Generating Landmark Navigation Instructions from Maps as a Graph-to-Text Problem

Raphael Schumann
Computational Linguistics
Heidelberg University, Germany

Stefan Riezler
Computational Linguistics & IWR
Heidelberg University, Germany

{rschuman|riezler}@cl.uni-heidelberg.de

Abstract

Car-focused navigation services are based on turns and distances of named streets, whereas navigation instructions naturally used by humans are centered around physical objects called landmarks. We present a neural model that takes OpenStreetMap representations as input and learns to generate navigation instructions that contain visible and salient landmarks from human natural language instructions. Routes on the map are encoded in a location- and rotation-invariant graph representation that is decoded into natural language instructions. Our work is based on a novel dataset of 7,672 crowd-sourced instances that have been verified by human navigation in Street View. Our evaluation shows that the navigation instructions generated by our system have similar properties as human-generated instructions, and lead to successful human navigation in Street View.

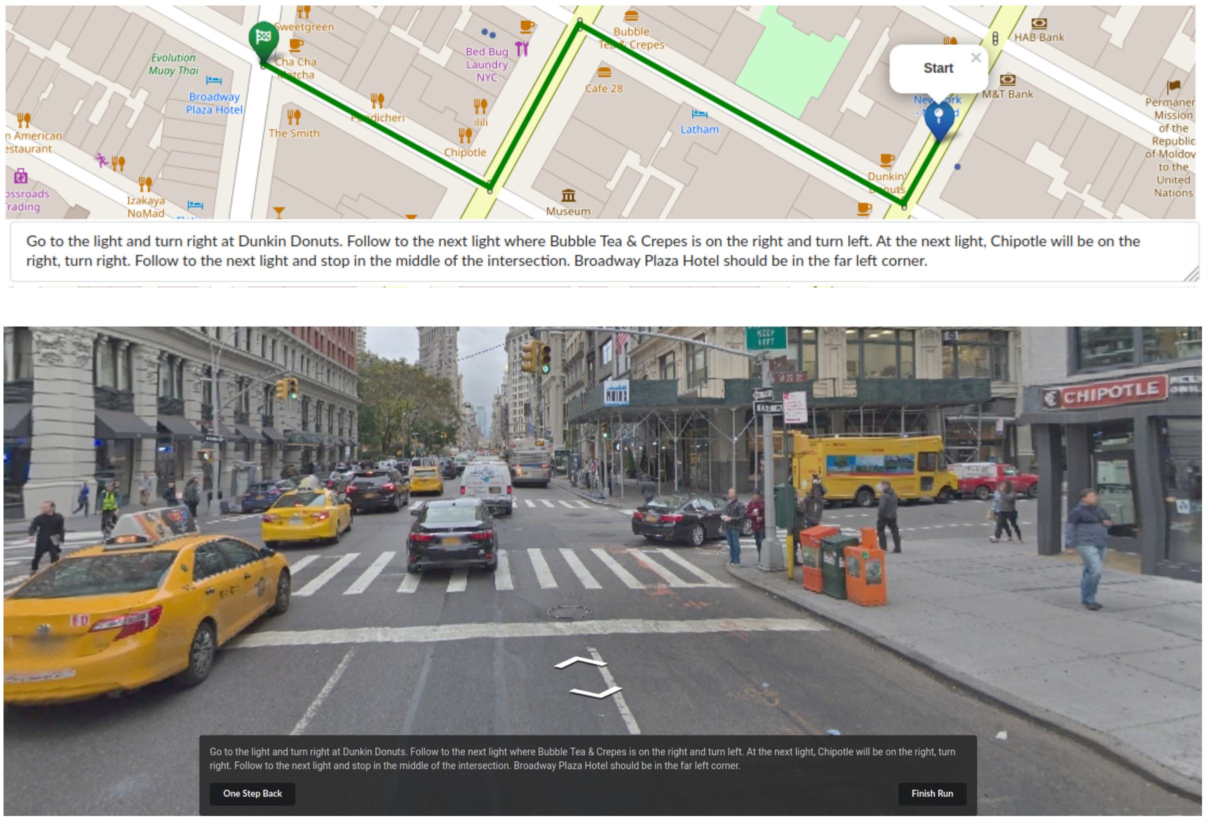
1 Introduction

Current navigation services provided by the automotive industry or by Google Maps generate route instructions based on turns and distances of named streets. In contrast, humans naturally use an efficient mode of navigation based on visible and salient physical objects called landmarks. As shown by Tom and Denis (2004), route instructions based on landmarks are easier processed and memorized by humans. May et al. (2003) recommend that in pedestrian navigation systems, ”landmarks should be used as the primary means of providing directions”. Another navigation scenario where landmarks are useful is if GPS tracking is poor or not available, and if information is inexact regarding distances (e.g., in human estimates) or street names (e.g., for users riding a bicycle). We present a neural model that takes a real-world map representation from OpenStreetMap[1] as input and generates navigation instructions that contain salient landmarks, learned directly from human natural language instructions.

In our framework, routes on the map are learned by discretizing the street layout, connecting street segments with adjacent points of interest, thus encoding visibility of landmarks, and encoding the route and surrounding landmarks in a location- and rotation-invariant graph. Based on crowd-sourced natural language instructions for such map representations, a graph-to-text mapping is learned that decodes graph representations into natural language route instructions that contain salient landmarks. Our work is accompanied by a dataset of 7,672 instances of routes in OpenStreetMap and corresponding crowd-sourced natural language instructions. The navigation instructions were generated by workers on the basis of maps including all points of interest, but no street names. They were verified by different workers who followed the navigation instructions on Google Street View[2].

Experimental results on randomly sampled test routes show that our graph-to-text model produces landmarks with the same frequency found in human reference instructions. Furthermore, the time-normalized success rate of human workers finding the correct goal location on Street View is 0.664. Since these routes can have a partial overlap with routes in the training set, we further performed an evaluation on completely unseen routes. The rate of produced landmarks drops slightly compared to human references, and the time-normalized success rate also drops slightly to 0.629. While there is still room for improvement, our results showcase a promising direction of research, with a wide potential of applications in various existing map

[1] www.openstreetmap.org
[2] www.google.com/streetview


Go to the light and turn right at Dunkin Donuts. Follow to the next light where Bubble Tea & Crepes is on the right and turn left. At the next light, Chipotle will be on the right, turn right. Follow to the next light and stop in the middle of the intersection. Broadway Plaza Hotel should be in the far left corner.

Go to the light and turn right at Dunkin Donuts. Follow to the next light where Bubble Tea & Crepes is on the right and turn left. At the next light, Chipotle will be on the right, turn right. Follow to the next light and stop in the middle of the intersection. Broadway Plaza Hotel should be in the far left corner.

Figure 1: The data collection is split into two tasks. In the navigation instructions task (top) annotators see a rendered map and write instructions to follow the route. The navigation run task (bottom) is used to validate navigation instructions. A different annotator tries to find the goal location in Street View.

applications and navigation systems.

The main contributions of this paper are:

- We collect and publish a large scale dataset of natural language landmark navigation instructions that are validated by human navigation runs in Street View.
- We present a method to represent geospatial routes as a graph and propose an appropriate graph-to-text architecture that learns to generate navigation instructions from real-world data.

2 Related Work and Datasets

Mirowski et al. (2018) published a subset of Street View covering parts of New York City and Pittsburgh. Street View is a navigable environment that is build from connected real-world 360°panoramas. This data is used by Hermann et al. (2020) to train a visual agent to follow turn-by-turn instructions generated by Google Maps API. Chen et al. (2019) published a Street View dataset[3] with more recent and higher resolution panorama images that covers the lower half of Manhattan. They further introduce the Touchdown task that has the goal to navigate Street View in order to find a hidden teddy bear. The data for that task is obtained from annotation workers that follow a predefined route in Street View and write down navigation instructions along the way. A central difference between Touchdown and our dataset is the annotation modality: Touchdown annotators use panorama images along the route, while our instruction writers only see the rendered route on a map. See Section 4.3 for a more detailed discussion.

