Natural Language Generation in the Context of Providing Indoor Route Instructions

Andrea F. Daniele **Mohit Bansal** **Matthew R. Walter**
Toyota Technological Institute at Chicago
Chicago, IL, 60637
{afdaniele, mbansal, mwalter}@ttic.edu

***Abstract*—Modern robotics applications require robots to be able to communicate with humans seamlessly and effectively. We propose a structured algorithm for natural language generation in the context of indoor route instructions that allows a robot to combine metrical, topological and semantic information about the surrounding environment while generating natural language instructions. Our method learns a policy for giving natural language route instructions based on corpora of human instructions through learning from demonstration. An intermediate formal language allows the robot to reason first about which environmental features are relevant to the instruction with regards to minimizing ambiguity (i.e., the content selection problem), and secondly how to convey this information in a coherent manner by converting the structured representation into a natural language English sentence (i.e., the surface generation problem).**

I. Introduction

Natural language provides an efficient and flexible means for robots to communicate with humans. In order to interact with people through language, robots must be able to successfully generate and interpret free form sentences. One of the most common tasks we perform every day is navigation, and in order to reach our destination we constantly use route instructions. Depending on the specific situation, we can ask other people for instructions (e.g., when asking staff members how to get to the exit in a museum) or we can rely on automatic navigation devices (e.g., while driving a car in a new city).

Psychological studies have analyzed the usability of route instructions [12, 18], with a particular focus on identifying which are the characteristics that constitute "good" instructions. Waller and Lippa [25] showed that the acquired route knowledge depends on the function that landmarks serve during learning.

Existing commercial systems for route instructions generation typically rely on predefined patterns, populating fields (e.g., *"turn <direction>"*) in a turn-by-turn manner. While correct, the resulting instructions tend to be rigid and unnatural. Even though they are reliable while navigating road networks, they are not serviceable in the context of indoor route instructions. In fact, they are prone to ambiguities as when the environment structure is aliased.

The need for indoor navigation systems is constantly growing, especially in the context of assistive technologies where artificial intelligence and robotics fields provided outstanding tools during the last decades. The ability of robots to communicate with humans has the potential to improve the quality of life of people living with impairments. However, natural language generation is a challenging task, even when the context is fixed.

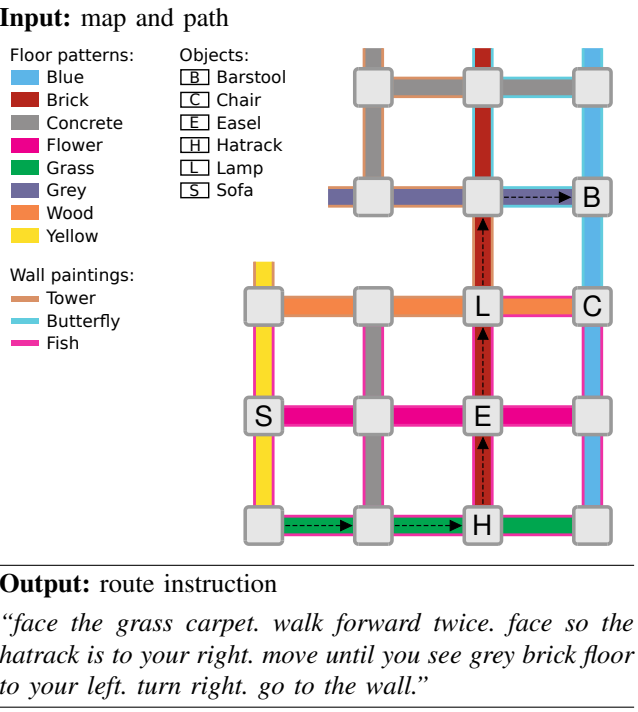

Fig. 1. An example of route instruction generated by our system.

One of the most challenging aspects to generating instructions is the choice of *what to talk about* in the instruction. This problem is known as *Content Selection*. We can hypothesize that people tend to generate instructions that reference salient landmarks, be it large, unique objects in indoor environments, or famous or otherwise easily recognized buildings when navigating outdoors. However, it is difficult to derive a set of rules for identifying such landmarks, since a variety of different factors participate in this task [12] (e.g., age, culture, and personal style).

