Listen, Attend, and Walk: Neural Mapping of Navigational Instructions to Action Sequences

Hongyuan Mei Mohit Bansal Matthew R. Walter
Toyota Technological Institute at Chicago
Chicago, IL 60637
{hongyuan,mbansal,mwalter}@ttic.edu

Abstract—**We take an end-to-end, sequence-to-sequence learning approach to the task of following natural language route instructions, i.e., mapping natural language instructions to action sequences. Our model is a bidirectional, alignment-based, long short-term memory recurrent neural network (LSTM RNN) that encodes the free-form navigational instruction sentence and the corresponding representation of the environment state. We propose a multi-level aligner as part of our network that empowers the model to focus on salient sentence "regions", using both high- and low-level input representations. This alignment-based LSTM then decodes the learned representation to obtain the inferred action sequence. Adding bidirectionality to the network helps further. In contrast to existing methods, our model uses no additional information or resources about the task or language (e.g., parsers or seed lexicons) and still achieves the best results reported to-date on a benchmark single-sentence dataset and gives competitive results for the limited-training multi-sentence setting. We evaluate our model through a series of ablation studies that elucidate the contributions of the primary components of our model.**

I. Introduction

Robots must be able to understand and successfully execute natural language navigational instructions if they are to work seamlessly alongside people in unstructured environments. However, interpreting such free-form instructions is challenging due to their ambiguity and variability, such as differences in amount of detail given, multiple ways of describing the same landmark or direction, spelling and grammar errors, etc. (see Chen and Mooney [6]).

Previous work in this domain [6, 5, 15, 16, 1, 2] usually requires specialized knowledge and resources like semantic parsers, seed lexicons, and re-rankers (which, in turn, need additional annotations) to interpret ambiguous, free-form natural language instructions. In contrast, the goal of our work is to learn to map instructions to actions via an end-to-end deep learning method that assumes no prior linguistic knowledge, i.e., it has to learn the meaning of all the words, spatial relations, syntax, and compositional semantics from just the raw training sequence pairs and be able to map the natural language instruction to an executable action sequence.

We propose a recurrent neural network with long short-term memory [13] to both encode the navigational instruction sequence bidirectionally and to decode the representation to an action sequence. LSTMs are well-suited to this task as they have been shown to be effective in learning the temporal dependencies that exist over such sequences, especially for the tasks of machine translation and image caption generation [27, 29, 18, 9, 7, 14]. Moreover, we also learn the alignment between words in the input navigational instruction and actions in the output action sequence using an alignment-based LSTM approach [4, 31]. We propose a *multi-level* aligner that empowers the model to focus on important sentence "regions" using both high- and low-level input representations.

We evaluate our model on the benchmark MacMahon et al. [20] dataset and achieve the best results reported to-date on the single-sentence task (which contains only 2000 training pairs), without using any prior knowledge about the task such as semantic parsers, seed lexicons, or re-rankers used in previous work. Moreover, our model exhibits accuracies that are not only high, but are also stable (robust) across multiple runs. On the multi-sentence task (executing a full paragraph), where the amount of training pairs is even smaller (just a few hundred pairs), our model is competitive with state-of-the-art and performs better than several existing methods, all of which use specialized linguistic resources. We perform a series of ablation studies in order to analyze the primary components of our method, including the encoder, alignment, and bidirectionality. We then conclude with directions for current/future work, including better (parameter-tying) models, the addition of extra (e.g., synthetic) data, or extra model power (e.g., reinforcement learning and a re-ranker over k-best action sequences).

II. Related Work

A great deal of attention has been paid of late to algorithms that allow robots and autonomous virtual agents to follow free-form navigational route instructions [20, 19, 6, 5, 15, 16, 12]. These methods solve what Harnad [11] refers to as the symbol grounding problem, that of mapping linguistic elements to their corresponding manifestation in the external world. Initial research in natural language symbol grounding focused on manually prescribed mappings between language and a set of pre-defined environment features and a set of actions [30, 20]. More recent work in statistical language understanding learns to convert free-form instructions into their referent symbols by observing the use of language in a perceptual context [24]. These methods represent natural language grounding in terms of manually defined linguistic, spatial, and semantic features [19, 22, 28]. They learn the parameters of these models from natural language corpora, typically requiring extensive manual annotation to pair each phrase to its corresponding grounding.

