




FindVehicle and VehicleFinder: a NER dataset for natural language-based vehicle retrieval and a keyword-based cross-modal vehicle retrieval system

Runwei Guan^{1,2,3,4} · Ka Lok Man² · Feifan Chen² · Shanliang Yao^{1,2,3,4} ·
Rongsheng Hu⁵ · Xiaohui Zhu² · Jeremy Smith¹ · Eng Gee Lim² ·
Yutao Yue^{3,4,6} 

Received: 6 December 2022 / Revised: 13 May 2023 / Accepted: 16 July 2023
© The Author(s) 2023

Abstract

Natural language (NL) based vehicle retrieval is a task aiming to retrieve a vehicle that is most consistent with a given NL query from among all candidate vehicles. Because NL query can be easily obtained, such a task has a promising prospect in building an interactive intelligent traffic system (ITS). Current solutions mainly focus on extracting both text and image features and mapping them to the same latent space to compare the similarity. However, existing methods usually use dependency analysis or semantic role-labelling techniques to find keywords related to vehicle attributes. These techniques may require a lot of pre-processing and post-processing work, and also suffer from extracting the wrong keyword when the NL query is complex. To tackle these problems and simplify, we borrow the idea from named entity recognition (NER) and construct FindVehicle, a NER dataset in the traffic domain. It has 42.3k labelled NL descriptions of vehicle tracks, containing information such as the location, orientation, type and colour of the vehicle. FindVehicle also adopts both overlapping entities and fine-grained entities to meet further requirements. To verify its effectiveness, we propose a baseline NL-based vehicle retrieval model called VehicleFinder. Our experiment shows that by using text encoders pre-trained by FindVehicle, VehicleFinder achieves 87.7% precision and 89.4% recall when retrieving a target vehicle by text command on our home-made dataset based on UA-DETRAC [1]. From loading the command into VehicleFinder to identifying whether the target vehicle is consistent with the command, the time cost is 279.35 ms on one ARM v8.2 CPU and 93.72 ms on one RTX A4000 GPU, which is much faster than the Transformer-based system. The dataset is open-source via the link <https://github.com/GuanRunwei/FindVehicle>, and the implementation can be found via the link <https://github.com/GuanRunwei/VehicleFinder-CTIM>.

Both are equally contributed to this work.

✉ Yutao Yue
yueyutao@idpt.org

Extended author information available on the last page of the article

Keywords Cross modal learning · Named entity recognition · Intelligent traffic system · Vehicle retrieval · Human-computer interaction · Object detection

1 Introduction

Vehicle retrieval is a task that aims to find the target vehicle from a large image gallery given a query image, which is an image-to-image matching technique also known as vehicle re-identification [2–5]. It has promising prospects in building ITS [6–9] for smart cities [10]. However, an image-based vehicle retrieval system also has its defects in practice. For example, such a system needs an image to provide characteristics of the target vehicle, which is not always easy to obtain in the real world. The performance of an image-based vehicle retrieval system may also be limited because there is only one type of modality to provide spatial and temporal information.

To alleviate these problems, Natural Language (NL), as another essential modality in the real world, has received more and more attention from researchers in recent years. A natural language-based vehicle retrieval system aims to identify the target vehicle using an NL description. Such a system has a broader range of application scenarios, such as finding a vehicle when a bystander provides only an informal description. Most current natural language vehicle retrieval implementations construct the text encoder and visual encoder to extract features from both data types. They then project the obtained text and visual embeddings into the same latent space to compare their similarity. In addition, both visual and NL data will be carefully modified by these methods for more effective representation. For example, vehicle track images are cropped to generate a global motion image [11–14]. As for NL, some keywords related to vehicle attributes (e.g., colour, vehicle type and orientation) are extracted in the given NL query [11, 12, 14, 15]. Although these works achieve acceptable performance on the CityFlow-NL [16] benchmark, they can still be improved, especially in terms of NL. We find that when implementing keyword extraction, existing methods are usually based on dependency analysis (e.g., using NLTK package) or semantic role labelling techniques to determine whether the word is a keyword or not. These techniques only assign the part of speech to the words in the sentence. It means that pre-determined rules and post-processing are required to divide the extracted keywords into corresponding vehicle attributes, making the whole process complex [15, 17]. Such methods can also suffer from extracting the wrong keyword if the NL description is complex. This can lead to error propagation in subsequent modules and reduce model performance.

In fact, keyword extraction is already a mature technology in natural language processing (NLP), also known as named entity recognition (NER). The main obstacle that prevents us from applying the state-of-the-art NER model to solve the above problem is the lack of a domain-specific corpus with high-quality annotations. Therefore, to alleviate this problem, we propose a named entity labelled natural language dataset focused on the traffic domain, called FindVehicle. It consists of descriptions of the vehicle from the point of view of urban traffic surveillance cameras. Some example descriptions from our dataset are shown. We also compare them with instances selected from other traffic domain datasets using natural language, namely Talk2Car [18] and CityFlow-NL [16]. All details are given in Table 1. We carefully construct the vehicle descriptions to match real traffic scenarios and to enrich more detailed information about the target vehicles. Our dataset includes eight types of vehicle features, namely vehicle location, orientation, brand, model, type, colour, distance from the traffic surveillance camera, and velocity. In contrast, Talk2Car [18] only records vehicle type,

Table 1 Datasets of Vehicle Retrieval

Dataset	Data Samples	Informative[1]	Amount	HasNER
Talk2Car[2]	My friend is getting out of the car. Stop and let me out too! Yeah that would be my son on the stairs next to the bus. My mum is on the right! Park near her, she might want a lift. A cargo truck drives down an intersection with many smaller cars.	Low	11959	No
CityFlow-NL[3]	The large green flatbed 18 wheeler is going straight. A green truck drives through an intersection, followed by a sedan.	Medium	9374	No
FindVehicle	A [grey] [sedan] drives [right] at a speed of [58km per hour]. Maybe a [[Ford] [Mondeo]][4]. A [Volvo] [truck] is parked on the side of the road , behind a [[Ferrari] [458]]. The [blue] [[BMW] [320]] driving [away] is [150 meters] away from us	High	42341	Yes

and CityFlow-NL [16] has only four types of information, which are vehicle colour, type, action, and scene. More vehicle information in the description text means that the data can more accurately reflect the traffic scene in real life while reducing the challenge in NL-based vehicle retrieval tasks caused by the ambiguity of natural language. Both FindVehicle and CityFlow-NL [16] have the description of the relationship with other vehicles (surrounding vehicles). Therefore, we do not treat the surrounding vehicle as a separate feature. Furthermore, FindVehicle is annotated with multi-granularity named entity labels in order to be able to meet further requirements in the future.

To verify the effectiveness of the proposed dataset, we construct a simple and highly efficient cross-modal vehicle retrieval system called VehicleFinder. Unlike current transformer-based models [19, 20], which have huge parameters and slow inference time, VehicleFinder has only 8.81 million parameters. This means that it can achieve real-time performance in the actual scenario and is more friendly to edge devices. VehicleFinder is trained and tested on our homemade text-to-image dataset called Vehicle-TI based on the training set of UA-DETRAC [1]. The keywords fed into VehicleFinder are extracted by a NER model pre-trained on FindVehicle. The experiment result shows that VehicleFinder gets 87.7% precision and 89.4% recall when detecting the vehicle. Its latency is 279.35 ms on one 8-core ARM v8.2 CPU.

To conclude, the main contributions of this paper include:

1. We propose the first NER dataset (benchmark) in the traffic domain called FindVehicle, which has 42.3 thousand sentences, 1.361 million tokens, 202.5 thousand entities and 21 entity types. FindVehicle is not only a dataset that contains both flat and overlapping entities, but also has both coarse-grained and fine-grained entity types.
2. We propose a text-image cross-modal vehicle retrieval system called VehicleFinder to prove the effectiveness of our proposed NER dataset. VehicleFinder is a highly efficient model with favourable performance that can achieve real-time performance and be applied to edge devices.
3. During the experiment, we construct a text-image vehicle matching dataset called Vehicle-TI. Vehicle-TI has 335,040 training samples, 179,520 test samples and 83,776 validation samples.

The rest of this paper is organized as follows: Section 2 presents the related work of our paper; Section 3 presents the critical information of FindVehicle and how we construct it; Section 4 presents the statistics details of FindVehicle; Section 5 presents VehicleFinder, our text-image cross-modal vehicle retrieval system; Section 6 includes the baselines of FindVehicle; Section 7 presents the experiment details of VehicleFinder; Section 8 presents the conclusion of this paper and our future work; Section 9 presents some challenges of FindVehicle.

2 Related work

2.1 Named entity recognition

Named entity recognition (NER) is a classical sequence tagging task in NLP. It is to locate and classify the words or sentences with specific types in the text. The input of NER model is a sequence with part-of-speech (POS) taggings, as it shows in Equation 1,

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

where n denotes the number of words segmented by word segmentation program. t_i is the POS of the word w_i .

The process of NER based on word segmentation and POS tagging is splitting, combining (determining entity boundaries) and reclassifying (determining entity categories) some words. The output is an optimal sequence WC^* , TC^* with a pair format of (*word category* (WC), *tagging category* (TC)), as it shows in Equation 2,

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

where $m \leq n$, $wc_i = [w_j, \dots, w_{j+k}]$, $tc_i = [t_j, \dots, t_{j+k}]$, $1 \leq k$, $j + k \leq n$.

