Syntactic and semantic parsing for natural language understanding

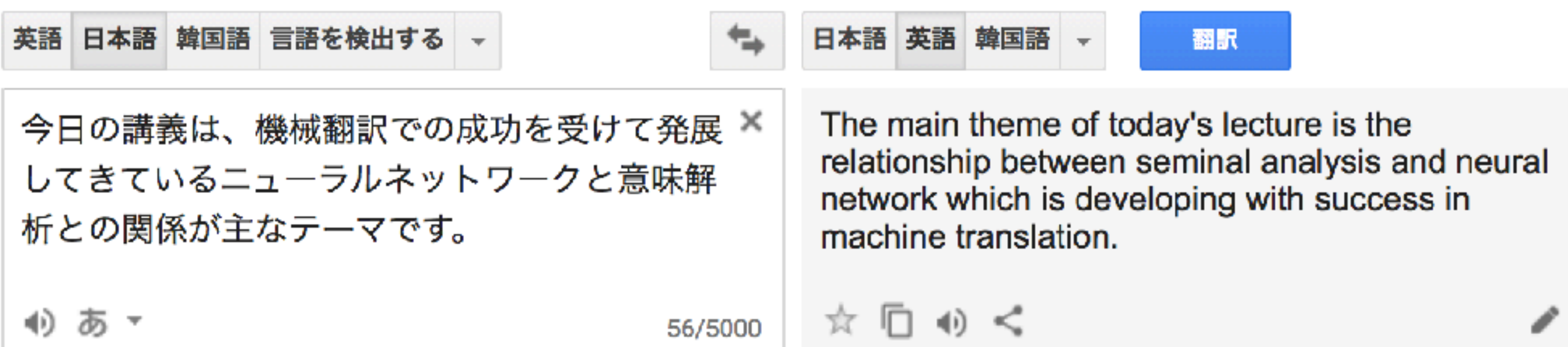
4. Neural semantic parsing

Hiroshi Noji

Neural networks and NLP

- Today we are in the **deep-learning era**
  - Deep learning (neural networks) gets quite successful in many AI problems
  - NLP is not the exception

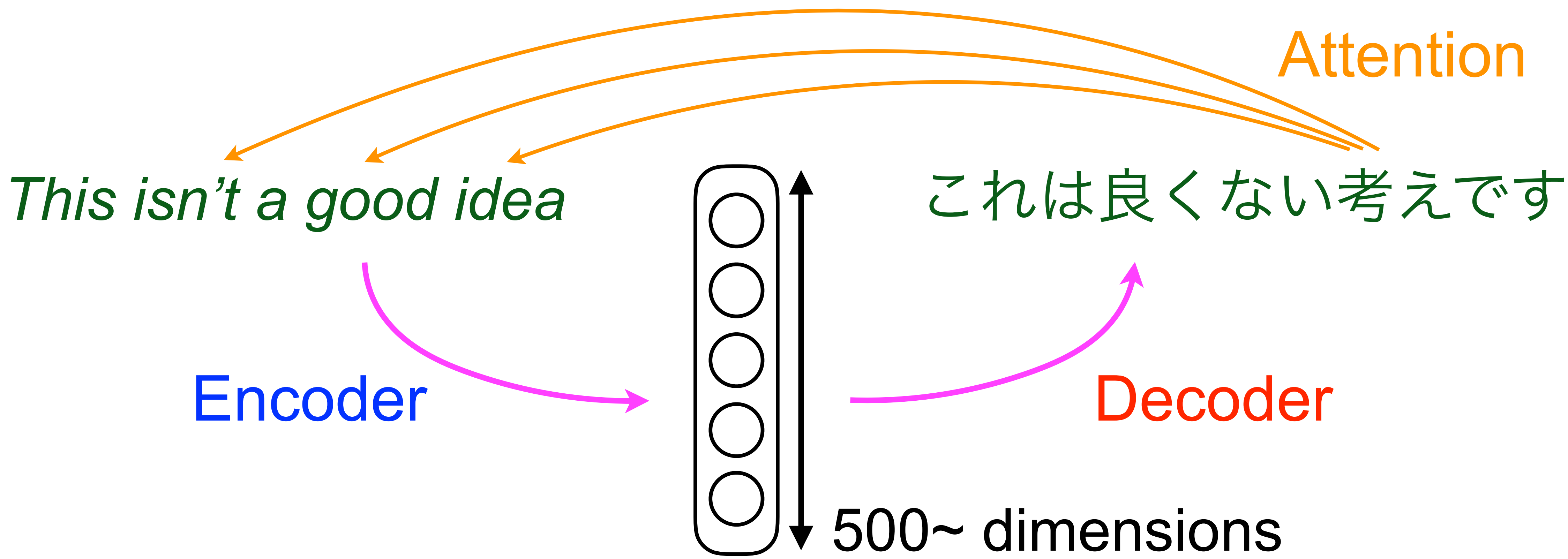
- One successful deep learning for NLP is machine translation
  - Google released **neural machine translation** in 2016, which greatly improves the quality of translation