One class of grounded language acquisition methods treats the language understanding problem as one of learning a parser that maps free-form language into its formal language equivalent. For example, Matuszek et al. [22] assume no prior linguistic knowledge and employ a general-purpose supervised semantic parser learner. Alternatively, Chen and Mooney [6] parse free-form route instructions into formal action specifications that can then be carried out by the agent [20]. They learn the parser in a weakly supervised manner from natural language instruction and action sequence pairs, together with the corresponding world representation. Alternatively, Kim and Mooney [15] frame grounded language learning as probabilistic context free grammar (PCFG) induction and use learned lexicons [6] to control the space of production rules, which allows them to scale PCFGs to the navigation domain. Kim and Mooney [16] improve upon the accuracy by adding a subsequent re-ranking step that uses a weakly supervised discriminative classifier. Meanwhile, Artzi and Zettlemoyer [1] learn a CCG-based semantic parser to convert free-form navigational instructions to their manifestation in a lambda-calculus representation. Their novelty lies in the ability to learn these parsers with only weak supervision. As with Kim and Mooney [16], they improve upon the accuracy through re-ranking. Artzi et al. [2] extend their CCG parser learning by using statistics of the corpus to control the size of the lexicon, resulting in improved multi-sentence accuracy.

A second class of grounded language learning techniques function by mapping free-form utterances to their corresponding object, location, and action referents in the agent's world model. These methods learn a probabilistic model that expresses the association between each word in the instruction and its matching referent in the world model. The problem of interpreting a new instruction then becomes one of inference in this learned model. Kollar et al. [19] build a generative model over the assumed flat, sequential structure of language that includes a combination of pre-specified and learned models for spatial relations, adverbs, and verbs. Tellex et al. [28] later propose a framework that learns a discriminative model that expresses the hierarchical, compositional structure of language. They factor the probability distribution according to the parse structure of the free-form command and employ a log-linear factor graph to express the learned correspondence between linguistic elements and the space of groundings (objects, locations, and actions).

We adopt an alternative formulation and treat the problem of interpreting route instructions as a sequence-to-sequence learning problem. We learn this mapping in an end-to-end fashion using a neural network, without assuming prior linguistic structure or requiring annotated corpora. In this manner, our method is similar to recent work in sequence-to-sequence learning for machine translation [27, 4, 8] and image and video caption synthesis [18, 21, 9, 29, 7, 14]. Many approaches take an encoder-decoder approach to caption synthesis, whereby they embed images and captions into a joint embedding space, and subsequently decode the representation to generate captions [18, 21]. We also propose an encoder-decoder approach, encoding the free-form route instruction and then decoding the representation to identify the corresponding action sequence for a given world state. Our decoder includes a representation of alignment to focus on portions of the sentence relevant to the current action, similar to a technique that has proven effective in machine translation [4] and machine vision [23, 3, 31].

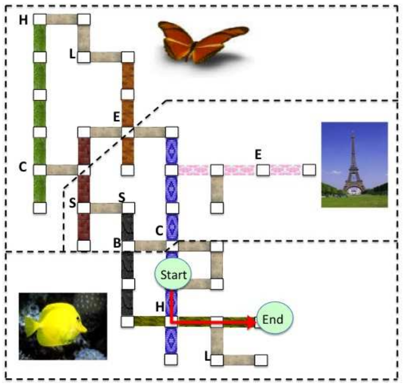

Fig. 1: An environment example from Chen and Mooney [6].

III. Task Definition

We consider the problem of mapping natural language navigational instructions to action sequences based only on knowledge of the local, observable environment. These instructions may take the form of isolated sentences (single-sentence) or full paragraphs (multi-sentence). We are interested in learning this mapping from corpora of training data of the form $(x^{(i)}, a^{(i)}, y^{(i)})$ for $i = 1, 2, \ldots, n$, where $x^{(i)}$ is a natural language instruction, $a^{(i)}$ is the corresponding action sequence, and $y^{(i)}$ is a representation of the observable environment. The model learns to produce the correct action sequence $a^{(i)}$ given a previously unseen $(x^{(i)}, y^{(i)})$ pair. The challenges of this task arise from the fact that the instructions are free-form, contain numerous spelling and grammatical errors, and are ambiguous in their meaning. Further, the model is only aware of the local area in the agent's line-of-sight.