In brief, the NER modal could be written as Equation 3 shows,

$$(WC^*, TC^*) = \operatorname{argmax}_{(WC, TC)} P(WC, TC | W, T) \quad (3)$$

where W is the word sequence while T is the tagging sequence. $P(\cdot)$ is a conditional probability model.

Hidden Markov Models [21] and Conditional Random Fields [22] are two typical machine learning models for NER. Convolutional neural network [23], recurrent neural network [24], transformer [25], and graph neural network [26], these deep learning models all achieve the state-of-the-art results in NER.

Moreover, many NER datasets have been proposed in past years. These [27–31] are the well-known NER datasets (benchmarks). In these datasets, there are mainly three kinds of named entities: flat entity, overlapped entity and discontinuous entity. [32] proposed a unified neural framework to concurrently solve the three NER problems.

2.2 Text-image vehicle retrieval

Vehicle retrieval based on test-image cross-modal learning is a hot spot these years [11, 13–15, 33–37]. The model could find out the highest matching vehicle based on the description with the text format. There are mainly two formats according to the architecture. The first is the end-to-end neural network based on early retrieval, where the features of images and text are fused in the early stage. The second is the non-end-to-end system based on late retrieval, where images and text features are extracted individually and loaded into a decision module.

2.3 Contrastive language image pretraining

Contrastive language image pretraining (CLIP) combines the modalities of language and image in one neural network, which is mainly for multi-modal tasks based on natural language and computer vision. Prior to this, most computer vision work was trained based on pre-defined labels, and supervision limited the generalization and usefulness of neural networks. There has been a lot of work in the field of NLP using a large amount of corpus data for self-supervised learning, and the effect of these models has surpassed manually labelled datasets [38, 39]. In the field of CV, the current mainstream method is still to use large-scale datasets with labelled information for pre-training [40]. Vanilla CLIP [19] creatively uses text as a supervision signal to train a vision model and achieves conspicuous results on ImageNet [40]. In addition, vanilla CLIP [19] is also very good at zero-shot tasks. [20] proposes a CLIP framework called DenseCLIP, which is good at dense prediction tasks, such as semantic segmentation and dense object detection. [41] proposes a new contrastive loss to normalize the location and geometric information of image and text features in the semantic space.

3 The construction of FindVehicle

3.1 Brief introduction

FindVehicle is the first NER dataset in traffic. It is based on the image samples of UA-DETRAC [1]. FindVehicle contains various descriptions of traffic participants on the road from the view of traffic surveillance cameras, mainly vehicles. A description contains many attributes of one or several vehicles. These attributes all could be detected by traffic sensors, such as surveillance cameras, lidar and radar. Moreover, FindVehicle also incorporates much real-world prior knowledge, such as the vehicle brand and model. Furthermore, FindVehicle contains both coarse-grained and fine-grained entities. Entities include both flat and overlapped entities.

3.2 Entity types

As Fig. 1 shows, there are 21 entity types in FindVehicle, 8 coarse-grained entities and 13 fine-grained entities. These entities are all the attributes of vehicles, which all follow the distribution of the real world. Moreover, FindVehicle also contains both flat and overlapped entities.

3.2.1 Coarse-grained entity

There are 8 kinds of coarse-grained entities, including *vehicle_location*, *vehicle_orientation*, *vehicle_brand*, *vehicle_model*, *vehicle_type*, *vehicle_color*, *vehicle_range* and *vehicle_velocity*.

vehicle_location indicates the locations of vehicles from the view of the traffic surveillance cameras, such as *bottom right*, *top-left*, etc.

vehicle_orientation indicates the directions of vehicles' heads from the view of the traffic surveillance cameras, such as *this way*, *away*, etc.

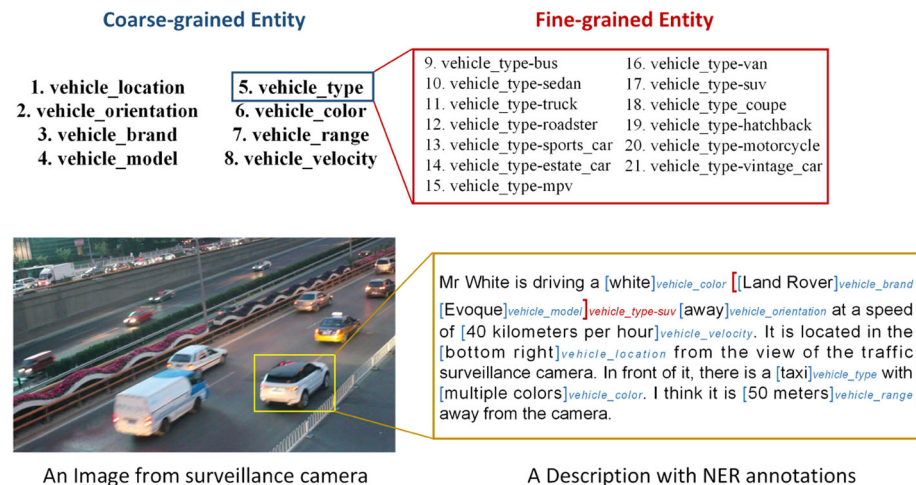


Fig. 1 Entity types and an annotated sample of FindVehicle. Images are from UA-DETRAC [1]

vehicle_brand indicates the brands of vehicles. FindVehicle contains 65 vehicle brands all over the world.

vehicle_model indicates the models of vehicle brands. There are 4793 models of different vehicle brands in FindVehicle. For example, Q7 is one of the models of Audi.

vehicle_type indicates the types of vehicles, such as sedan, suv, etc.

vehicle_color indicates the colors of vehicles, such as silver grey, rose red, etc.

vehicle_range indicates the distance between the vehicle and the traffic surveillance camera, such as 18m, 123 meters, etc.

vehicle_velocity indicates the speed of the moving vehicle on the road, such as 50 kilometres per hour, 120 km/h, etc.

3.2.2 Fine-grained entity

As it shows in Fig. 1, in FindVehicle, there are 13 kinds of fine-grained entities, which belong to the coarse-grained entity *vehicle_type*, for example, BMW X5 is a fine-grained entity of *vehicle_type-suv*. Fine-grained entities contain the human prior knowledge of cars.

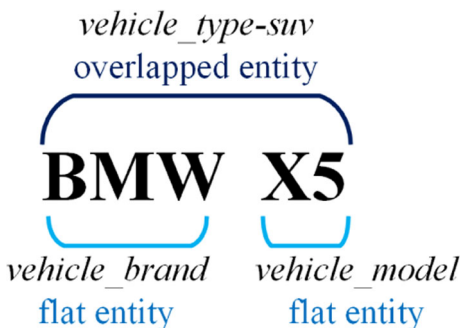
3.2.3 Flat and overlapped entity

Overlapped entities exist in coarse-grained entities *vehicle_brand*, *vehicle_model* and fine-grained entities *vehicle_type*-.*. For example, as Fig. 2 shows, the label of BMW is *vehicle_brand* while the label of X5 is *vehicle_model*, for a car enthusiast, the label of BMW X5 is *vehicle_type-suv*.

3.3 Corpus collection

As Fig. 4 shows, the corpus collection includes two parts, the corpus with simple context and the corpus with complex context. The corpus with simple context denotes the short sentences, which are presented in the column of Data Samples in Table 1. As Fig. 3 presents, firstly, we sample some target vehicles with bounding boxes and labels in UA-DETRAC [1]. Based on these samples, we create a relational table to save the attributes of the corresponding vehicle. Each item in the table represents one vehicle with several attributes. Furthermore, to increase the complexity of the dataset, we replace some formal phrase-type and word-type entities with our informal expression habits and add some rare entities which do not exist in UA-DETRAC [1]. Moreover, for the entity generation of three entities *vehicle_brand*,

Fig. 2 An example of flat and overlapped entities



Relational tables of vehicle attributes

Based on UA-DETRAC

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
.....

By car enthusiasts & Wikipedia

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
.....

insert

Descriptive sentences with various patterns

I want to find out the [vehicle_type] located in the [vehicle_location 1] and [vehicle_location 2] whose colors are [vehicle_color 1] and [vehicle_color 2]. I think they are about [vehicle_range] away from me.

There is a [vehicle_color 1] [vehicle_brand 1] [vehicle_model 1] and a [vehicle_color 2] [vehicle_brand 2] [vehicle_model 2] in the [vehicle_location] of the image that driven

Wow, there is a [vehicle_color 1] [vehicle_type 1] driving [vehicle_orientation 1] with [vehicle_velocity 1] and a [vehicle_color 2] [vehicle_type 2] driving [vehicle_orientation 2] at a speed of

.....

Fig. 3 The generation of corpus with simple context

vehicle_model and *vehicle_type**, we invite three car enthusiasts to collect and integrate data based on their extensive car knowledge and the search results of Wikipedia. They write data with different expressions and curate 65 vehicle brands, 4793 vehicle models and 13 vehicle types in total. Secondly, we recruit four volunteers to write descriptive sentences with various patterns in their tone and expression habits. All volunteers are well-educated and have adequate English linguistic knowledge. Thirdly, we insert the target vehicles with their attributes into these patterns by our sentence auto-generation framework.

As the sample in Fig. 1 presents, the corpus with complex context indicates narrative long sentences or paragraphs with persons' subjective emotions and imagination. Instead of generating a corpus with simple context by combining labor and computers, a corpus with complex context is made by human beings only. Four members of our team write down the corresponding sentences and paragraphs with their own writing habits and imagination by observing the images in UA-DETRAC [1].

