Text2Map: From Navigational Instructions to Graph-Based Indoor Map Representations Using LLMs

Ammar Karkour[1], Khaled A. Harras[1], and Eduardo Feo-Flushing[1]

Abstract— **In spatial navigation, the shift from manual cartography to digital map representations has revolutionized how we interact with and comprehend outdoor and indoor environments. While digital mapping has substantially advanced outdoor navigation with robust techniques like satellite imagery and sophisticated data labeling, the full potential of indoor digital mapping remains untapped. Accurate indoor mapping promises to enhance the operational efficiency of mobile robots, improving their ability to interact with human environments, and bolstering emergency response capabilities. However, its realization is impeded by the complexity of current methods and the need for heavy manual labor, expert knowledge, and specialized equipment. To address these challenges, we introduce Text2Map – a novel methodology that harnesses natural language navigational instructions, the power of off-the-shelf Large Language Models (LLMs), and Few-shot Learning, to create graph-based digital maps of indoor spaces. This approach simplifies the mapping process for widespread use, leveraging crowd-sourceable ubiquitous navigation instructions as a data source without requiring specialized map data formats or hardware. Our paper presents the Text2Map system architecture, details the creation of the first dedicated dataset, and evaluates the system's efficacy, highlighting the substantial potential and scalability of our approach. Text2Map achieves a Graph-Edit-Distance (GED) ranging from 0.5X to 2X the total number of regions in a building and an Edge Similarity score between 0.87 and 0.9. These results highlight the precision, robustness, and effectiveness of our methodology. Our work paves the way for a more accessible and streamlined approach to indoor digital mapping, setting the stage for broader adoption in human and mobile robot navigation applications.**

I. Introduction

Mapping has always been a cornerstone of understanding and using spaces effectively. The transition from traditional cartography to digital mapping has leveraged technological advancements to offer a more comprehensive view of our surroundings, enhancing efficiency across various applications, from personal navigation to intricate logistic operations [1], [2]. This evolution has benefited outdoor navigation through satellite imagery and advanced data labeling techniques. However, the potential for digital mapping to transform indoor spaces has yet to be realized. Accurate indoor maps promise to boost the operational efficiency of autonomous systems, where precise and comprehensive abstractions of spatial layouts are crucial for navigation, task execution, and other indoor smart applications [3], [4]. By leveraging indoor maps, humans and mobile robots can optimize routes [5], [6], aid in indoor localization solutions and applications [7]–[9], and interact more seamlessly in human-centric environments. Realizing this potential depends on streamlining the process of creating indoor maps. This involves minimizing the reliance on specialized knowledge and equipment, reducing costs, and making digital mapping more accessible.

The methods for creating digital indoor maps are diverse, each with its own set of tools and outputs *(Section II)*. Modeling techniques rely on specialized software or scanning equipment to construct representations such as Computer-Aided Design (CAD) drawings, Building Information Modeling (BIM) structures, Point Clouds, or Texture Meshes. Although these models provide detailed representations, they often lack semantic and topological context, essential for various applications [1], [2], [10]. On the other hand, indoor map data formats such as CityGML, indoorGML [11], and Apple's IMDF offer an abundance of semantic and topological information but come with their own challenge of complex formats that require expertise to handle them [12]. The challenges of creating indoor map representations stem from the intricacies of both modeling and data formats. The reliance on experts familiar with specific software, data formats, and scanning hardware, escalates costs and hinders the widespread adoption of digital indoor mapping [13]. These complexities highlight the need for low barrier solutions to create navigable indoor maps, that do not require deep expertise in map data formats or specialized hardware.

In this paper, we present *Text2Map*, a novel solution that simplifies the creation of digital navigable indoor maps without requiring expert knowledge or specialized equipment *(Section III)*. By leveraging simple natural language navigation instructions, and the capabilities of off-the-shelf Large Language Models (LLMs), *Text2Map* eliminates the complexity associated with advanced map data format construction. *Text2Map* contains a *Prompting Engine* that refines and generates navigation sequences that cover the indoor space and utilizes *Few-shot learning* to instruct the LLM. This results in a graph-based *Connectivity Matrix* that serves as the building's map. Adopting human language negates the need for expertise in specialized software or hardware. This approach is effective because navigation instructions are widely used, providing us with a large amount of data that can be captured through *crowd-sourcing*. This simplification will expedite the creation of navigable map representations and significantly reduce costs. These improvements are essential for large-scale buildings, buildings in locations where obtaining or using advanced scanning tools is challenging, and buildings that frequently change their indoor layouts, such as exhibition centers. Furthermore, a graph-based map format composed of matrices and vectors is well-studied and

[1]School of Computer Science, Carnegie Mellon University, `{akarkour, kharras, efeoflus}@andrew.cmu.edu`

readily adaptable by various graph algorithms to execute different navigation tasks, thereby eliminating the need to construct complicated specialized parsers.

We detail the creation of the *Text2Map Dataset*, a pioneering dataset that uses human navigation instructions to build connectivity matrices *(Section V)*. Text2Map encompasses 90 buildings and 22k unique instructions, generating an exponential number of unique navigation sequences per building. We evaluate *Text2Map* using Graph-Edit-Distance (GED) and Edge Similarity metrics, conduct an ablation study, and evaluate performance across various scenarios and building sizes. Our findings show that Text2Map achieves a GED of **0.5X** to **2X** the number of regions in a building and an Edge Similarity of **0.87** to **0.9**, demonstrating its effectiveness and efficiency compared to human effort.