In this paper, we specifically consider the route instruction dataset generated by MacMahon et al. [20]. The data includes free-form commands and action sequences within three different virtual worlds ("Grid," "L," and "Jelly") consisting of interconnected hallways (Fig. 1). There is a pattern (grass, brick, wood, gravel, blue, flower, or yellow octagons) on each hallway floor and a painting (butterfly, fish, or Eiffel Tower) on its wall, and there are objects (hat rack, lamp, chair, sofa, barstool, and easel) at intersections. After having explored an environment, instructors were asked to give written commands that describe how to navigate from one location to another without access to a map overview. Each instruction was then given to several human followers who were tasked with navigating in the virtual world without a map overview, and their paths were recorded. Of this data, many of the raw instructions include spelling and grammatical errors, while others are incorrect (e.g., misusing "left" and "right"), approximately 10% of the single sentences have no associated action, and 20% have no feasible paths.

IV. The Model

We formulate the problem of interpreting natural language route instructions as inference over a probabilistic model $P(a_{1:T}|y_{1:T}, x_{1:N})$, where $a_{1:T} = (a_1, a_2, \dots, a_T)$ is the action sequence, y_t is the world state at time t, and $x_{1:N} = (x_1, x_2, \dots, x_N)$ is the natural language instruction,

$$a^*_{1:T} = \underset{a_{1:T}}{\arg\max}\ P(a_{1:T}|y_{1:T}, x_{1:N}) \tag{1a}$$

$$= \underset{a_{1:T}}{\arg\max} \prod_{t=1}^{T} P(a_t|a_{1:t-1}, y_t, x_{1:N}) \tag{1b}$$

This problem can be viewed as one of mapping the given instruction sequence $x_{1:N}$ to the action sequence $a_{1:T}$. A natural means of learning this sequence-to-sequence mapping is to use a recurrent neural network model to first encode the input sentence

$$h_j = f(x_j, h_{j-1}) \tag{2a}$$

$$z_t = c(h_1, h_2, \dots h_N), \tag{2b}$$

where h_j is the encoder hidden state for word $j \in \{1, \dots, N\}$, f and c are nonlinear functions, which we define shortly, and the context vector z_t is the encoded version of the language instruction at action time step $t \in \{1, \dots, T\}$. Next, a different RNN is used to decode the context vector z_t to arrive at the desired likelihood (1)

$$P(a_{1:T}|y_{1:T}, x_{1:N}) = \prod_{t=1}^{T} P(a_t|a_{1:t-1}, y_t, x_{1:N}) \tag{3a}$$

$$P(a_t|a_{1:t-1}, y_t, x_{1:N}) = g(s_{t-1}, z_t, y_t), \tag{3b}$$

where s_{t-1} is the decoder hidden state at time $t-1$, and g is a nonlinear function. Inference then follows by maximizing this posterior to determine the desired action sequence. This approach is similar to encoder-decoder methods for machine translation [8].

Our model (Fig. 2(a)) employs LSTMs as the nonlinear functions f and g due to their ability to learn long-term dependencies that exist over the instruction and action sequences, without suffering from exploding or vanishing gradients. Our model also integrates *multi-level* alignment to focus on parts of the instruction that are more salient to the current action at multiple levels of abstraction. We next describe each component of our network in detail and illustrate them in Figure 2.

a) Encoder: Our encoder takes as input the natural language route instruction represented as a sequence $x_{1:N} = (x_1, x_2, \dots, x_N)$, where x_1 and x_N are the first and last words in the sentence, respectively. We treat each word x_i as a K-dimensional one-hot vector, where K is the vocabulary size. We feed this sequence into an LSTM-RNN that summarizes the temporal relationships between previous words and returns a sequence of hidden annotations $h_{1:N} = (h_1, h_2, \dots, h_N)$, where the annotation h_j summarizes the words up to and including x_j in the sequence.

