Fast Online Lexicon Learning for Grounded Language Acquisition

David L. Chen

SE367 Cognitive Science

Instructor
Prof. Amitabh Mukherjee

Presentation by
Abul Aala Nalband

Learning to Interpret Natural Language

- Recent work:
  - How to map natural-language instructions into actions that can be performed by a computer particularly the task of navigation

- Goal of the navigation task:
  - take a set of natural language directions
  - transform it into a navigation plan that can be understood by the computer
  - execute that plan to reach the desired destination

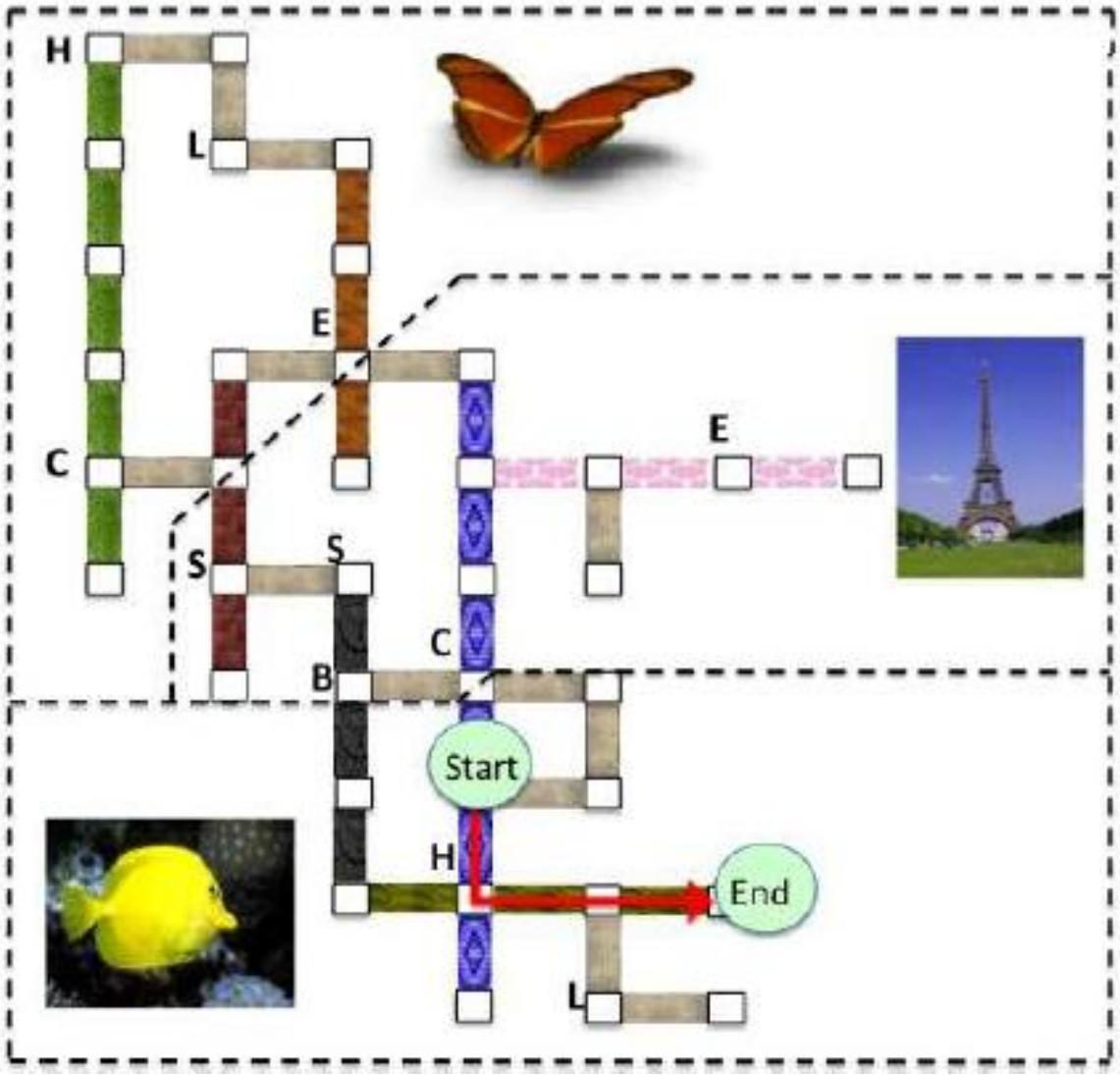


Fig. : This is an example of a route in our virtual world. The world consists of interconnecting hallways with varying floor tiles and paintings on the wall (butterfly, fish, or Eiffel Tower.) Letters indicate objects (e.g. 'C' is a chair) at a location.

