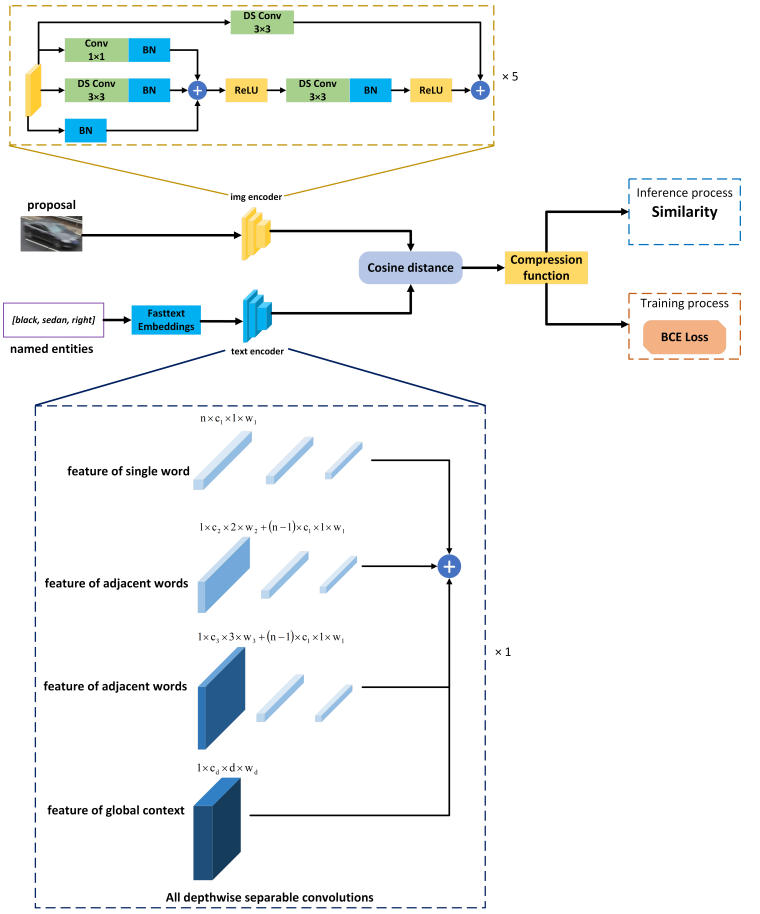


图 8 对比文本-图像模块（CTIM）的架构。除卷积核大小为 1×1 的卷积操作外，所有卷积操作均为深度可分离卷积。因为深度可分离卷积包含卷积核大小为 1×1 的卷积操作。c_i 表示特征图的通道数，其卷积核高度为 i。w_i 表示特征图的宽度，其卷积核高度为 i。

$$
\begin{aligned}
&\hat{x}_i = BN(Conv_{3\times3}(x_i)) + BN(Conv_{1\times1}(x_i)) + BN(x_i), \hat{x}_i \in R^{c\times h\times w} \\
&x_{i+1} = BN(Conv_{3\times3}(ReLU(\hat{x}_i))), x_{i+1} \in R^{2c\times \frac{h}{2}\times \frac{w}{2}} \\
&\hat{x}_{i+1} = ReLU(x_{i+1}) + Conv_{3\times3}(x_i), \hat{x}_{i+1} \in R^{2c\times \frac{h}{2}\times \frac{w}{2}}
\end{aligned} \tag{4}
$$

在文本编码器分支中，首先使用 Fasttext（wiki-news-300d-1M）的预训练嵌入对命名实体进行嵌入 [46]。与 Word2vec [47] 和 GloVe [48] 相比，Fasttext 能够基于现有词语推断词典中未收录词语的嵌入，因此对该系统而言更加鲁棒。嵌入矩阵的形状为 $d \times 300$，其中 d 表示

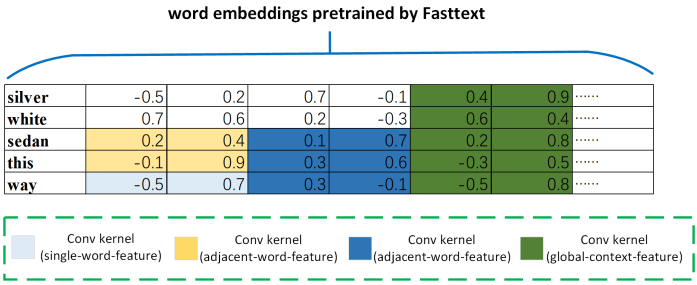


图 9 我们的文本编码器中的四种多尺度卷积操作。

命名实体的数量，300 是每个命名实体的向量长度。之后，我们采用四组多尺度深度可分离卷积操作，同时提取不同尺度的特征。第一组包含 n 个卷积操作，卷积核大小为 $1 \times w_1$，用于提取命名实体中单个词的特征。第二组包含一个卷积操作，卷积核大小为 $2 \times w_2$，以及 $n-1$ 个卷积操作，卷积核大小为 $1 \times w_1$，其中 $2 \times w_2$ 卷积核用于提取相邻词的关联特征。其余 $1 \times w_1$ 个卷积操作用于增强非线性表示。第三组首先通过一个 $3 \times w_3$ 卷积核进行卷积处理，同样用于提取相邻词的特征，词窗口大小为三。之后的操作与第二组相同。第四组包含一个卷积操作，卷积核大小为 $d \times w_d$，用于提取全局上下文的特征。最后，将这四组的输出相加，得到命名实体的综合表示。这四种卷积操作如图9所示。

获得候选项和命名实体的表示后，我们将它们的形状对齐，以计算它们的余弦距离。余弦距离用于衡量候选项向量与命名实体向量之间的距离。无论在高维还是低维情况下，它都能保持相同的相似性，因此是反映方向相对差异的稳健指标。公式5 展示了余弦距离。

$$CosineD = \frac{\mathbf{A} \cdot \mathbf{B}}{\|\mathbf{A}\|\|\mathbf{B}\|} = \frac{\sum_{i=1}^{n} A_i B_i}{\sqrt{\sum_{i=1}^{n} A_i^2} \sqrt{\sum_{i=1}^{n} B_i^2}}, CosineD \in [-1, 1] \tag{5}$$

