Semantic Discourse Segmentation and Labeling for Route Instructions

Nobuyuki Shimizu
Department of Computer Science
State University of New York at Albany
Albany, NY 12222, USA
nobuyuki@shimizu.name

Abstract

In order to build a simulated robot that accepts instructions in unconstrained natural language, a corpus of 427 route instructions was collected from human subjects in the office navigation domain. The instructions were segmented by the steps in the actual route and labeled with the action taken in each step. This flat formulation reduced the problem to an IE/Segmentation task, to which we applied Conditional Random Fields. We compared the performance of CRFs with a set of hand-written rules. The result showed that CRFs perform better with a 73.7% success rate.

1 Introduction

To have seamless interactions with computers, advances in task-oriented deep semantic understanding are of utmost importance. The examples include tutoring, dialogue systems and the one described in this paper, a natural language interface to mobile robots. Compared to more typical text processing tasks on newspapers for which we attempt shallow understandings and broad coverage, for these domains vocabulary is limited and very strong domain knowledge is available. Despite this, deeper understanding of unrestricted natural language instructions poses a real challenge, due to the incredibly rich structures and creative expressions that people use. For example,

> ”Just head straight through the hallway ignoring the rooms to the left and right of you, but while going straight your going to eventually see a room facing you, which is north, enter it.”

> ”Head straight. continue straight past the first three doors until you hit a corner. On that corner there are two doors, one straight ahead of you and one on the right. Turn right and enter the room to the right and stop within.”

These utterances are taken from an office navigation corpus collected from undergrad volunteers at SUNY/Albany. There is a good deal of variety.

Previous efforts in this domain include the classic SHRDLU program by Winograd (1972), using a simulated robot, and the more ambitious IBL (Instruction-based Learning for Mobile Robots) project (Lauria et al, 2001) which tried to integrate vision, voice recognition, natural language understanding and robotics. This group has yet to publish performance statistics. In this paper we will focus on the application of machine learning to the understanding of written route instructions, and on testing by following the instructions in a simulated office environment.

2 Task

2.1 Input and Output

Three inputs are required for the task:

- Directions for reaching an office, written in unrestricted English.
- A description of the building we are traveling through.
- The agent’s initial position and orientation.

The output is the location of the office the directions aim to reach.

2.2 Corpus Collection

In an experiment to collect the corpus, (Haas, 1995) created a simulated office building modeled after the actual computer science department at SUNY/Albany. This environment was set up like a popular first person shooter game such as Doom, and the subject saw a demonstration of the route he/she was asked to describe. The subject wrote directions and sent them to the experimenter, who sat at another computer in the next room. The experimenter tried to follow the directions; if he reaches the right destination, the subject got $1. This process took place 10 times for each subject; instructions that the experimenter could not follow correctly were not added to the corpus. In this manner, they were able to elicit 427 route instructions from the subject pool of 44 undergraduate students.

2.3 Abstract Map

To simplify the learning task, the map of our computer science department was abstracted to a graph. Imagine a track running down the halls of the virtual building, with branches into the office doors. The nodes of the graph are the intersections, the edges are the pieces of track between them. We assume this map can either be prepared ahead of time, or dynamically created as a result of solving Simultaneous Localization and Mapping (SLAM) problem in robotics (Montemerlo et al, 2003).

2.4 System Components

Since it is difficult to jump ahead and learn the whole input-output association as described in the task section, we will break down the system into two components.

Front End:
$RouteInstruction \rightarrow ActionList$
Back End:
$ActionList \times Map \times Start \rightarrow Goal$

The front-end is an information extraction system, where the system extracts how one should move from a route instruction. The back-end is a reasoning system which takes a sequence of moves and finds the destination in the map. We will first describe the front-end, and then show how to integrate the back-end to it.

One possibility is to keep the semantic representation close to the surface structure, including under-specification and ambiguity, and leaving the back-end to resolve the ambiguity. We will pursue a different route. The disambiguation will be done in the front-end; the representation that it passes to the back-end will be unambiguous, describing at most one path through the building. The task of the back-end is simply to check the sequence of moves the front-end produced against the map and see if there is a path leading to a point in the map or not. The reason for this is two fold. One is to have a minimal annotation scheme for the corpus, and the other is to enable the learning of the whole task including the disambiguation as an IE problem.