3.4 NER annotation

As Fig. 4 shows, in our NER annotation framework, there are two processes for the corpus with simple and complex contexts, respectively. The annotations of the corpus with simple context are completed simultaneously with sentence auto-generation by our annotation auto-generation framework. After that, the correction framework of auto-generation will automatically identify whether the NER annotations by the auto-generation framework have errors. If the data had an error, the annotation process would be interrupted and report the location of the error, and then we would check and fix it. If it had no error, the corpus with annotations would be loaded into the dataset directly.

The annotations of the corpus with complex context are totally manual. They are based on the common sense and knowledge of annotators. Annotators are all volunteers who are knowledgeable about vehicles and good at narrative writing.

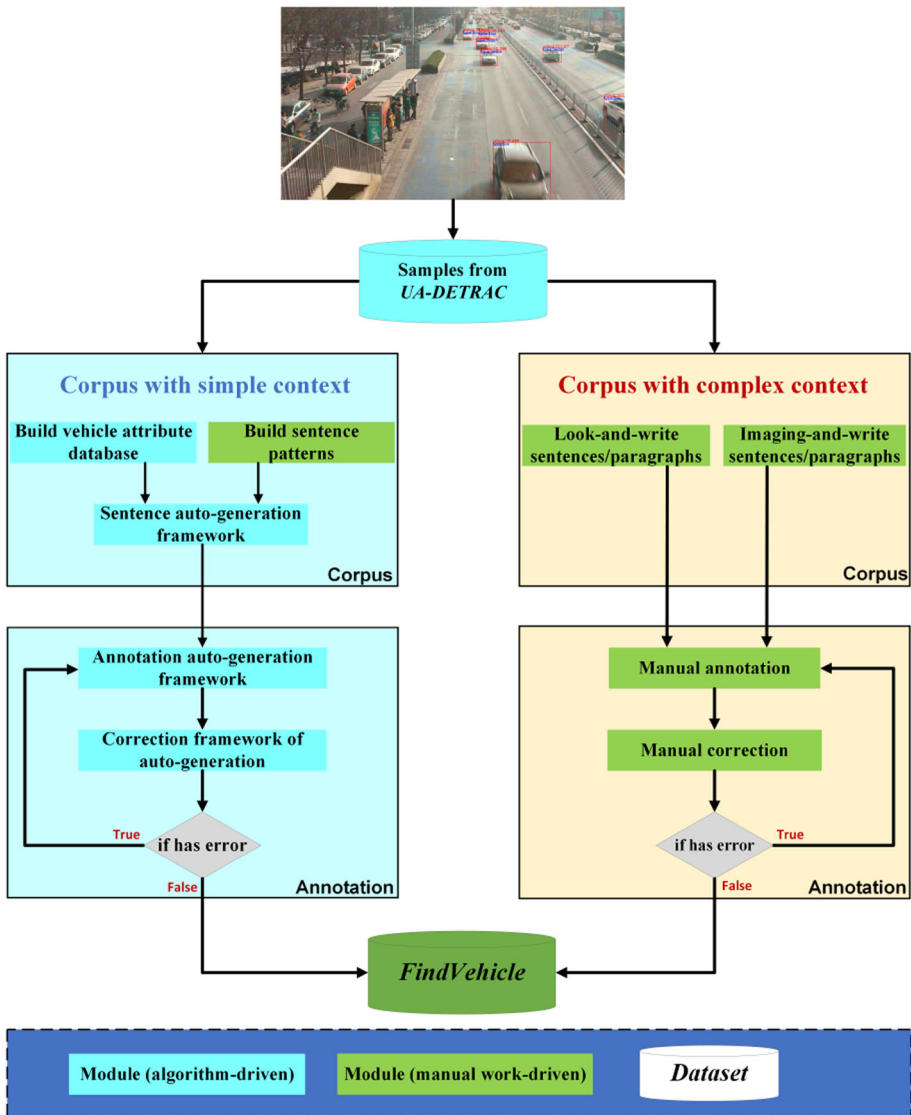


Fig. 4 The framework of corpus collection and annotation of FindVehicle

As Fig. 5 shows, we organize the data in two formats, JSON and CoNLL-style [27]. The value of the key *ner_label* is the annotated named entities. The values of *ner_label* in each element is [entity type, start index of char span, end index of char span, start index of token span, end index of token span]. Our annotation considers char-level and token-level, meeting different needs of the NER models. The key *re_label* denotes the indexes of values of *ner_label* that refer to one target in the context of a sentence.

Json	CoNLL-Style
<pre> { id: 1, data: "Just now, a grass green minivan loaded with oranges driven by Mr. Green rolled over. All the oranges fell off, and the green grass on the roadside was soaked with orange juice. A police car with orange colour is coming." ner_label: [["vehicle_color", 12, 23, 4, 6], ["vehicle_type", 24, 31, 6, 7], ["vehicle_type", 175, 185, 37, 39], ["vehicle_color", 91, 104, 40, 42]] re_label: [[0, 1], [2, 3]] } </pre>	<pre> a grass B-vehicle_color green E-vehicle_color minivan B-vehicle_type A police B-vehicle_type car E-vehicle_type with orange B-vehicle_color colour E-vehicle_color </pre>

Fig. 5 The two annotation formats of FindVehicle

4 Data statistics of FindVehicle

4.1 Size and distribution of FindVehicle

FindVehicle is the first NER dataset in traffic with the annotations of automatic labeling and manual labelling together. As Table 2 shows, we present the statistics of FindVehicle and other widely used well-known NER datasets, including CoNLL'03 [27], WikiGold [28], WNUT'17 [29], I2B2 [42] and OntoNotes [30]. FindVehicle has 42.3 thousand sentences, 1.361 million tokens, 202.5 thousand entities and 21 entity classes. As Fig. 6 presents, the entity types are long-tail distributed to reflect the real-world traffic scenario.

4.2 Dataset split

FindVehicle is a hybrid NER dataset containing both flat and overlapped entities. We split it into a training set and a test set. The details of these two sets are shown in Table 3. For the training set, there are 84.6k coarse-grained entities and 18.2k fine-grained entities. In addition, there are 84.2k flat entities and 18.6k overlapped entities. For the test set, there are

Table 2 Statistics of FindVehicle and other well-known NER datasets

Datasets	Sentences	Tokens	Entities	Entity Classes	Domain
WikiGold [28]	1.7k	39k	3.6k	4	General
WNUT'17 [29]	4.7k	86.1k	3.1k	6	Social Media
CoNLL'03 [27]	22.1k	301.4k	35.1k	4	Newswire
I2B2 [42]	107.9k	805.1k	28.9k	23	Medical
OntoNotes [30]	103.8k	2067k	161.8k	18	General
FindVehicle (ours)	42.3k	1361.1k	202.5k		Traffic

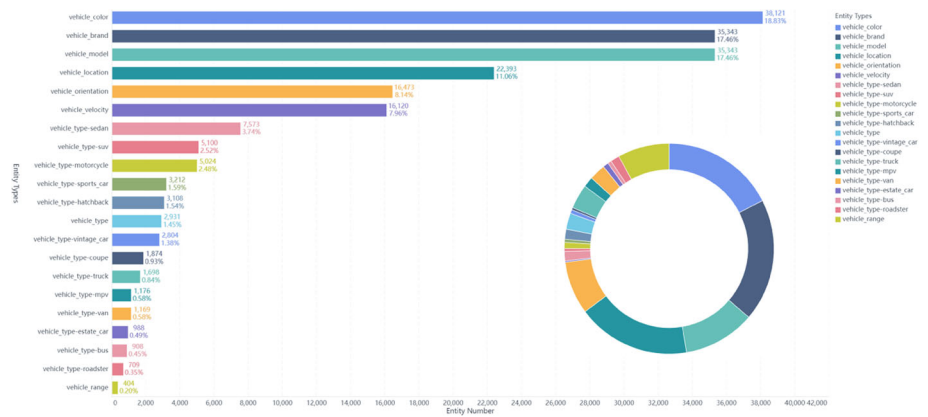


Fig. 6 Statistics by entities in FindVehicle

82.5k coarse-grained entities and 17.4k fine-grained entities. Besides, 82.7k flat entities and 17.2k overlapped entities are in the test set.

5 VehicleFinder

VehicleFinder is a lightweight text-image cross-modal vehicle retrieval system. Users could find out the target vehicle through the description of its type, color and orientation. As Fig. 7 presents, VehicleFinder has two branches. One is to extract proposals by a vision detector while the other is to extract named entities by a text detector. We adopt NanoDet [43] as the vision detector and BiLSTM-CRF [24] as the text detector. The NanoDet [43] is pretrained on UA-DETRAC [1] while the BiLSTM-CRF [24] is pretrained on our FindVehicle. The proposals and name entities will be loaded into the contrastive text-image module (CTIM) to compare the semantic similarity of data of two modalities.

As Fig. 8 shows, there are two encoder branches in CTIM to encode the data of image and text modalities, respectively. The output of CTIM is the similarity of the image and text, whose value domain is between 0 and 1. An output below 0.5 indicates that the image and text are unrelated, while an output above 0.5 indicates that they are related. CTIM is a complete convolution module whose convolution operations are all the depthwise separable convolution [44], dramatically reducing the parameter number, especially in the deep layers of the neural network. CTIM could perform as a plug-and-play module in some cross-modal systems.

