

# Interpreting Natural Language Instructions Using Language, Vision, and Behavior

LUCIANA BENOTTI, Universidad Nacional de Córdoba, Argentina/CONICET, Argentina

TESSA LAU, Savioke, Inc., Sunnyvale, CA, USA

MARTÍN VILLALBA, University of Potsdam, Germany/Universidad Nacional de Córdoba, Argentina

We define the problem of automatic instruction interpretation as follows: Given a natural language instruction, can we automatically predict what an instruction follower (for example, a robot) should do in the environment in order to follow that instruction? Previous approaches to automatic instruction interpretation have required either extensive domain dependent rule writing or extensive manually annotated corpora. This paper presents a novel approach that leverages a large amount of unannotated, easy-to-collect data from humans interacting in a game-like environment. Our approach uses an automatic annotation phase based on artificial intelligence planning, for which two different annotation strategies are compared, one based on behavioral information and the other based on visibility information. The resulting annotations are used as training data for different automatic classifiers. This algorithm is based on the intuition that the problem of interpreting a situated instruction can be cast as a classification problem of choosing among the actions that are possible in the situation. Classification is done by combining language, vision, and behavior information. Our empirical analysis shows that machine learning classifiers achieve 77% accuracy on this task on available English corpora and 74% on similar German corpora. Finally, the inclusion of human feedback in the interpretation process is shown to boost performance to 92% for the English corpus and 90% for the German corpus.

Categories and Subject Descriptors: I.2.7 [Natural Language Processing]: Language Understanding

General Terms: Design, Algorithms, Performance

Additional Key Words and Phrases: natural language interpretation, multi-modal understanding, action recognition, visual feedback, situated virtual agent, unsupervised learning

## ACM Reference Format:

Benotti, L., Lau, T., and Villalba, M. 2013. Interpreting natural language instructions using language, vision and behavior *ACM Trans. Interact. Intell. Syst.* 9, 4, Article 39 (March 2010), 23 pages.

DOI: <http://dx.doi.org/10.1145/0000000.0000000>

## 1. INTRODUCTION AND MOTIVATION

We define the problem of *automatic instruction interpretation* as follows: given an instruction in natural language, can we automatically predict what an instruction follower (for example, a robot) should do in the environment in order to follow that instruction? The problem of interpreting instructions in natural language has been studied since the early days of artificial intelligence [Winograd 1972]. Mapping instructions into automatically executable actions would enable the creation of natural language interfaces to many applications such as web pages [Lau et al. 2009], operating sys-

---

Author's addresses: L. Benotti, Famaf, Medina Allende S/N, Córdoba, Argentina; T. Lau, Savioke, Inc., Sunnyvale, CA, USA; M. Villalba, University of Potsdam Karl-Liebknecht-Str 24-25, D-14476 Potsdam, Germany.

Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies show this notice on the first page or initial screen of a display along with the full citation. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, to republish, to post on servers, to redistribute to lists, or to use any component of this work in other works requires prior specific permission and/or a fee. Permissions may be requested from Publications Dept., ACM, Inc., 2 Penn Plaza, Suite 701, New York, NY 10121-0701 USA, fax +1 (212) 869-0481, or [permissions@acm.org](mailto:permissions@acm.org).

© 2010 ACM 2160-6455/2010/03-ART39 \$15.00

DOI: <http://dx.doi.org/10.1145/0000000.0000000>

tems [Branavan et al. 2009], games [Orkin and Roy 2009] and robots [Kollar et al. 2010]. In this article, we focus on the task of navigation and manipulation of objects in a spatial environment [Vogel and Jurafsky 2010; Chen and Mooney 2011].

Current symbolic approaches to the problem, which require intensive rule authoring to be fit for a new task [Dzikovska et al. 2008], are brittle due to the variance in natural language present in instructions. Current statistical approaches, on the other hand, require extensive manual annotations of the corpora used for training [MacMahon et al. 2006; Matuszek et al. 2010; Gorniak and Roy 2007; Rieser and Lemon 2010]. Manual annotation and rule authoring by natural language engineering experts are bottlenecks for developing conversational systems for new domains.

This paper proposes a fully automated approach to interpreting natural language instructions to complete a task in a spatial environment based on unsupervised recordings of human-human interactions performing that task in that spatial environment. Given unannotated corpora collected from humans following other humans' instructions, our system automatically segments the corpus into labeled training data for a classification algorithm. Our interpretation algorithm is based on the observation that similar instructions uttered in similar contexts should lead to similar actions being taken in the spatial environment. Given a previously unseen instruction, our system outputs actions that can be directly executed in the spatial environment world, based on what humans did when given similar instructions in the past. In order to do this the system first infers a formal navigation plan for each observed instruction based on a human reaction to it. This planning step is used as a normalization in order to overcome the noise present in our data. Using this as supervision, it then learns classifiers that can map novel instructions into navigation plans executable by a (simulated) robot.

The effect of including instructor feedback in the interpretation process is investigated with the result that it can boost performance to 92% for the English corpus and 90% for the German corpus. The instructor provides feedback to the system by rephrasing misinterpreted instructions and giving the system the opportunity to reinterpret the instruction.

The interpretation approach proposed combines different sources of information—position, area of visibility, language and spatial actions—to first identify the actions that are possible and salient in the current situation, and then choose the action that is intended by the instruction giver. The identification of the salient actions greatly reduces the complexity of the interpretation task allowing for it to be cast as a classification problem that can scale to complex spatial environments.

Specifically, we make these contributions:

- A novel model that leverages a large amount of unannotated, easy-to-collect data from humans interacting in a game-like environment. The model uses an automatic annotation phase based on artificial intelligence planning, for which two different annotation strategies are compared, one based on behavioral information and the other based on visibility information.
- A classification-based approach that combines different sources of information—position, area of visibility, language and spatial actions—to first identify the actions that are possible and salient in the current situation, and then choose the action that is intended by the instruction giver. This approach achieves 77% accuracy on existing English corpora, and 74% on similar German corpora.
- A framework for reinterpretation based on users' online corrections that achieves 92% accuracy for the English corpora and 90% for the German corpora with just one correction from the user.

In sum, we introduce a general framework for learning to interpret navigation instructions based on unlabeled observations of humans following such instructions.

The rest of the paper is organized as follows. Sections 2 and 3 introduce the general problem of instruction interpretation in spatial domains and describe the training corpora we use in this article. Then, Section 4 describes our novel framework for interpretation of instructions which does planning-based normalization of semantics and automatic annotation and compares different methods for situated interpretation, namely, machine learning, information extraction, and machine translation. The evaluation results of the different methods proposed in the previous section are presented in Section 5. Section 6 extends our interpretation framework by integrating online user corrections. Section 7 situates our work in the context of previous work while Section 8 discusses the generalisability and limitations of the proposed approach and concludes.

## 2. INSTRUCTION GIVING IN SPATIAL ENVIROMENTS

Instruction giving in spatial environments involves an instruction giver (IG) and an instruction follower (IF) who collaborate in order to achieve the goal of a task. The task is situated in an environment that has a spatial dimension such as the real world or a simulated virtual world. The IG gives instructions to the IF to help him complete the goal of the task (for example, to find an object). The IF follows the instructions by moving in the enviroment in which he is situated and manipulating objects inside it. There are two main lines of research related to automating instruction giving in spatial environments: instruction generation and instruction interpretation. In the instruction generation problem an automated system plays the role of the IG [MacMahon et al. 2006; Gargett et al. 2010; Cuayáhuitl and Dethlefs 2011]. In the instruction interpretation problem an automated system plays the role of the IF [Gorniak and Roy 2007; Matuszek et al. 2010; Vogel and Jurafsky 2010; Chen and Mooney 2011]. In this article, we focus on instruction interpretation.

The challenges in interpreting instructions situated in a spatial environment are many. We describe them along two dimensions: 1) variation and creativity in natural language free text, 2) multimodality of situated interpretation.

### 2.1. Interpreting free text natural language instructions in interaction

Even in rather simple spatial environments, people describe the same route and the same objects in extremely different ways. It is also well known that, when talking through a chat interface, people use language creatively, sometimes dropping or shortening some words, or not respecting the usual word order, especially when confronted with time pressure. This is especially true in instruction giving settings due to the fact that the other person is waiting for the instruction.

Below are some examples of instructions given by different people for the same route shown in Figure 1.

- 1) *out go*
- 2) *walk down the yellwo passage*
- 3) *straight*
- 4) *back to the corridor*
- 5) *ok now the door on the left*
- 6) *take the opening with yellow wall paper*

People describe routes using landmarks (4) or specific actions (2). They may describe the same object differently (5 vs 6). Instructions also differ in their scope (3 vs 1). Moreover instructions not only contain spelling errors but they do not follow usual word order. In sum, navigation and manipulation instructions contain considerable variation which makes interpreting them a challenging problem.