Our work puts the task of natural language navigation upside down by learning to generate human-like navigation instructions from real-world map data instead of training an agent to follow human generated instructions. Prior work in this area has used rule-based systems to identify landmarks (Rousell and Zipf, 2017) or to generate landmark-based navigation instructions (Dräger and Koller, 2012; Cercas Curry et al., 2015). Despite having all points of interest on the map available, our approach learns to verbalize only those points of interest that have been deemed salient by inclusion in a human navigation instruction. Previous approaches that learn navigation instructions from data have

[3]www.streetlearn.cc

been confined to simplified grid-based representations of maps for restricted indoor environments (Daniele et al., 2017). de Vries et al. (2018) tackles the problem in a more sophisticated outdoor environment but the model fails to verbalize useful instructions when conditioned on more than one possible landmark. Other work generates navigation instructions from indoor panoramas along a path but provides no explicit evaluation like human navigation success. They rather use the instructions to augment the training routes for a vision and language navigation agent (Fried et al., 2018).

3 Task

The task addressed in our work is that of automatically generating Natural Language Landmark Navigation Instructions (NLLNI) from real-world open-source geographical data from OpenStreetMap. The instructions are generated *a priori* (Janarthanam et al., 2012) for the whole route. Training data for NLLNI was generated by human crowdsourcing workers who were given a route on an OpenStreetMap rendering of lower Manhattan, with the goal of producing a succinct natural language instruction that does not use street names or exact distances, but rather is based on landmarks. Landmarks had to be visible on the map and included, e.g., churches, cinemas, banks, shops, and public amenities such as parks or parking lots. Each generated navigation instruction was validated by another human crowdsourcing worker who had to reach the goal location by following the instruction on Google Street View.

NLLNI outputs are distinctively different from navigation instructions produced by OpenRouteService, Google Maps, or car navigation systems. While these systems rely on stable GPS signals such that the current location along a grid of streets can be tracked exactly, we aim at use cases where GPS tracking is not available, and knowledge of distances or street names is inexact, for example, pedestrians, cyclists, or users of public transportation. The mode of NLLNI is modeled after human navigation instructions that are naturally based on a small number of distinctive and visible landmarks in order to be memorizable while still being informative enough to reach the goal. A further advantage of NLLNI is that they are based on map inputs which are more widely available and less time dependent than Street View images.

4 Data Collection

Because there is no large scale dataset for NLLNI that is generated from map information only, we collect data via crowdsourcing. The annotator is shown a route on the map and writes navigation instructions based on that information (Figure 1, top). We take the approach of Chen et al. (2019) and determine correctness of navigation instructions by showing them to other annotators that try to reach the goal location in Street View (Figure 1, bottom).

4.1 Resources and Preparation

We use the static Street View dataset provided by Chen et al. (2019). This allows us to make the experiments in this work replicable. Because the panorama pictures were taken at the end of 2017, we export an OpenStreetMap extract of Manhattan from that time. OpenStreetMap (OSM) is an open source collection of geodata that can be used to render maps of the world. It features detailed street layouts and annotations for points of interest (POI) like amenities, infrastructure or land use[4]. We discretize the street layout by creating a node every ten meters along the roads. The resulting structure is further referenced to as the OSM graph with nodes consisting of street segments. Based on that graph, we sample routes of length between 35 and 45 nodes. A route is the shortest path between its start and end node. It includes a minimum of three intersections (i.e., a node with more than two edges) and ends in proximity to a POI. We further assure that it is possible to follow the route in Street View by verifying that a corresponding subgraph exists in the Street View graph.

4.2 Crowdsourcing

We use Amazon Mechanical Turk (AMT)[5] to acquire annotators. Before working on the actual tasks, workers were required to pass a tutorial and qualification test. The tutorial introduces the tasks, teaches basic mechanics of Street View and explains meaning of map icons. A feature of AMT and additional IP address[6] lookup ensures that annotators are located in the United States. This increases the probability of working with native English speakers and people familiar with US street environments. We paid \$0.35 per navigation instructions task and \$0.20 for the navigation run

[4] `openstreetmap.org/wiki/Map_Features`
[5] `www.mturk.com`
[6] IP addresses were not saved and are not part of the dataset.

Dataset	#Instructions	Environment	Data Source	#Nodes	Avg. Length	Vocabulary	Avg. Tokens
Talk the Walk	786	gridworld	3D rendering	100	6.8	587	34.5
Room-to-Room	21,567	indoor	panoramas	10,800	6.0	3,156	29.0
Touchdown	9,326	outdoor	panoramas	29,641	35.2	4,999	89.6
Talk2Nav	10,714	outdoor	panoramas and map	21,233	40.0	5,240	68.8
Room-X-Room	126,069	indoor	panoramas	10,800	7.0	388K	78.0
map2seq	**7,672**	**outdoor**	**map**	**29,641**	**40.0**	**3,826**	**55.1**

Table 1: Overview of natural language navigation instructions datasets. The instructions in our dataset rely solely on information present in OpenStreetMap. **Dataset**: Talk the Walk (MacMahon et al., 2006); Room-to-Room (Anderson et al., 2018b); Touchdown (Chen et al., 2019); Talk2Nav (Vasudevan et al., 2020); Room-X-Room (Ku et al., 2020); map2seq (this work). **#Instructions**: Number of instructions in the dataset. **Environment**: Type of navigation environment. **Data Source**: Type of information the annotator uses to write the navigation instructions. **#Nodes**: Number of nodes in the discretized environment. **Avg. Length**: Average number of nodes per route. **Vocabulary**: Number of unique tokens in the instructions. **Avg. Tokens**: Number of tokens per route instruction.

Phenomenon	R-to-R		Touchdown		map2seq		Example
	c	μ	c	μ	c	μ	
Reference to unique entity	25	3.7	25	9.2	25	6.3	... turn right where **Dough Boys** is on the corner ...
Coreference	8	0.5	15	1.1	8	0.5	... is a bar, Landmark tavern, stop outside of **it** ...
Comparison	1	0.0	3	0.1	0	0.0	... there are two lefts, **take the one that is not sharp** ...
Sequencing	4	0.2	21	1.6	24	1.8	... continue straight at the **next** intersection ...
Count	4	0.2	9	0.4	11	0.6	... go through the next **two** lights ...
Allocentric spatial relation	5	0.2	17	1.2	9	0.5	... go through the next **light with Citibank at the corner.** ...
Egocentric spatial relation	20	1.2	23	3.6	25	3.2	... at the end of the park **on your right**...
Imperative	25	4.0	25	5.2	25	5.3	... **head** down the block and **go** through the double lights ...
Direction	22	2.8	24	3.7	25	3.5	... head **straight** to the light and make a **right** ...
Temporal condition	7	0.4	21	1.9	7	0.3	... go straight **until you come** to the end of a garden area ...
State verification	2	0.1	18	1.5	12	0.6	... **you should see** bike rentals on your right ...

Table 2: Linguistic analysis of 25 randomly sampled navigation instructions. Numbers for Room-to-Room (Anderson et al., 2018b) and Touchdown (Chen et al., 2019) taken from the latter. c is the number of instructions out of the 25 which contain the phenomenon at least once. μ is the mean number of times each phenomenon occurs.

task. Furthermore, we paid a bonus of \$0.15 for successfully reaching the goal location and \$0.25 for validated navigation instructions. The amounts were chosen on the basis of \$10/hour. The annotation procedure involved two phases. First, an annotator wrote navigation instructions for a given route. Afterwards, a different annotator used the instructions to navigate to the goal location. If one of two annotators did so successfully, the navigation instructions were considered valid.

Navigation Instructions Task As shown in Figure 1 (top), the annotator sees a route on a map which is rendered without street names. Workers were told to write navigation instructions as if ”a tourist is asking for directions in a neighborhood you are familiar with” and to ”mention landmarks to support orientation”. The navigation instructions were written in a text box below the map which is limited to 330 characters.

Navigation Run Task Figure 1 (bottom) shows the Street View interface with navigation instructions faded-in at the bottom. It is possible to look around 360° and movement is controlled by the white arrows. In addition there is a button on the bottom left to backtrack which proved to be very helpful. The initial position is the start of the route facing in the correct direction. The annotators finish the navigation run with the bottom right button either when they think the goal location is reached or if they are lost. The task is successful if the annotator stops the run within a 25 meter radius around the goal location.

