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

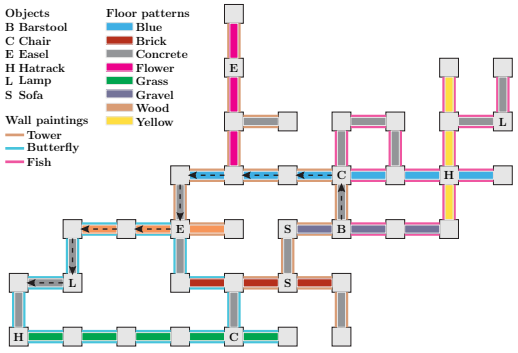
Abstract

We propose a neural sequence-to-sequence model for direction following, a task that is essential to realizing effective autonomous agents. Our alignment-based encoder-decoder model with long short-term memory recurrent neural networks (LSTM-RNN) translates natural language instructions to action sequences based upon a representation of the observable world state. We introduce a multi-level aligner that empowers our model to focus on sentence "regions" salient to the current world state by using multiple abstractions of the input sentence. In contrast to existing methods, our model uses no specialized linguistic resources (e.g., parsers) or task-specific annotations (e.g., seed lexicons). It is therefore generalizable, yet still achieves the best results reported to-date on a benchmark single-sentence dataset and competitive results for the limited-training multi-sentence setting. We analyze our model through a series of ablations that elucidate the contributions of the primary components of our model.

Introduction

Robots must be able to understand and successfully execute natural language navigational instructions if they are to work seamlessly alongside people. For example, someone using a voice-commandable wheelchair might direct it to "Take me to the room across from the kitchen," or a soldier may command a micro aerial vehicle to "Fly down the hallway into the second room on the right." However, interpreting such free-form instructions (especially in unknown environments) is challenging due to their ambiguity and complexity, such as uncertainty in their interpretation (e.g., which hallway does the instruction refer to), long-term dependencies among both the instructions and the actions, differences in the amount of detail given, and the diverse ways in which the language can be composed. Figure 1 presents an example instruction that our method successfully follows.

Previous work in this domain (Chen and Mooney 2011; Chen 2012; Kim and Mooney 2012; 2013; Artzi and Zettlemoyer 2013; Artzi, Das, and Petrov 2014) largely requires specialized resources like semantic parsers, seed lexicons, and re-rankers to interpret ambiguous, free-form natural language instructions. In contrast, the goal of our work is to learn to map instructions to actions in an end-to-end fashion that assumes no prior linguistic knowledge. Instead, our model learns the meaning of all the words, spatial relations, syntax, and compositional semantics from just the raw training sequence pairs, and learns to to translate the free-form instructions to an executable action sequence.



Place your back against the wall of the "T" intersection. Go forward one segment to the intersection with the blue-tiled hall. This interesction [sic] contains a chair. Turn left. Go forward to the end of the hall. Turn left. Go forward one segment to the intersection with the wooden-floored hall. This intersection conatains [sic] an easel. Turn right. Go forward two segments to the end of the hall. Turn left. Go forward one segment to the intersection containing the lamp. Turn right. Go forward one segment to the empty corner.

Figure 1: An example of a route instruction-path pair in one of the virtual worlds from MacMahon, Stankiewicz, and Kuipers (2006) with colors that indicate floor patterns and wall paintings, and letters that indicate different objects. Our method successfully infers the correct path for this instruction.

We propose a recurrent neural network with long short-term memory (LSTM) (Hochreiter and Schmidhuber 1997) to both encode the navigational instruction sequence bidirectionally and to decode the representation to an action sequence, based on a representation of the current world state. LSTMs are well-suited to this task, as they have been shown to be effective in learning the temporal de-

pendencies that exist over such sequences, especially for the tasks of image captioning, machine translation, and natural language generation (Kiros, Salakhutdinov, and Zemel 2014; Donahue et al. 2014; Chen and Zitnick 2015; Karpathy and Fei-Fei 2015; Vinyals et al. 2015; Sutskever, Vinyals, and Lee 2014; Rush, Chopra, and Weston 2015; Wen et al. 2015). Additionally, we learn the correspondences between words in the input navigational instruction and actions in the output sequence using an alignment-based LSTM (Bahdanau, Cho, and Bengio 2014; Xu et al. 2015). Standard alignment methods only consider high-level abstractions of the input (language), which sacrifices information important to identifying these correspondences. Instead, we introduce a *multi-level* aligner that empowers the model to use both high- and low-level input representations and, in turn, improves the accuracy of the inferred directions.