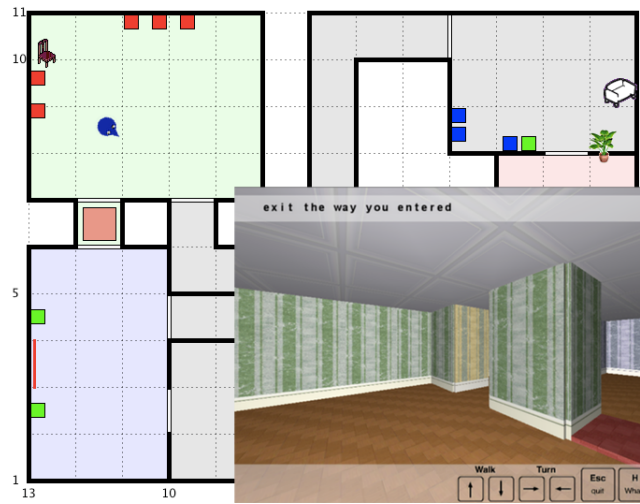


Fig. 1. A screenshot of a virtual world. The world consists of interconnecting hallways, rooms and objects.

## 2.2. Situatedness: integrating language and vision

The language used in the instructions is not only creative and lexically rich but it is also extremely *situated*. That is, most instructions cannot be interpreted without taking into account the spatial environment, that is the actions that are affordable and the objects that are visible in the situation in which the instructions were said.

The position of the IF has a direct effect on the properties included in the referring expressions as exemplified by the use of the property *closest* in the left picture in Figure 2; likewise the use of the word *right* in the picture on the right is appropriate due to the current orientation of the IF's camera.

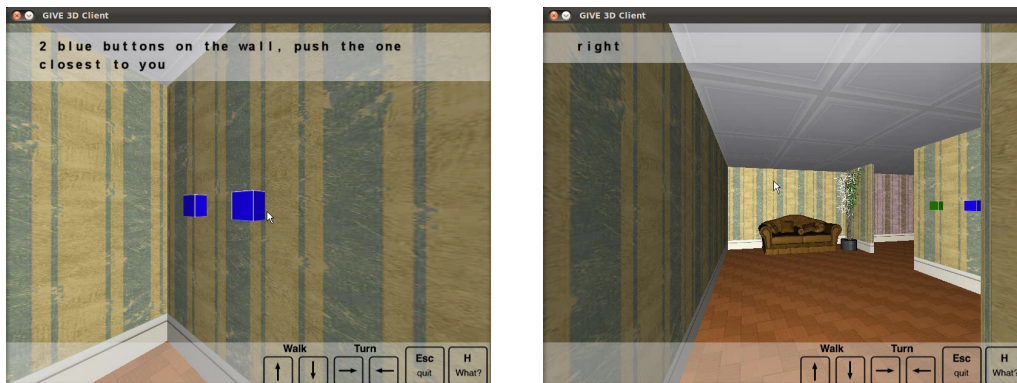


Fig. 2. Instructions depend on the position and orientation of the IF.

Moreover which objects are currently visible to the IF has an effect on the content of the instructions, and so does the distance to these objects from the IF's position. For instance, the instruction shown in the left picture in Figure 3 includes the complex referring expression *the red you see in the far room* that not only discards the green button as a potential referent but also specifies that the target is far, while the picture

on the right leaves the referring expression implicit due to the visual saliency of the target (the target is the only visible object).

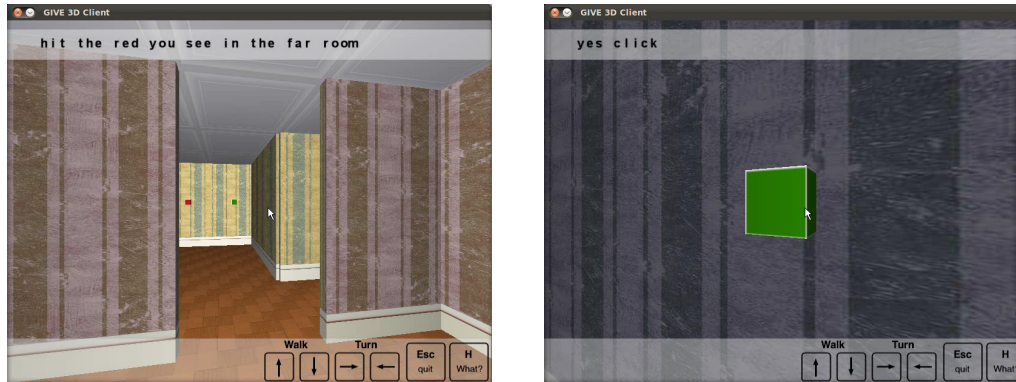


Fig. 3. References to an object depend on other visible objects and distance.

The interpretation approach proposed in this article combines different sources of information—position, area of visibility, language and spatial actions—to first identify the actions that are possible and salient in the current situation, and then choose the action that is intended by the IG.

### 3. HUMAN-HUMAN CORPORA SITUATED IN VIRTUAL WORLDS

In this section we describe the particular instruction giving task and the data that we use in this article. Our spatial environment consists of three virtual worlds simulating houses which contain rooms, corridors, furniture, buttons (i.e., switches) that open/close doors and activate/deactivate alarms etc. Screenshots and a part of a map of the virtual worlds can be seen in the figures of the previous section. In these worlds the IG and the IF collaborate in order to perform a treasure hunting task that involves cracking the combination of a safe by interacting with objects in different rooms. The task can be lost if the IF triggers an alarm by stepping on red tiles spread around the world or by interacting with the wrong objects. The IF can move around in the virtual world, but has no knowledge of the map of the world or the specific behavior of objects within that world (for example, which buttons to press to open doors or deactivate alarms). The IG has access to a complete map of the world with information on the effects of the actions on the different objects and types instructions to the IF in order to guide him to accomplish the task. In this setup, the IF and IG interact in real time.

With these three virtual worlds, the GIVE-2 Corpus [Gargett et al. 2010] was collected in order to help natural language generation system developers on the natural language generation shared task known as the GIVE Challenge [Koller et al. 2010]. In this article we use the corpus not for generating but for interpreting instructions. The GIVE-2 corpus consists of the collected interaction logs, which record the experimental session with enough detail to allow for a smooth replay. Specifically, interaction logs include all the instructions sent by the IG, all the actions performed by the IF, and the total time of the session. Furthermore, the IF's position and orientation was logged every 200 milliseconds, making it possible to extract information about their movements in response to instructions and other events. The IF was instructed to press a 'help key' if an instruction was not understood. In this case the IG is supposed to rephrase his/her instruction.

The GIVE-2 corpus was recorded using the GIVE-2 Corpus platform described in [Gargett et al. 2010]. It is important to mention, however, that most modern game engines include a ‘replay’ feature, which could be used to collect a corpus similar to the GIVE-2 corpus. This feature allows the player to record a sequence of game play and then watch it over again, perhaps from a different viewpoint, or in slow motion. Combined with a facility for logging instructions, this replay facility could easily be used to collect large amounts of training data similar to the GIVE-2 Corpus. In Section 8 we discuss the generalisability of our data collection and training environment in more detail and we consider how much of a development overhead it represents in comparison to other forms of data collection or annotation.

The GIVE-2 corpus consists of two corpora, one from German speakers and one from English speakers. The data was collected from 15 German speaking pairs and from 21 English speaking pairs. The participants were mostly students from one German and one US university. All 30 German speaking participants were native speakers of German—17 were female and 13 male. Of the 42 English speaking participants, 35 were native English speakers, while the others self-rated their English skills as near-native or very good—16 were female, 26 male.

The German corpus obtained in this way consists of 2763 instructions, spread over 45 rounds. On average, each round contained 61.4 instructions (Standard Deviation,  $SD = 24.0$ ) and took about 752 seconds ( $SD = 245.8$ ). For the English corpus, there were 63 rounds consisting of 3417 instructions in total. Rounds consisted on average of 54.2 ( $SD = 20.4$ ) instructions, and took about 553 seconds ( $SD = 178.4$ ). The German corpus contained 1531 distinct words while the English corpus contained 1293. Instructions contained 5 words on average.

