Sequence to Sequence models

Rasmus Berg Palm

Technical University of Denmark
&
Tradeshift

Sequence to Sequence models

$$\mathbf{x} \rightarrow \text{Model} \rightarrow \mathbf{y}$$

Sequence to Sequence models

x: “the dog ate my homework”

y: “El perro se comió mi tarea”

Sequence to Sequence models

$$\mathbf{x}\text{: } [x_1, x_2, \ldots, x_n]$$

x_i: one-hot encoded word

$$\mathbf{x}\text{.shape} = [n, \text{x-vocab}]$$

Sequence to Sequence models

$$\mathbf{y}\text{: } [y_1, y_2, \ldots, y_m]$$

y_i: one-hot encoded word

$$\mathbf{y}.\text{shape} = [m, \text{y-vocab}]$$

Sequence to Sequence models

Sutskever, Ilya, Oriol Vinyals, and Quoc V. Le. "**Sequence to sequence learning with neural networks**." Advances in neural information processing systems. 2014.

https://papers.nips.cc/paper/5346-sequence-to-sequence-learning-with-neural-networks.pdf

Sequence to Sequence models

1. Squeeze the entire **x** into a single vector v
2. Generate **y** conditioned on v

Sequence to Sequence models

1. Squeeze the entire **x** into a single vector v

Ideas?

Sequence to Sequence models

Bag of Words

$$v = \mathrm{sum}(\mathbf{x})$$

nah...

Sequence to Sequence models

Bag of Embeddings

v = sum(embed(**x**))

nah...

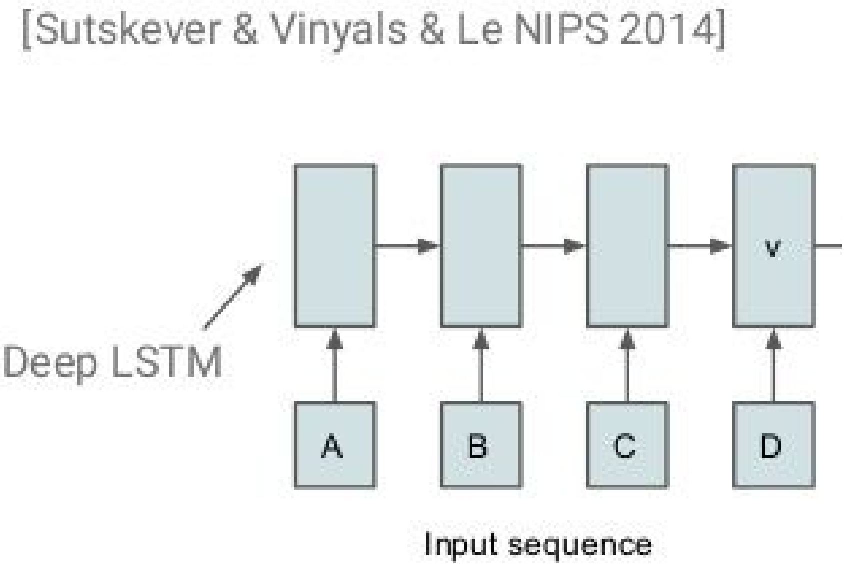
Sequence to Sequence models

LSTM encoding

v = LSTM(**x**)[-1]

yup!

Sequence to Sequence models



Sequence to Sequence models

1. Squeeze the entire **x** into a single vector v
2. Generate **y** conditioned on v

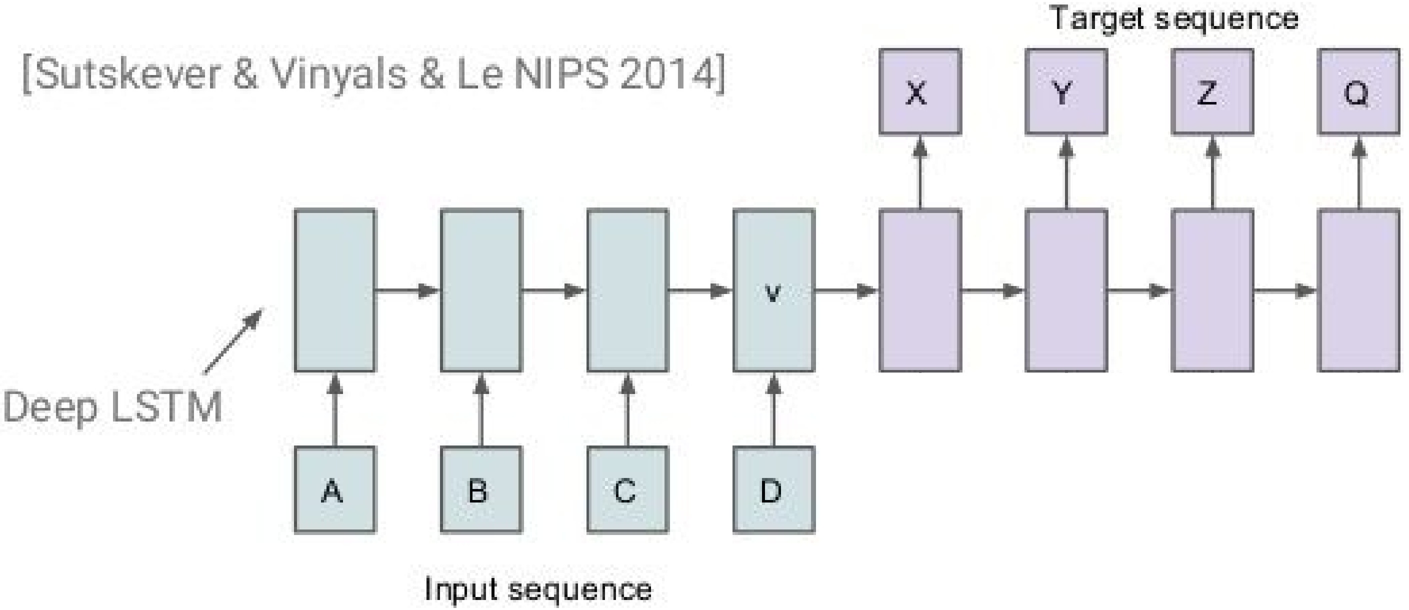
Ideas?

Sequence to Sequence models

$$\mathbf{y} = \text{LSTM}^*$$

*Set initial hidden state to v

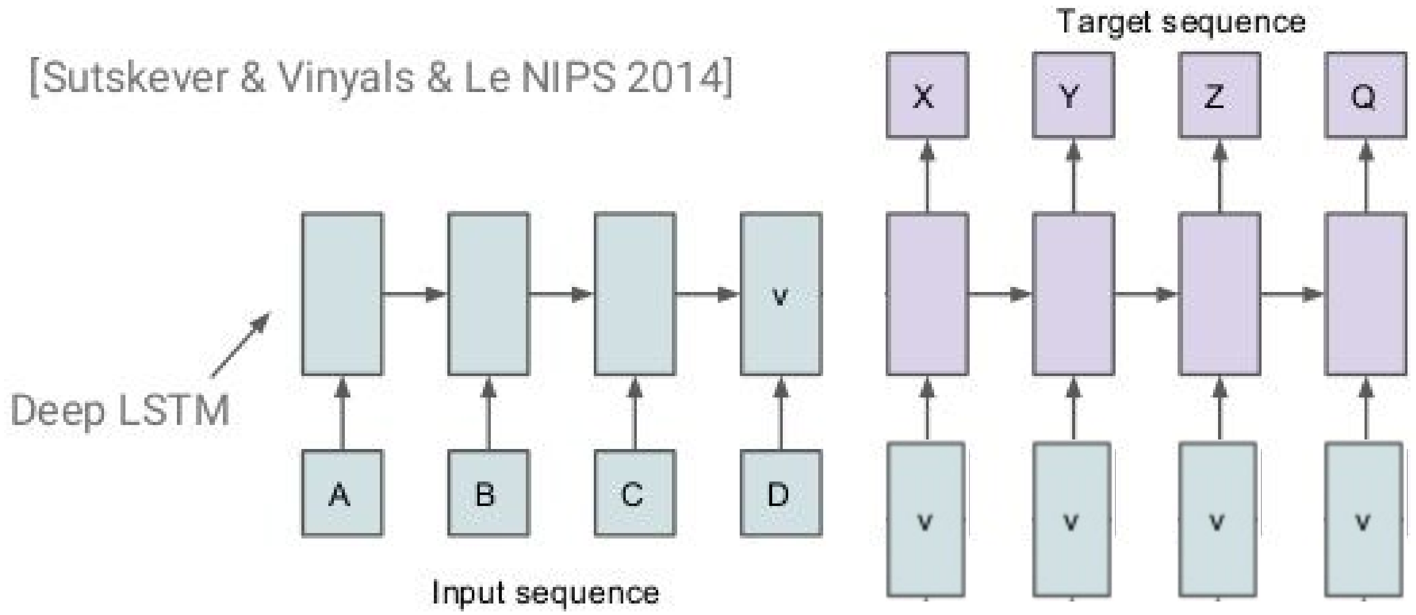
Sequence to Sequence models



Sequence to Sequence models

Variant that is easier to code (and better)

Sequence to Sequence models



Sequence to Sequence models

OK.

Let’s do it!

https://github.com/rasmusbergpalm/normalization

Sequence to Sequence models

x: 12 November 2016

y: 2016-11-12

Sequence to Sequence models

```
# Encoder
source = Input(shape=(None,), dtype='int32', name='source')
embedded = Embedding(output_dim=128, input_dim=train.source_vocab_size(), mask_zero=True)(source)
last_hid = LSTM(output_dim=128)(embedded)

# Decoder
repeated = RepeatVector(train.target.padded.shape[1])(last_hid)
decoder = LSTM(output_dim=128, return_sequences=True)(repeated)
output = TimeDistributed(Dense(output_dim=train.target_vocab_size(), activation='softmax'))(decoder)
model = Model([source], output=[output])
```

Sequence to Sequence models

http://localhost:5000

Sequence to Sequence models

Trick

Feed the LSTM the last output it made

Sequence to Sequence models

Two ways to implement

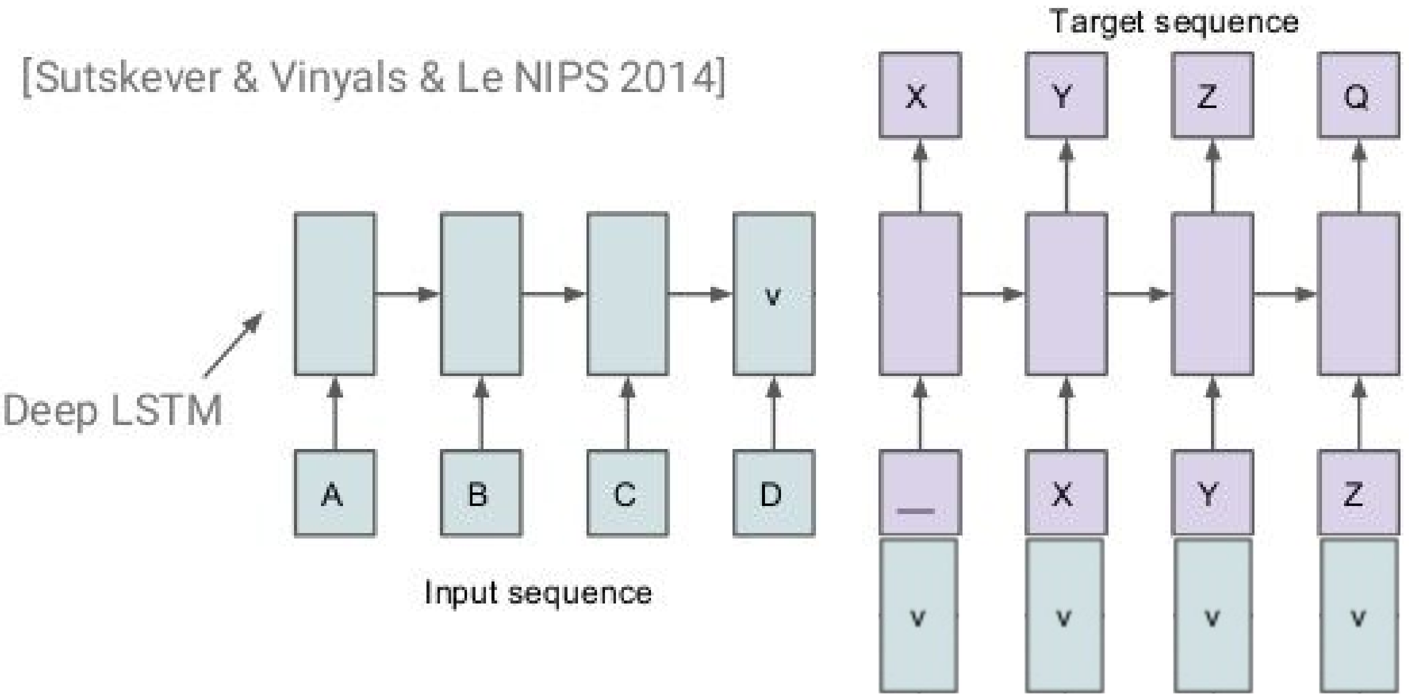
1. Feed the actual probabilities outputted

   Hard, not used very often

2. Feed the target shifted by one

   Easy, very used. AKA. “teacher forcing”