其中，n 是向量的分量数。A_i 和 B_i 分别表示 i个分量对应的文本向量和图像向量。

然而，余弦距离的取值范围是 $[-1, 1]$。这意味着，余弦距离的结果不能直接输入二元交叉熵损失（二元交叉熵损失），因为二元交叉熵损失（公式6）无法处理负数。

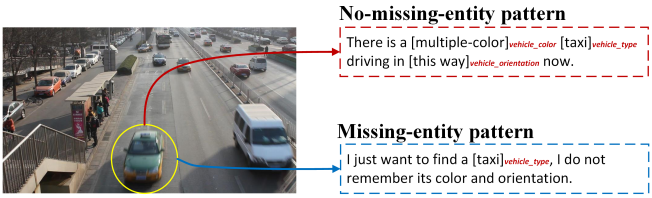


图 10 两种命令模式。

$$L_{BCE} = -\sum_{i=1}^{N}[y_i ln(\hat{y}_i) + (1 - y_i) ln(1 - \hat{y}_i)] \tag{6}$$

其中， N 表示一个批次中的样本数。 $y_i \in \{0, 1\}$ 是真实标签，而 $\hat{}$ $y_i \in [-1, 1]$ 是神经网络预测的余弦距离结果。显然，如果 $\hat{}y_i$小于零，$\hat{}y_i$ 就不在 $ln(\cdot)$ 的定义域内。

因此，如公式 7 所示，我们使用公式7 将余弦距离从 $[-1, 1]$ 压缩到 $[0, 1]$, ，其结果可以作为二元交叉熵损失的输入。线性压缩函数是一个单调递增函数，其值域为 $[0, 1]$。它处处可微。单调性确保变量从 $[-1, 1]$ 映射到 $[0, 1]$时，其相对位置不会发生变化。处处可微的性质确保它能够很好地参与神经网络中的反向传播。

$$Comp(x) = \frac{1}{2}x + \frac{1}{2}, Comp(x) \in [0, 1] \tag{7}$$

其中， $x \in [-1, 1]$ 表示通过余弦距离计算得到的结果。

因此，损失函数的完整形式如公式8所示。

$$L(y_i, \hat{y}_i) = -\sum_{i=1}^{N}[y_i ln(Comp(\hat{y}_i)) + (1 - y_i) ln(1 - Comp(\hat{y}_i))] \tag{8}$$

最后，我们的 VehicleFinder 将计算从命令中提取的命名实体与视觉检测器提取的候选目标之间的相似度。我们会按照候选目标与命名实体之间的相似度降序排列。排序列表 $proposals$ 随后将输入决策模块。在决策模块中，考虑到用户不一定总能详细描述车辆特征，我们会考虑两种命令模式并分别处理，以增强系统的鲁棒性，同时也让系统更加用户友好。如图10 所示，第一种是无实体缺失模式，第二种是实体缺失模式。无实体缺失模式表示命令中包含全部三种命名实体： 车辆类型、车辆颜色和车辆朝向。实体缺失模式表示命令中包含一种或两种命名实体，而另外一种或两种命名实体未被提及。

如算法 1 所示，我们首先设置两个阈值 th_{nm} 和 th_m，分别表示无实体缺失模式的阈值和实体缺失模式的阈值。变量 $proposals$ 包含车辆：相似度对，表示候选目标及其与从指令中提取的命名实体之间的相似度。对于无实体缺失模式，如果存在 sim 大于 th_{nm}的车辆：相似度对，则将该车辆：相似度对添加到 $retainVehicle$中。如果不存在 sim 大于 th_{nm}的车辆：相似度对，决策模块将继续搜索 sim 大于 th_m的车辆：相似度对。如果存在 sim 大于 th_m的车辆：相似度对，也会将该车辆：相似度对添加到 $retainVehicle$中。 th_{nm} 和 th_m 根据实验结果设定。默认情况下，我们假设 th_{nm} 大于 th_m。

算法 1 决策模块

输入： th_{nm}, th_m, $proposals = [\{vehicle:\ sim\}]$, $retainVehicle = []$

如果 $proposals$.exists($vehicle.sim \geq th_{nm}$) **则**
  对于 p 在 $proposals$ **执行**
    如果 $p.sim \geq th_{nm}$ **执行**
      $retainVehicle$.append(p)
    结束如果

否则如果 $proposals$.exists($th_m \leq vehicle.sim < th_{nm}$) **则**
  对于 p 在 $proposals$ **执行**
    如果 $p.sim \geq th_m$ **执行**
      $retainVehicle$.append(p)
    结束如果
  结束如果

结束如果

返回 $retainVehicle$

6 FindVehicle 的实验

在 FindVehicle 的实验中，我们为 FindVehicle 构建了基线模型。

6.1 训练集与测试集的设置

我们选取了三个具有代表性且处于最先进水平的模型，在 FindVehicle 上进行训练和测试，分别为 BiLSTM-CRF [24], BERT-CRF [49] 和 FLERT [50]。

BiLSTM-CRF [24] 结合了双向长短期记忆网络和条件随机场。双向长短期记忆网络充当编码器层，并以词嵌入作为输入，条件随机场则充当解码器

表 4 FindVehicle 训练集上各车型的实现细节

车型	训练轮次	BS	ILR	Opt	Sch[1]
BiLSTM-CRF [24]	80	32	0.001	AdamW	余弦
BERT-CRF [49]	100	4	0.0015	AdamW	余弦
FLERT [50]	80	8	0.001	AdamW	余弦

