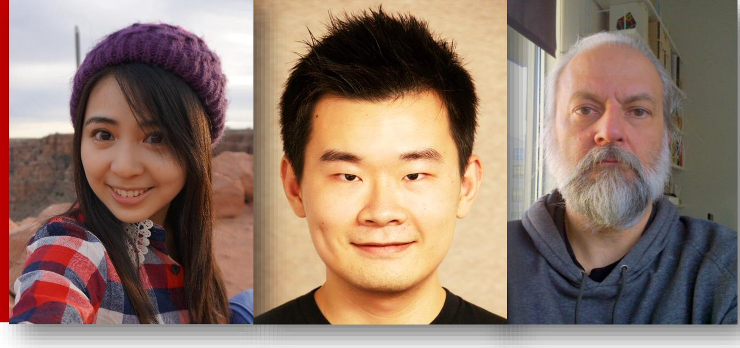
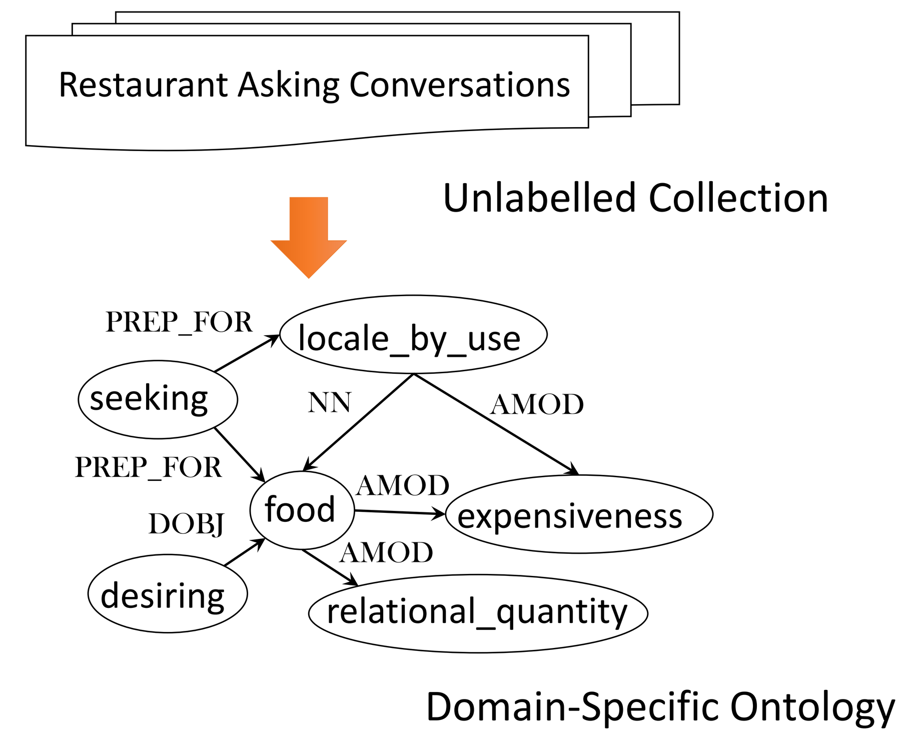
Jointly Modeling Inter-Slot Relations by Random Walk on Knowledge Graphs for Unsupervised Spoken Language Understanding