Outline

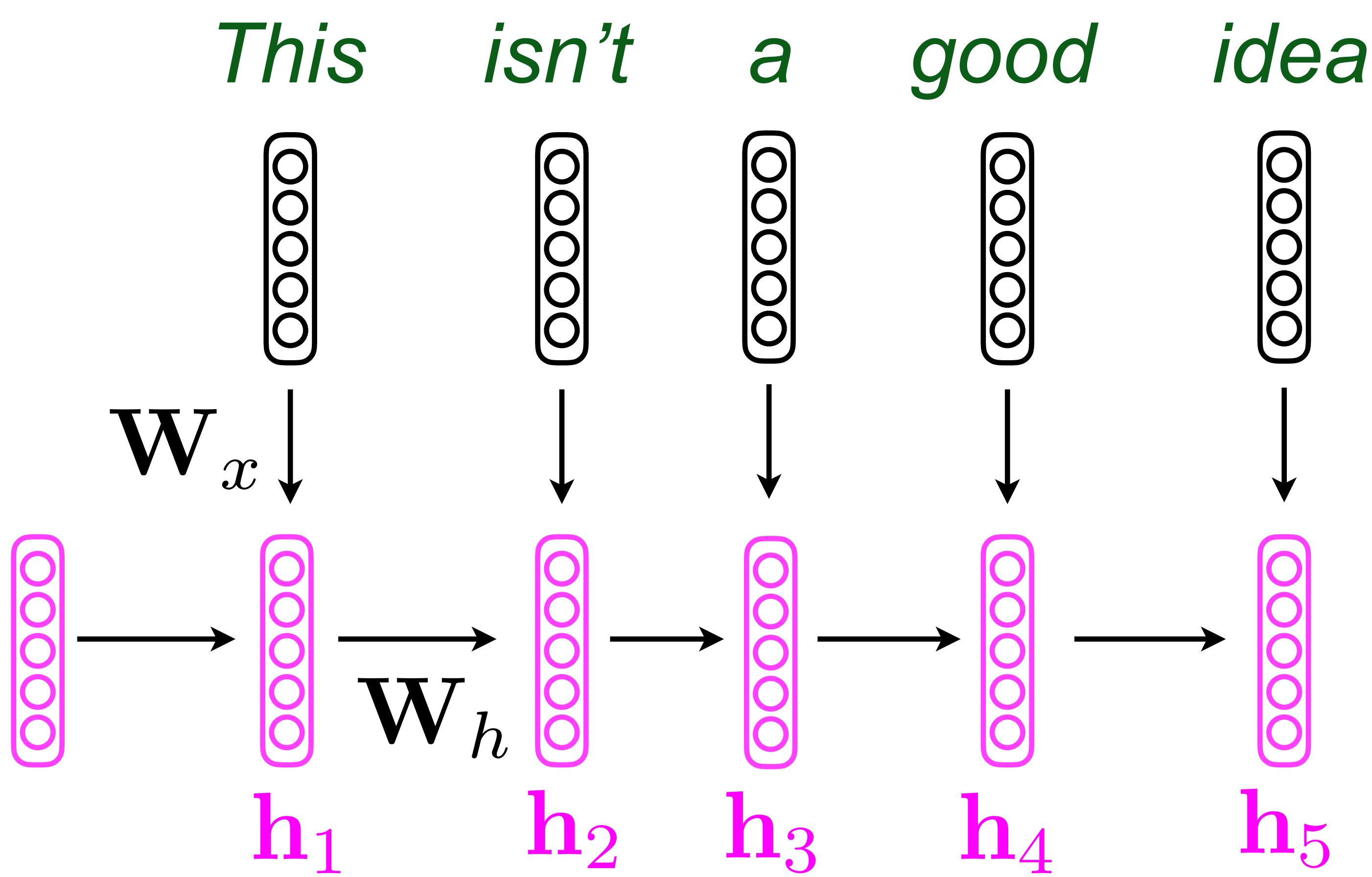
- Quick overview of recurrent neural networks
- Initial approach for neural semantic parsing
  - Naive application of RNN (encoder-decoder) to generate logical forms, and its problems
- Recent advancements to alleviate the problems
  - Data augmentation for generating pseudo examples
  - Approaches for guaranteeing executability, and well-formedness

What’s neural machine translation?