[1] BS 表示批量大小；ILR 表示初始学习率；Opt 表示优化器；Sch 表示调度器。

表 5 混淆矩阵

预测 / 标签	正例	负例
正例	真正例（TP）	假负例（FN）
负例	假阳性（FP）	真阴性（TN）

根据编码器输出的隐藏状态，为每个词元确定标签。

BERT-CRF [49] 将双向长短期记忆网络的词嵌入替换为从 BERT 学习的子词嵌入，并将编码器从双向长短期记忆网络更换为 Transformer。

FLERT [50] 是一种将文档级特征作为额外信息纳入的命名实体识别模型。通过在查询句子的两侧（左侧和右侧）添加上下文文本，FLERT 能够捕获文档级特征，并且比之前的模型呈现出更好的预测结果。

对于每个模型，我们使用最合适的超参数，使模型能够平稳收敛。我们在一块 TITAN RTX GPU 上训练和测试这些模型。表4 展示了实现细节。

此外，如公式 9、10 和 11 所示，我们选择精确率、召回率和 F1 分数作为测试集的评估指标，这些指标基于混淆矩阵（表 5）。

$$Precision = \frac{TP}{TP + FP} \tag{9}$$

$$Recall = \frac{TP}{TP + FN} \tag{10}$$

$$F1 = \frac{2 \times Precision \times Recall}{Precision + Recall} \tag{11}$$

6.2 FindVehicle 的基线方法

表 6展示了各车型在 FindVehicle 测试集上的评估结果。显然，基于 Transformer 的模型表现优于 RNN 模型。BiLSTM-CRF [24] 的 F1 分数为 49.5%，是所有车型中最低的。FLERT [50] 的 F1 分数为 80.9%，是最高值，比 BERT-CRF 高 3% [49]。

表 6 三种模型在 FindVehicle 测试集上的评估结果

车型	精确率（%）	召回率 (%)	F1（%）
BiLSTM-CRF [24]	50.1	50.4	49.5
BERT-CRF [49]	77.7	78.4	77.9
FLERT [50]	80.6	81.3	80.9

表 7 FLERT [50] 在 FindVehicle 所有类别上的评估结果

实体类别	精确率（%）	召回率 (%)	F1（%）
车辆颜色	91.8	91.9	91.8
车辆品牌	91.9	91.8	91.8
车辆型号	91.7	91.7	91.7
车辆位置	91.7	91.8	91.8
车辆速度	90.4	89.8	90.1
车辆朝向	91.5	88.7	90.1
车辆范围	92.1	91.9	92.0
车辆类型	91.6	91.8	91.7
车辆 轿车	81.9	83.0	82.4
车辆类型-SUV	84.7	86.1	85.4
车辆类型-摩托车	87.7	91.5	89.6
车辆类型-跑车	84.1	85.5	84.8
车辆类型-掀背车	68.8	66.5	67.6
车辆类型-老爷车	81.6	82.9	82.2
车辆类型-轿跑车	72.7	77.4	75.0
车辆类型-卡车	74.7	81.0	77.7
车辆类型-厢式车	62.9	74.0	68.1
车辆类型-多用途汽车	63.1	66.8	64.9
车辆类型-旅行车	62.0	58.7	60.3
车辆类型-公交车	70.7	68.0	69.3
车辆类型-敞篷跑车	44.9	32.4	37.8

表 8 FindVehicle 中 FLERT [50]对扁平实体和重叠实体的评估结果

实体类型	精确率（%）	召回率 (%)	F1（%）
平坦实体	91.6	91.5	91.6
重叠实体	71.5	72.6	71.9

此外，我们还对命名实体的全部 21 个类别的评估结果进行了统计。我们以 FLERT 的评估结果 [50] 为例，如表7 所示，细粒度实体的所有评估指标值都远低于粗粒度实体。这表明，对于神经网络而言，细粒度实体比粗粒度实体更难识别。此外，我们还计算了 FLERT 对扁平实体和重叠实体的评估结果 [50]。如表8所示，扁平实体的三项指标值比重叠实体高出约 20%。在 FindVehicle 中，重叠实体的识别仍是一项挑战。

表9 不同命名实体识别数据集测试集上的车型表现

F1 数据集 / 车型	CoLL'03 [27]	WNUT'17 [29]	Ontonotes [30]	**FindVehicle（本文方法）**
BiLSTM-CRF [24]	91.7	42.6	87.1	**49.5**
BERT-CRF [49]	93.4	59.8	92.0	**77.9**
FLERT [50]	94.1	61.1	92.3	**80.9**

Triple keywords	Proposals	Labels
(bus, multi-color, this way)		1
(sedan, black, away)		1
(suv, yellow, right)		0

图11 Vehicle-TI 的数据格式。

6.3 不同命名实体识别数据集上的车型比较

我们还比较了不同命名实体识别数据集上各车型的表现（表9），包括 CoNLL'03（4 个类别） [27],WNUT'17（6 个类别） [29],Ontonotes（18 个类别） [30] 以及我们的 FindVehicle（21 个类别）。我们使用 F1 分数作为评估指标。可以看到，三个车型在 FindVehicle 上的 F1 分数（公式11）均低于在 CoNLL'03 [27] 和 Ontonotes（18 个类别） [30],上的分数，这表明我们的数据集中存在一定挑战。

7 VehicleFinder 实验

VehicleFinder 实验分为四个部分：CTIM、视觉检测器、文本检测器和 VehicleFinder。

7.1 CTIM 实验

7.1.1 训练与评估设置

我们基于 UA-DETRAC 的训练集构建了一个名为 Vehicle-TI 的文本到图像数据集 [1] 用于训练和测试我们的 CTIM。如图 11 所示，Vehicle-TI 中的每个数据样本都包含一个三元关键词（文本模态）和一个候选项（图像

表 10 CTIM 在 Vehicle-TI 训练集上的实现细节

车型	训练轮次	BS	ILR	Opt	Sch[1]
CTIM	50	64	0.001	AdamW	Step

[1] BS 表示批量大小；ILR 表示初始学习率；Opt 表示优化器；Sch 表示调度器。

表11 CTIM 在 Vehicle-TI 测试集上的评估

车型	参数（M）	延迟（ms） 8 核 ARM v8.2	延迟（ms） i7-12700	延迟(ms) RTX A4000	准确率(%)
CTIM	3.84	131.42	67.43	39.47	97.7