We adopt an encoder architecture (Fig. 2(c)) similar to that of Graves et al. [10],

$$\begin{pmatrix} i^e_j \\ f^e_j \\ o^e_j \\ g^e_j \end{pmatrix} = \begin{pmatrix} \sigma \\ \sigma \\ \sigma \\ \tanh \end{pmatrix} T^e(x_j) \tag{4a}$$

$$c^e_j = f^e_j \odot c^e_{j-1} + i^e_j \odot g^e_j \tag{4b}$$

$$h_j = o^e_j \odot \tanh(c^e_j) \tag{4c}$$

where T^e is an affine transformation, σ is the logistic sigmoid that restricts its input to $[0, 1]$, i^e_j, f^e_j, and o^e_j are the input, output, and forget gates of the LSTM, respectively, and c^e_j is the memory cell activation vector. The memory cell c^e_j summarizes the LSTM's previous memory c^e_{j-1} and the current input, which are modulated by the forget and input gates, respectively. The forget and input gates enable the LSTM to regulate the extent to which it forgets its previous memory and the input, while the output gate regulates the degree to which the memory affects the hidden state.

Our encoder employs *bidirectionality*, encoding the sentences in both the forward and backward directions, an approach that has been found to be successful in speech recognition and machine translation [4, 8, 10]. In this way, the hidden annotations $h_j = (\overrightarrow{h}_j^\top; \overleftarrow{h}_j^\top)^\top$ concatenate forward $\overrightarrow{h}_j$ and backward annotations $\overleftarrow{h}_j$, each determined using Equation (4c).

b) Multi-level Aligner: The context representation of the instruction is computed as a weighted sum of the word vectors x_j and encoder states h_j. Whereas most previous work align only based on the hidden annotations h_j, we found that also including the original input word x_j in the aligner improves performance. Intuitively, this allows the decoder to not just reason over the high-level, context-based representation of the input sentence h_j, but to also consider the original *low-level* word representation x_j. Moreover, this permits the model to better match salient words in the input sentence (e.g., "chair") to the corresponding landmarks in the world state y_t used in the decoder.

$$z_t = \sum_j \alpha_{tj} \begin{pmatrix} x_j \\ h_j \end{pmatrix} \tag{5}$$

The weight α_{tj} associated with each pair (x_j, h_j) is computed as

$$\alpha_{tj} = \exp(\beta_{tj}) / \sum_j \exp(\beta_{tj}), \tag{6}$$

where the alignment term $\beta_{tj} = f(s_{t-1}, x_j, h_j)$ weighs the extent to which the word at position j and those around it match the output at time t. The alignment is modelled as a one-layer neural perceptron

$$\beta_{tj} = v^\top \tanh(W s_{t-1} + U x_j + V h_j), \tag{7}$$

where v, W, U, and V are learned parameters.

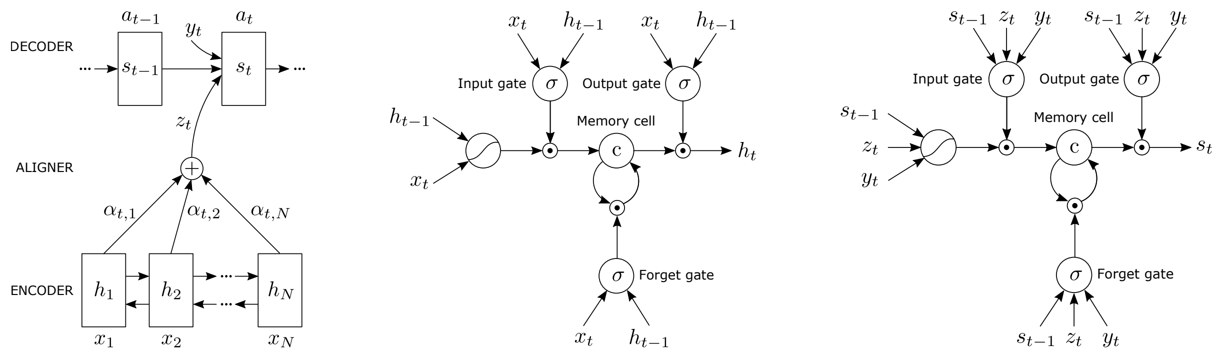

Fig. 2: A diagram of our (a) encoder-decoder architecture used to infer the action sequence $a_{1:T}$ for a given natural language instruction $x_{1:N}$. Our model uses a (b) bidirectional LSTM-RNN to encode each instruction and (c) an alignment-based LSTM-RNN decoder to represent the conditional distribution over actions.