3 Semantic Analysis

Note that in this paper, given an instruction, one *step* in the instruction corresponds to one *action* shown to the subject, one *episode* of action detection and tracking, and one *segment* of the text.

In order to annotate unambiguously, we need to detect and track both landmarks and actions. A **landmark** is a hallway or a door, and an **action** is a sequence of a few moves one will make with respect to a specific landmark.

The moves one can make in this map are:
(M1). Advancing to x,
(M2). Turning left/right to face x, and
(M3). Entering x.

Here, x is a landmark. Note that all three moves have to do with the same landmark, and two or three moves on the same landmark constitute one action. An action is ambiguous until x is filled with an unambiguous landmark. The following is a made-up example in which each move in an action is mentioned explicitly.

a. ”Go down the hallway to the second door on the right. Turn right. Enter the door.”

But you could break it down even further.

b. ”Go down the hallway. You will see two doors on the right. Turn right and enter the second.”

One can add any amount of extra information to an instruction and make it longer, which people seem to do. However, we see the following as well.

c. ”Enter the second door on the right.”

In one sentence, this sample contains the advance, the turn and the entering. In the corpus, the norm is to assume the move (M1) when an expression indicating the move (M2) is present. Similarly, an expression of move (M3) often implicitly assumes the move (M1) and (M2). However, in some cases they are explicitly stated, and when this happens, the action that involves the same landmark must be tracked across the sentences.

Since all three samples result in the same action, for the back-end it is best not to differentiate the three. In order to do this, actions must be tracked just like landmarks in the corpus.

The following two samples illustrate the need to track actions.

> d. ”Go down the hallway until you see two doors. Turn right and enter the second door on the right.”

In this case, there is only one action in the instruction, and ”turn right” belongs to the action *”advance to the second door on the right, and then turn right to face it, and then enter it.*”

> e. ”Proceed to the first hallway on the right. Turn right and enter the second door on the right.”

There are two actions in this instruction. The first is *”advance to the first hallway on the right, and then turn right to face the hallway.*” The phrase ”turn right” belongs to this first action. The second action is the same as the one in the example (d). Unless we can differentiate between the two, the execution of the unnecessary turn results in failure when following the instructions in the case (d).

This illustrates the need to track actions across a few sentences. In the last example, it is important to realize that ”turn right” has something to do with a door, so that it means ”turn right to face a door”. Furthermore, since ”enter the second door on the right” contains ”turning right to face a door” in its semantics as well, they can be thought of as the same action. Thus, the critical feature required in the annotation scheme is to track actions and landmarks.

The simplest annotation scheme that can show how actions are tracked across the sentences is to segment the instruction into different episodes of action detection and tracking. Note that each episode corresponds to exactly one action shown to the subject during the experiment. The annotation is based on the semantics, not on the the *mentions* of moves or landmarks. Since each segment involves exactly one landmark, we can label the segment with an action and a specific landmark. For example,

Token	Node Part	Transition Part
make	$\langle$*B-GHL1*$, 0\rangle$	$\langle$*B-GHL1*, *I-GHL1*$, 0, 1\rangle$
left	$\langle$*I-GHL1*$, 1\rangle$	$\langle$*I-GHL1*, *I-GHL1*$, 1, 2\rangle$
,	$\langle$*I-GHL1*$, 2\rangle$	$\langle$*I-GHL1*, *B-EDR1*$, 2, 3\rangle$
first	$\langle$*B-EDR1*$, 3\rangle$	$\langle$*B-EDR1*, *I-EDR1*$, 3, 4\rangle$
door	$\langle$*I-EDR1*$, 4\rangle$	$\langle$*I-EDR1*, *I-EDR1*$, 4, 5\rangle$
on	$\langle$*I-EDR1*$, 5\rangle$	$\langle$*I-EDR1*, *I-EDR1*$, 5, 6\rangle$
the	$\langle$*I-EDR1*$, 6\rangle$	$\langle$*I-EDR1*, *I-EDR1*$, 6, 7\rangle$
right	$\langle$*I-EDR1*$, 7\rangle$	

Table 1: Example Parts: linear-chain CRFs

GHR1 := ”advance to the first hallway on the right, then turn right to face it.”