模态）和标签，这些内容从 UA-DETRAC 中提取并重构[1]。三元关键词包含车辆的类型、颜色和朝向。标签表示候选项是否与三元关键词的描述一致，其中 1 表示一致（正样本），0 表示不一致（负样本）。正样本用于拉近文本与图像的特征编码，而负样本用于拉远文本与图像的特征编码。Vehicle-TI 中共有 598,336 个样本，其中 335,040 个用于训练集，179,520 个用于测试集，83,776 个用于验证集。

如表 10 所示，我们使用批量大小为 64 的批次训练轮次为 50 的 CTIM。初始学习率为 0.001，并使用 AdamW 优化 CTIM [51]。学习率通过 Step 调度器进行调度。

此外，我们将 0.7 设为车辆候选框与三元关键词一致性的阈值。若 CTIM 的输出高于 0.7，则表示车辆候选框与三元关键词一致（强相关）；否则，我们认为它们不相关或弱相关。

7.1.2 评估结果

表11 展示了 CTIM 在 Vehicle-TI 测试集上的评估结果。CTIM 仅有 384 万个参数，在识别车辆图像与三元关键词之间的一致性方面达到了 97.7% 的准确率（公式12）。

$$Accuracy = \frac{TP + TN}{TP + FP + TN + FN} \tag{12}$$

此外，我们还测试了 CTIM 在不同设备上的推理速度。在 NVIDIA Jetson AGX Xavier 的 8 核 ARM v8.2 上，CTIM 识别一个样本耗时 131.42 ms。在 i7-12700 上进行测试时，CTIM 的延迟为 67.43 ms。此外，在一块 RTX A4000 上，CTIM 耗时 39.47 ms。上述结果证明，CTIM 在边缘设备和主机设备上都能保持较高的性能。

表 12 Vehicle-TI 测试集上的消融与对比实验

车型	参数（M）	延迟（ms） 8 核 ARM v8.2 i7-12700	延迟（ms）	延迟（ms） RTX A4000	准确率（%）
CTIM	3.84	131.42	67.43	39.47	97.7
CTIM-Conv-CosineD-Linear	36.67	185.64	79.72	55.54	97.2
CTIM-DSConv-Siamese	99.72	211.79	134.45	96.18	25.7
CLIP（ResNet-50+Transformer）	102.58	2236.12	1395.77	527.86	96.5
Bert-Siamese	189.53	2819.87	1657.53	886.62	97.3

7.1.3 CTIM 与其他车型的比较

如上所述，CTIM 中的所有卷积操作均为深度可分离卷积。此外，我们使用余弦距离和线性压缩函数来衡量和处理文本与图像模态之间的相似性。我们仍将其称为 CTIM。

首先，我们将所有深度可分离卷积操作替换为普通卷积操作，得到 CTIM-Conv-CosineD。

其次，我们将 CTIM 中的余弦距离和线性压缩函数替换为全连接层。这是普通孪生神经网络中通过全连接层拟合相似度的操作。我们将其称为 CTIM- DSConv-Siamese。

第三，我们采用最知名的对比式语言与图像预训练车型 CLIP [19]。我们采用 ResNet-50 作为图像编码器，采用 Transformer 作为文本编码器。CLIP 的参数总数为 1.0258 亿。

最后，我们对基于 BERT 的孪生神经网络 [52] 进行微调，使其适用于我们的任务。其架构基于 Transformer，与前述神经网络截然不同。我们将其称为 Bert-Siamese。

如表 12 所示，CTIM 的所有指标均为最佳。相比之下，CTIM-Conv-CosineD-Linear 有 3660 万个参数，比 CTIM 多 3283 万个。此外，它在不同设备上的运行速度均慢于 CTIM。

其次，CTIM-DSConv-Siamese 的参数量在所有基于 CNN 的神经网络中最大，达到 99.72 百万。它在不同设备上的推理速度也最慢，准确率仅为 25.7%。

第三，采用 ResNet-50 和 Transformer 的 CLIP 含有 102.58 百万参数。在 8 核 ARM v8.2 上推理一个文本图像对样本约需 2.2 秒，在测试集上的准确率为 96.5%。

最后，尽管 Bert-Siamese [52] 的表现与我们的 CTIM 接近，但其参数量高达 189.53 百万。在 8 核 ARM v8.2 上识别一个样本近需 3 秒，速度过慢，无法部署在边缘设备上。

7.2 视觉检测器实验

视觉检测器用于从图像中提取车辆候选项。我们采用 NanoDet-m [43] 作为视觉检测器，它是一种轻量级检测器，具有

Tab第 13 UA-DETRAC 训练集上的 NanoDet-m 实现细节 [1]

车型	训练轮次	BS	ILR	Opt	Sch[1]
NanoDet-m [43]	200	16	0.001	AdamW	余弦

[1] BS 表示批量大小；ILR 表示初始学习率；Opt 表示优化器；Sch 表示调度器。

表 14 在 UA-DETRAC 上对 NanoDet-m 的评估 [1]

车型	参数量（M）	延迟（毫秒） 8 核 ARM v8.2	延迟（毫秒） i7-12700	延迟（毫秒） RTX A4000	召回率（%）
NanoDet-m [43]	0.95	7.99	3.72	1.14	86.7

表 15 BiLSTM-CRF 推理速度评估 [24]

车型	参数量（M）	延迟（ms） 8 核 ARM v8.2	延迟（ms） i7-12700	延迟（毫秒） RTX A4000
BiLSTM-CRF [24]	4.02	148.57	87.19	51.73