Yun-Nung (Vivian) Chen, William Yang Wang, and Alexander I. Rudnicky



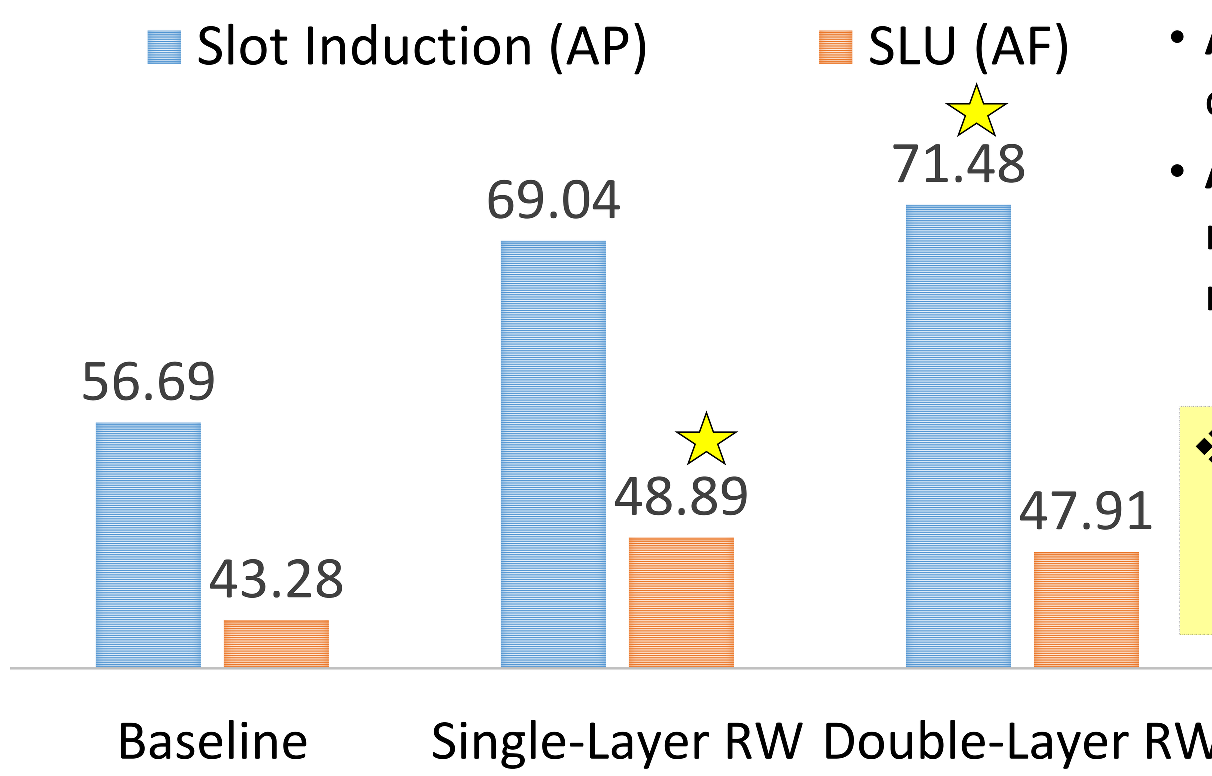
Summary

- Motivation
  - SDSs require a predefined semantic ontology; can it be learned from data?
  - Inter-slot relations can provide a coherent ontology
  - Typed dependencies enable learning of ontology structure
- Approach
  1) Construct two-layer knowledge graph represent slots, words, and relations
  2) Compute scores for edges (relations) and nodes (slots) by random walk
  3) Identify important slots associated with relations
- Result
  - Using embedding similarity, we achieve 70% AP for slot induction and 48% AF for SLU
  - The automatically acquired slot set enhances the interpretability of semantic slots



Experimental Results

- Domain: restaurant recommendation in an in-car setting (WER = 37%)
  - Dialogue slots: addr, area, food, phone, postcode, pricerange, task, type



