A new dataset and model for learning to understand navigational instructions

Ozan Arkan Can
Computer Science and Engineering
Koç University
Istanbul, Turkey
`ocan13@ku.edu.tr`

Deniz Yuret
Computer Science and Engineering
Koç University
Istanbul, Turkey
`dyuret@ku.edu.tr`

Abstract

In this paper, we present a state-of-the-art model and introduce a new dataset for grounded language learning. Our goal is to develop a model that can learn to follow new instructions given prior instruction-perception-action examples. We based our work on the SAIL dataset which consists of navigational instructions and actions in a maze-like environment. The new model we propose achieves the best results to date on the SAIL dataset by using an improved perceptual component that can represent relative positions of objects. We also analyze the problems with the SAIL dataset regarding its size and balance. We argue that performance on a small, fixed-size dataset is no longer a good measure to differentiate state-of-the-art models. We introduce SAILx, a synthetic dataset generator, and perform experiments where the size and balance of the dataset are controlled.

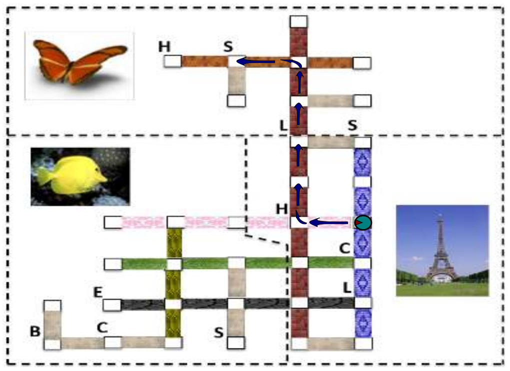

Figure 1: An illustration of a map and a set of instructions from the SAIL dataset (MacMahon et al., 2006). The letters indicate items (e.g. S for Sofa), the figures indicate wall paintings for each area divided by dashed lines, and the floor patterns distinguish the flooring. The circle represents the initial position of the agent and blue arrows represent the execution of the instruction set: "Take the pink path to the red brick intersection. Go right on red brick. Go all the way to the wood intersection. Go left on wood. Position one is where the sofa is."

1 Introduction

This paper explores the task of learning to follow natural language instructions in the simple domain of navigating a maze-like environment. We propose a new model with a unique perceptual representation and use the SAIL dataset (MacMahon et al., 2006) to compare our model with previous work. We describe the specifics of the dataset and the problems with its size and balance below. To address these problems we introduce a synthetic data generator SAILx and perform experiments where we control the size and balance of the datasets.

In the SAIL dataset (Figure 1, Section 2), an agent in a maze like environment receives sensory information from its line of sight. The agent is asked to navigate from a starting position to a target position which is described by a free-form natural language instruction. The aim of the agent is to follow the instruction by generating a sequence of actions. The agent can take one of four possible actions, {*MOVE, RIGHT, LEFT, STOP*}. The *RIGHT* and *LEFT* actions change the orientation of the agent, where the *MOVE* action transports the agent to the next position in the direction it is facing. The agent ends its trip if it takes the *STOP* action, hits a wall, or ex-

Task and frequency	Definition	Examples
Language only (31.7%)	Instructions contain only linguistic dependencies. The perceptual information is not required to solve the task.	turn right, move two steps, turn right and take a step
Turn to X (7.0%)	The agent should understand the specified perceptual phenomenon and take one or two "turn" actions.	turn to the chair, turn to the green path
Move to X (13.4%)	The agent should understand the specified perceptual phenomenon and take one or more "move" actions.	move to the sofa, go to the end
Turn and Move to X (1.7%)	The agent should understand the specified perceptual phenomenon. The agent should first complete orientation step(s), then the movement(s).	turn and move to the chair, go to the easel
Orient (5.2%)	The agent should orient itself along the specified perceptual phenomenon. There might be more than one conditions.	turn so that the wall is on your back
Description (9.6%)	Instructions describe a specific agent-environment configuration. This type of instructions are generally used to describe the final position. The most of the instructions require to take only STOP action.	you should be at the intersection of blue and brown
Move Until (8.7%)	The agent should move in the forward direction until a specific perceptual phenomenon occurs.	walk forward until you reach the blue floors
Any combination (22.7%)	Instructions contain two parts and at least one of them has a perceptual dependency.	at the black road intersection take a left

Table 1: Task definitions and examples.

ceeds the maximum number of actions.

Joint modeling of language and the world with descriptive representations is a crucial requirement for grounded language acquisition. Previous studies (Chen and Mooney, 2011; Chen, 2012; Artzi and Zettlemoyer, 2013; Artzi et al., 2014; Mei et al., 2015) represent the world indicating only the existence and direction of objects and environmental properties (Section 6). However, this representation misses the spatial relations between objects, environmental properties and the agent, such as the distance between objects, the order of objects and their relative positions with respect to the agent. To capture all the spatial information available to the agent, we propose a new grid based representation where we map the agent's view into a grid without any positional information loss (Section 4.1). We use a sequence to sequence neural network model with perceptual attention (Section 4.2) that takes advantage of the proposed perceptual representation and achieves state-of-the-art results on the SAIL dataset.