c) Decoder: Our architecture uses an LSTM decoder (Fig. 2(c)) that takes as input the current world state y_t, the context of the instruction z_t, and the LSTM's previous hidden state s_{t-1}.

The output is the conditional probability distribution over the next action (3), represented as a deep output layer [25],

$$\begin{pmatrix} i_t^d \\ f_t^d \\ o_t^d \\ g_t^d \end{pmatrix} = \begin{pmatrix} \sigma \\ \sigma \\ \sigma \\ \tanh \end{pmatrix} T^d \begin{pmatrix} Ey_t \\ s_{t-1} \\ z_t \end{pmatrix} \tag{8a}$$

$$c_t^d = f_t^d \odot c_{t-1}^d + i_t^d \odot g_t^d \tag{8b}$$

$$s_t = o_t^d \odot \tanh(c_t^d) \tag{8c}$$

$$P(a_t|a_{1:t-1}, y_t, x_{1:N}) = \text{softmax}\left(L_0(Ey_t + L_s s_t + L_z z_t)\right) \tag{8d}$$

where E is an embedding matrix and L_0, L_s, and L_z are parameters to be learned.

d) Training: The encoder and decoder models are trained so as to predict the action sequence $a^*_{1:T}$ according to Eqn. 1 for a given instruction $x_{1:N}$ and world state $y_{1:T}$ drawn from the training corpora. We use the negative log-likelihood of the demonstrated action at each time step t as our loss function,

$$L = -\log P(a_t^*|y_t, z_t). \tag{9}$$

As the entire model is a differentiable function, the parameters can be learned by back-propagation.

e) Inference: Having trained the model, we generate action sequences by finding the maximum a posteriori actions under the learned model (1). For the single-sentence task, we perform this search using standard beam search to maintain a list of the current k best hypotheses.[1] We iteratively consider the set of k-best sequences up to time t as candidates to generate sequences of size $t+1$ and keep only the resulting best k of them. For multi-sentence, we perform the search sentence-by-sentence, and initialize the beam of the next sentence with the entire list of previous k best hypotheses. Also, as used in previous related work [27, 29, 32], we perform inference over an ensemble of randomly initialized models. At each action time step t, we generate actions using the average of the posterior likelihoods of the ensemble models.[2]

f) World State: The world state y_t encodes the observable world at time t. We make the standard assumption that the agent is able to observe all elements of the environment that are within line-of-sight. In the specific domain that we consider for evaluation, these elements include the floor patterns, wall paintings, and objects that are not occluded by walls. We define the features in each direction as a bag-of-words vector and concatenate all directions to form the world state.

V. Experimental Setup

g) Dataset: We train and evaluate our model using the publicly available SAIL route instruction data collected by MacMahon et al. [20]. The dataset contains 706 non-trivial navigational instruction paragraphs, produced by six instructors for 126 unique start and end position pairs spread evenly across three virtual worlds. These instructions are segmented into individual sentences and paired with an action sequence [6]. Table I presents the corpus statistics with standard deviation in parentheses.

TABLE I: The statistics of the Chen and Mooney [6] dataset

	Quantity
Paragraphs	706
Sentences	3237
Vocabulary	591
Words per sentence	7.82 (5.14)
Actions per sequence	3.04 (2.43)

h) Training Details: We follow the same procedure as Chen and Mooney [6], training with the segmented data and testing on both single- and multi-sentence versions. We train our models using three-fold cross-validation based on the three maps. In each fold, we retain one map as test and partition the two-map training data into training (90%) and validation

[1]We use a beam width of ten to be consistent with previous work [2].

[2]We use five models as is done in previous work (see Section VI-01).