Sequence to Sequence models

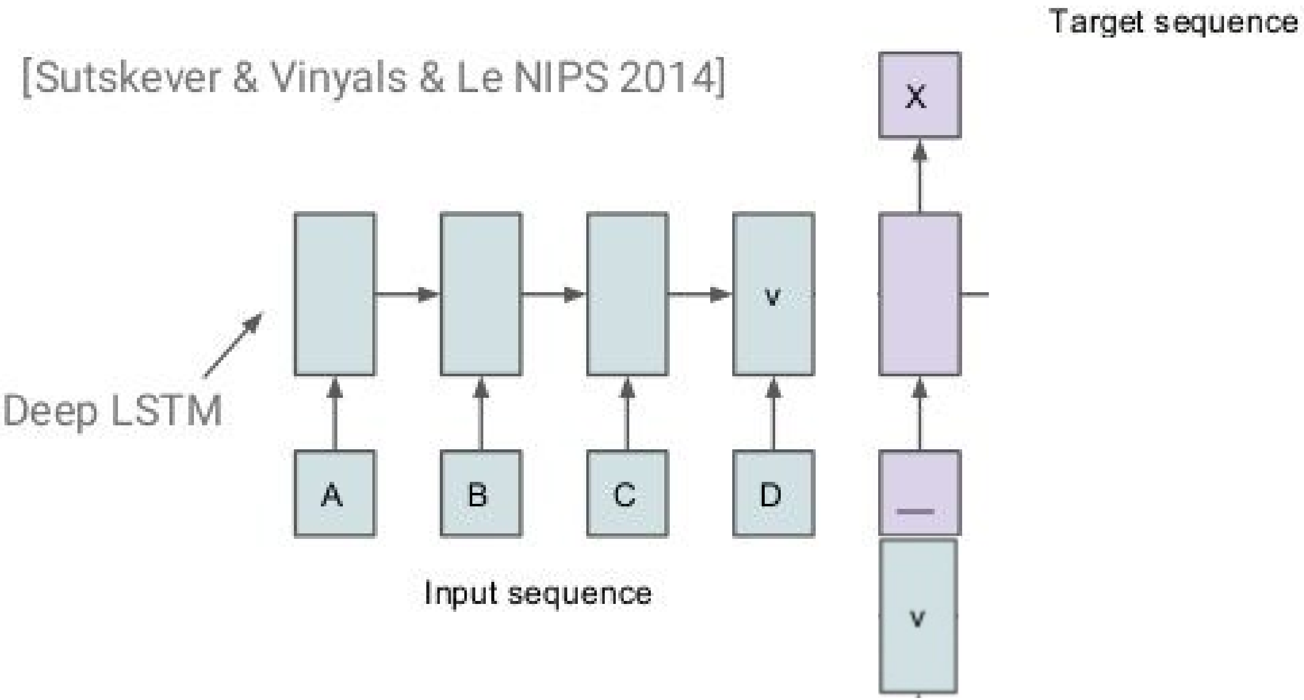


Sequence to Sequence models

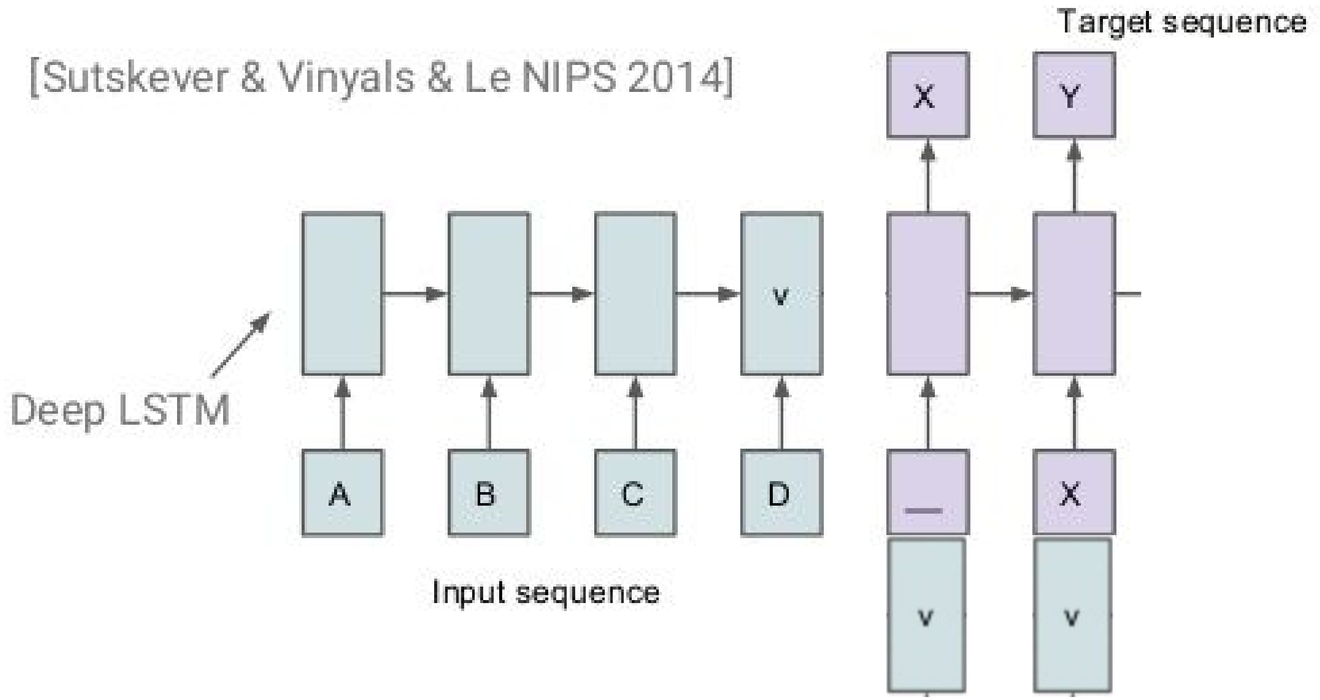
But...

How to generate at test time then?

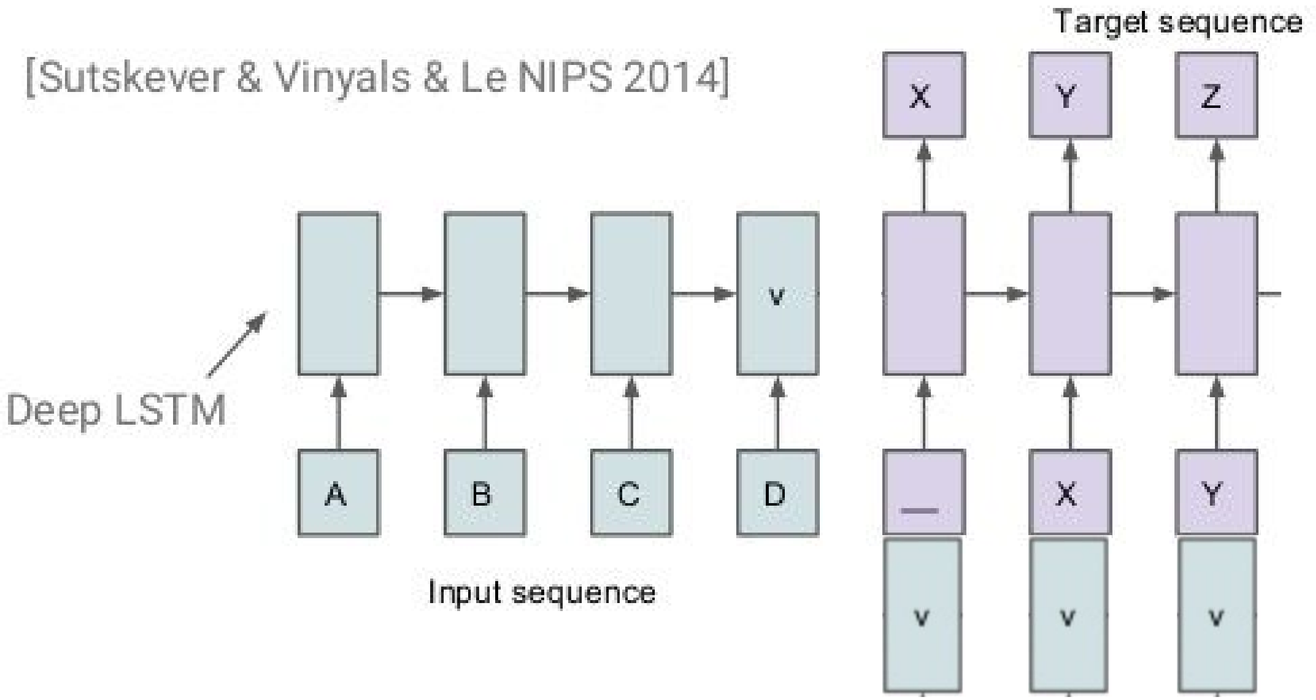
Sequence to Sequence models



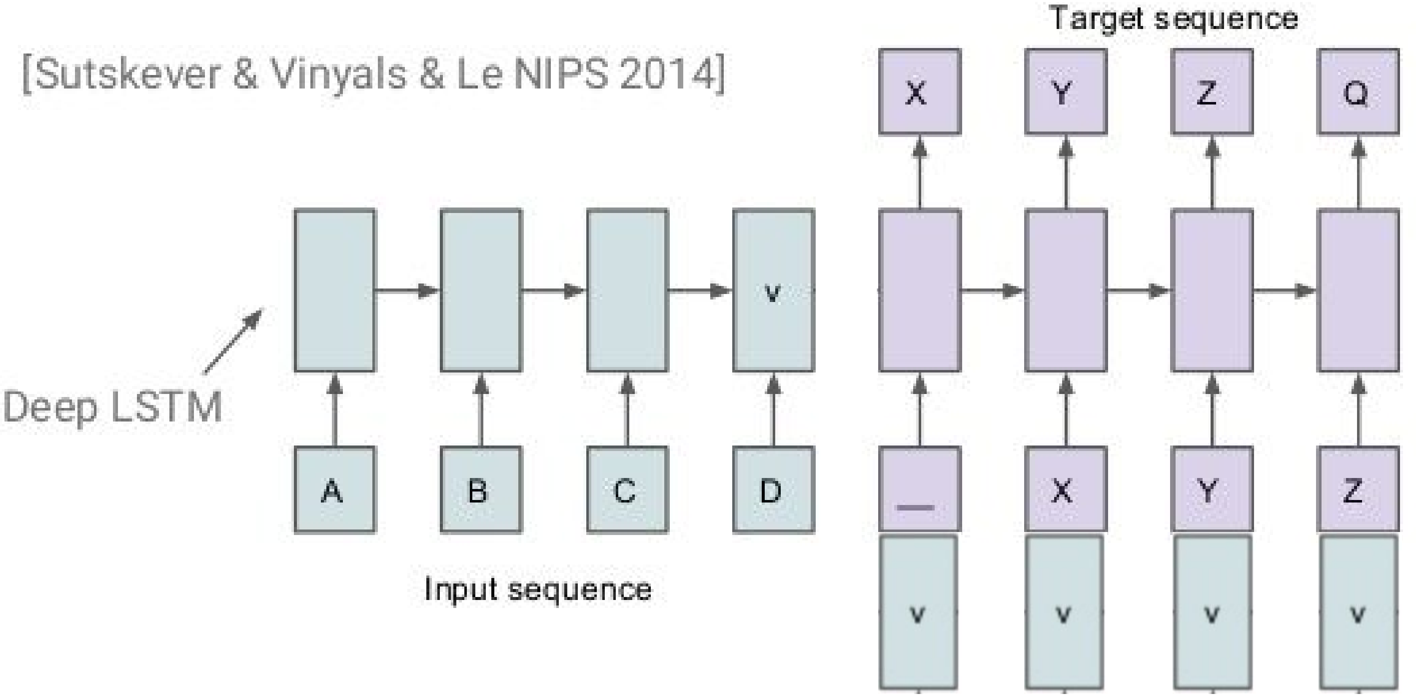
Sequence to Sequence models



Sequence to Sequence models



Sequence to Sequence models



Sequence to Sequence models

Exercise left for the reader

Implement teacher forcing in the date parser

Sequence to Sequence models

The great weakness.

Ideas?

Sequence to Sequence models

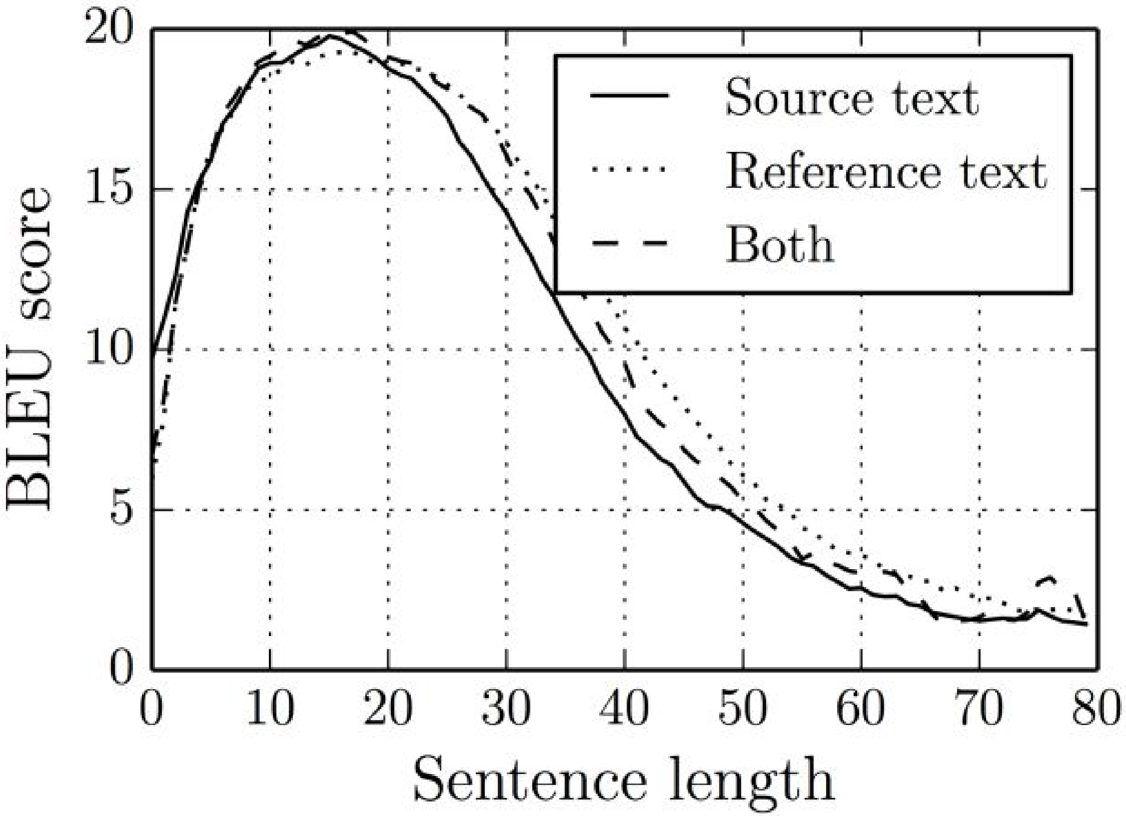
The great weakness.

x is n long

v is fixed size

Hard to compress when n grows

Sequence to Sequence models



Sequence to Sequence models

The great solution.

Ideas?

