2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)
October 14-18, 2024. Abu Dhabi, UAE

Mind the Error! Detection and Localization of Instruction Errors in Vision-and-Language Navigation

Francesco Taioli[1,4], Stefano Rosa[2], Alberto Castellini[1], Lorenzo Natale[2], Alessio Del Bue[2], Alessandro Farinelli[1], Marco Cristani[1], Yiming Wang[2,3]

https://intelligolabs.github.io/R2RIE-CE/

***Abstract*— Vision-and-Language Navigation in Continuous Environments (VLN-CE) is one of the most intuitive yet challenging embodied AI tasks. Agents are tasked to navigate towards a target goal by executing a set of low-level actions, following a series of natural language instructions. All VLN-CE methods in the literature assume that language instructions are exact. However, in practice, instructions given by humans can contain errors when describing a spatial environment due to inaccurate memory or confusion. Current VLN-CE benchmarks do not address this scenario, making the state-of-the-art methods in VLN-CE fragile in the presence of erroneous instructions from human users. For the first time, we propose a novel benchmark dataset that introduces various types of instruction errors considering potential human causes. This benchmark provides valuable insight into the robustness of VLN systems in continuous environments. We observe a noticeable performance drop (up to -25%) in Success Rate when evaluating the state-of-the-art VLN-CE methods on our benchmark. Moreover, we formally define the task of Instruction Error Detection and Localization, and establish an evaluation protocol on top of our benchmark dataset. We also propose an effective method, based on a cross-modal transformer architecture, that achieves the best performance in error detection and localization, compared to baselines. Surprisingly, our proposed method has revealed errors in the validation set of the two commonly used datasets for VLN-CE, *i.e.,* R2R-CE and RxR-CE, demonstrating the utility of our technique in other tasks.**

I. Introduction

Interacting with agents through natural language is a long-term goal of embodied AI as it is potentially the most intuitive mode for human-robot communication. The emerging research on Vision-and-Language Navigation (VLN) [1], [2] is along this path, aiming to develop embodied agents that, following a given instruction in the format of natural language, can reach a target destination in a 3D environment, *e.g.,*

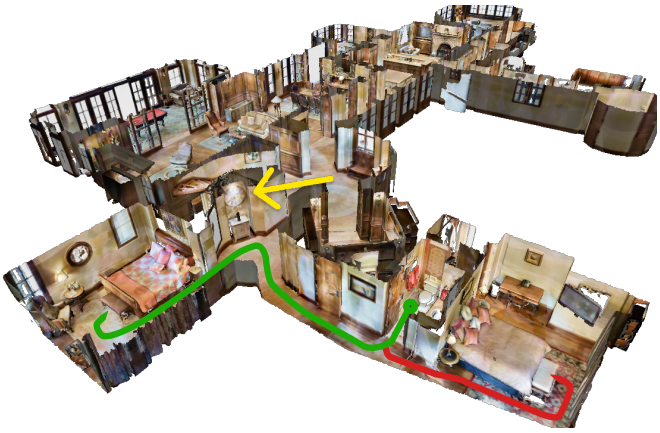

Fig. 1: An agent navigates in a scene, following instructions expressed in natural language, for example *"Exit the bathroom and go left (✓ right), then turn left at the big clock and go into the bedroom and wait next to the bed."* By just changing "right" to "left" in the instruction, the agent terminates the exploration in the wrong location, ignoring the fact that along the path it did not see the "big clock" (yellow arrow).

"Exit the bedroom and turn left. Walk straight past the grey couch and stop near the rug." VLN is a challenging task. First, it requires a *strong* vision-language alignment, *i.e.,* the alignment between the visual information (RGB-D inputs) and the natural language instruction. Second, VLN agents need to reason about which part of the instruction has been executed, and which parts need to be carried out. Lastly, agents must be able to generalize from seen to unseen environments, as well as from simulation to real-world [3].

To facilitate the study of VLN, many benchmark datasets have been proposed. For instance, Room-to-Room (R2R) [1] is the first benchmark dataset for VLN, which is built on top of Matterport3D [4] scenes and it is equipped with discrete navigation graphs (*i.e.,* discrete environments) [1]. Follow-up works have further extended the R2R dataset from different aspects, *e.g.,* enabling VLN in the continuous environments, a more practical scenario, where agents can navigate to any unobstructed point via a set of low-level actions (R2R-CE) [5], levelling up the scale of the dataset [6], or providing multilingual fine-grained visual groundings that relate each word to pixels/surfaces in the environment [7]. All the previous benchmarks, however, only consider *correct* language instructions. This consideration can be brittle in

[1] University of Verona, Verona, Italy.
[2] Istituto Italiano di Tecnologia (IIT), Genova, Italy.
[3] Fondazione Bruno Kessler, Trento, Italy.
[4] Polytechnic of Turin, Turin, Italy.
francesco.taioli@polito.it

We acknowledge the CINECA award under the ISCRA initiative, for the availability of high-performance computing resources and support. This work was partially sponsored by the PNRR project FAIR - Future AI Research (PE00000013), under the NRRP MUR program funded by the NextGenerationEU. This project received funding from the European Union's Horizon Research and Innovation Programme G.A. n. 101070227. This study was also carried out within the PNRR research activities of the consortium iNEST (Interconnected North-Est Innovation Ecosystem) funded by the European Union Next-GenerationEU (Piano Nazionale di Ripresa e Resilienza (PNRR) – Missione 4 Componente 2, Investimento 1.5 – D.D. 1058 23/06/2022, ECS_00000043)

reality as human often gives instructions that are approximate or ambiguous, or even prone to error as based on their memory. Moreover, in some cases, subjects with cognitive or perceptual impairments may provide inaccurate descriptions when naming a task [8]. As shown in Fig. 1, simply changing the word *"right"* to *"left"* can cause the agent to terminate the navigation in a wrong location, even if the central part of the instruction still contains informative details to discern that the sentence has some mistakes. Our preliminary studies, summarized in Fig. 2, show that state-of-the-art VLN methods[1] fed with *(i)* correct instructions (in green) and *(ii)* instructions with up to 3 errors (in red) have a noticeable gap (up to -25%) in terms of Success Rate. This analysis shows that it is critical to understand the presence of errors in the instruction and to locate them.