仅有 95 万个参数。该模型在 UA-DETRAC 训练集上进行训练[1]。实现细节见表 13。

此外，为了让视觉检测器尽可能少漏检目标，我们使用召回率而非精确率作为评估指标。如表14所示，NanoDet-m [43] 在测试集上的召回率为 86.7%。

7.3 文本检测器实验

文本检测器用于从用户指令中提取关键词（命名实体）。在第 6.2 节提及的所有命名实体识别模型中，BiLSTM-CRF 的参数较少、推理速度较快，因此我们在 Find- Vehicle 数据集上训练 BiLSTM-CRF，将其用作文本检测器，以提取以下类型的命名实体：车辆类型、车辆颜色 和车辆朝向。实现细节见表 4。

如表 15 所示，BiLSTM-CRF 有 402 万个参数。在 8 核 ARM v8.2 处理器上，它从 FindVehicle 的一个样本中提取全部命名实体需要 148.57 毫秒。此外，在 i7-12700 和 RTX A4000 上测试时，耗时分别为 87.19 毫秒和 51.73 毫秒。

7.4 VehicleFinder 的评估

7.4.1 评估设置

我们从 UA-DETRAC 测试集 [1] 中随机抽取 2000 张图像，作为 VehicleFinder 的自建测试集。对于每张图像，我们编写一段检索文本，对应图像中的一辆或多辆车辆。测试集的格式如图 12 所示。每个条目包含图像路径列 图像路径、目标 ID 目标 *ID*、边界框左上角的横坐标

表 16 BiLSTM-CRF 对不同类型命名实体的 F1 分数 [24]在 FindVehicle 上

车型	F1（$vehicle_type$）	F1（$vehicle_color$）	F1（$vehicle_orientation$）
BiLSTM-CRF [43]	90.56	89.79	90.17

img_path	*target_id*	*left*	*top*	*width*	*height*	*retrieval_text*
img1.jpg	1	25.813	290.99	116.197	106.17	The car is driven this way. It seems a Sedan.
img2.jpg	2	101.12	181.55	78.65	67.34	Hello, robot, please help me to find out a silver grey mini-van driving this way.
.....	.....	.....	.....	.....	.....	.....

图 12 VehicleFinder 自制测试集的格式。

左、边界框左上角的纵坐标 上、边界框的宽度 宽度、边界框的高度 高度 以及检索内容检索文本。根据这些检索文本，2000 张图像中共有 3917 辆目标车辆。我们采用精确率、召回率和 F1 分数来评估 VehicleFinder，相关公式见式 13、14 和15。我们还在三种不同设备上测试了 VehicleFinder。

$$Precision_V = \frac{num(detected\ vehicles\ \&\ detected\ vehicles\ in\ the\ testset)}{num(detected\ vehicles)} \tag{13}$$

$$Recall_V = \frac{num(detected\ vehicles\ \&\ detected\ vehicles\ in\ the\ testset)}{num(all\ vehicles\ in\ the\ testset)} \tag{14}$$

$$F1 = \frac{2 \times Precision_V \times Recall_V}{Precision_V + Recall_V} \tag{15}$$

7.4.2 评估结果

表 17 显示，我们的 VehicleFinder(CTIM) 有 881 万个参数，包含视觉检测器、文本检测器和 CTIM。将两个阈值 th_{nm} 和 th_m 分别设为 0.70 和 0.30 后，我们的 VehicleFinder(CTIM) 达到 87.7% 的精确率、89.4% 的召回率和 88.5% 的 F1 分数。图 13 展示了 VehicleFinder(CTIM) 的测试结果。可以看出，能够根据描述精准检索出目标车辆。

此外，我们还收集了对照组的结果。VehicleFinder(CTIM-Conv-CosineD-Linear) 的精确率为 87.4%，召回率为 87.9%，F1 分数为 87.6%，参数量为 4164 万。VehicleFinder(CTIM- DSConv-Siamese) 有 1.0469 亿个参数，其 F1 分数仅为 12.5%，在所有模型中最低。VehicleFinder(CLIP) 的参数总量为 1.0755 亿，F1 分数为 87.5%。VehicleFinder(Bert-Siamese) 的参数量几乎

表 17 VehicleFinder 在我们自制测试集上的评估

车型	参数（M）	精确率（%）	召回率(%)	F1(%)
VehicleFinder(CTIM)	8.81	87.7	89.4	88.5
VehicleFinder(CTIM-Conv-CosineD-Linear)	41.64	87.4	87.9	87.6
VehicleFinder(CTIM-DSConv-Siamese)	104.69	11.7	13.4	12.5
VehicleFinder(CLIP)	107.55	86.6	88.4	87.5
VehicleFinder(Bert-Siamese)	194.50	87.5	89.4	88.4

与我们的 VehicleFinder(CTIM) 相同，其 F1 分数为 88.4%，但参数量过于庞大。

我们从命令加载到 VehicleFinder(CTIM) 的那一刻开始计时，直到 VehicleFinder(CTIM) 完成对一辆车的识别为止，以此计算 VehicleFinder(CTIM) 的延迟。如公式16 所示，T_{ner} 表示命名实体识别所需的时间，而 T_{cti} 表示命名实体一致性与一个车辆候选项的识别时间，其中包括文本检测器和 CTIM 的推理时间。我们忽略系统调度不同模型所需的时间。表18展示了 VehicleFinder(CTIM) 的推理速度评估结果。延迟最长的是在一台 8 核 ARM v8.2 CPU 上的 279.35 毫秒，最短的是在一台 RTX A4000 GPU 上的 93.72 毫秒。这表明 VehicleFinder(CTIM) 既可部署在边缘设备上，也可部署在主机设备上，但主机设备是更好的选择。