4.3 Dataset

The data collection resulted in **7,672 navigation instructions that were manually validated in Street View**. For additional 1,059 instructions, the validation failed, which amounts to a validation rate of 88%. Of the validated instructions, 1,033 required a second try in the navigation run task. On average, instructions are 257 characters long, with a minimum length of 110, and a maximum of 330 characters. We release the segmented OSM graph, the routes in that graph paired with the collected navigation instructions, and the data split used in

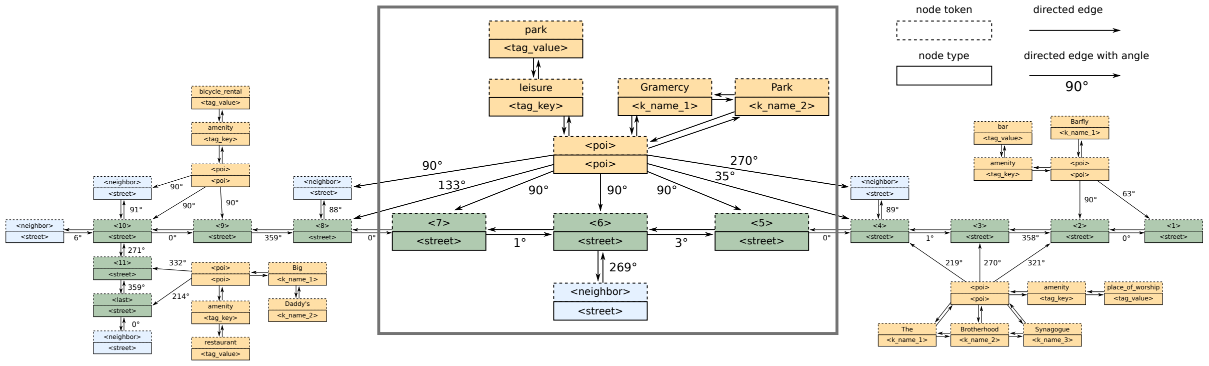


Figure 2: Graph representation of the route in Figure 3. The framed middle part is magnified for readability. Some nodes are left out for sake of clear visualization. Also, node colors are for visualization only and not encoded in the graph. Green nodes are part of the route. Blue nodes are neighboring street segments. Orange nodes belong to OSM points of interest. Angles are relative to route direction and start clockwise at 0° which is facing forward.

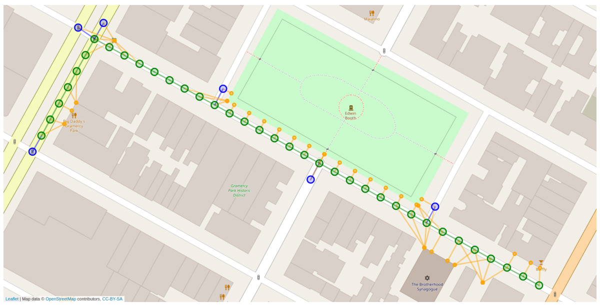


Figure 3: Route rendered on the map with street segments and landmark visibility.

our experiments[7]. Table 1 gives a comparison of different datasets with natural language landmark navigation instructions. Our dataset is the only one that uses only map information to generate navigation instructions. The advantage of relying solely on map data is the global availability and longevity of the encoded features. In contrast, navigation instructions written from Street View include temporary features like construction utilities, street advertisements, or passing vehicles. Table 2 shows a qualitative linguistic analysis of the navigation instructions of different datasets. In general, navigation instructions are driven by giving directions in imperative formulation while referencing to entities along the route. In contrast to the Touchdown task where including store names was prohibited, the entities in our instructions are often referenced to by their name. Although the instruction writers in our setting did not see the route in first person perspective, objects are vastly referenced to in egocentric manner (egocentric with respect to the navigating agent). This is because the annotator knows the starting direction and can infer the facing direction for the rest of the route. Because the initial facing direction in Touchdown is random, the first part of their instructions is about rotating the agent. This explains the higher number of occurrences of the state verification phenomenon. In our dataset, state verification is usually used to ensure the correct stopping position. The different setting of data collection is also reflected by the temporal condition phenomenon. Annotators of Touchdown write down instructions while navigating Street View and thus experience the temporal component first hand, while our annotators have a time independent look at the route.

5 Method

The underlying OSM geodata of the rendered map is an XML tree of nodes located in the latitude-longitude coordinate system. The nodes are composed into ways and polygons[8]. These elements in connection with their annotations are used to render the visual map. In the next subsection we propose our approach to represent a route and its surrounding map features as a graph that includes all necessary information for generating landmark navigation instructions. The second subsection describes the neural graph-to-text architecture that is trained to learn inductive representations of the individual route graphs and to decode navigation instructions from them.

5.1 Map-to-Graph Representation

The basis of the graph for a single route is the OSM subgraph (Section 4.1) that includes the ac-

[7] www.cl.uni-heidelberg.de/statnlpgroup/map2seq/

[8] www.openstreetmap.org/wiki/Elements

tual route nodes. Further, neighboring street segment nodes are added. This is depicted in Figure 3 as green and blue circles, respectively. In order to decide on the visibility of the POIs, we employ a technique similar to that of Rousell and Zipf (2017). For each street segment, the POIs in a radius of 30 meters are identified. If a line drawn between the street segment and the POI is not interrupted by a building polygon, the POI is considered visible from that particular street segment. If the POI itself is (inside) a polygon, then the line is drawn to the closest point on the POI polygon. The orange circles in Figure 3 show the results of the visibility check and how they naturally fit into the graph structure. Each point of interest in OSM has one or more tags in the form of key and value pairs. They store properties like type or name. Note that we only determine the geometric visibility of the POIs and do not incorporate any hand-crafted salience scores as to what would be a good landmark. Instead, saliency of a landmark is implicitly learned from natural language verbalization of the POI in the human-generated instruction.

An example graph representation of the route in Figure 3 is given in Figure 2. Formally, a route representation is a directed graph $\mathcal{G} = (\mathbb{V}, \mathbb{E})$, where $\mathbb{V}$ denotes the set of nodes and $\mathbb{E}$ the set of edges. A node v consists of a node type v^t and a node token v^w. There are V^t node types and V^w node tokens. Street segments are of type <*street*>. A point of interest has the node type <*poi*>. An OSM tag key has the node type <*tag_key*> and an OSM tag value has the node type <*tag_value*>. The node token further specifies nodes in the graph. Street segments that belong to the route have a node token <*P*> according to their sequential position P. The last route segment has the special token <*last*>. Other street segment nodes have the <*neighbor*> token. The actual key and value literals of an OSM tag are the node tokens of the respective node. The OSM name tag is split into multiple nodes with type <*k_name_N*> where N is the word position and the node token is the word at that position.

All adjacent street segment nodes are connected with an edge in both directions. If a POI is visible from a particular street segment, there is an edge from the corresponding POI node to that street segment node. Each POI node is connected with their tag key nodes. A tag value node is connected to its corresponding tag key node. The name tag nodes of the same POI are connected with each other. Some edges have a geometric interpretation. This is true for edges connecting a street segment with either a POI or with another street segment. These edges $(u, v) \in \mathbb{E}^A, \mathbb{E}^A \subset \mathbb{E}$ have a label attached. The label $ang(u, v)$ is the binned angle between the nodes relative to route direction. The continuous angle $[0°, 360°)$ is assigned to one of 12 bins. Each bin covers 30° with the first bin starting at 345°. The geometric distance between nodes is not modeled explicitly because street segments are equidistant and POI visibility is determined with a maximum distance. The proposed representation of a route and its surroundings as a directed graph with partially geometric edges is location- and rotation-invariant, which greatly benefits generalization.

5.2 Graph-to-Text Architecture

By representing a route as a graph, we can frame the generation of NLLNI from maps as a graph-to-text problem. The encoder learns a neural representation of the input graph and the sequence decoder generates the corresponding text. The architecture follows the Transformer (Vaswani et al., 2017) but uses graph attentional layers (Veličković et al., 2018) in the encoder. Graph attention injects the graph structure by masking (multi-head) self-attention to only attend to nodes that are first-order neighbors in the input graph. The geometric relations between some nodes are treated as edge labels which are modeled by distinct feature transformation matrices during node aggregation (Schlichtkrull et al., 2018).

The input to a layer of the encoder is a set of node representations, $\mathbf{x} = \{\boldsymbol{x}_1, \boldsymbol{x}_2, \dots, \boldsymbol{x}_N\}, \boldsymbol{x}_i \in \mathbb{R}^{d_m}$, where N is the number of nodes and d_m is the model size. Each layer $l : \mathbb{R}^{d_m} \rightarrow \mathbb{R}^{d_m}$ takes $\mathbf{x}$ and produces new node representations $\mathbf{x}'$. The input to the first layer is constructed from the concatenation of type and token embedding: $\boldsymbol{x}_i = ReLU(\boldsymbol{W}^F[\boldsymbol{E}^T_{v_i^t} || \boldsymbol{E}^W_{v_i^w}])$ where $\boldsymbol{W}^F \in \mathbb{R}^{2d_m \times d_m}$ is a weight matrix, $\boldsymbol{E}^T \in \mathbb{R}^{d_m}$ and $\boldsymbol{E}^W \in \mathbb{R}^{d_m}$ are embedding matrices for node types and node tokens, respectively.