In this work, we first formally define the types of errors that may occur in language instructions for the VLN task in indoor environments, including *Direction, Room, Object, Room&Object* and a combination of *All* types of errors. Based on these definitions, we propose a novel benchmark in continuous environments built on top of the R2R-CE dataset [5], in which we artificially inject errors of the different types to thoroughly evaluate the capability of VLN methods to deal with these mistakes. Moreover, we propose a novel task, *i.e., Detection and Localization of Instruction Errors*, which aims to detect and localize errors within a given instruction, an important intermediate task for further addressing instruction errors in VLN policy learning. Then, we propose a method based on a cross-modal transformer, fusing together the language features of the instruction with the observations of the agent, achieving competitive performance in solving the Detection and Localization of Instruction Errors, compared to a CLIP Alignment baseline. Notably, our approach could be useful in spotting annotation errors in existing benchmarks: we discovered 8 episodes in the R2R-CE and 10 episodes in the RxR-CE dataset that contain either incorrect or ambiguous instructions, which should not be considered during evaluation. Our contributions are summarized below:

- We categorize the errors that exist in language instructions in the VLN-CE task, and establish the first benchmark (R2RIE-CE) for VLN in continuous environments with instruction errors.
- We experimentally show that state-of-the-art VLN methods are not robust to instruction errors using our proposed benchmark, necessitating the study of instruction errors in VLN.
- We formalize the novel task of *Detection and Localization of Instruction Errors* for VLN agents, and propose an effective method, Instruction Error Detector & Localizer (*IEDL*), based on a novel Instruction-Trajectory compatibility model. Our method effectively connects the semantic meaning of the language instruction with the sequence of visual observations of the agent, establishing a competitive baseline.
- We discover errors in the ground-truth annotations of the R2R-CE dataset, and episodes that should be removed from the validation split of RxR-CE dataset, demonstrating an additional use of our technique.

[1]Leaderboard at https://eval.ai/web/challenges/challenge-page/719/leaderboard/1966/success. Only models with public weights are evaluated, on the latest version of the R2R-CE dataset (version 1.3)

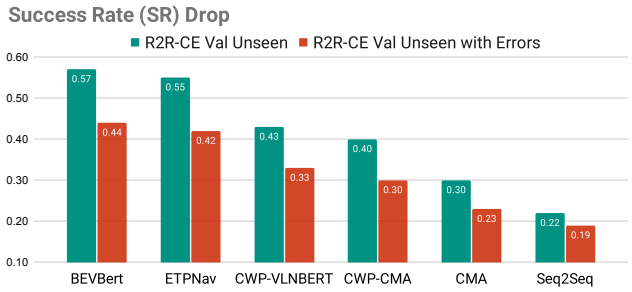

Fig. 2: Comparison of the Success Rate (SR) of different methods (in order, [9], [10], [11], [11], [5], [5]) working on continuous environments[1]. We show the SR on the standard R2R-CE dataset split Val Unseen (green) and the drop in SR performance when errors are present (red). Interestingly, we see up to -25% drop in SR when up to three errors among {*Direction, Room, Object*} per episode are present.

II. Related Works

We discuss recent methods on VLN and failure analysis on VLN methods. We also cover the relevant study on instruction and trajectory alignment.

Vision-and-Language Navigation. The VLN task [1] has drawn increasing attention since its introduction. The seminar work [1] presents the Room-to-Room (R2R) navigation task and the Matterport3D Simulator based on the Matterport3D indoor environment [4] with real 360-degree RGB-D scans. Importantly, methods based on the R2R navigation task define their action space based on an undirected graph that is constructed from panoramic viewpoints (*i.e.,* nav-graph), forming a discrete environment. Early VLN agents operating under this paradigm capture historical observation by means of recurrent unit [1], [12], [13] or by using attention mechanism [14], [15], [16]. Instead, in [5], VLN in Continuous Environment (VLN-CE) is introduced, in which agents are allowed to move freely, thus removing the assumption of known environment topologies, short-range oracle navigation and perfect agent localization. In doing that, they translated the nav-graph R2R trajectories to the continuous environments in the Habitat simulator [3]. The authors find significantly lower absolute performance in the continuous settings. To bridge the discrete-continuous gap, [11] propose a predictor to generate a set of candidate waypoints during navigation. Given the long time horizon of an episode under this setting, recent methods proposed metric maps [17] or topology memory [10], [9] to represent the history of observations. The current state-of-the-art method, BEVBert [9], proposed a hybrid map to balance the demand for both short-term reasoning and long-term planning while

introducing a pre-training map paradigm. Differently from the above-mentioned VLN works, in which the focus is to reach the target goal in the most efficient way, we aim at introducing and addressing a novel task, namely the *Detection and Localization of Instruction Errors* within the VLN context.

Failure Analysis on VLN. Several works have investigated the behaviour and failure of VLN agents. The study in [18] shows that the agents refer to both object and direction tokens when making their navigation decisions. A similar work [19] further evaluates to which degree the spatial and directional language cues inform the navigation decisions. They focus on the path-ranking models and replace directions/nouns/objects and numbers tokens with the `[MASK]` token. The study in [20] investigates the significant drop in performance when tested on unseen environments, locating the environment bias to be in the low-level visual appearance. Moreover, Yang *et al.* [21] developed a methodology to study agent behaviour on a skill-specific basis. Their method is based on generating skill-specific interventions and measuring changes in action predictions, considering the stop behaviour, unconditional (directional) and conditional (object- and room-seeking) skills.

Our work differs from the previous studies, as we investigate the instruction errors and their impact for navigation policies in the VLN task. We manipulate among a large set of tokens to inject instruction errors, instead of just masking certain tokens out as in [19]. We do not focus on path-ranking models [20], the domain gap towards unseen environments [20], and skill-specific interventions [21].

Instruction and Trajectory Alignment. Instruction and Trajectory Alignment is the ability to align semantically the concepts represented by these two modalities, an important aspect for obtaining a VLN policy. Huang *et al.* [22] introduces the Cross-Modal Alignment (CMA) task, *i.e.,* discriminating instruction-trajectory path pairs from negative pairs, by maintaining the original instruction and altering the path sequence. Zhao *et al.* [23] perform a study on the quality of the instruction generated by VLN instruction generation models [13], [24] using human wayfinders. Moreover, they propose an Instruction-Trajectory compatibility model to classify high and low-quality instructions for trajectories from the R2R `Val Unseen` and `Val Seen` sets with the discrete VLN setting. These instructions are considered to be high quality if 2 out of 3 human wayfinders reached the goal. Differently from [22], they also consider instruction perturbations. In [25] the authors proposed a Contrastive Instruction-Trajectory Learning framework to boost the generalizability of the navigation policy. Their method selects sub-optimal trajectories by the hop distance from an anchor trajectory. Positive instructions are generated by substituting words with their synonym, inserting or substituting them with context or by back-translation; instead, negative ones are generated by splitting the original instruction into sub-instructions and shuffling or repeating randomly to become an intra-negative instruction. The authors assume that the augmented instructions should preserve the semantic information of the original ones.