Sequence to Sequence models

Bahdanau, Dzmitry, Kyunghyun Cho, and Yoshua Bengio. "**Neural machine translation by jointly learning to align and translate**." arXiv preprint arXiv:1409.0473 (2014)

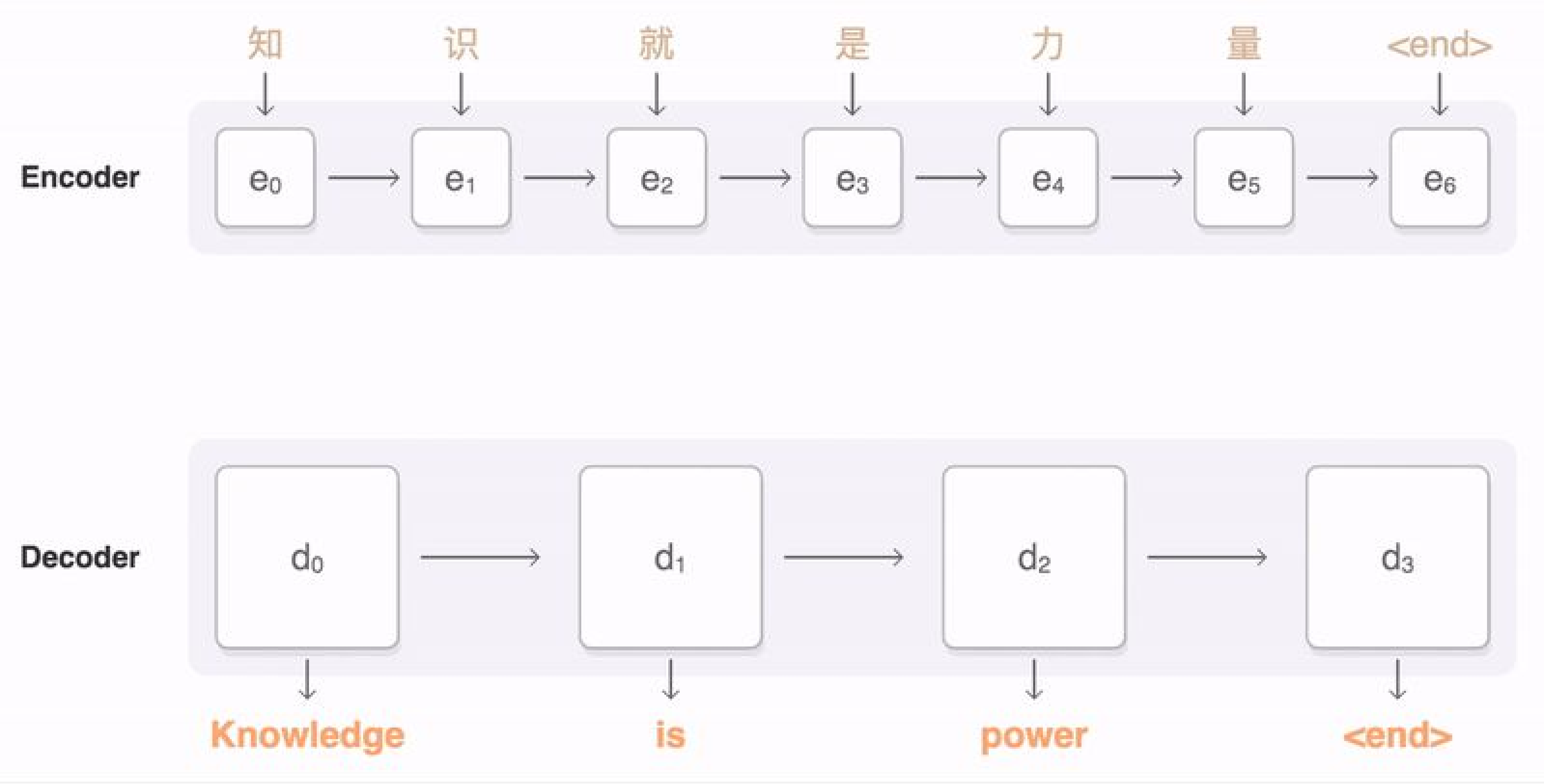
https://arxiv.org/abs/1409.0473

Sequence to Sequence models

Let the decoder look at the entire input sequence for every output

AKA. “Attention”

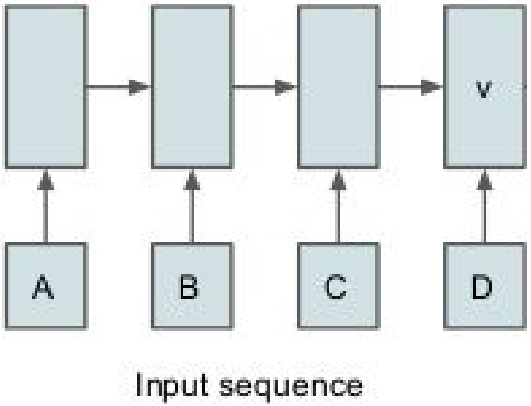
Sequence to Sequence models

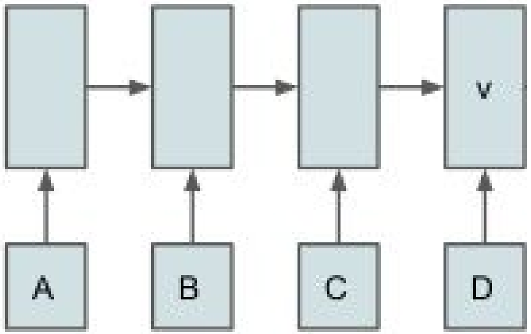


Sequence to Sequence models

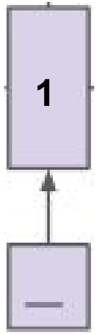
Attention is tricky...

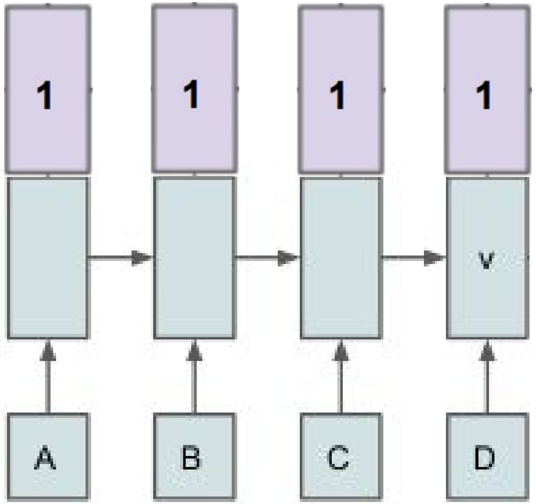
But you're clever :]


v
A
B
C
D
Input sequence

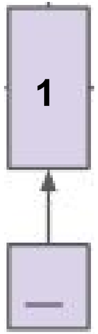


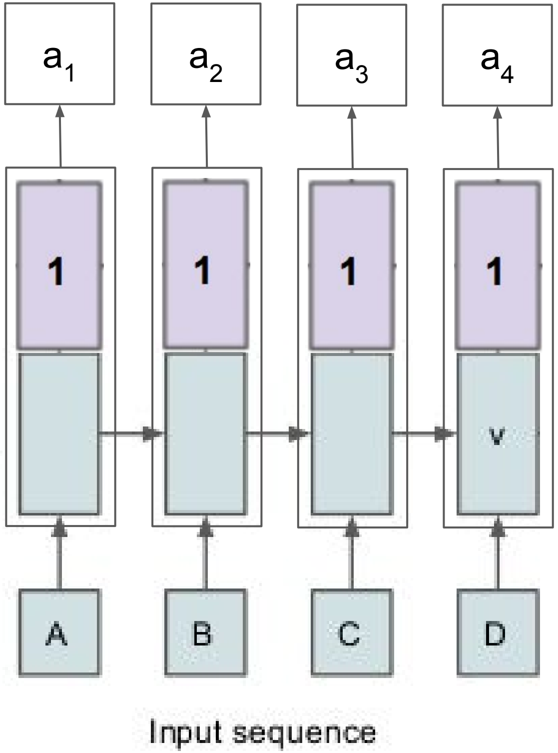
Input sequence



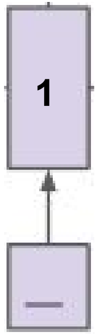


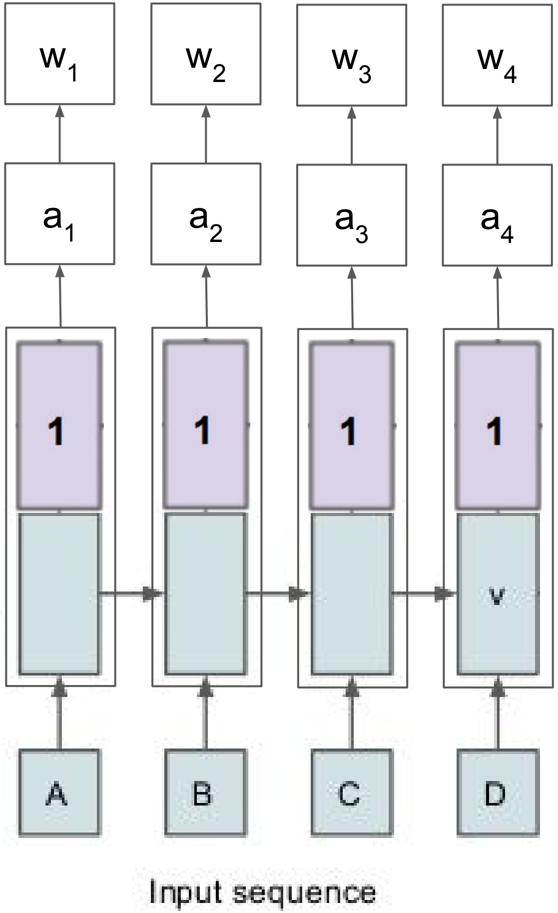
Input sequence





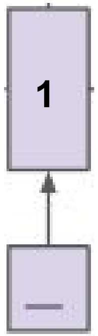
$a_i = f(e_i, d_j)$ (1,)