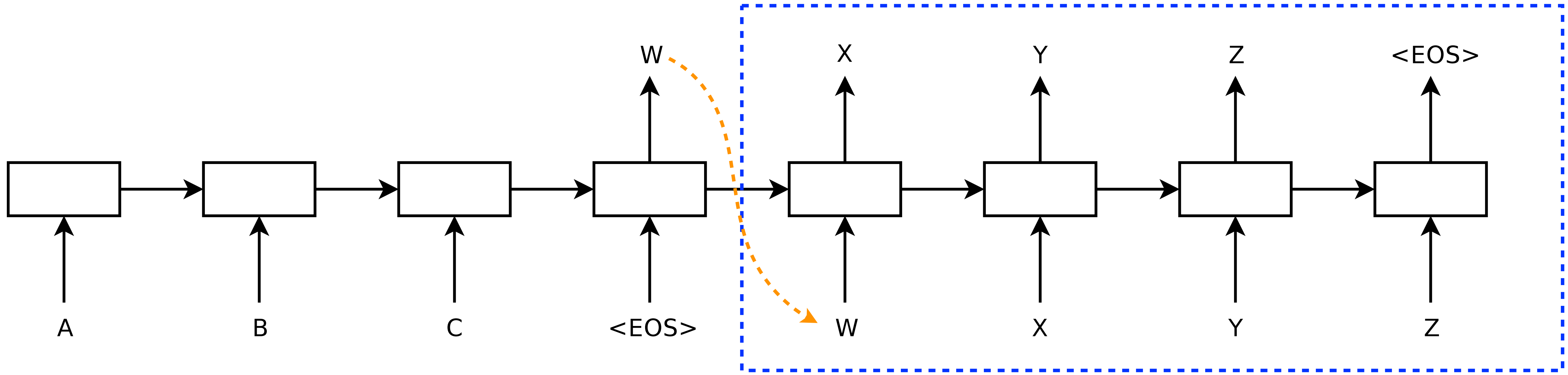
- Input sentence is first **encoded** into a fixed-length vector
- **Decoder** then outputs a sentence in the target language
  - **Attention** helps to decide which word to look at for generating next word
- **Recurrent neural networks** are used in encoder/decoder

Recurrent neural networks



- One of most important techniques in the current NLP
- **Hidden vectors** are updated by receiving every new word
  - Finally we may get a vector, which (we hope) encodes all information of the input sentence (including word order)