While the above-discussed works are relevant, their objective is to obtain a better Instruction and Trajectory Alignment for the VLN policy learning. Instead, our work investigates how instruction errors impact the VLN policy, and addresses the novel task of detection and localization of instruction errors for a given policy. Moreover, our study is based on VLN-CE, while the Instruction and Trajectory Alignment is mostly based on the discrete VLN operating on top of the nav-graph.

III. Task Definition

In this section, we first introduce VLN in Continuous Environments (VLN-CE); then, we define the types of instruction errors and finally formalize our novel task of *Detection and Localization of Instruction Errors*.

VLN-CE. For each episode, the agent is given an instruction $\mathcal{I}$ composed of F words, *i.e.,* $\mathcal{I} = \{w_1, ..., w_F\}$. At each time step t, the agent receives a visual observation O_t, *i.e.,* an RGB-Depth image, collected during the navigation. We define $\mathcal{O} = \{O_1, ..., O_T\}$ as the set of visual observations over a navigation episode, where T is the total number of steps executed into the environment by a policy π. The goal of VLN-CE is to learn a policy π for reaching a target goal, which maps the instruction $\mathcal{I}$ and observation O_t to a set of low-level actions $a_t \in \{$`Forward 0.25m, Turn Left 15°, Turn Right 15°, Stop` $\}$. Note that in this work we assume the policy π to be given, *e.g.,* the state-of-the-art policy [9], since we are interested in solving the task of Detection and Localization of Instruction Errors, not the VLN-CE task.

Types of Instruction Error. Humans are prone to errors when providing directions to a destination. For instance, one might say, *"Go down the hallway and turn right (✓left) when you see the plant near the bathroom (✓bedroom)"*. We categorize the common instruction errors based on their semantics, to facilitate a systematic assessment of their impact on VLN-CE. Furthermore, for a better practical significance, we focus on instruction errors that are likely to occur in the real world, considering common human mistakes due to either confusion or inaccurate memory of a scene structure. Specifically, we identify three types of individual errors, detailed below:

(i) Direction Error: words indicating directions, such as *left* or *backward*, are important ingredients in human instructions for navigation tasks. Each direction has its natural antonym counterpart, such as *left/right*, or *forward/backward*, which can contribute to common instruction errors due to human confusion. If an agent is wrongly given an instruction with the opposite direction, *e.g.,* left (✓right), its path can deviate heavily from the correct one, as illustrated in Fig. 1. The occurrence of a *direction error* is defined if at least one ground-truth direction is replaced by a different direction word in an instruction. We mainly focus on the more frequent *direction error*, in which the antonym direction replaces the ground-truth direction.

(ii) Object Error: words indicating objects that are observed along the navigation also play an important role in guiding agents' motion. However, humans may not remember the

exact scene including the object's disposition. Humans could likely confuse objects in the instruction, especially among those that are often co-located in a common space. For example, an error such as sofa (✓chair) is more likely to happen than toilet (✓chair). We define an *object error* as an error that occurs if at least one object class in the instruction is wrongly indicated to a different type. In particular, we focus on *object errors* that account for such *common sense* priors, *i.e.,* object co-occurrence, in a common space.

(iii) Room Error: words indicating rooms are fundamental to guide the agent towards its destination since they provide context along the navigation. As with *object errors*, humans may confuse one room with another when providing instructions, or they may not recall the exact layout of the scene. It is more likely humans confuse rooms among those that are more geometrically adjacent to each other. For example, as the bedroom is often next to the bathroom, the instruction likely contains the error as: *"Go into the door. Once inside, go into the bathroom (✓bedroom). Stop in front of the closet, near the bed."*. We can define the *room error* as the occurrence of at least one room in the instruction is wrongly indicated to another room type. In particular, we focus on *room error* that accounts for the room adjacency priors.

(iv) Room&Object Error: as both *object errors* and *room errors* are attributed to the inaccurate human memory regarding a scene, we also consider the *Room&Object errors* as cases in which both types of errors co-occur in a given instruction. For example, *"Exit the room and turn right. Continue forward until you arrive at the living room (✓kitchen). Go around the table and stop in front of the lamp (✓sink), near the stove."*

(v) All Error: finally, all individual types of errors can occur in the same instruction due to inaccurate scene-memory and direction confusion. An *all error* exists if there is at least a *direction error* and a *room&object error*. For example, *"Exit the bathroom and go left (✓right), then turn left at the big clock and go into the bathroom (✓bedroom) and wait next to the closet (✓bed) in front of the armchair."*.

Instruction Error Detection and Localization. Given an instruction $\mathcal{I}$, which describes in natural language how to reach a target position, and a set of observation $\mathcal{O}$ collected by a VLN agent controlled by a given policy π, the task of *Instruction Error Detection* aims to learn a function $d_\pi : \mathcal{I} \times \mathcal{O} \rightarrow \{\text{True}, \text{False}\}$, that returns True if the instruction contains errors, False otherwise. If the detection function d_π returns True, the task of *Instruction Error Localization* can also be performed. It aims to identify the positions of the error occurrences within the instruction. Namely, positions of wrong words in the instruction are computed. Formally, the localization function can be defined as $l_\pi : \mathcal{I} \times \mathcal{O} \rightarrow \mathcal{P}(\{0, 1, \ldots, len(\mathcal{I}) - 1\})$, where $len(\mathcal{I})$ is the total number of words in the instruction $\mathcal{I}$, and $\mathcal{P}(\cdot)$ is the power set operator.

IV. Benchmark

We establish the first benchmark dataset **R2RIE-CE**, short for *R2R with Instruction Errors in Continuous Environments*, for assessing VLN-CE methods with erroneous instructions and for the novel task of Detection and Localization of Instruction Errors. Our benchmark dataset is built on top of the R2R-CE [5] dataset, which serves for evaluating VLN-CE methods. The R2R-CE dataset is split into `Train`, `Val Seen` and `Val Unseen`. We will use `Train` set for training data and focus on the most challenging `Val Unseen` for validation, which has 1839 episodes with new paths, instructions, and scenes that are not observed during training.