We highlight our key contributions as follows:

- *Text2Map*: a novel solution utilizing off-the-shelf LLMs and Few-shot learning to generate digital navigable indoor maps from natural language navigational instructions.
- Built *Text2Map Dataset*, the first dataset dedicated to generating indoor connectivity matrices from human navigation instructions, and established a comprehensive evaluation framework with GED and Edge Similarity metrics.
- Conducted a thorough experimental analysis, including an ablation study to assess the significance of *Text2Map*'s components and experiments to evaluate its performance across various conditions and building complexities.

II. Related Work

This section explores the evolution of indoor mapping from robotics to data-only methods, emphasizing the reduced need for specialized hardware. We also discuss how LLMs can overcome the limitations of data-only methods, similar to their use in other graph-related tasks.

In the field of *mobile robotics*, effective navigation depends on the robot's ability to sense and model its environment, facilitated by Simultaneous Localization And Mapping (SLAM) [14]. This process, essential for integrating sensory data and self-localization, has driven progress in spatial mapping, allowing robots to navigate, plan routes, and circumvent obstacles. However, these spatial maps, focused solely on geometric data, lack semantic details, limiting their utility in human-centered applications and impeding interactive human-robot scenarios. To bridge this gap, enhancements in semantic indoor mapping have been introduced, where an additional layer of semantic information is integrated into the mapping process. This augmentation can involve direct human input through dialogue with humans, as demonstrated by Bastianelli et al. [15], or by using machine learning models for region recognition and scene understanding. Techniques range from constructing hierarchical 3D scene graphs that elucidate room relationships [16] to deploying vision-to-language models for scene description [17] and the use of region classifiers to label RGB-D observations [18]. Despite the depth of study on indoor mapping within robotics, the reliance on advanced hardware, such as robots and sensors, still poses a barrier to widespread adoption, limiting the ubiquity of these solutions.

To eliminate the need for specialized equipment, such as robots and scanners, to construct map representations, *data-only* indoor mapping methods leverage readily available information and data about a building to build indoor map representations. The most common approach is to create map data formats such as CityGML, indoorGML manually [11], and Apple's IMDF, which provide rich semantic and topological details but require extensive manual effort and specialized knowledge of the data formats [12]. Recent advances propose the use of natural language descriptions of the characteristics of buildings, followed by deep learning models such as Stanford Scene Graph Parser [19] or Generative Adversarial Networks (GANs) [20], to generate graph-based indoor maps. This strategy reduces the reliance on complex data formats. Still, it requires detailed knowledge of the space's design and function, posing challenges for crowd-sourcing such tasks or generally assigning them. Moreover, training these models necessitates large datasets of indoor layouts, which are challenging to compile due to privacy and data sensitivity concerns.

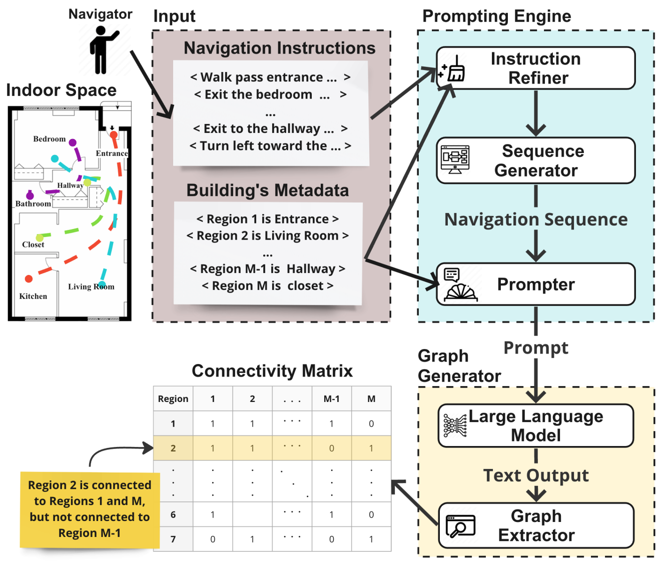
Our method simplifies the complexity of data collection by using straightforward navigational instructions that can be easily crowd-sourced or authored by anyone. To transform these simple pieces of relational data into graph-based maps, we leverage the capabilities of LLMs, which excel at extracting accurate relational information and converting it into graphical representations. Integrating LLMs into graph-based tasks has significantly propelled the field of graph-based applications forward, introducing innovative ways to augment and improve algorithms that use graphs and analyze relational data to generate graph-based representations. For example, utilizing LLMs like GPT-3.5 and GPT-4 has enhanced graph neural networks (GNNs) by processing and improving text-based node attributes for tasks such as paper classification, thus boosting GNN performance [21], [22]. On the other hand, LLMs have facilitated graph construction for applications such as market analysis by analyzing textual reports and identifying connections between different entities [23]. Overall, this synergy between LLMs and graph models boosts existing algorithm performance and opens new avenues for analyzing relational data.

III. Text2Map: Components and Design

In this section, we delve into the components of Text2Map as depicted in Fig. 1, highlighting their functions and the challenges that influenced our design and technical decisions. We begin by formally defining the task, and then discuss the input, prompting engine, and graph extractor components.

A. Formal Task Definition

Given a navigation sequence $N = \langle n_1, n_2, \ldots, n_L \rangle$, where L represents the length of the sequence, and each n_i is a natural language navigation instruction for known start and end points, with the entire sequence covering the entire indoor space. Additionally, given a sequence of room-like



Region	1	2	. . .	M-1	M
1	1	1	. . .	1	0
2	1	1	. . .	0	1
⋮	⋮	⋮	⋱	⋮	⋮
6	1		. . .	1	0
7	0	1	. . .	0	1

Fig. 1: Text2Map Overview

regions $R = \langle r_1, r_2, \ldots, r_M \rangle$, where M denotes the number of room-like regions in the indoor space, and each r_i is a descriptive sentence formatted as *"region r_i is a X"*, with X being the label/name of the region. The task of the model is to process these inputs and generate a graph G, where

$$G = \langle \langle x_{11}, x_{12}, \ldots, x_{1M} \rangle_1, \\ \langle x_{21}, x_{22}, \ldots, x_{2M} \rangle_2, \\ \ldots, \\ \langle x_{M1}, x_{M2}, \ldots, x_{MM} \rangle_M \rangle$$

, and $x_{ij} \in \{0, 1\}$. In this graph representation, region i is connected to region j if and only if $x_{ij} = x_{ji} = 1$.