The output of a single graph attention head is the weighted sum of neighboring node representations:

$$\bar{\boldsymbol{x}}_i = \sum_{j|(v_j, v_i) \in \mathbb{E}} \alpha_{ij}(\boldsymbol{W}^U_{r(i,j)} \boldsymbol{x}_j) \tag{1}$$

The weight coefficient is computed as $\alpha_{ij} = \mathrm{softmax}_j(e_{ij}) = \frac{\exp{(e_{ij})}}{\sum_{k|(v_k, v_i) \in \mathbb{E}} \exp{(e_{ik})}}$ where e_{ij}

	BLEU	Len.	Landm.	SDTW	SR	SNT
	200 instances test set					
reference	-	53.5	2.76	.728	.855	**.878**
rule based	0.71	53.1	12.44	.405	.460	**.455**
seq2seq	13.12	52.9	1.95	.139	.160	**.206**
graph2text	18.60	52.6	2.41	.475	.540	**.676**
g2t+pretrain	18.81	52.5	2.44	.471	.540	**.537**
	700 instances test set					
reference	-	53.5	2.72	.726	.861	**.830**
g2t+pretrain	17.39	53.0	2.41	.475	.551	**.664**

Table 3: Evaluation of navigation instructions produced by models and human reference on **partially seen** test routes. Evaluation metrics are explained in Section 6.3.

	BLEU	Len.	Landm.	SDTW	SR	SNT
	200 instances test set					
reference	-	57.5	2.68	.725	.824	**.791**
rule based	0.67	52.3	10.96	.472	.525	**.512**
seq2seq	11.12	51.8	1.58	.074	.100	**.137**
graph2text	14.07	50.5	1.74	.344	.400	**.534**
g2t+pretrain	15.64	50.3	2.33	.367	.429	**.530**
	700 instances test set					
reference	-	54.2	2.69	.727	.843	**.807**
g2t+pretrain	16.27	53.2	2.30	.407	.473	**.629**

Table 4: Evaluation of navigation instructions produced by models and human reference on **unseen** test routes.

measures the compatibility of two node representations:

$$e_{ij} = LeakyReLU(\boldsymbol{a}^T[\boldsymbol{W}^V\boldsymbol{x}_i||\boldsymbol{W}^U_{r(i,j)}\boldsymbol{x}_j]) \quad (2)$$

where $\boldsymbol{a} \in \mathbb{R}^{2d_h}$, $\boldsymbol{W}^V \in \mathbb{R}^{d_m \times d_h}$, $d_h = d_m/h$ is the attention head dimension and h is the number of heads. In the case of a geometric relation between nodes, the weight matrix $\boldsymbol{W}^U_{r(i,j)} \in \mathbb{R}^{d_m \times d_h}$ is selected according to the angle label between the nodes: $r(i,j) = ang(u_i, u_j)$, otherwise $r(i,j) = unlabeled$. The output of each head is concatenated and after a skip connection forwarded to the next encoder layer. The encoder layer is applied L times and the final node representations $\mathbf{x}^*$ are used in the decoder context attention mechanism. Thus, no modification of the Transformer decoder is necessary and L decoder layers are used. Further, the decoder can copy node tokens from the input into the output sequence (See et al., 2017).

The described architecture is able to model all aspects of the input graph. Graph attention models directed edges. Edge labels model the geometric relation between nodes. Heterogeneous nodes are represented by their type embedding and token embedding. The sequentiality of the route is encoded by tokens (<*1*>, <*2*>, ...) of the respective nodes. This is analogous to absolute position embeddings which provide word order information for text encoding (Vaswani et al., 2017; Devlin et al., 2019).

6 Experiments

6.1 Baselines

We consider two baselines. A **rule based** system that uses a single heuristic to construct instructions by stringing together all POIs and intersections along the route, and following each intersection by the turning direction. Similar, POIs are followed by 'left' or 'right' depending on which side of the street they appear. The end of the route is signaled by the 'stop' token. The second baseline is a **seq2seq** (sequence-to-sequence) model that is trained on pairs of rule based navigation instructions and crowdsourced instructions. The seq2seq model follows the Transformer architecture (Vaswani et al., 2017) with copy mechanism and is trained with the same hyperparameters as the graph-to-text model. Examples are given in Figure 4.

6.2 Experimental Setup

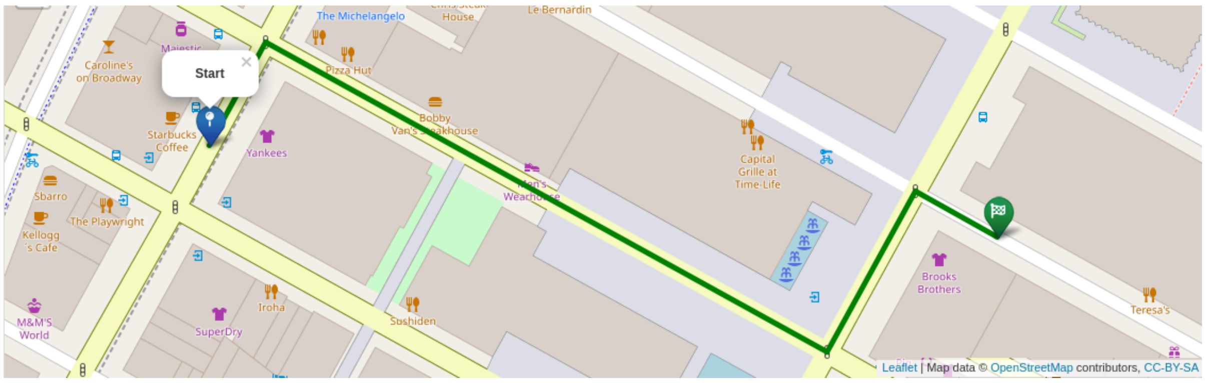
We construct a graph for each route as described above. On average there are 144 nodes in a graph and 3.4 edges per node. There are 8 different node types and a vocabulary of 3,791 node tokens. The hyperparameters for the graph-to-text architecture are set as follows: The embedding and hidden size is set to 256. We use 6 encoder and decoder layers with 8 attention heads. Cross entropy loss is optimized by Adam (Kingma and Ba, 2015) with a learning rate of 0.5 and batch size of 12. The embedding matrix for node tokens and output tokens is shared. Additionally we experiment with pretraining the graph-to-text model with above mentioned rule based instructions as target. This teaches the model sequentiality of route nodes and basic interpretation of the angle labels. We generate 20k instances for pretraining and further fine tune on the human generated instances. Both models and the seq2seq baseline are trained on 5,667 instances of our dataset. The best weights for each model are selected by token accuracy based early stopping on the 605 development instances.

6.3 Evaluation Metrics

BLEU is calculated with SacreBLEU (Post, 2018) on lower-cased and tokenized text.

Length is the average length in number of tokens.

Landmarks is the number of landmark occur-



reference: *At the light with Fridays on the corner, turn right. Continue down the long street to the next light with Nine West on the right corner, then turn left. Go to the next light with Brooks Brothers on the right corner, then turn right and stop.*
rule based: *Starbucks Coffee left subway entrance right Best Buy Mobile left Yankees right bus stop left bus stop left light right The Michelangelo left TGI Fridays left Pizza Hut left Bobby Van 's left park right Men 's Wearhouse left fountain left fountain left subway entrance left light left Nine West right Rockefeller Center left subway entrance right Brooks Brothers right light right stop*
seq2seq: *Go straight to the light and make a left. Go straight to the next light and make a left. Go straight to the light and make a right. Stop one step after turning with Brooks Brothers to your right.*
graph2text: *Walk to the light with TGI Fridays on the corner and turn right. Walk down the long block to the next light with Nine West on the left corner, then turn left. Walk to the next light with Brooks Brothers on the far right corner, then turn right.*
g2t+pretrain: *Turn right at the first set of lights with TGI Fridays on the left corner. Pass a park on the right and turn left at the lights. Pass the fountain on the right and turn right at the lights. Take two steps and stop. Brooks Brothers is on the right corner.*

Figure 4: Route from partially seen test set paired with instructions generated by different systems.

rences per instance. Occurrences are identified by token overlap between navigation text and tag values of POIs along the route. E.g., landmarks in the instructions in Figure 1 are: *Dunkin Donuts, Bubble Tea & Crepes, Chipotle, Broadway Hotel.*

SDTW is success weighted by normalized Dynamic Time Warping (Ilharco et al., 2019). Distance between two nodes is defined as meters along the shortest path between the two nodes and threshold distance is 25 meters.