We evaluate our model on a benchmark navigation dataset (MacMahon, Stankiewicz, and Kuipers 2006) and achieve the best results reported to-date on the single-sentence task (which contains only 2000 training pairs), without using any additional resources such as semantic parsers, seed lexicons, or rerankers used in previous work. On the multi-sentence task of executing a full paragraph, where the amount of training pairs is even smaller (just a few hundred pairs), our model performs better than several existing methods and is competitive with the state-of-the-art, all of which use specialized linguistic resources, extra annotation, or reranking. We perform a series of ablation studies in order to analyze the primary components of our model, including the encoder, multi-level representations, alignment, and bidirectionality.

Related Work

A great deal of attention has been paid of late to algorithms that allow robots and autonomous agents to follow free-form navigational route instructions (MacMahon, Stankiewicz, and Kuipers 2006; Kollar et al. 2010; Chen and Mooney 2011; Chen 2012; Kim and Mooney 2012; 2013; Kong et al. 2014; Hemachandra et al. 2015). These methods solve what Harnad (1990) refers to as the symbol grounding problem, that of associating linguistic elements with their corresponding manifestation in the external world. Initial research in natural language symbol grounding focused on manually-prescribed mappings between language and sets of predefined environment features and actions (Winograd 1970; MacMahon, Stankiewicz, and Kuipers 2006). 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 (Mooney 2008). These methods represent natural language grounding in terms of manually defined linguistic, spatial, and semantic features (Kollar et al. 2010; Matuszek, Fox, and Koscher 2010; Tellex et al. 2011). They learn the model parameters from natural language corpora, often requiring expensive 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, Fox, and Koscher (2010) assume no prior linguistic knowledge and employ a general-purpose supervised semantic parser learner. Alternatively, Chen and Mooney (2011) parse free-form route instructions into formal action specifications that a robot control process can then execute. 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 (2012) frame grounded language learning as probabilistic context free grammar (PCFG) induction and use learned lexicons (Chen and Mooney 2011) to control the space of production rules, which allows them to scale PCFGs to the navigation domain. Kim and Mooney (2013) improve upon the accuracy by adding a subsequent re-ranking step that uses a weakly supervised discriminative classifier. Meanwhile, Artzi and Zettlemoyer (2013) learn a combinatory categorical grammar (CCG)-based semantic parser to convert free-form navigational instructions to their manifestation in a lambda-calculus representation. As with Kim and Mooney (2013), they improve upon the accuracy through re-ranking. Artzi, Das, and Petrov (2014) 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. (2010) 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. (2011) later propose 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 using any prior linguistic structure, resources, or annotation, which improves generalizability. Our method is inspired by the recent success of such sequence-to-sequence methods for machine translation (Sutskever, Vinyals, and Lee 2014; Bahdanau, Cho, and Bengio 2014; Cho et al. 2014), image and video caption synthesis (Kiros, Salakhutdinov, and Zemel 2014; Mao et al. 2014; Donahue et al. 2014; Vinyals et al. 2015; Chen and Zitnick 2015; Karpathy and Fei-Fei 2015), and natural language generation (Rush, Chopra, and Weston 2015; Wen et al. 2015), which similarly adopt an encoder-decoder approach. Our model encodes the input free-form route instruction and then decodes the embedding to identify the corresponding output action sequence based upon the lo-

