Top Down Intro to Neural Semantic Parsing

Ofir

Plan

1. Intro to NNs and RNNs — Short!
2. Model 1: seq2seq — Meaty!
3. Model 2: seq2tree
4. How to improve these models with attention — Important!
5. Results & future work

NNs for supervised learning

Examples and labels: $\{(x^{(n)}, y^{(n)})\}$

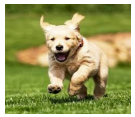
Neural network: $f(x^{(n)}; W)$

Loss on an example: $L(f(x^{(n)}; W), y^{(n)})$

Objective: Find W that minimizes $\Sigma\, L(f(x^{(n)}, W), y^{(n)})$

Generalization





1	$f(x^{(1)}; W) = 0.9$	L =0.1
0	$f(x^{(2)}; W) = 0.02$	L =0.02
1	$f(x^{(3)}; W) = 0.3$	L =0.7

Language modeling

U.S. intelligence agencies have …

High Probability:

- a
- received

Low Probability:

- Antarctica
- Dog
- Cat

Language models continuously make predictions

U.S. intelligence agencies have not…

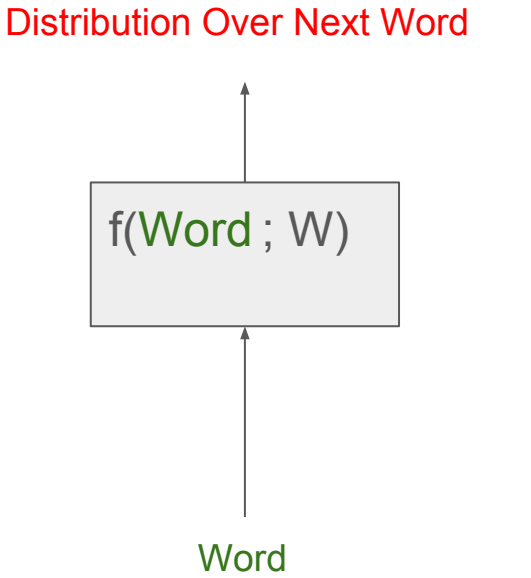
High Probability:

- received
- announced
- discovered

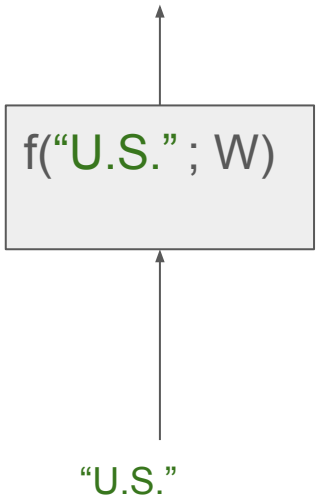
Low Probability:

- a
- the
- Dog

Simple model



Probability that next word is…
president = 0.07
government = 0.05
the = 0.0003
…..



Simple model: Data

U.S. intelligence agencies have not corroborated the allegations about the president-elect's personal life and finances……

{(U.S., intelligence),

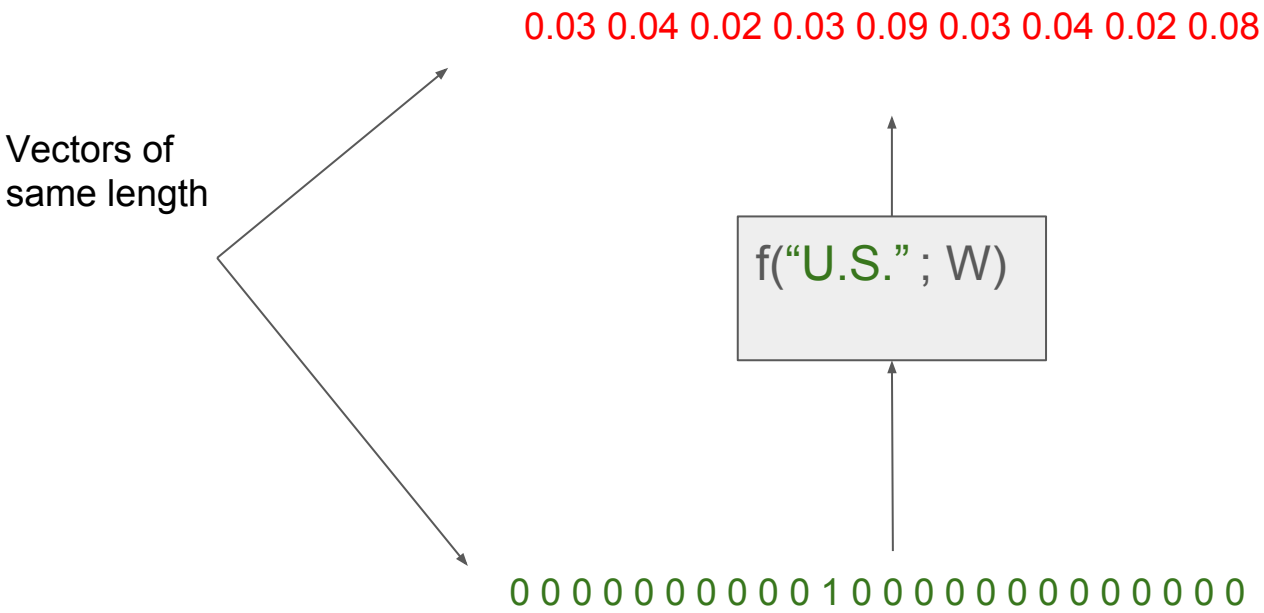
(intelligence, agencies),

(agencies, have),

…}

Small implementation detail:

cat (1,0,0)
hello (0,1,0)
the (0,0,1)



Each word has an ID between 1 and | size of vocab | .

Loss at every timestep of decoder:

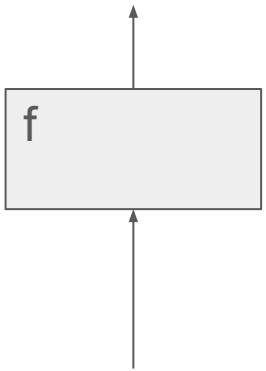
-ln(P(correct_word))

P(correct_word) = 1 → Loss = -ln(1) = 0

P(correct_word) = 0 → Loss = -ln(0) = ∞

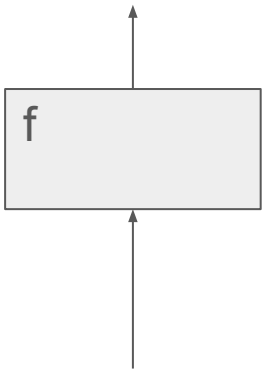
Problem: No memory

Intelligence = 0.05
Government = 0.3
President = 0.07
…..



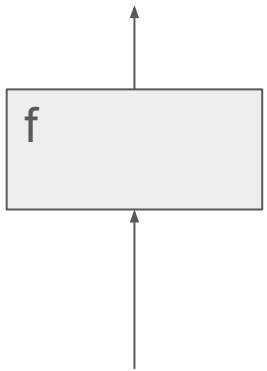
“U.S.”

Intelligence = 0.09
Government = 0.02
President = 0.04
…..



“intelligence”

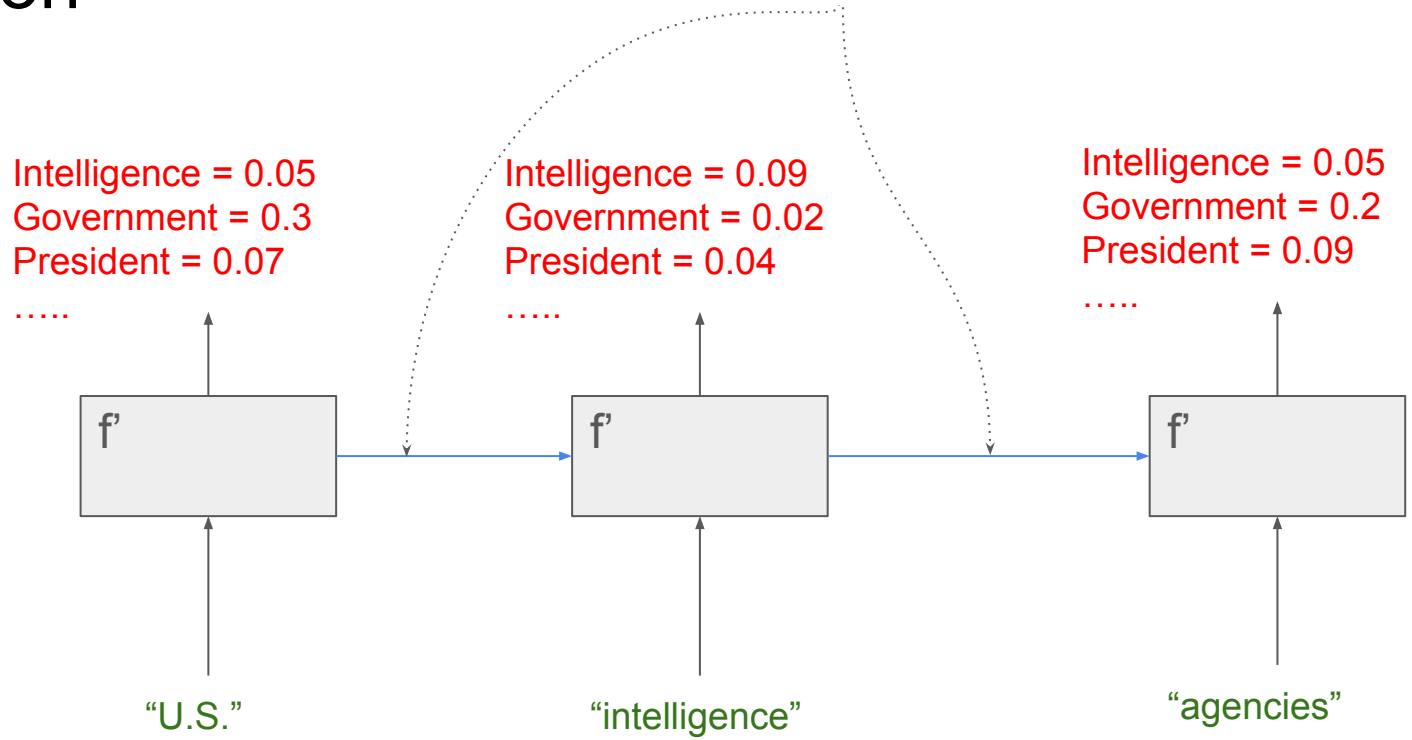
Intelligence = 0.05
Government = 0.2
President = 0.09
…..