SR is the first try success rate in the navigation run task. Success is achieved if the human navigator stops within a radius of 25 meters around the goal.

SNT is success weighted by navigation time: $\frac{1}{N}\sum_{i=1}^{N} S_i \frac{\bar{t}_i}{t_i}$, where S_i is a binary success indicator that is 1 if the annotator stops within a 25 meter radius around the goal. t_i is the time until the navigation run is finished. We empirically estimate the expected navigation time $\bar{t}_i$ as 1.3 seconds[9] per node in the route. This estimation ranges from 45.5 seconds for routes with 35 nodes to 58.5 seconds for routes with 45 nodes. SNT is inspired by SPL (Anderson et al., 2018a) but considers trajectory time instead of trajectory length.

6.4 Experimental Results and Analysis

Results of our experimental evaluation are shown in Table 3 and 4. We evaluate on unseen data, i.e., routes without any overlap with routes in the training set, and on partially seen data, i.e., routes randomly sampled from the training area with partial overlaps.[10] For the baseline models we perform the human evaluation on a 200 instances subset of the full 700 instances test set.

On the partially seen test set with 200 instances, our proposed graph-to-text models outperform the baseline models in terms of the success based metrics. In the unseen setup, the rule based baseline achieves a better success rate, but falls short when success is weighted by navigation time. This result shows that the instructions generated by the rule based system are exact by including all possible landmarks, but obviously do not resemble natural language and high evaluation time suggests that they are hard to read. Despite moderate BLEU scores and reasonable amount of produced landmarks, the seq2seq baseline fails to generate useful navigation instructions. The pretrained graph-to-text model performs better than its plain counterpart in the unseen setup. It produces more correct landmarks and higher success rates. In the extended evaluation the pretrained graph-to-text model is compared with the reference on 700 instances in each test set. Under the central evaluation metric of success normalized by time (SNT), our model reaches .664 and .629 on partially seen and unseen test data, respectively.

An example output for each system together with the input map is shown in Figure 4. The rule based instruction is complete, but ignores saliency

[9]Average over all successful navigation runs in the dataset.

[10]The data split is shown in the Appendix.

of landmarks and is hard to read. The seq2seq baseline generates a navigation instruction that sounds human-like and also includes salient landmarks found on the map. However, the directions are incorrect in this example. The graph-to-text based models get the directions right and produce fluent natural language sentences. They include landmarks at the correct sequential position. A further qualitative evaluation of instructions generated by the graph-to-text models is given in the Appendix.

7 Conclusion

We presented a dataset and suitable graph-to-text architecture to generate landmark navigation instructions in natural language from OpenStreetMap geographical data. Our neural model includes novel aspects such as a graphical representation of a route using angle labels. Our dataset consists of a few thousand navigation instructions that are verified for successful human navigation. The dataset is large enough to train a neural model to produce navigation instructions that are very similar in several aspects to human-generated instructions on partially seen test data. However, performance naturally drops on unseen data including new types of landmarks in new combinations.

Acknowledgments

We would like to thank Christian Buck and Massimiliano Ciaramita for initial fruitful discussions about this work. The research reported in this paper was supported by a Google Focused Research Award on "Learning to Negotiate Answers in Multi-Pass Semantic Parsing".

References

Peter Anderson, Angel X. Chang, Devendra Singh Chaplot, Alexey Dosovitskiy, Saurabh Gupta, Vladlen Koltun, Jana Kosecka, Jitendra Malik, Roozbeh Mottaghi, Manolis Savva, and Amir Roshan Zamir. 2018a. On evaluation of embodied navigation agents. *CoRR*, abs/1807.06757.

Peter Anderson, Qi Wu, Damien Teney, Jake Bruce, Mark Johnson, Niko Sünderhauf, Ian Reid, Stephen Gould, and Anton van den Hengel. 2018b. Vision-and-language navigation: Interpreting visually-grounded navigation instructions in real environments. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Amanda Cercas Curry, Dimitra Gkatzia, and Verena Rieser. 2015. Generating and evaluating landmark-based navigation instructions in virtual environments. In *Proceedings of the 15th European Workshop on Natural Language Generation (ENLG)*, pages 90–94, Brighton, UK. Association for Computational Linguistics.

H. Chen, A. Suhr, D. Misra, N. Snavely, and Y. Artzi. 2019. Touchdown: Natural language navigation and spatial reasoning in visual street environments. In *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 12530–12539.

Andrea F. Daniele, Mohit Bansal, and Matthew R. Walter. 2017. Navigational instruction generation as inverse reinforcement learning with neural machine translation. *2017 12th ACM/IEEE International Conference on Human-Robot Interaction (HRI*, pages 109–118.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Markus Dräger and Alexander Koller. 2012. Generation of landmark-based navigation instructions from open-source data. In *Proceedings of the 13th Conference of the European Chapter of the Association for Computational Linguistics*, pages 757–766, Avignon, France. Association for Computational Linguistics.

Daniel Fried, Ronghang Hu, Volkan Cirik, Anna Rohrbach, Jacob Andreas, Louis-Philippe Morency, Taylor Berg-Kirkpatrick, Kate Saenko, Dan Klein, and Trevor Darrell. 2018. Speaker-follower models for vision-and-language navigation. In *Advances in Neural Information Processing Systems*, volume 31, pages 3314–3325. Curran Associates, Inc.

Karl Moritz Hermann, Mateusz Malinowski, Piotr Mirowski, Andras Banki-Horvath, Keith Anderson, and Raia Hadsell. 2020. Learning to follow directions in street view. *Proceedings of the AAAI Conference on Artificial Intelligence*, 34(07):11773–11781.

Gabriel Ilharco, Vihan Jain, Alexander Ku, Eugene Ie, and Jason Baldridge. 2019. Effective and general evaluation for instruction conditioned navigation using dynamic time warping. *NeurIPS Visually Grounded Interaction and Language Workshop*.

Srinivasan Janarthanam, Oliver Lemon, and Xingkun Liu. 2012. A web-based evaluation framework for spatial instruction-giving systems. In *Proceedings of the ACL 2012 System Demonstrations*, pages 49–54, Jeju Island, Korea. Association for Computational Linguistics.

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *ICLR (Poster)*.

Alexander Ku, Peter Anderson, Roma Patel, Eugene Ie, and Jason Baldridge. 2020. Room-Across-Room: Multilingual vision-and-language navigation with dense spatiotemporal grounding. In *Conference on Empirical Methods for Natural Language Processing (EMNLP)*.

Matt MacMahon, Brian Stankiewicz, and Benjamin Kuipers. 2006. Walk the talk: Connecting language, knowledge, and action in route instructions. In *Proceedings of the 21st National Conference on Artificial Intelligence - Volume 2*, AAAI'06, page 1475–1482. AAAI Press.

Andrew J. May, Tracy Ross, Steven H. Bayer, and Mikko J. Tarkiainen. 2003. Pedestrian navigation aids: information requirements and design implications. *Personal and Ubiquitous Computing*, 7(6):331–338.

Piotr Mirowski, Matt Grimes, Mateusz Malinowski, Karl Moritz Hermann, Keith Anderson, Denis Teplyashin, Karen Simonyan, koray kavukcuoglu, Andrew Zisserman, and Raia Hadsell. 2018. Learning to navigate in cities without a map. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems 31*, pages 2419–2430. Curran Associates, Inc.

Matt Post. 2018. A call for clarity in reporting BLEU scores. In *Proceedings of the Third Conference on Machine Translation: Research Papers*, pages 186–191, Belgium, Brussels. Association for Computational Linguistics.

Adam Rousell and Alexander Zipf. 2017. Towards a landmark-based pedestrian navigation service using osm data. *ISPRS International Journal of Geo-Information*, 6(3):64.