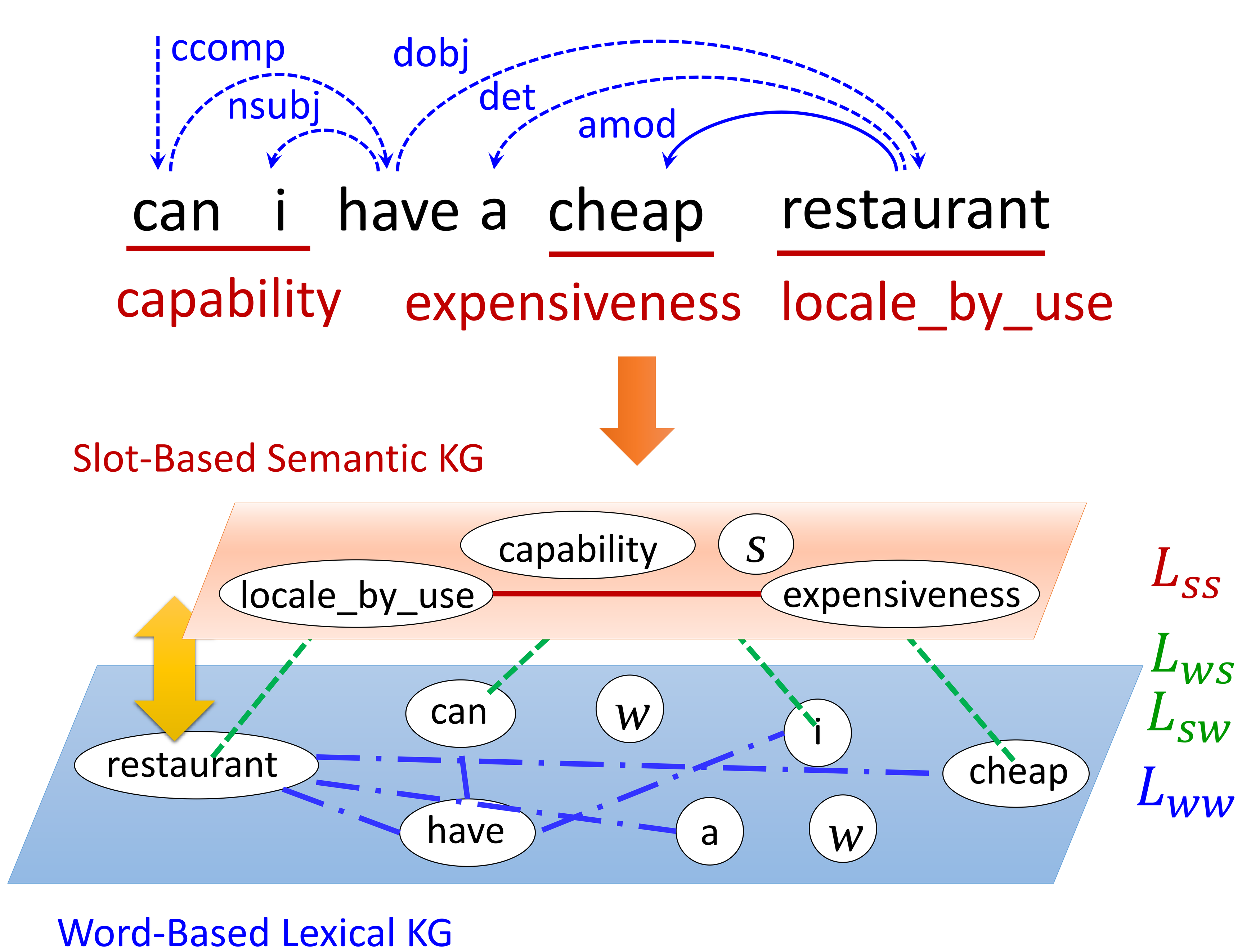
- **AP:** Average Precision given a ranked list of induced slots and associated scores
- **AF:** Average micro F-measure of SLU models at all cut-off positions in the ranked list

❖ The double-layer random walk performs best for slot induction and almost best for SLU.

Can a dialogue system automatically learn open domain knowledge?

Step 1: Knowledge Graph Construction

Structure can be constructed via the unlabeled collection with frame-semantic parsing and syntactic dependency parsing.



Step 2: Weight Measurement

Idea: the edge weights can represent the relation importance

- We train dependency-based word/slot embeddings [1].

Then the relation matrices can be built

- Slot-to-slot relation L_{ss}: similarity between slot embeddings
- Word-to-slot relation L_{ws} or L_{sw}: frequency of the slot-word pair
- Word-to-word relation L_{ww}: similarity between word embeddings

Assumption: the slots with more dependencies to more important slots should be more important

The random walk algorithm computes the importance for each slot

slot importance — original frequency score — scores propagated from word-layer then propagated within slot-layer

$$\begin{cases} \boxed{r_s^{(t+1)}} = (1-\alpha)\boxed{r_s^{(0)}} + \alpha \boxed{L_{ss} L_{sw} r_w^{(t)}} \\ r_w^{(t+1)} = (1-\alpha) r_w^{(0)} + \alpha L_{ww} L_{ws} r_s^{(t)} \end{cases}$$

❖ The converged scores suggest whether the slots are important for this domain based on the automatically built structure [2].