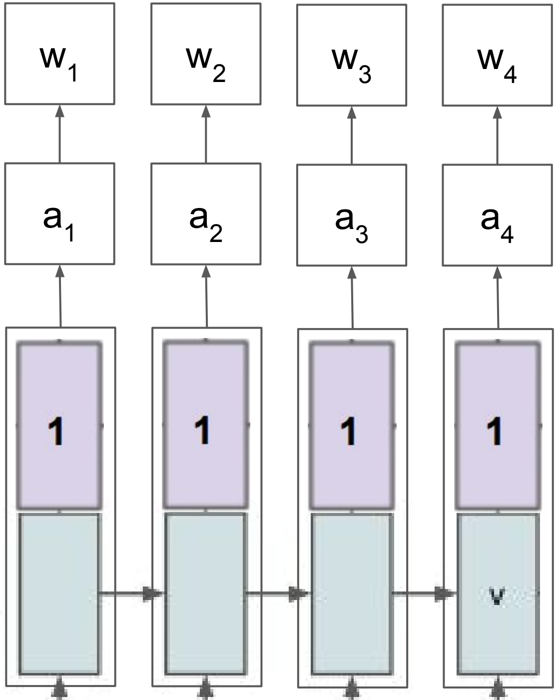




$$w_i = \mathrm{softmax}(a_i) \ (1,)$$

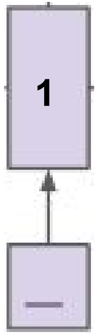
$$a_i = f(e_i, d_j) \ (1,)$$

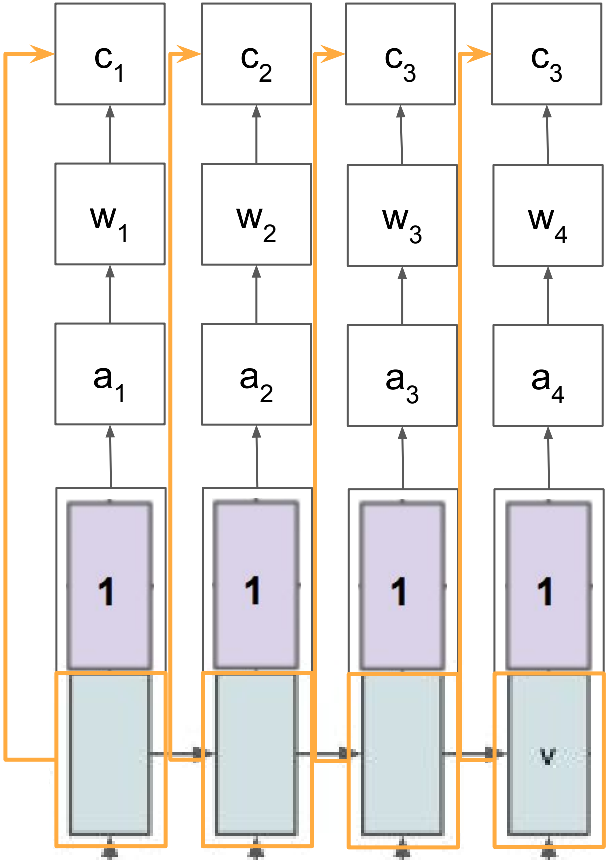




$$w_i = \text{softmax}(a_i) \ (1,)$$

$$a_i = f(e_i, d_j) \ (1,)$$

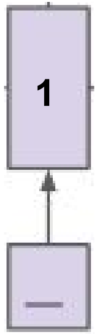


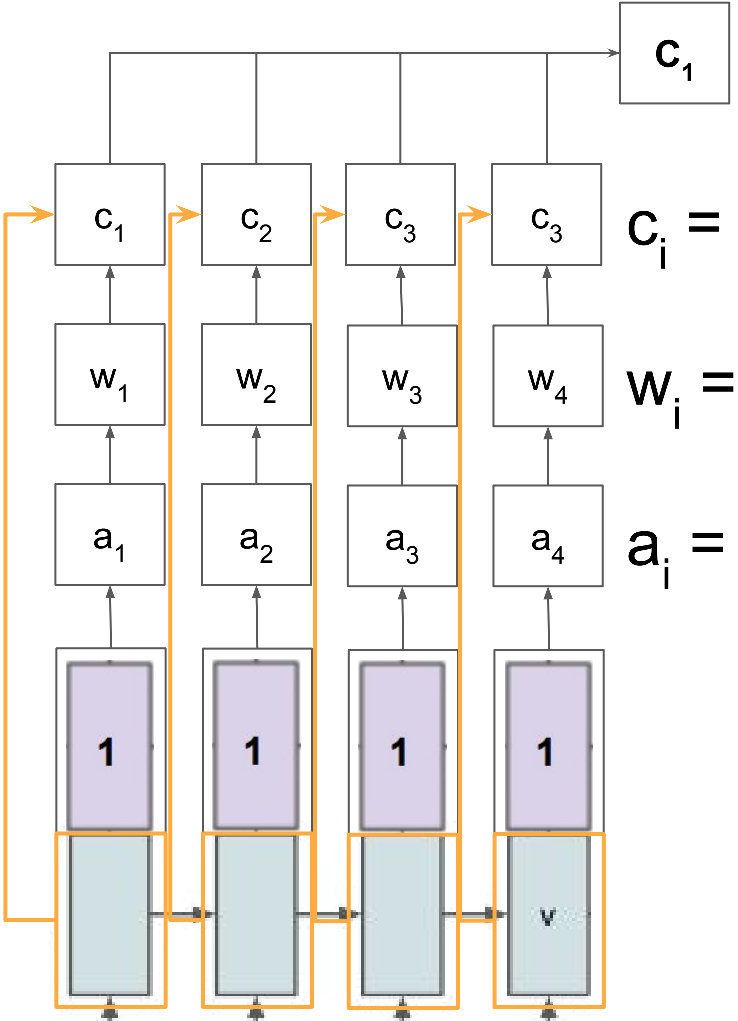


$$c_i = w_i e_i \ (128,)$$

$$w_i = \mathrm{softmax}(a_i) \ (1,)$$

$$a_i = f(e_i, d_j) \ (1,)$$



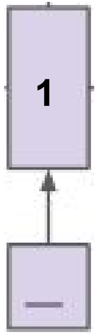


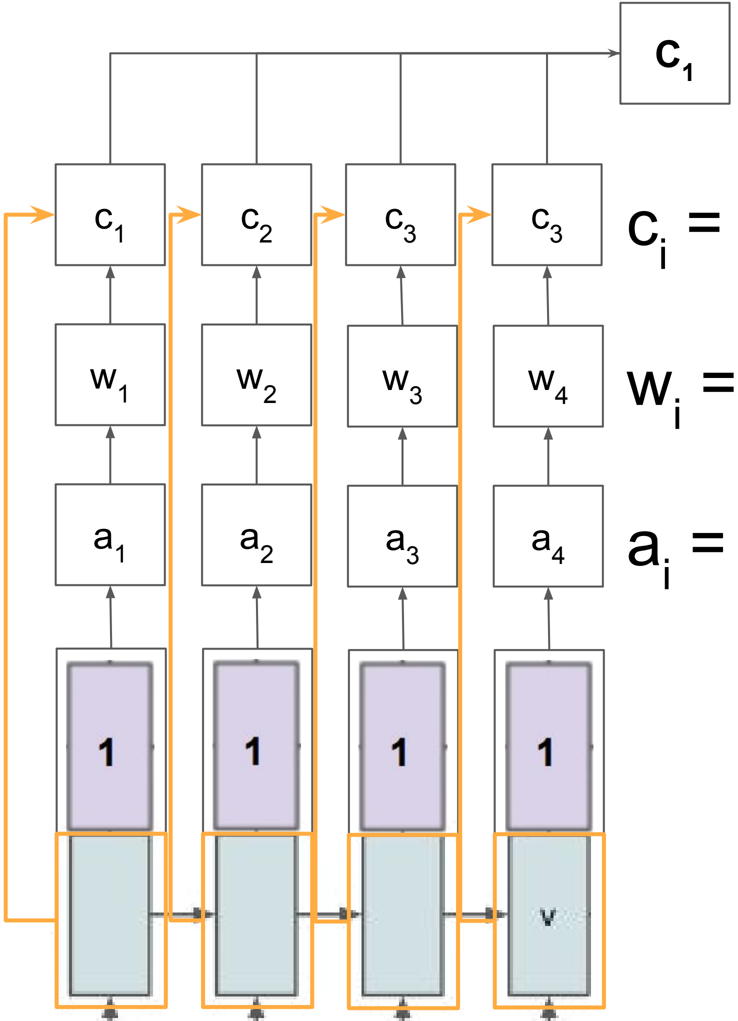
$\mathbf{C_1} = \text{sum}(c_i)$ (128,)

$c_i = w_i e_i$ (128,)

$w_i = \text{softmax}(a_i)$ (1,)

$a_i = f(e_i, d_j)$ (1,)



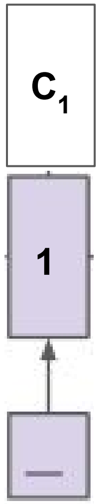


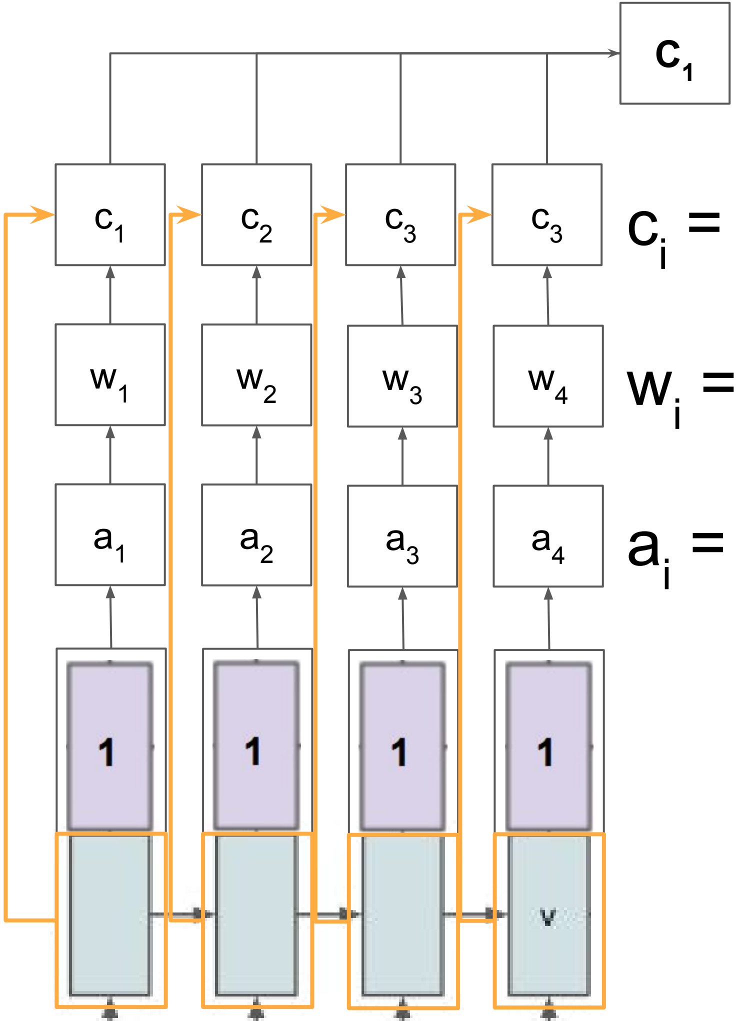
$C_1 = \text{sum}(c_i)$ (128,)

$c_i = w_i e_i$ (128,)

$w_i = \text{softmax}(a_i)$ (1,)

$a_i = f(e_i, d_j)$ (1,)



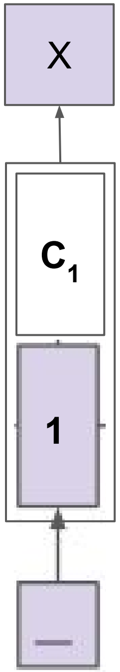


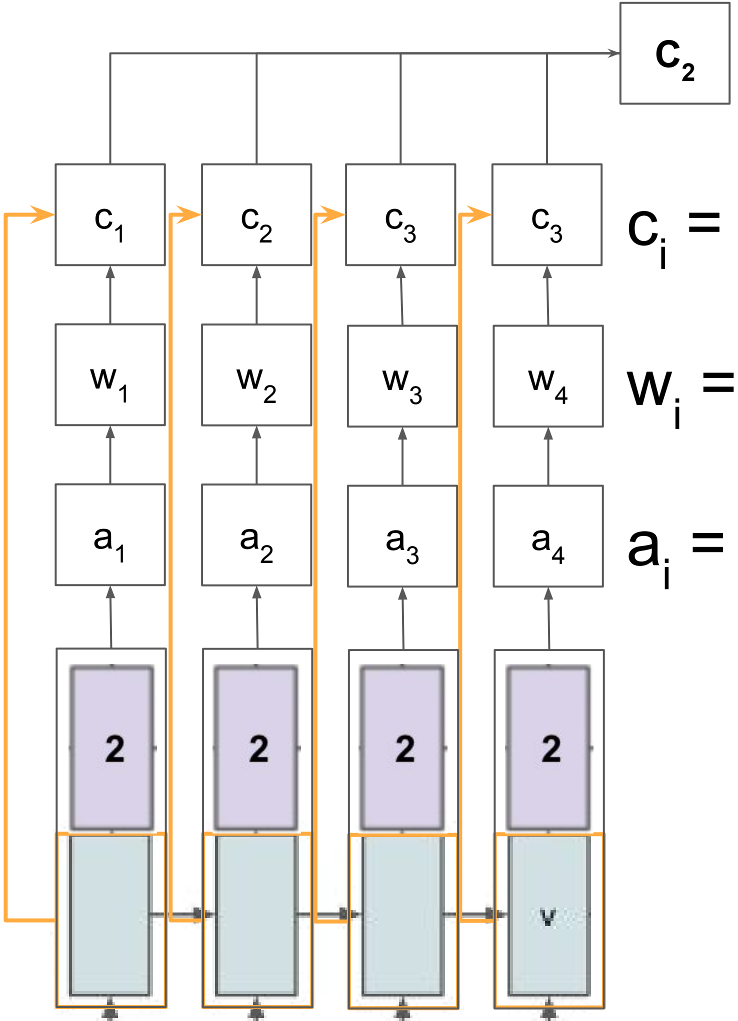
$$C_1 = \mathrm{sum}(c_i)\ (128,)$$

$$c_i = w_i e_i\ (128,)$$

$$w_i = \mathrm{softmax}(a_i)\ (1,)$$

$$a_i = f(e_i, d_j)\ (1,)$$



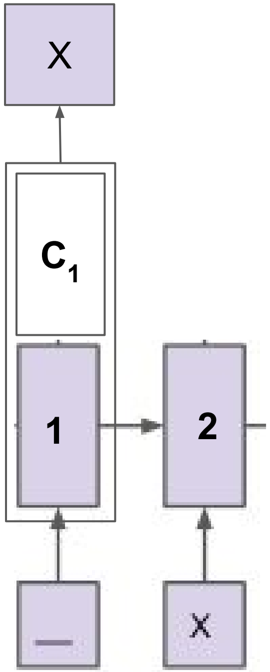


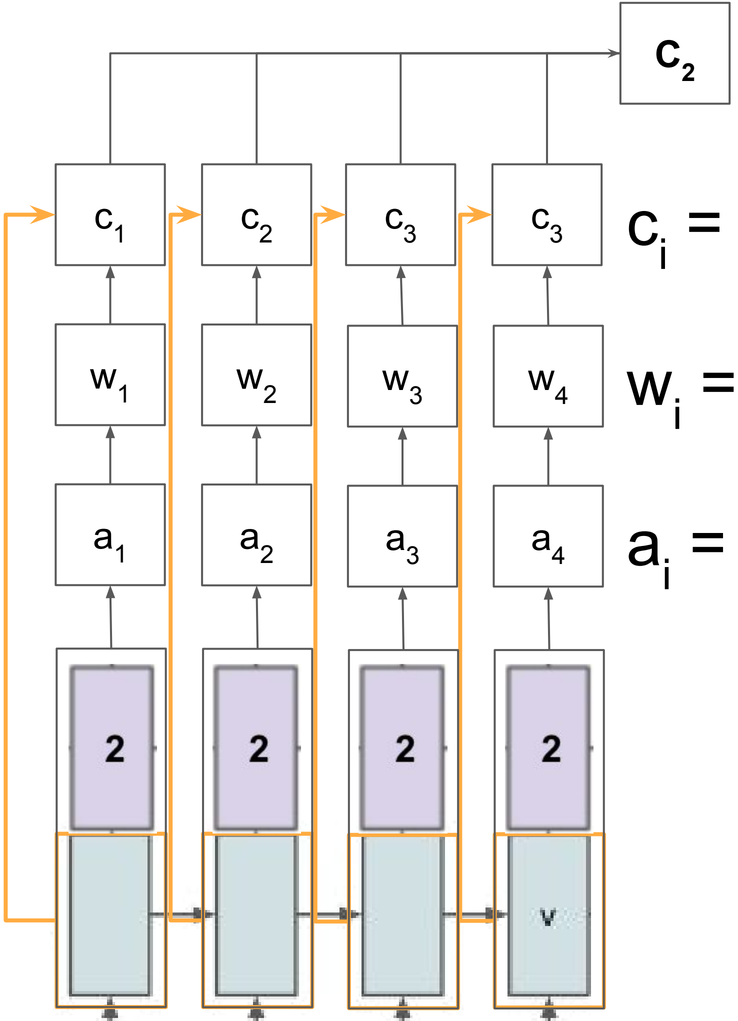
$\mathbf{C_2} = \text{sum}(c_i)$ (128,)

$c_i = w_i e_i$ (128,)

$w_i = \text{softmax}(a_i)$ (1,)

$a_i = f(e_i, d_j)$ (1,)



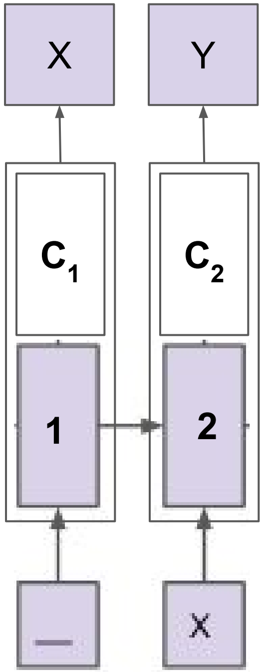


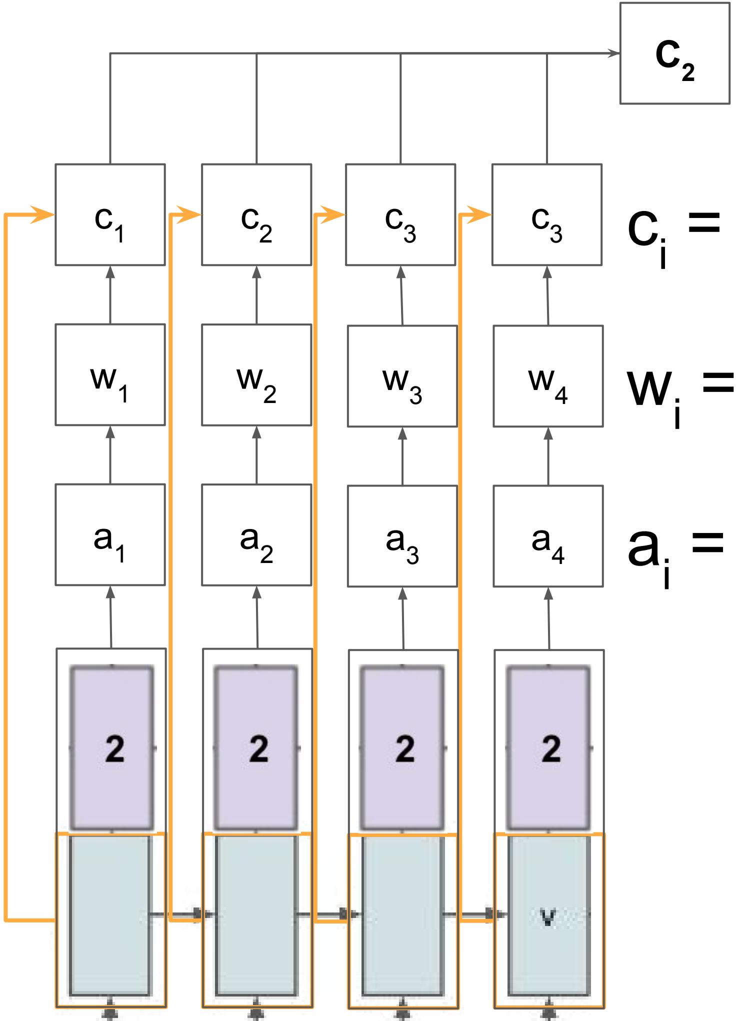
$$\mathbf{C_2} = \mathrm{sum}(c_i)\ (128,)$$