EDR2 := ”advance to the second door on the right, then turn right to face it, then enter it.”

GHLZ := ”advance to the hallway on the left at the end of the hallway, then turn left to face it.”

EDSZ := ”advance to the door straight ahead of you, then enter it.”

Note that GH=go-hall, ED=enter-door, R1=first-right, LZ=left-at-end, SZ=ahead-of-you. The total number of possible actions is 15.

This way, we can reduce the front-end task into a sequence of tagging tasks, much like the noun phrase chunking in the CoNLL-2000 shared task (Tjong Kim Sang and Buchholz, 2000). Given a sequence of input tokens that forms a route instruction, a sequence of output labels, with each label matching an input token was prepared. We annotated with the BIO tagging scheme used in syntactic chunkers (Ramshaw and Marcus, 1995).

```
make        B-GHL1
left        I-GHL1
,           I-GHL1
first       B-EDR1
door        I-EDR1
on          I-EDR1
the         I-EDR1
right       I-EDR1
```

4 Systems

4.1 System 1: CRFs

4.1.1 Model: A Linear-Chain Undirected Graphical Model

From the output labels, we create the parts in a linear-chain undirected graph (Table 1). Our use of term **part** is based on (Bartlett et al, 2004).

For each pair (x^i, y^i) in the training set, x^i is the token (in the first column, Table 1), and y^i

Transition $\langle L0, L, j-1, j\rangle$	Node $\langle L, j\rangle$
no lexicalization	no lexicalization
	x_{j-4}
	x_{j-3}
	x_{j-2}
	x_{j-1}
	x_j
	x_{j+1}
	x_{j+2}
	x_{j+3}
	x_{j-1}, x_j
	x_{j+0}, x_{j+1}

Table 2: Features

is the part (in the second and third column, Table 1). There are two kinds of parts: node and transition. A node part tells us the position and the label, $\langle$*B-GHL1*$, 0\rangle$, $\langle$*I-GHL1*$, 1\rangle$, and so on. A transition part encodes a transition. For example, between tokens 0 and 1 there is a transition from tag B-GHL1 to I-GHL1. The part that describes this transition is: $\langle$*B-GHL1*, *I-GHL1*$, 0, 1\rangle$.

We factor the score of this linear node-transition structure as the sum of the scores of all the parts in y, where the score of a part is again the sum of the feature weights for that part.

To score a pair (x^i, y^i) in the training set, we take each part in y^i and check the features associated with it via lexicalization. For example, a part $\langle$*I-GHL1*$, 1\rangle$ could give rise to binary features such as,

- Does (x^i, y^i) contain a label "I-GHL1"? (No Lexicalization)
- Does (x^i, y^i) contain a token "left" labeled with "I-GHL1"? (Lexicalized by x_1)
- Does (x^i, y^i) contain a token "left" labeled with "I-GHL1" that's preceded by "make"? (Lexicalized by x_0, x_1)

and so on. The features used in this experiment are listed in Table 2.

If a feature is present, the feature weight is added. The sum of the weights of all the parts is the score of the pair (x^i, y^i). To represent this summation, we write $s(x^i, y^i) = \mathbf{w}^\top \mathbf{f}(x^i, y^i)$ where $\mathbf{f}$ represents the feature vector and $\mathbf{w}$ is the weight vector. We could also have $\mathbf{w}^\top \mathbf{f}(x^i, \{p\})$ where p is a single part, in which case we just write $s(p)$.

Assuming an appropriate feature representation as well as a weight vector $\mathbf{w}$, we would like to find the highest scoring $y = argmax_{y'}(\mathbf{w}_k^\top \mathbf{f}(y', x))$ given an input sequence x. We next present a version of this decoding algorithm that returns the best y consistent with the map.