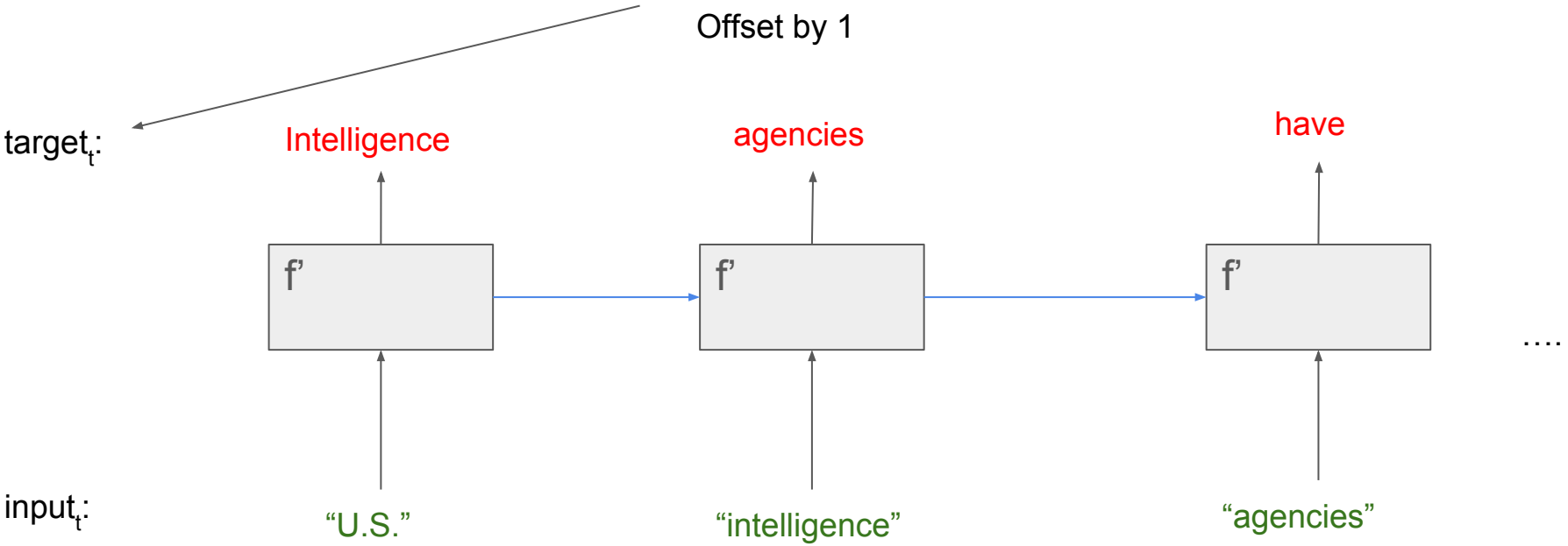
“agencies”

Solution

State: represents what we've seen until now.

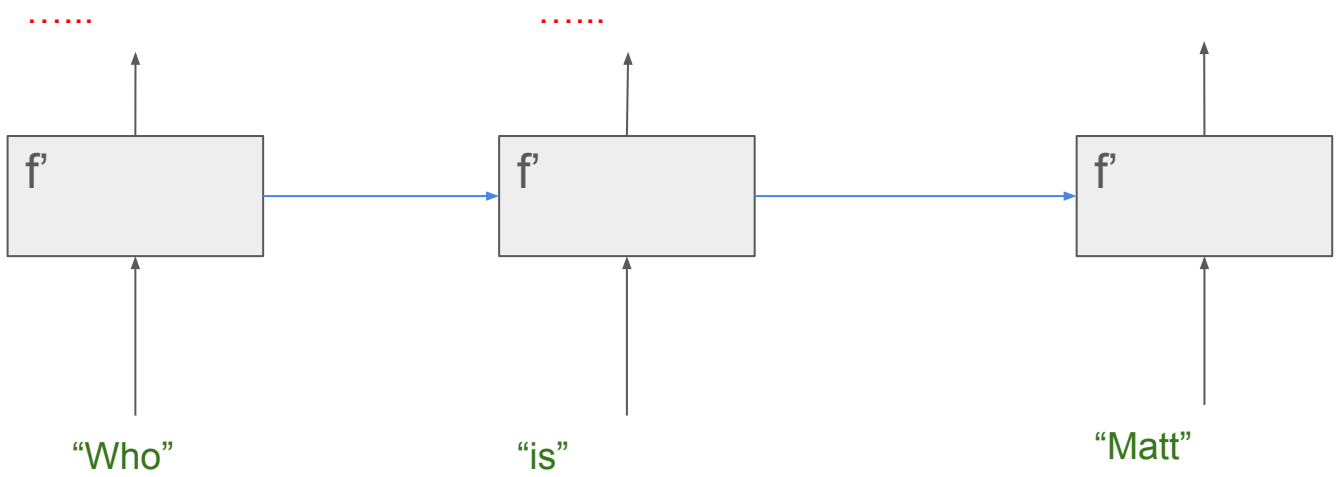


At train time:

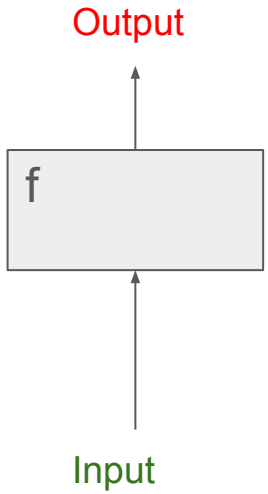


At test time: “Who is Matt _ ?”

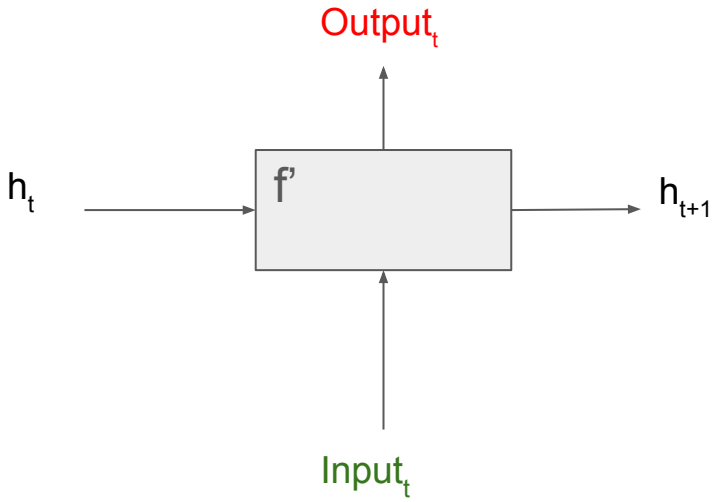
Damon 0.05
Lauer 0.041
Terry 0.040
……



Comparison:



1 input, 1 output



2 inputs, 2 outputs

Intro done. Questions?

Semantic parsing as a seq2seq problem:

what microsoft jobs do not require a bscs? →

answer(J,(company(J,'microsoft'),job(J),not((req_deg(J,'bscs')))))

No knowledge base involved in this problem.

Another example

dallas to san francisco leaving after 4 in the afternoon please →

(lambda $0 e (and ((departure time $0) 1600:ti) (from $0 dallas:ci) (to $0 san francisco:ci)))

input/output is a series of tokens:

[“what”, “microsoft”, “jobs”, “do”, “not”, “require”, “a”, “bscs”, “?”, “</s>”] →

[“<s>”,“answer”, “(“, “J” “,” “(“, “company”, ….., “</s>”]

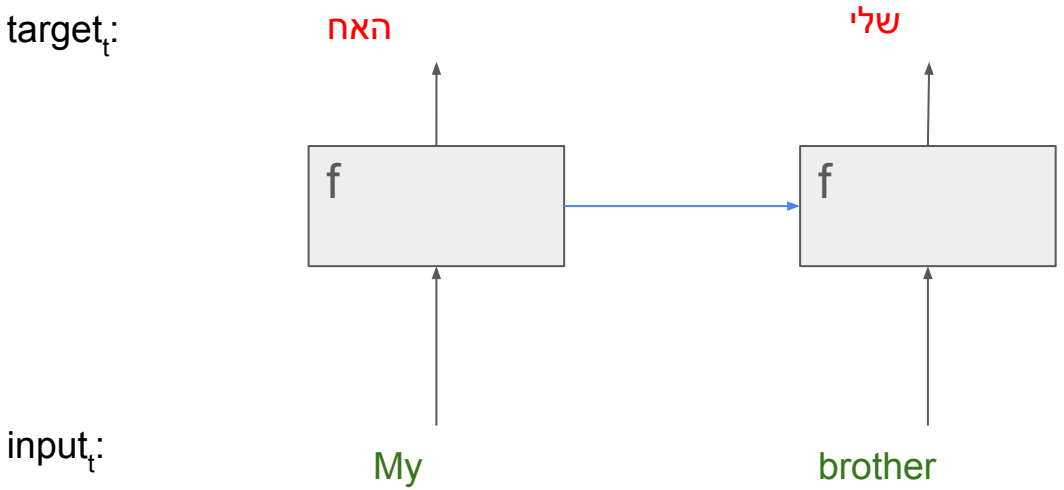
EOS token

Lets look at a simple example:

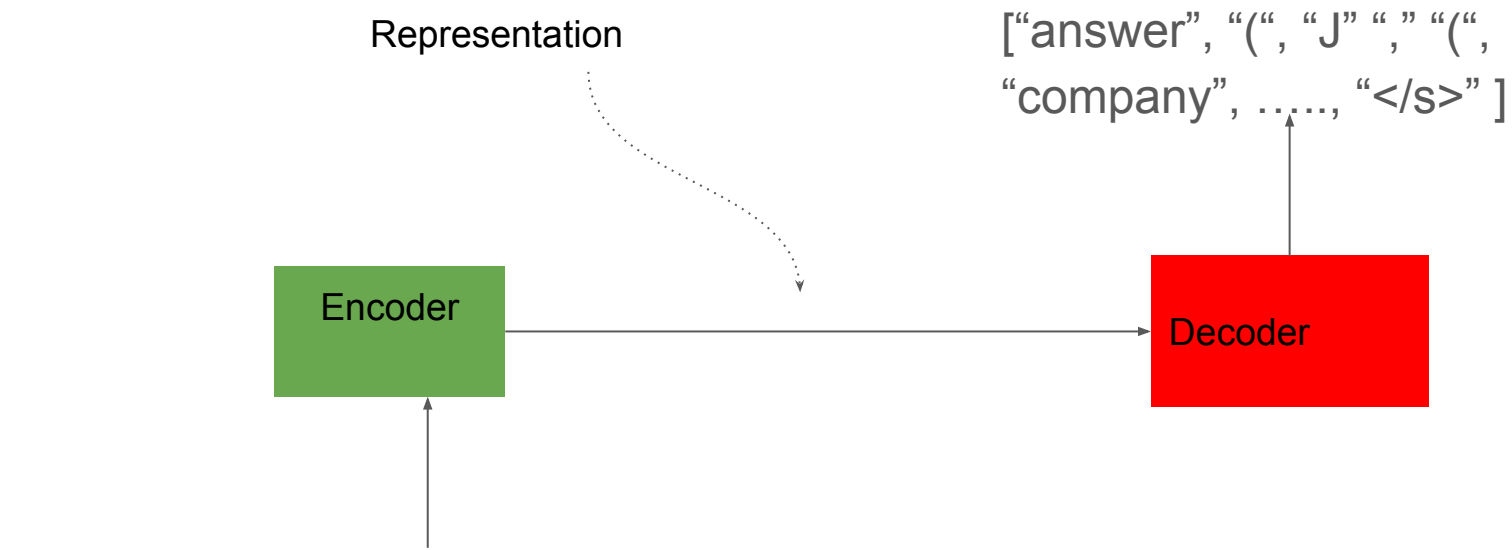
My brother →

האח שלי

Solution?

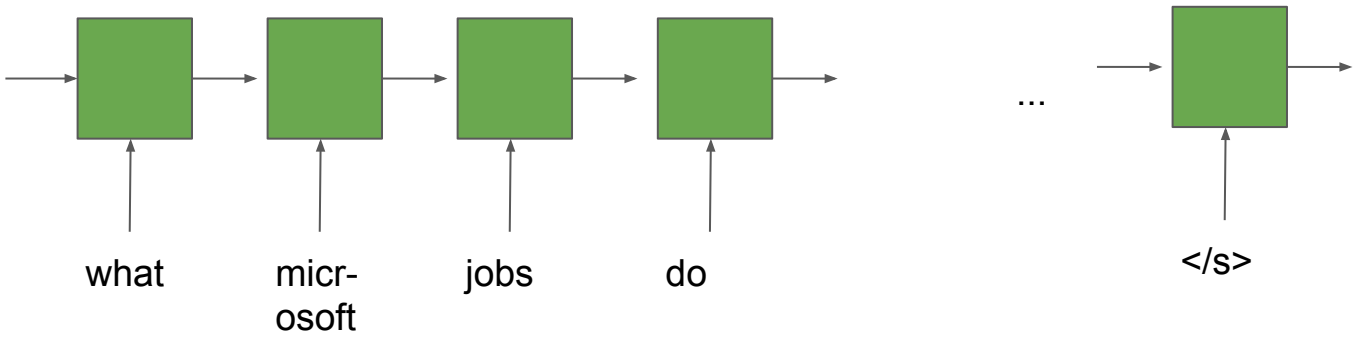


What we want

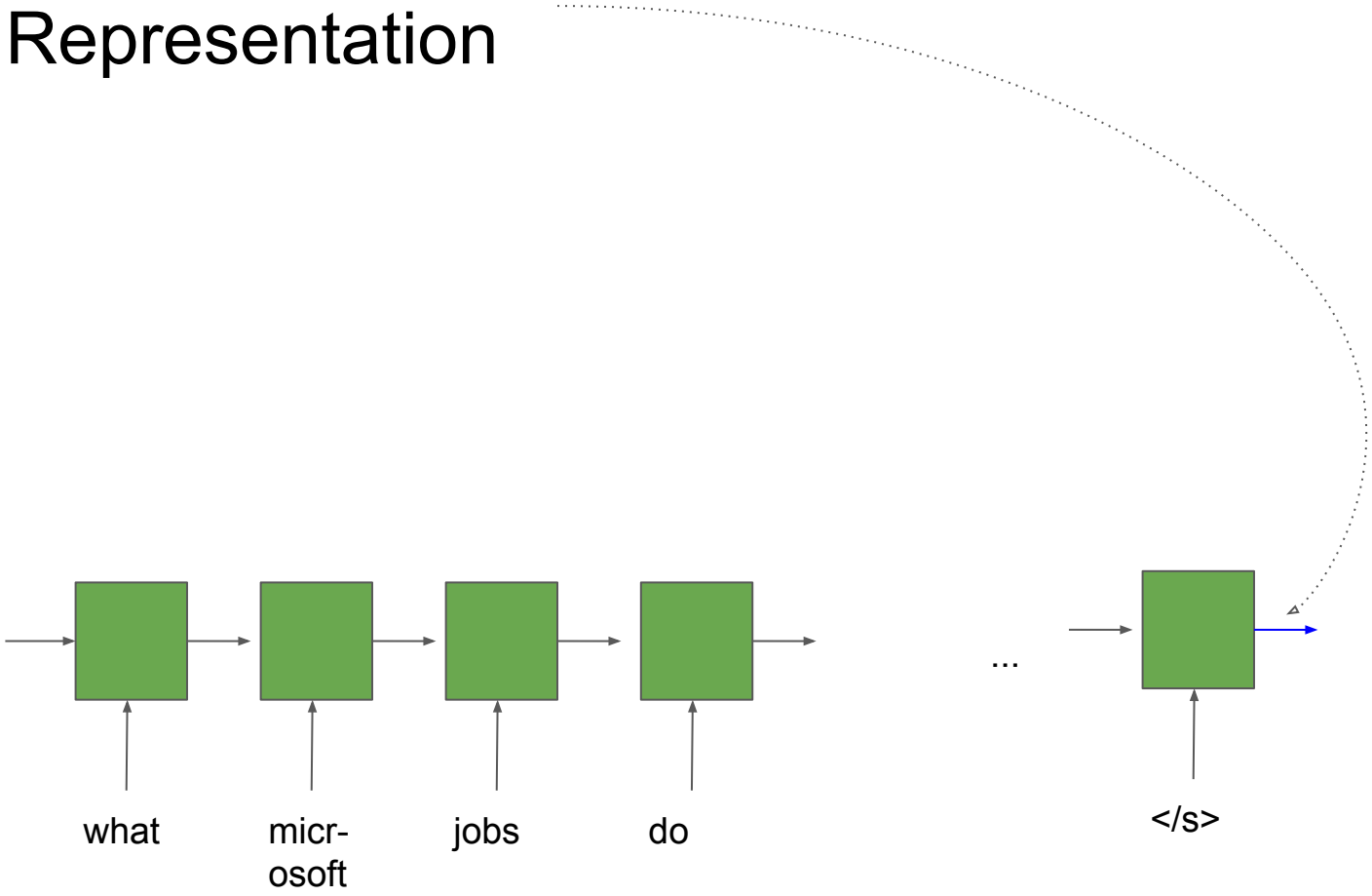

Representation
["answer", "(", "J" "," "(",
"company", ….., "</s>"]
Encoder
Decoder
["what", "microsoft", "jobs", "do", "not",
"require", "a", "bscs", "?", "</s>"]

An example run of this model (trained):