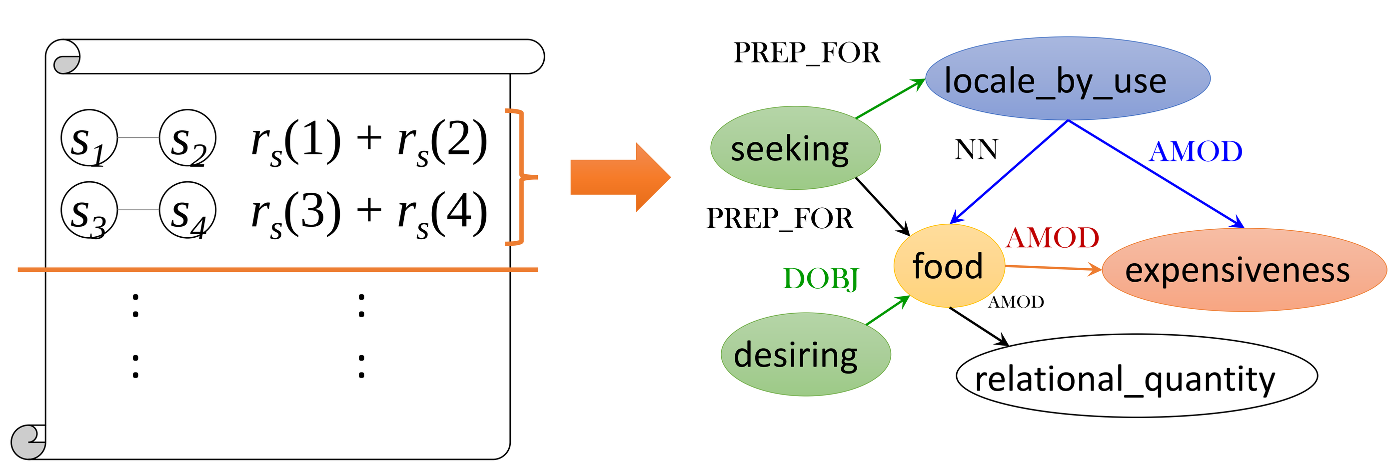
[1] Levy and Goldberg, " Dependency-Based Word Embeddings," Proc. of ACL, 2014.

[2] Released code: https://github.com/yvchen/MRRW

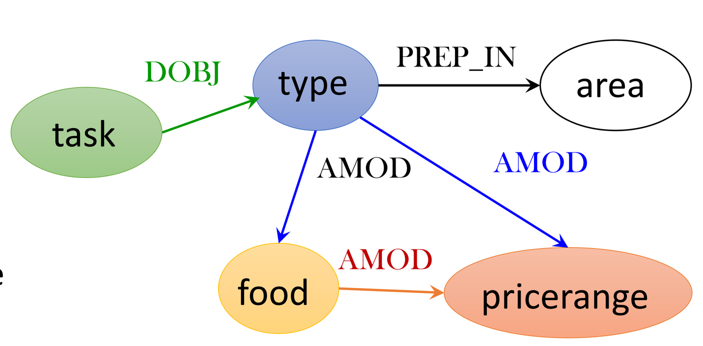
Step 3: Domain Slot/Relation Identification

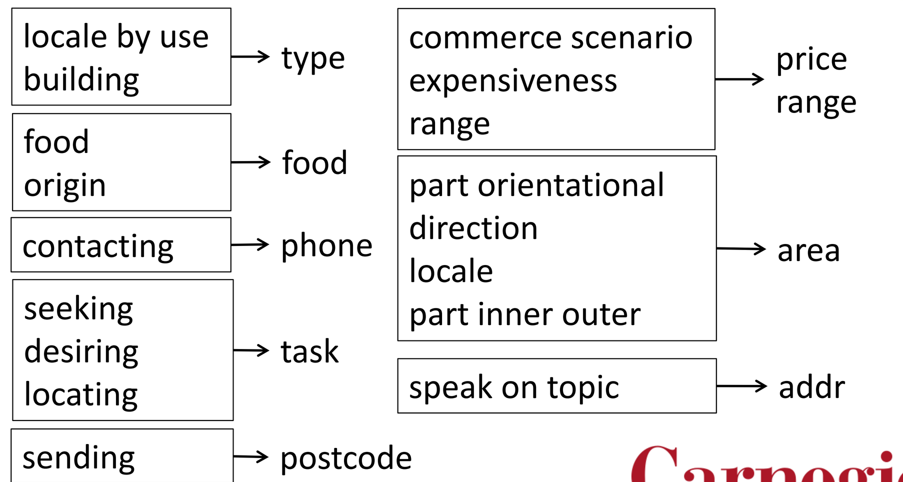
Converged slot importance helps us identify domain ontology:

1. Rank slot pairs by summing up their converged scores
2. Select slot pairs with higher scores according to a threshold



the reference ontology with the most frequent dependencies





Carnegie Mellon University