B. Input

Our input for creating navigable map representations consists of two primary components. The first part is a textual description of the indoor space articulated through navigation instructions. Each navigation instruction has known starting and ending room-like regions and spans multiple regions across the path it describes. The second part includes metadata about the building's structure, specifically detailing the room-like regions within the building and their corresponding labels. It is reasonable to expect that individuals tasked with mapping an indoor space can navigate through it and provide descriptive instructions on moving from one location to another. Additionally, it is generally feasible for them to gather information about the various room-like regions they aim to include in their navigable map.

C. Prompting Engine

The *Prompting Engine*, which includes the *Instruction Refiner*, *Sequence Generator*, and *Prompter*, processes raw natural language navigation instructions along with the building's metadata. This engine enhances and supplements the instructions with the necessary data, facilitating the LLM to generate precise indoor maps efficiently. We discuss the functionality of each component and the challenges it solves.

1) Instruction Refiner: One of the main challenges of our task is that people do not necessarily refer to the regions they are in while annotating navigation instructions but rather to the environment around them, i.e., the objects within each region. Not referring to the regions would make our task harder because the graph generator would need to infer some regions only based on the objects in it. As seen in the example below, this navigation path covers regions 8 (lobby), 13 (living room), 10 (stairs), and 7 (hallway), but they are not directly mentioned in it.

> *Walk past the dining table and head up the stairs. Stop at the top of the stairs.*

To address this issue, we define an *Instruction Refiner* component, that complements each instruction with extra-regional clues, and highlights regions that can be inferred from their environmental description. Even though any navigation instruction has known starting and ending regions, these regions may not be explicitly mentioned, as the person describing might assume that the agent navigating the space is already at the starting region and can deduce that they arrived at the ending region once the instruction ends. We refine each instruction by adding the start and end regions which are given by definition (section III-B). In addition, the *Instruction Refiner* scans each instruction for the mentioned regions (key-words search) and aids it with its unique *ID*, obtained from the building's metadata, as seen in the example below.

> ***You are in the lobby (region 8).*** *Walk past the dining table and head up the stairs* ***(region 10).*** *Stop at the top of the stairs.* ***You arrived at hallway (region 7).***

This approach ensures that while the model needs to deduce some regions along the path, keeping the task challenging, it at least has clear starting and ending points, and can easily identify different regions based on their *ID*.

2) Sequence Generator: Our solution utilizes navigation instructions used daily by individuals, and thus can be crowd-sourced or built in-house. The nature of such instructions is that they describe a path starting and ending in different regions. These paths do not usually cover all the regions in the space, and different paths have a lot of overlapping covered regions. The goal of our *Sequence Generator* is to group the instructions so that the combination of regions covered by all the instructions in a sequence covers the whole building. At the same time, the number of instructions in a sequence is small to minimize the required manual effort.

Given a set of navigation instructions $\mathcal{I}$, where each covers known starting and ending regions, and intermediate regions deduced by the *Instruction Refiner*, the *Sequence Generator* separates these instructions into groups based on the regions they cover. This process generates subsets of $\mathcal{I}$, with each subset corresponding to a region, ensuring that every instruction within a subset incorporates that region in its path. Let the building comprise l regions with k total navigation

Algorithm 1 Sequence Generation Algorithm

```
1: Input: A set of navigation instructions $\mathcal{I}$, total regions $l$
2: Output: An iterator over distinct sequences $\mathcal{S}$
3: function GroupByRegion($\mathcal{I}$)
4:     $\mathcal{R} \leftarrow \{\}$
5:     for each $i \in \mathcal{I}$ do
6:         for each $r \in i$.regions do
7:             $\mathcal{R}[r] \leftarrow \mathcal{R}[r] \cup \{i\}$
8:     return $\mathcal{R}$
9: end function
10: function GenerateCombinations($\mathcal{R}$, $l$)
11:     define $gen(r = 1, \text{path} = [])$:
12:         if $r > l$ then
13:             yield path
14:         else
15:             for each $i \in \mathcal{R}[r]$ do
16:                 yield from $gen(r + 1, \text{path} + [i])$
17:     return $gen()$
18: end function
19: $\mathcal{R} \leftarrow$ GroupByRegion($\mathcal{I}$)
20: $\mathcal{S} \leftarrow$ GenerateCombinations($\mathcal{R}$, $l$)
21: iterate over $\mathcal{S}$
```

instructions. The average number of instructions per region is calculated as $m = \frac{k}{l}$. When constructing sequences, an instruction is selected from each region's subset. Therefore, the sequence length of a building is $\leq l$, where l represents the total number of regions in the building. Applying the fundamental principle of counting, the approximate total number of distinct sequences for a building is estimated as m^l, considering l regions and m instructions per region.

This methodology limits the length of a navigation sequence to the number of regions in a building while ensuring full coverage of the space. Notice that the number of generated sequences is exponential in the number of regions of the building, which means that using a small number of instructions, we can generate a lot of different sequences that describe the indoor space. To avoid exponential run-time, we use lazy evaluation of sequences, generating them as needed rather than all at once, as seen in Algorithm 1.