The SAIL dataset has certain problems regarding its size and balance. It consists of 3237 instruction sentences with associated worlds and actions. Even though the task domain and vocabulary are fairly small, this number may be insufficient for a couple of reasons. Typically one third of the dataset is used as a test set, and this is not large enough to differentiate models close in performance in a statistically significant manner. Certain words or tasks occur too few times in the training set to meaningfully generalize their meaning. In fact we often observe correct actions performed for the wrong reasons in state-of-the-art models (e.g. "move to the chair" always causing two steps to be taken, because that was the correct action in the only example in the training set).

Controlling the balance between different types of instructions may also be important to get a fair diagnosis of a model. Table 1 shows our rough categorization of the types of tasks found in the SAIL dataset. For each type, we state the proportion of sentences in the dataset of that type, the description of the task from the perspective of the agent, and some example instructions. One striking observation is that roughly a third of the dataset consists of "language only" instructions, i.e. instructions like "turn right" that the agent can follow without any perceptual information about the world. In fact, our experiments show that a blind agent that does not perceive the world can guess the correct action for 64% of the instruction sentences where the state-of-the-art performance is around 70%. On the other hand, semantically more complicated categories like "move until" may not have enough examples that will allow a learning agent to generalize correctly.

To overcome the sparse and unbalanced data problems, we propose SAILx, a synthetic data generator (Section 3) following the recent studies on artificial data generation. Goyal et al. (2016) and Agrawal et al. (2016) showed that human annotated datasets may contain latent biases and neural networks are able to leverage those biases to achieve high performance. Kuhnle and Copestake (2017a) and Kiela et al. (2016) discussed that artificially generated data allows us to examine language understanding abilities of multimodal systems. Following this direction, our algorithmic data generation procedure provides control over the task, the language, and the world with the ability to focus on individual grounding problems.

The standard way of evaluating models comparing their performance on a small, fixed-sized dataset may not be adequate because of the aforementioned size and balance problems. On the other hand, fixed-sized datasets that are too large may fail to differentiate between models of enough capacity that are all able to solve a given task. Using a data generator we can avoid the pitfalls of fixed-sized datasets. We can instead compare models directly in terms of their generalization power, by using the number of examples they take to learn a given task by reaching a threshold performance. We evaluate our model and compare it to others by measuring the number of instances they take to reach 90% test set accuracy on various tasks with the SAILx generator (Section 5.2).

Our key contributions include,

- We present a state-of-the-art model that benefits from the perceptual attention and improved word representation.
- We develop a synthetic data generation framework to overcome the small and unbalanced dataset problems.
- We present an experimental methodology to compare models in terms of learning efficiency.

2 SAIL Dataset

We use the SAIL dataset published by MacMahon et al. (2006). In this dataset, there are 3 different maps (named *Grid*, *Jelly*, *L*). Each map consists of different number of nodes and edges. Nodes might have an item (barstool, chair, easel, hatrack, lamp or sofa). Halls have different floor patterns (blue, brick, concrete, flower, grass, gravel, wood or yellow) and different wall paintings (butterfly, fish or tower).

The dataset was generated by using two sets of participants, six instructors and thirty-six followers. The instructors studied the mazes until they were able to efficiently navigate. Then they were asked to give a list of instructions to navigate from a starting position to a goal position without looking at the map. All written errors in the instructions, both syntactic (typos, grammar errors, etc.) and semantic (confusing left and right, calling a chair as a sofa, etc.), were kept without modification. The followers tried to follow the written list of instructions in the same maze without any prior knowledge of the environment. They were able to complete 69.64% of the paragraph length instructions accurately.

Chen and Mooney (2011) split paragraphs into individual sentences and paired each sentence with the corresponding segment of the path. We call this version of the dataset *Single-Sentence* and the original one *Paragraph*. *Single-Sentence* contains 3237 instructions and *Paragraph* contains 706 sets of instructions each paired with their corresponding maps and paths.

To better understand the nature of the dataset, we analyzed the instructions in terms of their perceptual and linguistic requirements. *Single-Sentence* instructions can be split into different categories: for example language only instructions such as "turn left" where there is no need for perception and ones such as "turn to the chair" which require perception. We detail the categorization and give the definition of each task in Table 1. Although the dataset contains challenging language grounding problems, one-third of the data consists of language only instructions which do not require perceptual understanding, and even a model that does not use perceptual information is able to perform surprisingly well (Section 5.3).

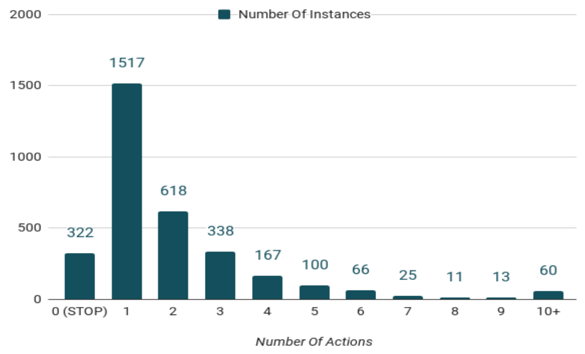

Figure 2: The distribution of the length of action sequences.

Another drawback of the SAIL dataset is the lack of diversity for the action sequences that the agent should take to complete the instructions. It can be seen from Figure 2 that more than half of the instructions require a single movement followed by a stop action or only the stop action itself.

Given the problems with the dataset, machine learning models have a tendency to map specific instructions to specific action sequences due to the lack of diversity in the physical configurations (e.g. mapping "move to the chair" instruction to (MOVE, STOP) action sequence because the agent had always received that instruction when it was one step away from the chair).