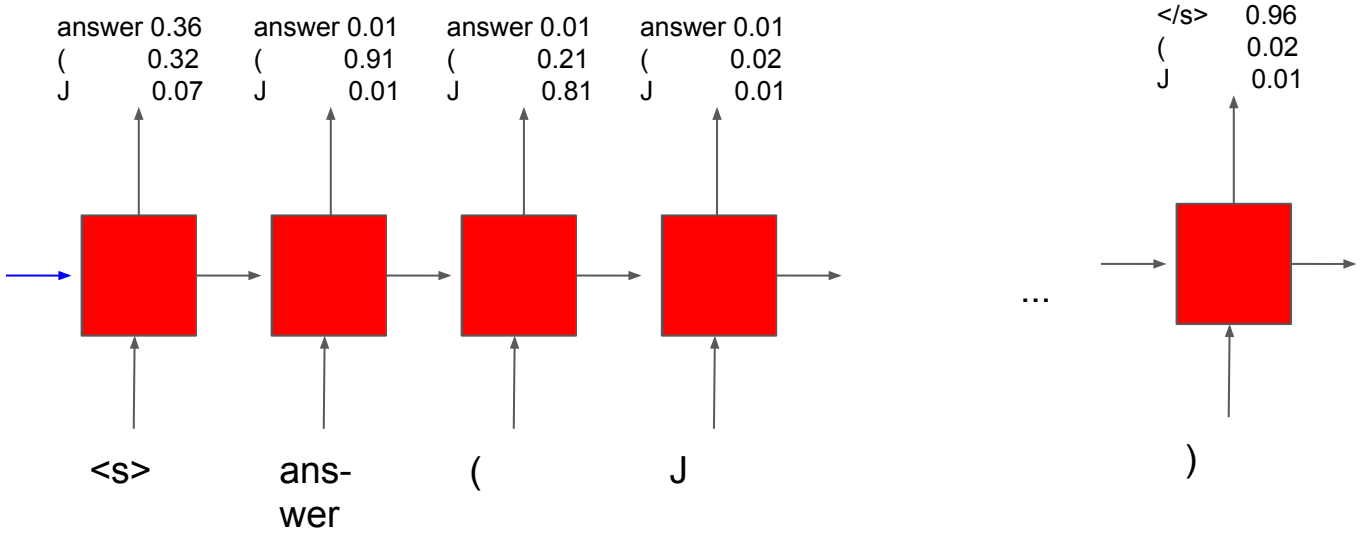
Encoder



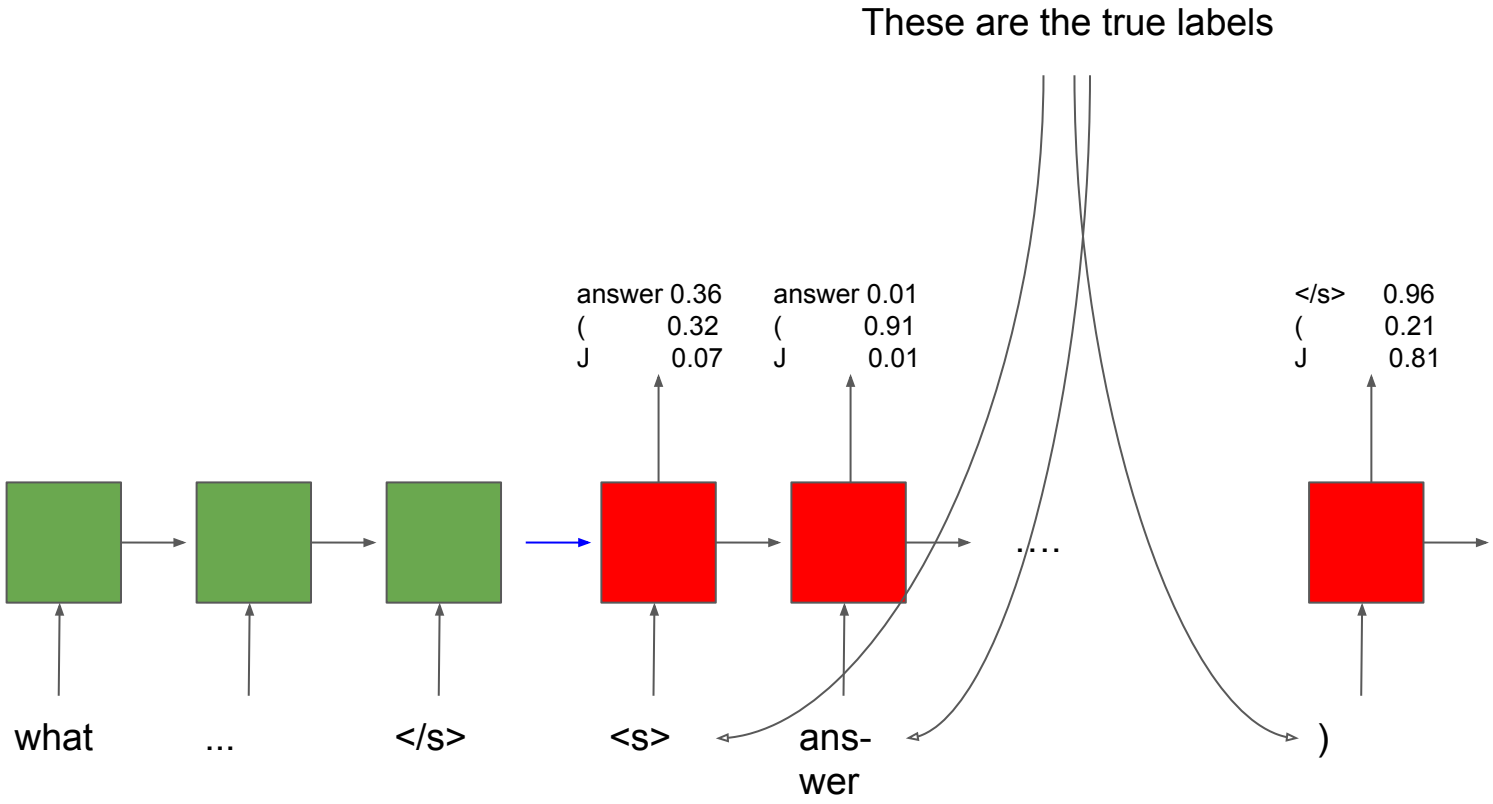
Representation



Decoder


answer 0.36
(0.32
J 0.07
answer 0.01
(0.91
J 0.01
answer 0.01
(0.21
J 0.81
answer 0.01
(0.02
J 0.01
</s> 0.96
(0.02
J 0.01
...
<s>
ans-
wer
(
J
)

How is this trained?

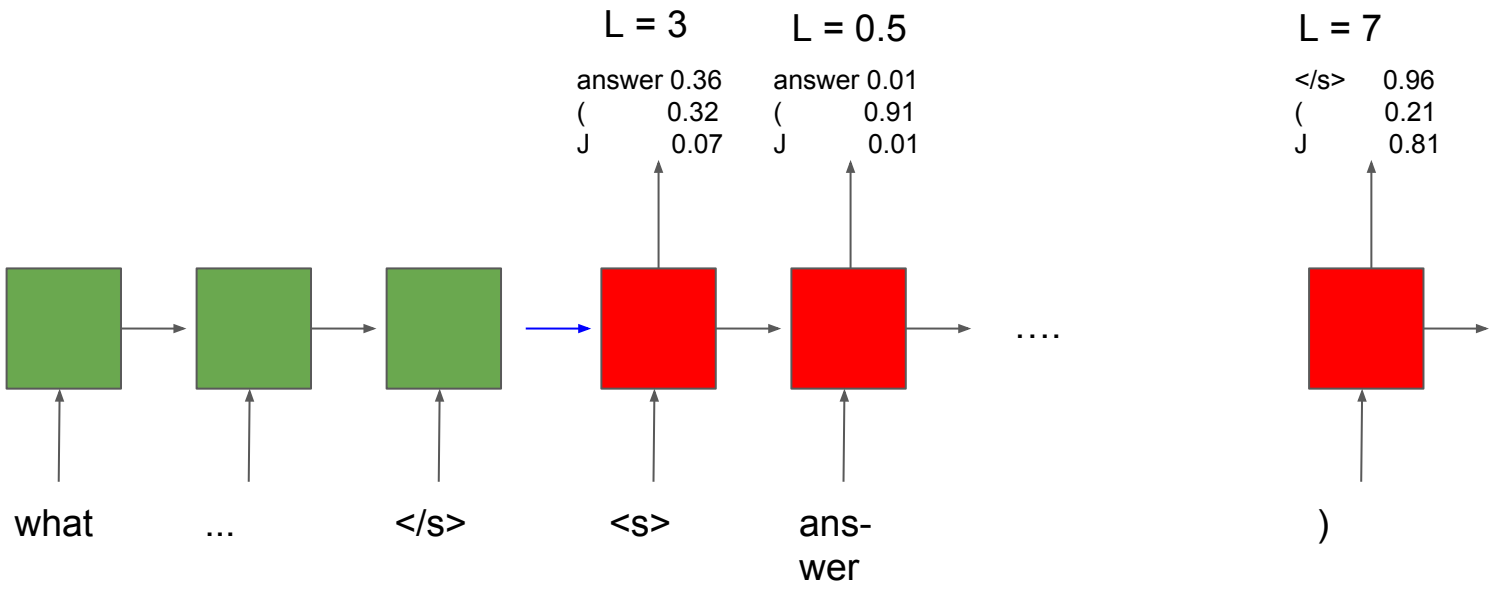


Loss at every timestep of decoder:

-ln(P(correct_word))

P(correct_word) = 1 → Loss = -ln(1) = 0

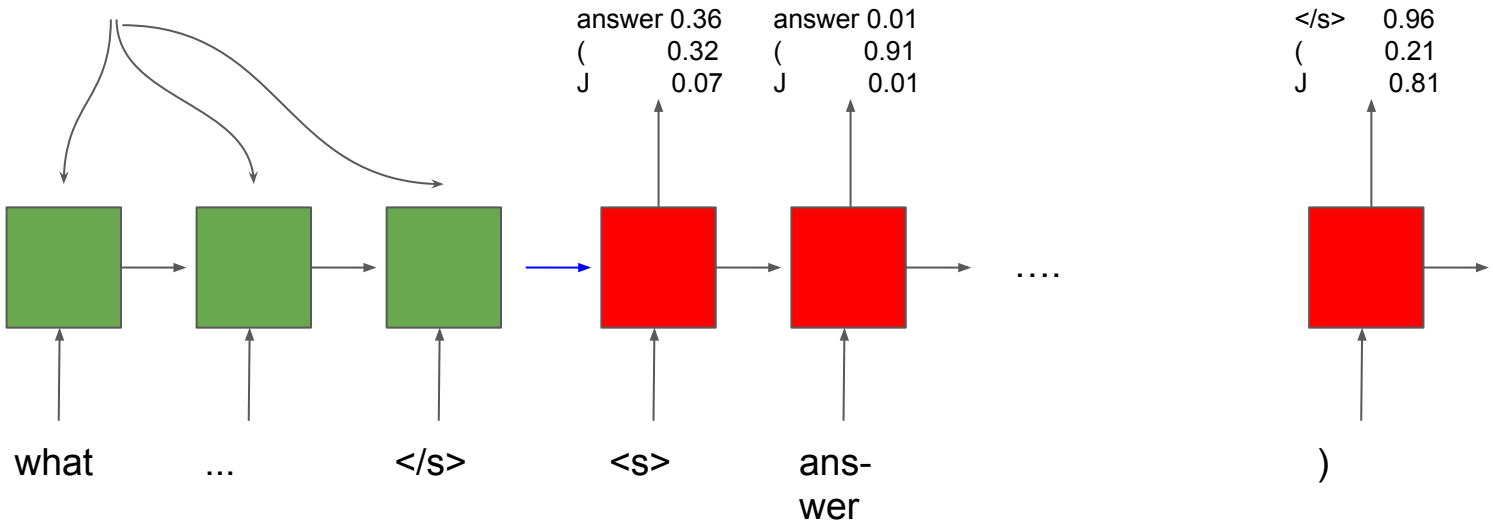
P(correct_word) = 0 → Loss = -ln(0) = ∞


L = 3
answer 0.36
(0.32
J 0.07
L = 0.5
answer 0.01
(0.91
J 0.01
L = 7
</s> 0.96
(0.21
J 0.81
....
what
...
</s>
<s>
ans-
wer
)

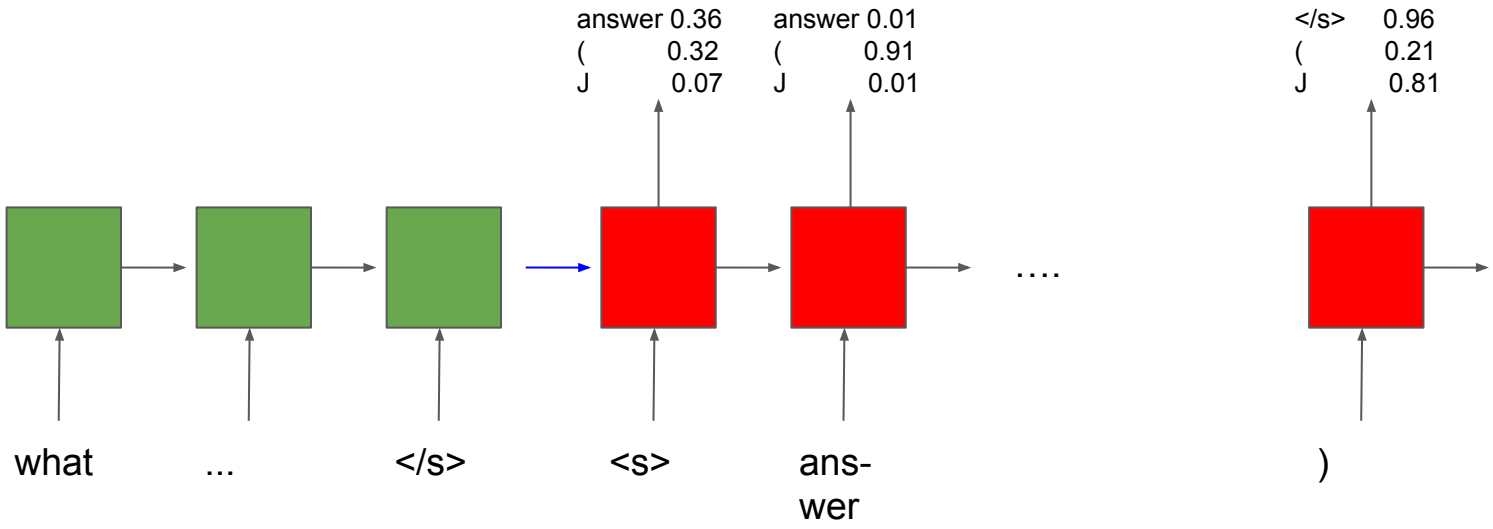
Small note about the network:

Also for decoder.