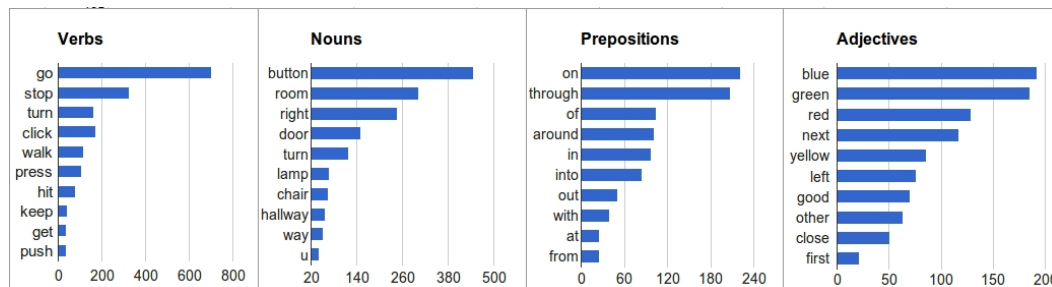


Fig. 4. The ten most frequent verbs, nouns, prepositions and adjectives found in the English corpus.

Figure 4 show the ten most frequent verbs, nouns, prepositions and adjectives found in the English GIVE-2 corpus. The lexical variability of the corpus is large and its distribution is biased towards spatial words grounded in the task and the spatial environment where the task is situated.

#### 4. LEARNING TO INTERPRET HUMAN-GENERATED INSTRUCTIONS

Our method for instruction interpretation consists of two phases: annotation and interpretation. *Annotation* is performed only once and consists of automatically associating each IG instruction with an IF reaction. *Interpretation* is performed every time the system receives an instruction and consists of predicting an appropriate reaction given reactions previously observed in the corpus.

Our method is based on the assumption that a reaction is a direct result of the instruction that occurred just prior to it. In other words, we assume that the IF re-



action makes explicit his interpretation of the instruction. Therefore, if two different instructions precede the same reaction, then they must be paraphrases of each other. We define paraphrases as instructions that cause the same reaction even though their semantics can differ because they use different tools to reach the same goal. For instance, Figure 5 shows a case in which references to different landmarks (*the picture* and *the red tile*<sup>1</sup>) are used to make the follower press a green button. The semantics of both instructions is different but their goal is the same.

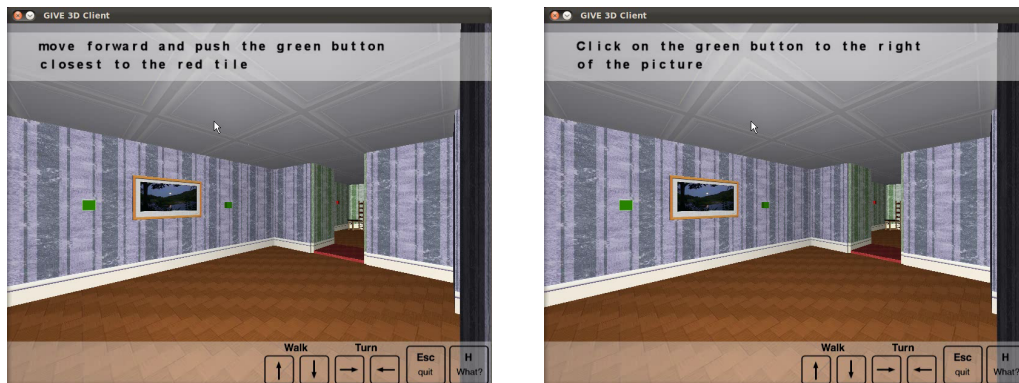


Fig. 5. Both figures show paraphrases of the same instruction. They use different vocabulary but communicate the same goal.

By learning from previously-seen instructions and reactions, our algorithm can predict reactions for previously-unseen instructions.

#### 4.1. Annotation phase

The key challenge in learning from massive amounts of easily-collected data is to automatically annotate an unannotated corpus. Our annotation method consists of two parts: first, *segmenting* a low-level interaction trace into instructions and corresponding reactions, and second, *discretizing* those reactions into canonical action sequences.

Segmentation enables our algorithm to learn from traces of IFs interacting directly with a virtual world. Since the IF can move freely in the virtual world, his actions are a stream of continuous behavior. Segmentation divides these traces into reactions that follow from each instruction of the IG. Consider the following example starting at the situation shown in Figure 1:

*IG(1): go through the yellow opening*  
*IF(2): [walks out of the room]*  
*IF(3): [turns left at the intersection]*  
*IF(4): [enters the room with the sofa]*  
*IG(5): push the green button by the door*  
*IF(6): [turns to make the green button visible]*  
*IF(7): [pushes the green button]*

It is not clear whether the IF is doing  $\langle 3, 4 \rangle$  because he is reacting to 1 or because he is being proactive. While one could manually annotate this data to remove extraneous

<sup>1</sup>In the picture, the red tile is located in the visible hallway; it is an alarm that prevents the IF from using that hallway.

actions, our goal is to develop automated solutions that enable learning from massive amounts of data.

We decided to approach this problem by experimenting with two segmentation algorithms: 1) a full segmentation that outputs all of the IF's *behavior* up until a new instruction is given, and 2) a partial segmentation that outputs only those actions taken in the context *visible* to the IF at the time the instruction was given.

We define *behavior segmentation* (BHV) as follows. A reaction  $r_k$  to an instruction  $i_k$  begins right after the instruction  $i_k$  is uttered and ends right before the next instruction  $i_{k+1}$  is uttered. In the example, instruction 1 corresponds to the reaction  $\langle 2, 3, 4 \rangle$ . We define *visibility segmentation* (VIS) as follows. A reaction  $r_k$  to an instruction  $i_k$  begins right after the instruction  $i_k$  is uttered and ends right before the next instruction  $i_{k+1}$  is uttered or right after the IF leaves the area visually accessible from where  $i_k$  was uttered. We define the area of visual accessibility with respect to a position as the area that is visible from all angles in that position (by turning  $360^\circ$ ). In the example, instruction 1's reaction would be limited to  $\langle 2 \rangle$  ( $\langle 3, 4 \rangle$  are discarded) because the intersection is not visible from where the instruction was uttered. The reaction to 5 would be  $\langle 6, 7 \rangle$ , because both of those actions are taken within the same visually accessible area.

The VIS segmentation is based on the empirical observation that, in situated interaction, the content of the instructions is constrained by *visually* accessible affordances [Gibson 1979; Stoia et al. 2006]. Visually accessible affordances are those actions executable from the current situation which refer to objects that are directly visible or can be made visible by turning around in the current position. Stoia [2006] shows that when generating instructions, IG prefer to navigate the IF to a position where the objects to be manipulated are visually accessible instead of giving complex high level instructions. Based on this observation, it is likely that an IG only expects an IF to perform actions within his field of view, thus motivating our definition of the VIS segmentation method.

The BHV and VIS methods define how to segment an interaction trace into instructions and their corresponding reactions. However, users frequently perform noisy behavior that is irrelevant to the goal of the task. For example, after hearing an instruction, an IF might step back in order to have a better view of the room before following the instruction. A reaction should not include such irrelevant actions. In addition, IFs may accomplish the same goal using different behaviors: two different IFs may interpret "go to the pink room" by following different paths to the same destination. We would like to be able to generalize both reactions into one canonical reaction.

To accomplish this, our approach *discretizes* reactions into higher-level action sequences reducing noise and variation. Our discretization algorithm uses an *automated planner* and a *planning representation* of the task. This planning representation includes: (1) the task goal, (2) the actions which can be taken in the virtual world, and (3) the current state of the virtual world. Using the planning representation, the planner calculates an optimal path between the starting and ending states of the reaction, eliminating all unnecessary actions. While we use the classical planner FF [Hoffmann 2003], our technique could also work with other *classical planners* [Nau et al. 2004] or other non-classical planning techniques such as *probabilistic planning* [Bonet and Geffner 2005]. It is also not dependent on a particular discretization of the world in terms of actions.

Now we are ready to define *canonical reaction*  $c_k$ . Let  $S_k$  be the state of the virtual world when instruction  $i_k$  was uttered,  $S_{k+1}$  be the state of the world where the reaction ends (as defined by BHV or VIS segmentation), and  $D$  be the planning domain representation of the virtual world. The *canonical reaction* to  $i_k$  is defined as the sequence of actions returned by the planner with  $S_k$  as initial state,  $S_{k+1}$  as goal state



and  $D$  as planning domain. Note that  $S_{k+1}$  depends on whether BHV or VIS segmentation is used.

The annotation of the corpus then consists of automatically associating each instruction to its (discretized) reaction using an automated planner. We annotated in this way the 3417 instructions of the English corpus and the 2763 instructions of the German corpus. 22% of the English instructions and 17% of the German instructions contained an empty reaction. A reaction is empty if the IF did not execute an action in response to it and waited for another instruction. This is the case in two possible situations, either when the instruction was not understood and the IF asked for a rephrased instruction or when the instruction explicitly indicated that the IF should stop moving. For training and evaluating our algorithms we use those instructions that do not have an empty reaction: 2665 English instructions and 2293 German instructions. A fragment of the annotated corpus is shown in the Appendix.

Once the corpus has been annotated, it can be used by the interpretation phase to develop automatically-generated reactions to human instructions.