Another challenge is to identify how much information to convey in the instruction. In fact, longer instructions tend to be more detailed than shorter ones, but they require a bigger memory; conversely, shorter sentences can be easily remembered, but they tend to be too generic and open to

CAS Structure	CAS Command
Turn(direction=**None**)	Turn(direction=**Left**)
Travel(distance=Distance(count=**None**))	Travel(distance=Distance(count=**2**))
Face(faced=Verify(desc=[Thing(type=**None**,value=**None**)]))	Face(faced=Verify(desc=[Thing(type=**Obj**,value=**Chair**)]))

Fig. 2. Compound Action Specifications (CAS) structures and commands examples

Context	**Description** (binary)
t	change orientation
w	change position
tw	change orientation and then position
wt	change position and then orientation
w_obj_at	the final place contains an object
w_past_obj	pass an object while walking
w_dead_end	the final place is a dead-end
w_goal	the final pose is the goal pose
t_start	it is the first action to take
t_new_carp	new floor color is faced from the final pose
t_obj_side	an object is visible from the final pose
t_obj_at	the place where to turn at contains an object
t_new_pict	new wall color is faced from the final pose
t_at_T	the place where to turn at is a dead-end

Fig. 3. List of contexts used as features for paths

Property	**Description**
nsl	number of key information to remember *(e.g., Turn(direction=*Right*) → 1)*
cmd	low-level command groundtruth *(i.e., walk, turn, walk+turn, turn+walk)*
dep	CAS command maximum depth *(e.g., Travel(dis=Dist(count=None)) → 2)*
eta	number of defined attributes
pcp	number of floor colors mentioned
ppc	number of wall colors mentioned
htw	whether or not to head towards an object
nln	number of landmarks mentioned
trf	turn reference frame *(i.e., allocentric, egocentric)*

Fig. 4. List of properties used as features for CAS structures

multiple (sometimes wrong) interpretations.

Natural language generation tasks are often formulated as three sub-problems: *sentence planning*, which performs the choice of how much information to convey, *content selection*, which selects which information to provide, and *surface generation*, which generates natural language sentences according to these choices.

Significant progresses have been made on this task (Chen and Mooney [7], Angeli et al. [2], Kim and Mooney [13], Konstas and Lapata [15], Mei et al. [21]). Many approaches solve the three sub-problems separately while others face them in a end-to-end fashion. Often the sentence planning sub-problem is considered as part of the content selection problem. We formulate our model as a sequence of three independent layers each of which solves a specific sub-problem.

We propose an algorithm that exploits an intermediate formal language to bridge the gap between a low-level metric representation of paths and the high-level structure of natural language sentences. At test time, we are given a *path* - defined on a *map* - and we are asked to generate a *natural language instruction*. We solve the sentence planning problem learning a policy for generating human-like route instructions from demonstrations by modeling the task as a Markov Decision Process (MDP) and applying Inverse Reinforcement Learning (IRL). Subsequently, a greedy-search algorithm performs the content selection task selecting the most appropriate information to include in the instruction. Finally, a set of automatically extracted templates and a language model based on a recurrent neural network with long short-term memory units (LSTM-RNN) are used to perform the surface generation task.

We evaluate our method using a publicly available dataset of route instructions (SAIL) and show how the system components affects the performances of the overall method performing a series of ablations.

II. Related Work

Much attention has been paid recently to developing algorithms that enable robots to interpret natural language manipulation and route instructions [19, 24, 14, 3, 4, 11, 8, 20]. Conversely, natural language generation considers the problem of representing low-level information the robot produced in a higher-level format people can easily understand (e.g., an English sentence).

Selective generation is a relatively new research area and many approaches have been investigated [**? ? ? ? ? ?** 13**?**]. Barzilay and Lee [6] propose a technique to learn topics within text using unsupervised methods and a Hidden Markov Model (HMM) to model their order.

Barzilay and Lapata [5] consider all the generation decisions jointly. This allows their model to capture dependencies between utterances and create sentences that are more consistent with how humans perform this task. Liang et al. [16] propose a generative semi-Markov model that simultaneously segments text into utterances and aligns each utterance with the corresponding meaning in the world state.