(10%) sets, the latter of which is used to tune hyperparameters. We retain the data for the third map as the test set. We repeat this process for each of the three folds and report (size-weighted) average test results over these folds. We later refer to this training procedure as “vDev.” Additionally, some previous methods (p.c.) adopted a slightly different training strategy whereby each fold trains on two maps and uses the test map to decide on the stopping iteration. In order to compare against these methods, we also train a separate version of our model in this way, which we refer to as “vTest.”

For optimization, we empirically found Adam [17] to be very effective for training with this dataset. Using stochastic gradient descent with decaying learning rate, we found that the initial rate, the decay function, and norm clipping each affect the time required for convergence. With Adam, however, we found the validation success rate, defined shortly, increases to about 65% after the first epoch, while the convergence level is around 80%. The training usually converges within 50 epochs.

The only two regularization strategies we use are dropout [26, 32] and early stopping based on the validation task metric. Similar to previous work [31], we found that the validation log-likelihood is not well correlated with the task metric.

i) Evaluation Metrics: We evaluated our end-to-end model on both the single-sentence and multi-sentence versions of the corpus. For single-sentence, following previous work, the strict evaluation metric deems a trial to be successful iff the final position and orientation exactly match those of the original demonstration. For multi-sentence, we disregard the orientation as previous work did, since the end goals in MacMahon et al. [20] are defined without orientation; however, this setting is still more challenging than single-sentence due to cascading errors over individual sentences.

VI. Results and Analysis

In this section, we discuss the overall performance of our model on the single- and multi-sentence benchmarks and compare against previous work. We then present an analysis of our model through a series of ablation studies.

j) Primary Result: We first investigate the ability to navigate to the intended destination for a given natural language instruction. Table II reports the overall accuracy of our model for both the single- and multi-sentence settings. We report two statistics with our model (vDev and vTest) in order to directly compare against existing work.[3]

As we can see from Table II, we surpass state-of-the-art results on the single-sentence route instruction task (for both vDev and vTest settings), despite using no linguistic knowledge or resources. Our multi-sentence accuracy, which is working with a really small amount of training data (a few hundred paragraph pairs), is competitive with state-of-the-art and outperforms several previous methods that employ specialized linguistic models in the form of semantic parsers, lexicons, and re-rankers. Our multi-sentence inference strategy performs beam search based on log-loss sequentially over each sentence using the single-sentence model. It should be possible to improve this, for example, with an additional re-ranking step, which has been shown to enhance accuracy in this domain [16, 1, 2], and with deep reinforcement learning to allow for path corrections. This a direction of current work.

[3]Subsequent evaluations are on vDev unless otherwise noted.

TABLE II: Overall accuracy, where bold denotes state-of-the-art

Method	Single-sentence	Multi-sentence
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, Das, and Petrov (2014)	64.36	**35.44**
Our model (vDev)	68.52	25.36
Our model (vTest)	**70.03**	29.49
Human Followers	—	69.64

k) Model Stability: As a consequence of training, the accuracy of our model will vary across different training runs. In order to understand the stability of our model, we performed a set of five train+test experiments using both of the aforementioned validation approaches. Table III reports the average accuracy and standard deviation over these five runs for our models and the previous state-of-the-art [2] (which is comparable to our vTest setting). Our model achieves state-of-the-art accuracy and is more stable than that of Artzi et al. [2].

TABLE III: The stability of our model

	Our model (vDev)	Our model (vTest)	Artzi et al. 2013
Single	62.74 (1.02)	66.27 (0.29)	65.28 (5.09)
Multi	19.15 (0.37)	23.05 (1.52)	31.93 (3.26)

l) Ensemble Ablation: As used in previous related work [27, 29, 32], we evaluate our method using an ensemble of randomly initialized models. While we used five models for our primary evaluation, here we show the performance curve with ensembles of different size — we see that our model performance can further improve with larger ensembles.

TABLE IV: Accuracy with ensemble size (vDev)

	1	5	10
Single-sentence	63.79	68.52	**69.98**
Multi-sentence	19.37	25.36	**26.07**

TABLE V: Accuracy with ensemble size (vTest)

	1	5	10
Single-sentence	66.64	70.03	**71.05**
Multi-sentence	23.50	29.49	**30.34**

m) Per-Map Results: Next, we report per-map results for both single- and multi-sentence (Table VI).