Same function

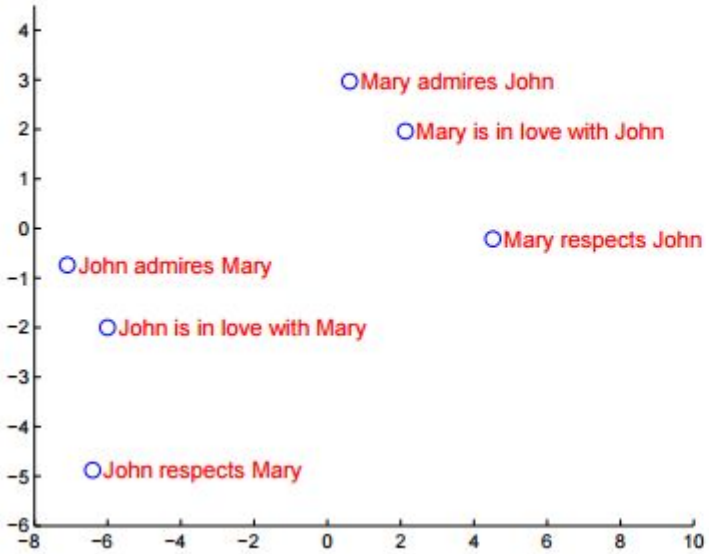


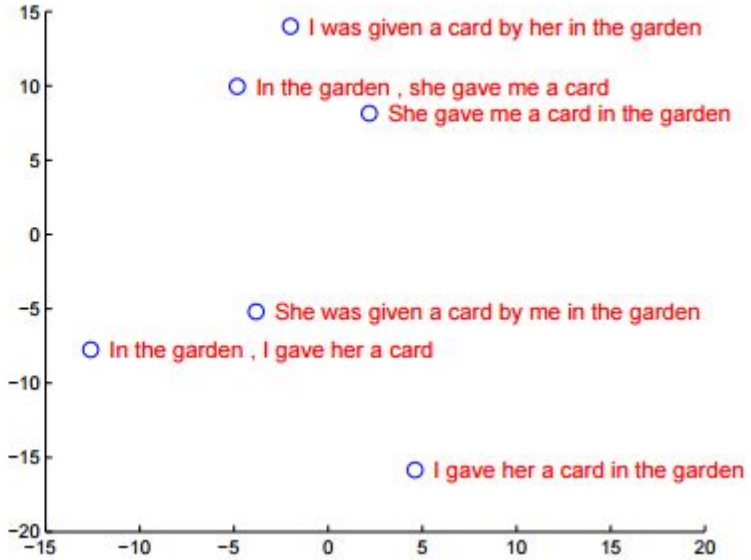
Top prob isn't guaranteed to be optimal, so use beam search



End of seq2seq model

Questions?





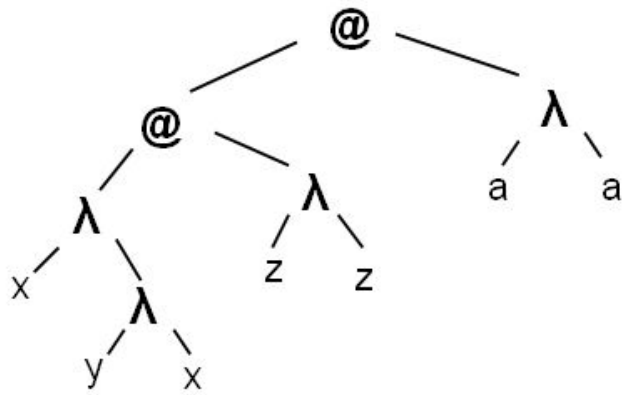
We modeled both sentences & logical forms as sequences.

Logical forms as trees

But logical forms represent trees:

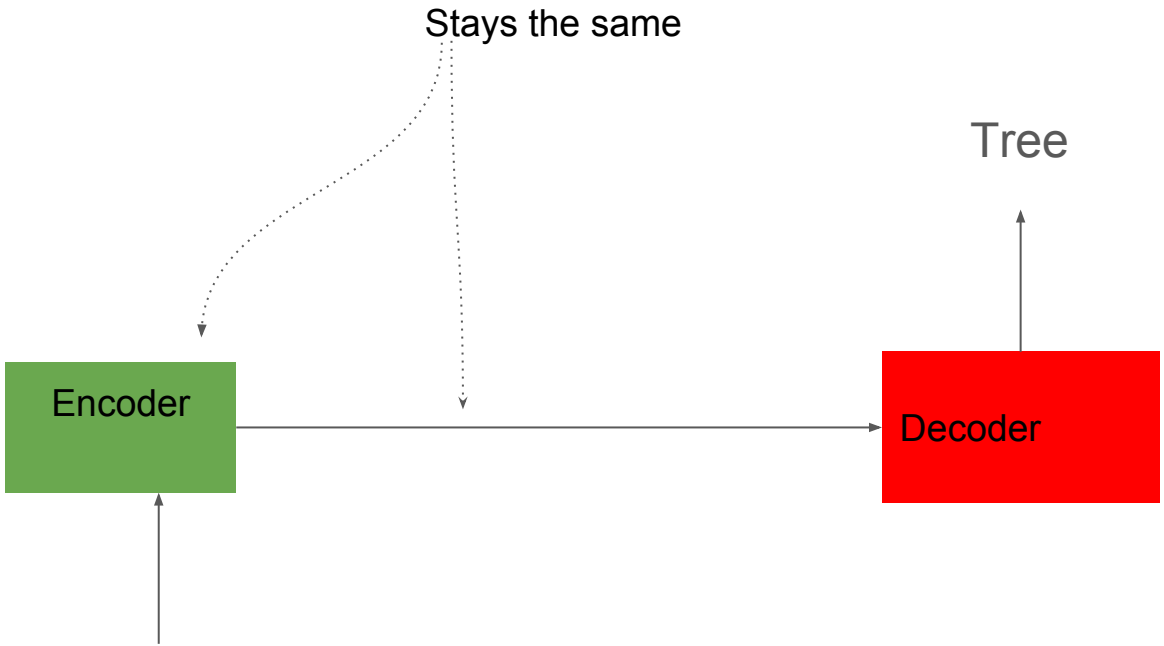
Lambda expression:

((λ x . (λ y . x)) (λ z . z)) (λ a . a)



So lets build a decoder that instead of outputting a sequence would output a tree.

What we want



[“what”, “microsoft”, “jobs”, “do”, “not”, “require”, “a”, “bscs”, “?”, “</s>”]

We will use a sequence model to generate a tree

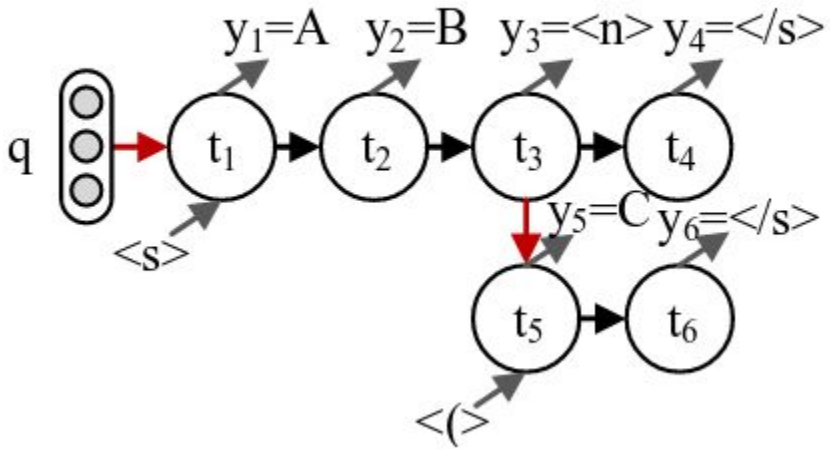
How?

Layer by layer, starting from the top.

Preprocessing

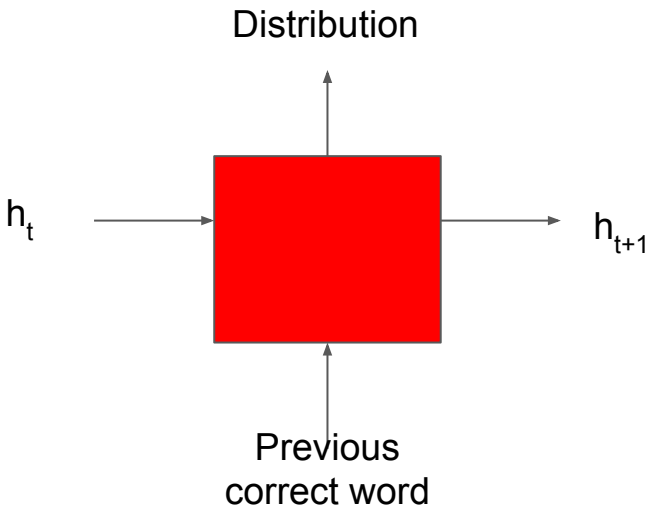
Take output sequence, replace all “(…)” with “<n>”.