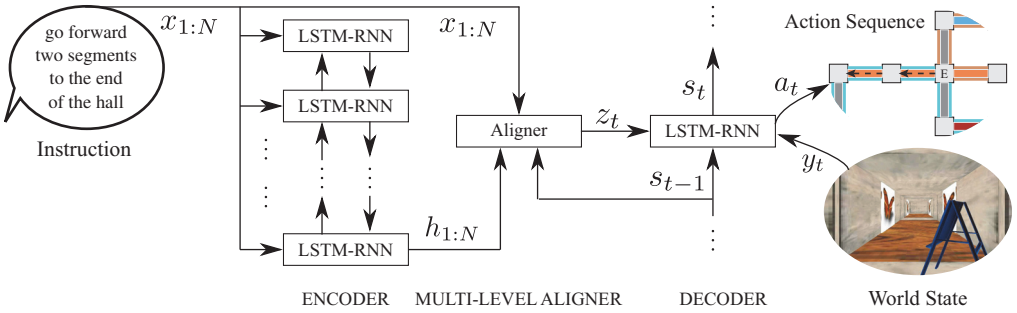


Figure 2: Our encoder-aligner-decoder model with multi-level alignment

cal, observable world state (which we treat as a third sequence type that we add as an extra connection to every decoder step). Moreover, our decoder also includes alignment to focus on the portions of the sentence relevant to the current action, a technique that has proven effective in machine translation (Bahdanau, Cho, and Bengio 2014) and machine vision (Mnih et al. 2014; Ba, Mnih, and Kavukcuoglu 2014; Xu et al. 2015). However, unlike the standard alignment techniques, our model learns to align based not only on the high-level input abstraction, but also the low-level representation of the input instruction, which improves performance. Recently, Andreas and Klein (2015) use a conditional random field model to learn alignment between instructions and actions; our LSTM-based aligner performs substantially better than this approach.

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 variable length natural language instruction, $a^{(i)}$ is the corresponding action sequence, and $y^{(i)}$ is the observable environment representation. The model learns to produce the correct action sequence $a^{(i)}$ given a previously unseen $(x^{(i)}, y^{(i)})$ pair. The challenges arise from the fact that the instructions are free-form and complex, contain numerous spelling and grammatical errors, and are ambiguous in their meaning. Further, the model is only aware of the local environment in the agent's line-of-sight.

In this paper, we consider the route instruction dataset generated by MacMahon, Stankiewicz, and Kuipers (2006). The data includes free-form route instructions and their corresponding action sequences within three different virtual worlds. The environments (Fig. 1) consist of interconnected hallways with a pattern (grass, brick, wood, gravel, blue, flower, or yellow octagons) on each hallway floor, a painting (butterfly, fish, or Eiffel Tower) on the walls, and 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 subsequent access to the map. Each instruction was then given to several human followers who were tasked with navigating in the virtual world without a map, and their paths were recorded. Many of the raw instructions include spelling and grammatical errors, others are incorrect (e.g., misusing "left" and "right"), approximately 10% of the single sentences have no associated action, and 20% have no feasible path.

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 $a_{1:T} = (a_1, a_2, \ldots, a_T)$ is the action sequence, y_t is the world state at time t, and $x_{1:N} = (x_1, x_2, \ldots, x_N)$ is the natural language instruction,

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

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

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

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

$$z_t = c(h_1, h_2, \ldots h_N), \quad (2b)$$

where h_j is the encoder hidden state for word $j \in \{1, \ldots, N\}$, and f and c are nonlinear functions that we define shortly. Next, the context vector z_t (computed by the aligner) encodes the language instruction at time $t \in \{1, \ldots, T\}$. Next, another RNN decodes the context vector z_t to arrive at the desired likelihood (1)

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

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

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.

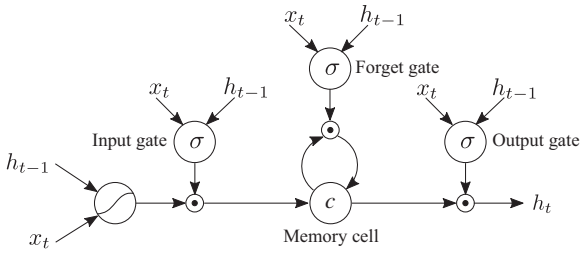


Figure 3: Long Short-term Memory (LSTM) unit.

Our model (Fig. 2) 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.

Encoder Our encoder takes as input the natural language route instruction represented as a sequence $x_{1:N} = (x_1, x_2, \ldots, x_N)$, 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 $h_{1:N} = (h_1, h_2, \ldots, h_N)$, where the annotation h_j summarizes the words up to and including x_j.

We adopt an LSTM encoder architecture (Fig. 3) similar to that of Graves, Abdel-rahman, and Hinton (2013),

$$\begin{pmatrix} i_j^e \\ f_j^e \\ o_j^e \\ g_j^e \end{pmatrix} = \begin{pmatrix} \sigma \\ \sigma \\ \sigma \\ \tanh \end{pmatrix} T^e \begin{pmatrix} x_j \\ h_{j-1} \end{pmatrix} \tag{4a}$$

$$c_j^e = f_j^e \odot c_{j-1}^e + i_j^e \odot g_j^e \tag{4b}$$

$$h_j = o_j^e \odot \tanh(c_j^e) \tag{4c}$$

where T^e is an affine transformation, σ is the logistic sigmoid that restricts its input to $[0, 1]$, i_j^e, f_j^e, and o_j^e are the input, output, and forget gates of the LSTM, respectively, and c_j^e is the memory cell activation vector. The memory cell c_j^e summarizes the LSTM's previous memory c_{j-1}^e 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 (Graves, Abdel-rahman, and Hinton 2013; Bahdanau, Cho, and Bengio 2014; Cho et al. 2014). 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.