In order to create erroneous instructions, we artificially inject various types of Instruction Errors (as defined in Sec. III) into the given natural language instructions in R2R-CE. Our error composition carefully considers error priors induced by human causes, including inaccurate scene memory and direction confusion. For each type of error, *i.e., Direction*, *Object*, *Room*, *Room&Object* and *All*, we create a corresponding validation set based on the `Val Unseen` validation split of R2R-CE. The original validation split can contain episodes that may not be appropriate for our evaluation. The natural language instruction should not be too short and the episodes should contain words relevant to the specific error setup. For example, episodes must have Direction words for the validation set with *Direction Error*, or episodes must contain Direction, Object, and Room words for the validation set with *All Errors*. Specifically, prior to constructing the validation set with each type of error, we first exclude episodes with a minimum length of less than τ words, and episodes lacking words relevant to the specific experiments, such as Direction error words for the *Direction Error* dataset. Following this filtering stage, we obtain a set of correct episodes $\mathcal{E}_\mathcal{C}$ for each type of Instruction Error.

Then, for each episode $e_i \in \mathcal{E}_\mathcal{C}$, we create a corresponding erroneous episode in which we perturbed the instruction with the respective error. Specifically, *(i)* for the *Direction Error*, we consider the following set of directional words that occur frequently in the instructions: *left/right*, *go down/go up*, *into/out of*, *forward/backward*, *inside/outside*, *go around/go back*, *leftmost/rightmost*. We introduce the *Direction Error* by swapping the ground-truth direction with its antonym; *(ii)* For the *Object Error*, we first identify a set of common object categories $\mathcal{C}$ that frequently appear in a set of language instructions, excluding synonyms. Each object class $c_i \in \mathcal{C}$ is associated with a set of object classes $\mathcal{C}_i$ comprised of the classes that often co-locate in the same room. We introduce an *Object Error* by swapping the ground-truth object class c_i to a random class $c_j \in \mathcal{C}_i$; *(iii)* For the *Room Error*, we consider the following set of rooms $\mathcal{R}$ that are common in indoor environments: *kitchen, archway, bathroom, bedroom, gym, lounge, hallway, living room, office, dining room, laundry, restroom*. Each room $r_i \in \mathcal{R}$ is associated with a set of rooms $\mathcal{R}_i$ that are often adjacent to r_i[2]. We introduce a *Room Error* by swapping the ground-truth room r_i to a random room $r_j \in \mathcal{R}_i$; *(iv)* For the *Room&Object Error*, we introduce in an instruction both *Room* and *Object Error*; *(v)* For the

[2]The list is obtained by leveraging ConceptNet relations and manual curation

All Error, we introduce in an instruction both *Direction* and *Room&Object Error*. The perturbed episodes will then be stored in the corresponding set of perturbed episodes, $\mathcal{E}_{\mathcal{P}}$. For each perturbed episode $e_i \in \mathcal{E}_{\mathcal{P}}$, both the error type and the position of the perturbed word are stored as metadata for the assessment of Detection and Localization of Instruction Errors. Eventually, for each type of instruction error, we obtain sets $\mathcal{E}_{\mathcal{C}}$ and $\mathcal{E}_{\mathcal{P}}$. Tab.I presents the statistics of the created validation set with each type of error.

TABLE I: Statistics of the dataset used in this benchmark.

Error type	# Episodes	Errors per episode	Mean instr. length (tokens)
Direction	3218	1	31.59
Room	2064	1	32.05
Object	3162	1	30.94
Room & Object	1734	2	33.35
All	1586	3	34.58

V. Method

We instantiate our Instruction Error Detector & Localizer (*IEDL*) on BEVBert [9], currently the state-of-the-art method for VLN-CE[1], but our method is *model-agnositic*. BEVBert is composed of a hybrid map to balance the demand of VLN for both short-term reasoning and long-term planning: *(i)* a local metric map is used to explicitly aggregate incomplete observations and remove duplicates; *(ii)* a *graph-based* topological map equipped with a global action space is used for long-term planning, and *(iii)* a pre-training framework, based on the hybrid map, is used to learn multimodal map representation which enhances spatial-aware cross-modal reasoning. In the following, we will refer to policy π as an optimal policy trained using the BEVBert model. Note that during the training process of our method, policy π is frozen, thus it will not receive any parameter updates.

Model structure. The full pipeline of *IEDL* is shown in Fig. 3. We define the instruction embeddings as $\Upsilon \in \mathbb{R}^{W \times D}$, namely the textual instruction $\mathcal{I}$ undergoes a tokenization and padding procedure, up to $W = 80$ tokens [9], [10], with associated text BERT [26] embeddings. Moreover, we refer to D as the dimensionality of the latent space. We consider the trajectory set of image embeddings $\Gamma = \{V_1, ..., V_T\}$ (where V_t is the corresponding image embedding of visual observation O_t defined in Section III) as the set containing the viewpoint panorama visual features $V_t \in \mathbb{R}^D$ (*i.e.,* embeddings) of the node chosen at each time step t by policy π, following instruction embeddings Υ. As in [9], the embedding V_t contains features extracted from a pre-trained Vision Transformer (ViT-B/16-CLIP [27]), which are then fed into a panoramic encoder [14] to obtain contextual view embeddings. For each episode i, we first collect the set of observation embeddings (*i.e.,* trajectory) $\Gamma \in \mathbb{R}^{T \times D}$ from policy π. To retain positional information, we use sine and cosine positional encoding. Similar to BERT [26] `[CLS]` token, we prepend a learnable `[CLS]` $\in \mathbb{R}^D$ embedding to set Γ, which will be later used for our downstream tasks. Then, trajectory set Γ and instruction embeddings Υ are fused together by feeding them to a cross-modal multi-layer transformer with k layers (see right part of Fig.3). We adopted an architecture similar to [28] but without the bi-directional cross-attention layer since our focus is not on having more than one cross-modal aligned output. For each layer of the transformer, we first perform cross-attention using trajectory set Γ as a query (Q) and instruction embeddings Υ as a context (KV). Then, we perform self-attention followed by a feed-forward layer. The `[CLS]` token (enriched with visual-language fused features after the cross-modal transformer) is then fed to two classification heads that produce the output of our Instruction Error Detector & Localizer: *(i)* the trajectory-instruction matching head $f_d : \mathbb{R}^D \rightarrow \mathbb{R}$ (which implements function d_π presented in Sec. III) predicts the alignment score $\sigma(a) \in [0, 1]$, where $\sigma(\cdot)$ is the sigmoid operator; *(ii)* the error localization head $f_l : \mathbb{R}^D \rightarrow \mathbb{R}^W$ (which implements function l_π presented in Sec. III) predicts the token errors' localization within the instruction. Both heads are implemented by a Multilayer Perceptron (MLP), a ReLU activation function, followed by a Layernorm layer and a final MLP.