3 Synthetic Dataset (SAILx)

SAILx is a synthetic data generator we have developed that randomly generates maps, paths, and associated natural language instructions similar to the original SAIL data. Synthetic data generation allows us to investigate the problem of learning navigational instructions with fine-grained control over the map, the task, and the language. One can determine the complexity of the environment by controlling the different aspects such as the size of the map, length of the path, allowed items, flooring patterns, wall paintings and their locations. The language generation can be modified by controlling the vocabulary and text templates. The user chooses the category of each instance from the ones listed in Table 1. Thus datasets of different size and balance can be generated. We describe the data generation procedure in this section.

3.1 Map and Path Generation

We generate mazes (size of 8x8, in this study) using the recursive backtracker algorithm (Priestley and Ward, 1994) by selecting a random starting point. Once we obtain a maze, we decorate a random subset of nodes with random items. We divide the maze into two or three areas randomly and set the wall paintings of halls of each area with a distinct pattern. We choose the flooring patterns randomly but use the same pattern within a hall. By default, we use the same set of objects, paintings and floor patterns as the SAIL dataset.

To generate a target path we select random start and goal points that are far enough (at least four steps in this study) from each other. We find the shortest path between the two points using the A^* algorithm (Hart et al., 1968). If the user asks for a task pattern which does not match the generated path, we reject the path.

3.2 Instruction Generation

We generate instructions in the *Single-Sentence* form using a large subset of the vocabulary from the original SAIL dataset. We segment a path into turning and moving parts and use one or two segments for each sentence. To generate the instructions, we use physical task patterns and pattern dependent text templates. As an example, for the *Move to X* task, "reaching an end" is a physical task pattern and "/move/go/walk to the end/wall" and "take the path/hall/corridor until the end/wall" are two text templates. We select the optional parts of a template randomly.

3.3 Coverage

Task	Frequency	Overall	Non-Unique
Language Only	31.7%	90.25%	98.11%
Turn to X	7.01%	41.41%	73.75%
Move to X	13.38%	33.72%	60.95%
Turn and Move to X	1.73%	50.0%	70.0%
Orient	5.16%	64.67%	94.5%
Description	9.64%	14.42%	70.83%
Move Until	8.71%	2.84%	42.86%
Any combination	22.67%	5.31%	23.86%

Table 2: The task distribution and coverage statistics. The second column shows the frequency of each task in the SAIL dataset. The third column gives the percentage of the original instructions SAILx is able to generate. The last column gives the percentage for instructions that occur more than once.

We determined the task patterns and the text templates by examining the SAIL dataset and tried to generate instructions like the ones generated by human participants. Table 2 quantifies the proportion of the original instructions that our synthetic generator is able to generate. We are able to generate most of the instructions that occur more than once in the SAIL dataset.

3.4 The Fixed-Sized SAILx Dataset

For researchers who want to compare model performance on a fixed-sized dataset, we release a large set of pre-generated instructions with corresponding paths and maps. This dataset contains 105k instances and Figure 3 presents the proportion of each subtask. The number of instances was chosen to allow convergence to 90% test set performance on most of the subtasks given in Table 1. The performance of our model on this dataset is given in Section 5.

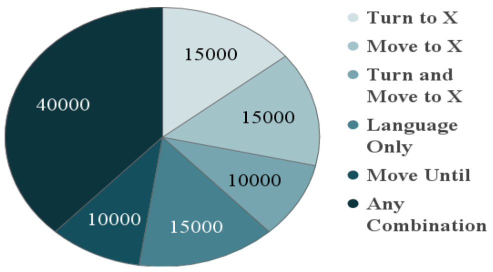

Figure 3: The task distribution in the fixed-sized SAILx data.

4 Model

We use a sequence to sequence neural network model with perceptual attention to learn the meanings of navigational instructions. In this section, we first describe input representations for the natural language instructions and perceptual states. Next, we describe the architecture of the model. Finally, we outline the training and inference procedures.

4.1 Input Representation

Natural Language Instructions

Since instructors are allowed to use free-form language, instructions may contain uppercase letters, hyphenations (e.g. *blue-tiled*), shortened words (e.g. *fwd* as short version of *forward*) and typos (e.g. *aesal* instead of *easel*) in the SAIL dataset (The SAILx dataset does not contain any misspellings). Considering the small number of instructions, we split hyphenated words into subwords to prevent sparsity. We left shortened words and typos as they are. We represented each word in the vocabulary with a one-hot vector.

Perceptual States

In previous studies, a common choice to represent the perceptual information for the agent is the concatenation of bag-of-features vectors for each direction and the agent's current position. For the SAIL dataset, the agent can observe items, flooring patterns and wall paintings of the halls. Although this representation captures the features and the directions of items, it can not capture the spatial relations.

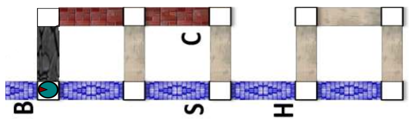

Figure 4: Spatial relations between items: The circle represents the agent. The following narrative is used to describe that the agent is supposed to stay at the current position: "you should be two alleys away from a sofa and then a hatrack beyond that". To process the given instruction, the agent should understand the distance between itself and the items in the instruction, and their relative ordering.