In the branch of the image encoder, there are five same encoder units. An encoder unit will initially put the input feature map $x_i \in R^{c \times h \times w}$ into three branches, where c, h, w respectively denote the channel, height and width of a feature map. The first three branches with different convolution kernel sizes are used to extract the feature with different receptive fields. The output feature map $\hat{x}_i \in R^{c \times h \times w}$ will be activated by ReLU [45], then increase the channels and reduce the spatial size through a depthwise separable convolution operation with the 3×3 kernel. After a batch normalization and a ReLU activation, the output feature map is $x_{i+1} \in R^{2c \times \frac{h}{2} \times \frac{w}{2}}$. To alleviate the gradient vanishing and explosion in the training stage, a long residual path with a depthwise separable convolution is connected with the

Table 3 Data Split of FindVehicle

Dataset	Coarse-grained Entities	Fine-grained Entities	Flat Entities	Overlapped Entities
Training set	84.6k	18.2k	84.2k	18.6k
Test set	82.5k	17.4k	82.7k	17.2k

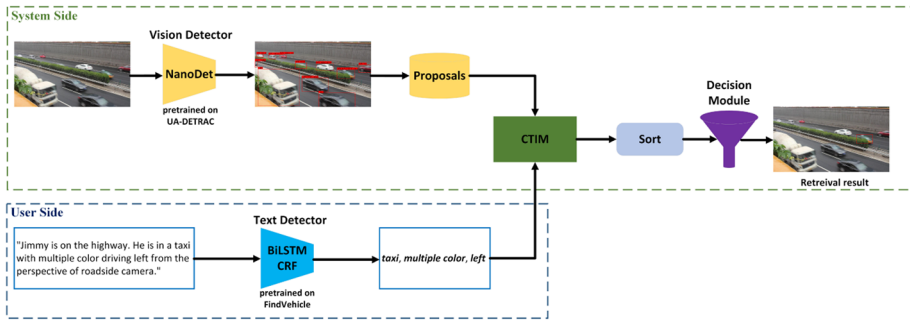


Fig. 7 The architecture of VehicleFinder

output feature map. The final output of the encoder unit is $\hat{x}_{i+1} \in R^{2c \times \frac{h}{2} \times \frac{w}{2}}$. The whole process is presented in Equation 4.

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

In the branch of text encoder, named entities will be firstly embedded by pretrained embeddings of Fasttext (wiki-news-300d-1M) [46]. Fasttext could infer the embeddings of words not in the word dictionary based on the existing words, which is more robust than Word2vec [47] and GloVe [48] for the system. The shape of the embedding matrix is $d \times 300$, where d indicates the number of named entities and 300 is the vector length of each named entity. After that, we adopt four groups of multi-scale depthwise separable convolution operations to extract the feature with different scales concurrently. The first group is n convolution operations of the kernel size $1 \times w_1$, which is to extract the feature of a single word in named entities. The second group has one convolution operation of the kernel size $2 \times w_2$ and $n - 1$ convolution operations of the kernel size $1 \times w_1$, where the convolution of the $2 \times w_2$ kernel is to extract the associated feature of adjacent words. The rest $1 \times w_1$ convolution operations are to enhance the non-linear representation. The third group is firstly processed by a convolution of the $3 \times w_3$ kernel, which is also for the feature extraction of adjacent words with a word window size of three. Then the following operations are the same as the second group. The fourth group is a convolution operation with the kernel size of $d \times w_d$, which is to extract the feature of the global context. Finally, the outputs of these four groups will be added to get a comprehensive representation of the name entities. The four convolution operations are shown in Fig. 9.

After we get the representations of the proposal and named entities, we align their shape to calculate their cosine distance. Cosine distance measures the distance between vectors of the proposal and named entities. It could maintain the same similarity in the high-dimensional case as the low-dimensional case, which is a robust indicator of the relative difference in direction. Equation 5 shows cosine distance.

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

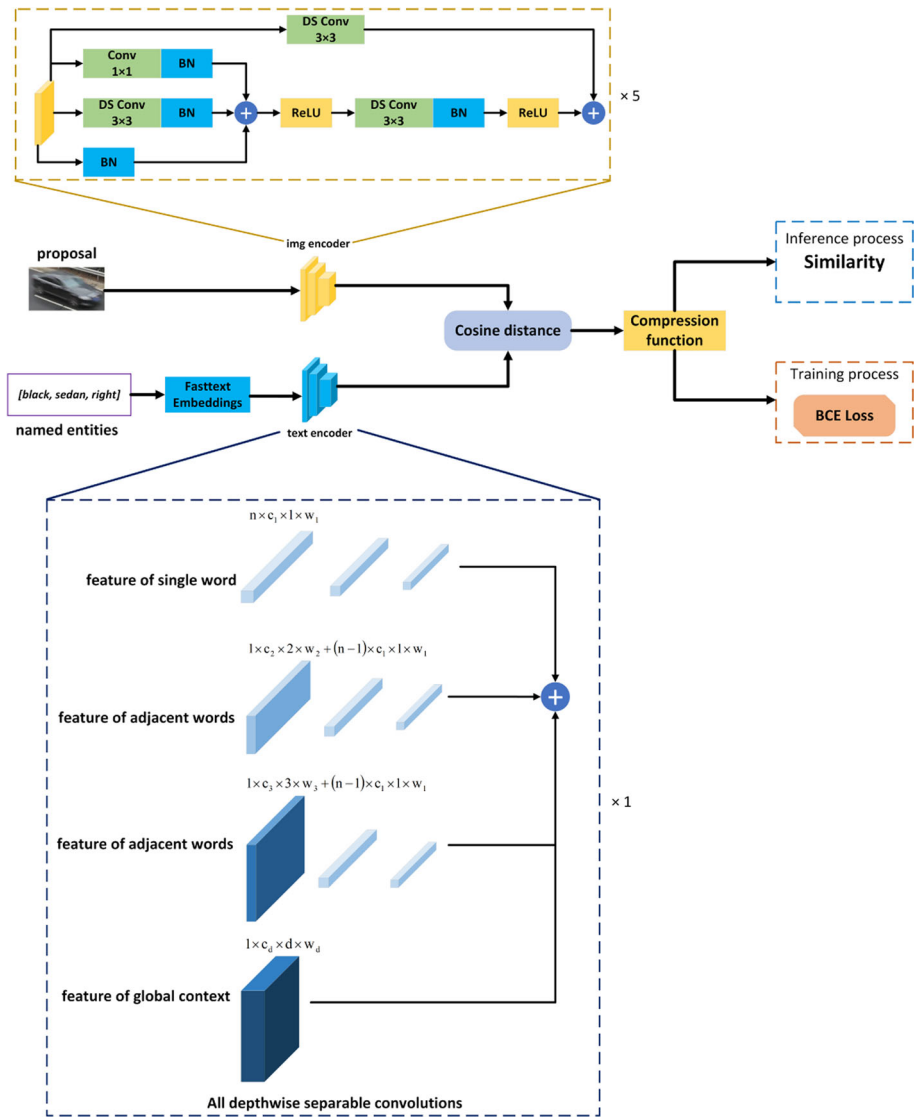


Fig. 8 The architecture of Contrastive Text-Image Module (CTIM). All convolution operations are all depthwise separable convolutions, except for the convolution operation with the kernel size of 1×1 . Because the depthwise separable convolution contains the convolution operation with the 1×1 kernel size. c_i denotes the channel number of the feature map whose kernel height is i . w_i denotes the width of the feature map whose kernel height is i

where n is the number of vector's components. A_i and B_i respectively denote the text and image vector of i th component.

However, the value domain of cosine distance is $[-1, 1]$. It means that the result of cosine distance could not be directly fed to binary cross entropy loss (BCE loss) because BCE loss

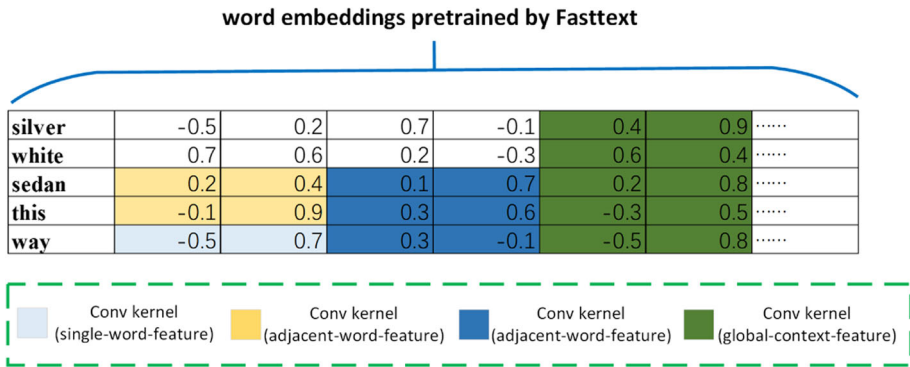


Fig. 9 The four multi-scale convolution operations in our text encoder

(Equation 6) could not process the negative number.

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

where N indicates the number of samples in a batch. $y_i \in \{0, 1\}$ is the ground truth while $\hat{y}_i \in [-1, 1]$ is the result of cosine distance predicted by the neural network. Apparently, \hat{y}_i is not in the definitional domain of $\ln(\cdot)$ if \hat{y}_i is below zero.