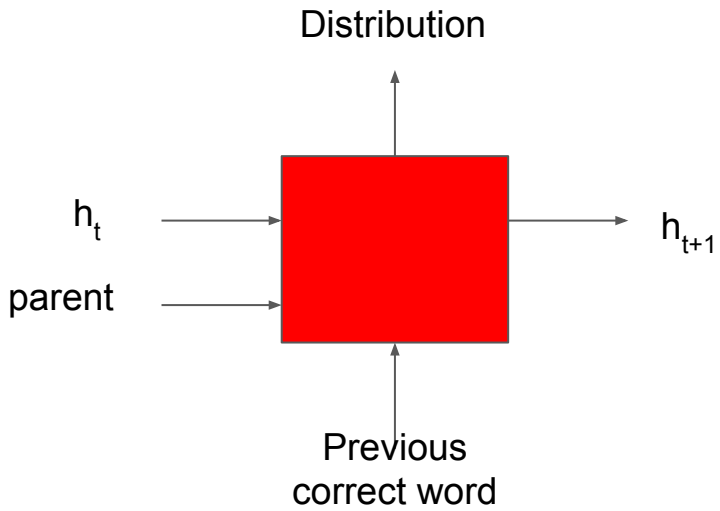
Decoding: A B (C)

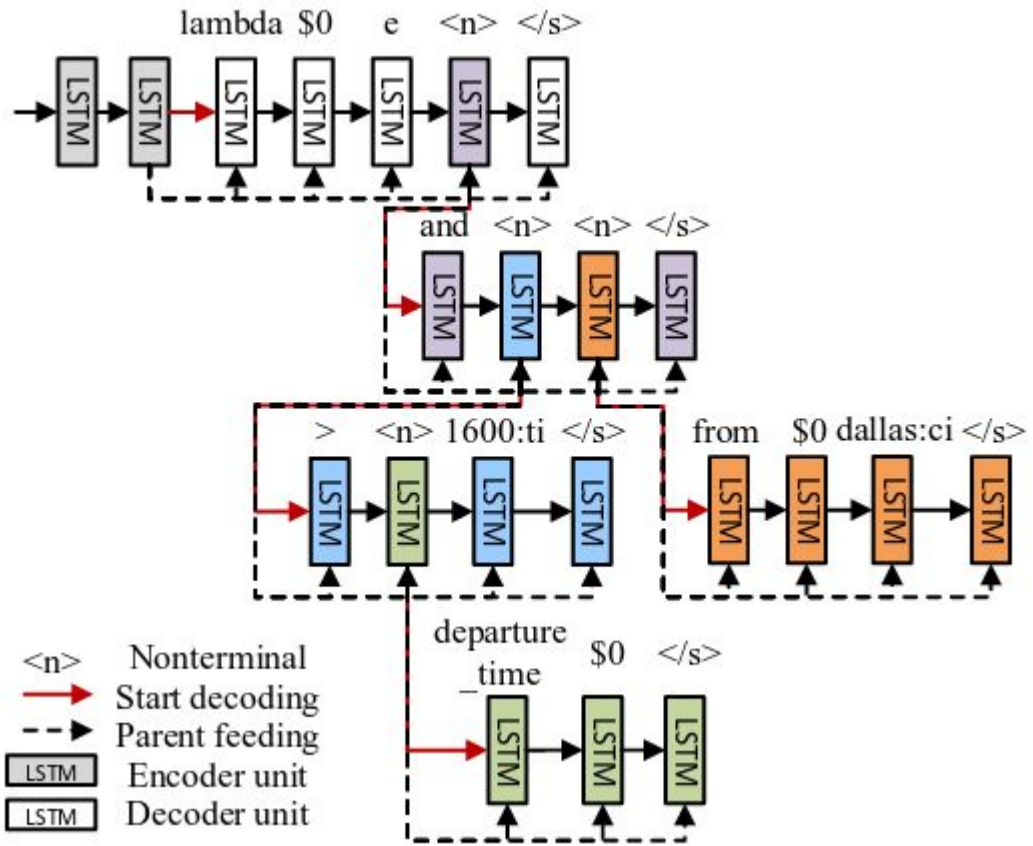


Parent feeding

Every unit gets not only the output of its previous timestep, but also its parent's output.






lambda $0 e <n> </s>
and <n> <n> </s>
> <n> 1600:ti </s>
from $0 dallas:ci </s>
departure _time $0 </s>
LSTM
<n> Nonterminal
Start decoding
Parent feeding
LSTM Encoder unit
LSTM Decoder unit

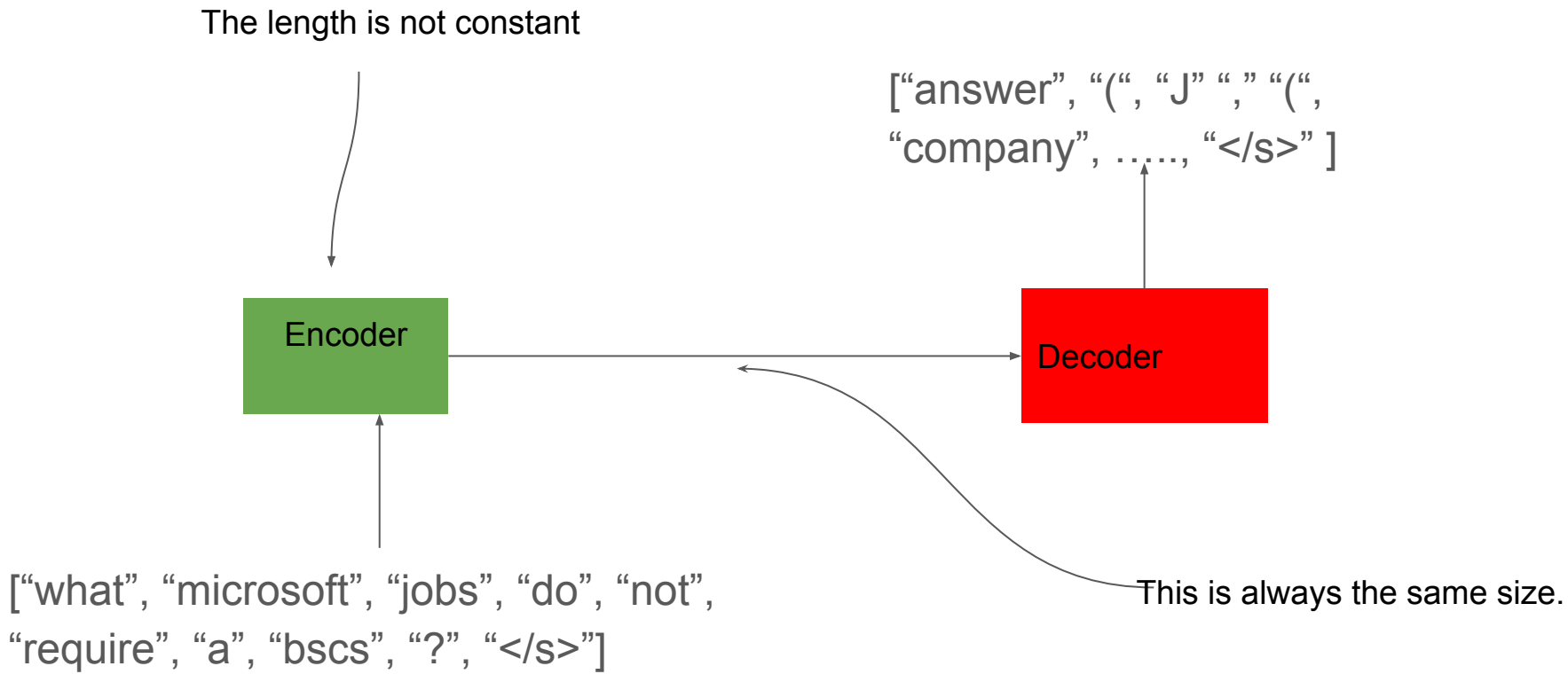
End of tree model

Questions?

Attention

“You can't cram the meaning of a whole %&!$# sentence into a single $&!#* vector!" -Raymond Mooney

Reminder


The length is not constant
[“answer”, “(“, “J” “,” “(“,
“company”, ….., “</s>”]
Encoder
Decoder
[“what”, “microsoft”, “jobs”, “do”, “not”,
“require”, “a”, “bscs”, “?”, “</s>”]
This is always the same size.

Bahdanu et. al. (2014) showed that as sentence length passes 20 words, translation quality drops.

How can we fix this?

Lets “attend” at every timestep only to the relevant words.

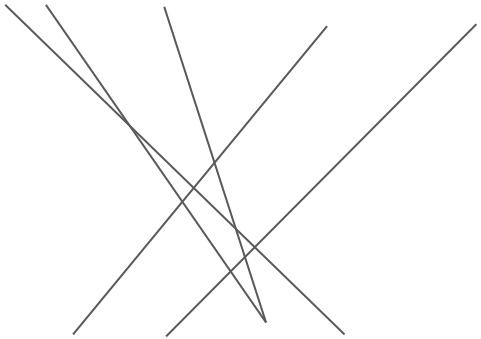
Simple example

She wanted a green bicycle. →

היא רצתה אופניים ירוקים

Simple example

She wanted a green bicycle. →



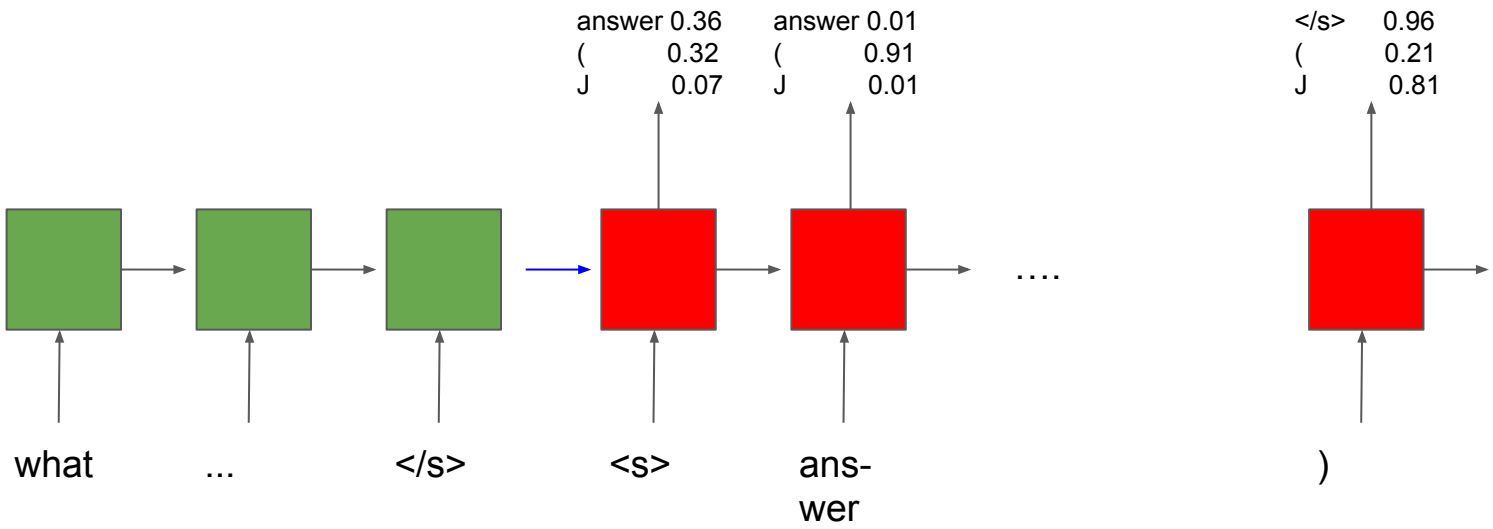
היא רצתה אופניים ירוקים

This also occurs in logical forms!