#### 4.2. Interpretation phase

The annotation phase results in a collection of  $(i_k, c_k)$  pairs. The interpretation phase uses these pairs to interpret new instructions in three steps. First, we *filter* the set of pairs into those whose reactions can be directly executed from the current state. Second, we *group* the filtered pairs according to their reactions into what we call *paraphrase groups*. Third, we *select* the group with instructions most similar to the new instruction, and then we output that group's reaction. Figure 6 shows the output of the first two steps: three groups of pairs whose reactions can all be executed from the IF's current position.

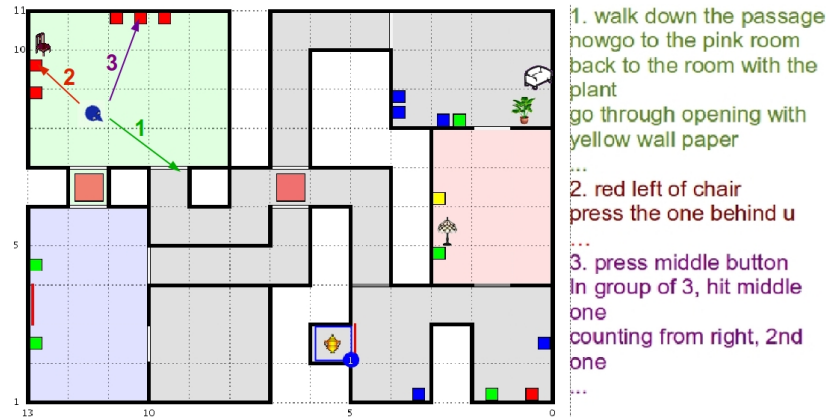


Fig. 6. Sample instruction groups for depicted situation. Arrows show the reaction associated with each group. Each group corresponds to a visually perceived affordance.

The input of the filtering step includes the corpora described in Section 3 annotated as explained in Section 4.1 and a planning representation [Nau et al. 2004] of the state of the virtual world at the moment in which the instruction to be interpreted is received, designed  $S_k$ . State  $S_k$  includes the position of the IF and the state of all the objects in the virtual world, such as which alarms are active and which doors are open.

The filtering step considers all possible  $(i_k, c_k)$  pairs in the corpus and removes those whose reaction  $c_k$  is not applicable in  $S_k$ . To determine this, our algorithm simulates the execution of plan  $c_k$  using  $S_k$  as the initial state. If any precondition of  $c_k$  cannot

be met at some point during the simulated execution, then we say that  $c_k$  can not be executed in  $S_k$  and therefore we filter this  $(i_k, c_k)$  pair from consideration. Those pairs that remain after filtering are the output of the filtering process.

Filtering is computationally expensive. To reduce the amount of computation required, we applied a situated optimization technique that pre-filters the corpora to those instructions uttered from the same *visually accessible region* as the instruction being interpreted. We precompute visually accessible regions by partitioning the virtual world into adjacent areas such that all points from each area are visible from all other points in that area. After applying this optimization, the filtering step can be performed in real time. A sample output of the filtering step is illustrated by all the instructions in Figure 6.

The next step of our algorithm is *grouping*. In this step, all filtered instructions are organized into groups of instructions with the same annotated reaction. Since all instructions in a group are associated with the same reaction, they must, by our definition of paraphrase, be paraphrases of each other.

The third step of our algorithm is to *select* the group whose reaction is the intended by the IG's instruction. Given the groups that have been formed in the previous step, our algorithm selects the one whose paraphrases are most similar to the new incoming instruction. We define similarity in terms of language and visibility based features that we describe below. We then output the selected group's reaction as the system's reaction to this instruction.

We treat this third step as a classification problem. In this article, we compare six different classification algorithms grouped into three different methods, based on word similarity, machine translation, and machine learning.

**4.2.1. Word-similarity-based group selection.** The first set of methods use nearest-neighbor classification with three different similarity metrics. Jaccard and Overlap coefficients measure the degree of overlap between two sets, differing only in the normalization of the final value [Nikraves et al. 2005]. Levenshtein Distance is a string metric for measuring the amount of differences between two sequences of words [Levenshtein 1966]. We implement all these algorithms at the unigram level, considering an instruction as a set of words. The Jaccard coefficient measures similarity between sets, and is defined as the size of the intersection divided by the size of the union of the sets. The Overlap coefficient is defined as the size of the intersection divided by the maximum of the sizes of sets. We calculate Jaccard and Overlap for each (instruction, paraphrase) pair and we average over all paraphrases in each paraphrase group to get the score for that paraphrase group. We select the paraphrase group with the highest average score. We calculate Levenshtein Distance by associating a cost of 1 to each deletion and insertion and a cost of 2 to each substitution. We select the set of paraphrases whose average Levenshtein distance to the interpreted instruction is lowest.

Let us compare these three methods by means of an example. Suppose the instruction *red* is to be interpreted and there are two groups of paraphrases whose reaction can be executed in the current situation. Each paraphrase group contains only one instruction, namely *blue* and *hit the red one*<sup>2</sup>. The Levenshtein Distance between *red* and *blue* is 2 and between *red* and *hit the red one* is 3 so *blue* is selected as the paraphrase. The Jaccard index between *red* and *blue* is 0 and between *red* and *hit the red one* is  $\frac{1}{4}$  so *hit the red one* is selected. The Overlap index is the same as the Jaccard index for this example. This example illustrates the fact that Levenshtein Distance penalizes

<sup>2</sup>In our data there is an average of 20 instructions per paraphrase group but we restrict the example to one for illustration purposes.

differences in length between the instructions and this is not suitable for our data in which speakers frequently drop words and use ellipses. Therefore we expect Jaccard and Overlap algorithms to have a better performance than Levenshtein Distance.

**4.2.2. Machine translation-based group selection.** The next classification method employs a strategy in which we considered each group as a set of possible machine translations of our instruction, using the BLEU metric [Papineni et al. 2002] to select the group that reports the highest score. The BLEU metric was proposed in the machine translation area for evaluating the quality of text which has been machine-translated from one natural language to another. BLEU is a modified form of n-gram precision that compares a candidate translation against multiple reference translations which are supposed to be correct. We consider the instruction that we are interpreting as the candidate translation and the paraphrases sets as the reference translations. BLEU is calculated as the sum, for each distinct word in the candidate translation, of its *ratio of appearance* in the candidate translation. This ratio is calculated by dividing the *word clipped occurrences* by the length of the candidate translation. The word clipped occurrences are calculated as the minimum between the number of occurrences of the word in the candidate translation and the maximum number of occurrences of the word in a reference translation.

For example, suppose that we are interpreting *push the blue button* and the corresponding paraphrase group is *{push it, the blue one}*. Then the BLEU metric is  $\frac{1}{4} + \frac{1}{4} + \frac{1}{4} + \frac{0}{4} = \frac{3}{4}$ . BLEU is equal to one when all the words in the candidate appear in some reference and its equal to zero when none of them appear. Since BLEU, Jaccard, and Overlap calculate variations of n-gram precision we expect them to exhibit similar performance. We empirically test this hypothesis in the next section.

**4.2.3. Machine learning-based group selection.** Finally, we trained an SVM classifier [Cortes and Vapnik 1995] and a decision tree classifier [Murthy 1998] using the unigrams of each paraphrase and the position and visibility area of the IF as features, and setting their group as the output class. The particular implementations we used were the libSVM wrapper described in [Chang and Lin 2011] with a radial kernel and default parameters and the decision trees were implemented using the classifier J48 as implemented in the machine learning WEKA package [Hall et al. 2009].

Since the machine learning-based classifiers make use of additional information beyond the words in each paraphrase group, we expect them to perform better than the previous two methods, which use only word similarity.

In the next section we compare the performance of these two machine learning methods to the other algorithms and we describe the model learned from our data.

## 5. ALGORITHM COMPARISON AND EVALUATION

For the evaluation phase, we annotated both the English and German corpora entirely, and then we did a five fold partition and we calculated the average on the five partition results. For each pair  $(i_k, c_k)$  in the testing set, we used our algorithm to predict the reaction to the selected instruction, and then compared this result against the annotated reaction. Since the reactions are canonical, the prediction is considered correct if the predicted reaction is identical to the annotated one. Table I shows the results.

Since each virtual world contains thousands of possible actions, the possibility of predicting the intended action without information is low. However, if you consider the position of the player and the afforded actions from that position, there are on average 4.7 actions per position in the virtual worlds, giving a chance of selecting the right action without considering the content of the instruction of 21%. Hence, the *affordability baseline* accuracy is 21%.

Table I. Accuracy comparison between BHV and VIS segmentation on the English and German corpora.