- Example instructions :
  - “Go towards the coat rack and take a left at the coat rack, go all the way to the end of the hall and this is 4”
  - “Position 4 is a dead end of the yellow floored hall with fish on the walls”
  - “turn so that the wall is on your right side, walk forward once, turn left, walk forward twice”

- Challenges:
  - Even ignoring spelling and grammatical errors as well as logical errors, navigation instructions can be quite diverse and contain different information which makes interpreting them a challenging problem

- Approach:
  - The system is given the training data in the form of :

    $\{(e_1, a_1, w_1), (e_2, a_2, w_2), \ldots, (e_n, a_n, w_n)\}$

    where
    - e_i is a natural language instruction,
    - a_i is an observed action sequence,
    - w_i is a description of the current state of the world including the patterns of the floors and walls and positions of any objects
- Objective:
  - To build a system that can produce the correct a_j given a previously unseen (e_j, w_j) pair

- Problem here?
  - Direct correspondence between e_i and a_i is not possible

- But, e_i corresponds to an unobserved plan p_i that when executed in w_i will produce a_i
- Thus, we need to first infer the correct p_i from the training data and then build a semantic parser that can translate from e_i to p_i

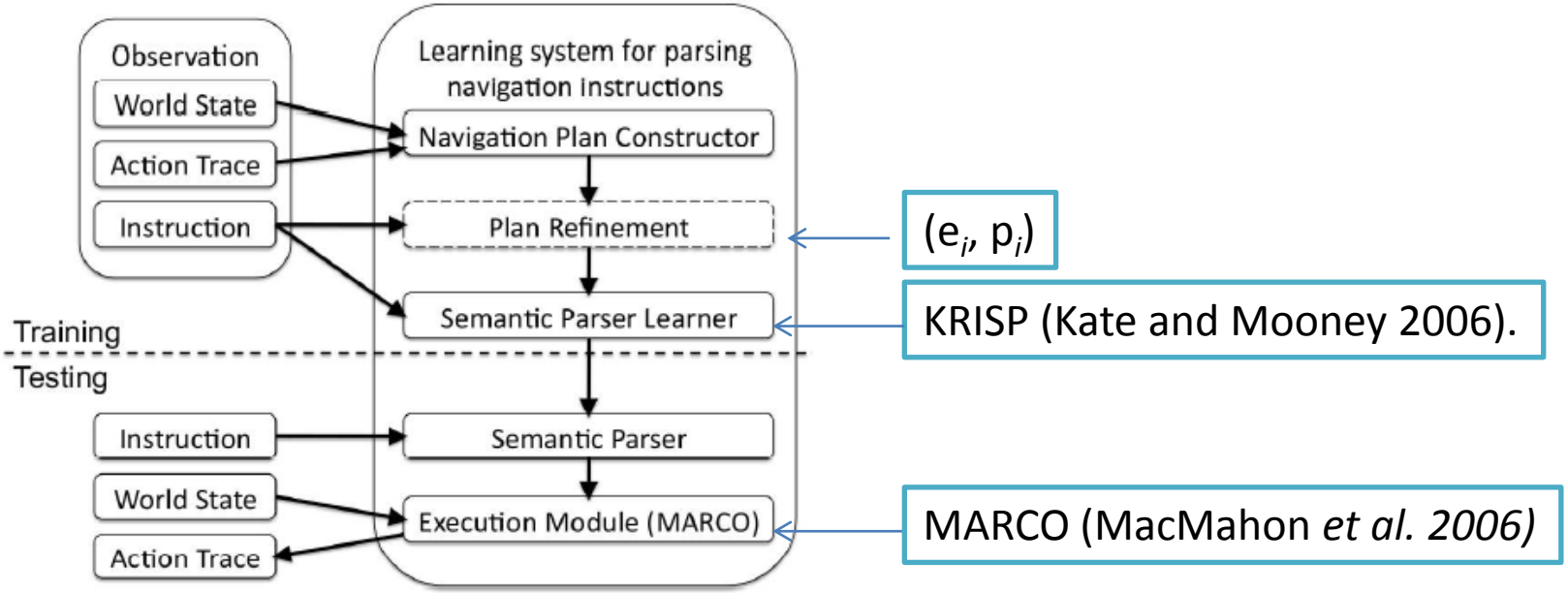


Fig. : Overview of the system

- Navigation plan:
  - Basic plans - turn left, walk forward two steps
  - Landmarks plan - face the pink flower hallway, go to the sofa
- Learning a lexicon
  - Build a semantic lexicon by finding the common parts of the formal representations associated with different occurrences of the same word or phrases
  - We represent the navigation plans in graphical form and compute common parts by taking intersections of the two graphs
- Scoring function & refining:
  - To evaluate a pair of an n-gram w and a graph g:

    $$\text{Score}(w, g) = p(g|w) - p(g|\neg w)$$

  - Refining the plan p_i to p'_i by removing extra components from landmark plans
- Advantage:
  - The algorithm produced a good lexicon for their application of learning to interpret navigation instructions
- Drawbacks:
  - It only works in batch settings and does not scale well to large datasets
  - Intersection process is time-consuming to perform.

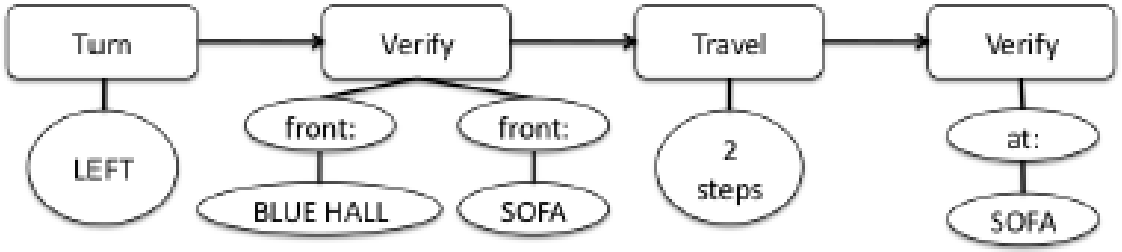
Instruction: "Go away from the lamp to the intersection of the red brick and wood"

Basic:
Turn ( ) ,
Travel (steps: 1)

Landmarks:
Turn ( ) ,
Verify (left: WALL , back: LAMP , back: HATRACK , front: BRICK HALL) ,
Travel (steps: 1) ,
Verify (side: WOOD HALL)

Fig. : Example of automatically generated plans

Training Example 1: Turn and walk to the couch



Training Example 2: Walk to the couch and turn left

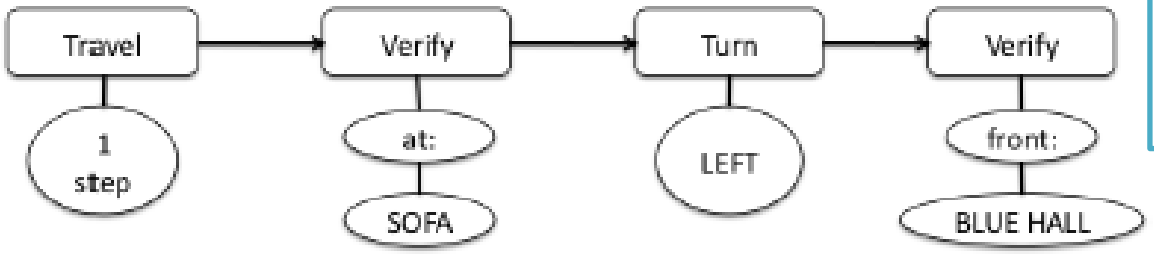
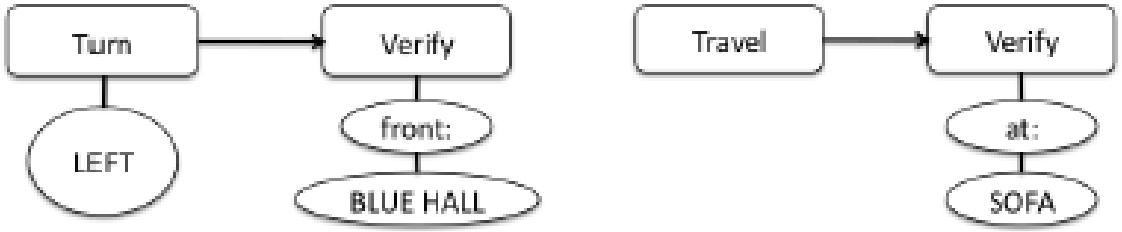