3) Prompter: The role of the *Prompter* is to combine the generated sequences and the buildings' meta-data into one prompt using a template and different strategies to prompt the *LLM*. We use the following template to pass the information mentioned in section III-A:

Act as a computer scientist cartographer and create a map representation of an indoor building. I will provide you with 3 things. First, the expected output format. Second, information about the room-like regions of the building. We might have regions that have the same name/label. Third, instructions for an agent to navigate the building. The navigation instructions are independent of each other and they are not ordered. Also, please only focus on the room-like regions of the building, and do not include any information about the objects in it.

OUTPUT FORMAT:
Return only a JSON object. The JSON contains one key named connectivity_graph and its value is a Python dictionary. The dictionary contains a key for each region, and the value for each key is a list of indices of regions connected to it.

ROOM-LIKE REGIONS INFORMATION:
...
NAVIGATION INSTRUCTIONS:
...

Following the formation of the prompt, the *prompter* utilizes *Few-shot learning* as a prompting technique. Few-shot learning is a prompting technique where we provide a pre-trained model with examples of the task we want to execute to help the model learn to produce good results [24]. We define shots by examples of different buildings other than the one being tested. Toward this goal, the *Prompter* utilizes Zero-shot, One-shot, and Few-shot strategies to get the best results from the LLM.

D. Graph Generator

The *Graph Generator* contains an off-the-shelf LLM that processes the prompts generated by the Prompting Engine to generate text that contains the Connectivity Matrix representing the map of the indoor place. Then, a *Graph Extractor* parses the LLM's output to extract the Connectivity Matrix.

1) Pre-trained LLM: LLMs like OpenAI's GPT series have significantly advanced the field of natural language processing. These models, now easily accessible via cloud services and APIs, offer robust capabilities in text generation, comprehension, and more [25]. Their widespread availability has made cutting-edge AI accessible to all, enabling researchers and developers to integrate advanced language understanding into applications effortlessly. Our solution capitalizes on such advancement and utilizes an LLM as one of its main components. The LLM would take the prompts given to it by the *Prompting Engine* and deduces the different connections between the room-like regions of the space.

2) Graph Extractor: The *LLM* outputs a text output that contains the connectivity matrix of the described space. The *Graph Extractor* takes this output and parses it to extract the output graph representation.

IV. Evaluation Framework

In this section, we detail the creation of the Text2Map dataset, the first of its kind for the task described in Section III-A, which we employ for evaluation. We discuss the selected LLMs for our analysis, emphasizing their alignment with our project's aim to minimize reliance on inaccessible tools. Furthermore, we outline the evaluation metrics—BP2-Graph Edit Distance (BP2-GED) and Edge Similarity—explaining their selection rationale. Together, these elements constitute our experimental framework for the evaluation process.

A. Creating The Text2Map Dataset

Creating navigable map representations of indoor spaces from natural language navigation instructions using language models is an uncharted area with no existing data sets for model evaluation or training. To overcome this dataset shortfall, we created the first dataset specifically tailored for this

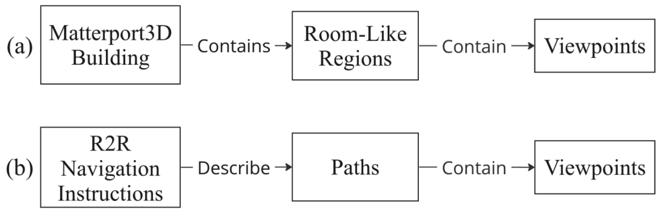


Fig. 2: Hierarchical structure of (a) Matterport3D buildings and (b) R2R Navigation Instructions.

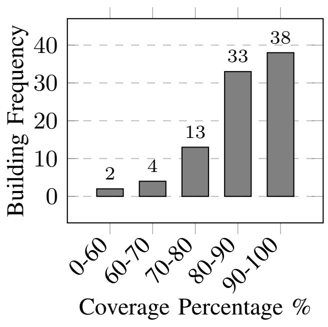


Fig. 3: The distribution of Matterport3D buildings based on the coverage percentage achieved through R2R navigation path instructions

Parameter	Value
Buildings Count	90
Regions Range	1-72
Prompts/Building	5
Total Prompts	450

Regions	Building Freq
01 to 10	17
11 to 20	26
21 to 30	30
31 to 72	17

Fig. 4: *Text2Map-Test* dataset details summary

task. For this purpose, we utilized the Room-to-Room (R2R) and Matterport3D datasets, which provide a solid foundation to build our dataset, *Text2Map*. The R2R dataset, crucial for Vision-Language-Navigation (VLN) tasks, integrates spoken instructions for navigating from start to destination within the Matterport3D dataset's environments, spanning 90 diverse buildings and featuring more than 10,000 panoramic views from 194,400 RGB-D images, including houses, apartments, hotels, offices, and churches. Matterport3D's data contain connectivity graphs and object information, enabling agents to navigate through its simulator and experience varied viewpoints. The R2R dataset itself includes 21,567 crowd-sourced open-vocabulary navigation instructions, averaging 29 words and detailing paths often crossing multiple room-like regions and over 10 meters in length [26], [27].

1) Building Text2Map Inputs: As shared earlier, our inputs are navigation instructions and metadata per building. To build our inputs, we first group the R2R instructions based on their building ID. Next, we group the instructions in each building by the regions they cover. Fig. 2 shows the hierarchical structure of the data in R2R and Matterport3D and how they can be mapped to each other. This way we can clearly know what regions each instruction covers, and consequently, produce a lot of navigation sequences that can describe each building. Additionally, Matterport3D provides us with metadata files for each building, which we parse to format the metadata in our desired formats.