Therefore, as Equation 7 presents, we use Equation 7 to compress the results of cosine distance from $[-1, 1]$ to $[0, 1]$, which can be the input to BCE loss. The linear compression function is a monotonically increasing function whose value domain is $[0, 1]$. It is differentiable everywhere. Monotonicity ensures that the relative position of the variable does not change when it maps from $[-1, 1]$ to $[0, 1]$. The property of being differentiated everywhere ensures that it can participate well in backpropagation in neural networks.

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

where $x \in [-1, 1]$ denotes the result calculated by cosine distance.

Therefore, the complete form of the loss function is presented in Equation 8.

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

Finally, our VehicleFinder will calculate the similarities between named entities extracted from the command and object proposals extracted by the vision detector. We will sort object proposals in terms of the similarity with named entities descendingly. The ranking list *proposals* will then be fed to a decision module. In the decision module, considering that the user cannot always describe the vehicle characteristics in detail, we take two patterns of commands into account and process them respectively to enhance the system's robustness, which could also be user-friendly. As Fig. 10 presents, the first is the no-missing-entity pattern and the second is the missing-entity pattern. No-missing-entity pattern indicates the command contains all three named entities, *vehicle_type*, *vehicle_color* and *vehicle_orientation*. Missing-entity pattern indicates the command contains one or two named entities, and the other one or two named entities are not mentioned.



Fig. 10 Two patterns of commands

As Algorithm 1 presents, we firstly set two thresholds th_{nm} and th_m , which mean the threshold for no-missing-entity pattern and the threshold for missing-entity pattern. The variable *proposals* containing *vehicle: sim* pairs indicates object proposals and their similarity with named entities extracted from the command. For no-missing-entity pattern, if existing the *vehicle: sim* pair whose *sim* is larger than th_{nm} , the *vehicle: sim* pair would be appended to *retainVehicle*. If not existing the *vehicle: sim* pair whose *sim* is larger than th_{nm} , the decision module would continue to search for the *vehicle: sim* pair whose *sim* is larger than th_m . If existing the *vehicle: sim* pair whose *sim* is larger than th_m , the *vehicle: sim* pair would also be appended to *retainVehicle*. th_{nm} and th_m are set based on the results of experiments. We assume by default that th_{nm} is greater than th_m .

Algorithm 1 Decision Module

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

```

if proposals.exists(vehicle.sim  $\geq th_{nm}$ ) then
  for p in proposals do
    if p.sim  $\geq th_{nm}$  do
      retainVehicle.append(p)
    end if
  end for
else if proposals.exists( $th_m \leq vehicle.sim < th_{nm}$ ) then
  for p in proposals do
    if p.sim  $\geq th_m$  do
      retainVehicle.append(p)
    end if
  end for
end if
return retainVehicle

```

6 Experiments of FindVehicle

In the experiments of FindVehicle, we make the baselines of our FindVehicle.

6.1 Settings of training and evaluation

We select three representative and state-of-the-art models to train and test on FindVehicle, which were BiLSTM-CRF [24], BERT-CRF [49] and FLERT [50].

BiLSTM-CRF [24] combines BiLSTM and CRF. BiLSTM acts as the encoder layer and takes word embeddings as input, CRF serves as a decoder to determine the tag for each token based on hidden states outputted from an encoder.

BERT-CRF [49] replaces word embeddings of BiLSTM with subword-embeddings learned from BERT, and changes the encoder from BiLSTM to Transformer.

FLERT [50] is a NER model that takes document-level features as an extra account. By adding context text on both sides (left and right) to the query sentence, FLERT captures document-level features and presents a better predict result than the previous model.

For each model, we use the most suitable hyperparameters that make the model converge smoothly. We train and test these models on one TITAN RTX GPU. Table 4 shows the implementation details.

Furthermore, as Equation 9, 10 and 11 present, we choose precision, recall and F1 score as the evaluation metrics of the test, which are based on the confusion matrix (Table 5).

$$Precision = \frac{TP}{TP + FP} \quad (9)$$

$$Recall = \frac{TP}{TP + FN} \quad (10)$$

$$F1 = \frac{2 \times Precision \times Recall}{Precision + Recall} \quad (11)$$

6.2 Baselines of FindVehicle

Table 6 shows the evaluation results of models on the test set of FindVehicle. It is apparent that Transformer-based models perform better than the RNN-based model. BiLSTM-CRF [24] got 49.5% F1 score, which is the lowest value among models. FLERT [50] achieved 80.9% F1 score, which is the highest value, 3% higher than BERT-CRF [49].

Furthermore, we do the statistics on the evaluation results for all 21 classes of named entities. We take the evaluation results of FLERT [50] as the example, as Table 7 shows, all the evaluation metric values of fine-grained entities are much lower than those of coarse-grained entities. It denotes that the recognition of fine-grained entities is harder than coarse-grained entities for neural networks. Moreover, we also calculate the evaluation results of flat entities and overlapped entities by FLERT [50]. As Table 8 shows, the values of three metrics of flat entity are about 20% higher than overlapped entities'. The recognition of overlapped entities is still a challenge in FindVehicle.

6.3 Comparison of models on different NER datasets

We also compare the performances of models on different NER datasets (Table 9), including CoNLL'03 (4 classes) [27], WNUT'17 (6 classes) [29], Ontonotes (18 classes) [30] and our FindVehicle (21 classes). We use F1 score as the evaluation metric. We can see that F1 scores (Equation 11) of three models on FindVehicle are all lower than the scores on CoNLL'03 [27] and Ontonotes (18 classes) [30], which indicates that there are some challenges in our dataset to some extent.

Table 4 Implementation Details of Models on The Training Set of FindVehicle

Model	Epochs	BS	ILR	Opt	Sch ¹
BiLSTM-CRF [24]	80	32	0.001	AdamW	Cosine
BERT-CRF [49]	100	4	0.0015	AdamW	Cosine
FLERT [50]	80	8	0.001	AdamW	Cosine

¹BS means batch size, ILR means initial learning rate, Opt means optimizer, Sch means scheduler

Table 5 Confusion Matrix

Predict Label	Positive	Negative
Positive	True Positive (TP)	False Negative (FN)
Negative	False Positive (FP)	True Negative (TN)

Table 6 Evaluation Results of Three Models on The Test Set of FindVehicle

Model	Precision (%)	Recall (%)	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

Table 7 Evaluation Results of FLERT [50] for All The Classes of FindVehicle

Entity Class	Precision (%)	Recall (%)	F1 (%)
vehicle_color	91.8	91.9	91.8
vehicle_brand	91.9	91.8	91.8
vehicle_model	91.7	91.7	91.7
vehicle_location	91.7	91.8	91.8
vehicle_velocity	90.4	89.8	90.1
vehicle_orientation	91.5	88.7	90.1
vehicle_range	92.1	91.9	92.0
vehicle_type	91.6	91.8	91.7
vehicle_sedan	81.9	83.0	82.4
vehicle_type-suv	84.7	86.1	85.4
vehicle_type-motorcycle	87.7	91.5	89.6
vehicle_type-sports_car	84.1	85.5	84.8
vehicle_type-hatchback	68.8	66.5	67.6
vehicle_type-vintage_car	81.6	82.9	82.2
vehicle_type-coupe	72.7	77.4	75.0
vehicle_type-truck	74.7	81.0	77.7
vehicle_type-van	62.9	74.0	68.1
vehicle_type-mpv	63.1	66.8	64.9
vehicle_type-estate_car	62.0	58.7	60.3
vehicle_type-bus	70.7	68.0	69.3
vehicle_type-roadster	44.9	32.4	37.8

Table 8 Evaluation Results of FLERT [50] for Flat and Overlapped Entities of FindVehicle

Entity Type	Precision (%)	Recall (%)	F1 (%)
Flat Entity	91.6	91.5	91.6
Overlapped Entity	71.5	72.6	71.9

Table 9 Performances of Models on Test Sets of Different NER Datasets

F1 / Models	Datasets			
	CoLL'03 [27]	WNUT'17 [29]	Ontonotes [30]	FindVehicle (ours)
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

Table 10 Implementation Details of NanoDet-m on The Training Set of UA-DETRAC

Model	Epochs	BS	ILR	Opt	Sch ¹
NanoDet-m [43]	200	16	0.001	AdamW	Cosine

¹BS means batch size; ILR means initial learning rate; Opt means optimizer; Sch means scheduler

7 Experiments of VehicleFinder

There are four parts in the experiments of VehicleFinder, which are vision detector, text detector, CTIM and VehicleFinder.

7.1 Experiments of vision detector

Vision detector is to extract proposals of vehicles from the image. We adopt NanoDet-m [43] as the vision detector, which is a lightweight detector with only 0.95 million parameters. It is trained on the training set of UA-DETRAC [1]. The implementation details are shown in Table 10.

Moreover, we want the vision detector to miss as few targets as possible, so we use recall as the evaluation metric instead of precision. As Table 11 presents, NanoDet-m [43] gets 86.7% recall rate on the test set.

7.2 Experiments of text detector

Text detector is to extract keywords (named entities) from the user command. BiLSTM-CRF has relatively few parameters and fast inference among all NER models mentioned in Section 6.2, so we train a BiLSTM-CRF on our FindVehicle as the text detector, which is to extract named entities with types of *vehicle_type*, *vehicle_color* and *vehicle_orientation*. The implementation details are shown in Table 4.