Michael Schlichtkrull, Thomas N. Kipf, Peter Bloem, Rianne van den Berg, Ivan Titov, and Max Welling. 2018. Modeling relational data with graph convolutional networks. In *The Semantic Web*, pages 593–607, Cham. Springer International Publishing.

Abigail See, Peter J. Liu, and Christopher D. Manning. 2017. Get to the point: Summarization with pointer-generator networks. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1073–1083, Vancouver, Canada. Association for Computational Linguistics.

Ariane Tom and Michel Denis. 2004. Language and spatial cognition: comparing the roles of landmarks and street names in route instructions. *Applied Cognitive Psychology*, 18(9):1213–1230.

Arun Balajee Vasudevan, Dengxin Dai, and Luc Van Gool. 2020. Talk2nav: Long-range vision-and-language navigation with dual attention and spatial memory.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 5998–6008. Curran Associates, Inc.

Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. 2018. Graph attention networks. In *International Conference on Learning Representations*.

Harm de Vries, Kurt Shuster, Dhruv Batra, Devi Parikh, Jason Weston, and Douwe Kiela. 2018. Talk the walk: Navigating new york city through grounded dialogue. *CoRR*, abs/1807.03367.

Appendices

A Dataset Split

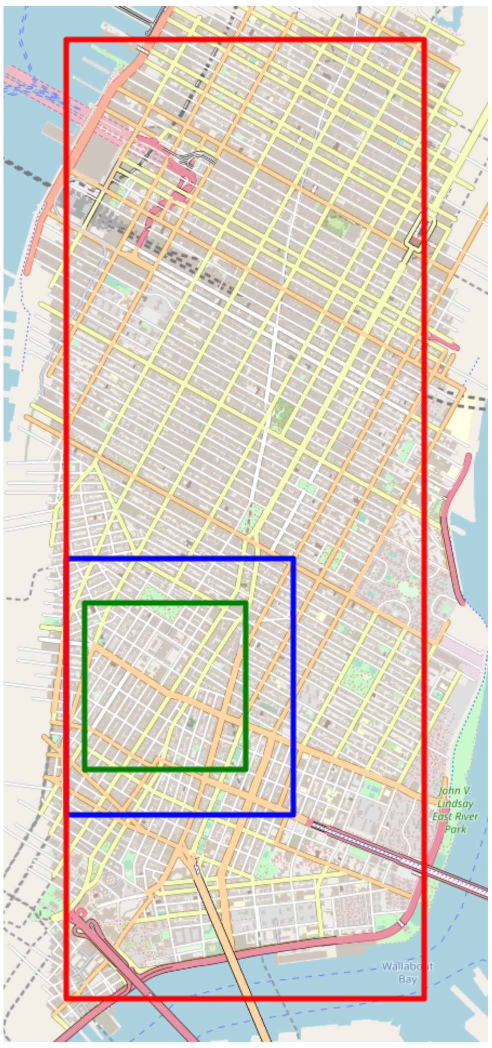


Figure 5: Dataset splits

All 700 routes that are exclusively in the green rectangle are in the unseen test set. All 605 routes that cross the green border are in the development set. None of those development set routes extend further than the blue rectangle. The training set consists of routes within the red rectangle but outside of the green rectangle. The partially seen test set consists of 700 randomly sampled routes from the training set (and removed from the training set). Partially seen means that subsequences of those routes can be present in the training set.

B Evaluation Navigation Success Rate Analysis

We analyze the navigation success rate with respect to properties of the corresponding routes. Figure 6

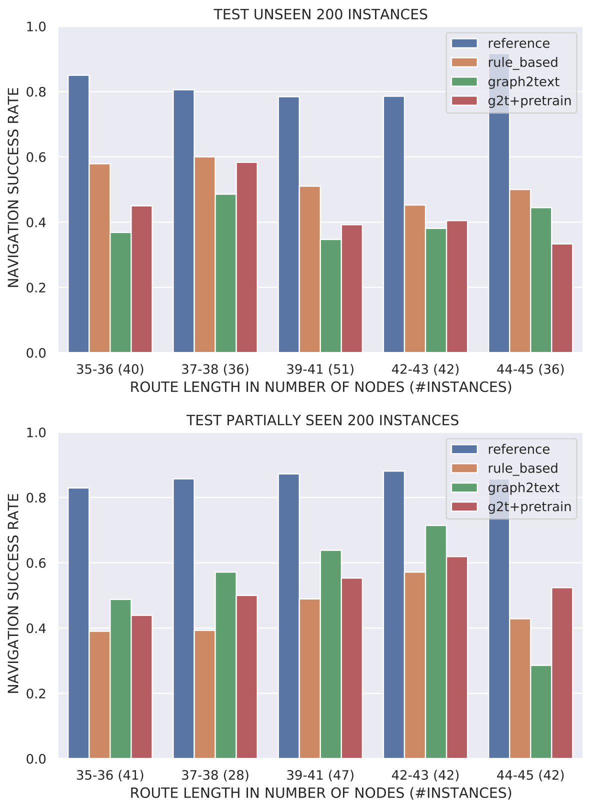


Figure 6: Navigation success rate in respect of route length. Length is measured in number of nodes in a route.

shows that the length of the route has little influence on the navigation success rate on the partially seen test set. On the unseen data there is tendency in favor of shorter routes for the g2t+pretrain model. The reference instructions do not show such bias. Figure 8 shows navigation success with respect to number of turns in a route which is another complexity indicator. The success rate drops with an increasing number of turns for all systems but not for the reference instructions. The analysis reveals that performance of our model drops with increasing route complexity while it is stable for reference instructions. The rule based system appears to be more stable with increasing number of turns in comparison to the learned models.

C Landmarks

Table 5 and 6 presents a scoring of types of landmarks produced by our pretrained model. A comparison of landmarks produced in human-generated reference instructions to those produced in model-generated instructions shows a large overlap on partially seen data, and ranking is similar to hand-crafted salient scores used in work in geoinformat-

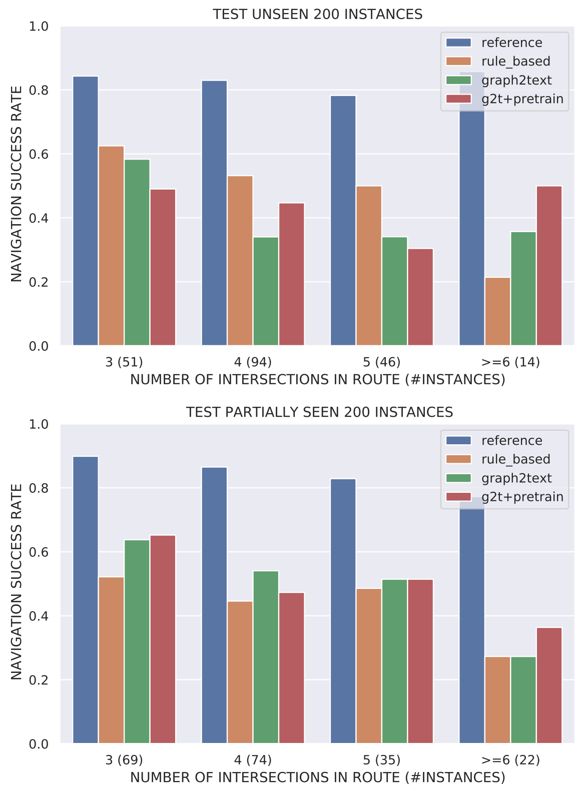


Figure 7: Navigation success rate in respect of number of intersections in a route. Each node in the route with more than two neighbors is counted as an intersection.

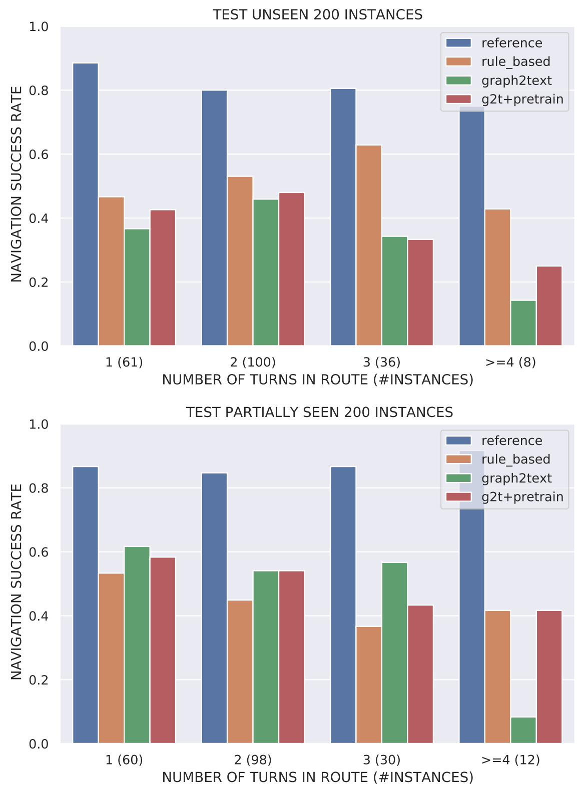


Figure 8: Navigation success rate in respect of number of turns in a route. A turn is defined as an intersection that isn't crossed in straight direction (345°to 15°).