Figure 4 illustrates the need for understanding of spatial relations. The current position of the agent is the targeted final position for a paragraph. The last instruction of that paragraph mentions this by describing the final position referring to the surrounding objects. The agent must be able to recognize its position and the relative position of the surrounding objects to understand the given instruction.

To capture the spatial properties of items and their relations with the agent, we propose a grid-based representation (Figure 5) for perception. In this representation, the agent senses the world as a grid of cells. Each cell contains a binary vector representing either a node, a hall or a non-walkable cell. If the cell is a node, one of the item bits is set. If it is a hall the corresponding bits for the floor and wall patterns are set. In addition to material bits, one of the last three bits specifies the type of the cell.

We always fill the grid in an agent-centric orientation. Each row contains features about a direction. And the first cell of each row always contains information about the current location of the agent. We fill the grid in a clockwise manner. We copy the first row to the last row to preserve the geometric relations. We use 20 columns to fit the grid representation into actual maps.

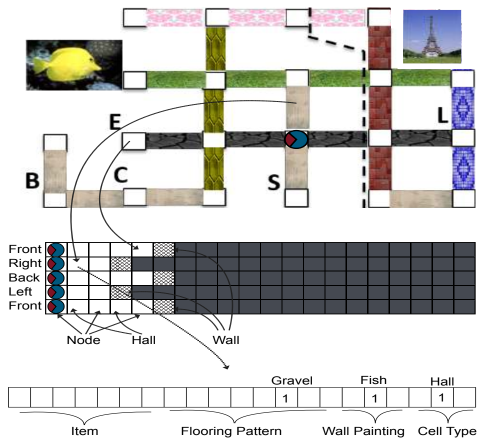

Figure 5: An example grid representation for the perceptual information.

4.2 The Neural Architecture

Our model[1] is a sequence to sequence neural model (Sutskever et al., 2014) with a perceptual attention module. It consists of three major components: an encoder, a decoder, and a convolutional neural network (CNN) with a channel attention mechanism. The encoder and the decoder model the input and output sequences by using Long Short-Term Memory (LSTM) units (Hochreiter and Schmidhuber, 1997). The CNN processes the perceptual input and the attention mechanism controls the focus on its features. Here, we give a detailed description of each component.

Encoder

The encoder takes the natural language instruction as a sequence of one-hot word vectors ($\mathbf{w} = (w_1, w_2, ..., w_I)$) and uses a bidirectional LSTM (Graves and Schmidhuber, 2005) to process the sequence in both forward and backward directions, producing a hidden state for each position with the following formulation:

$$\begin{aligned} x_i &= W_e w_i \\ f_i &= \text{LSTM}(W_f, x_i, f_{i-1}) \\ b_i &= \text{LSTM}(W_b, x_i, b_{i+1}) \\ h &= f_I \oplus b_1 \end{aligned} \tag{1}$$

where W_e is the embedding matrix that maps each word w_i into a dense vector x_i, W_f and f_i are the parameters and the hidden state for the forward LSTM, W_b, b_i are the parameters and the hidden state for the backward LSTM, h is the final hidden state for the encoder and is obtained by concatenating f_I and b_1. We use zero vectors for the initial f_0 and b_{I+1}.

A CNN with channel attention

We designed the CNN architecture (see Figure 6) such that the first layer of the network would be able to detect objects/properties on the perceptual input p_t by applying a filter bank. Then, a decoder-controlled attention vector weighs the properties captured on the output channels of this first layer. The purpose of the attention vector is to direct perception to parts of the input relevant to the instruction. We formulate the attention vector as follows:

$$\beta_{ti} = \frac{\exp(e_{ti})}{\sum_{j=1}^{I} \exp(e_{tj})} \tag{2}$$

where β_{ti} is the weight of input channel i at time t. β_{ti} is calculated as the softmax of an attention score e_{ti} which is a function of the hidden state of the decoder s_{t-1}:

$$e_{ti} = \text{NN}(W_a, s_{t-1}) \tag{3}$$

We use a feedforward neural network NN with parameters W_a and s_{t-1} as the input for the computation of the attention score.

After the attention layer, we have further convolutional layers[2] to learn higher order relations or make comparisons among different directions. We apply a relu activation after each convolution layer except the last one, where we use the sigmoid function, giving the final perceptual state c_t.

Decoder

The decoder generates a sequence of actions $\boldsymbol{a} = (a_1, a_2, .., a_T)$ given the perceptual state c_t and the language state h. The decoder defines a probability over the generated action sequence $\boldsymbol{a}$ as follows:

$$P(\boldsymbol{a}) = \prod_{t=1}^{T} P(a_t | s_t, c_t) \tag{4}$$

[1]We implemented our model in julia using Knet (Yuret, 2016).