As Table 12 shows, BiLSTM-CRF has 4.02 million parameters. It spends 148.57 ms extracting all named entities from a sample in FindVehicle on the 8-core ARM v8.2. In addition, it spends 87.19 ms and 51.73 ms when tested on i7-12700 and RTX A4000 (Table 13).

7.3 Experiments of CTIM

7.3.1 Settings of training and evaluation

We construct a text-image-pair dataset called Vehicle-TI based on the training set of UA-DETRAC [1] to train and test our CTIM. As Fig. 11 shows, each data sample in Vehicle-TI

Table 11 Evaluation of NanoDet-m on UA-DETRAC [1]

Model	Param(M)	Latency(ms) 8-core ARM v8.2	Latency(ms) i7-12700	Latency(ms) RTX A4000	Recall(%)
NanoDet-m [43]	0.95	7.99	3.72	1.14	86.7

Table 12 Inference Speed Evaluation of BiLSTM-CRF [24]

Model	Param(M)	Latency(ms) 8-core ARM v8.2	Latency(ms) i7-12700	Latency(ms) RTX A4000
BiLSTM-CRF [24]	4.02	148.57	87.19	51.73

has a triple keyword (text modal), a proposal (image modal) and a label, which are extracted and reconstituted from UA-DETRAC [1]. A triple keyword contains the type, color and orientation of the vehicle. The label indicates whether the proposal is consistent with the description of the triple keyword, where 1 means consistent (positive sample) and 0 means inconsistent (negative sample). Positive sample is to make the feature encodings of text and image closer while the negative sample is to make the feature encodings of text and image farther. There are 598,336 samples in Vehicle-TI, 335,040 for training, 179,520 for test and 83,776 for validation.

As Table 14 shows, we train CTIM for 50 epochs with a batch size of 64. The initial learning rate is 0.001 and CTIM is optimized by AdamW [51]. The learning rate is scheduled by the Step scheduler.

Furthermore, we set a threshold of 0.7 as the boundary of the consistency of the vehicle proposal and the triple keyword. If the output of CTIM is above 0.7, it indicates that the vehicle proposal and the triple keyword are consistent (strong-related), if not, we think they are not related or weak-related.

7.3.2 Evaluation results

Table 15 presents the evaluation results of CTIM on the test set of Vehicle-TI. CTIM has only 3.84 million parameters, which gets 97.7% accuracy (Equation 12) for the identification of consistency between vehicle images and triple keywords.

$$Accuracy = \frac{TP + TN}{TP + FP + TN + FN} \quad (12)$$

Moreover, we also test the inference speed of CTIM on different devices. CTIM spends 131.42 ms identifying one sample on an 8-core ARM v8.2 of NVIDIA Jetson AGX Xavier. When tested on an i7-12700, CTIM gets 67.43 ms latency. In addition, it costs CTIM 39.47 ms on one RTX A4000. The above proves that CTIM can maintain high performance on both edge and host devices.

7.3.3 Comparison of CTIM with other models

As the above mention, all convolution operations in CTIM are depthwise separable convolution. In addition, we use cosine distance and linear compression function to measure and process the similarity between text and image modalities. We still call it CTIM.

Table 13 F1 Scores of Different Kinds of Named Entities by BiLSTM-CRF [24] on FindVehicle

Model	F1 (<i>vehicle_type</i>)	F1 (<i>vehicle_color</i>)	F1 (<i>vehicle_orientation</i>)
BiLSTM-CRF [43]	90.56	89.79	90.17




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

Fig. 11 The data format of Vehicle-TI, which is for the training and test of CTIM

We firstly replace all depthwise separable convolution operations in CTIM with the normal convolution operations. We call it CTIM-Conv-CosineD.

Secondly, we replace the cosine distance and linear compression function in CTIM with fully connected layers, which is the operation in normal Siamese neural networks to fit the similarity by fully connected layers. We call it CTIM-DSCConv-Siamese.

Thirdly, we adopt the most well-known contrastive language and image pretraining model, CLIP [19]. We adopt ResNet-50 as the image encoder and Transformer as the text encoder. The total parameter number of CLIP is 102.58 million.

Last but not least, we fine-tune a bert-based Siamese neural network [52] to make it adapt to our task. Its architecture is Transformer-based, totally different from the aforementioned neural networks. We call it Bert-Siamese.

As Table 15 presents, CTIM performs the best for both accuracy and inference speed. In contrast, CTIM-Conv-CosineD-Linear has 36.6 million parameters, which is 32.83 million more than CTIM. Furthermore, its speed on different devices is slower than CTIM.

Secondly, the parameter number of CTIM-DSCConv-Siamese is the largest among all CNN-based neural networks, which is 99.72 million. Its inference speed on different devices is also the slowest and the accuracy is only 25.7%.

Thirdly, CLIP with ResNet-50 and Transformer has 102.58 million parameters. It gets 96.5% accuracy on the test set.

Last but not least, although Bert-Siamese [52] has a close performance with our CTIM, it has huge parameters of 189.53 million. It spends nearly 3 seconds to identify one sample on 8-core ARM v8.2, which is far too slow to deploy on edge devices.

Table 14 Implementation Details of CTIM on The Training Set of Vehicle-TI

Model	Epochs	BS	ILR	Opt	Sch ¹
CTIM	50	64	0.001	AdamW	Step

¹BS means batch size ILR means initial learning rate Opt means optimizer Sch means scheduler

Table 15 Comparison of Various Text-Image Siamese Network on Test Set of Vehicle-TI

Model	Param(M)	Latency(ms) 8-core ARM v8.2	Latency(ms) i7-12700	Latency(ms) RTX A4000	Accuracy(%)
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-DSCConv-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

Based on the performances of the above models, we find that cosine distance is a much better choice for measuring the similarity of text and image features than fully-connected layers, where the accuracy of CTIM is 72% higher than CTIM-DSCConv-Siamese. Furthermore, transformer-based encoders do not behave as expected, where CLIP and Bert-Siamese get lower accuracy than CTIM. Due to the limited features of named entities, transformer-based encoders could not play to their strengths.

7.4 Evaluation of VehicleFinder

7.4.1 Settings of evaluation

We randomly sample 2000 images from the test set of UA-DETRAC [1] as our homemade test set for VehicleFinder. For each image, we write a piece of retrieval text, which corresponds to one or more vehicles in the image. The format of the test set is presented in Fig. 12. Each item includes columns of the image path *img_path*, target id *target_id*, the upper-left abscissa of bounding box *left*, the upper-left ordinate of bounding box *top*, the width of bounding box *width*, the height of bounding box *height* and the retrieval content *retrieval_text*. There are 3917 target vehicles based on retrieval text in these 2000 images. We adopt precision, recall and F1 score to evaluate our VehicleFinder, which are presented in Equation 13, 14 and 15. We also test our VehicleFinder on three different devices.

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

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

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

7.4.2 Evaluation results

Table 16 shows that our VehicleFinder(CTIM) has 8.81 million parameters, containing the vision detector, the text detector and CTIM. After setting two thresholds th_{nm} and th_m as 0.70 and 0.30 respectively, our VehicleFinder(CTIM) achieves 87.7% precision, 89.4% recall and 88.5% F1 score. Fig. 13 presents the test results of VehicleFinder(CTIM). We can observe that the targeted vehicles could be preciously retrieved based on the description.

Furthermore, we also collect the results of the control group. VehicleFinder(CTIM-Conv-CosineD-Linear) achieves 87.4% precision, 87.9% recall and 87.6% F1 score with 41.64 million parameters. VehicleFinder(CTIM-DSCConv-Siamese) has 104.69 million parameters and its F1 score is only 12.5%, which is the lowest among all. VehicleFinder(CLIP) has 107.55 million parameters total and gets 87.5% F1 score. VehicleFinder(Bert-Siamese) has

<i>img_path</i>	<i>target_id</i>	<i>left</i>	<i>top</i>	<i>width</i>	<i>height</i>	<i>retrieval_text</i>
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.
.....

Fig. 12 The format of the homemade test set for VehicleFinder

Table 16 Evaluation of VehicleFinder on Our Homemade Test Set

Model	Param(M)	Precision(%)	Recall(%)	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-DSCConv-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

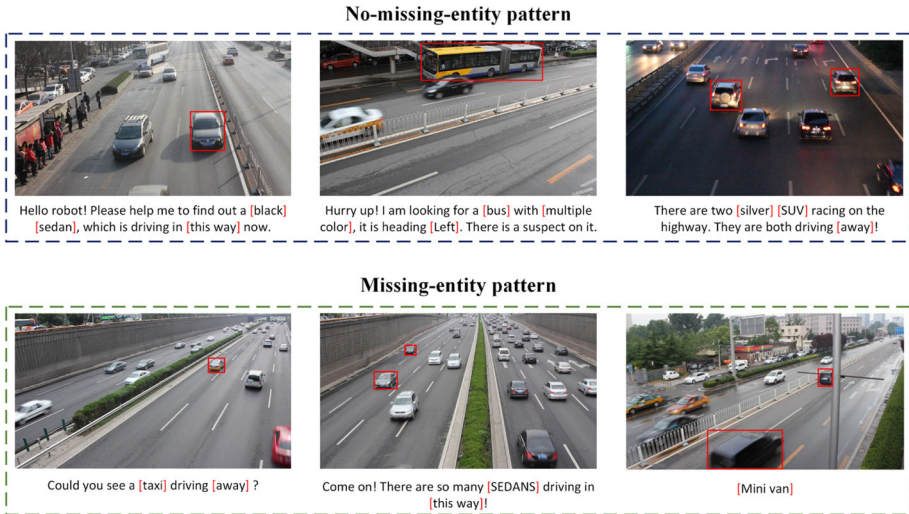


Fig. 13 Samples of inference results by VehicleFinder on UA-DETRAC [1]

the almost same F1 score (88.4%) as our VehicleFinder(CTIM), but its parameters are too huge. This further proves that Bert may not be a better choice than RNNs for encoding named entities because named entities are mainly short text with few contextual features.