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 based only on the hidden annotations h_j, we found that also including the original input word x_j in the aligner improves performance. This multi-level representation 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. By adding x_j, the model offsets information that is lost in the high-level abstraction of the instruction. Intuitively, the model is able to better match the salient words in the input sentence (e.g., "easel") directly to the corresponding landmarks in the current world state y_t used in the decoder. The context vector then takes the form

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

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

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

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.

Decoder Our architecture uses an LSTM decoder (Fig. 3) 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 $P_{a,t} = P(a_t | a_{1:t-1}, y_t, x_{1:N})$ over the next action (3), represented as a deep output layer (Pascanu et al. 2014)

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

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

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

$$q_t = L_0(E y_t + L_s s_t + L_z z_t) \tag{8d}$$

$$P_{a,t} = \text{softmax}(q_t) \tag{8e}$$

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

Training We train the encoder and decoder models so as to predict the action sequence $a_{1:T}^*$ according to Equation 1 for a given instruction $x_{1:N}$ and world state $y_{1:T}$ from the

training corpora. We use the negative log-likelihood of the demonstrated action at each time step t as our loss function,

$$L = -\log P(a_t^*|y_t, x_{1:N}). \quad (9)$$

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

Inference Having trained the model, we generate action sequences by finding the maximum a posteriori actions under the learned model (1). One action sequence is completed when the "stop" action is emitted. 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 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 list of previous k best hypotheses. As a common denoising method in deep learning (Sutskever, Vinyals, and Lee 2014; Zaremba, Sutskever, and Vinyals 2014; Vinyals et al. 2015), we perform inference over an ensemble of randomly initialized models.[2]

Experimental Setup

Dataset We train and evaluate our model using the publicly available SAIL route instruction dataset collected by MacMahon, Stankiewicz, and Kuipers (2006). We use the raw data in its original form (e.g., we do not correct any spelling errors). 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. See corpus statistics in Chen and Mooney (2011).

World State The world state y_t encodes the local, 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 represent the world state as a concatenation of a simple bag-of-words vector for each direction (forward, left, and right). The choice of bag-of-words representation avoids manual domain-specific feature-engineering and the combinatoriality of modeling exact world configurations.

Evaluation Metrics We evaluate our end-to-end model on both the single-sentence and multi-sentence versions of the corpus. For the single-sentence task, 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 final orientation, as previous work does. However, this setting is still more challenging than single-sentence due to cascading errors over individual sentences.

[1] We use a beam width of 10 to be consistent with previous work.

[2] At each time step t, we generate actions using the avg. of the posterior likelihoods of 10 ensemble models, as in previous work.

Table 1: Overall accuracy (state-of-the-art in bold)

Method	Single-sent	Multi-sent
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**
Andreas and Klein (2015)	59.60	–
Our model (vDev)	69.98	26.07
Our model (vTest)	**71.05**	30.34

Training Details We follow the same procedure as Chen and Mooney (2011), 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 (10%) sets, the latter of which is used to tune hyperparameters.[3] 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 found Adam (Kingma and Ba 2015) to be very effective for training with this dataset. The training usually converges within 50 epochs. We performed early stopping based on the validation task metric. Similar to previous work (Xu et al. 2015), we found that the validation log-likelihood is not well correlated with the task metric.

Results and Analysis

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

Table 2: Model components ablations

	Full Model	High-level Aligner	No Aligner	Unidirectional	No Encoder
Single-sentence	69.98	68.09	68.05	67.44	61.63
Multi-sentence	26.07	24.79	25.04	24.50	16.67

[3] We only tuned the no. of hidden units and the drop-out rate (Srivastava et al. 2014; Zaremba, Sutskever, and Vinyals 2014).

Primary Result We first investigate the ability to navigate to the intended destination for a given natural language instruction. Figure 1 illustrates an output example for which our model successfully executes the input natural language instruction. Table 1 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 with existing work.[4]

As we can see from Table 1, 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 use additional, specialized resources in the form of semantic parsers, logical-form lexicons, and re-rankers.[5] We note that our model yields good results using only greedy search (beam-width of one). For vDev, we achieve 68.05 on single-sentence and 23.93 on multi-sentence, while for vTest, we get 70.56 on single-sentence and 27.91 on multi-sentence.