[2]We use one additional layer for SAILx dataset and two additional layers for SAIL dataset

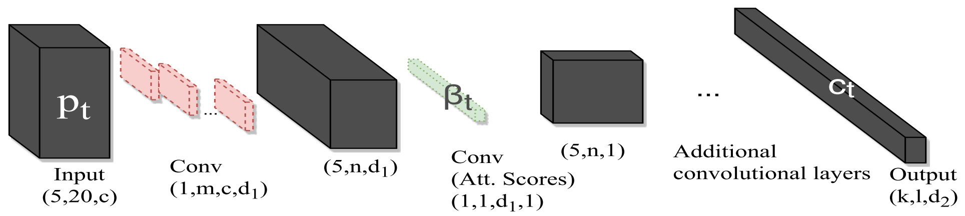

Figure 6: The convolutional neural network architecture with channel attention. The perceptual input p_t is a 5×20 grid and c is the length of the binary vector representation (Section 4.1) for each grid position. The first layer applies a $(1, m, c, d_1)$ filter bank, where $1 \times m$ is the size of the filter and d_1 is the number of filters. The output of the first layer is a $(5, n, d_1)$ tensor where $n = 20 - m + 1$. β_t is the attention vector determined by the decoder hidden state. After further convolutional layers, the output of the whole architecture, c_t, has (k, l, d_2) dimensions.

Our architecture models each conditional probability using the following formulation:

$$\begin{aligned} c_t &= \text{CNN}(W_c, p_t, \beta_t) \\ s_t &= \text{LSTM}(W_d, c_t \oplus a_{t-1}, s_{t-1}) \\ o_t &= W_1 s_t + W_2 c_t + b \\ P(a_t|s_t, c_t) &= \text{softmax}(o_t) \end{aligned} \tag{5}$$

where p_t is the perceptual state at time step t, c_t is the output of the CNN with parameters W_c and the attention distribution β_t. s_t is the hidden state of the decoder LSTM with parameters W_d. The input of the LSTM is the concatenation of c_t and a_{t-1}. The unnormalized output o_t is obtained by a linear combination of c_t and s_t. o_t is normalized by the softmax operation to determine the conditional probabilities for possible actions.

4.3 Training

We use the cross-entropy loss to maximize the probability of the ground-truth action sequence. If the correct action sequence is $[a_1, a_2, ..., a_T]$, we can train the model by minimizing the negative log-likelihood:

$$L = -\log P(a_1, a_2, ..., a_T|\mathbf{w}, p_{1:T}) \tag{6}$$

$$= -\log \prod_{t=1}^{T} P(a_t|s_t, c_t) \tag{7}$$

$$= -\sum_{t=1}^{T} \log P(a_t|s_t, c_t) \tag{8}$$

Since we model conditional probabilities with a differentiable recurrent model (3), weights of the model can be learned with backpropagation through time (Werbos, 1990).

4.4 Inference

Once we train the model, we generate action sequences by searching over alternative paths with beam search (Sutskever et al., 2014; Rush et al., 2015; Mei et al., 2015) using the distribution $P(a_t|s_t, c_t)$. For the *Paragraph* instances, we execute the beam search sentence by sentence, keeping the same beam between sentences. We generate actions for a sentence until the *STOP* action is taken, the agent hits a wall, or reaches a maximum number of actions. We also use an ensemble of trained models to obtain the action distribution by taking the average of the $P(a_t|s_t, c_t)$ distributions predicted by each model.

5 Experiments & Results

We conducted experiments on both the SAIL and the SAILx datasets. First, we describe the baseline models used in the experiments. Then, we give the details of the experiments on the SAIL dataset and compare our model with the state-of-the-art. Finally, we explain our experiments on the SAILx dataset and the propose evaluation metric for model comparison.

5.1 Baselines

Language Only (L.O.). This model is a regular encoder-decoder architecture without an attention mechanism. The encoder encodes the instruction with a bidirectional LSTM and the decoder predicts the action sequence taking the previous action as input at each time step. We feed the *STOP* action to the decoder as the initial input. This model does not use any perceptual information. We use gold actions as the input of the decoder during training. At test time, we use the actions predicted by the decoder.

Bag-Of-Features (B.O.F). This baseline model is also a regular encoder-decoder model with a bag-of-features representation (Section 4.1) for world states. The encoder encodes the instruction with a bidirectional LSTM and the decoder predicts the action sequence taking the bag-of-features world state as input at each time step.

5.2 SAIL dataset

We experimented on both the *Single-Sentence* and the *Paragraph* version of the SAIL dataset. Following the previous studies, a trial is counted as successful if and only if the final position and orientation match with the ground-truth path for the *Single-Sentence* version. For the *Paragraph* version, matching the final position is sufficient for success.

We use the *Single-Sentence* version of the data for the training and we test the model on both *Single-Sentence* and *Paragraph* datasets. We use six fold cross validation for the tuning of the model using splits for each map (*Grid, Jelly, L*). We left two maps as training data and split the remaining one into development data (50%) and test data (50%). We repeat this experiment by swapping the test and development data. This process is carried out for each map as development/test data. We run each fold ten times and report the size-weighted average of runs as the final score. We tune the hyper-parameters of the model depending on the average score of the six folds instead of tuning for each map.

The common approach on this dataset is using the test data for development, and denoted by "vTest". This approach is problematic because of the usage of the test data in the tuning process. Mei et al. (2015) proposed another approach (vDev) to tune the model by using 10% of the two maps as the development data and the remaining (90%) as the training data. This approach is also problematic because there may be linguistic differences among the instructors for different maps. By using half of the test map data for development and the other half for test, we avoid both problems.

We use Adam (Kingma and Ba, 2014) with default parameters for the optimization and gradient clipping (Mikolov et al., 2010) with the norm threshold 5. We use the success rate of the development data for early stopping. We stop the training if the model does not improve the development score within 10 iterations. We update the model after seeing each paired instruction and action sequence.