We calculate the latency of our VehicleFinder(CTIM) from the moment that the command is loaded into VehicleFinder(CTIM) to the moment that the VehicleFinder(CTIM) completes the identification of one vehicle. As Equation 16 presents, T_{ner} means the time of named entity recognition and T_{cti} means the identification time of the consistency of named entities and one vehicle proposal. It includes the inference time of the text detector and CTIM. We ignore the time for the system to schedule different models. Table 17 shows the inference speed evaluation of our VehicleFinder(CTIM). The longest latency is 279.35 ms on one 8-core ARM v8.2 while the shortest latency is 93.72 ms on one RTX A4000. It implies that our VehicleFinder(CTIM) could be deployed on both edge devices and host devices, but host devices are the better choice.

Moreover, according to experiment results in Table 16. We find transformer-based CLIP and Bert achieves worse performances than our CTIM. It implies that huge transformer-based text encoders do not perform better than lightweight LSTMs on short text feature extraction, since short text has few features for extraction.

Last but not least, we also test the VehicleFinder(CTIM) on our collected images in some traffic scenes. Based on the images, we recruited two volunteers to describe the vehicles that they want to find out in the images. As Fig. 14 presents, VehicleFinder(CTIM) can still accurately find out the targeted vehicles based on the volunteers' descriptions. In addition, inference of corner cases is included as Fig. 15 shows, which means VehicleFinder(CTIM) can keep robust to some extent when confronted with some adverse phenomena.

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

Table 17 Inference Speed Evaluation of VehicleFinder

Model	Param(M)	Latency(ms) 8-core ARM v8.2	Latency(ms) i7-12700	Latency(ms) 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-DSCConv-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



Fig. 14 Inference results of VehicleFinder on our collected images

8 Conclusion and future work

We propose the first NER dataset FindVehicle in traffic domain, which contains different sentences that describe the vehicles in different traffic scenes. Named entities include several attributes of vehicles that could be detected by perception sensors. FindVehicle is a NER dataset that contains both flat and overlapped entities. All the named entities in it are annotated by both machine annotation algorithms and human annotators. Annotation includes both coarse-grained and fine-grained entity annotation. FindVehicle could be used to assist text-image cross-modal tasks in traffic scenes and act as the pretrained corpus of the territory of traffic. Furthermore, We propose an efficient text-image cross-modal vehicle retrieval system called VehicleFinder. VehicleFinder achieves 87.7% precision when identifying target vehicles by text commands, which spends 279.35 ms on one 8-core ARM v8.2 CPU and 93.72 ms on one RTX A4000 GPU. Our VehicleFinder could help traffic supervisors find out the target vehicle from a large number of images or videos based on natural language. Last but not least, we construct a text-to-image vehicle-matching dataset called Vehicle-TI.

In the future, firstly, we will continue to maintain our FindVehicle. Secondly, we will extend FindVehicle by adding the corpus of some special traffic scenes, and connecting samples of FindVehicle to images of real traffic scenes, which would be a new dataset (benchmark). Thirdly, we will explore text-video cross-modal vehicle retrieval.

9 Discussion

The discussion is divided into two parts, the challenges of FindVehicle and the limitation of our cross-modal vehicle retrieval system VehicleFinder.

In FindVehicle, long-tail data distribution, the recognition of vehicle brands out of the distribution, and the recognition of fine-grained and overlapped entities are three challenges



Fig. 15 Inference results of corner cases: occluded targets, dark environment and strong light interference

Look, there is a [blue] [suv] and a [red] [sedan] both driving [away]. I think they are [50km/h] and [65 kilometers per hour] relative to us.



Target vehicle 1: [blue, suv, away, 50km/h]

Target vehicle 2: [red, sedan, away, 65 kilometers per hour]

Fig. 16 The challenge of multiple entity clustering

worth exploring. Moreover, as Fig. 16 shows, identifying whether the extracted named entities refer to the same vehicle is a considerable challenge, equivalent to clustering named entities according to context.

The first limitation of our VehicleFinder is that a description can only contain the attributes of one vehicle. Our VehicleFinder is not adaptive to context with multiple vehicles because we only adopt NER in keyword extraction instead of combining NER with relation extraction, which is also a challenge in the future. The second limitation is that the granularity of the keywords used to describe the attributes of the vehicles is not fine enough, which is attributed to the limitation of the human cost of the annotation effort. We will continue to pay attention to and research this field in the future.

Acknowledgements The authors acknowledge XJTLU-JITRI Academy of Industrial Technology for giving valuable support to the joint project. This work is also partially supported by the Xi'an Jiaotong-Liverpool University (XJTLU) AI University Research Centre, Jiangsu (Provincial) Data Science and Cognitive Computational Engineering Research Centre at XJTLU (funding: XJTLU-REF-21-01-002). The authors sincerely acknowledge Sihao Dai, Zhou Yuan, Wenjie Zhou for their help in the project.

Code Availability All the code are available to access in the first author's GitHub repository.

Declarations

Conflicts of interest The authors declare that they have no conflict of interest.

Open Access This article is licensed under a Creative Commons Attribution 4.0 International License, which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons licence, and indicate if changes were made. The images or other third party material in this article are included in the article's Creative Commons licence, unless indicated otherwise in a credit line to the material. If material is not included in the article's Creative Commons licence and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder. To view a copy of this licence, visit <http://creativecommons.org/licenses/by/4.0/>.


References

1. Wen L, Du D, Cai Z, Lei Z, Chang M-C, Qi H, Lim J, Yang M-H, Lyu S (2020) Ua-detrac: A new benchmark and protocol for multi-object detection and tracking. *Computer Vision and Image Understanding* 193:102907
2. Hongye, L., Tian, Y., Wang, Y., Pang, L., Huang, T.: Deep relative distance learning: Tell the difference between similar vehicles. *computer vision and pattern recognition* (2016)
3. Liu, X., Liu, W., Mei, T., Ma, H.: A deep learning-based approach to progressive vehicle re-identification for urban surveillance. *European conference on computer vision* (2016)
4. Liu, X., Liu, W., Ma, H., Fu, H.: Large-scale vehicle re-identification in urban surveillance videos. *international conference on multimedia and expo* (2016)
5. Adaimi G, Kreiss S, Alahi A (2021) Deep visual re-identification with confidence. *Transportation research part C: emerging technologies* 126:103067