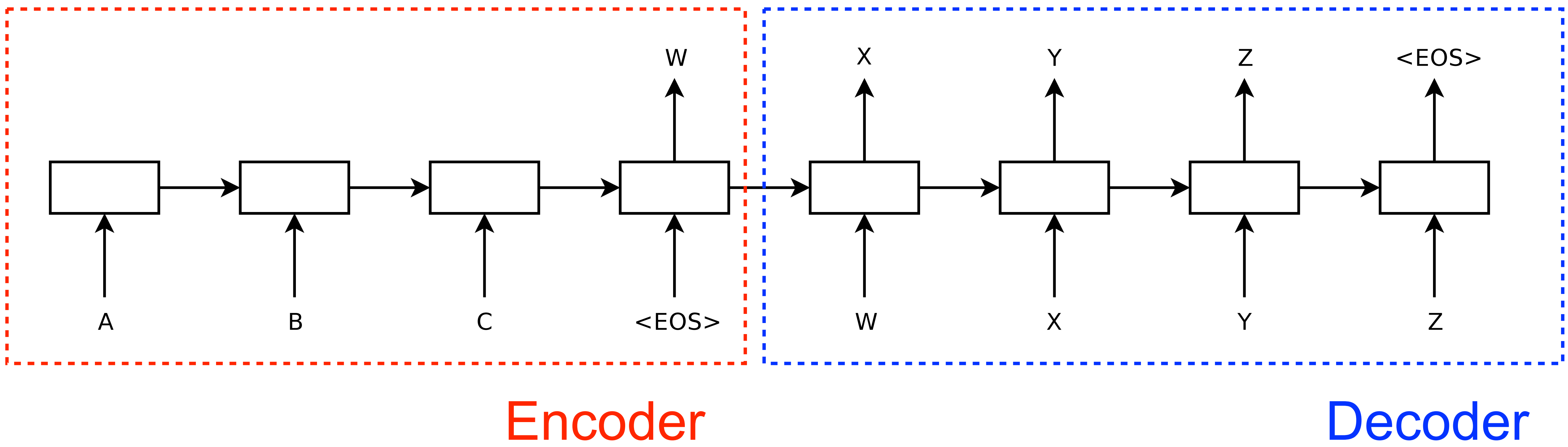
RNN as a generator



- RNN can also be a language generator, which outputs a word following the current context (outputs so far)
- (Theoretically) it remembers all words generated so far, and decide the next word based on them

Sutskever et al., Sequence to Sequence Learning with Neural Networks. In NIPS 2014.

Encoder-decoder



- Encoder-decoder is a general framework to combine two RNNs, one for encoding an input sentence, and another for decoding (generating new sentence)
- Neural machine translation is one application of encoder-decoder

Empirical success of RNN

- While it is quite simple, RNN and seq2seq had a great success empirically in many NLP applications
- Maybe this power comes from their strong ability of generation that generates a next word considering all previous history (and attention, which we don't discuss today)

Machine learning and NLP

- Sometimes (often?) new machine learning techniques affect the idea of NLPers, and the research trend of NLP
  - In early 2000s, support vector machine (SVM) was introduced
  - Many NLPers were excited to apply SVM to NLP problems
  - But in other words, they got more focusing on **the problems that can be solved by SVM** (tools in hand)
  - Examples are relatively superficial NLP problems
    - E.g., POS tagging, word segmentation, etc.

- NLP is not merely an application of machine learning; however, **many innovations in NLP are driven by new techniques in machine learning**

Now, neural networks and NLP

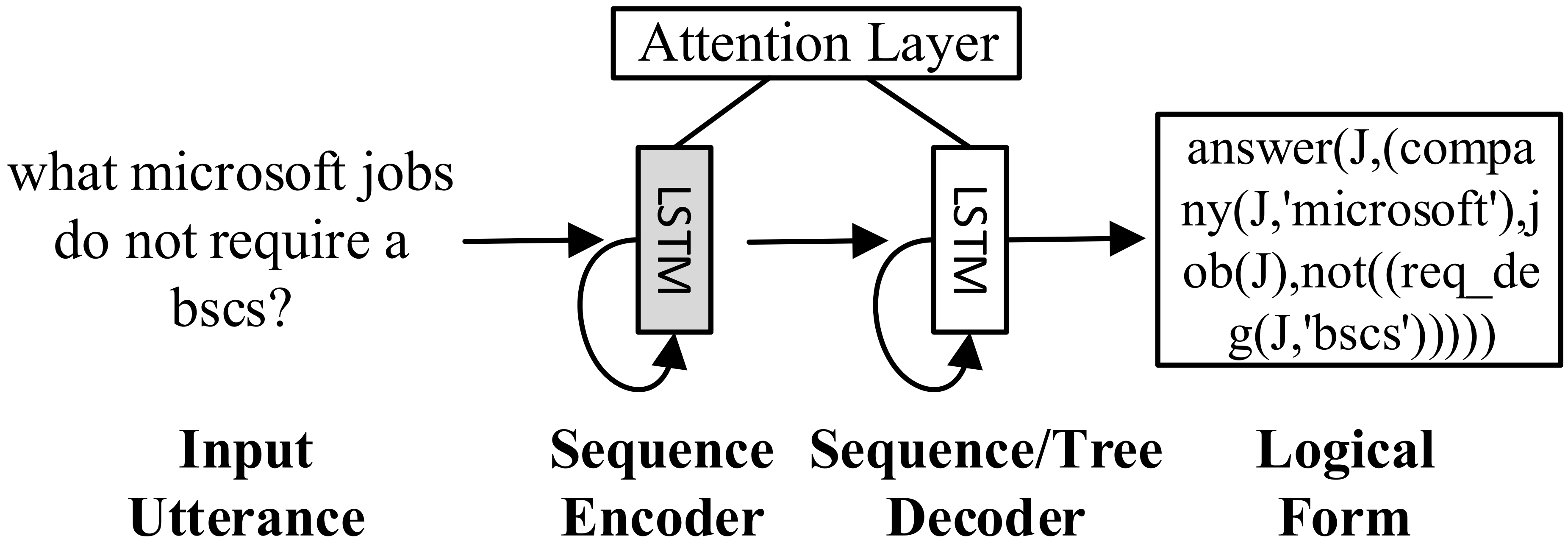
- Since 2014/2015, neural networks gain much popularity
  - Now NLPers are excited on this new technology, and are trying to apply it to any problems on which neural networks might be applicable
- Semantic parsing is not an exception in this trend
  - Semantic parsing with NNs was first appeared in 2016
  - In 2017, further ideas are proposed on this direction
- Today we will see the role of neural networks in these recent research efforts