Angeli et al. [2] rearrange the generation problem as a sequence of three classification sub-problems (i.e., macro-content selection, micro-content selection and template selection) and solve them using three log-linear classifiers. They use a set of context-dependent rules for automatic template

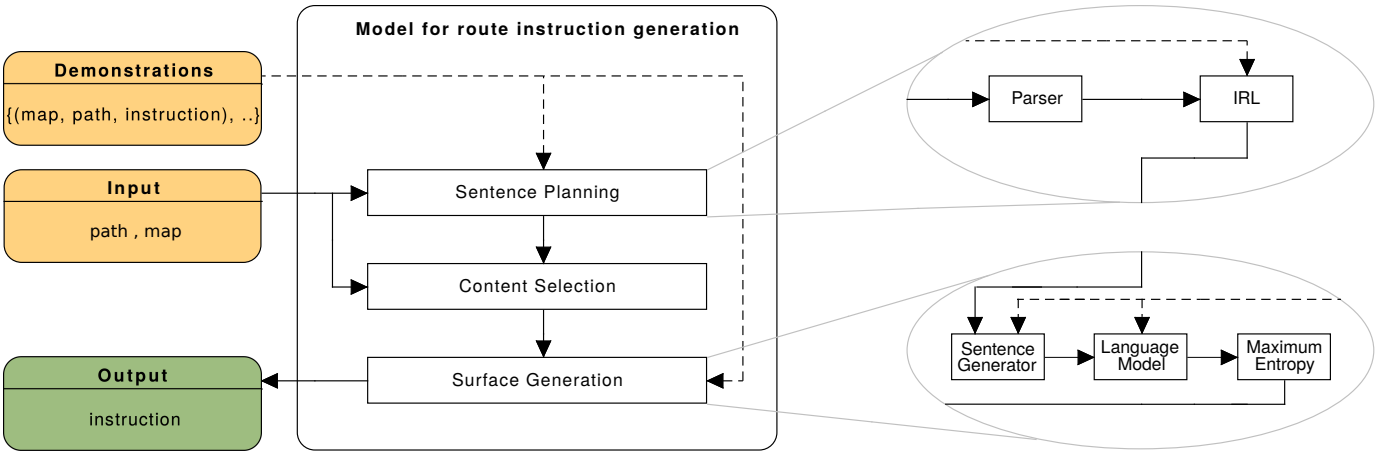

Fig. 5. An overview of our system

extraction from the training data. Mei et al. [21] propose and end-to-end model that employs a neural encoder-aligner-decoder network to jointly solve content selection and surface generation tasks.

Regarding the problem of generating natural language instructions, several efforts have been made to study how people perform such task - from a psychological point of view (Hund et al. [12], Lovelace et al. [18]) - and also how natural language aware path planning algorithms can produce easy-to-describe routes (Haque et al. [10], Richter and Duckham [23], Goeddel and Olson [9]).

Look et al. [17] asked 10 people to provide written route instructions for the Stata Center building (MIT), in order to develop a set of features for *"good"* instructions. They exploit the knowledge acquired from the collected data to define a batch of templates along with a set of application rules.

?] leverages cognitive principles about the way humans think about space in order to generate route directions that are easier to understand. **?**] derived a set of rules for generating route directions that appear to be more human-like by manually analyzing a corpus of human written directions.

Inverse reinforcement learning (IRL) techniques have been used to capture the behavior of human instructors from demonstrations. Oswald et al. [22] model the content selection problem as a Markov decision process and learn a policy from demonstrations using maximum-entropy IRL. Unlike our work, they do not address the surface generation problem.

III. Task definition

Natural language generation is the task of constructing a sequence of words in a human language given a low-level representation of what to communicate, such that the final sentence is syntactically correct and its semantics matches the intended message. In the context of providing indoor route instructions, the input is a tuple *(map, path)* consisting of an environment representation (map) and a route (e.g., as a sequence of waypoints), while the output is a natural language instruction. In addition to syntactic and semantic correctness, the final sentence should also be short and unambiguous. Such characteristics are fundamental to providing route instructions: the former ensures that the users can easily understand and remember the instructions, while the latter increases the likelihood that the instructions are followed correctly.

IV. The Model

Our framework (shown in Fig. 5) is composed of three layers: *sentence planning*, *content selection* and *surface generation*. The whole system takes as input a couple *(path,map)* and provides as output the corresponding *instruction* (i.e., English sentence).

A. Sentence Planning

The sentence planning layer takes as input a tuple *(path, map)* and outputs a set of CAS *structures* (see Sec. IV-A1). It is composed of two sub-blocks: natural language *Parser* and *IRL* (inverse reinforcement learning).