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 3 displays the fraction of test results that reach within d nodes of the destination. Often, the method produces action sequences that reach points close to the desired destination.

Table 3: Accuracy as a function of distance from destination

Distance (d)	0	1	2	3
Single-sentence	71.73	86.62	92.86	95.74
Multi-sentence	26.07	42.88	59.54	72.08

Multi-level Aligner Ablation Unlike most existing methods that align based only on the hidden annotations h_j, we adopt a different approach by also including the original input word x_j (Eqn. 5). As shown in Table 2, the multi-level representation ("Full Model") significantly improves performance over a standard aligner ("High-level Aligner"). Figure 4 visualizes the alignment of words to actions in the map environment for several sentences from the instruction paragraph depicted in Figure 1.

Aligner Ablation Our model utilizes alignment in the decoder as a means of focusing on word "regions" that are more salient to the current world state. We analyze the effect of the learned alignment by training an alternative model in which the context vector z_t is an unweighted average (Eqn. 5). As shown in the Table 2, learning the alignment does improve the accuracy of the resulting action sequence. Note that the "No Aligner" model still maintains all connections between the instruction and actions, but is just using non-learned, uniform weights.

Bidirectionality Ablation We train an alternative model that uses only a unidirectional (forward) encoder. As shown in Table 2, the bidirectional encoder ("Full Model") significantly improves accuracy.

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 the salient words. Table 2 presents the results with and without the encoder and demonstrates that there is a substantial 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 sentence sequentially (in both directions). This advantage helps resolve ambiguities, such as "turn right before ..." versus "turn right after ...".

Conclusion

We presented an end-to-end, sequence-to-sequence approach to mapping natural language navigational instructions to action plans given the local, observable world state, using a bidirectional LSTM-RNN model with a multi-level aligner. We evaluated our model on a benchmark route instruction dataset and demonstrates 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 specialized linguistic knowledge or resources. We further performed a number of ablation studies to elucidate the contributions of our primary model components.

References

Andreas, J., and Klein, D. 2015. Alignment-based compositional semantics for instruction following. In *EMNLP*.

Artzi, Y., and Zettlemoyer, L. 2013. Weakly supervised learning of semantic parsers for mapping instructions to actions. *TACL* 1:49–62.

Artzi, Y.; Das, D.; and Petrov, S. 2014. Learning compact lexicons for ccg semantic parsing. In *EMNLP*.

Ba, J.; Mnih, V.; and Kavukcuoglu, K. 2014. Multiple object recognition with visual attention. *arXiv:1412.7755*.

Bahdanau, D.; Cho, K.; and Bengio, Y. 2014. Neural machine translation by jointly learning to align and translate. *arXiv:1409.0473*.

Chen, D. L., and Mooney, R. J. 2011. Learning to interpret natural language navigation instructions from observations. In *AAAI*.

Chen, X., and Zitnick, C. L. 2015. Mind's eye: A recurrent visual representation for image caption generation. In *CVPR*.

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

[5] Note that with no ensemble, we are still state-of-the-art on single-sentence and better than all comparable approaches on multi-sentence: Artzi et al. (2013, 2014) use extra annotations with a logical-form lexicon and Kim and Mooney (2013) use discriminative reranking, techniques that are orthogonal to our approach and should likely improve our results as well.

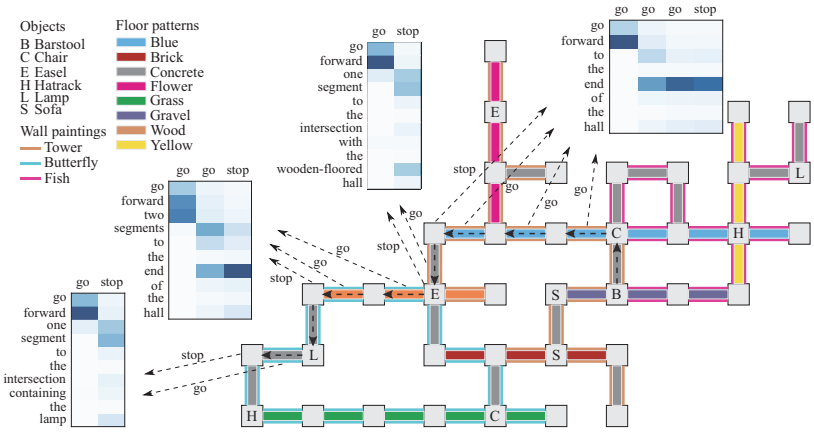


Figure 4: Visualization of the alignment between words to actions in a map for a multi-sentence instruction.