相比之下，首先，VehicleFinder(CTIM-Conv-CosineD-Linear) 在一台 ARM v8.2 CPU 上比 VehicleFinder(CTIM) 慢 86.42 毫秒，在一台 RTX A4000 GPU 上慢 40.85 毫秒。其次，VehicleFinder(CTIM-DSConv-Siamese) 在一台 ARM v8.2 CPU 上比 VehicleFinder(CTIM) 慢 119.05 毫秒，在一台 RTX A4000 GPU 上慢 73.73 毫秒。第三，VehicleFinder(CLIP) 在 ARM v8.2 CPU 上推理一个文本图像对需要两秒以上。第四，VehicleFinder(Bert-Siamese) 的推理速度最慢，在一台 ARM v8.2 CPU 上的延迟超过 3 秒（3091.33 毫秒），在一台 RTX A4000 GPU 上也接近 1 秒。VehicleFinder(Bert-Siamese) 的推理速度实在太慢，无法部署在边缘设备或主机设备上用于实际交通场景。

最后但同样重要的是，我们还在一些交通场景的收集图像上测试了 VehicleFinder(CTIM)。基于这些图像，我们招募了两名志愿者来描述他们想要在图像中找到的车辆。如图14 所示，VehicleFinder(CTIM)仍能根据志愿者的描述准确找到目标车辆。此外，如图15 所示，我们还对边缘情况进行了推理，这意味着 VehicleFinder(CTIM)在遇到一些不利现象时仍能在一定程度上保持稳健。

$$Latency = T_{ner} + T_{cti} \tag{16}$$

8 结论与未来工作

我们提出了交通领域首个命名实体识别数据集 FindVehicle，其中包含不同的句子，用于描述不同交通场景中的车辆。命名实体包括若干可检测的车辆属性



图 13 VehicleFinder 在 UA-DETRAC 上的推理结果样本 [1]。



图 14 VehicleFinder 在我们收集的图像上的推理结果。

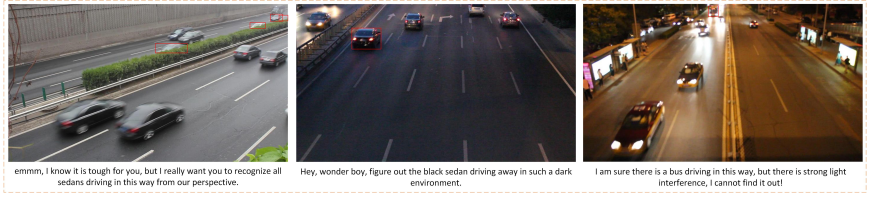


图 15 边缘情况的推理结果：目标被遮挡、环境昏暗以及强光干扰。

表 18 VehicleFinder 推理速度评估

车型	参数（M）	延迟（毫秒） 8 核 ARM v8.2	延迟（毫秒） i7-12700	延迟（毫秒） RTX A4000
VehicleFinder(CTIM)	8.81	279.35	169.34	97.72
VehicleFinder(CTIM-Conv-CosineD-Linear)	41.64	365.77	212.59	138.57
VehicleFinder(CTIM-DSConv-Siamese)	104.69	398.35	243.93	171.45
VehicleFinder(CLIP)	107.55	2398.77	1487.53	588.56
VehicleFinder(Bert-Siamese)	194.50	3094.33	1809.51	973.83

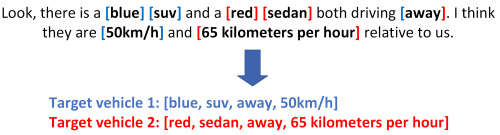


图 16 多实体聚类的挑战。

由感知传感器采集。FindVehicle 是一个命名实体识别数据集，包含平面实体和重叠实体。该数据集中的所有命名实体均由机器标注算法和人工标注人员共同标注。标注内容包括粗粒度和细粒度实体标注。FindVehicle 可用于辅助交通场景中的文本-图像跨模态任务，并可作为交通领域的预训练语料库。此外，我们提出了一种高效的文本-图像跨模态车辆检索系统，名为 VehicleFinder。VehicleFinder 通过文本指令识别目标车辆时，精确率达到 87.7%；在一颗 8 核 ARM v8.2 CPU 上耗时 279.35 毫秒，在一块 RTX A4000 GPU 上耗时 93.72 毫秒。VehicleFinder 可帮助交通监管人员根据自然语言，从大量图像或视频中找出目标车辆。最后，我们构建了一个名为 Vehicle-TI 的文本到图像车辆匹配数据集。

未来，首先，我们将继续维护 FindVehicle。其次，我们将通过添加一些特殊交通场景的语料库来扩展 FindVehicle，并将 FindVehicle 的样本与真实交通场景的图像关联起来，从而形成一个新的数据集（基准数据集）。第三，我们将探索文本-视频跨模态车辆检索。

9 讨论

讨论分为两部分：FindVehicle 面临的挑战，以及我们的跨模态车辆检索系统 VehicleFinder 的局限性。

在 FindVehicle 中，长尾数据分布、分布外车辆品牌的识别，以及细粒度实体和重叠实体的识别，是三个值得探索的挑战。此外，如图 16 所示，判断提取出的命名实体是否指代同一辆车辆也是一项相当大的挑战，这相当于根据上下文对命名实体进行聚类。

我们的 VehicleFinder 的第一个局限是，一段描述只能包含一辆车的属性。由于我们在关键词提取中仅采用命名实体识别，而没有将命名实体识别与关系抽取结合起来，因此 VehicleFinder 无法适应涉及多辆车的上下文，这也是未来面临的一项挑战。第二个局限是，用于描述车辆属性的关键词粒度不够细，这归因于标注工作的人力成本限制。我们将继续关注并研究这一领域。

致谢。 作者感谢 XJTLU-JITRI工业技术学院为联合项目提供宝贵支持。本工作还得到西交利物浦大学（XJTLU）人工智能大学研究中心、江苏省西交利物浦大学数据科学与认知计算工程研究中心的部分支持（资助编号：XJTLU-REF-21-01-002）。作者诚挚感谢 Sihao Dai、Zhou Yuan 和 Wenjie Zhou 在项目中提供的帮助。

声明

- **数据和材料的可用性**：所有数据均可在第一作者的 GitHub 仓库中获取。
- **代码可用性**：所有代码均可在第一作者的 GitHub 仓库中获取。
- **利益冲突**：作者声明不存在利益冲突。