Training. We train the model with two separate losses. Specifically, $\mathcal{L}_d$ for the trajectory-instruction matching f_d head is trained using Binary Cross-Entropy loss. Instead, $\mathcal{L}_l$ for the error localization f_l head is trained using standard Cross-Entropy Loss. Since multiple errors can be present within an instruction, we sum the loss for each localization term. Thus, we minimize the following loss $\mathcal{L}$:

$$\mathcal{L} = \lambda_1 \mathcal{L}_d + \frac{\lambda_2}{E} \sum_{i=1}^{E} \mathcal{L}_l$$

where E is the number of errors actually present in the instruction, λ_1 and λ_2 are the hyperparameters for the weights of the two losses, respectively.

VI. Experiments

We first benchmark a state-of-the-art policy [9] on our dataset, to show how different types of error affect its performance. We then evaluate the proposed *IEDL* model on the *Detection and Localization of Instruction Errors* task in comparison to two baseline methods. Finally, we show *IEDL* can identify episodes with annotation errors in existing dataset. In the following, we introduce the performance metrics and the compared baselines.

Performance Metrics. We use standard metrics for evaluating VLN performance as in prior works [1], [29]. An episode in the VLN task is considered successful if the distance between the final position of the agent and the target location is less than 3 meters. The main VLN metrics are Success Rate (`SR`), and Success rate weighted by (normalized inverse) Path Length (`SPL`). Moreover, to assess the performance of the task of *Detection and Localization of Instruction Errors*, we adopt the Area Under the ROC Curve (`AUC`) as in [23], *i.e.,* the area beneath the True Positive Rate plotted against the False Positive Rate. We consider an instruction to be positive if it contains at least one error. In this work, we consider `AUC` as the main metric. To measure the performance regarding the task of *Instruction Error Localization*, we introduce a novel measure termed as *Absolute Token Distance* (`ATD`).

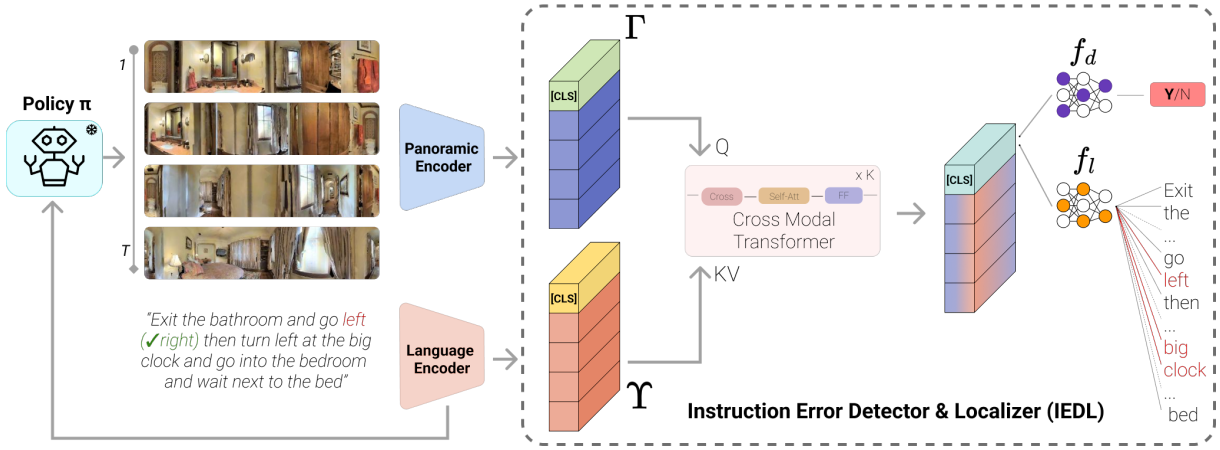

Fig. 3: Architecture of our proposed *IEDL* model, representing the scenario depicted in Fig 1. The frozen policy π follows Instruction Υ, producing a sequence of observation $\mathcal{O}$. Then, a panoramic encoder and a language encoder produce, respectively, the trajectory visual features Γ and instruction features Υ. We then feed the trajectory set Γ and Υ to a cross-modal multi-layer transformer to produce visual-language aligned features. Finally, two specialized heads perform *Instruction Error Detection* and *Instruction Error Localization*, respectively.

The ATD metric is defined as the absolute difference between the predicted position of the perturbed token and the true position of the perturbed token. Formally, for an episode i, let ℓ^i_j be the j^{th} ground-truth index of the token that has been perturbed to form an error, and let $\hat{\ell}^i_j$ be the predicted index of the token in the instruction containing the error, we define ATD as:

$$ATD = \frac{1}{N_{\mathcal{E}_P}} \sum_{i=1}^{N_{\mathcal{E}_P}} \frac{1}{J_i} \sum_{j=1}^{J_i} |\ell^i_j - \hat{\ell}^i_j|, \tag{1}$$

where $N_{\mathcal{E}_P}$ is the number of perturbed episodes $\mathcal{E}_P$, and J_i is the number of perturbed tokens in the episode i.

Implementation details. For details on BEVBert, we refer the reader to [9]. We set $k = 4$ as the number of transformer layers for our multi-layer cross-modal transformer and $D = 768$ for all the feature dimensions. In our experimental trials, we set both λ_1 and λ_2 to 1 empirically for our loss function $\mathcal{L}$. We train *IEDL* model for up to 9k iterations, using the AdamW optimizer. Specifically, for all experiments described below, we train our *IEDL* method using the *All Error* type benchmark *without* common sense from the *training set* of R2R-CE, thus we do not enforce the co-location of objects from object set $\mathcal{C}$ and adjacency for room set $\mathcal{R}$. This ensures that additional common sense bias is not injected during training. We then select the model that achieves the best AUC score, using the *All* error type benchmark *with* common sense from the validation splits Val Unseen. The selected model is used for evaluation across the remaining benchmarks.