	Reference			**Model**		
Top	**OSM tag**		**Score**	**OSM tag**		**Score**
1	amenity:	bank	0.41	amenity:	pharmacy	0.39
2	leisure:	park	0.35	shop:	furniture	0.38
3	amenity:	pharmacy	0.32	amenity:	bank	0.37
4	shop:	furniture	0.30	leisure:	garden	0.29
5	cuisine:	burger	0.29	cuisine:	burger	0.28
6	leisure:	garden	0.29	shop:	supermarket	0.25
7	cuisine:	coffee_shop	0.26	cuisine:	coffee_shop	0.25
8	amenity:	place_of_worship	0.25	cuisine:	american	0.24
9	cuisine:	american	0.23	shop:	convenience	0.22
10	amenity:	bicycle_rental	0.23	cuisine:	italian	0.21

Table 5: Frequency of OSM tags of landmark occurrences in the instructions for the **partially seen test set**, normalized by the number of occurrences in the input graph.

	Reference			**Model**		
Top	**OSM tag**		**Score**	**OSM tag**		**Score**
1	amenity:	cinema	0.58	cuisine:	juice	0.64
2	shop:	wine	0.53	amenity:	pharmacy	0.55
3	shop:	computer	0.53	shop:	convenience	0.50
4	amenity:	pharmacy	0.51	amenity:	cinema	0.46
5	cuisine:	coffee_shop	0.49	cuisine:	coffee_shop	0.46
6	tourism:	hotel	0.44	shop:	computer	0.45
7	shop:	convenience	0.42	tourism:	hotel	0.41
8	shop:	houseware	0.31	shop:	pet	0.39
9	shop:	supermarket	0.31	shop:	beauty	0.38
10	amenity:	bank	0.28	shop:	wine	0.38

Table 6: Frequency of OSM tags of landmark occurrences in the instructions for the **unseen test set**, normalized by the number of occurrences in the input graph.

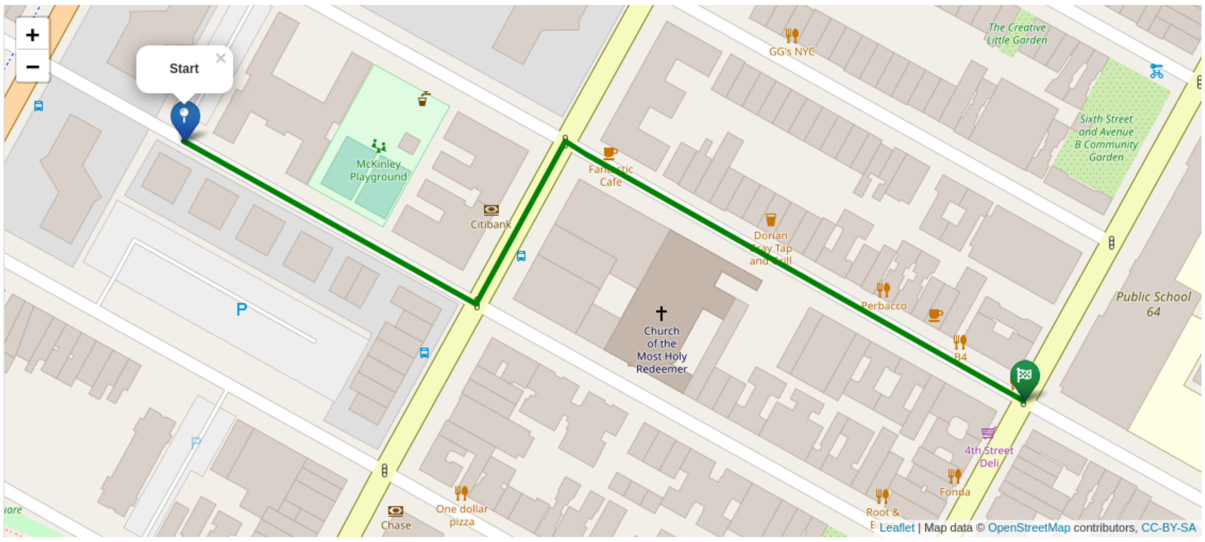
ics (Rousell and Zipf, 2017). The distribution of landmarks in the unseen test data is different from the partially seen data. To some extent, the model is able to adapt to the unseen environment.

D Annotation Instructions

The AMT workers got the following instructions for the writing task:

The goal of this task is to write navigation instructions for a given route. Imagine a tourist is asking for directions in a neighborhood you are familiar with and try to mention useful landmarks to support orientation. Another annotator will later read your instructions in order to find the goal location in StreetView (Navigation Run Task). If the other annotator successfully navigates to the goal location, your instruction is validated.

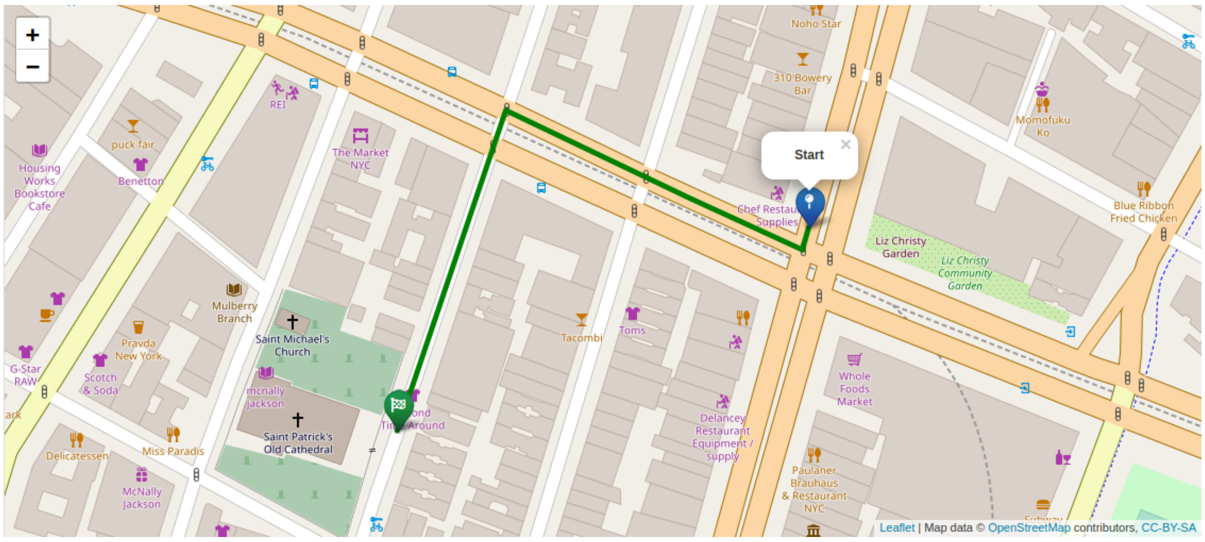
E Examples



Start
McKinley Playground
Citibank
Fantastic Cafe
GG's NYC
The Creative Little Garden
Sixth Street and Avenue B Community Garden
Dorian Gray Tap and Grill
Perbacco
Church of the Most Holy Redeemer
B4
Public School 64
4th Street Deli
Fonda
Root &
One dollar pizza
Chase
P
Leaflet | Map data © OpenStreetMap contributors, CC-BY-SA