参考文献

[1] Wen, L.、Du, D.、Cai, Z.、Lei, Z.、Chang, M.-C.、Qi, H.、Lim, J.、Yang, M.-H.、Lyu, S.：UA-DETRAC：用于多目标检测与跟踪的新基准数据集和协议。计算机视觉与图像理解 **193**，102907（2020）[2]Hongye, L.、Tian, Y.、Wang, Y.、Pang, L.、Huang, T.：深度相对距离学习：区分相似车辆。计算机视觉与模式识别（2016）[3]Liu, X.、Liu, W.、Mei, T.、Ma, H.：一种基于深度学习的城市监控渐进式车辆再识别方法。欧洲计算机视觉会议（2016）[4] Liu, X.、Liu, W.、Ma, H.、Fu, H.：城市监控视频中的大规模车辆再识别。多媒体与博览会国际会议（2016）[5] Adaimi, G.、Kreiss, S.、Alahi, A.：基于置信度的深度视觉再识别。《交通研究C辑：新兴技术》**126**，103067（2021）[6] El Hamdani, S.、Benamar, N.、Younis, M.：智能交通系统中的行人支持：挑战、解决方案与开放问题。《交通研究C辑：新兴技术》**121**，102856（2020）[7] Ganin, A.A.、Mersky, A.C.、Jin, A.S.、Kitsak, M.、Keisler, J.M.、Linkov, I.：智能交通系统（ITS）的韧性。《交通

28 *FindVehicle* 与 *VehicleFinder*《研究：C辑：新兴技术》 **100**，318–329（2019）[8]Chien, C.-F., Chen, H.-T., Lin, C.-Y.：《基于蓝牙信标的低成本路边停车管理系统》。《传感器》**20(16)**，4559（2020）[9] Sharma, P., Singh, A., Singh, K.K., Dhull, A.：《面向智能交通系统的改进的基于区域的卷积网络车辆识别》。《多媒体工具与应用》**81(24)**，34893–34917（2022）[10] Kong, F., Zhou, Y., Chen, G.：《基于无线传感器网络的智能交通系统多媒体数据融合方法》。《多媒体工具与应用》**79(47)**，35195–35207（2020）[11] Park, E.-J., Kim, H., Jeong, S., Kang, B., Kwon, Y.：《基于关键词的车辆检索》。载于：IEEE/CVF计算机视觉与模式识别会议论文集，第4220–4227页（2021）[12]Zhao, C., Chen, H., Zhang, W., Chen, J., Zhang, S., Li, Y., Li, B.：《用于基于自然语言的车辆检索、结合空间关系建模的对称网络》（2022）[13] Bai, S., Zheng, Z., Wang, X., Lin, J., Zhang, Z., Zhou, C., Yang, H., Yang, Y.：《连接语言与视觉以实现基于自然语言的车辆检索》。载于：IEEE/CVF计算机视觉与模式识别会议论文集，第4034–4043页（2021）[14] Xu, B., Xiong, Y., Zhang, R., Feng, Y., Wu, H.：《基于自然语言的车辆检索及显式跨模态表征学习》。载于：IEEE/CVF计算机视觉与模式识别会议论文集，第3142–3149页（2022）[15] Nguyen, T.M., Pham, Q.H., Doan, L.B., Trinh, H.V., Nguyen, V.-A., Phan, V.-H.：《用于基于自然语言的车辆检索的对比学习》。载于：IEEE/CVF计算机视觉与模式识别会议论文集，第4245–4252页（2021）[16] Feng, Q., Ablavsky, V., Sclaroff, S.：《Cityflow-nl：通过自然语言描述在城市级范围内跟踪和检索车辆》。arXiv：计算机视觉与模式识别（2021）[17] Zhang, J., Lin, X., Jiang, M., Yu, Y., Gong, C., Zhang, W., Tan, X., Li, Y., Ding, E., Li, G.：《用于基于自然语言的车辆检索的多粒度检索系统》（2022）[18]Deruyttere, T., Vandenhende, S., Grujicic, D., Van Gool, L., Moens, M.F.：《Talk2car：掌控你的自动驾驶汽车》。载于：会议论文集

发表于2019年自然语言处理实证方法会议和第9届自然语言处理国际联合会议（EMNLP-IJCNLP），第2088–2098页（2019）[19] Radford, A., Kim, J.W., Hallacy, C., Ramesh, A., Goh, G., Agarwal, S., Sastry, G., Askell, A., Mishkin, P., Clark, J., 等：从自然语言监督中学习可迁移的视觉模型。载于：国际机器学习会议，第8748–8763页（2021）。PMLR[20] Rao, Y., Zhao, W., Chen, G., Tang, Y., Zhu, Z., Huang, G., Zhou, J., Lu, J.：Denseclip：利用上下文感知提示的语言引导密集预测。载于：IEEE/CVF计算机视觉与模式识别会议论文集，第18082–18091页（2022）[21] Morwal, S., Jahan, N., Chopra, D.：使用隐马尔可夫模型（HMM）进行命名实体识别。《国际自然语言计算期刊》（IJNLC）第**1** 卷（2012）[22] Xu, Z., Qian, X., Zhang, Y., Zhou, Y.：基于条件随机场的分词、命名实体识别甚至词性标注混合模型。载于：第六届SIGHAN中文语言处理研讨会论文集（2008）[23] Gui, T.,Ma, R., Zhang, Q.,Zhao, L., Jiang, Y.-G., Huang, X.：基于CNN并重新思考词典的中文命名实体识别。载于：IJCAI，第4982–4988页（2019）[24] Huang, Z., Xu, W., Yu, K.：用于序列标注的双向LSTM-CRF模型。arXiv预印本arXiv:1508.01991（2015）[25] Li, X., Yan, H., Qiu, X., Huang, X.：Flat：使用平面晶格Transformer进行中文命名实体识别。arXiv预印本arXiv:2004.11795（2020）[26] Sui, Y., Bu, F., Hu, Y., Yan, W., Zhang, L.：Trigger-GNN：用于嵌套命名实体识别的基于触发器的图神经网络。arXiv预印本arXiv:2204.05518（2022）[27]Sang, E.F., De Meulder, F.：CONLL-2003共享任务简介：语言无关的命名实体识别。arXiv预印本cs/0306050（2003）[28] Balasuriya, D., Ringland, N., Nothman, J., Murphy, T., Curran, J.R.：维基百科中的命名实体识别。载于：2009年“人民网络与自然语言处理相遇：协作构建的语义资源”研讨会（人民网络）论文集，第10–18页（2009）[29] Derczynski, L., Nichols, E., van Erp, M., Limsopatham, N.：WNUT2017新实体与新兴实体识别共享任务结果。载于：

