KU LEUVEN

# NLP models for problem solving

Pietro Totis

SMiLee'19
06/06/2019

# Probability problems

“Given that 10 percent of the population is left-handed, find the probability that out of 15 people, there will be at least three left-handed students.”

NLP

```
group(population).

given(exactly(rel(10/100,population),
population, left-handed)).

take_wr(people, people, 15).

probability(atleast(3, people, left-handed)).

property(population, [left-handed]).
```

Solver

0.18406

# Challenges

- Name consistency
- Quantities recognition
- Background knowledge
- Properties enumeration
- Question understanding
- Constraints recognition
- ...

# Baseline approach

## An end-to-end model: setting

- Single-layer LSTM architecture
- With/without pre-trained word embeddings
- Discretized model:
  - Problem casted as a classification task: predict a “bin” where the correct value falls
- Regression model:
  - Predict the exact value of the solution

D. Saxton, E. Grefenstette, F. Hill, and P. Kohli. Analysing mathematical reasoning abilities of neural models. In ICLR, 2019.

# Baseline approach

## An end-to-end model: results

### Discretized

| Bins | Baseline | LSTM-discrete |
|---|---|---|
| 2 bins | 0.5 | 0.659 |
| 5 bins | 0.2 | 0.375 |
| 10 bins | 0.1 | 0.215 |
| 20 bins | 0.05 | 0.152 |

### Regression

| Model | MAE |
|---|---|
| Random baseline | 0.348 |
| LSTM-regression | 0.206 |

# Predicting predicates

Different networks for different predicates:

- Take with or without replacement
  - Same LSTM architecture, binary classification → ~ 80% accuracy
- Group name
- Property list
- ...

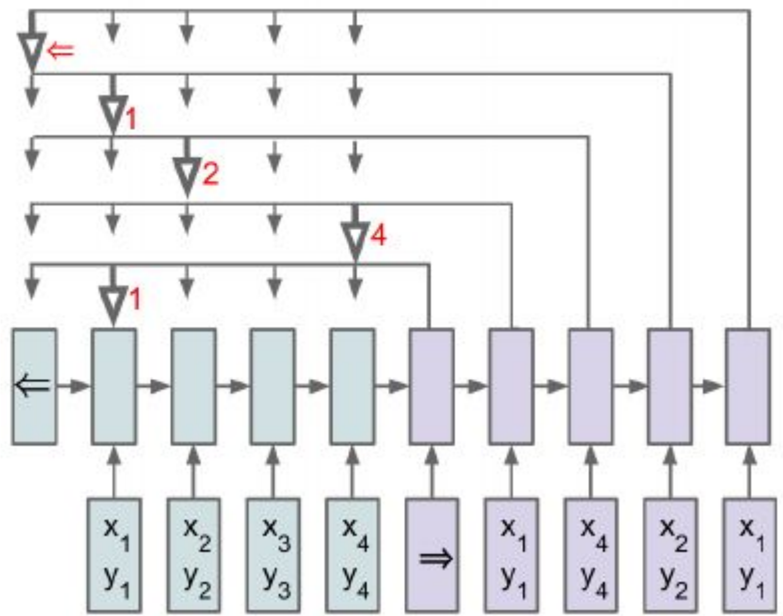
# Pointer networks

**Problem:** traditional RNNs require the output dictionary to be of fixed size.

**Example:** combinatorial optimization problems.

**Solution:** repurpose the attention mechanism to create pointers to input elements.

**Note:** only for problems with discrete outputs corresponding to positions in the input




O. Vinyals, M. Fortunato, and N. Jaitly. Pointer networks. In Advances in Neural Information Processing Systems, pages 2692–2700, 2015b.

# Early results

| Predicate | Acc |
|---|---|
| take($X_1$,_,_) | 0,720 |
| take($X_1$,$X_2$,_) | 0,607 |
| take($X_1$,$X_2$,n) | 0,557 |
| take($X_1$,$X_2$,n) (+ POS-tags) | 0,598 |
| take(_,_,n) | 0,845 |
| take(_,_,n) (+POS-tags) | 0,829 |

| Predicate | Acc |
|---|---|
| take_wr($X_1$,_,_) | 0,598 |
| take_wr($X_1$,$X_2$,_) | 0,436 |
| group(X) | 0,650 |
| group(X) (+ POS-tags) | 0,669 |

# Future work

- Mapping entities and numbers to variables for reducing sparsity
- Syntactic trees as input to the network: more information about the connections between entities, number, properties…
- Coarse-to-fine decoding: predict simple predicates, then refine, i.e. first predict the functor and then the arguments (slot filling approach)
- Output constraints
- …