Outline

- Quick overview of recurrent neural networks
- Initial approach for neural semantic parsing
  - Naive application of RNN (encoder-decoder) to generate logical forms, and its problems
- Recent advancements to alleviate the problems
  - Data augmentation for generating pseudo examples
  - Approaches for guaranteeing executability, and well-formedness

Initial approach

Dong and Lapata. Language to Logical Form with Neural Attention. In ACL 2016.



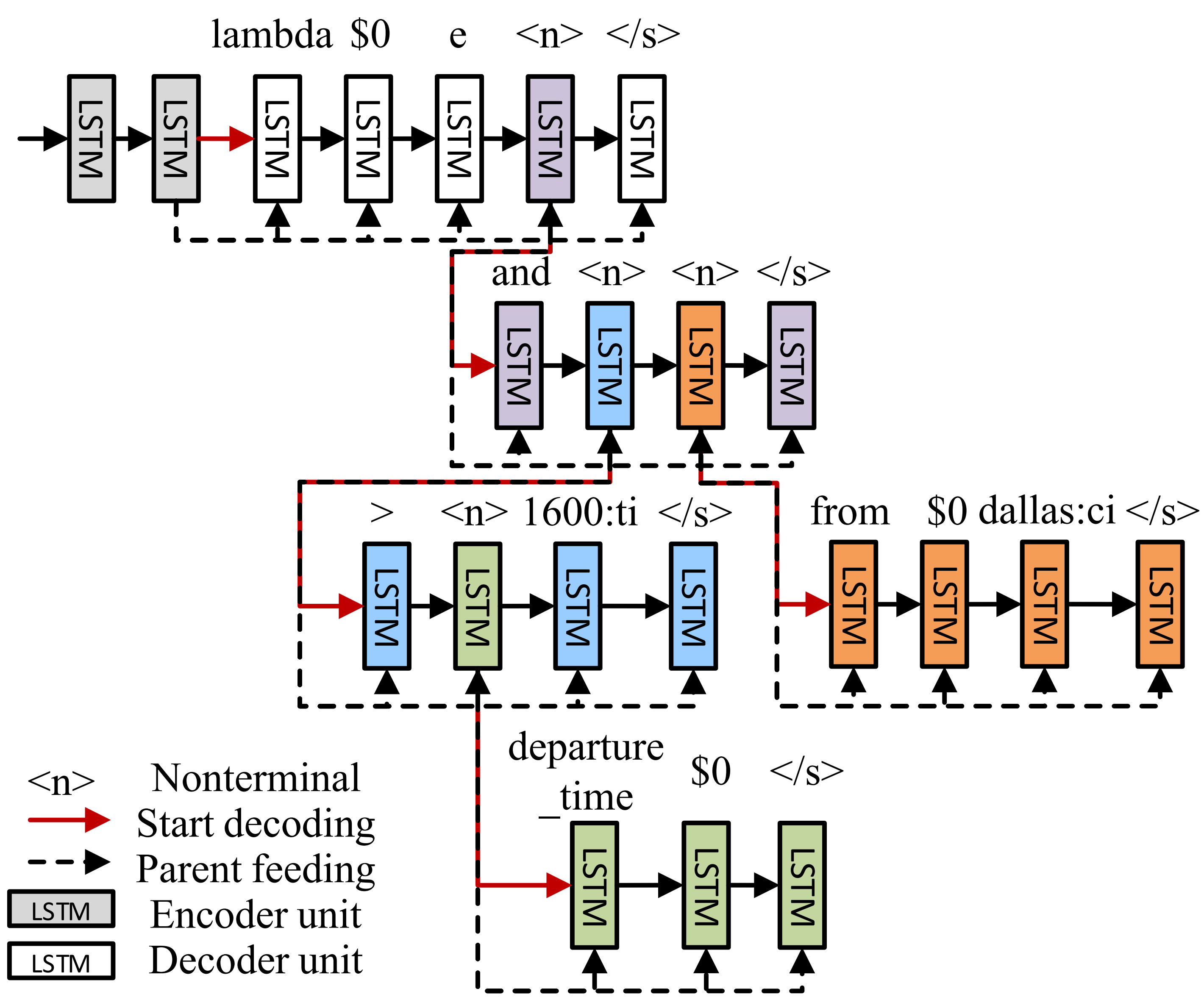
- Just casting semantic parsing as (neural) machine translation
  - Translating from an input utterance to a logical form
- Though the idea is quite simple, they report reasonable accuracies