1) Compound Action Specifications: In order to bridge the gap between low-level information of the input domain (i.e., a sequence of waypoint poses) and high-level elements of the output domain (i.e., words) we employ the Compound Action Specification (CAS) model, a formal language representation suited to navigation commands that combines metrical, topological and semantic information into a single structure. A CAS consists of five *actions* (i.e., Travel, Turn, Face, Verify, Find), each of which is associated with a number of *attributes* used to characterize specific commands (e.g., Travel.distance, Turn.direction). We distinguish between *structures* and *commands*. The former is a CAS instruction where the attributes values are left blank, the latter is an instruction where all the attributes values are fixed. In other words, a structure defines a class of instructions, while a command is an instance of one of such classes. Fig. 2 shows some examples of CAS structures and commands. For a complete description of the CAS language, see MacMahon et al. [19].

2) Parser: We exploit the work done in [19] to convert an English instruction into a CAS command. Unlike the original implementation, our parser generates an *aligned* command. The alignment connects concepts in the command with the

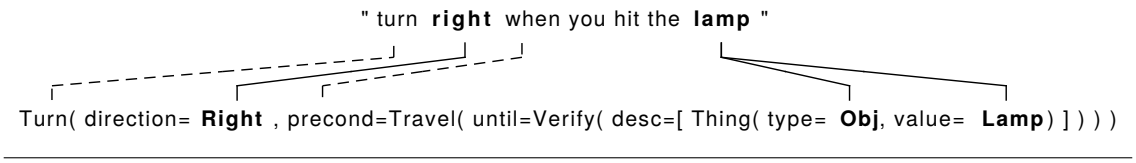

Fig. 6. An example of english instruction aligned with the corresponding CAS command

words in the sentence that generated them. An example of aligned command is shown in Fig. 6.

3) Inverse Reinforcement Learning: While giving directions, instructors make decisions about which kind of instruction to provide. Even if different factors participate in this decision (e.g., culture, age), different people tend to provide similar route instructions for similar paths. Hence, we can assume that people maximize a common unknown reward function while giving instructions. We formulate the problem as a Markov decision process (MDP) and employ inverse reinforcement learning (IRL) to learn a policy that seeks to mimic the preferences that people use in generating free-form route instructions, based upon a series of demonstrations.

An MDP is defined by a set of states $s \in S$, a number of actions $a \in A$, the transition dynamics between states $P(s'|a, s)$ and a reward function $R : S \times A \rightarrow \mathbb{R}$. The goal of reinforcement learning is to compute a policy $\pi(a|s)$ such that the expected cumulative reward is maximized. Conversely, the goal of inverse reinforcement learning is to recover an unknown reward function from a set of demonstrations. We represent both input *(path,map)* and output (CAS *structure*) as vectors of features, named c (i.e., contexts) and p (i.e., properties) respectively. In our model, S and A corresponds to the enumeration of all possible configurations of the vectors c and p respectively. Hence, each action a corresponds to a specific configuration of the vector p and each state s corresponds to a specific configuration of the vector c. We employ IRL to learn a policy $\pi(p|c)$ that decides which properties p an instruction should have given a context c.

We use 14 *contexts* as features for paths and 9 instruction *properties* as features for CAS commands. For each demonstration, *map* and *path* are represented by a single binary vector of 14 elements (indicating which contexts are active and which are not) while the *instruction* is represented by an integer-valued vector of 9 elements. The lists of contexts and instruction properties we use in our model are shown in Fig. 3 and 4 respectively.

Given the input $u = (p, m)$, the corresponding vector representation c_u is computed such that its i-th element ${}^i c_u$ is equal to 1 if the i-th context c_i is active. The desirable feature vector $p_u^\star$ for the CAS structure is computed such that:

$$ {}^i p_u^\star = \arg\max_v \ \pi(p_i = v|c_u) \quad \forall v \in values(p_i). $$

where ${}^i p_u^\star$ is the i-th element of $p_u^\star$, p_i indicates the i-th property and $values(p_i)$ is the set of discrete values of the property p_i. Using the desirable properties values, the first k_c nearest neighbor CAS commands are extracted from the aligned demonstrations and converted into the corresponding CAS structures C. A distance matrix $D \in \mathbb{R}^{k_c \times k_c}$ is computed such that the element $d_{ij} \in [0, 1]$ represents the dissimilarity between the structures c_i and c_j in C. The spectral clustering technique is used to group the structures by similarity and the outliers are removed. The set of distinct structures in C constitutes the output of the sentence planning layer.