Method	Single-Sentence	Paragraph
Chen and Mooney (2011)	54.40	16.18
Chen (2012)	57.28	19.18
Kim and Mooney (2012)	57.22	20.17
Kim and Mooney (2013)	62.81	26.57
Artzi and Zettlemoyer (2013)	65.28	31.93
Artzi et al. (2014)	64.36	**35.44**
Andreas and Klein (2015)	59.60	-
Kočiskỳ et al. (2016) (vDev)	63.25	-
Mei et al. (2015) (vDev) (ens=10)	69.98	26.07
Mei et al. (2015) (ens=10)	71.05	30.34
Human (MacMahon et al., 2006)	-	69.64
Our Model (avg)	68.53	25.51
Our Model (ens=10)	**72.82**	32.57

Table 3: Overall results. Results of the previous works were obtained using the "vTest" procedure if not explicitly stated otherwise.

Table 3 shows the overall performance of our model against the previous studies with the "vTest" experimenting scheme. Our final model achieved the state-of-the-art result on *Single-Sentence* even though we do not tune the model for each map and use the test set for tuning. The performance on the *Paragraph* dataset is comparable with the best model (Artzi et al., 2014) which uses an additional semantic lexicon for the initialization of a semantic parser.

Task	L.O.	B.O.F.	Ours
Language Only	86.65	87.23	89.38
Turn to X	85.02	87.66	88.11
Move to X	62.82	73.21	76.91
Turn and Move to X	44.64	64.29	73.21
Orient	92.22	93.41	92.81
Description	86.54	86.22	89.42
Move Until	38.3	47.87	46.45
Any combination	20.16	33.24	35.56
Overall	63.61	69.54	71.58

Table 4: The performance (test accuracy) comparison of baselines and the proposed model. Results were obtained using ensemble of three models for each architecture.

Table 4 presents detailed results of baseline models compared to our final model. The *Language Only* baseline achieves surprisingly good results on the visual tasks without receiving perceptual information. One reason for this is that some instructions for the visual tasks contain non-perceptual clues to solve the task, e.g. *"turn **right** to the chair", "move **forward two steps** to the easel"*. The ability of the network to exploit the bias hidden in the action sequence distribution (see Figure 2) might be another factor for the high performance of the language only model. For example, most of the action sequences are (*RIGHT, STOP*) in the *Orient* task.

The Bag-Of-Features baseline performs better than the Language Only baseline in almost all tasks. Since both models have the same architecture for instruction modeling, it is expected that they result in similar performance in the Language Only task which we observe. Our final model further improves the B.O.F baseline in most tasks.

5.3 SAILx dataset

We use our synthetic dataset to compare the learning efficiencies of the proposed grid based model and the bag-of-features baseline. We train each model on a stream of data until the test accuracy exceeds a threshold. We use Adam (Kingma and Ba, 2014) with default parameters for the optimization. We use gradient clipping (Mikolov et al., 2010) and set the norm of the gradients 5.

Figure 7 demonstrates the number of instances required to reach 90% test accuracy by the B.O.F. baseline and our final model[3]. We did not include the L.O. baseline in this figure because it does not reach 90% on most of the tasks. We performed the hyper-parameter search for each model and for each task using golden section search (Kiefer, 1953).

Our grid based model achieves better or equal efficiency in almost all tasks when it is compared to the performance of the B.O.F baseline. The B.O.F baseline is not able to solve Any Combination and Norestriction tasks within the 250k instances limit. Additionally, both models show similar performance on the *Language Only* task because of their similar architecture for language processing.

In Table 5 we compare the performance of the baseline models and the grid based model on the fixed-sized SAILx dataset (Section 3). We split this data into train (70%), dev (15%) and test (15%) sets while preserving the proportion of each subtask. The grid based model is significantly better than the B.O.F baseline. The L.O baseline reaches an accuracy higher than the proportion of the language only data. One reason might be that since the model uses the previous action as the input of the decoder, it captures some action sequence patterns.

[3]The test accuracy is calculated as a moving average of accuracy on the next unseen instance: $moving\_average = 0.95 * moving\_average + 0.05 * accuracy\_on\_current\_batch$.

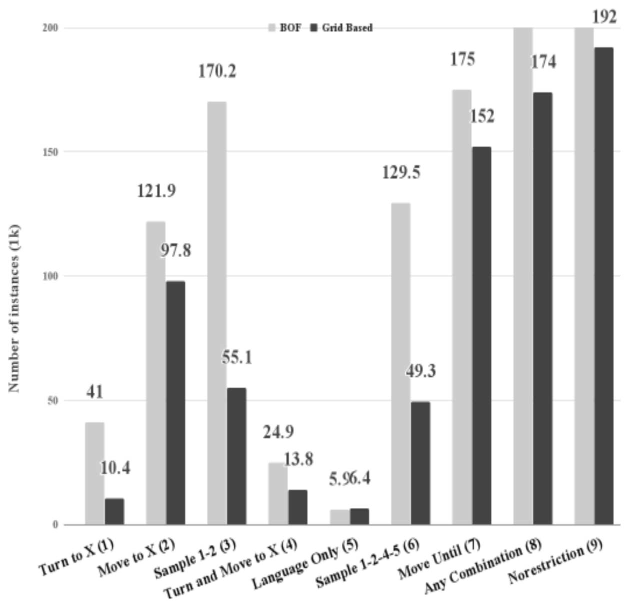

Figure 7: The learning efficiency comparison of the B.O.F baseline and the grid based model. The y axis shows the number of instances required to reach 90% accuracy on unseen data.