TABLE VI: Per-map results

	Grid	L	Jelly	Total
Single-sentence	65.68	65.14	73.24	68.52
Multi-sentence	27.68	23.73	24.79	25.36

n) Distance Evaluation: Our evaluation required that the action sequence reach the exact desired destination. It is of interest to consider how close the model gets to the destination when it is not reached. Table VII displays the fraction of test data that reach within d nodes of the destination. Often, the final position reached by the model is quite close to the desired destination.

TABLE VII: Accuracy as a function of distance from the intended destination (without orientation)

Distance (d)	0	1	2	3
Single-sentence	71.61	86.16	92.49	95.43
Multi-sentence	25.36	41.60	56.98	71.80

o) Bidirectionality Ablation: We consider the advantage of using a bidirectional encoder by training an alternative model that uses only a unidirectional (forward) encoder. As shown in Table VIII, the bidirectional encoder significantly improves accuracy.

TABLE VIII: Removing bidirectionality

	Bidirectional	Unidirectional
Single-sentence	68.52	66.57
Multi-sentence	25.36	23.21

Having shown the performance of our model, we perform further analyses and ablations to better understand the importance of our encoder and aligner. For these experiments, we evaluate the unidirectional version of our model for time efficiency.

p) Encoder Ablation: We further evaluate the benefit of encoding the input sentence and consider an alternative model that directly feeds word vectors as randomly initialized embeddings into the decoder and relies on the alignment model to choose focus words. Table IX presents the results with and without the encoder and demonstrates that there is a significant gain in encoding the input sentence into its context representation. We believe the difference is due to the RNN's ability to incorporate sentence-level information into the word's representation as it processes the prefix of the sentence sequentially. This advantage helps resolve ambiguities, such as "turn right before ..." versus "turn right after ...".

TABLE IX: Accuracy with and without encoder

	Encoder	No Encoder
Single-sentence	66.57	59.66
Multi-sentence	23.21	16.10

TABLE X: Accuracy with and without alignment

	Aligner	No Aligner
Single-sentence	66.57	65.49
Multi-sentence	23.21	22.07

q) Aligner Ablation: Our model utilizes alignment in the decoder as a means of focusing on word "regions" that are more salient. We consider the effect of alignment by training an alternative model in which the context vector z_t is an unweighted average (Eqn. 5). As shown in the Table X, alignment does improve the accuracy of the resulting action sequence, though the improvement is not as significant as that from the encoder.

VII. Conclusion

We presented an end-to-end, sequence-to-sequence approach to map natural language navigational instructions to action plans, using a bidirectional LSTM-RNN model with a multi-level aligner. We evaluate our model on a benchmark route instruction dataset and demonstrate that it achieves a new state-of-the-art on single-sentence execution and yields competitive results on the more challenging multi-sentence domain, despite working with very small training datasets and using no linguistic knowledge or resources. We further performed a number of ablation studies to elucidate the effects of the primary components of our model. We are currently evaluating other extensions to our model to improve single- and multi-sentence learning, including the use of word embeddings, a neural re-ranker over global action sequences, deep reinforcement learning, and adding synthetic data.

Acknowledgments

We thank Yoav Artzi, David Chen, Oriol Vinyals, and Kelvin Xu for their helpful comments.

References

[1] Yoav Artzi and Luke Zettlemoyer. Weakly supervised learning of semantic parsers for mapping instructions to actions. *TACL*, 1:49–62, 2013.
[2] Yoav Artzi, Dipanjan Das, and Slav Petrov. Learning compact lexicons for ccg semantic parsing. In *EMNLP*, 2014.
[3] J. Ba, V. Mnih, and K. Kavukcuoglu. Multiple object recognition with visual attention. *arXiv:1412.7755*, 2014.
[4] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*, 2014.
[5] David L Chen. Fast online lexicon learning for grounded language acquisition. In *ACL*, 2012.
[6] David L. Chen and Raymond J. Mooney. Learning to interpret natural language navigation instructions from observations. In *AAAI*, 2011.
[7] Xinlei Chen and C Lawrence Zitnick. Mind's eye: A recurrent visual representation for image caption generation. In *CVPR*, 2015.
[8] Kyunghyun Cho, Bart van Merrienboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using RNN encoder-decoder for statistical machine translation. In *EMNLP*, Octobar 2014.