6. El Hamdani S, Benamar N, Younis M (2020) Pedestrian support in intelligent transportation systems: challenges, solutions and open issues. *Transportation research part C: emerging technologies* 121:102856
7. Ganin AA, Mersky AC, Jin AS, Kitsak M, Keisler JM, Linkov I (2019) Resilience in intelligent transportation systems (its). *Transportation Research Part C: Emerging Technologies* 100:318–329
8. Chien C-F, Chen H-T, Lin C-Y (2020) A low-cost on-street parking management system based on blue-tooth beacons. *Sensors* 20(16):4559
9. Sharma P, Singh A, Singh KK, Dhull A (2022) Vehicle identification using modified region based convolution network for intelligent transportation system. *Multimedia Tools and Applications* 81(24):34893–34917
10. Kong F, Zhou Y, Chen G (2020) Multimedia data fusion method based on wireless sensor network in intelligent transportation system. *Multimedia Tools and Applications* 79(47):35195–35207
11. Park, E.-J., Kim, H., Jeong, S., Kang, B., Kwon, Y.: Keyword-based vehicle retrieval. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 4220–4227 (2021)
12. Zhao, C., Chen, H., Zhang, W., Chen, J., Zhang, S., Li, Y., Li, B.: Symmetric network with spatial relationship modeling for natural language-based vehicle retrieval. (2022)
13. Bai, S., Zheng, Z., Wang, X., Lin, J., Zhang, Z., Zhou, C., Yang, H., Yang, Y.: Connecting language and vision for natural language-based vehicle retrieval. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 4034–4043 (2021)
14. Xu, B., Xiong, Y., Zhang, R., Feng, Y., Wu, H.: Natural language-based vehicle retrieval with explicit cross-modal representation learning. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 3142–3149 (2022)
15. Nguyen, T.M., Pham, Q.H., Doan, L.B., Trinh, H.V., Nguyen, V.-A., Phan, V.-H.: Contrastive learning for natural language-based vehicle retrieval. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 4245–4252 (2021)
16. Feng, Q., Ablavsky, V., Sclaroff, S.: Cityflow-nl: Tracking and retrieval of vehicles at city scale by natural language descriptions. [arXiv: Computer Vision and Pattern Recognition](https://arxiv.org/abs/2012.04481) (2021)
17. Zhang, J., Lin, X., Jiang, M., Yu, Y., Gong, C., Zhang, W., Tan, X., Li, Y., Ding, E., Li, G.: A multi-granularity retrieval system for natural language-based vehicle retrieval. (2022)
18. Deruyttere, T., Vandenhende, S., Grujicic, D., Van Gool, L., Moens, M.F.: Talk2car: Taking control of your self-driving car. In: *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pp. 2088–2098 (2019)
19. Radford, A., Kim, J.W., Hallacy, C., Ramesh, A., Goh, G., Agarwal, S., Sastry, G., Askell, A., Mishkin, P., Clark, J.: Learning transferable visual models from natural language supervision. In: *International Conference on Machine Learning*, pp. 8748–8763 (2021). PMLR
20. Rao, Y., Zhao, W., Chen, G., Tang, Y., Zhu, Z., Huang, G., Zhou, J., Lu, J.: Densclip: Language-guided dense prediction with context-aware prompting. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 18082–18091 (2022)
21. Morwal, S., Jahan, N., Chopra, D.: Named entity recognition using hidden markov model (hmm). *International Journal on Natural Language Computing (IJNLC) Vol 1* (2012)
22. Xu, Z., Qian, X., Zhang, Y., Zhou, Y.: Crf-based hybrid model for word segmentation, ner and even pos tagging. In: *Proceedings of the Sixth SIGHAN Workshop on Chinese Language Processing* (2008)
23. Gui, T., Ma, R., Zhang, Q., Zhao, L., Jiang, Y.-G., Huang, X.: Cnn-based chinese ner with lexicon rethinking. In: *Ijcai*, pp. 4982–4988 (2019)
24. Huang, Z., Xu, W., Yu, K.: Bidirectional lstm-crf models for sequence tagging. [arXiv preprint http://arxiv.org/abs/1508.01991](https://arxiv.org/abs/1508.01991) [arXiv:1508.01991](https://arxiv.org/abs/1508.01991) (2015)
25. Li, X., Yan, H., Qiu, X., Huang, X.: Flat: Chinese ner using flat-lattice transformer. [arXiv preprint http://arxiv.org/abs/2004.11795](https://arxiv.org/abs/2004.11795) [arXiv:2004.11795](https://arxiv.org/abs/2004.11795) (2020)
26. Sui, Y., Bu, F., Hu, Y., Yan, W., Zhang, L.: Trigger-gnn: A trigger-based graph neural network for nested named entity recognition. [arXiv preprint http://arxiv.org/abs/2204.05518](https://arxiv.org/abs/2204.05518) [arXiv:2204.05518](https://arxiv.org/abs/2204.05518) (2022)
27. Sang, E.F., De Meulder, F.: Introduction to the conll-2003 shared task: Language-independent named entity recognition. [arXiv preprint cs/0306050](https://arxiv.org/abs/cs/0306050) (2003)
28. Balasuriya, D., Ringland, N., Nothman, J., Murphy, T., Curran, J.R.: Named entity recognition in wikipedia. In: *Proceedings of the 2009 Workshop on the People’s Meets NLP: Collaboratively Constructed Semantic Resources (People’s Web)*, pp. 10–18 (2009)
29. Derczynski, L., Nichols, E., van Erp, M., Limsopatham, N.: Results of the wnut2017 shared task on novel and emerging entity recognition. In: *Proceedings of the 3rd Workshop on Noisy User-generated Text*, pp. 140–147 (2017)

30. Weischedel, R., Palmer, M., Marcus, M., Hovy, E., Pradhan, S., Ramshaw, L., Xue, N., Taylor, A., Kaufman, J., Franchini, M., et al.: Ontonotes release 5.0 Idc2013t19. Linguistic Data Consortium, Philadelphia, PA **23** (2013)
31. Ding, N., Xu, G., Chen, Y., Wang, X., Han, X., Xie, P., Zheng, H., Liu, Z.: Few-nerd: A few-shot named entity recognition dataset. In: Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers), pp. 3198–3213 (2021)
32. Li, J., Fei, H., Liu, J., Wu, S., Zhang, M., Teng, C., Ji, D., Li, F.: Unified named entity recognition as word-word relation classification. arXiv preprint <http://arxiv.org/abs/2112.10070> arXiv:2112.10070 (2021)
33. Scribano, C., Sapienza, D., Franchini, G., Verucchi, M., Bertogna, M.: All you can embed: Natural language based vehicle retrieval with spatio-temporal transformers. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 4253–4262 (2021)
34. Khorramshahi, P., Rambhatla, S.S., Chellappa, R.: Towards accurate visual and natural language-based vehicle retrieval systems. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 4183–4192 (2021)
35. Sun, Z., Liu, X., Bi, X., Nie, X., Yin, Y.: Dun: Dual-path temporal matching network for natural language-based vehicle retrieval. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 4061–4067 (2021)
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.: Tracked-vehicle retrieval by natural language descriptions with domain adaptive knowledge. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 3300–3309 (2022)
37. TT PHUNG, T., Q. LY, N., T. VO, T., TN HO, M.: Deep feature learning network for vehicle retrieval. In: 2021 The 5th International Conference on Machine Learning and Soft Computing, pp. 18–21 (2021)
38. Devlin, J., Chang, M.-W., Lee, K., Toutanova, K.: Bert: Pre-training of deep bidirectional transformers for language understanding. arXiv preprint <http://arxiv.org/abs/1810.04805> arXiv:1810.04805 (2018)
39. Floridi L, Chiriatti M (2020) Gpt-3: Its nature, scope, limits, and consequences. *Minds and Machines* 30(4):681–694
40. Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., Fei-Fei, L.: Imagenet: A large-scale hierarchical image database. In: 2009 IEEE Conference on Computer Vision and Pattern Recognition, pp. 248–255 (2009). IEEE
41. Goel, S., Bansal, H., Bhatia, S., Rossi, R.A., Vinay, V., Grover, A.: Cyclicp: Cyclic contrastive language-image pretraining. arXiv preprint <http://arxiv.org/abs/2205.14459> arXiv:2205.14459 (2022)
42. Stubbs A, Uzuner Ö (2015) Annotating longitudinal clinical narratives for de-identification: The 2014 i2b2/uthealth corpus. *Journal of biomedical informatics* 58:20–29
43. RangilYu: NanoDet-Plus: Super fast and high accuracy lightweight anchor-free object detection model. <https://github.com/RangilYu/nanodet> (2021)
44. Chollet, F.: Xception: Deep learning with depthwise separable convolutions. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 1251–1258 (2017)
45. Agarap, A.F.: Deep learning using rectified linear units (relu). arXiv preprint <http://arxiv.org/abs/1803.08375> arXiv:1803.08375 (2018)
46. Bojanowski P, Grave E, Joulin A, Mikolov T (2017) Enriching word vectors with subword information. *Transactions of the association for computational linguistics* 5:135–146
47. Mikolov, T., Chen, K., Corrado, G., Dean, J.: Efficient estimation of word representations in vector space. arXiv preprint <http://arxiv.org/abs/1301.3781> arXiv:1301.3781 (2013)
48. Pennington, J., Socher, R., Manning, C.D.: Glove: Global vectors for word representation. In: Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), pp. 1532–1543 (2014)
49. Souza, F., Nogueira, R., Lotufo, R.: Portuguese named entity recognition using bert-crf. arXiv preprint <http://arxiv.org/abs/1909.10649> arXiv:1909.10649 (2019)
50. Schweter, S., Akbik, A.: Flert: Document-level features for named entity recognition. arXiv preprint <http://arxiv.org/abs/2011.06993> arXiv:2011.06993 (2020)
51. Loshchilov, I., Hutter, F.: Decoupled weight decay regularization. arXiv preprint <http://arxiv.org/abs/1711.05101> arXiv:1711.05101 (2017)
52. VILCEK, A., MOTTAGHINEJAD, S., SHI, S., GUPTA, K., PASUMARTY, S., PANG, L., MEHROTRA, P.: Transformer-based deep siamese network for at-scale product matching and one-shot hierarchy classification (2018)

Authors and Affiliations

Runwei Guan^{1,2,3,4} · Ka Lok Man² · Feifan Chen² · Shanliang Yao^{1,2,3,4} ·
Rongsheng Hu⁵ · Xiaohui Zhu² · Jeremy Smith¹ · Eng Gee Lim² ·
Yutao Yue^{3,4,6} 

Runwei Guan
Runwei.Guan@liverpool.ac.uk

Ka Lok Man
Ka.Man@xjtlu.edu.cn

Feifan Chen
sgfchen5@liverpool.ac.uk

Shanliang Yao
shanliang.yao@liverpool.ac.uk

Rongsheng Hu
1033170432@stu.jiangnan.edu.cn

Xiaohui Zhu
Xiaohui.Zhu@xjtlu.edu.cn

Jeremy Smith
J.S.Smith@liverpool.ac.uk

Eng Gee Lim
enggee.lim@xjtlu.edu.cn

- ¹ Department of Electrical Engineering and Electronics, University of Liverpool, Liverpool, United Kingdom
- ² School of Advanced Technology, Xi'an Jiaotong-Liverpool University, Suzhou 215123, China
- ³ XJTLU-JITRI Academy of Technology, Xi'an Jiaotong-Liverpool University, Suzhou 215123, China
- ⁴ Institute of Deep Perception Technology, JITRI, Wuxi 214000, China
- ⁵ Faculty of Information Engineering and Automation, Kunming University of Science and Technology, Kunming 650500, China
- ⁶ Department of Mathematical Sciences, University of Liverpool, Liverpool L69 7ZX, United Kingdom