Algorithm	English corpus		German corpus	
	BHV	VIS	BHV	VIS
Levenshtein	34%	52%	43%	54%
Jaccard	45%	70%	53%	69%
Overlap	44%	70%	51%	69%
BLEU	44%	67%	54%	66%
DT	48%	73%	57%	70%
SVM	50%	<b>77%</b>	59%	<b>74%</b>

Comparing the BHV and VIS segmentation strategies, VIS obtains better results than BHV for all algorithms (the differences are statistically significant,  $p < 0.01$ ). After performing a manual error analysis of the BHV classification results we concluded that the problem with this segmentation strategy is that it splits groups of instructions that have the same meaning. Navigation instructions are frequently underspecified in that they do not explicitly specify the exact final location that is intended. Take for instance the instructions (1) *exit the room* and (2) *go out* as if uttered from the IF position in Figure 1, and suppose that after exiting the room, the IF of (1) takes the left corridor but the IF of (2) takes the right corridor. As a result (1) and (2) would be correctly considered as paraphrases by the VIS algorithm but incorrectly considered as instructions with different meaning by the BHV algorithm. A correct segmentation strategy is crucial for the performance of our interpretation classifiers since it determines the training data.

Given the size of our dataset, the differences in algorithm performance reported in Table I are statistically significant when greater than 2.6 ( $p < 0.05$ ). The best performing algorithms are the machine learning methods Support Vector Machines (SVM) and Decision Trees (DT). Their performance is always better than the rest in a statistically significant way ( $p < 0.05$ ). As we anticipated in the previous section we believe this is due to the fact that these algorithms are able to integrate information from different sources (visibility, reaction and language information) into one classification model. They are able to learn, for instance, that spatial words such as *left*, *right*, *straight* and so on, have different meanings according to the orientation of the IF. Furthermore, they capture the fact that synonyms such as *push* and *press* indicate that the user has to manipulate a button. It is also interesting to see how a DT is trained on hundreds of features (all unigrams in the paraphrase groups plus visibility features) but includes only a couple dozen features in each learned model.

Summing up, the machine learning methods are able to learn the words that are relevant for the corpus domain and as well as the interplay between the visibility area and its effect on the language used.

We also evaluated human performance on the English corpus given the same type of information that is accessible to the system, namely, current position and visibility area of the player plus input instruction, but not conversational context. We asked two human annotators to independently annotate the instructions from the English corpus and we obtained 81% as their average accuracy (with respect to the IF's reaction) and a Cohen Kappa of 0.75, which is considered very good [Carletta 1996]. Annotators' disagreements have a high correlation (.88) to the instructions that are not correctly interpreted by our best algorithm (VIS with SVM). We performed an error analysis and we found that almost all problematic instructions either required the conversational context to be interpreted, or made references to previous actions. In other words, they required access to the *interactional common ground* of the IG and IF. Examples of such instructions are *go back* and *keep going*, which require knowledge about the action executed right before the current one; *exit the way you entered* and *press the green*

*button again*, which require knowledge about actions executed previously during the interaction; *yes* and *click it* which requires knowledge about the action or object that was in the interactional focus at that point during the interaction.

Due to the fact that our best performing algorithm is only 4 points under human performance when no conversational context is available, we believe that no major improvement would be gained from using natural language processing methods such as word order similarity measures instead of using just uni-gram information as we do. In some initial experiments we used n-grams of up to size four and the results consistently decreased. We believe this is due to the loose word order that the speakers use in the corpora due to its chat-based nature and the time-constrained nature of the interaction. Moreover, the utterances in the corpora are quite short with only 5 words in average therefore we believe that methods such as latent semantic analysis would not have an impact on overall performance. When interpreting longer and less noisy instructions such methods may prove useful, for example on data such as that used by [Branavan et al. 2009] who developed a system which learns to follow Windows Help guides. The GIVE is free form conversational English/German, whereas the Windows instructions are written by a professional.

We compared human annotations to our automatic VIS segmentation annotations. Only 5% of the instructions were considered by a human to have a longer reaction than that annotated by the automatic VIS segmentation strategy. This means that 72% of the instructions interpreted by the VIS with SVM algorithm reached the final goal of the instruction while 5% of the instructions started to react in the correct direction but did not reach the final goal.

## 6. ONLINE USER CORRECTIONS

When humans give instructions in a situated setting, they monitor the interpretation of their instructions by observing the actions of the interpreter; if the interpreter misunderstands an instruction the instructor can correct him. Empirical studies [Purver 2004] show that most corrections come in the form of rephrases (over 80% in this study) of the original instruction. If one instruction did not work, the IG tries again by rephrasing his instruction, until the IF is able to perform the correct reaction.

To the best of our knowledge, previous work in instruction interpretation does not process online corrections. In this section, we describe how we have extended our instruction interpretation algorithm to incorporate online IG corrections in order to increase our system's accuracy, that is the automated IF's accuracy.

Below we first explain our method for reinterpretation of instructions by incorporating corrections and then we describe our evaluation of this method.

### 6.1. Leveraging corrections

We have extended our framework to include a “cancel” button that indicates to the system that the reaction it has chosen to perform is incorrect and “rewinds” the position of the IF to where the misinterpreted instruction was received. The IG (a human user) can then issue a second instruction, which is assumed to be a paraphrase of her original instruction. The system will then reinterpret the original instruction by combining its interpretation with the interpretation of the new instruction received. This reinterpretation is done in two steps. First, the paraphrase group previously selected is discarded. Then the scores obtained for the misinterpreted instruction and the new one are averaged. This algorithm is inspired by the *belief tracking* techniques used for speech recognition [Williams and Young 2007]. In this area, previous hypotheses of the speech recognizer are preserved and their ranking is combined with the new hypothesis.

If the new instruction results in the correct response being generated (indicated by the user neglecting to push the cancel button before issuing the next instruction), then the system removes this new instruction and reaction pair from the test set and adds it to the training set, so that it can react correctly next time.

We believe that this form of human feedback and online learning, which mimics the interactive feedback available in human-human interaction, enables the system to react more quickly and correctly to user instructions. In the following subsection, we present an empirical evaluation measuring how well system performance improves with user corrections.

## 6.2. The impact of corrections

In order to be able to compare the results with and without user corrections, we have devised a new method for measuring the impact of user corrections on system performance, inspired by evaluation methodologies for online learning from time series data [Gama et al. 2009], that takes advantage of the same corpora used for evaluation in the previous section.

We begin by splitting the corpus into two equal halves: a training set and a test set. The training set is input into the annotation phase of our algorithm to construct a set of  $(i_k, c_k)$  pairs. We do the same for the test set, constructing a set of  $(i_k', c_k')$  pairs for the test set. Starting from the training set, we then iteratively run the interpretation phase using the test set as input, until a termination condition is reached. For each interpretation phase, the system is given one new instruction  $i_k$  from the test set and the system's reaction  $interp(i_k)$  is compared to the correct reaction  $c_k$ . If the system produced the correct reaction, we mark this example as successful, add this example to the training set, and continue with the next instruction.

If the system produces an incorrect reaction, then we simulate the user hitting the “cancel” button and providing an alternate instruction with the same meaning (i.e., the same reaction) as follows. Since the user knows what is the reaction that he intended, he can produce an alternate instruction with the same reaction of the misinterpreted one. To simulate the production of an instruction with the same meaning, during evaluation, we select from the test set an (instruction, reaction) pair  $(i_k', c_k')$  such that  $c_k' = c_k$ . If  $interp(i_k') = c_k$ , then the system produced the correct reaction given the new instruction. In this case, we mark the interaction containing both instructions (the initial one and the alternate instruction) as successful. Both instructions are removed from the test set and added to the training set.

This evaluation methodology is inspired by evaluation methodologies for online learning from time series data [Gama et al. 2009]. The test set is fixed, but the system under test changes during this measurement process, because we are simulating an environment in which user feedback during system usage enables the system to improve its performance. We measure the overall accuracy of this evolving system as a proxy for how users might experience its performance in a deployed setting.

The algorithm thus explained was given two tries to select a correct reaction: the original instruction, plus one alternate. We can also increase the system's chances of selecting a correct reaction by giving it up to four tries. Figure 7 shows the accuracy for each of the classification methods as the number of tries increases from one (no corrections) to four on the English corpus. The performance on the German corpus evolves in a similar way.

As expected, accuracy increases as the number of tries increases. The best algorithm (SVM) reaches 92% with just one correction from the IG in the English corpus and 90% on the German corpus.