$$c_i = w_i e_i\ (128,)$$

$$w_i = \mathrm{softmax}(a_i)\ (1,)$$

$$a_i = f(e_i, d_j)\ (1,)$$



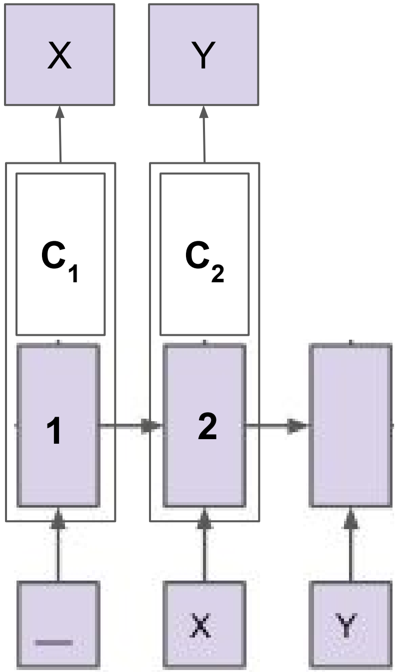


$$C_2 = \mathrm{sum}(c_i) \; (128,)$$

$$c_i = w_i e_i \; (128,)$$

$$w_i = \mathrm{softmax}(a_i) \; (1,)$$

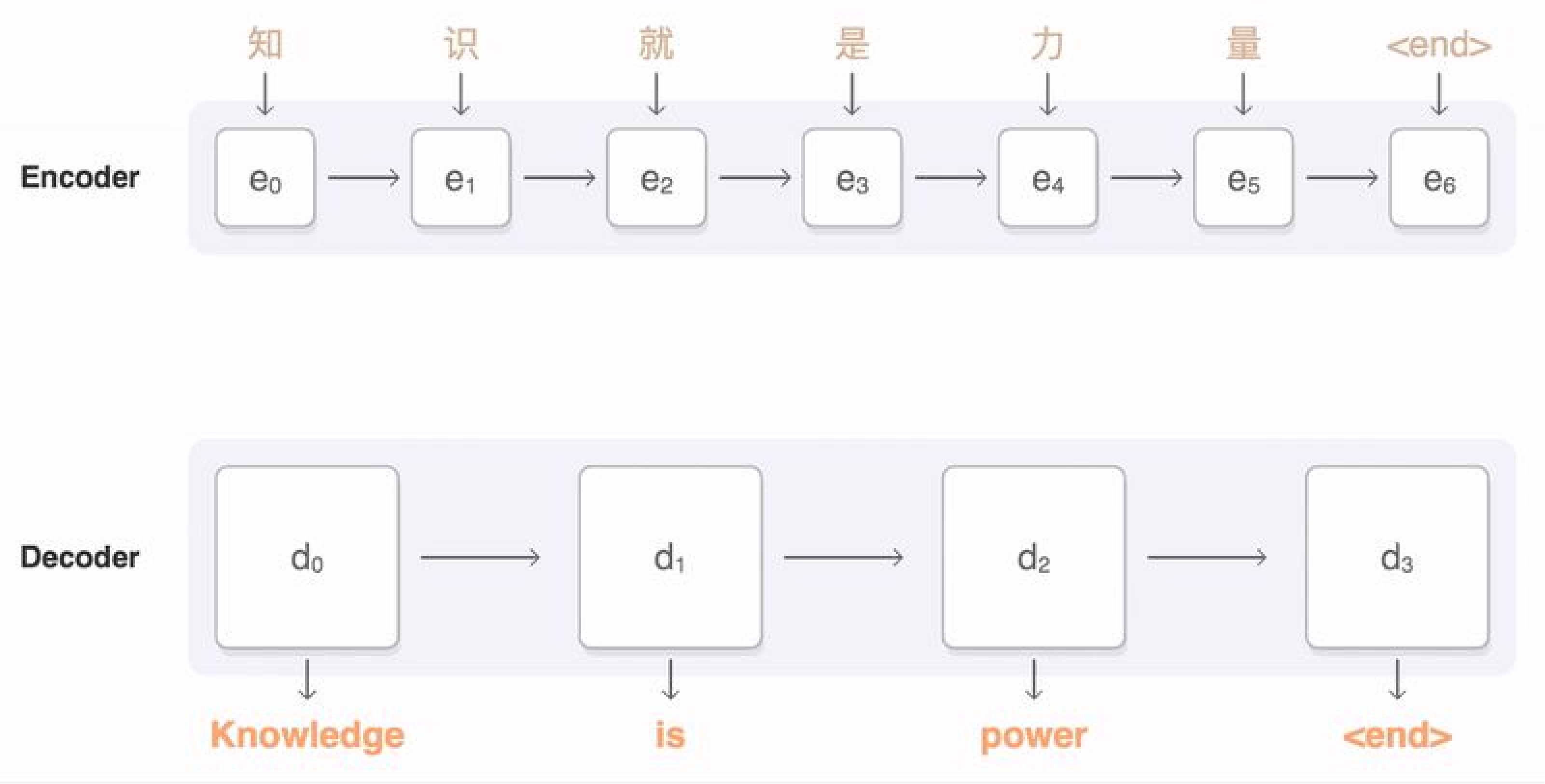
$$a_i = f(e_i, d_j) \; (1,)$$



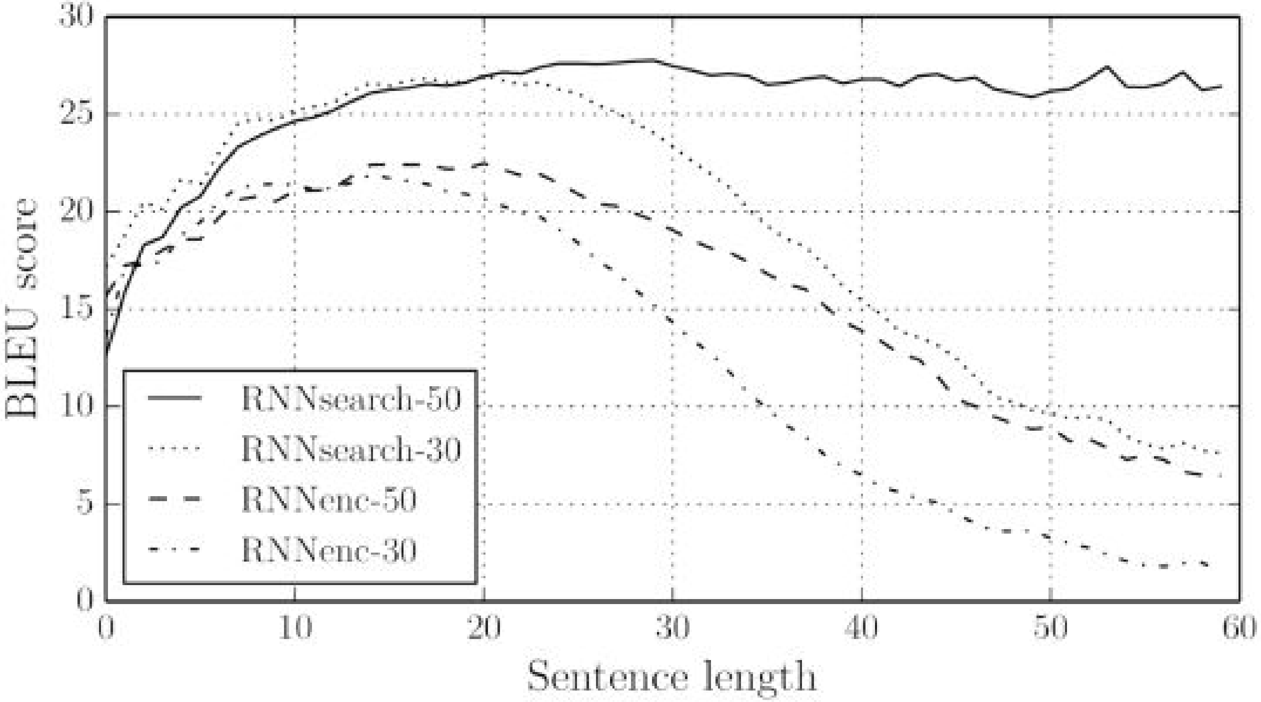
Sequence to Sequence models

Phew!