Baselines. As no baseline for this task is available, we compare *IEDL* method with a random baseline and a simple zero-shot alignment baseline:

(i) Random. Each instruction embeddings Υ_i for episode i is randomly classified as correct or wrong. We then predict J_i random token indices, of which each $\hat{\ell}^i_j \in [0, len(\Upsilon_i)]$, where J_i is the number of errors expected in the dataset (see Tab. I).

(ii) CLIP Alignment. For each episode i, the set of room and object tokens $\mathcal{K}$ is extracted from instruction $\mathcal{I}$ via an off-the-shelf POS tagger [30]. Then, for each observation O_t at time step t, the top-k predicted room and object labels are extracted using CLIP [27], forming the set of objects and rooms by observation $\mathcal{S}$. The text prompts that are used for finding rooms and objects in the images are composed using the room and object list, as mentioned in Sec. III. Each CLIP prompt is defined as *"a photo of a: <room or object>"*. An instruction is then classified as containing errors if $\mathcal{K} \nsubseteq \mathcal{S}$ (*i.e.,* instruction tokens $\mathcal{K}$ are not observed during navigation). To localize the errors, we retrieve the predicted token indexes as $\{\hat{l} \,|\, k_{\hat{l}} \in \mathcal{K} : k_{\hat{l}} \notin \mathcal{S}\}$.

Effect of errors on VLN agent's performance. This experiment analyzes how each error type affects the performance of a VLN policy in the continuous environment settings in terms of SR and SPL. Additionally, an analysis of the drop in SR is also computed as follows. For each error type, two separate evaluations are performed using only the correct $\mathcal{E}_\mathcal{C}$ and perturbed $\mathcal{E}_\mathcal{P}$ episode sets, respectively. Formally, the delta of SR between the two evaluations is defined as $\Delta_{SR}(\%) = SR(\mathcal{E}_\mathcal{C}) - SR(\mathcal{E}_\mathcal{P})$. Since $\mathcal{E}_\mathcal{P}$ is constructed directly from the $\mathcal{E}_\mathcal{C}$ set, the difference in SR between the two shows the effect of this error type on the policy. These results are reported in Tab. II under the SR, SPL and $\Delta_{SR}(\%)$ columns. It can be observed that the *Direction* type of error has the largest effect on the navigation policy, with a -18.64% relative decrease in SR. Following that, the *Object* error type with common sense leads to a relative decrease of -8.47% and the *Room* error with common sense to a decrease of -6.66%. Similarly to [18], we find that VLN agents heavily rely on directional and object tokens. Interestingly, we also find that, differently from [18], VLN agents in continuous environments are more affected by perturbation of directional tokens than by

TABLE II: Results of our proposed *IEDL* method on our proposed benchmark. We show the SR and SPL metrics of the frozen policy, and the drop in SR performance when errors are present (ΔSR%). We then analyze the classification (AUC) and Localization (ATD) performance of different methods. Error types with * indicate benchmark with common sense. We highlight in gray the main metric, *i.e.,* AUC.

Origin split	Error type	Policy [9]			Random		CLIP Alignment		**IEDL**	
		SR ↑	**SPL** ↑	Δ_{SR}(%)	**AUC** ↑	**ATD** ↓	**AUC** ↑	**ATD** ↓	**AUC** ↑	**ATD** ↓
R2R-CE Val Unseen	Direction	0.53	0.43	-18.64	0.50	10.54	0.50	11.05	0.58	8.13
	Room*	0.58	0.49	-6.66	0.50	11.03	0.57	9.63	0.80	7.73
	Object*	0.56	0.46	-8.47	0.51	10.94	0.59	8.76	0.74	9.21
	Room&Object *	0.57	0.47	-11.47	0.49	11.56	0.64	7.98	0.91	7.34
	All *	0.52	0.43	-30.64	0.51	12.22	0.63	8.68	0.94	6.14
	Avg.	0.55	0.46	-15.17	0.50	11.26	0.59	9.22	**0.79**	**7.71**

perturbation of object tokens (-18.64% *vs* -8.47%). We hypothesize that this is due to the "navigation-graph" of the discrete R2R environment, which may lead to strong implicit assumptions [5]. This finding suggests that more focus should be placed on grounding this type of directional information. Finally, combining the two errors *Room&Object* with common sense shows a drop in performance of -11.47%, less than the sum of the two separately. Lastly, the error type *All* with common sense, which combines all of the above, shows a decrease of -30.64%. These results, particularly in the *Direction* case, are a clear sign for the robotic community that efforts need to be made to increase awareness on this weakness in current VLN agents.

Does the instruction contain an error? In this test, the performance of the classification head f_d of *IEDL* is evaluated against two baselines in terms of AUC scores. The results are reported in Tab. II. Here, the AUC scores of a random baseline are presented as a means to identify potential biases and establish a lower bound. Given that the ratio between $\mathcal{E}_\mathcal{C}$ and $\mathcal{E}_\mathcal{P}$ episodes is 50%, AUC for the random baseline is always ~ 0.50, as expected. The CLIP Alignment baseline is presented here as a simple method that reflects how humans would solve the same problem in real life: checking if the content of the instruction is grounded to the observations. Also, note that CLIP Alignment does not require training. CLIP Alignment seems to be effective in the presence of *Object* and *Room* types of errors, with a total 0.64 of AUC score in *Room&Object* when both errors are present. Interestingly, CLIP Alignment performs on par with the random baseline for the *Direction* error type. The intuition is that, when errors are present near the end of the instruction, CLIP Alignment can still ground objects and room tokens to the observations (thus $\mathcal{K} \subseteq \mathcal{S}$), but may subsequently become lost due to the error. Finally, in Tab. II, we show our proposed *IEDL* method. We can see that *IEDL* achieves the best AUC in all the benchmarks by a large margin. A lower AUC of 0.58 for *IEDL* also seems to highlight the challenges of the *Direction* error type. Finally, *IEDL* has a much higher average AUC score (0.79) compared to Random (0.50) and CLIP Alignment (0.59).