4.1.2 Decoding: the Viterbi Algorithm and Inferring the Path in the Map

The action labels are unambiguous; given the current position, the map, and the action label, there is only one position one can go to. This backend computation can be integrated into the Viterbi algorithm. The function 'go' takes a pair of (action label, start position) and returns the end position or null if the action cannot be executed at the start position according to the map. The algorithm chooses the best among the label sequences with a legal path in the map, as required by the condition $(cost > bestc \wedge end \neq null)$. Once the model is trained, we can then use the modified version of the Viterbi algorithm (Algorithm 4.1) to find the destination in the map.

```
**Algorithm 4.1:** DECODE PATH($x, n, start, go$)

for each label $y_1$
  $node[0][y_1].cost \leftarrow s(\langle y_1, 0\rangle)$
  $node[0][y_1].end \leftarrow start;$
for $j \leftarrow 1$ to $n-1$
  for each label $y_{j+1}$
    $bestc \leftarrow -\infty;$
    $end \leftarrow null;$
    for each label $y_j$
      $cost \leftarrow node[j][y_j].cost$
        $+s(\langle y_j, y_{j+1}, j, j+1\rangle)$
        $+s(\langle y_{j+1}, j+1\rangle);$
      $end \leftarrow node[j][y_j].end;$
      if $(y_j \neq y_{j+1})$
        $end \leftarrow go(y_{j+1}, end);$
      if $(cost > bestc \wedge end \neq null)$
        $bestc \leftarrow cost;$
    if $(bestc \neq -\infty)$
      $node[j+1][y_{j+1}].cost \leftarrow bestc;$
      $node[j+1][y_{j+1}].end \leftarrow end;$
$bestc \leftarrow -\infty;$
$end \leftarrow null;$
for each label $y_n$
  if $(node[j][y_n].cost > bestc)$
    $bestc \leftarrow node[j][y_n].cost;$
    $end \leftarrow node[j][y_n].end;$
return $(bestc, end)$
```

4.1.3 Learning: Conditional Random Fields

Given the above problem formulation, we trained the linear-chain undirected graphical model as Conditional Random Fields (Lafferty et al, 2001; Sha and Pereira, 2003), one of the best performing chunkers. We assume the probability of seeing y given x is

$$P(y|x) = \frac{exp(s(x,y))}{\sum_{y'} exp(s(x,y'))}$$

where y' is all possible labeling for x , Now, given a training set $T = \{(x^i y^i)\}_{i=1}^{m}$, We can learn the weights by maximizing the log-likelihood, $\sum_i logP(y^i|x^i)$. A detailed description of CRFs can be found in (Lafferty et al, 2001; Sha and Pereira, 2003; Malouf, 2002; Peng and McCallum, 2004). We used an implementation called CRF++ which can be found in (Kudo, 2005)

4.2 System 2: Baseline

Suppose we have clean data and there is no need to track an action across sentences or phrases. Then, the properties of an action are mentioned exactly once for each episode.

For example, in *”go straight and make the first left you can, then go into the first door on the right side and stop”* , LEFT and FIRST occur exactly once for the first action, and FIRST, DOOR and RIGHT are found exactly once in the next action. In a case like that, the following baseline algorithm should work well.

- Find all the mentions of LEFT/RIGHT,
- For each occurrence of LEFT/RIGHT, look for an ordinal number, LAST, or END (= end of the hallway) nearby,
- Also, for each LEFT/RIGHT, look for a mention of DOOR. If DOOR is mentioned, the action is about entering a door.
- If DOOR is not mentioned around LEFT/RIGHT, then the action is about going to a hallway by default,
- If DOOR is mentioned at the end of an instruction without LEFT/RIGHT, then the action is to go straight into the room.
- Put the sequence of action labels together according to the mentions collected.

	count	average length
GHL1	128	8.5
GHL2	4	7.7
GHLZ	36	14.4
GHR1	175	10.8
GHR2	5	15.8
GHRZ	42	13.6
EDL1	98	10.5
EDL2	81	12.3
EDL3	24	13.9
EDLZ	28	13.7
EDR1	69	10.4
EDR2	55	12.9
EDR3	6	13.0
EDRZ	11	16.4
EDSZ	55	16.2

Table 3: Steps found in the dataset

In this case, all that’s required is a dictionary of how a word maps to a concept such as DOOR. In this corpus, ”door”, ”office”, ”room”, ”doorway” and their plural forms map to DOOR, and the ordinal number 1 will be represented by ”first” and ”1st”, and so on.