We define the accuracy of the system as the fraction of successful interactions divided by the total number of interactions. We define an interaction as a set of utterances

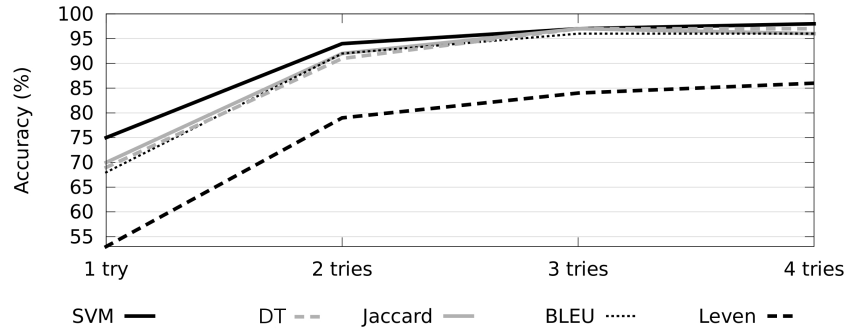


Fig. 7. Accuracy values with corrections over the English corpus.

with the same reaction. An interaction may contain between one and four utterances, depending on how many tries each algorithm needs in order to successfully interpret the user intention. Since our test set contains a fixed number of utterances, the number of interactions decreases as the number of tries increases. Figure 8 shows the number of interactions for each algorithm plotted against the number of tries. With the same number of tries, better algorithms have a slightly higher number of interactions since they need fewer alternate instructions in order to get the correct reaction. This metric reflects a user's experience when using our system. It measures the percentage of cases in which the system was able to actually understand the user intention.

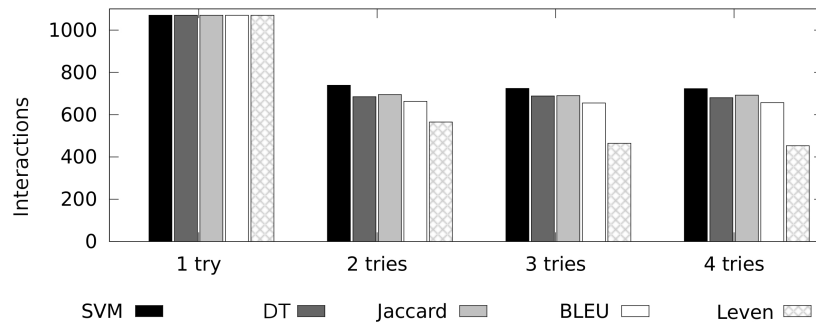


Fig. 8. Number of interactions for each algorithm with corrections over the English corpus.

It should be noticed that, in this experiment, we only simulate an interaction in which user corrections occur and we assume that randomly sampled paraphrases are suitable corrections. In an actual interaction, corrective utterances are likely to be related to the original instruction. For example, they may avoid the use of words that may have caused the misunderstanding, or they may be elliptical to the original utterance. Our evaluation methodology has the limitation of not being able to capture this relation. We intend to explore actual interactive evaluations in future work.

## 7. RELATED WORK

The problem of interpreting instructions in natural language has been studied since the early days of artificial intelligence with systems such as SHRDLU [Winograd



1972]. SHRDLU was one of the earliest systems to use a formal natural language grammar to parse instructions and map them into automatically executable actions. Since then many other such symbolic systems have been developed [Koller et al. 2004; MacMahon et al. 2006; Dzikovska et al. 2008; Benotti 2009].

The main advantage of such systems is that their syntax-semantics interfaces are formally defined and deep meaning representations of the instructions can be built. Therefore, symbolic systems are readily *portable* to new situations inside the same domain. For example, in the GIVE domain, if a symbolic system can interpret the instruction “hit the one to the left of the chair”, it will probably be able to interpret “hit the one to the right of the chair”. This advantage is particularly useful if the system is to be used in different situations inside the same domain whose characteristics cannot be predicted (or are too many to be modeled) at design time.

The main disadvantage of symbolic systems is that they are brittle due to the lexical and grammatical variance found in conversational language. As a result, some authors report very *low coverage* when interpreting natural language instructions using formal grammars. For example, Lau et al. [2009] found that only 3% of crowdsourced instructions describing how to accomplish certain tasks on the web (such as creating a web-mail account or buying online) were successfully interpreted using formal grammars. Other authors report a much better coverage using symbolic methods. For example, MacMahon et al. [2006] found that 61% of the instructions were correctly followed by a grammar-based system. The difference in the results can be explained by two factors. First, the corpus used by [Lau et al. 2009] is noisy and conversational while the corpus used by [MacMahon et al. 2006] contains full grammatical sentences. Second, the grammar used by [MacMahon et al. 2006] includes many semantic language patterns such as termination conditions like “Turn so that you see a chair in front.” Summing up, current symbolic approaches to instruction interpretation require intensive grammar authoring to be fit for a new domain.

In order to address the coverage problem, statistical semantic parsers have been used in the area of natural language interpretation. Semantic parsers map utterances directly from surface form into their logical form without the need of a syntax-semantics interface. They are typically trained from examples of utterances annotated with semantic logical forms [Rieser and Lemon 2010; Kwiatkowski et al. 2010; Jones et al. 2012]. This approach improves the coverage while retaining the portability advantage of symbolic approaches. The main disadvantage of this approach is that this type of annotation is *expensive* since it must be done by people that are not only domain experts but are also proficient in the semantic formalism used.

Recent research has investigated the problem of learning to interpret natural language utterances without the need of logical form annotations. In general the methodology used is to learn the correspondence of two natural inputs such as question-answer pairs, instructions-world reaction, etc. One such approach has been proposed for interpreting questions [Liang et al. 2013]. The goal of this work is to learn a semantic parser from question-answer pairs, where the logical form is modeled as a latent variable. Other researchers have also pushed in this direction in various ways: learning a semantic parser based on bootstrapping and estimating the confidence of its own predictions by interacting with the world [Goldwasser et al. 2011], learning a semantic parser from user interactions with a dialog system [Artzi and Zettlemoyer 2011], and learning to execute natural language instructions from just a reward signal using reinforcement learning [Branavan et al. 2009; Vogel and Jurafsky 2010; Goldwasser et al. 2011]. In general, supervision from the world is indirectly related to the learning task, but it is often much more plentiful and natural to obtain. Hence, all these methods retain the portability advantage of other approaches and require *cheaper* forms of supervision than supervised learning of semantic parsers.



Fig. 9. A screenshot of a virtual world. Many objects of the same type are present in the world.

The performance of such systems vary from one domain to the other (for example, instruction giving in spatial domains versus question answering domains) so our results are not comparable to them. Our work is the most similar to that of [Chen and Mooney 2011]. Their system also learns to follow navigation instructions from pairs of instructions and map traces (which includes visibility information) with no prior linguistic knowledge. In situated instruction giving the integration of linguistic information with visual and physical contextual information is crucial due to the elliptical characteristic of language as we discussed in Section 2.

However, there are subtle but important differences between the work of [Chen and Mooney 2011] and ours. To begin with, their method retains the portability advantage. This makes it more suitable to be applied in dynamic domains as mentioned in the beginning of this section. In contrast, our approach requires less manual supervision. Their segmentations must be performed manually and the canonical plans annotated by hand [Chen 2012, p. 84]. Our system performs segmentation and annotation automatically, providing a significant benefit despite being unable to predict unseen reactions. Moreover, they work on simpler worlds. In particular, the references to objects are not ambiguous since all the objects in the world are different. We intend to be able to interpret instructions on realistic house-like worlds that contain many objects of the same type and require the use of multiple types of properties to be able to distinguish them (see for instance a screenshot of a complex world in Figure 9). Finally, their instructions were not collected in an interactive setup; as a result, their instructions are more structured and less elliptical. In the corpus they use, people describe complete tasks in a discourse that the other person follows without interaction. As a consequence, language and action are not aligned and they have to align and segment them manually before being able to learn from them.

An interesting similarity between Chen and Mooney [2011] and our proposal is that both approaches can not only be used for interpretation but also for generation of natural language [Chen et al. 2010; Benotti and Denis 2011].

The framework we present in this article learns from conversational data in which instructions are collected while a person gives instructions and another person follows them in real time. A limitation of our approach is that corpora in exactly the same spatial domains are required for the approach to be effective. However, our approach is

able to use data that is widely available instead of collecting our own, such as corpora of gamers playing online where language and action co-occurs [Leuski et al. 2012; Small et al. 2011]. If corpora in the required virtual worlds are not available for a particular application, its collection is simple and inexpensive since no trained annotators are required, just a person playing the role that the system will need to play. We further discuss this issue in the next section.

This article reports on an expanded version of work originally published as a short paper [Benotti et al. 2012]. Relative to the prior publication, we provide significantly more details about the algorithms, and add a new German corpus to the evaluation, showing that the empirical results are similar to those previously obtained with an English-language corpus. Moreover, we include a five-fold cross-validation which makes our results more robust. We also added visibility features to our machine learning algorithms which considerably improve their performance. We experiment with a new algorithm (namely, decision trees) which gives us insights on what the algorithm is learning. Finally we performed a human evaluation which gives us an upper bound to what our algorithm is expected to learn.