Can we localize the error? This experiment evaluates the ability of the localization head f_l of *IEDL* to locate the errors within an instruction. The results in terms of ATD for the different baselines are also shown in Tab. II. The random baseline achieves a mean ATD of 11.26. The CLIP Alignment baseline performs better for all error types compared to random. The policy of considering as token index error $\hat{l}$ the instruction token that was not observed during navigation seems effective, especially in the *Room&Object* benchmark. Finally, the proposed *IEDL* achieves the best localization performance across all benchmarks, except for the *Object*, in which CLIP Alignment seems to be particularly effective. However, it should be noted that, in the *Object* case, the AUC score for CLIP Alignment is 0.59 *vs* **0.74** of *IEDL*, which indicates the ability of CLIP Alignment in identifying only a subset of objects that have been swapped. Finally, it is worth noticing that the mean ATD of *IEDL* is 7.71, which is close to the length of a typical sub-sentence within each instruction, showing that our method can be used to localize errors at the sub-sentence level.

What if we apply *IEDL* on the R2R-CE dataset? This experiment shows how a pre-trained *IEDL* can be used as a semi-automatic tool for potentially identifying error-containing episodes, including those within the original R2R-CE dataset. Specifically, we apply our *IEDL* f_d detection head on the original Val Unseen split of R2R-CE. We then isolate episodes that produce an alignment score a above a set threshold $\tau_a = 0.99$. Surprisingly, *IEDL* isolates 25 episodes out of the total 1839. Upon manual inspection, we confirmed that 8 out of the 25 isolated videos were indeed wrong ground-truth annotations. This experiment suggests the possibility that evaluations in VLN-CE may be skewed due to several wrongly annotated episodes in existing datasets, which should be addressed in future evaluations. Examples can be found in the supplementary materials.

Can *IEDL* generalize to other datasets and models? To test this hypothesis, we apply the trained *IEDL* on top of another policy, *i.e.,* ETPNav [10], on the challenging RxR-CE [7]. RxR-CE trajectories and instructions are much longer than R2R-CE (with an average length of 110 tokens *vs* 30 tokens, respectively) and more detailed. Without *any* training on this dataset, *IEDL* flagged 118 episodes out of 3669 (we set $\tau_a = 0.99$, on the episode with language en-IN and en-US). Surprisingly, after visual inspection 10 of these episodes were found to contain an error, either in a direction, an object, in the goal position, or a mesh issue, and should be removed from the validation split.

VII. Conclusion

We presented the novel benchmark R2RIE-CE, a new benchmark for VLN-CE, in which different types of errors are injected into textual instructions. Based on our experiments, state-of-the-art VLN-CE methods are affected by instruction perturbations. On top of R2RIE-CE, we also established the novel task of *Detection and Localization of Instruction Errors*. Our proposed method, *IEDL*, composed of detection and localization heads, can detect and localize errors within a sub-sentence distance on the original instruction. As a further experiment, by running *IEDL* on the standard R2R-CE and RxR-CE, we were able to find 8 and 10 episodes, respectively, that should be removed from the validation set. Our benchmark and proposed method support the development of more reliable and versatile agents capable of handling real-world ambiguity and uncertainty. In future work, we will investigate error-aware policy learning to improve the navigation performance in the VLN-CE task, and online instructions errors detection [31].

References

[1] P. Anderson, Q. Wu, D. Teney, J. Bruce, M. Johnson, N. Sunderhauf, I. Reid, S. Gould, and A. van den Hengel, "Vision-and-Language Navigation: Interpreting Visually-Grounded Navigation Instructions in Real Environments," in *CVPR*, Jun. 2018. http://dx.doi.org/10.1109/cvpr.2018.00387

[2] J. Gu, E. Stefani, Q. Wu, J. Thomason, and X. Wang, "Vision-and-Language Navigation: A Survey of Tasks, Methods, and Future Directions," in *Proc. of Association for Computational Linguistics*. ACL, 2022. http://dx.doi.org/10.18653/v1/2022.acl-long.524

[3] M. Savva, A. Kadian, O. Maksymets, Y. Zhao, E. Wijmans, B. Jain, J. Straub, J. Liu, V. Koltun, J. Malik, D. Parikh, and D. Batra, "Habitat: A Platform for Embodied AI Research," in *ICCV*, Oct. 2019. http://dx.doi.org/10.1109/iccv.2019.00943

[4] A. Chang, A. Dai, T. Funkhouser, M. Halber, M. Niebner, M. Savva, S. Song, A. Zeng, and Y. Zhang, "Matterport3D: Learning from RGB-D Data in Indoor Environments," in *3DV*, CA, USA, oct 2017, pp. 667–676. https://doi.ieeecomputersociety.org/10.1109/3DV.2017.00081

[5] J. Krantz, E. Wijmans, A. Majumdar, D. Batra, and S. Lee, *Beyond the Nav-Graph: Vision-and-Language Navigation in Continuous Environments*. ECCV, 2020, p. 104–120. http://dx.doi.org/10.1007/978-3-030-58604-1_7

[6] Z. Wang, J. Li, Y. Hong, Y. Wang, Q. Wu, M. Bansal, S. Gould, H. Tan, and Y. Qiao, "Scaling Data Generation in Vision-and-Language Navigation," in *ICCV*, Oct. 2023. http://dx.doi.org/10.1109/iccv51070.2023.01103

[7] A. Ku, P. Anderson, R. Patel, E. Ie, and J. Baldridge, "Room-Across-Room: Multilingual Vision-and-Language Navigation with Dense Spatiotemporal Grounding," in *EMNLP*, 2020. http://dx.doi.org/10.18653/v1/2020.emnlp-main.356

[8] J. Hort, J. Laczó, M. Vyhnálek, M. Bojar, J. Bureš, and K. Vlček, "Spatial navigation deficit in amnestic mild cognitive impairment," *Proc. of the National Academy of Sciences*, vol. 104, no. 10, p. 4042–4047, Mar. 2007. http://dx.doi.org/10.1073/pnas.0611314104

[9] D. An, Y. Qi, Y. Li, Y. Huang, L. Wang, T. Tan, and J. Shao, "BEVBert: Multimodal Map Pre-training for Language-guided Navigation," *ICCV*, 2023. https://doi.org/10.48550/arXiv.2212.04385

[10] D. An, H. Wang, W. Wang, Z. Wang, Y. Huang, K. He, and L. Wang, "ETPNav: Evolving Topological Planning for Vision-Language Navigation in Continuous Environments," *TPAMI*, pp. 1–16, 2024.