B. Content Selection

The second layer of our system is the content selection layer. It takes as input a set of CAS structures and chooses the attributes values such that the final CAS commands are both valid and not ambiguous. Since the CAS language is formal, we can compute the likelihood of a CAS command c to be a valid instruction for a path p define on a map m as:

$$ P(c \mid p, m) = \frac{\delta(c \mid p, m)}{\sum_{j=1}^{M} \delta(c \mid \hat{p_j}, m)} \tag{1} $$

The index j iterates over all the possible paths that have the same starting pose of p and $\delta(c \mid p, m)$ is defined as:

$$ \delta(c \mid p, m) = \begin{cases} 1 & if \quad \eta(c) = \phi(c, p, m) \\ 0 & otherwise \end{cases} $$

where $\eta(c)$ is the number of attributes defined in c, and $\phi(c, p, m)$ is the number of attributes defined in c that are also valid with respect to the inputs p, m.

For each structure in input, a number of CAS commands are generated by iterating over the possible attributes values. For each configuration, correctness and ambiguity are evaluated using the probability function in (1). A command is valid if its likelihood is greater than a threshold P_t. The iteration stops when k_t valid commands are found or there are no other structures to iterate on. Since the number of possible configurations for a structure increases exponentially with respect to the number of attributes of the structure itself, a greedy-search algorithm is used to make the problem computationally tractable. The iteration algorithm is forced to use only objects and aspects of the world visible along the path. A set of valid CAS commands constitutes the output of the content selection layer.

C. Surface Generation

The last layer is called surface generation. It takes as input a set of CAS commands and outputs a natural language instruction. Each command results in a separate candidate route instruction, and this layer returns the natural language command with the greatest confidence.

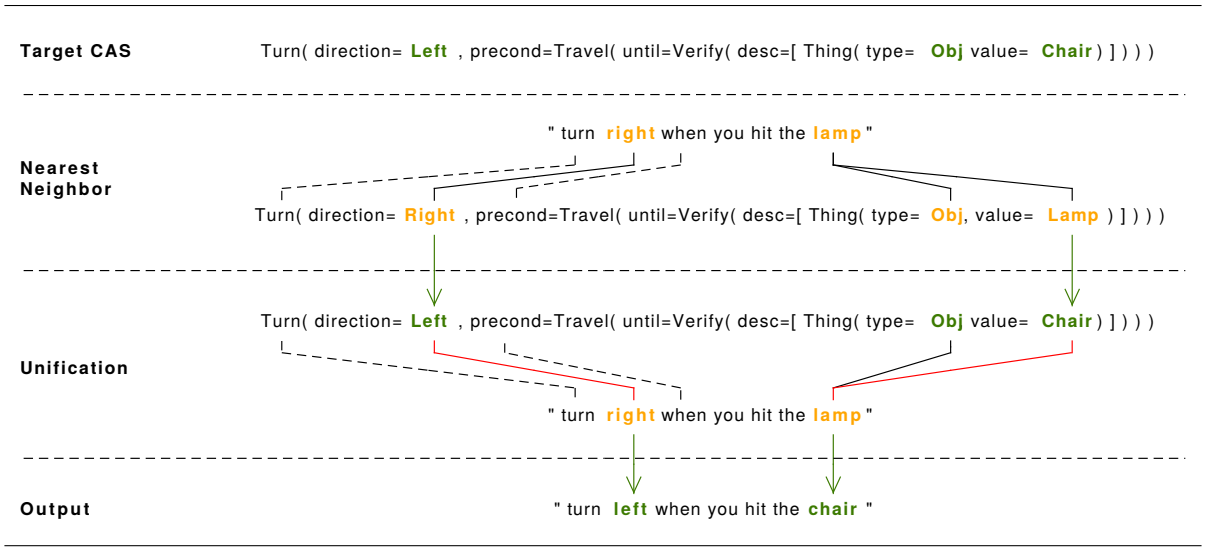

Fig. 7. An example of surface generation using a demonstration as template

1) Sentence Generator: For each command c_k, a set of k_s nearest neighbors is extracted from the set of aligned CAS commands generated during the training phase. We use the set of natural language instructions H related to such neighbors to compute a distance matrix $B \in \mathbb{R}^{k_s \times k_s}$, where $1.0 - b_{ij} \in B$ denotes the Ratcliff/Obershelp similarity between the instruction sentences h_i and h_j. The demonstration $(cas^\star, h^\star)$ that is used as template for generation is that which minimizes the average distance,

$$h^\star = \arg\min_{h_i \in H} \frac{1}{k_s} \sum_{j=1}^{k_s} b_{ij}. \tag{2}$$

The *unification* set $U(cas^\star, cas_t)$ is defined as the smallest collection of changes to be made in $cas^\star$ to match the target command cas_t. For example:

U(*Turn(direction=Left)*, *Turn(direction=Right)*) = {*Left*→*Right*}

The unification set is computed and the attributes alterations applied to the template command invalidating the alignment between the sentence and CAS command itself. Finally, the broken alignments are restored by changing the words in the sentence according to the new attributes values. An example of the surface generation process is shown in Fig. 7.