	Dev		Test	
	Avg.	Ens.	Avg.	Ens.
L.O	54.85	54..41	55.03	54.64
B.O.F	84.9	86.04	84.61	85.42
Ours	91.93	93.69	91.82	93.48

Table 5: Results on the generated fixed data. The average and ensemble scores were obtained by using 10 models for each architecture.

6 Related Work

In this section, we first summarize the studies on artificial data generation. Next, we outline the general literature on grounded language learning.

Artificial Data Generation

Similar to SAILx, SHAPEWORLD (Kuhnle and Copestake, 2017b), SHAPES (Andreas et al., 2016), CLEVR (Johnson et al., 2016) datasets provide control over the language and the world to examine subproblems of the visual question answering problem. bAbI tasks (Weston et al., 2015) also provide controlled subtasks, however, it is text only whereas our synthetic dataset is multi-modal.

Yu et al. (2017) propose an environment for navigating in a 2-D maze like environment, XWORLD. In contrast to the SAIL environment, in XWORLD, the environment is fully observable to the agent. (2017) propose a new 3D environment built on the ViZDoom API (Kempka et al., 2016). They manually generated 70 navigation instructions where each instruction is a combination of (action, attribute(s), object) triple. (2017) propose a simulated environment in 3-D. The language used in both environments involve only referring expressions to navigate the agent.

Grounded Language Learning

MacMahon et al. (2006) published the SAIL dataset and demonstrated a rule based system that uses manually aligned language and world features. The most studied approach in this domain is mapping natural language instructions into a formal meaning representation. Chen and Mooney (2011) presented a system that translates instructions to formal executable plans by training the KRISP semantic parser (Kate and Mooney, 2006) with aligned instruction and action-sequence pairs. Later, Chen (2012) improved the system by modifying the underlying semantic grammar and tested the system also on Chinese instructions. Kim and Mooney (2012) approached grounded language learning as probabilistic context free grammar (PCFG) induction introduced by Borschinger (2011) to the navigation domain. They further improve this technique using a re-ranking module with the task-specific weak signal (Kim and Mooney, 2013). Artzi and Zettlemoyer (2013) learned a semantic parser seeded with a manual lexicon. The parser operates on a combinatory categorical grammar (CCG) to translate the instruction into a lambda-calculus formalism for the semantic representation. They improved their system using a re-ranker and controlling the size of the lexicon in a data driven fashion (Artzi et al., 2014).

Another approach is learning to translate the instruction into an action sequence in an end-to-end fashion. Andreas and Klein (2015) modeled the instruction following task as scoring possible execution plans depending on the alignment with a given instruction. They use a conditional random field model to learn this alignment. Mei et al. (2015) use a textual attention-based (Bahdanau et al., 2014) encoder decoder neural network in contrast to our model which has an attention mechanism over the channels for the visual attributes. Although previous work uses attention to words, we observed that adding this mechanism does not bring any improvements. We suspect that initializing the decoder with the encoder's last hidden state is enough to convey the linguistic information in the SAIL domain. Our grid based representation for the perceptual states brings an improvement over the bag-of-features representation of (Mei et al., 2015). Kočiskỳ et al. (2016) utilizes an auto-encoder objective to enable semi-supervised training by leveraging randomly generated unsupervised data (random action sequences). They use a similar model to the bag-of-features baseline and show that the model is able to benefit from unsupervised training.

7 Conclusion

We have developed a synthetic data generation framework that can be used to provide an unlimited dataset perception-instruction-action instances to train and evaluate grounded language learning models. We proposed an evaluation metric to measure the learning efficiency of a model using the number of instances to reach a particular performance rather than the accuracy reached on a fixed-sized dataset. We developed a novel grid based representation for the perceptual states where spatial relations are missing in previous approaches. Our model resulted in state-of-the-art accuracy in *Single-Sentence* and achieved comparable results in *Paragraph* without using any external resources.

Acknowledgements

This work was supported by the Scientific and Technological Research Council of Turkey (TUBITAK) grants 114E628 and 215E201.

References

Aishwarya Agrawal, Dhruv Batra, and Devi Parikh. 2016. Analyzing the behavior of visual question answering models. *arXiv preprint arXiv:1606.07356*.

Jacob Andreas and Dan Klein. 2015. Alignment-based compositional semantics for instruction following. *arXiv preprint arXiv:1508.06491*.

Jacob Andreas, Marcus Rohrbach, Trevor Darrell, and Dan Klein. 2016. Neural module networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 39–48.

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

Yoav Artzi, Dipanjan Das, and Slav Petrov. 2014. Learning compact lexicons for ccg semantic parsing. In *EMNLP*, pages 1273–1283.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*.

Benjamin Börschinger, Bevan K Jones, and Mark Johnson. 2011. Reducing grounded learning tasks to grammatical inference. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 1416–1425. Association for Computational Linguistics.

Devendra Singh Chaplot, Kanthashree Mysore Sathyendra, Rama Kumar Pasumarthi, Dheeraj Rajagopal, and Ruslan Salakhutdinov. 2017. Gated-attention architectures for task-oriented language grounding. *arXiv preprint arXiv:1706.07230*.