[11] Y. Hong, Z. Wang, Q. Wu, and S. Gould, "Bridging the Gap Between Learning in Discrete and Continuous Environments for Vision-and-Language Navigation," in *CVPR*, Jun. 2022. http://dx.doi.org/10.1109/cvpr52688.2022.01500

[12] X. Wang, Q. Huang, A. Celikyilmaz, J. Gao, D. Shen, Y.-F. Wang, W. Y. Wang, and L. Zhang, "Reinforced Cross-Modal Matching and Self-Supervised Imitation Learning for Vision-Language Navigation," in *CVPR*, Jun. 2019. http://dx.doi.org/10.1109/cvpr.2019.00679

[13] D. Fried, R. Hu, V. Cirik, A. Rohrbach, J. Andreas, L.-P. Morency, T. Berg-Kirkpatrick, K. Saenko, D. Klein, and T. Darrell, "Speaker-Follower Models for Vision-and-Language Navigation," in *NeurIPS*, vol. 31. Curran Associates, Inc., 2018. https://proceedings.neurips.cc/paper_files/paper/2018/file/6a81681a7af700c6385d36577ebec359-Paper.pdf

[14] S. Chen, P.-L. Guhur, C. Schmid, and I. Laptev, "History Aware Multimodal Transformer for Vision-and-Language Navigation," in *NeurIPS*, vol. 34. Curran Associates, Inc., 2021, pp. 5834–5847. https://proceedings.neurips.cc/paper_files/paper/2021/file/2e5c2cb8d13e8fba78d95211440ba326-Paper.pdf

[15] Y. Hong, Q. Wu, Y. Qi, C. Rodriguez-Opazo, and S. Gould, "VLN⟳BERT: A Recurrent Vision-and-Language BERT for Navigation," in *CVPR*, Jun. 2021. http://dx.doi.org/10.1109/cvpr46437.2021.00169

[16] S. Chen, P.-L. Guhur, M. Tapaswi, C. Schmid, and I. Laptev, "Think Global, Act Local: Dual-Scale Graph Transformer for Vision-and-Language Navigation," in *CVPR*, Jun. 2022. http://dx.doi.org/10.1109/cvpr52688.2022.01604

[17] Z. Wang, X. Li, J. Yang, Y. Liu, and S. Jiang, "GridMM: Grid Memory Map for Vision-and-Language Navigation," in *ICCV*, oct 2023, pp. 15 579–15 590. https://doi.ieeecomputersociety.org/10.1109/ICCV51070.2023.01432

[18] W. Zhu, Y. Qi, P. Narayana, K. Sone, S. Basu, X. Wang, Q. Wu, M. Eckstein, and W. Y. Wang, "Diagnosing Vision-and-Language Navigation: What Really Matters," in *NAACL*, Seattle, United States, Jul. 2022, pp. 5981–5993. https://aclanthology.org/2022.naacl-main.438

[19] M. Hahn, A. Raj, and J. M. Rehg, "Which way is 'right'?: Uncovering limitations of Vision-and-Language Navigation models," in *AAMAS*, Richland, SC, 2023, p. 2415–2417. https://api.semanticscholar.org/CorpusID:253381693

[20] Y. Zhang, H. Tan, and M. Bansal, "Diagnosing the Environment Bias in Vision-and-Language Navigation," in *IJCAI*, ser. IJCAI-PRICAI-2020, Jul. 2020. http://dx.doi.org/10.24963/ijcai.2020/124

[21] Z. Yang, A. Majumdar, and S. Lee, "Behavioral Analysis of Vision-and-Language Navigation Agents," in *CVPR*, Jun. 2023. http://dx.doi.org/10.1109/cvpr52729.2023.00253

[22] H. Huang, V. Jain, H. Mehta, A. Ku, G. Magalhaes, J. Baldridge, and E. Ie, "Transferable Representation Learning in Vision-and-Language Navigation," in *ICCV*, Oct. 2019. http://dx.doi.org/10.1109/iccv.2019.00750

[23] M. Zhao, P. Anderson, V. Jain, S. Wang, A. Ku, J. Baldridge, and E. Ie, "On the Evaluation of Vision-and-Language Navigation Instructions," in *EACL*, Online, Apr. 2021, pp. 1302–1316. https://aclanthology.org/2021.eacl-main.111

[24] H. Tan, L. Yu, and M. Bansal, "Learning to Navigate Unseen Environments: Back Translation with Environmental Dropout," in *NAACL*, Minneapolis, Minnesota, Jun. 2019, pp. 2610–2621. https://aclanthology.org/N19-1268

[25] X. Liang, F. Zhu, Y. Zhu, B. Lin, B. Wang, and X. Liang, "Contrastive Instruction-Trajectory Learning for Vision-Language Navigation," *AAAI*, vol. 36, no. 2, pp. 1592–1600, Jun. 2022. https://ojs.aaai.org/index.php/AAAI/article/view/20050

[26] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of deep bidirectional transformers for language understanding," in *NAACL*, vol. 1, 2019, p. 4171 – 4186. https://www.scopus.com/inward/record.uri?eid=2-s2.0-85083815650&partnerID=40&md5=4986c6d6076c0c91df84d17216b47216

[27] A. Radford, J. W. Kim, C. Hallacy, A. Ramesh, G. Goh, S. Agarwal, G. Sastry, A. Askell, P. Mishkin, J. Clark, G. Krueger, and I. Sutskever, "Learning Transferable Visual Models From Natural Language Supervision," in *ICML*, vol. 139. PMLR, 18–24 Jul 2021, pp. 8748–8763. https://proceedings.mlr.press/v139/radford21a.html

[28] H. Tan and M. Bansal, ""LXMERT: Learning cross-modality encoder representations from transformers"," in *EMNLP-IJCNLP*, K. Inui, J. Jiang, V. Ng, and X. Wan, Eds., Hong Kong, China, Nov. 2019, pp. 5100–5111. https://aclanthology.org/D19-1514

[29] P. Anderson, A. Chang, D. S. Chaplot, A. Dosovitskiy, S. Gupta, V. Koltun, J. Kosecka, J. Malik, R. Mottaghi, M. Savva *et al.*, "On evaluation of embodied navigation agents," *arXiv preprint arXiv:1807.06757*, 2018. https://doi.org/10.48550/arXiv.1807.06757

[30] S. Bird and E. Loper, "NLTK: the natural language toolkit," in *ACL*, 2004. http://dx.doi.org/10.3115/1219044.1219075

[31] F. Taioli, S. Rosa, A. Castellini, L. Natale, A. D. Bue, A. Farinelli, M. Cristani, and Y. Wang, "I2EDL: Interactive Instruction Error Detection and Localization," 2024. https://arxiv.org/abs/2406.05080