Pros and cons

- Pros (that authors argue):
  - The system can work with any types of logical forms
  - Lambda expression, prolog, python, SQL… Any grammar is ok
- Cons:
  - It does not handle grammar of logical forms explicitly
  - So it may outputs ungrammatical logical forms (well-formedness)
  - Even an output is grammatical, that may not be executable (well-typedness, executability)
  - We need full supervision to logical forms (= expensive!)
    - Maybe we have degraded in some sense? Remember we can achieve weak-supervision by some techniques, e.g., DCS

Top-down generation

- To overcome well-formedess issue, the authors proposed top-down generation (pure LSTM is sequential)
- However, top-down generation itself cannot force the outputs to be well-formed
- Since the grammar is still not explicitly encoded in the model



In 2016, researchers have revisited **supervised semantic parsing**, essentially because this is the problem that we can solve using new technology (RNN)

Outline

- Quick overview of recurrent neural networks
- Initial approach for neural semantic parsing
  - Naive application of RNN (encoder-decoder) to generate logical forms, and its problems
- Recent advancements to alleviate the problems
  - Data augmentation for generating pseudo examples
  - Approaches for guaranteeing executability, and well-formedness

Data augmentation

Jia and Liang. Data Recombination for Neural Semantic Parsing. In ACL 2016.

- Idea: from (sentence, logical form) training data, we increase its size by creating “pseudo” training data
- Motivation: RNNs become capable of generating perfect (grammatical) logical forms if we have abundant training data
- A popular approach in image recognition
  - For images, transforming inputs keeping semantics is not hard, e.g., rotation, enlargement, etc.
  - For language, such transformation seems non-trivial



Inducing synchronous CFG

Examples
("*what states border texas ?*",
`answer(NV, (state(V0), next_to(V0, NV), const(V0, stateid(texas)))))`
("*what is the highest mountain in ohio ?*",
`answer(NV, highest(V0, (mountain(V0), loc(V0, NV), const(V0, stateid(ohio))))))`

Rules created by ABSENTITIES
ROOT → ⟨ "*what states border* STATEID *?*",
`answer(NV, (state(V0), next_to(V0, NV), const(V0, stateid(`STATEID`)))))`⟩
STATEID → ⟨ "*texas*", `texas` ⟩
ROOT → ⟨ "*what is the highest mountain in* STATEID *?*",
`answer(NV, highest(V0, (mountain(V0), loc(V0, NV),`
`const(V0, stateid(`STATEID`))))))`⟩
STATEID → ⟨"*ohio*", `ohio`⟩

- We infer the substring of input sentence, and corresponding part in the logical form by some rules
- The process that the authors call "data recombination"

Overview

Original Examples

what are the major cities in utah ?
what states border maine ?

↓ Induce Grammar

Synchronous CFG

↓ Sample New Examples

Recombinant Examples

what are the major cities in [states border [maine]] ?
what are the major cities in [states border [utah]] ?
what states border [states border [maine]] ?
what states border [states border [utah]] ?