2) Building Text2Map Outputs: Matterport3D provides us with connectivity graphs between *viewpoints* within each building. We group these *viewpoints* based on the region they belong to. Based on the external connections between the viewpoints of each region and the viewpoints that belong to other regions, we build connectivity graphs that describe the connectivity between room-like regions of each building.

However, while these graphs cover all regions in the building, R2R navigation instructions do not necessarily cover the whole building, as seen in Fig. 3, in fact, on average only 87% of each building is fully covered. To solve this issue, we remove the regions that are not covered by the navigation instructions. This way we guarantee that we will not have any regions in the ground-truth connectivity graphs that are not covered by the navigation instructions.

3) Building the Text2Map-Test Dataset: Our dataset encompasses 22,000 navigation instructions across 90 buildings, highlighting the potential to generate an exponential number of unique sequences to describe each structure comprehensively. For our experiments, we strategically generate and utilize only 5 distinct sequences for each building. These sequences serve as the foundation for conducting various experiments, allowing us to explore the efficacy of different components and prompting techniques within our pipeline.

B. Chosen LLMs

For our experiments, we utilize OpenAI's GPT-4 and GPT-3.5 LLMs [28], [29]. Besides their powerful capabilities, another reason for choosing OpenAI's GPT models is their accessibility. Both GPT-3.5 and GPT-4 are available through OpenAI's API, which offers an easy and user-friendly way for developers and researchers to access these powerful language processing tools. This accessibility greatly benefits our research, as it permits us to utilize their advanced features without necessitating considerable computational power for running and training on our side. Consequently, we circumvent the use of inaccessible specialized tools, aligning with our research objectives. [29].

C. Evaluation Metrics

Text2Map creates a graph-based connectivity matrix. To assess its performance, we require metrics that measure its proficiency in grasping the structure of the space, focusing on structural similarity. Additionally, we need to measure Text2Map's ability to avoid falsely introducing non-existent edges. This section outlines the evaluation metrics chosen and explains the rationale for their selection.

1) Approximated Graph Edit Distance (BP2-GED): For our task, we utilize the *graph edit distance* (GED) metric to measure graph similarity. GED is widely used in pattern recognition, bioinformatics, and network analysis to compare graph-structured data [30]. We define the *GED* between two graphs $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$ as the minimum cost of edit operations required to transform G_1 into G_2, where V denotes vertices and E denotes edges:

$$\mathrm{GED}(G_1, G_2) = \min \left\{ \sum_{e \in \mathcal{E}} \mathrm{cost}(e) : \mathcal{E} \text{ transforms } G_1 \text{ into } G_2 \right\} \tag{1}$$

The sequence of edit operations $\mathcal{E}$ includes vertex and edge insertions, deletions, and substitutions. The cost associated with them e is defined as $\mathrm{cost}(e) = 1$ for all edit operations.

However, the problem of optimizing GED is known to be NP-complete. This means that the run-time for minimizing the edit costs may be huge even for rather small graphs. Consequently, rather than calculating the GED, we adopt the approximation technique developed by Fischer et al. [31], referred to as BP2. BP2 represents an innovative quadratic time approach that merges the Hausdorff Edit Distance (HED) [32] and the Greedy Edit Distance (BP-Greedy) [33] to estimate an upper limit of the GED. This method achieves an approximation error that ranges from **0. 4%** to **10. 8%**. Therefore, we use BP2 as an evaluation metric where:

$$\text{BP2-GED}(G_1, G_2) \geq \text{GED}(G_1, G_2) \tag{2}$$

2) Total Edge Similarity: We consider the Edge Similarity metric to evaluate the model's performance in correctly predicting graph connections, which helps in assessing the model's ability to avoid false positives and false negatives. Edge Similarity is mathematically defined as:

$$\text{EdgeSim}(G_A, G_B) = \frac{|TP| + |TN|}{|TP| + |FP| + |TN| + |FN|} \tag{3}$$

where:
- TP (True Positives): Edges present in both G_A and G_B,
- TN (True Negatives): Edges absent in both G_A and G_B,
- FP (False Positives): Edges present in G_B but absent in G_A,
- FN (False Negatives): Edges present in G_A but absent in G_B.

This metric encapsulates the accuracy of the model in edge prediction, illustrating its proficiency in identifying correct connections while avoiding incorrect ones.

V. Experiments and Results

We initiate with an ablation study to demonstrate the significance of each Text2Map component. Subsequently, we analyze Text2Map's overall performance relative to varying shot numbers and building sizes. Lastly, we compare Text2Map's performance against the required human manual effort. Overall, our results indicate that Text2Map secures a BP2-GED ranging from **0.5X** to **2X** the total number of regions in buildings containing up to 72 regions. Furthermore, it achieves an Edge Similarity score between **0.87** and **0.9**, underscoring the effectiveness of our methodology.

A. Ablation Study: Evaluating Component Contributions

Metadata	Prompt Template	Instructions Refinements	Structured Sequences	BP2-GED	Edge Similarity
X	X	X	✓	60	0.47
X	✓	✓	✓	45	0.81
✓	X	✓	✓	42	0.82
✓	✓	X	✓	37	0.85
✓	✓	✓	X	73	0.78
✓	✓	✓	✓	**35**	**0.87**

TABLE I: Results of Ablation Study on 90 Matterport3D buildings with GPT-4 under Zero-shot learning

We conduct an ablation study to evaluate the contribution of each component of our solution to its overall effectiveness, employing the *Text2Map-Test* dataset (section IV-A.3) and *GPT-4* under *Zero-shot* learning. In this process, we average the metrics across all 90 buildings to understand the broader impact. We start by introducing the baselines that we examine, explaining the reasoning behind each choice. Then we present the findings of the ablation study. The following outline the baselines utilized in our experiments:

1) Only *Structured Navigation Sequences*.
2) Text2Map without *Building's Metadata*.
3) Text2Map without *Prompting Template*.
4) Text2Map without *Instructions Refinements*.
5) Text2Map without *Structured Navigation Sequences*.