Sequence to Sequence models



Sequence to Sequence models



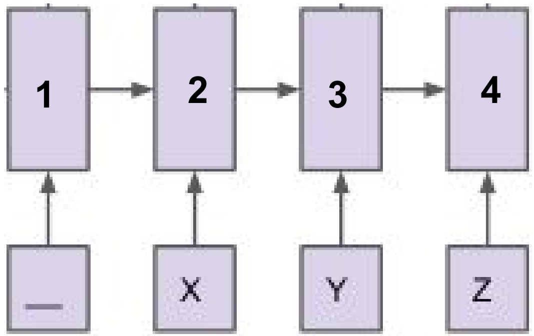
Exercise left for the reader

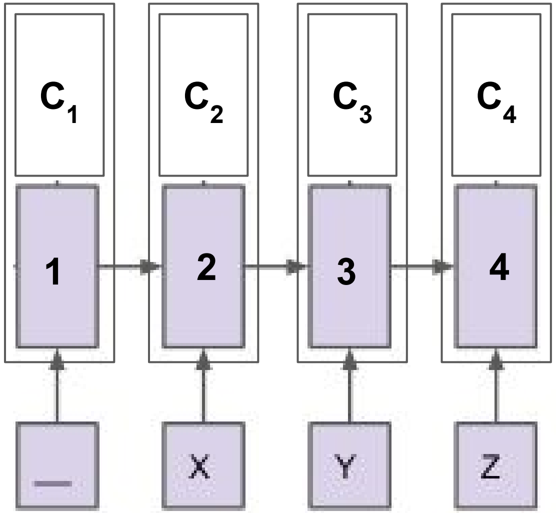
Implement attention for the date parser

Sequence to Sequence models

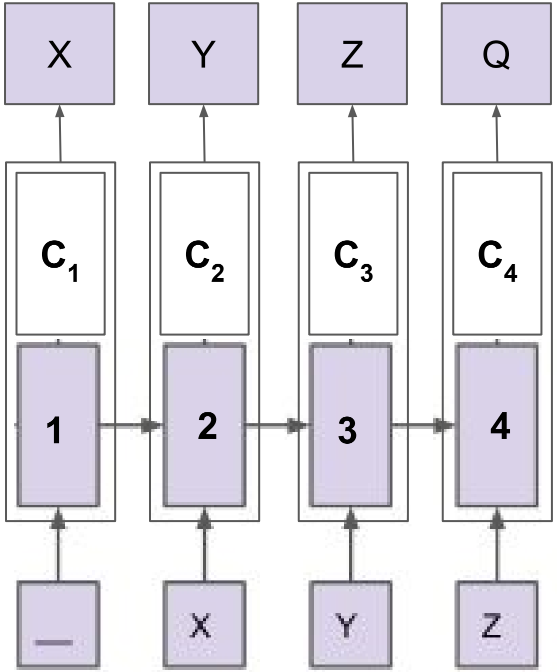
Trick

Teacher forcing makes attention easier to implement


1
2
3
4
_
X
Y
Z



C_1
C_2
C_3
C_4
1
2
3
4
_
X
Y
Z


X
Y
Z
Q
C1
C2
C3
C4
1
2
3
4
_
X
Y
Z

My own work

Cognitive Systems

Technical University of Denmark

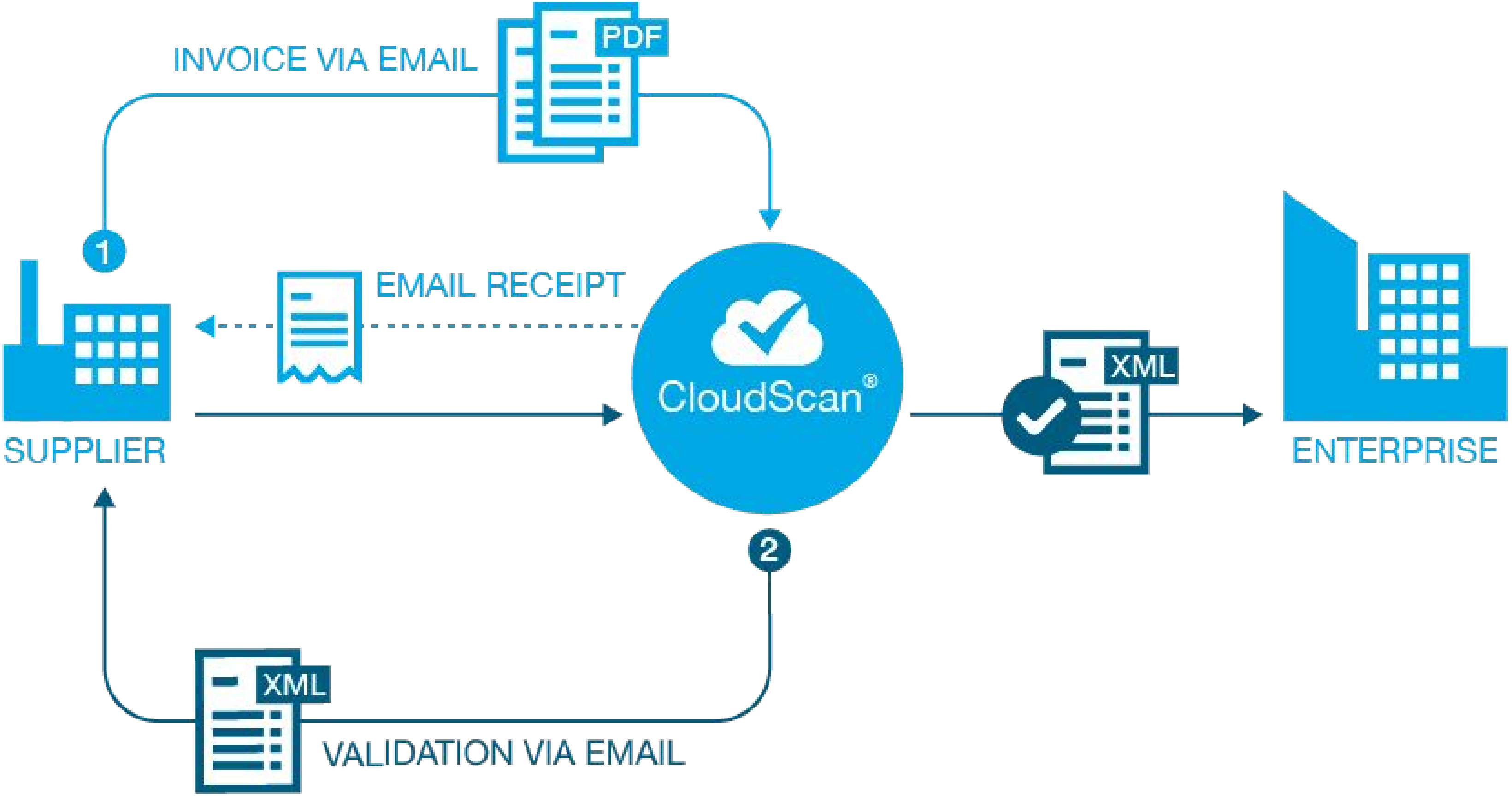
Ole Winther
Professor, PhD

Machine Learning Team

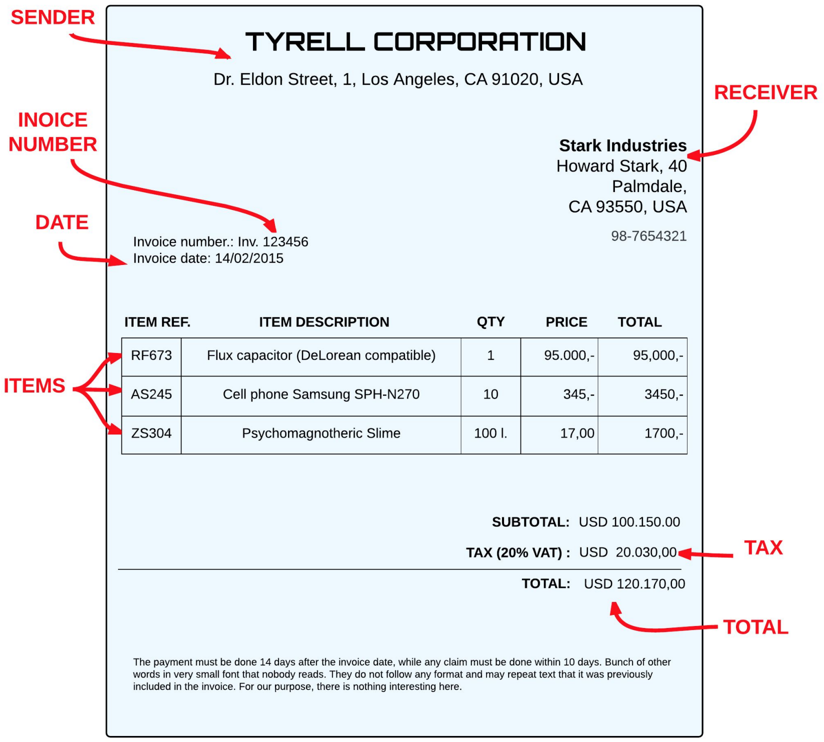
Tradeshift

Florian Laws
Team Lead, PhD

TRADESHIFT


INVOICE VIA EMAIL
PDF
1
EMAIL RECEIPT
CloudScan®
SUPPLIER
XML
ENTERPRISE
2
XML
VALIDATION VIA EMAIL

The actual data and problem



TYRELL CORPORATION

Dr. Eldon Street, 1, Los Angeles, CA 91020, USA

Stark Industries
Howard Stark, 40
Palmdale,
CA 93550, USA
98-7654321

Invoice number.: Inv. 123456
Invoice date: 14/02/2015

ITEM REF.	ITEM DESCRIPTION	QTY	PRICE	TOTAL
RF673	Flux capacitor (DeLorean compatible)	1	95.000,-	95,000,-
AS245	Cell phone Samsung SPH-N270	10	345,-	3450,-
ZS304	Psychomagnotheric Slime	100 l.	17,00	1700,-

SUBTOTAL: USD 100.150.00

TAX (20% VAT) : USD 20.030,00

TOTAL: USD 120.170,00

The payment must be done 14 days after the invoice date, while any claim must be done within 10 days. Bunch of other words in very small font that nobody reads. They do not follow any format and may repeat text that it was previously included in the invoice. For our purpose, there is nothing interesting here.

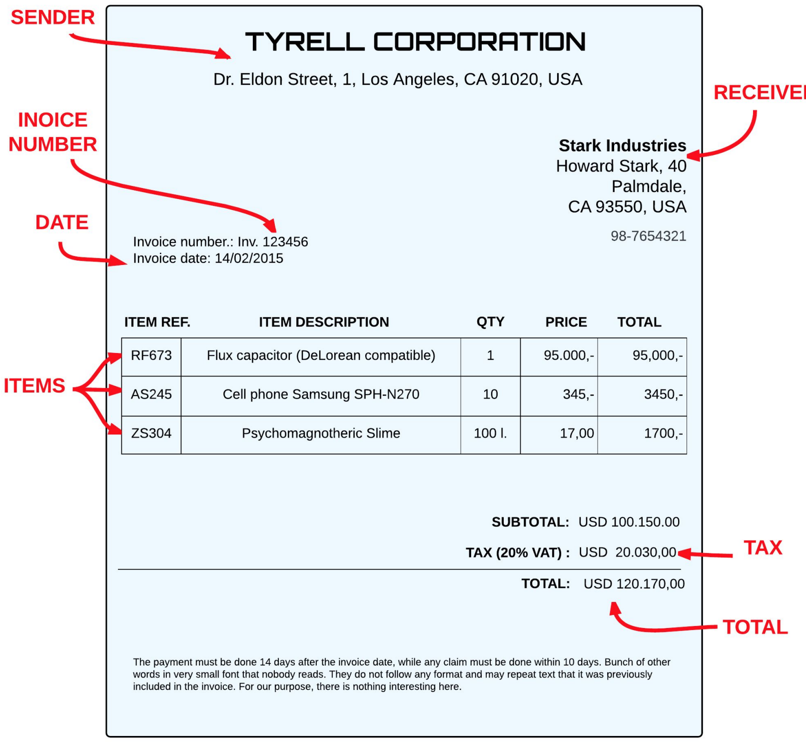
CloudScan


```xml
<?xml version="1.0" encoding="UTF-8"?>
<Invoice>

    <Number>123456</Number>
    <Date>2015-02-14</Date>

    <Sender>Tyrell Corporation</Sender>
    <Receiver>Stark Industries</Receiver>

    <TaxableAmount currency="USD">100150.00<TaxableAmount>
    <TaxPercent>20.00</TaxPercent>
    <TaxAmount currency="USD">20030.00</TaxAmount>
    <Total currency="USD">120170.00</Total>

</Invoice>
```

Interesting challenges

1. **Training data is PDF and XML pairs. No word annotations!**
2. Large handcrafted post-processing stage
3. Structured output (totals have to add up, etc.)
4. Modelling word context
5. Using image features

Missing word annotations



CloudScan


```xml
<?xml version="1.0" encoding="UTF-8"?>
<Invoice>

    <Number>123456</Number>
    <Date>2015-02-14</Date>

    <Sender>Tyrell Corporation</Sender>
    <Receiver>Stark Industries</Receiver>

    <TaxableAmount currency="USD">100150.00<TaxableAmount>
    <TaxPercent>20.00</TaxPercent>
    <TaxAmount currency="USD">20030.00</TaxAmount>
    <Total currency="USD">120170.00</Total>

</Invoice>
```

Addressing the lack of word annotations

‘End-to-End Information Extraction without Token-Level Supervision’

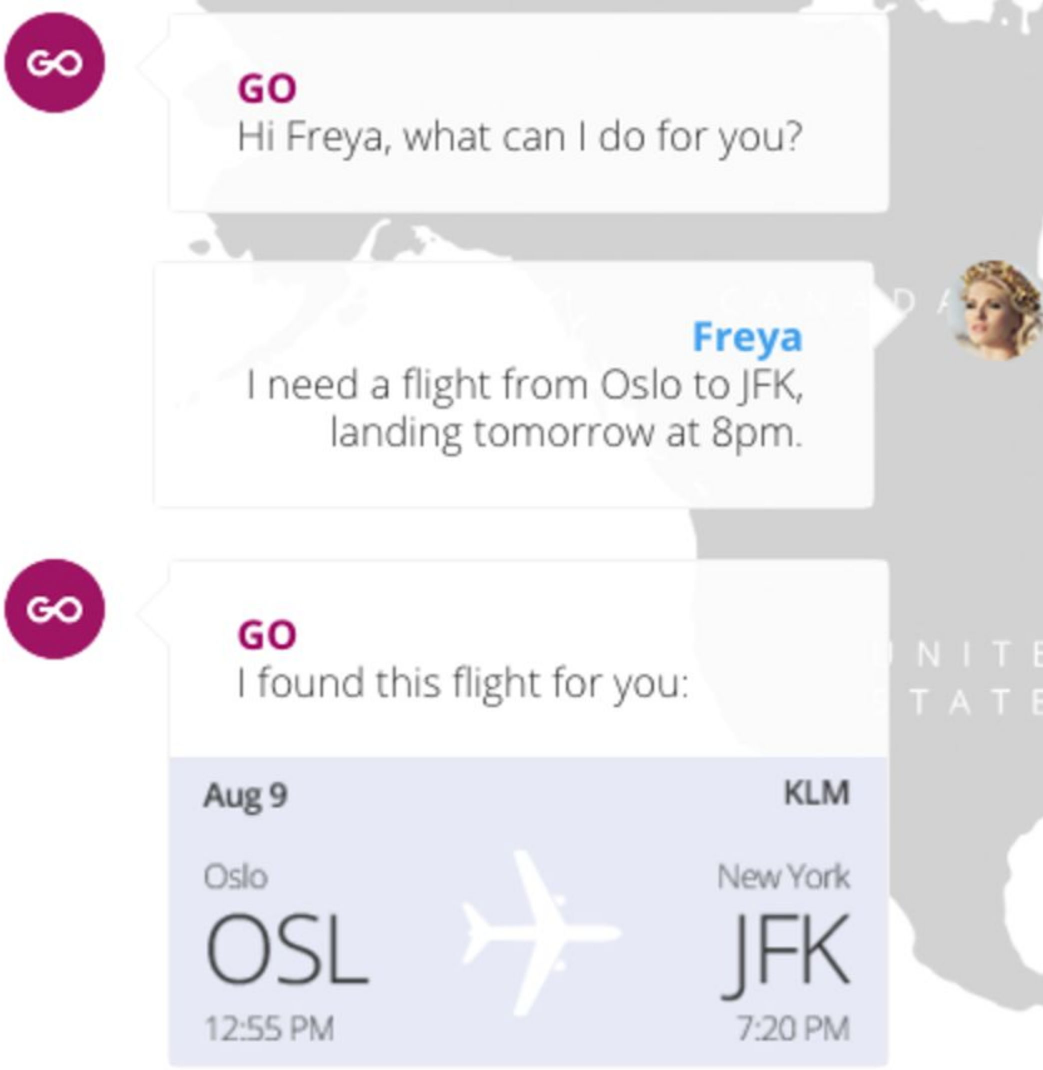
Rasmus Berg Palm, Dirk Hovy, Ole Winther, Florian Laws

https://arxiv.org/abs/1707.04913

Let’s create a travel concierge app

Takes natural language input.

Proposes flights.

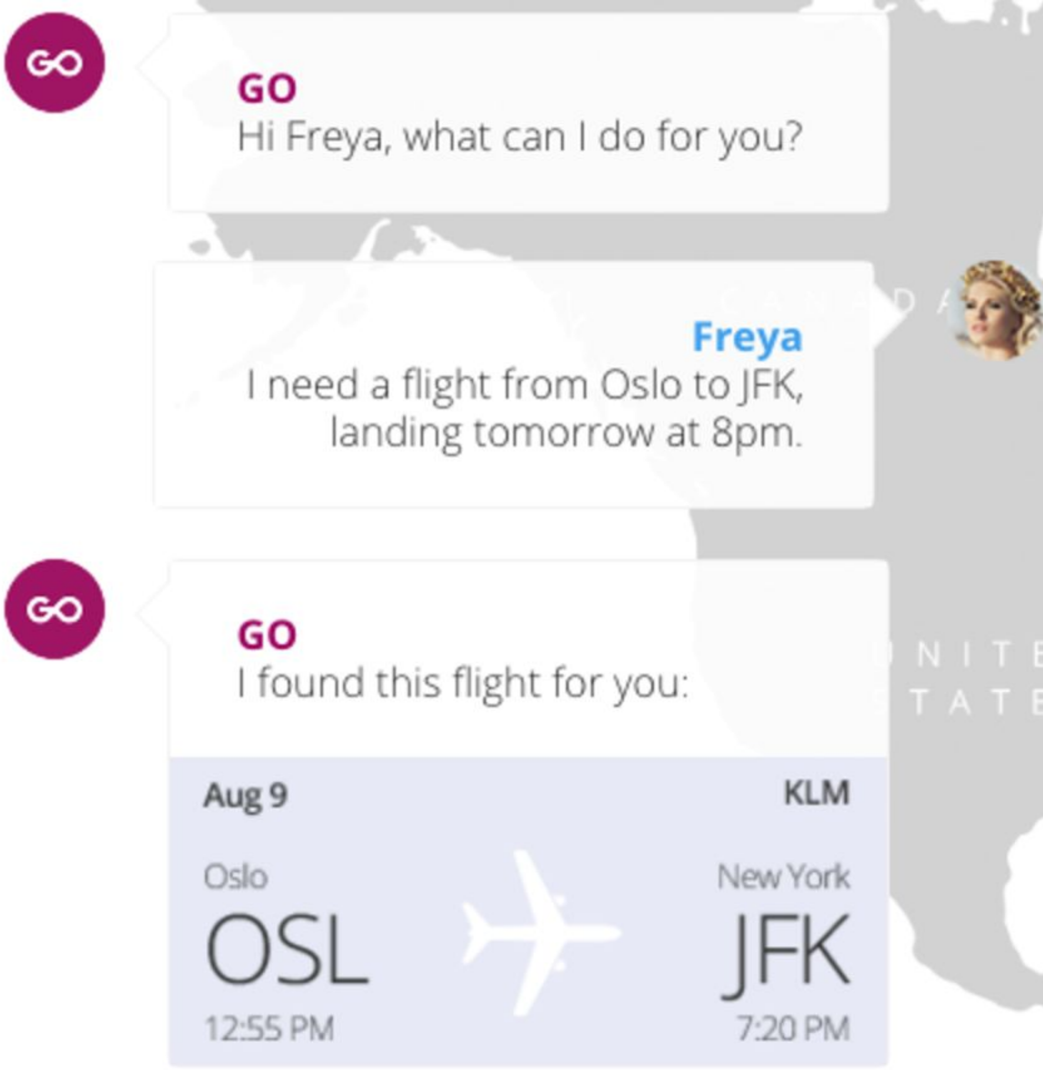


Behind the covers..

We use a flight search engine

The search engine accepts a fixed set of fields, e.g. "`from`", "`to`", "`day`", etc.