↓ Train Model

Sequence-to-sequence RNN

Outline

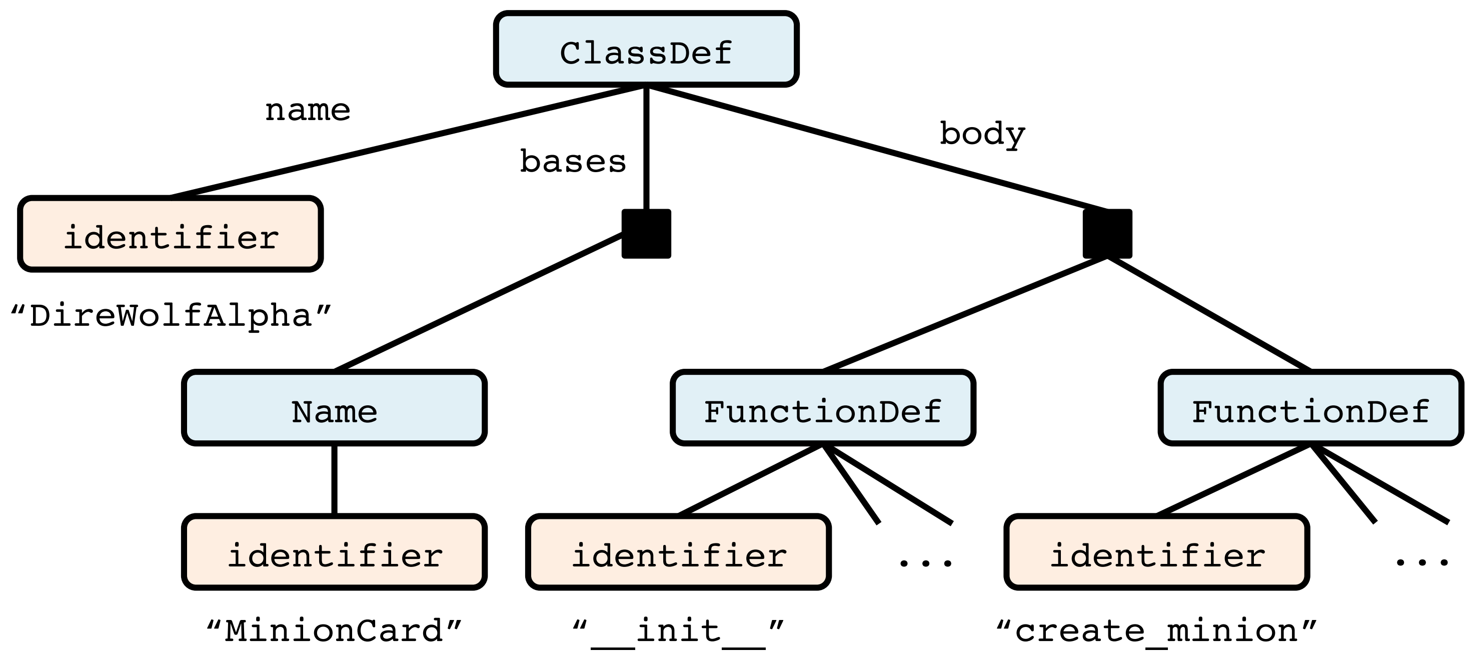
- Quick overview of recurrent neural networks
- Initial approach for neural semantic parsing
  - Naive application of RNN (encoder-decoder) to generate logical forms, and its problems
- Recent advancements to alleviate the problems
  - Data augmentation for generating pseudo examples
  - Approaches for guaranteeing executability, and well-formedness