Each baseline in our ablation study plays a crucial role in dissecting the functionality and contribution of distinct elements within Text2Map. We start with the most basic solution, which is just making sure that the navigation sequences are structured in a way that ensures full coverage of the whole space. Removing the *Building's Metadata* assesses the importance of contextual information about each building's specific characteristics and how they influence the model's comprehension in identifying correct regions. The absence of the *Prompting Template* allows us to evaluate the effectiveness of structured prompts in guiding the model towards more accurate output. We test this effectiveness by replacing the template with the following basic prompt *"Create a connectivity matrix from the following navigation instructions"*. By excluding *Instructions Refinements*, we test the sufficiency of the raw navigation instructions and identify the value added through our refinement process. Lastly, operating without *Structured Navigation Sequences* provides insights into the significance of having instructions that are guaranteed to fully cover the indoor space. We do this by testing against randomly generated sequences of length *l*, where *l* is the number of regions in the building.

The results of our ablation study, presented in Table I, offer a clear evaluation of the significance of each element within Text2Map. The study highlights performance degradation when any component is omitted. The removal of *Structured Sequences* notably incurs the most substantial performance decline, emphasizing their importance in ensuring comprehensive map coverage. The elimination of *Metadata* and *Prompt Template* also negatively affects performance, pointing to their role in improving the quality of the information processed by the LLM. The least affected by component removal is *Instructions Refinements*, suggesting potential areas for optimization, which we explore in subsequent sections. Overall, when comparing the simplest solution that guarantees complete map coverage (baseline 1) to our comprehensive Text2Map solution, it is evident that Text2Map significantly enhances performance, doubling both Edge-similarity and GED.

B. Impact of the Number of Few-Shot Learning Shots

Our Prompting Engine includes a Prompter component responsible for crafting prompts for the Graph Generator. We employed Few-Shot learning to enhance the LLMs' performance. To evaluate this strategy, we conducted experiments using Zero-Shot, One-Shot, and Five-Shot approaches. These experiments were performed using *GPT-3.5* and *GPT-4* on the *Text2Map-Test* dataset (section IV-A.3), encompassing

Shots Count	Regions Count	Buildings Count	BP2-GED	EdgeSim
	1-10	17	7	0.66
	10-20	26	27	0.84
Zero-shot	20-30	30	44	0.90
	30-72	17	77	0.93
	1-72	**90**	**39**	**0.84**
	1-10	17	7	0.73
	10-20	26	27	0.83
One-shots	20-30	30	39	0.89
	30-72	17	76	0.93
	1-72	**90**	**37**	**0.85**
	1-10	17	5	0.74
	10-20	26	22	0.85
Five-shot	20-30	30	40	0.88
	30-72	17	75	0.93
	1-72	**90**	**36**	**0.86**

TABLE II: Results of running our experiments on GPT-3.5

Shots Count	Regions Count	Buildings Count	BP2-GED	EdgeSim
	1-10	17	6	0.77
	10-20	26	25	0.86
Zero-shot	20-30	30	39	0.90
	30-72	17	73	0.94
	1-72	**90**	**35**	**0.87**
	1-10	17	6	0.83
	10-20	26	24	0.85
One-shot	20-30	30	38	0.90
	30-72	17	68	0.92
	1-72	**90**	**34**	**0.88**
	1-10	17	3	0.88
	10-20	26	18	0.85
Five-shot	20-30	30	34	0.90
	30-72	17	66	0.94
	1-72	**90**	**29**	**0.90**

TABLE III: Results of running our experiments on GPT-4

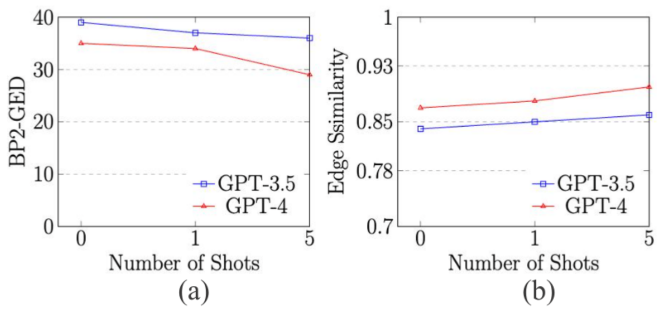


Fig. 5: Figures (a) and (b) illustrate the impact on BP2-GED and Edge Similarity with increasing shot numbers, respectively.

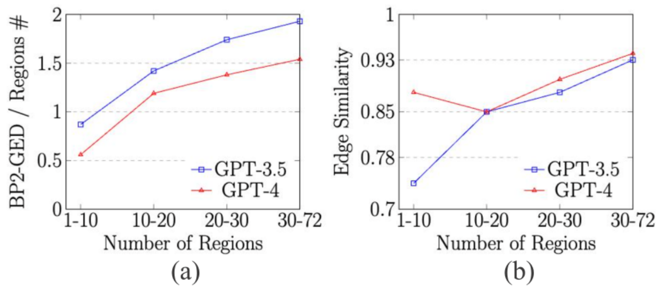


Fig. 6: Figures (a) and (b) depict how $\frac{\text{BP2-GED}}{\text{Number of Regions}}$ and Edge similarity change as the number of regions within each building increases under a five-shot learning prompting technique.

various building sizes. Details of the results are presented in tables II and III, while Fig. 5 displays the variation in the average values of our two metrics across the entire dataset, plotted against the changing number of shots.