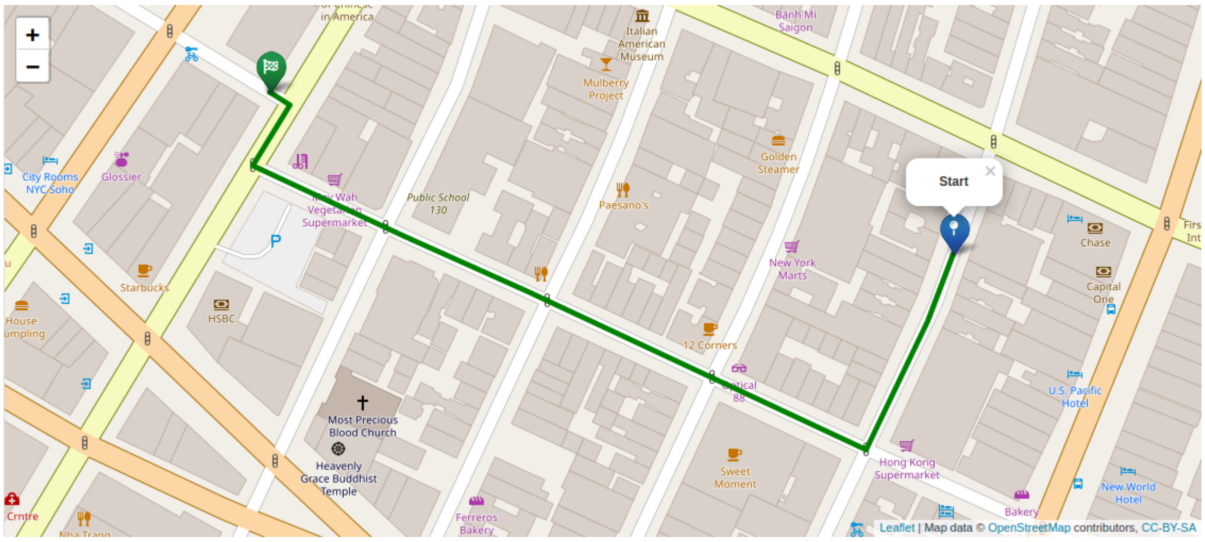
reference: *Head straight past the playground and turn left at the light. Turn right at the next light with Fantastic Cafe on the right. Continue down the very long street and stop at the next light with turntable 5060 on the near left corner.*
rule based: *parking left pitch left playground left pitch left bus stop right light left Citibank left light right Fantastic Cafe left Church of the right Dorian Gray Tap left Perbacco left Matcha Cafe Wabi left B4 left stop*
seq2seq: *Go straight to the light and turn left. Go straight to the first light and turn left. Go straight to the first light and stop a couple of steps before the next light. You should see Perbacco on your left.*
graph2text: *Head to the light and turn left. At the next light with Fantastic cafe on your near right corner, turn right. Go to the next light and stop in the middle of the intersection. B4 Gray Tap should be on your left.*
g2t+pretrain: *Head to the light and turn left. At the following light with Fantastic cafe on the far left corner, turn right. Go to the next light with 4th street Deli on the near left corner and stop in the middle of the intersection.*

Figure 9: Route from partially seen test set with successful navigation for g2t+pretrain.



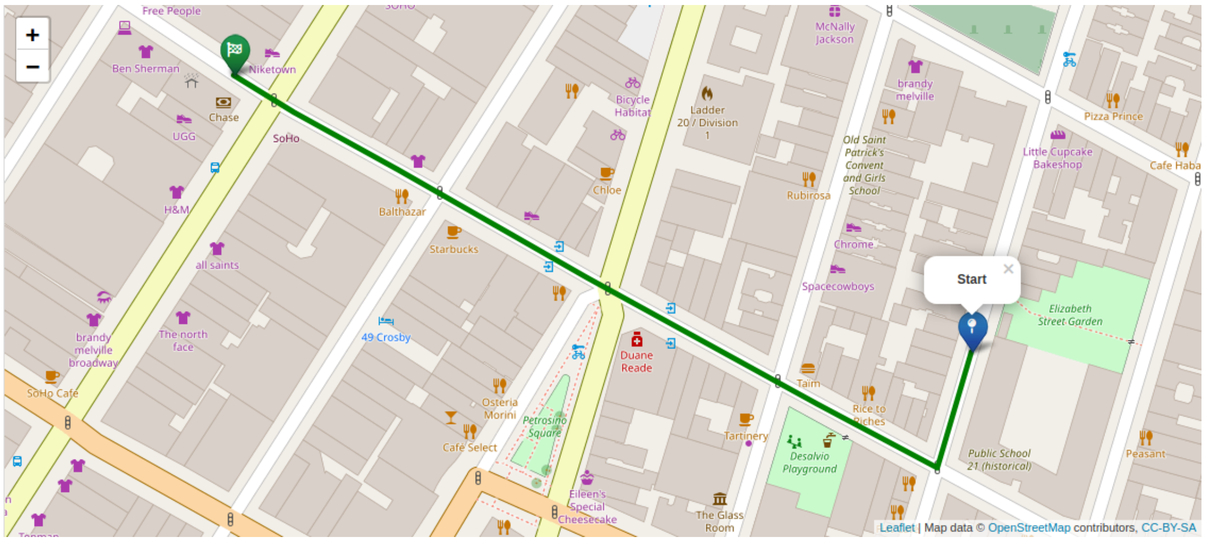
reference: *Turn right at the light right ahead of you. Go through the following light. The next light is a double intersection. Turn left at the first of the lights, and go through the second one. Pass a graveyard on the right and stop at the end of it, right as the church starts. It's about 2/3 down the block.*
rule based: *Garden left light right Chef Restaurant Supplies right light straight bus stop left light left light straight grave yard right stop Saint Patrick's right*
seq2seq: *Go to the light and turn right. Go through the following light, Restaurant on the corner. Pass a bus stop on the right, and go through the light. At the following light, bus stop on the corner, turn left. Go through the following light, and stop after a step or two. Saint Mary's church will be on the right.*
graph2text: *Turn right at the first set of lights. Go through the next set of lights and then turn left at the following lights. Pass Second Time Around half way down the block. Stop at Saint's around Saint's church on the right.*
g2t+pretrain: *Go to the light and turn right. Go straight through 1 light and at the following light, there should be a bus stop on the far left corner. Turn left and go about 1/2 to the next light, stopping in front of Saint Patrick's church on the right and graveyard Memorial's on the right.*

Figure 10: Route from unseen test set with successful navigation for g2t+pretrain.



reference: *Go to the light and turn right. Go through the following light with Optical 88. The next light will have Da Gennaro on the far right corner, go through the light. Go through the following light as well, with a school on the corner. Turn right at the following light. Take the first left possible and stop after a few steps.*
rule based: *Hong Kong Supermarket left light right Sweet Moment left light straight 12 Corners right light straight Da Gennaro right Public School 130 right light straight parking left May Wah Vegetarian right Hair Lounge right light right intersection left stop*
seq2seq: *Go straight and take a right at the intersection where Hong Kong supermarket is. Go through the next three intersections and at the fourth one take a right and stop at Hair Lounge.*
graph2text: *Go to the light and turn right. Go through the following light, Optical 88 on the corner. Go through the following light as well, Da Gennaro on the corner. At the following light, Hair Lounge on the corner, turn right. Take a step and stop.*
g2t+pretrain: *Head to the light and turn right. Go past the next 2 lights with Da Gennaro on the right corner. At the 3rd light with May Wah Vegetarian on the far right corner, turn left. Take one step and stop.*

Figure 11: Route from partially seen test set with unsuccessful navigation for g2t+pretrain.



reference: *Go straight to the light and turn right. Go through the next 3 lights. At the 4th, there is a Chase and Nike store on the corners. Go through the intersection and stop on the other side at your destination.*
rule based: *Public School 21 left light right Lombardi's Pizza left park left Rice to Riches right drinking water left playground left Taïm right light straight Tartinery left Duane Reade left light straight restaurant left Premium Laces right Starbucks left light straight MUJI at MoMA right Balthazar left SoHo left light straight Niketown right stop Chase left Chase left shelter left*
seq2seq: *Head to the light and turn right. Go past the next 3 lights with Starbucks on the left corner. At the 4th light with Chase bank on the near left corner, cross the street and stop.*
graph2text: *Head to the light and turn right. Go past the next 2 lights with Starbucks on the far left corner. At the 3rd light with Chase on the far left corner, cross the street and stop. Chase should be on your left.*
g2t+pretrain: *Head to the light and turn right. Go past the next 3 lights with Starbucks on the left corner. At the 4th light with Chase on the near left corner, cross the street. Take one step and stop.*

Figure 12: Route from unseen test set with unsuccessful navigation for g2t+pretrain.