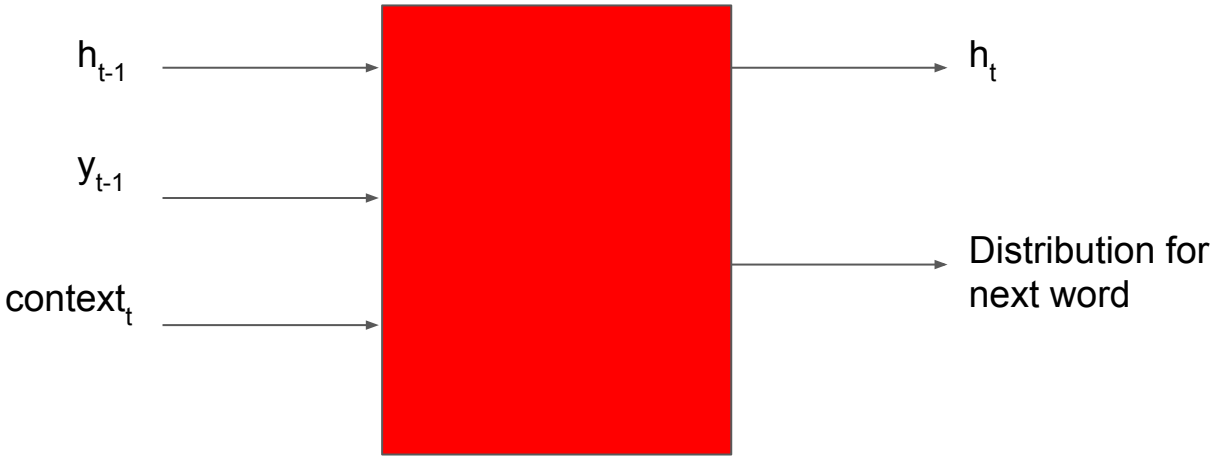
what microsoft jobs do not require a bscs? →

answer(J,(company(J,'microsoft'),job(J),not((req_deg(J,'bscs')))))

Reminder:

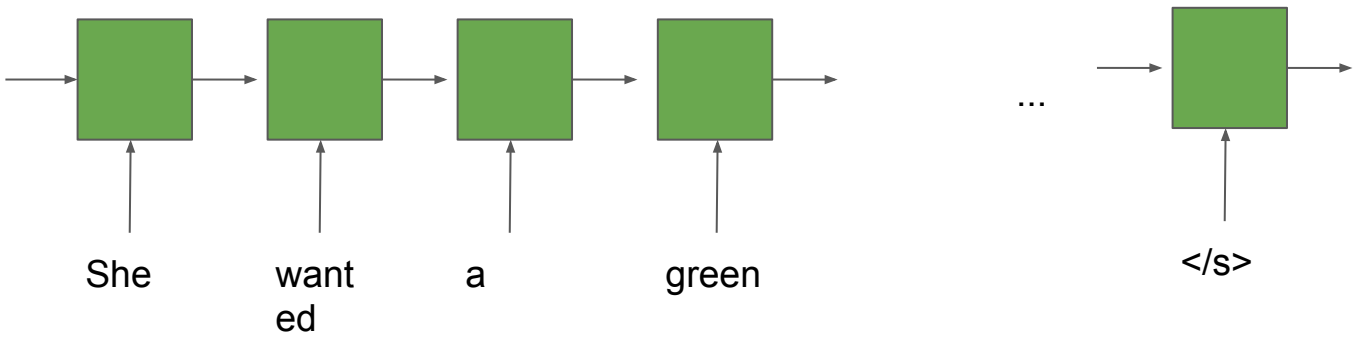

answer 0.36
(0.32
J 0.07
answer 0.01
(0.91
J 0.01
</s> 0.96
(0.21
J 0.81
....
what
...
</s>
<s>
ans-
wer
)

Attention decoder:

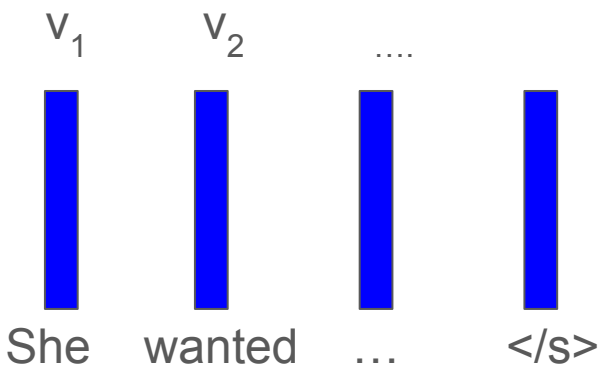


How do we calculate the context vector?

Encode the words once at the beginning



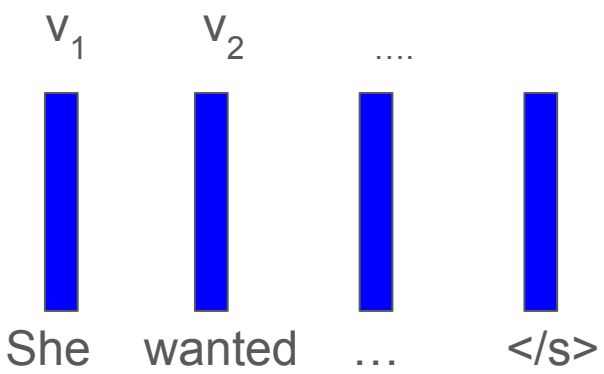
Now for every word we have a representation



t=1

$a(v_i , h_{t-1})$

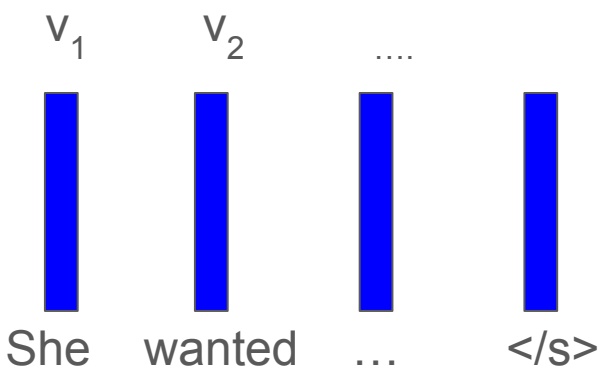
60 3 1 3 1 ...



t=2

$a(v_i , h_{t-1})$

70 70 1 3 1 ...

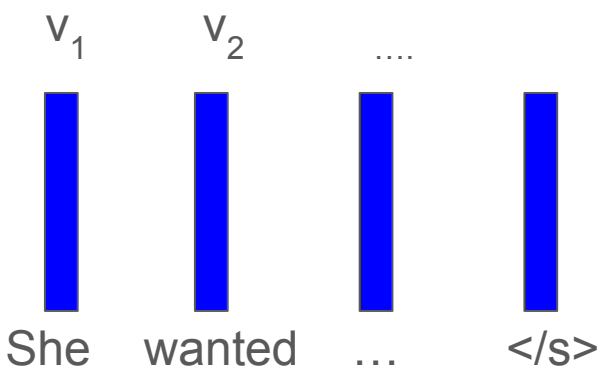


t=1

$a(v_i , h_{t-1})$

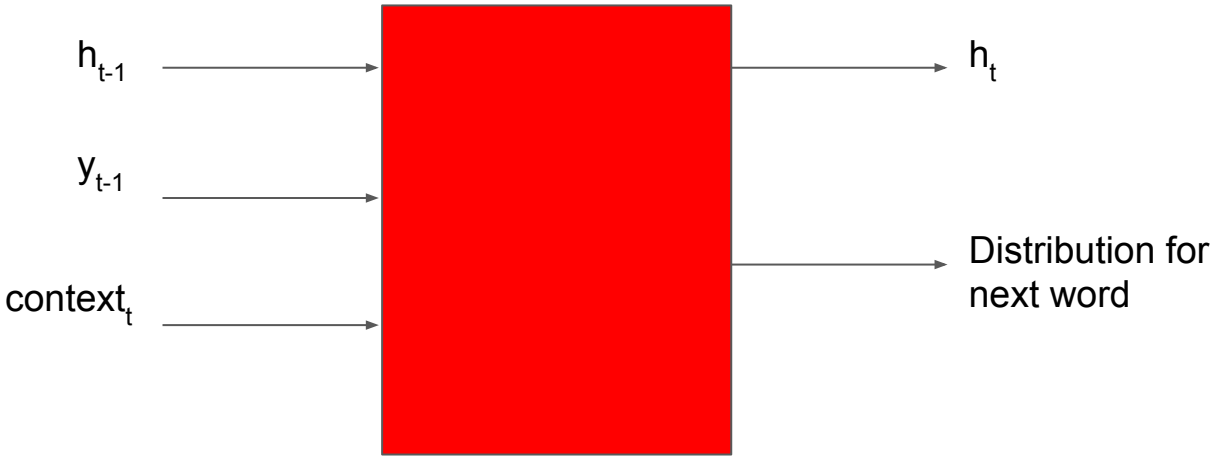
0.90 0.02 0.001 0.02 .001 ← normalize

60 3 1 3 1 ...