Tables II and III reveal that Text2Map with GPT-3.5 achieves 39 BP2-GED and 0.84 Edge Similarity at Zero-shot, improving to 36 and 0.86 at Five-shot. With GPT-4, the metrics improve from 35 BP2-GED and 0.87 Edge Similarity at Zero-shot to 29 and 0.90 at Five-shot. Consequently, Fig. 5 shows a positive correlation between similarity scores and the number of shots, and a negative correlation between BP2-GED and the number of shots, reflecting the improvement as the model is exposed to more examples.

C. *Impact of Building Size*

Text2Map is significantly influenced by increasing building sizes, where the size is measured by the number of regions within a building. As detailed in section IV-A.3, the *Text2Map-Test* dataset includes a range of building types, such as houses, hotels, offices, and churches, with region counts from 1 to 72, providing a good view of Text2Map's performance across varying building sizes.

Tables II and III illustrate the performance of Text2Map with different building sizes. Fig. 6, shows that as the number of regions within a building increases, BP2-GED relative to the number of regions increases, going from **0.5X** to **2X**, reflecting increased structural complexity. Interestingly, Edge Similarity exhibits an upward trend with increasing building regions. This occurs because the increase in regions expands the potential edge connections, thereby elevating the chances that the model can correctly identify or dismiss connections, reducing false positives and negatives.

D. *Manual Effort vs. Text2Map Performance*

Manual effort can be captured by the GED, as it indicates the number of needed manual operations to fix the generated connectivity graph. The better the performance of Text2Map, the fewer manual operations are needed after generating the graph. As of now, the number of manual operations is **0.5X** the number of regions in smaller buildings and it grows to **2X** as the number of regions in a building increases, as seen in Fig. 6. These results show a reasonable amount of added manual work compared to the building's size.

Another key aspect that we need to weigh is the manual effort required to obtain navigation instructions. Generating these instructions is a familiar task, akin to giving someone directions, which is straightforward in daily life. However, as the demand for a greater volume of instructions increases, so does the manual effort involved. As detailed in Section III-C.2, our current strategy involves selecting one navigation instruction per region. Therefore, in a building comprising l regions, we end up with l instructions, with an average path length of 10 meters. This method ensures that each area is included at least once, even though it may not provide the most efficient set of instructions that cover the entire building, as some instructions may overlap in the same area. Our ongoing research is focused on identifying those instructions that cover multiple regions along their paths and using them to drop unneeded instructions.

VI. Conclusion and Future Work

In this work, we presented an alternative method to solve the problem of creating digital navigable map representations for indoor spaces. We proposed using human language navigation instructions which eliminates the need for expertise in map data formats and specialized hardware, so now anyone can do such descriptions. To convert these language-based descriptions to navigable format we utilized LLMs to automatically generate a graph representation of the map where room-like regions are nodes, and edges indicate connections between these regions. With this approach in mind, we defined our task and described the methodology of creating the first dataset for this goal *Text2Map Dataset*. Following that we ran our experiments on OpenAI's GPT-3.5 and GPT-4 and utilized few-shot learning as a method of prompting. We reported our results, which showed that Text2Map achieves a GED of 0.5X the total number of regions in buildings and an Edge Similarity score of up to 0.9, which showed the great potential our method holds. For future work, we plan to refine the use of verbal navigation instructions for graph construction by developing methods to assess and enhance instruction quality, possibly using a language model to synthesize a coherent building description from multiple sequences.

References

[1] J. Chen and K. C. Clarke, "Indoor cartography," *Cartography and Geographic Information Science*, vol. 47, no. 2, pp. 95–109, 2020.

[2] M. A. Shah, B. Raj, and K. A. Harras, "Inferring room semantics using acoustic monitoring," in *IEEE 27th International Workshop on Machine Learning for Signal Processing (MLSP)*. IEEE, 2017.

[3] M. A. Shah, K. A. Harras, and B. Raj, "Sherlock: A crowd-sourced system for automatic tagging of indoor floor plans," in *IEEE 17th International Conference on Mobile Ad Hoc and Sensor Systems (MASS)*. IEEE, 2020.

[4] A. Essameldin, M. N. Hoque, and K. A. Harras, "More than the sum of its things: Resource sharing across iots at the edge," in *IEEE/ACM Symposium on Edge Computing (SEC)*. IEEE, 2020.

[5] A. Saeed, A. Abdelkader, M. Khan, A. Neishaboori, K. A. Harras, and A. Mohamed, "On realistic target coverage by autonomous drones," *ACM Transactions on Sensor Networks (TOSN)*, vol. 15, no. 3, pp. 1–33, 2019.

[6] M. Khan, K. Heurtefeux, A. Mohamed, K. A. Harras, and M. M. Hassan, "Mobile target coverage and tracking on drone-be-gone uav cyber-physical testbed," *IEEE Systems Journal*, vol. 12, no. 4, pp. 3485–3496, 2017.

[7] O. Hashem, K. A. Harras, and M. Youssef, "Deepnar: Robust time-based sub-meter indoor localization using deep learning," in *2020 17th Annual IEEE international conference on sensing, communication, and networking (SECON)*. IEEE, 2020, pp. 1–9.

[8] ——, "Accurate indoor positioning using ieee 802.11 mc round trip time," *Pervasive and Mobile Computing*, vol. 75, p. 101416, 2021.

[9] S. Mostafa, K. A. Harras, and M. Youssef, "Unicellular: An accurate and ubiquitous floor identification system using single cell tower information," in *Proceedings of the 31st ACM International Conference on Advances in Geographic Information Systems (SIGSPACIAL)*, 2023.