David L Chen and Raymond J Mooney. 2011. Learning to interpret natural language navigation instructions from observations. In *AAAI*, volume 2, pages 1–2.

David L Chen. 2012. Fast online lexicon learning for grounded language acquisition. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Long Papers-Volume 1*, pages 430–439. Association for Computational Linguistics.

Yash Goyal, Tejas Khot, Douglas Summers-Stay, Dhruv Batra, and Devi Parikh. 2016. Making the v in vqa matter: Elevating the role of image understanding in visual question answering. *arXiv preprint arXiv:1612.00837*.

Alex Graves and Jürgen Schmidhuber. 2005. Framewise phoneme classification with bidirectional lstm and other neural network architectures. *Neural Networks*, 18(5):602–610.

Peter E Hart, Nils J Nilsson, and Bertram Raphael. 1968. A formal basis for the heuristic determination of minimum cost paths. *IEEE transactions on Systems Science and Cybernetics*, 4(2):100–107.

Karl Moritz Hermann, Felix Hill, Simon Green, Fumin Wang, Ryan Faulkner, Hubert Soyer, David Szepesvari, Wojtek Czarnecki, Max Jaderberg, Denis Teplyashin, et al. 2017. Grounded language learning in a simulated 3d world. *arXiv preprint arXiv:1706.06551*.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9(8):1735–1780.

Justin Johnson, Bharath Hariharan, Laurens van der Maaten, Li Fei-Fei, C Lawrence Zitnick, and Ross Girshick. 2016. Clevr: A diagnostic dataset for compositional language and elementary visual reasoning. *arXiv preprint arXiv:1612.06890*.

Rohit J Kate and Raymond J Mooney. 2006. Using string-kernels for learning semantic parsers. In *Proceedings of the 21st International Conference on Computational Linguistics and the 44th annual meeting of the Association for Computational Linguistics*, pages 913–920. Association for Computational Linguistics.

Michał Kempka, Marek Wydmuch, Grzegorz Runc, Jakub Toczek, and Wojciech Jaśkowski. 2016. Vizdoom: A doom-based ai research platform for visual reinforcement learning. In *Computational Intelligence and Games (CIG), 2016 IEEE Conference on*, pages 1–8. IEEE.

Jack Kiefer. 1953. Sequential minimax search for a maximum. *Proceedings of the American mathematical society*, 4(3):502–506.

Douwe Kiela, Luana Bulat, Anita L Vero, and Stephen Clark. 2016. Virtual embodiment: A scalable long-term strategy for artificial intelligence research. *arXiv preprint arXiv:1610.07432*.

Joohyun Kim and Raymond J Mooney. 2012. Unsupervised pcfg induction for grounded language learning with highly ambiguous supervision. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 433–444. Association for Computational Linguistics.

Joohyun Kim and Raymond J Mooney. 2013. Adapting discriminative reranking to grounded language learning. In *ACL (1)*, pages 218–227.

Diederik Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Tomáš Kočiský, Gábor Melis, Edward Grefenstette, Chris Dyer, Wang Ling, Phil Blunsom, and Karl Moritz Hermann. 2016. Semantic parsing with semi-supervised sequential autoencoders. *arXiv preprint arXiv:1609.09315*.

Alexander Kuhnle and Ann Copestake. 2017a. Deep learning evaluation using deep linguistic processing. *arXiv preprint arXiv:1706.01322*.

Alexander Kuhnle and Ann Copestake. 2017b. Shapeworld-a new test methodology for multimodal language understanding. *arXiv preprint arXiv:1704.04517*.

Matt MacMahon, Brian Stankiewicz, and Benjamin Kuipers. 2006. Walk the talk: Connecting language, knowledge, and action in route instructions. *Def*, 2(6):4.

Hongyuan Mei, Mohit Bansal, and Matthew R Walter. 2015. Listen, attend, and walk: Neural mapping of navigational instructions to action sequences. *arXiv preprint arXiv:1506.04089*.

Tomas Mikolov, Martin Karafiát, Lukas Burget, Jan Cernockỳ, and Sanjeev Khudanpur. 2010. Recurrent neural network based language model. In *INTERSPEECH 2010, 11th Annual Conference of the International Speech Communication Association, Makuhari, Chiba, Japan, September 26-30, 2010*, pages 1045–1048.

Hilary A Priestley and Martin P Ward. 1994. A multipurpose backtracking algorithm. *Journal of Symbolic Computation*, 18(1):1–40.

Alexander M Rush, Sumit Chopra, and Jason Weston. 2015. A neural attention model for abstractive sentence summarization. *arXiv preprint arXiv:1509.00685*.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*, pages 3104–3112.

Paul J Werbos. 1990. Backpropagation through time: what it does and how to do it. *Proceedings of the IEEE*, 78(10):1550–1560.

Jason Weston, Antoine Bordes, Sumit Chopra, Alexander M Rush, Bart van Merriënboer, Armand Joulin, and Tomas Mikolov. 2015. Towards ai-complete question answering: A set of prerequisite toy tasks. *arXiv preprint arXiv:1502.05698*.

Haonan Yu, Haichao Zhang, and Wei Xu. 2017. A deep compositional framework for human-like language acquisition in virtual environment. *arXiv preprint arXiv:1703.09831*.

Deniz Yuret. 2016. Knet: beginning deep learning with 100 lines of julia. In *Machine Learning Systems Workshop at NIPS*.