5 Dataset

As noted, we have 427 route instructions, and the average number of steps was 1.86 steps per instruction. We had 189 cases in which a sentence boundary was found in the middle of a step. Table 3 shows how often action steps occurred in the corpus and average length of the segments.

One thing we noticed is that somehow people do not use a short phrase to say the equivalent of ”enter the door straight ahead of you”, as seen by the average length of EDSZ. Also, it is more common to say the equivalent of ”take a right at the end of the hallway” than that of ”go to the second hallway on the right”, as seen by the count of GHR2 and GHRZ. The distribution is highly skewed; there are a lot more GHL1 than GHL2.

6 Results

We evaluated the performance of the systems using three measures: overlap match, exact match, and instruction follow through, using 6-fold crossvaliadation on 427 samples. Only the action chunks were considered for exact match and overlap match. Overlap match is a lenient measure that considers a segmentation or labeling to be cor-

Exact Match	Recall	Precision	F-1
CRFs	66.0%	67.0%	66.5%

Overlap Match	Recall	Precision	F-1
Baseline	62.8%	49.9%	55.6%
CRFs	85.7%	87.0%	86.3%

Instruction Follow Through	success rate
Baseline	39.5%
CRFs	73.7%

Table 4: Recall, Precision, F-1 and Success Rate

rect if it overlaps with any of the annotated labels. Instruction follow through is the success rate for reaching the destination, and the most important measure of the performance in this domain. Since the baseline algorithm does not identify the token labeled with B-prefix, no exact match comparison is made. The result (Table 4) shows that CRFs perform better with a 73.7% success rate.

7 Future Work

More complex models capable of representing landmarks and actions separately may be applicable to this domain, and it remains to be seen if such models will perform better. Also, some form of co-reference resolution or more sophisticated action tracking should also be considered.

Acknowledgement

We thank Dr. Andrew Haas for introducing us to the problem, collecting the corpus and being very supportive in general.

References

P. Bartlett, M. Collins, B. Taskar and D. McAllester. 2004. Exponentiated gradient algorithms for large-margin structured classification. In *Advances in Neural Information Processing Systems (NIPS)*

A. Haas 1995. Testing a Simulated Robot that Follows Directions. *unpublished*

T. Kudo 2005. CRF++: Yet Another CRF toolkit. Available at *http://chasen.org/˜taku/software/CRF++/*

J. Lafferty, A. McCallum, and F. Pereira. 2001. Conditional Random Fields: Probabilistic Models for Segmenting and Labeling Sequence Data. In *Proceedings of International Conference on Machine Learning* .

R. Malouf. 2002. A Comparison of Algorithms for Maximum Entropy Parameter Estimation. In *Proceedings of Conference of Computational Natural Language Learning*

F. Peng and A. McCallum. 2004. Accurate Information Extraction from Research Papers using Conditional Random Fields. In *Proceedings of Human Language Technology Conference* .

F. Sha and F. Pereira. 2003. Shallow parsing with conditional random fields. In *Proceedings of Human Language Technology Conference* .

S. Lauria, G. Bugmann, T. Kyriacou, J. Bos, and E. Klein. 2001. Personal Robot Training via Natural-Language Instructions. *IEEE Intelligent Systems*, 16:3, pp. 38-45.

C. Manning and H. Schutze. 1999. *Foundations of Statistical Natural Language Processing.* MIT Press.

M. Montemerlo, S. Thrun, D. Koller, and B. Wegbreit. 2003. FastSLAM 2.0: An improved particle filtering algorithm for simultaneous localization and mapping that provably converges. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI).*

L. Ramshaw and M. Marcus. 1995. Text chunking using transformation-based learning. In *Proceedings of Third Workshop on Very Large Corpora. ACL*

E. F. Tjong Kim Sang and S. Buchholz. 2000. Introduction to the CoNLL-2000 shared task: Chunking. In *Proceedings of Conference of Computational Natural Language Learning* .

T. Winograd. 1972. *Understanding Natural Language*. Academic Press.