## 8. DISCUSSION AND CONCLUSIONS

We have presented an approach to instruction interpretation which learns from unannotated logs of humans following instructions in a virtual world. Our empirical analysis shows that our best algorithm achieves 77% accuracy on this task, with no manual annotation required. When user corrections are added, accuracy goes up to 92% for just one correction. We consider our results promising since similar state of the art approaches to instruction interpretation [Chen and Mooney 2011] report only 55% accuracy on manually segmented data.

The primary benefit of our approach is that it requires no manual corpus annotation, thus enabling use of larger corpora than traditional statistical approaches. Having said this, a question that remains is how widely applicable our approach is and what its requirements are. We require a capturing interface to log IF/IG behavior, and we need a large number of people to participate. Fortunately, these interfaces are readily available. Most multiplayer online games such as World of Warcraft and Counter Strike, among many others, already offer logging capabilities that provides the required capturing interface (in the gaming community, this is known as replay). Previous work has shown that large corpora can be crowdsourced using this technique not only for games and online worlds [Small et al. 2011; Orkins and Roy 2007] but also for real world simulations useful for generating robot behavior [Chernova et al. 2011].

In addition, we also require a visibility model and an action planner. Visibility models already exist in game engines to render only the portion of the world visible from the players current position. For the planner, we use Fast-Forward [Hoffmann 2003], which typically returns a plan in the GIVE domain in under 15ms. However, the in-game planning capabilities of non-player characters can also be used for this purpose, with the extra advantage of being optimized for performance in that virtual world.

To sum up the limitations: our approach requires the collection of detailed corpora of human behavior, but the tools to collect these corpora already exist in online gaming systems. Although our approach is limited to the same virtual world in which the corpora were collected, the availability of collection tools means our approach can scale quickly to new worlds. We intend our contribution to provide a valuable alternative in the spectrum between manual annotation and symbolic rule-based approaches.

An additional line for future work is to consider common ground [Clark 1996] features such as the actions performed by the IF before receiving the instruction and the objects that were under linguistic focus in the previous instructions.

Although we have presented our approach in the context of 3D virtual worlds, similar types of instruction following are seen in other domains such as the web [Lau et al. 2010], massively multiplayer online games, and human-robot interaction. In future work, we plan to apply our instruction interpretation algorithm to these domains and compare its performance.

Finally, a note about performance. While our approach is computationally intensive, it scales well thanks to filtering of the actions that are possible at the players current location. The average interpretation time is 158ms (stddev 173ms) on an Intel Core i3 in the GIVE worlds used. Due to the locality filtering, we do not expect our approach to degrade on bigger worlds and datasets.

**APPENDIX: Automatically annotated fragment of the GIVE corpus**

Utterance:	make a left and exit the room
Visible objects:	⟨button2,chair1⟩
Reaction:	⟨move(b2-room-1-9,room-1-9), move(room-1-9,room-1-8), move(room-1-8,room-1-7), move(room-1-7,room-1-6), move(room-1-6,room-1-5), move(room-1-5,d3-room-1-5)⟩
Utterance:	enter
Visible objects:	⟨door3>window4,button3⟩
Reaction:	⟨move(d3-room-1-5,d3-room-2), move(d3-room-2,room-2)⟩
Utterance:	go into the room on the right
Visible objects:	⟨alarm2,door1,door6,button3⟩
Reaction:	⟨move(room-2,d6-room-2), move(d6-room-2,a2-d6-room-3), move(a2-d6-room-3,room-3)⟩
Utterance:	push the green button to the left of the red button
Visible objects:	⟨door11,button5,button6⟩
Reaction:	⟨move(room-3,b6-room-3), manipulate-stateless(b6), move(b6-room-3,room-3)⟩
Utterance:	go into the room on your right
Visible objects:	⟨door7,door11⟩
Reaction:	⟨move(room-3,d11-room-3), move(d11-room-3,d11-room-7), move(d11-room-7,room-7)⟩
Utterance:	the red button left of lamp, click
Visible objects:	⟨button7,button11,lamp1,sofa1⟩
Reaction:	⟨move(room-7,b11-room-7), manipulate(b11), move(b11-room-7,room-7)⟩
Utterance:	on your right, push the yellow button
Visible objects:	⟨button7,button11,lamp1,sofa1⟩
Reaction:	⟨move(room-7,b10-room-7), manipulate-stateless(b10), move(b10-room-7,room-7)⟩
Utterance:	hit the blue button in the middle of the yellow and blue button
Visible objects:	⟨button7,button11,button9,button8,lamp1,door13⟩
Reaction:	⟨move(room-7,b8-b9-room-7), manipulate-stateless(b9), move(b8-b9-room-7,room-7)⟩
Utterance:	go into the room on the right
Visible objects:	⟨button7,button11,button9,button8,sofa1,door10,door13⟩
Reaction:	⟨move(room-7,d10-room-7), move(d10-room-7,d10-room-6), move(d10-room-6,room-6)⟩
Utterance:	the red button next to the blue button on your right
Visible objects:	⟨button13,button14,picture2,door9⟩
Reaction:	⟨move(room-6,b13-b14-room-6), manipulate(b14), move(b13-b14-room-6,room-6)⟩
Utterance:	and walk through the hall
Visible objects:	⟨button13,button14,picture2,door9⟩
Reaction:	⟨move(room-6,d9-room-6), move(d9-room-6,d9-room-5), move(d9-room-5,room-5)⟩

**ACKNOWLEDGMENTS**

We are particularly grateful to Julián Cerruti, ex-manager of the Silvergate Team in IBM Argentina for extensive discussions on the material presented in this article and for his continuous support. We also thank the Silvergate Team for helping us.

This work was partially supported by an IBM SUR Award granted to the Facultad de Matemática Astronomía y Física of the Universidad Nacional de Córdoba, and grants ANPCyT-PICT-2012-712, ANPCyT-PICT-2008-306, ANPCyT-PICT-2010-688 and the FP7-PEOPLE-2011-IRSES Project Mobility between Europe and Argentina applying Logics to Systems (MEALS).

## REFERENCES

- ARTZI, Y. AND ZETTLEMOYER, L. 2011. Bootstrapping semantic parsers from conversations. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Association for Computational Linguistics, Stroudsburg, PA, USA, 421–432.
- BENOTTI, L. 2009. Frolog: an accommodating text-adventure game. In *Proceedings of the 12th Conference of the European Chapter of the Association for Computational Linguistics: Demonstrations Session (EACL)*, A. Lascarides, C. Gardent, and J. Nivre, Eds. The Association for Computer Linguistics, 1–4.
- BENOTTI, L. AND DENIS, A. 2011. Prototyping virtual instructors from human-human corpora. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies: Systems Demonstrations (ACL)*. The Association for Computer Linguistics, 62–67.
- BENOTTI, L., VILLALBA, M., LAU, T., AND CERRUTI, J. 2012. Corpus-based interpretation of instructions in virtual environments. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*. Association for Computational Linguistics, Jeju Island, Korea, 181–186.
- BONET, B. AND GEFFNER, H. 2005. mGPT: a probabilistic planner based on heuristic search. *Journal of Artificial Intelligence Research* 24, 933–944.
- BRANAVAN, S., CHEN, H., ZETTLEMOYER, L., AND BARZILAY, R. 2009. Reinforcement learning for mapping instructions to actions. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the Association for Computational Linguistics and the 4th International Joint Conference on Natural Language Processing (ACL-IJNLP)*. Association for Computational Linguistics, Suntec, Singapore, 82–90.
- CARLETTA, J. 1996. Assessing agreement on classification tasks: the kappa statistic. *Computational Linguistics* 22, 2, 249–254.
- CHANG, C.-C. AND LIN, C.-J. 2011. LIBSVM: A library for support vector machines. *ACM Transactions on Intelligent Systems and Technology* 2, 27:1–27:27. Software available at <http://www.csie.ntu.edu.tw/~cjlin/libsvm>.
- CHEN, D. L. 2012. Learning Language from Ambiguous Perceptual Context. Ph.D. thesis, AT Austin.
- CHEN, D. L., KIM, J., AND MOONEY, R. J. 2010. Training a multilingual sportscaster: using perceptual context to learn language. *Journal Artificial Intelligence Research* 37, 1, 397–436.
- CHEN, D. L. AND MOONEY, R. J. 2011. Learning to interpret natural language navigation instructions from observations. In *Proceedings of the Twenty-Fifth Conference on Artificial Intelligence (AAAI)*, W. Burgard and D. Roth, Eds. AAAI Press, San Francisco, California, USA, 859–865.
- CHERNOVA, S., DEPALMA, N., AND BREAZEAL, C. 2011. Crowdsourcing real world human-robot dialog and teamwork through online multiplayer games. *AI Magazine* 32, 4, 100–111.
- CLARK, H. 1996. *Using Language*. Cambridge University Press.
- CORTES, C. AND VAPNIK, V. 1995. Support-vector networks. *Machine Learning* 20, 273–297.
- CUAYÁHUITL, H. AND DETHLEFS, N. 2011. Spatially-aware dialogue control using hierarchical reinforcement learning. *ACM Transactions on Speech and Language Processing* 7, 5:1–5:26.
- DZIKOVSKA, M. O., ALLEN, J. F., AND SWIFT, M. D. 2008. Linking semantic and knowledge representations in a multi-domain dialogue system. *Journal of Logic and Computation* 18, 405–430.
- GAMA, J. A., SEBASTIÃO, R., AND RODRIGUES, P. P. 2009. Issues in evaluation of stream learning algorithms. In *Proceedings of the 15th ACM International Conference on Knowledge Discovery and Data Mining (KDD)*. ACM, New York, NY, USA, 329–338.
- GARGETT, A., GAROUFI, K., KOLLER, A., AND STRIEGNITZ, K. 2010. The GIVE-2 corpus of giving instructions in virtual environments. In *Proceedings of the Seventh Conference on International Language Resources and Evaluation (LREC-2010)*, N. Calzolari, K. Choukri, B. Maegaard, J. Mariani, J. Odiijk, S. Piperidis, M. Rosner, and D. Tapias, Eds. European Language Resources Association, Malta.
- GIBSON, J. J. 1979. *The Ecological Approach to Visual Perception*. Houghton Mifflin.
- GOLDWASSER, D., REICHART, R., CLARKE, J., AND ROTH, D. 2011. Confidence driven unsupervised semantic parsing. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies (ACL-HLT)*. Association for Computational Linguistics, Stroudsburg, PA, USA, 1486–1495.
- GORNIK, P. AND ROY, D. 2007. Situated language understanding as filtering perceived affordances. *Cognitive Science* 31, 2, 197–231.
- HALL, M., FRANK, E., HOLMES, G., PFAHRINGER, B., REUTEMANN, P., AND WITTEN, I. H. 2009. The WEKA data mining software: an update. *ACM Special Interest Group on Knowledge Discovery in Data and Data Mining Explorations Newsletter* 11, 1, 10–18.
- HOFFMANN, J. 2003. The Metric-FF planning system: Translating “ignoring delete lists” to numeric state variables. *Journal of Artificial Intelligence Research (JAIR)* 20, 291–341.