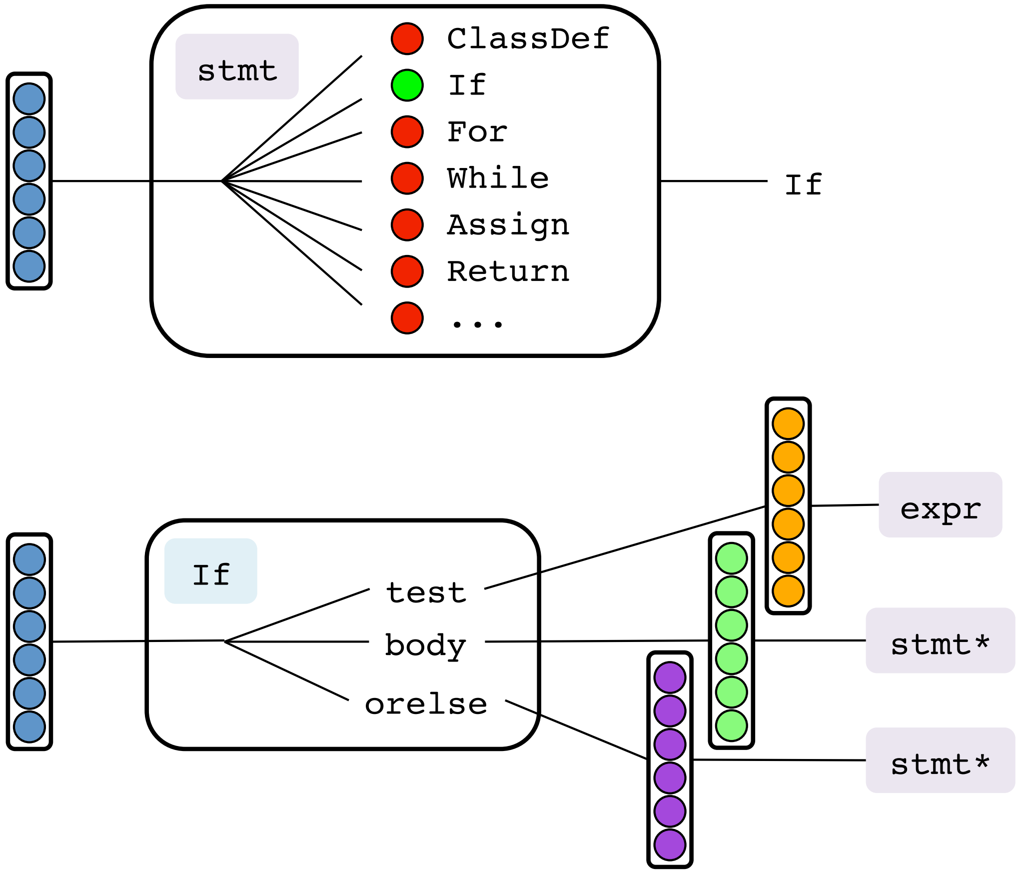
More recent approaches

- Data augmentation is an attempt to overcome “well-formedness” issue, but it is still incomplete
  - The model is unchanged from basic RNNs
- More recent approaches are exploring an improvement to decoder (model) itself, for guaranteeing well-formedness of the outputs

Abstract syntax networks

Rabinovich et al. Abstract Syntax Networks for Code Generation and Semantic Parsing. In ACL 2017.





- A recurrent neural network that generates an abstract syntax tree (AST; left)
- Left AST is for Python, and can be converted to Python code

Idea

- Abstract syntax network can only generate well-formed ASTs
- Can be constrained to output specific languages (e.g., Python, logical forms), by specifying the grammar

Making RNN cleverer by adding knowledge of the target language

what microsoft jobs do not require a bscs?
answer(company(J,'microsoft'),job(J),not((req_deg(J,'bscs'))))

```
expr
  = Apply(pred predicate, arg* arguments)
  | Not(expr argument)
  | Or(expr left, expr right)
  | And(expr* arguments)

arg
  = Literal(lit literal)
  | Variable(var variable)
```

Figure 9: The Prolog-style grammar we use for the JOBS task.

Weak supervision

A Krishnamurthy et al. Neural Semantic Parsing with Type Constraints for Semi-Structured Tables. In EMNLP 2017.

Maximize $$\mathcal{O}(\theta) = \sum_{i=1}^{n} \log \sum_{\ell \in \mathcal{L}^i} P(\ell | q^i, T^i; \theta)$$

Pseudo correct logical forms

- Similar approach to constraint the outputs to be well-formed
- They additionally show that weak-supervision, as in training of DCS, is possible for neural semantic parsing
- Perform back-propagation with an accumulate loss, assuming a logical form that answers correctly is correct

Other important papers

- Guu et al., ACL 2017.
  - From Language to Programs: Bridging Reinforcement Learning and Maximum Marginal Likelihood
  - Traditional objective in weak-supervision is max-marginal likelihood; they show this approach is more stable than reinforcement learning
- Liang et al., ACL 2017.
  - Neural Symbolic Machines: Learning Semantic Parsers on Freebase with Weak Supervision
  - Propose a hack for reinforcement learning to work well on neural semantic parsing with weak-supervision

Some thoughts

- On semantic parsing, (I think) deep learning has not provide significant contribution yet
  - Compared to, e.g., the significance in machine translation
  - Semantic parsing has to be robust while pure vector-space models are brittle
  - Currently, we need logical forms for satisfying the robustness
  - Current systems are just using RNNs, expecting its strong power as a generator (of logical forms)
  - But it is unclear whether RNNs have a clearer advantage over conventional methods, e.g., bottom-up construction of CCG or DCS

Some thoughts (cont.)

- True integration of deep-learning and semantic parsing has not been established
  - Such systems may directly operate on a database (?)
  - But it is unknown all linguistic operations can be well defined on a vector space
- It is inevitable that deep learning plays a significant role in semantic parsing, and now the community is exploring the promising approach