[9] Jeff Donahue, Lisa Anne Hendricks, Sergio Guadarrama, Marcus Rohrbach, Subhashini Venugopalan, Kate Saenko, and Trevor Darrell. Long-term recurrent convolutional networks for visual recognition and description. *arXiv:1411.4389*, 2014.

[10] A. Graves, M. Abel-rahman, and G. Hinton. Speech recognition with deep recurrent neural networks. In *ICASSP*, 2013.

[11] S. Harnad. The symbol grounding problem. *Physica D*, 42: 335–346, 1990.

[12] Sachi Hemachandra, Felix Duvallet, Thomas M. Howard, Nicholas Roy, Anthony Stentz, and Matthew R. Walter. Learning models for following natural language directions in unknown environments. In *ICRA*, May 2015.

[13] S. Hochreiter and J Schmidhuber. Long short-term memory. *Neural Computation*, 9(8), November 1997. doi: 10.1162/neco.1997.9.8.1735.

[14] A. Karpathy and L. Fei-Fei. Deep visual-semantic alignments for generating image descriptions. In *CVPR*, 2015.

[15] Joohyun Kim and Raymond J Mooney. Unsupervised pcfg induction for grounded language learning with highly ambiguous supervision. In *EMNLP*, pages 433–444, 2012.

[16] Joohyun Kim and Raymond J. Mooney. Adapting discriminative reranking to grounded language learning. In *ACL*, 2013.

[17] Diederik Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *ICLR*, 2015.

[18] Ryan Kiros, Ruslan Salakhutdinov, and Richard S. Zemel. Unifying visual-semantic embeddings with multimodal neural language models. *arXiv:1411.2539*, 2014. URL http://arxiv.org/abs/1411.2539.

[19] Thomas Kollar, Stefanie Tellex, Deb Roy, and Nicholas Roy. Toward understanding natural language directions. In *HRI*, 2010.

[20] M. MacMahon, B. Stankiewicz, and B. Kuipers. Walk the talk: Connecting language, knowledge, and action in route instructions. In *AAAI*, 2006.

[21] Junhua Mao, Wei Xu, Yi Yang, Jiang Wang, and Alan Yuille. Deep captioning with multimodal recurrent neural networks (m-RNN). *arXiv:1412.6632*, 2014.

[22] Cynthia Matuszek, Dieter Fox, and Karl Koscher. Following directions using statistical machine translation. In *HRI*, 2010.

[23] V. Mnih, N. Hees, A. Graves, and K. Kavukcuoglu. Recurrent models of visual attention. In *NIPS*, December 2014.

[24] Raymond J. Mooney. Learning to connect language and perception. In *AAAI*, 2008.

[25] R. Pascanu, C. Gulcehre, K. Cho, and Y. Bengio. How to construct deep recurrent neural networks. *arXiv:1312.6026*, 2014.

[26] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *J. Machine Learning Research*, 15(1):1929–1958, 2014.

[27] I. Sutskever, O. Vinyals, and Quoc V. Lee. Sequence to sequence learning with neural networks. In *NIPS*, December 2014.

[28] 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 *AAAI*, 2011.

[29] Oriol Vinyals, Alexander Toshev, Samy Bengio, and Dumitru Erhan. Show and tell: A neural image caption generator. In *CVPR*, June 2015.

[30] Terry Winograd. *Proceedures as a Representation for Data in a Computer Program for Understanding Natural Language*. PhD thesis, MIT, 1970.

[31] Kelvin Xu, Jimmy Ba, Ryan Kiros, Kyunghyun Cho, Aaron Courville, Ruslan Salakhutdinov, Richard Zemel, and Yoshua Bengio. Show, attend and tell: Neural image caption generation with visual attention. In *ICML*, 2015.

[32] Wojciech Zaremba, Ilya Sutskever, and Oriol Vinyals. Recurrent neural network regularization. *arXiv preprint arXiv:1409.2329*, 2014.