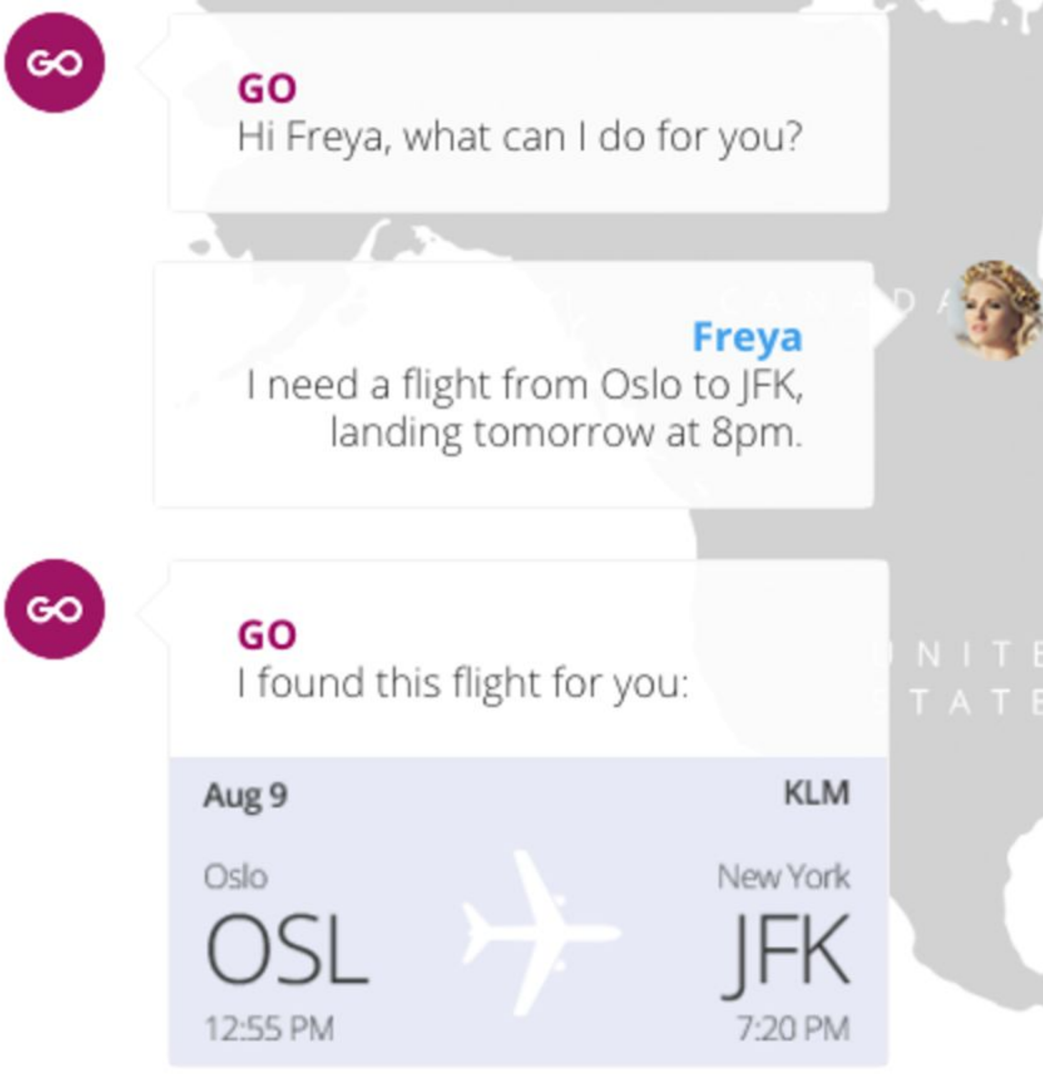
We return the top-1 hit



Behind the covers..

We hire human operators to extract the values for the search engine fields.

This labor is tedious for the operators and costly for us



So let’s automate this extraction of information

Token level

Token	Label
cheapest	`B-PRICE`
airfare	`O`
from	`O`
tacoma	`B-FROM`
to	`O`
st.	`B-TO`
louis	`I-TO`

Token level

Token	Label
cheapest	B-PRICE
airfare	O
from	O
tacoma	B-FROM
to	O
st.	B-TO
louis	I-TO

Token level

Token	Label
cheapest	B-PRICE
airfare	O
from	O
tacoma	B-FROM
to	O
st.	B-TO
louis	I-TO

Field level

Field	Value
FROM	tacoma
TO	st. louis
PRICE	cheapest
DAY	-
MONTH	-
YEAR	-
AIRLINE	-

Token level

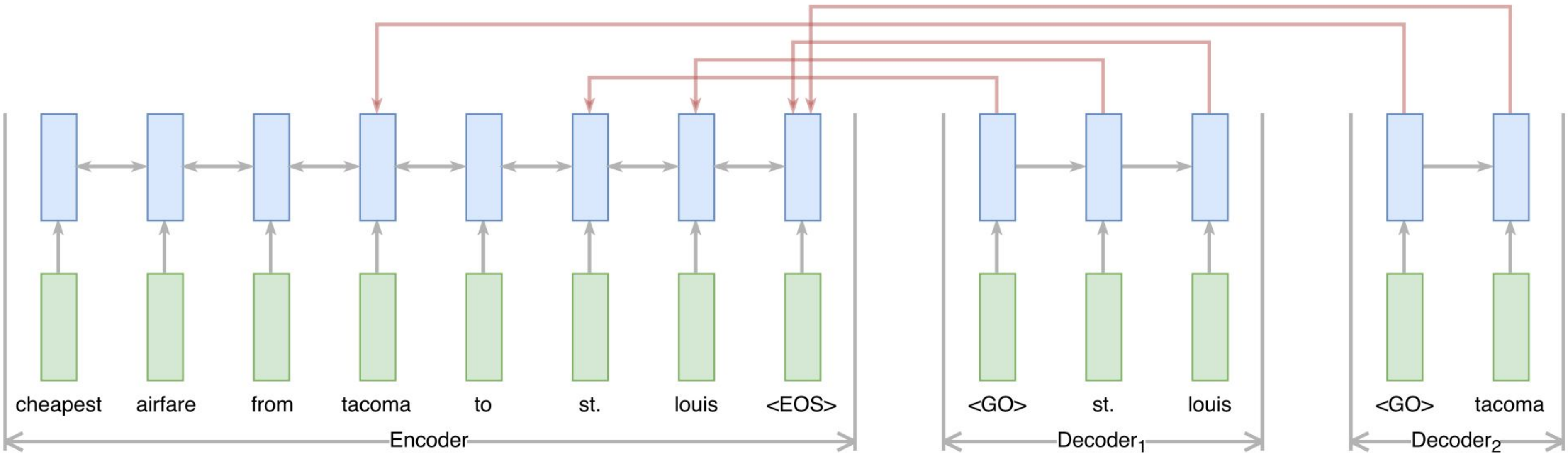
Token	Label
cheapest	`B-PRICE`
airfare	`O`
from	`O`
tacoma	`B-FROM`
to	`O`
st.	`B-TO`
louis	`I-TO`

Chunking

Field level

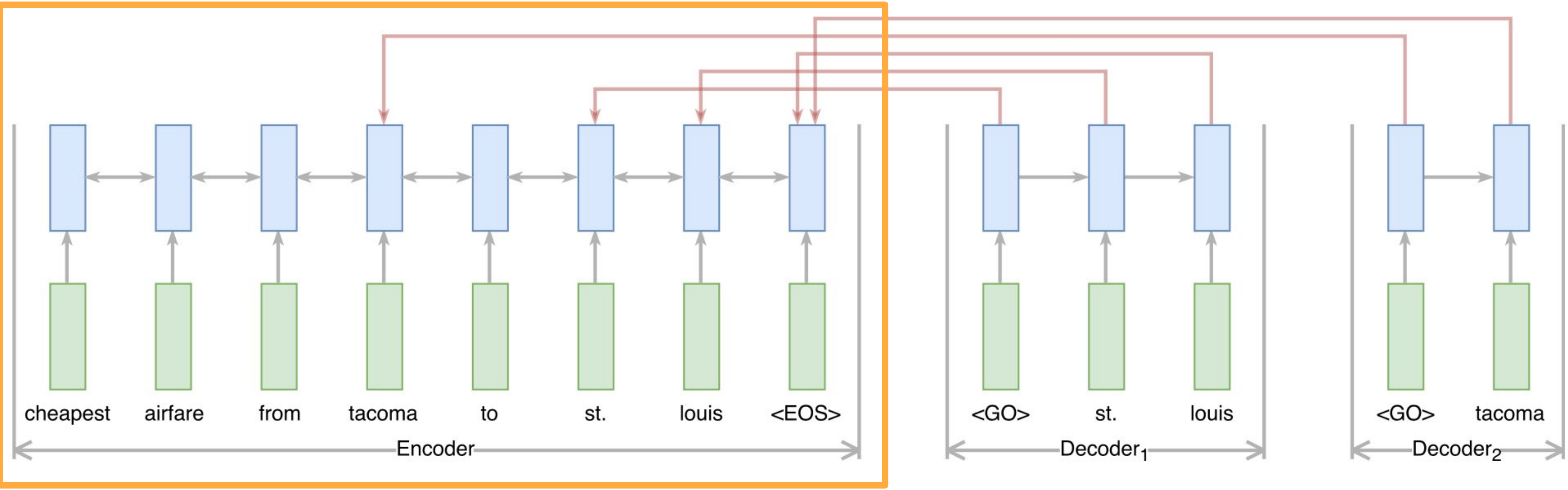
Field	Value
`FROM`	tacoma
`TO`	st. louis
`PRICE`	cheapest
`DAY`	-
`MONTH`	-
`YEAR`	-
`AIRLINE`	-

Our model



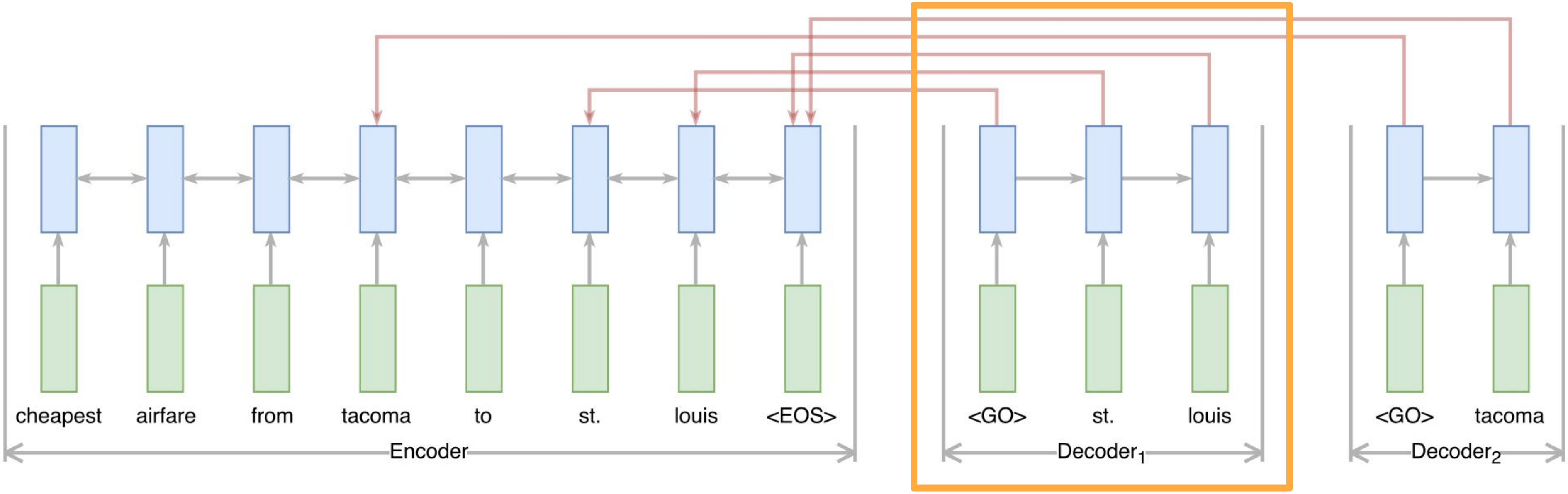
cheapest
airfare
from
tacoma
to
st.
louis
<EOS>
<GO>
st.
louis
<GO>
tacoma
Encoder
Decoder1
Decoder2

Our model

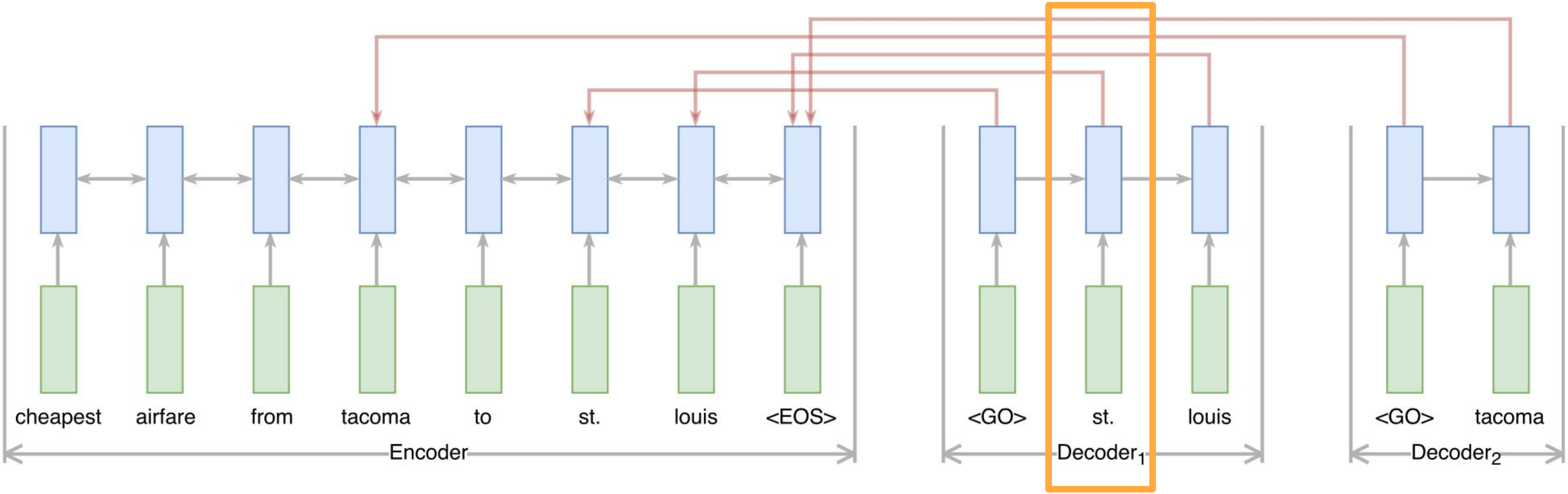


cheapest
airfare
from
tacoma
to
st.
louis
<EOS>
Encoder
<GO>
st.
louis
Decoder1
<GO>
tacoma
Decoder2