Our system learns how to map attributes values to words (e.g., Left→*left*, 1→*one*, Sofa→*bench*) using the alignments in the demonstrations.

2) Language Model: A language model is a method that assigns probability to natural language sentences. We employ a recurrent neural network with long short-term memory units (LSTM-RNN) as language model to score the instruction candidates in order to ensure that they are syntactically correct. An instruction is considered syntactically correct if its score is above a threshold x_t.

3) Maximum Entropy: Complex paths are split into sequence of simple segments and for each segment an independent set of candidate instructions is generated. Finally the best candidates for each segment are merged together to form a paragraph. During the training phase, different demonstrations can generate the same CAS command and, since we perform sentence clustering to increase the generation confidence (as described in Sec.IV-C1), our system tend to prefer particular sentence structures while generating instructions for similar CAS commands. This could lead to scenarios in which the final instruction contains repetitions like *"turn left"* or *"go straight"* and, according to Oswald et al. [22], repetitions make the instructions appear less natural. In order to promote language diversity, we provide as output the combination of instructions candidates such that the final paragraph maximizes the Shannon entropy.

V. Experiment Setup

A. Dataset

We train and evaluate our system using the publicly available SAIL route instruction dataset collected by MacMahon et al. [19]. We use the original data without correcting typos or wrong instructions. The dataset consists of 3213 demonstrations arranged in 706 paragraphs produced by 6 instructors for 126 different paths throughout 3 virtual environments. We partition the dataset in three parts, training (70%), validation (20%), and testing (10%) set. The publicly available PTB dataset is used to train the language model.

B. Evaluation metric

We evaluate our system by scoring the generated instructions using the modified BLEU-score [1]. Since a path can allow multiple valid CAS commands and hence different valid instructions, we evaluate the surface generation algorithm independently by generating the corresponding CAS command for each natural language instruction in the testing set. Finally, we use such commands as input for our surface generation algorithm and score the output sentence using the BLEU-score method with the original instruction as ground-truth.

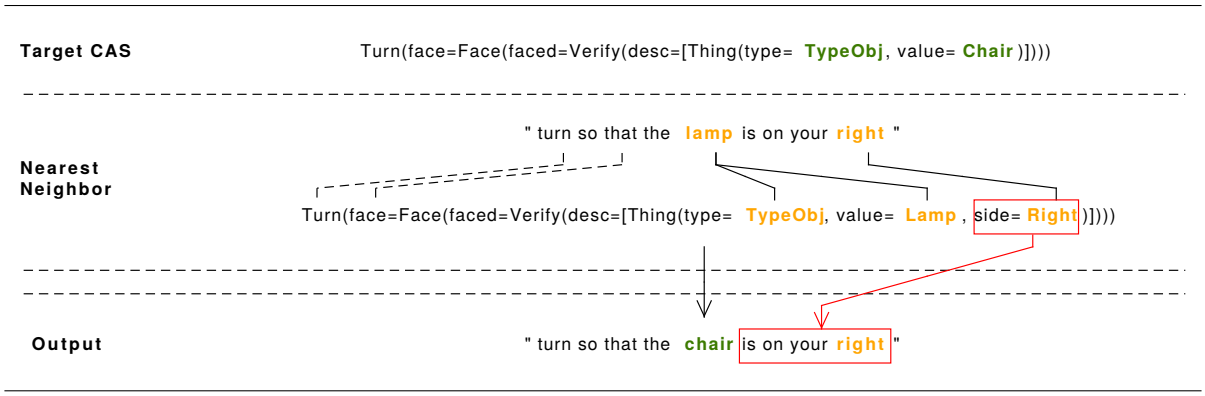

Fig. 8. An example of surface generation failure due to unwanted information inherited from the demonstration.