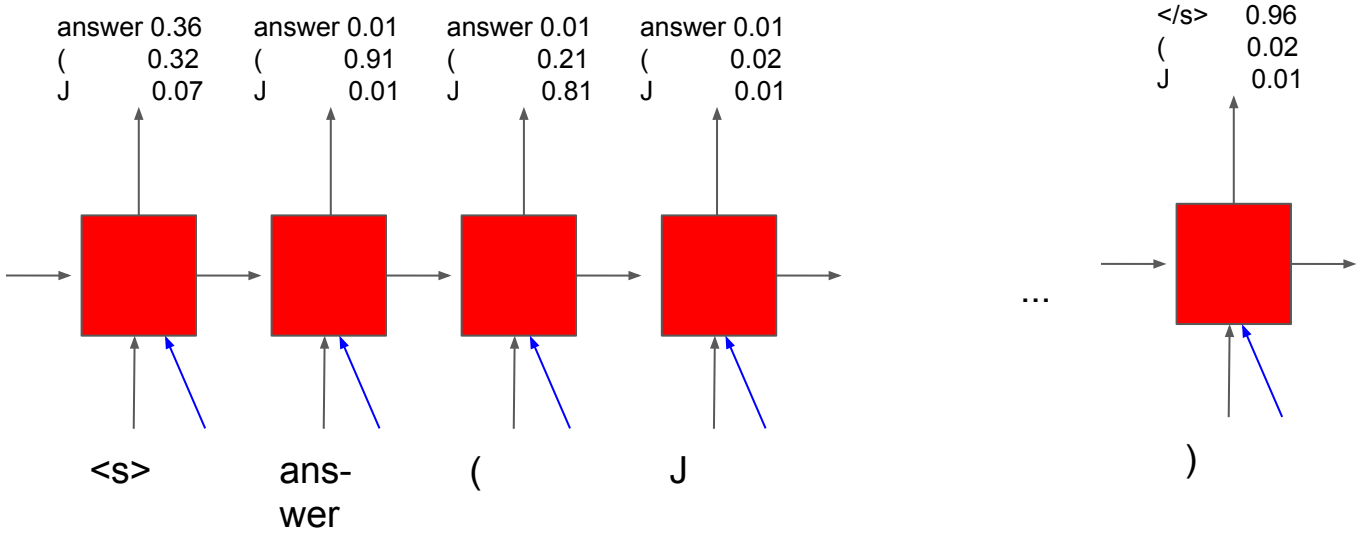


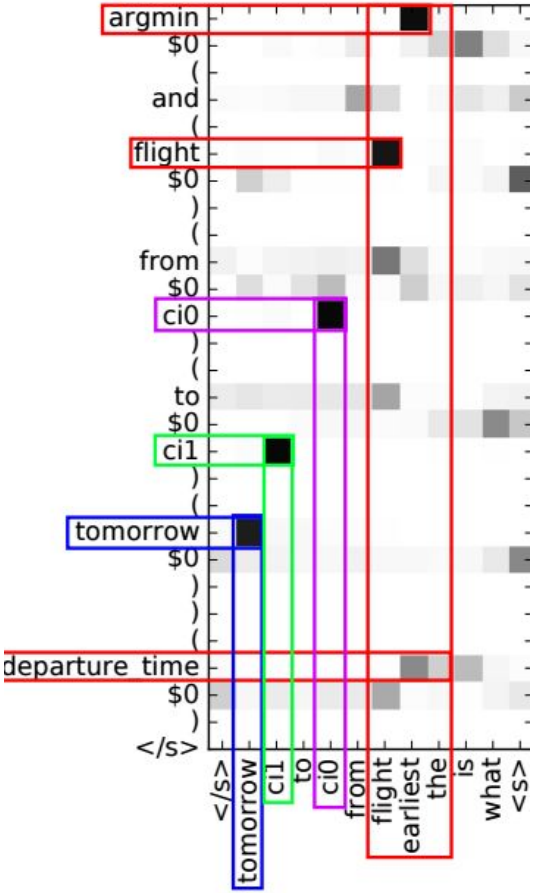
$context_t = \Sigma$ normalized($a(v_i , h_{t-1})$) * v_i

Attention decoder:

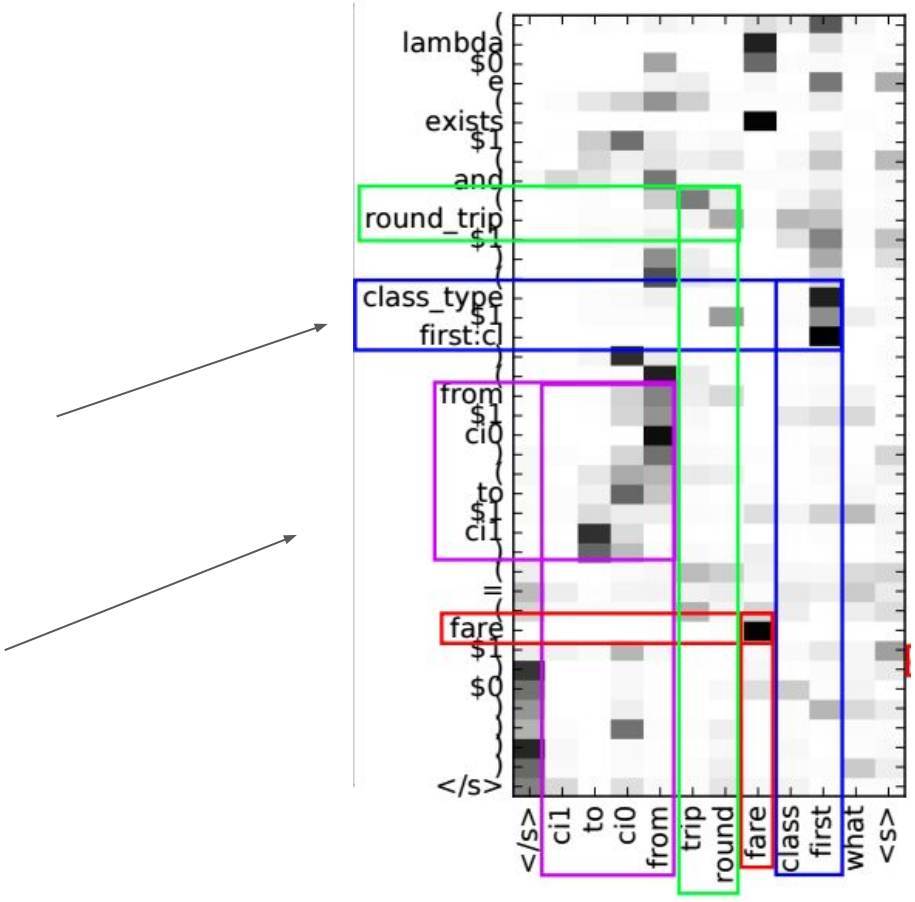


Decoder: now with attention





(c) what is the earliest flight from ci0 to ci1 tomorrow



(b) what's first class fare round trip from ci0 to ci1

Accuracy: The proportion of the input sentences that are correctly parsed to their gold standard logical forms

Method	Accuracy
SCISSOR (Ge and Mooney, 2005)	72.3
KRISP (Kate and Mooney, 2006)	71.7
WASP (Wong and Mooney, 2006)	74.8
λ-WASP (Wong and Mooney, 2007)	86.6
LNLZ08 (Lu et al., 2008)	81.8
ZC05 (Zettlemoyer and Collins, 2005)	79.3
ZC07 (Zettlemoyer and Collins, 2007)	86.1
UBL (Kwiatkowski et al., 2010)	87.9
FUBL (Kwiatkowski et al., 2011)	88.6
KCAZ13 (Kwiatkowski et al., 2013)	89.0
DCS+L (Liang et al., 2013)	87.9
TISP (Zhao and Huang, 2015)	88.9
SEQ2SEQ	84.6
− attention	72.9
− argument	68.6
SEQ2TREE	87.1
− attention	76.8

Table 3: Evaluation results on GEO. 10-fold cross-validation is used for the systems shown in the top half of the table. The standard split of ZC05 is used for all other systems.

Method	Accuracy
ZC07 (Zettlemoyer and Collins, 2007)	84.6
UBL (Kwiatkowski et al., 2010)	71.4
FUBL (Kwiatkowski et al., 2011)	82.8
GUSP-FULL (Poon, 2013)	74.8
GUSP++ (Poon, 2013)	83.5
TISP (Zhao and Huang, 2015)	84.2
SEQ2SEQ	84.2
− attention	75.7
− argument	72.3
SEQ2TREE	84.6
− attention	77.5

Table 4: Evaluation results on ATIS.

Method	Channel	+Func	F1
retrieval	28.9	20.2	41.7
phrasal	19.3	11.3	35.3
sync	18.1	10.6	35.1
classifier	48.8	35.2	48.4
posclass	50.0	36.9	49.3
SEQ2SEQ	54.3	39.2	50.1
− attention	54.0	37.9	49.8
− argument	53.9	38.6	49.7
SEQ2TREE	55.2	40.1	50.4
− attention	54.3	38.2	50.0

(a) Omit non-English.

Method	Channel	+Func	F1
retrieval	36.8	25.4	49.0
phrasal	27.8	16.4	39.9
sync	26.7	15.5	37.6
classifier	64.8	47.2	56.5
posclass	67.2	50.4	57.7
SEQ2SEQ	68.8	50.5	60.3
− attention	68.7	48.9	59.5
− argument	68.8	50.4	59.7
SEQ2TREE	69.6	51.4	60.4
− attention	68.7	49.5	60.2

(b) Omit non-English & unintelligible.

Method	Channel	+Func	F1
retrieval	43.3	32.3	56.2
phrasal	37.2	23.5	45.5
sync	36.5	24.1	42.8
classifier	79.3	66.2	65.0
posclass	81.4	71.0	66.5
SEQ2SEQ	87.8	75.2	73.7
− attention	88.3	73.8	72.9
− argument	86.8	74.9	70.8
SEQ2TREE	89.7	78.4	74.2
− attention	87.6	74.9	73.5

(c) ≥ 3 turkers agree with gold.

Table 5: Evaluation results on IFTTT.

Problems

- Rare words
- Cant remember past attention values

End. Questions?