Fig. : Examples of *landmarks plans and intersections constructed*

Intersection:



- Modifications:
  - Subgraph Generation Online Lexicon Learning (SGOLL) algorithm
    - Main insight is that most words or short phrases correspond to small graphs. Therefore we concentrate our attention on only candidate meanings that are less than a certain size.
  - Modifying the meaning representation grammar (MRG) for their formal semantic language

Pseudo-code for SGOLL algorithm

input A sequence of navigation instructions and the corresponding navigation plans $(e_1, p_1), \ldots, (e_n, p_n)$

output $Lexicon$, a set of phrase-meaning pairs

```
1:  main
2:      for training example (e_i, p_i) do
3:          Update((e_i, p_i))
4:      end for
5:      OutputLexicon()
6:  end main
7:
8:  function Update(training example (e_i, p_i))
9:      for n-gram w that appears in e_i do
10:         for connected subgraph g of p_i such that
            the size of g is less than or equal to m do
11:             Increase the co-occurrence count of g
                and w by 1
12:         end for
```

Main function

Here each (e_i, p_i) is processed and output is given

Update function

Here occurrence of each w against connected sub graph (g < m)of each p is validated

```
13:     end for
14:     Increase the count of examples, each n-gram
        w and each subgraph g
15: end function
16:
17:
18: function OutputLexicon()
19:     for n-gram w that has been observed do
20:         if Number of times w has been observed is
            less than minSup then
21:             skip w
22:         end if
23:         for subgraph g that has co-occurred with w
            do
24:             if score(w, g) > threshold t then
25:                 add (w, g) to Lexicon
26:             end if
27:         end for
28:     end for
29: end function
```

Output Lexicon function

Here only w > minimum support are used to calculate the score which if > t, then added to lexicon.

Default parameters

For up to 4-grams with threshold *t = 0.4, maximum s*ubgraph size *m = 3 and minimum support minSup = 10*

- Changing the Meaning Representation Grammar(MRG):
  - KRISP learns string-kernel classifiers that maps natural language substrings to MRG production rules.
- Original MRG:
  - contains many recursive rules that can generate an infinite number of actions or arguments. But, they often do not correspond well to any words or phrases in natural language
- For example, the rule in the

Original MRG

Generates an infinite number of travel actions from the root symbol say *S.*

Modified MRG

Generates an specific travel actions so they correspond better to patterns such as “go forward” or “walk N steps”.

- Experiments and Statistical results:

	Precision	Recall	F1
Chen and Mooney	90.22	55.10	68.37
SGOLL	92.22	55.70	69.43
SGOLL with new MRG	88.36	57.03	69.31

Fig. : Partial parse accuracy of the semantic parsers trained on the disambiguated navigation plans.

	Single-sentence	Complete
Chen and Mooney	54.40%	16.18%
SGOLL	57.09%	17.56%
SGOLL with new MRG	57.28%	19.18%
SGOLL with new MRG and MTurk data	57.62%	20.64%

Fig. : End-to-end navigation task completion rates.

	Computation Time
Chen and Mooney (2011)	2,227.63
SGOLL	157.30
SGOLL with MTurk data	233.27

Fig. : The time (in seconds) it took to build the lexicon.

- Chinese data – Experimental verification:
  - Showed that the results are very similar to English results.
  - Shows the generality of the system in its ability to learn other languages.

- **References:**

  - Learning to Interpret Natural Language Navigation Instructions from Observations. David L. Chen and Raymond J. Mooney (2011)

  - Fast Online Lexicon Learning for Grounded Language Acquisition. David L. Chen(2012)