C. Training details

During the training phase, a set of demonstrations (SAIL dataset) is used to build the robot linguistic knowledge. Each demonstration is a tuple *(map, path, instruction)*, where:

- $map = (l_{semantic}, l_{topologic}, l_{metric})$ is a three-layer, hierarchical representation of the environment;
- $path = \{(x, y, \theta), ...\}$ is a sequence of poses;
- $instruction$ is a natural language instruction for the *path*;

We use the training set to train *inverse reinforcement learning* and *sentence generator* layers. The PTB dataset is used to pre-train the *language model* layer while training and validation sets are used for a fine-tuning phase. We use the following values for the system parameters: $k_c = 100$, $k_s = 100$, $P_t = 0.8$, $k_t = 50$, and $x_t = 5.0$.

VI. Results and Analysis

We report the performance of our surface generation module using the modified BLEU-score [1]. We use the CAS commands from the test set as the input to our surface generation algorithm and compare against the ground-truth reference generation. Our system achieves a BLEU-score of 0.87 for the whole testing set. To the best of our knowledge, the SAIL dataset has never been used before for the purpose of route instruction generation. This does not allow us to make a fair comparison between our results and others. Fig. 1 shows an example of a route instruction paragraph generated by our system.

A. Sentence Clustering Ablation

The contribution of the sentence clustering algorithm to the overall system performance is evaluated by analyzing the surface generation layer output score when the equation 2 is replaced by a random selection. The resulting system achieves a BLEU-score of 0.84 for the whole testing set. Fig. 9 shows an example of generated instruction when the sentence clustering algorithm is not considered (*NoCluster*).

B. Language Model Ablation

We analyze the effect of the language model layer to the system performance by analyzing the output of our system when the language model assigns constant scores. Fig. 9 shows an example of generated instruction when the language model is not considered (*NoLangM*).

C. Maximum Entropy Ablation

We evaluate the effect of the maximum entropy layer to the system performance by analyzing the output of our system when the maximum entropy method assigns constant entropy. Fig. 9 shows how the generated instruction is affected by repetitions when the maximum entropy layer is not considered (*NoMaxEnt*).

VII. Discussion

Our results show how the exploitation of an intermediate formal language helps the natural language generation process by bridging the gap between low-level and high-level information and providing a skeleton for the final sentence. Moreover, it shows also that we can achieve high performances even using a small number of demonstrations.

The generated instructions sometimes contain unwanted information inherited from the original demonstrations. This happens when the nearest neighbor of the target CAS command contains more attributes than needed. An example is shown in Fig. 8. Future efforts will be focused on the application of deep learning oriented approaches. A sequence-to-sequence model could be used indeed to learn how to *translate* a CAS command into a natural language sentence.

VIII. Conclusion

We presented a model for natural language generation in the context of providing indoor route instructions that exploits a structured approach to produce unambiguous, easy to remember and grammatically correct human-like route instructions.

References

[1] *Natural Language Toolkit (http://www.nltk.org/)*.

[2] Gabor Angeli, Percy Liang, and Dan Klein. A simple domain-independent probabilistic approach to generation. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 502–512, 2010.

Method	Generated Instruction
Full	*"face the grass carpet. walk forward twice. face so the hatrack is to your right. move until you see grey brick floor to your left. turn right. go to the wall."*
NoMaxEnt	*"face the grass carpet. move until you see red brick floor to your left. turn left. move until you see grey brick floor to your right. turn right. move until you see blue brick floor to your right"*
NoCluster	*"turn and face the blue olive hallway so that the pink floored hallway is on your left and the wall is on your right. walk forward twice. face the brick carpet. move until you see grey brick floor to your left. turn right. go to the hallway."*
NoLangM	*"you should be facing down a grass hallway. it's second movement after a lamp pole. and then take another back once you see a wall in the middle of the hall. move until you see black brick floor to your left. turn right. move to the left."*

Fig. 9. An example of instruction generation under the effect of some ablations.

[3] Yoav Artzi and Luke Zettlemoyer. Weakly supervised learning of semantic parsers for mapping instructions to actions. *Transactions of the Association for Computational Linguistics*, 1:49–62, 2013.

[4] Yoav Artzi, Dipanjan Das, and Slav Petrov. Learning compact lexicons for ccg semantic parsing. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1273–1283, 2014.

[5] Regina Barzilay and Mirella Lapata. Collective content selection for concept-to-text generation. In *Proceedings of the conference on Human Language Technology and Empirical Methods in Natural Language Processing*, pages 331–338. Association for Computational Linguistics, 2005.