- JONES, B. K., JOHNSON, M., AND GOLDWATER, S. 2012. Semantic parsing with bayesian tree transducers. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Long Papers (ACL)*. Association for Computational Linguistics, Stroudsburg, PA, USA, 488–496.
- KOLLAR, T., TELLEX, S., ROY, D., AND ROY, N. 2010. Toward understanding natural language directions. In *Proceedings of the 5th ACM/IEEE international conference on Human-robot interaction (HRI)*. IEEE Press, Piscataway, NJ, USA, 259–266.
- KOLLER, A., DEBUSMANN, R., GABSDIL, M., AND STRIEGNITZ, K. 2004. Put my galakmid coin into the dispenser and kick it: Computational linguistics and theorem proving in a computer game. *Journal of Logic, Language and Information* 13, 2, 187–206.
- KOLLER, A., STRIEGNITZ, K., GARGETT, A., BYRON, D., CASSELL, J., DALE, R., MOORE, J., AND OBERLANDER, J. 2010. Report on the second challenge on generating instructions in virtual environments (GIVE-2). In *Proceedings of the 6th International Natural Language Generation Conference (INLG)*. Association for Computational Linguistics, Dublin.
- KWIATKOWSKI, T., ZETTMLOYER, L., GOLDWATER, S., AND STEEDMAN, M. 2010. Inducing probabilistic CCG grammars from logical form with higher-order unification. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Association for Computational Linguistics, Stroudsburg, PA, USA, 1223–1233.
- LAU, T., CERRUTI, J., MANZATO, G., BENGUALID, M., BIGHAM, J. P., AND NICHOLS, J. 2010. A conversational interface to web automation. In *Proceedings of the 23rd annual ACM symposium on User interface software and technology (UIST)*. ACM, New York, NY, USA, 229–238.
- LAU, T., DREWS, C., AND NICHOLS, J. 2009. Interpreting written how-to instructions. In *Proceedings of the 21st International Joint Conference on Artificial Intelligence (IJCAI)*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1433–1438.
- LEUSKI, A., EICKHOFF, C., GANIS, J., AND LAVRENKO, V. 2012. The BladeMistress corpus: From talk to action in virtual worlds. In *Proceedings of the Eighth International Conference on Language Resources and Evaluation (LREC)*, N. Calzolari, K. Choukri, T. Declerck, M. U. Dogan, B. Maegaard, J. Mariani, J. Odiijk, and S. Piperidis, Eds. European Language Resources Association, Istanbul, Turkey, May 23–25, 4060–4067.
- LEVENSHTIN, V. 1966. Binary codes capable of correcting deletions, insertions, and reversals. *Cybernetics and Control Theory* 10, 8, 707–710.
- LIANG, P., JORDAN, M., AND KLEIN, D. 2013. Learning Dependency-Based Compositional Semantics. *Computational Linguistics* 39, 2, 398–446.
- MACMAHON, M., STANKIEWICZ, B., AND KUIPERS, B. 2006. Walk the talk: connecting language, knowledge, and action in route instructions. In *Proceedings of the 21st National Conference on Artificial Intelligence - Volume 2 (AAAI)*. AAAI Press, 1475–1482.
- MATUSZEK, C., FOX, D., AND KOSCHER, K. 2010. Following directions using statistical machine translation. In *Proceedings of the 5th ACM/IEEE international conference on Human-robot interaction (HRI)*. ACM, New York, NY, USA, 251–258.
- MURTHY, S. K. 1998. Automatic construction of decision trees from data: A multi-disciplinary survey. *Data mining and knowledge discovery* 2, 4, 345–389.
- NAU, D., GHALLAB, M., AND TRAVERSO, P. 2004. *Automated Planning: Theory & Practice*. Morgan Kaufmann Publishers Inc., California, USA.
- NIKRAVESH, M., TAKAGI, T., TAJIMA, M., SHINMURA, A., OHGAYA, R., TANIGUCHI, K., KAWAHARA, K., FUKANO, K., AND AIZAWA, A. 2005. Soft computing for perception-based decision processing and analysis: Web-based BISC-DSS. In *Soft Computing for Information Processing and Analysis*, M. Nikraves, L. Zadeh, and J. Kacprzyk, Eds. Studies in Fuzziness and Soft Computing Series, vol. 164. Springer, Chapter 4, 93–188.
- ORKIN, J. AND ROY, D. 2009. Automatic learning and generation of social behavior from collective human gameplay. In *Proceedings of The 8th International Conference on Autonomous Agents and Multiagent Systems - Volume 1*. Vol. 1. International Foundation for Autonomous Agents and Multiagent Systems, 385–392.
- ORKINS, J. AND ROY, D. 2007. The restaurant game: Learning social behavior and language from thousands of players online. *Journal of Game Development* 3, 1, 39–60.
- PAPINENI, K., ROUKOS, S., WARD, T., AND ZHU, W.-J. 2002. BLEU: a method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics (ACL)*. Association for Computational Linguistics, Stroudsburg, PA, USA, 311–318.
- PURVER, M. 2004. The theory and use of clarification requests in dialogue. Ph.D. thesis, King's College, University of London.



- RIESER, V. AND LEMON, O. 2010. Learning human multimodal dialogue strategies. *Natural Language Engineering* 16, 3–23.
- SMALL, S. G., STROMER-GALLEY, J., AND STRZALKOWSKI, T. 2011. Multi-modal annotation of quest games in second life. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies - Volume 1 (ACL-HLT)*. Association for Computational Linguistics, Stroudsburg, PA, USA, 171–179.
- STOIA, L., BYRON, D. K., SHOCKLEY, D. M., AND FOSLER-LUSSIER, E. 2006. Sentence planning for real-time navigational instructions. In *Proceedings of the Human Language Technology Conference of the NAACL, Companion Volume: Short Papers*. NAACL-Short '06. Association for Computational Linguistics, Stroudsburg, PA, USA, 157–160.
- VOGEL, A. AND JURAFSKY, D. 2010. Learning to follow navigational directions. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics (ACL)*. Association for Computational Linguistics, Stroudsburg, PA, USA, 806–814.
- WILLIAMS, J. D. AND YOUNG, S. 2007. Partially observable markov decision processes for spoken dialog systems. *Computer Speech and Language* 21, 2, 393–422.
- WINOGRAD, T. 1972. *Understanding Natural Language*. Academic Press, New York.