Chen, D. L. 2012. Fast online lexicon learning for grounded language acquisition. In *ACL*.

Cho, K.; van Merrienboer, B.; Gulcehre, C.; Bahdanau, D.; Bougares, F.; Schwenk, H.; and Bengio, Y. 2014. Learning phrase representations using RNN encoder-decoder for statistical machine translation. In *EMNLP*.

Donahue, J.; Hendricks, L. A.; Guadarrama, S.; Rohrbach, M.; Venugopalan, S.; Saenko, K.; and Darrell, T. 2014. Long-term recurrent convolutional networks for visual recognition and description. *arXiv:1411.4389*.

Graves, A.; Abdel-rahman, M.; and Hinton, G. 2013. Speech recognition with deep recurrent neural networks. In *ICASSP*.

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

Hemachandra, S.; Duvallet, F.; Howard, T. M.; Roy, N.; Stentz, A.; and Walter, M. R. 2015. Learning models for following natural language directions in unknown environments. In *ICRA*.

Hochreiter, S., and Schmidhuber, J. 1997. Long short-term memory. *Neural Computation* 9(8).

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

Kim, J., and Mooney, R. J. 2012. Unsupervised PCFG induction for grounded language learning with highly ambiguous supervision. In *EMNLP*, 433–444.

Kim, J., and Mooney, R. J. 2013. Adapting discriminative reranking to grounded language learning. In *ACL*.

Kingma, D., and Ba, J. 2015. Adam: A method for stochastic optimization. In *ICLR*.

Kiros, R.; Salakhutdinov, R.; and Zemel, R. S. 2014. Unifying visual-semantic embeddings with multimodal neural language models. *arXiv:1411.2539*.

Kollar, T.; Tellex, S.; Roy, D.; and Roy, N. 2010. Toward understanding natural language directions. In *HRI*.

Kong, C.; Lin, D.; Bansal, M.; Urtasun, R.; and Fidler, S. 2014. What are you talking about? text-to-image coreference. In *Proceedings of CVPR*.

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

Mao, J.; Xu, W.; Yang, Y.; Wang, J.; and Yuille, A. 2014. Deep captioning with multimodal recurrent neural networks (m-RNN). *arXiv:1412.6632*.

Matuszek, C.; Fox, D.; and Koscher, K. 2010. Following directions using statistical machine translation. In *HRI*.

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

Mooney, R. J. 2008. Learning to connect language and perception. In *AAAI*.

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

Rush, A. M.; Chopra, S.; and Weston, J. 2015. A neural attention model for abstractive sentence summarization. In *EMNLP*.

Srivastava, N.; Hinton, G.; Krizhevsky, A.; Sutskever, I.; and Salakhutdinov, R. 2014. Dropout: A simple way to prevent neural networks from overfitting. *J. Machine Learning Research* 15(1):1929–1958.

Sutskever, I.; Vinyals, O.; and Lee, Q. V. 2014. Sequence to sequence learning with neural networks. In *NIPS*.

Tellex, S.; Kollar, T.; Dickerson, S.; Walter, M. R.; Banerjee, A. G.; Teller, S.; and Roy, N. 2011. Understanding natural language commands for robotic navigation and mobile manipulation. In *AAAI*.

Vinyals, O.; Toshev, A.; Bengio, S.; and Erhan, D. 2015. Show and tell: A neural image caption generator. In *CVPR*.

Wen, T.-H.; Gašić, M.; Mrkšić, N.; Su, P.-H.; Vandyke, D.; and Young, S. 2015. Semantically conditioned LSTM-based natural language generation for spoken dialogue systems. In *EMNLP*.

Winograd, T. 1970. *Proceedures as a Representation for Data in a Computer Program for Understanding Natural Language*. Ph.D. Dissertation, MIT.

Xu, K.; Ba, J.; Kiros, R.; Cho, K.; Courville, A.; Salakhutdinov, R.; Zemel, R.; and Bengio, Y. 2015. Show, attend and tell: Neural image caption generation with visual attention. In *ICML*.

Zaremba, W.; Sutskever, I.; and Vinyals, O. 2014. Recurrent neural network regularization. *arXiv:1409.2329*.