[6] Regina Barzilay and Lillian Lee. Catching the drift: Probabilistic content models, with applications to generation and summarization. *arXiv preprint cs/0405039*, 2004.

[7] David L Chen and Raymond J Mooney. Learning to sportscast: a test of grounded language acquisition. In *Proceedings of the 25th international conference on Machine learning*, pages 128–135. ACM, 2008.

[8] Istvan Chung, Oron Propp, Matthew R. Walter, and Thomas M. Howard. On the performance of hierarchical distributed correspondence graphs for efficient symbol grounding of robot instructions. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Hamburg, Germany, 10 2015.

[9] Robert Goeddel and Edwin Olson. Dart: A particle-based method for generating easy-to-follow directions. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1213–1219, 2012.

[10] Shazia Haque, Lars Kulik, and Alexander Klippel. Algorithms for reliable navigation and wayfinding. In *Spatial Cognition V Reasoning, Action, Interaction*, pages 308–326. Springer, 2006.

[11] T.M. Howard, S. Tellex, and N. Roy. A natural language planner interface for mobile manipulators. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2014.

[12] Alycia M Hund, Martin Schmettow, and Matthijs L Noordzij. The impact of culture and recipient perspective on direction giving in the service of wayfinding. *Journal of Environmental P/sychology*, 32(4):327–336, 2012.

[13] Joohyun Kim and Raymond J Mooney. Generative alignment and semantic parsing for learning from ambiguous supervision. In *Proceedings of the 23rd International Conference on Computational Linguistics: Posters*, pages 543–551. Association for Computational Linguistics, 2010.

[14] Thomas Kollar, Stefanie Tellex, Deb Roy, and Nicholas Roy. Toward understanding natural language directions. In *Proceedings of the ACM/IEEE International Conference on Human-Robot Interaction (HRI)*, pages 259–266, 2010.

[15] Ioannis Konstas and Mirella Lapata. Unsupervised concept-to-text generation with hypergraphs. In *Proceedings of the 2012 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 752–761. Association for Computational Linguistics, 2012.

[16] Percy Liang, Michael I Jordan, and Dan Klein. Learning semantic correspondences with less supervision. In *Proceedings of the Joint Conference of the Annual Meeting of the Association for Computational Linguistics and the International Joint Conference on Natural Language Processing (ACL/IJCNLP)*, pages 91–99, 2009.

[17] Gary Look, Buddhika Kottahachchi, Robert Laddaga, and Howard Shrobe. A location representation for generating descriptive walking directions. In *Proceedings of the International Conference on Intelligent User Interfaces*, pages 122–129, 2005.

[18] Kristin L Lovelace, Mary Hegarty, and Daniel R Montello. Elements of good route directions in familiar and unfamiliar environments. In *Spatial information theory. Cognitive and computational foundations of geographic information science*, pages 65–82. 1999.

[19] M. MacMahon, B. Stankiewicz, and B. Kuipers. Walk the talk: Connecting language, knowledge, and action in route instructions. In *Proceedings of the National Conference on Artificial Intelligence (AAAI)*, 2006.

[20] Hongyuan Mei, Mohit Bansal, and Matthew R Walter.

Listen, attend, and walk: Neural mapping of navigational instructions to action sequences. *arXiv preprint arXiv:1506.04089*, 2015.

[21] Hongyuan Mei, Mohit Bansal, and Matthew R. Walter. What to talk about and how? Selective generation using lstms with coarse-to-fine alignment. *CoRR*, abs/1509.00838, 2015. URL http://arxiv.org/abs/1509.00838.

[22] Stefan Oswald, Henrik Kretzschmar, Wolfram Burgard, and Cyrill Stachniss. Learning to give route directions from human demonstrations. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 3303–3308, 2014.

[23] Kai-Florian Richter and Matt Duckham. Simplest instructions: Finding easy-to-describe routes for navigation. In *Geographic information science*, pages 274–289. Springer, 2008.

[24] S. Tellex, T. Kollar, S. Dickerson, Matthew R. Walter, Ashis G. Banerjee, S. Teller, and N. Roy. Understanding natural language commands for robotic navigation and mobile manipulation. In *Proceedings of the National Conference on Artificial Intelligence (AAAI)*, pages 1507–1514, San Francisco, CA, 8 2011.

[25] David Waller and Yvonne Lippa. Landmarks as beacons and associative cues: their role in route learning. *Memory & Cognition*, 35(5):910–924, 2007.