[10] "Problems in indoor mapping and modelling," *The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, vol. 40, pp. 63–68, 2013.

[11] H.-G. Ryoo, T. Kim, and K.-J. Li, "Comparison between two ogc standards for indoor space: Citygml and indoorgml," in *Proceedings of the Seventh ACM SIGSPATIAL International Workshop on Indoor Spatial Awareness*, 2015, pp. 1–8.

[12] K.-J. Li, G. Conti, E. Konstantinidis, S. Zlatanova, and P. Bamidis, "Ogc indoorgml: A standard approach for indoor maps," in *Geographical and Fingerprinting Data to Create Systems for Indoor Positioning and Indoor/Outdoor Navigation*. Elsevier, 2019, pp. 187–207.

[13] Y. Deng, H. Ai, Z. Deng, W. Gao, and J. Shang, "An overview of indoor positioning and mapping technology standards," *Standards*, vol. 2, no. 2, pp. 157–183, 2022.

[14] H. Durrant-Whyte and T. Bailey, "Simultaneous localization and mapping: part i," *IEEE robotics & automation magazine*, vol. 13, 2006.

[15] E. Bastianelli, D. D. Bloisi, R. Capobianco, F. Cossu, G. Gemignani, L. Iocchi, and D. Nardi, "On-line semantic mapping," in *2013 16th International Conference on Advanced Robotics (ICAR)*. IEEE, 2013.

[16] N. Hughes, Y. Chang, and L. Carlone, "Hydra: A real-time spatial perception system for 3d scene graph construction and optimization," *Robotics: Science and Systems*, 2022.

[17] R. Bigazzi, L. Baraldi, S. Kousik, R. Cucchiara, M. Pavone, *et al.*, "Mapping high-level semantic regions in indoor environments without object recognition," in *Proceedings of the 2024 IEEE ICRA*, 2024.

[18] N. Sünderhauf, F. Dayoub, S. McMahon, B. Talbot, R. Schulz, P. Corke, G. Wyeth, B. Upcroft, and M. Milford, "Place categorization and semantic mapping on a mobile robot," in *2016 IEEE international conference on robotics and automation (ICRA)*. IEEE, 2016.

[19] Q. Chen, Q. Wu, R. Tang, Y. Wang, S. Wang, and M. Tan, "Intelligent home 3d: Automatic 3d-house design from linguistic descriptions only," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 12 625–12 634.

[20] W. Para, P. Guerrero, T. Kelly, L. J. Guibas, and P. Wonka, "Generative layout modeling using constraint graphs," in *Proceedings of the IEEE/CVF international conference on computer vision*, 2021.

[21] O. Agarwal, H. Ge, S. Shakeri, and R. Al-Rfou, "Knowledge graph based synthetic corpus generation for knowledge-enhanced language model pre-training," in *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, June 2021.

[22] J. Guo, L. Du, H. Liu, M. Zhou, X. He, and S. Han, "Gpt4graph: Can large language models understand graph structured data? an empirical evaluation and benchmarking. arxiv 2023," *arXiv preprint arXiv:2305.15066*.

[23] Z. Chen, L. N. Zheng, C. Lu, J. Yuan, and D. Zhu, "Chatgpt informed graph neural network for stock movement prediction," *arXiv preprint arXiv:2306.03763*, 2023.

[24] E. Perez, D. Kiela, and K. Cho, "True few-shot learning with language models," *Advances in neural information processing systems*, vol. 34, pp. 11 054–11 070, 2021.

[25] Y. Chang, X. Wang, J. Wang, Y. Wu, L. Yang, K. Zhu, H. Chen, X. Yi, C. Wang, Y. Wang, *et al.*, "A survey on evaluation of large language models," *ACM TIST*, 2023.

[26] P. Anderson, Q. Wu, D. Teney, J. Bruce, M. Johnson, N. Sünderhauf, I. Reid, S. Gould, and A. Van Den Hengel, "Vision-and-language navigation: Interpreting visually-grounded navigation instructions in real environments," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 3674–3683.

[27] 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 *2017 International Conference on 3D Vision (3DV)*, 2017, pp. 667–676.

[28] OpenAI, "Gpt-3.5 model documentation," https://platform.openai.com/docs/models/gpt-3-5, 2024.

[29] J. Achiam, S. Adler, S. Agarwal, L. Ahmad, I. Akkaya, F. L. Aleman, D. Almeida, J. Altenschmidt, S. Altman, S. Anadkat, *et al.*, "Gpt-4 technical report," *arXiv preprint arXiv:2303.08774*, 2023.

[30] M. Stauffer, T. Tschachtli, A. Fischer, and K. Riesen, "A survey on applications of bipartite graph edit distance," in *Graph-Based Representations in Pattern Recognition: 11th IAPR-TC-15 International Workshop, GbRPR 2017, Anacapri, Italy, May 16–18, 2017, Proceedings 11*. Springer, 2017, pp. 242–252.

[31] A. Fischer, K. Riesen, and H. Bunke, "Improved quadratic time approximation of graph edit distance by combining hausdorff matching and greedy assignment," *Pattern Recognition Letters*, vol. 87, 2017.

[32] A. Fischer, C. Y. Suen, V. Frinken, K. Riesen, and H. Bunke, "Approximation of graph edit distance based on hausdorff matching," *Pattern Recognition*, vol. 48, no. 2, pp. 331–343, 2015.

[33] K. Riesen, M. Ferrer, and H. Bunke, "Approximate graph edit distance in quadratic time," *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, vol. 17, no. 2, pp. 483–494, 2020.