30 *FindVehicle* 和 *VehicleFinder*《第三届噪声用户生成文本研讨会论文集》，第 140–147 页（2017 年）[30] Weischedel, R., Palmer, M., Marcus, M., Hovy, E., Pradhan, S., Ramshaw, L., Xue, N., Taylor, A., Kaufman, J., Franchini, M. 等：OntoNotes 5.0 版本发布，ldc2013t19。语言数据联盟，宾夕法尼亚州费城**23**（2013 年）[31] Ding, N., Xu, G., Chen, Y., Wang, X., Han, X., Xie, P., Zheng, H., Liu, Z.：Few-NERD：一个少样本命名实体识别数据集。载于：《计算语言学协会第 59 届年会及第 11 届自然语言处理国际联合会议论文集》（第 1 卷：长篇论文），第 3198–3213 页（2021 年）[32] Li, J., Fei, H., Liu, J., Wu, S., Zhang, M., Teng, C., Ji, D., Li, F.：将统一命名实体识别视为词-词关系分类。arXiv 预印本 arXiv:2112.10070（2021 年）[33] Scribano, C., Sapienza, D., Franchini, G., Verucchi, M., Bertogna, M.：尽可能多地嵌入：利用时空 Transformer 进行基于自然语言的车辆检索。载于：IEEE/CVF计算机视觉与模式识别会议论文集，第 4253–4262 页（2021 年）[34]Khorramshahi, P., Rambhatla, S.S., Chellappa, R.：迈向准确的视觉与基于自然语言的车辆检索系统。载于：IEEE/CVF计算机视觉与模式识别会议论文集，第 4183–4192 页（2021 年）[35]Sun, Z., Liu, X., Bi, X., Nie, X., Yin, Y.：DUN：用于基于自然语言的车辆检索的双路径时序匹配网络。载于：IEEE/CVF计算机视觉与模式识别会议论文集，第 4061–4067 页（2021 年）[36] Le, H.D.-A., Nguyen, Q.Q.-V., Nguyen, V.A., Nguyen, T.D.-M., Chung, N.M., Thai, T.-T., Ha, S.V.-U.：结合领域自适应知识、通过自然语言描述检索跟踪车辆。载于：IEEE/CVF计算机视觉与模式识别会议论文集，第 3300–3309 页（2022 年）[37] TT PHUNG, T., Q. LY, N., T. VO, T., TN HO, M.：用于车辆检索的深度特征学习网络。载于：2021 年第五届机器学习与软计算国际会议论文集，第 18–21 页（2021 年）[38] Devlin, J., Chang, M.-W., Lee, K., Toutanova, K.：BERT：用于语言理解的深度双向 Transformer 预训练。arXiv 预印本 arXiv:1810.04805（2018 年）

[39]Floridi, L., Chiriatti, M：Gpt-3：其性质、范围、局限与后果。《思维与机器》**30**(4),681–694 (2020)[40] Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., Fei-Fei, L：ImageNet：大型分层图像数据库。载于：2009年IEEE计算机视觉与模式识别会议论文集，第248–255页（2009）。IEEE[41]Goel, S., Bansal, H., Bhatia, S., Rossi, R.A., Vinay, V., Grover, A：CyCLIP：循环式对比语言-图像预训练。arXiv预印本 arXiv:2205.14459（2022）.[42] Stubbs, A., Uzuner, O：用于去标识化的纵向临床叙事标注：2014年i2b2/uthealth语料库。《生物医学信息学杂志》**58**，20–29（2015）[43] RangiLyu：NanoDet-Plus：超快速、高精度的轻量级无锚框目标检测车型。https://github.com/RangiLyu/nanodet (2021)[44] Chollet, F：Xception：采用深度可分离卷积的深度学习。载于：IEEE计算机视觉与模式识别会议论文集，第1251–1258页（2017）[45]Agarap, A.F：使用修正线性单元（ReLU）进行深度学习。arXiv预印本 arXiv:1803.08375（2018）[46] Bojanowski, P., Grave, E., Joulin, A., Mikolov, T：通过子词信息丰富词向量。《计算语言学协会汇刊》**5**，135–146（2017）[47] Mikolov, T., Chen, K., Corrado, G., Dean, J：高效估计向量空间中的词表示。arXiv预印本 arXiv:1301.3781（2013）[48]Pennington, J., Socher, R., Manning, C.D：GloVe：词语表示的全局向量。载于：2014年自然语言处理经验方法会议（EMNLP）论文集，第1532–1543页（2014）[49] Souza, F., Nogueira, R., Lotufo, R：使用BERT-CRF进行葡萄牙语命名实体识别。arXiv预印本 arXiv:1909.10649（2019）[50] Schweter, S., Akbik, A：FLERT：用于命名实体识别的文档级特征。arXiv预印本 arXiv:2011.06993（2020）[51] Loshchilov, I., Hutter, F：解耦权重衰减正则化。arXiv预印本 arXiv:1711.05101（2017）[52]VILCEK, A., MOTTAGHINEJAD, S., SHI, S., GUPTE, K.,

PASUMARTY, S.,PANG,L., MEHROTRA, P：基于Transformer的深度孪生网络，用于大规模产品匹配和单样本层级分类（2018）