Our model

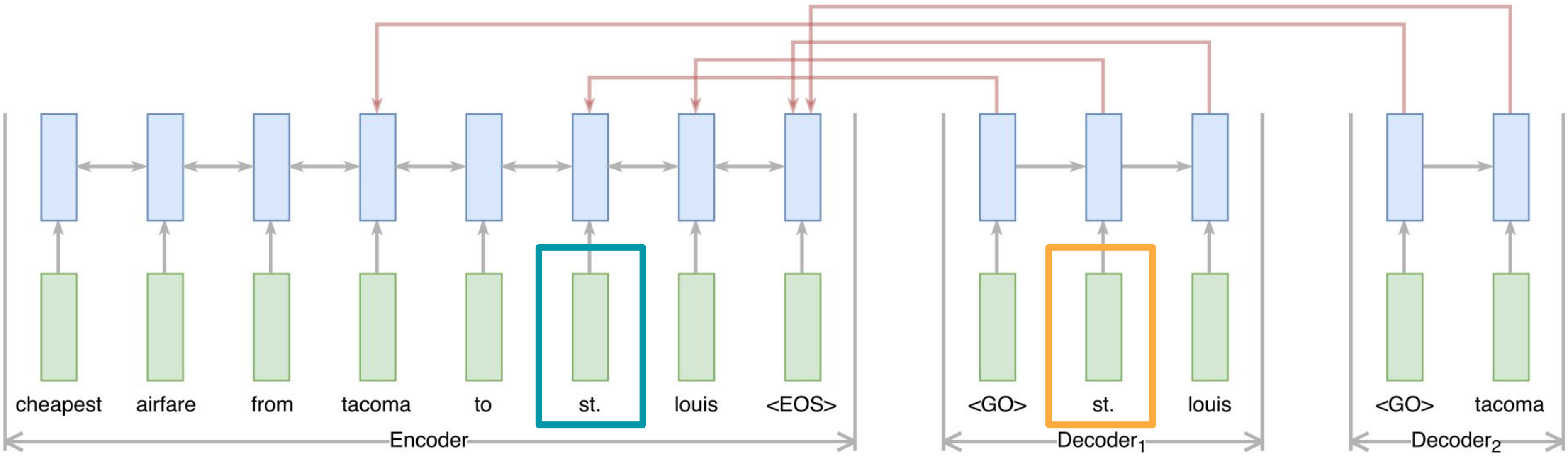

cheapest
airfare
from
tacoma
to
st.
louis
<EOS>
Encoder
<GO>
st.
louis
Decoder$_1$
<GO>
tacoma
Decoder$_2$

Our model



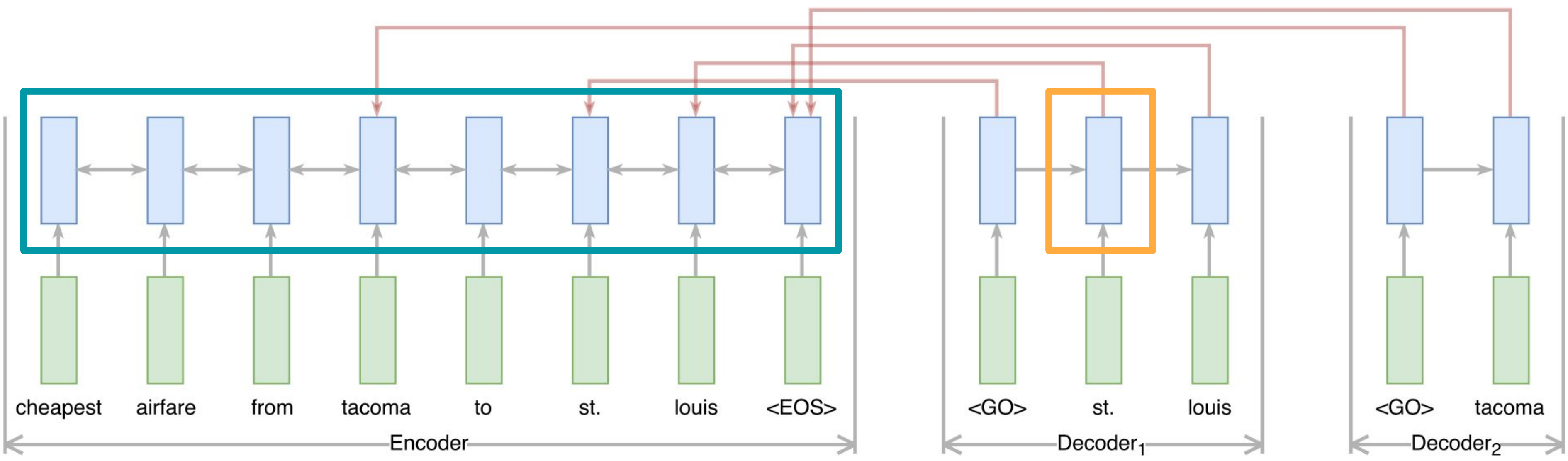
cheapest
airfare
from
tacoma
to
st.
louis
<EOS>
<GO>
st.
louis
<GO>
tacoma
Encoder
Decoder$_1$
Decoder$_2$

Our model

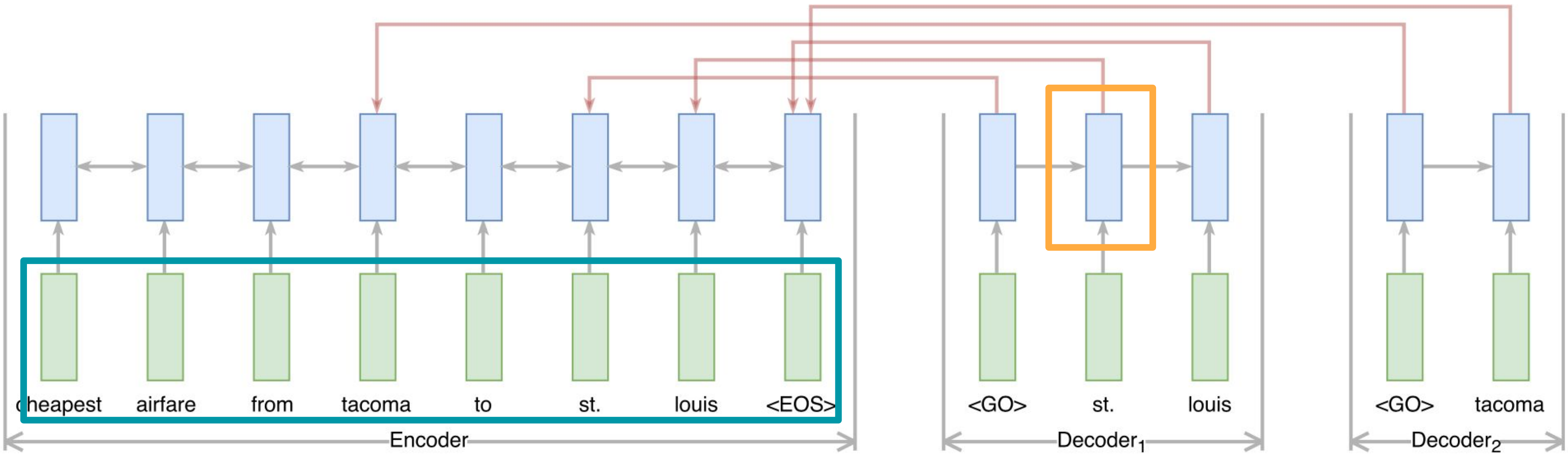


cheapest
airfare
from
tacoma
to
st.
louis
<EOS>
<GO>
st.
louis
<GO>
tacoma
Encoder
Decoder$_1$
Decoder$_2$

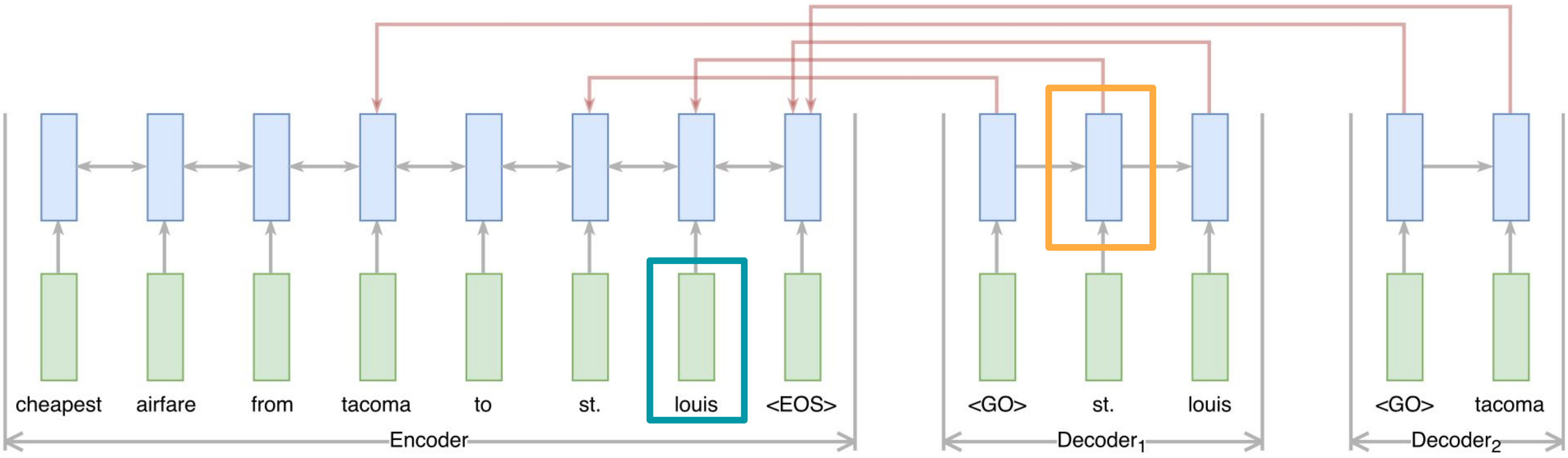
Our model


cheapest
airfare
from
tacoma
to
st.
louis
<EOS>
Encoder
<GO>
st.
louis
Decoder1
<GO>
tacoma
Decoder2

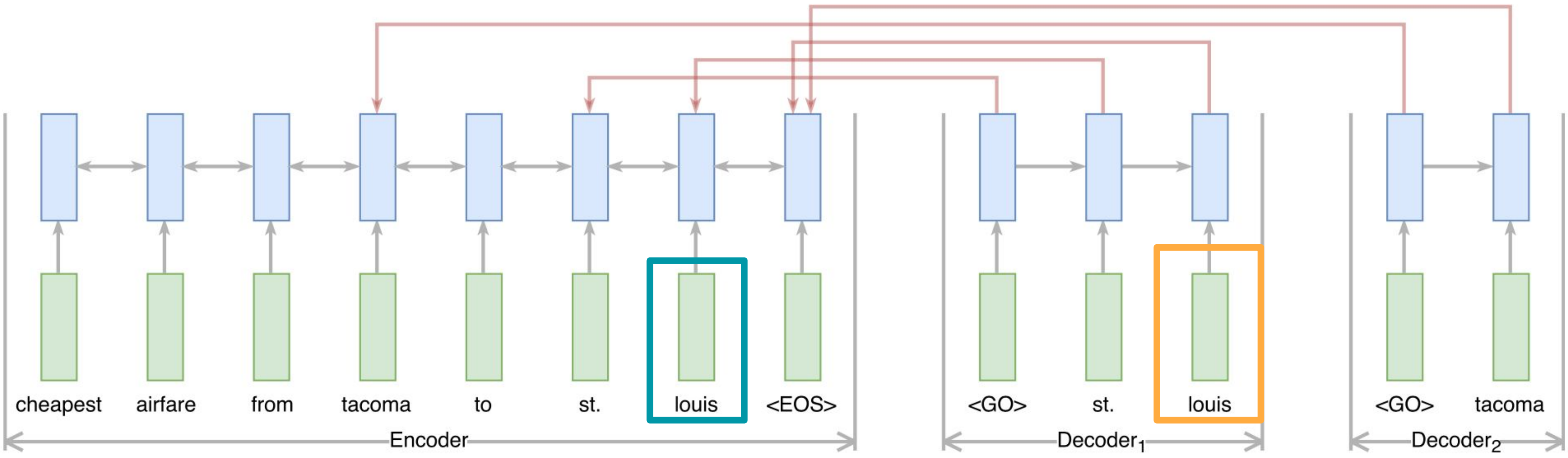
Our model


cheapest
airfare
from
tacoma
to
st.
louis
<EOS>
<GO>
st.
louis
<GO>
tacoma
Encoder
Decoder1
Decoder2

Our model

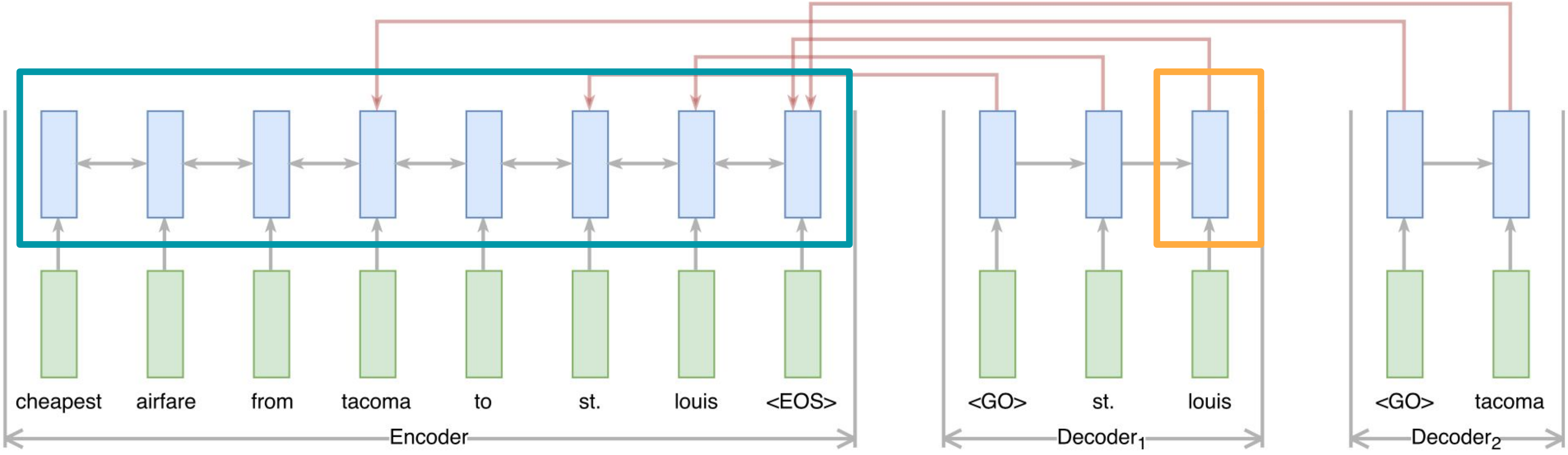

cheapest
airfare
from
tacoma
to
st.
louis
<EOS>
Encoder
<GO>
st.
louis
Decoder1
<GO>
tacoma
Decoder2

Our model



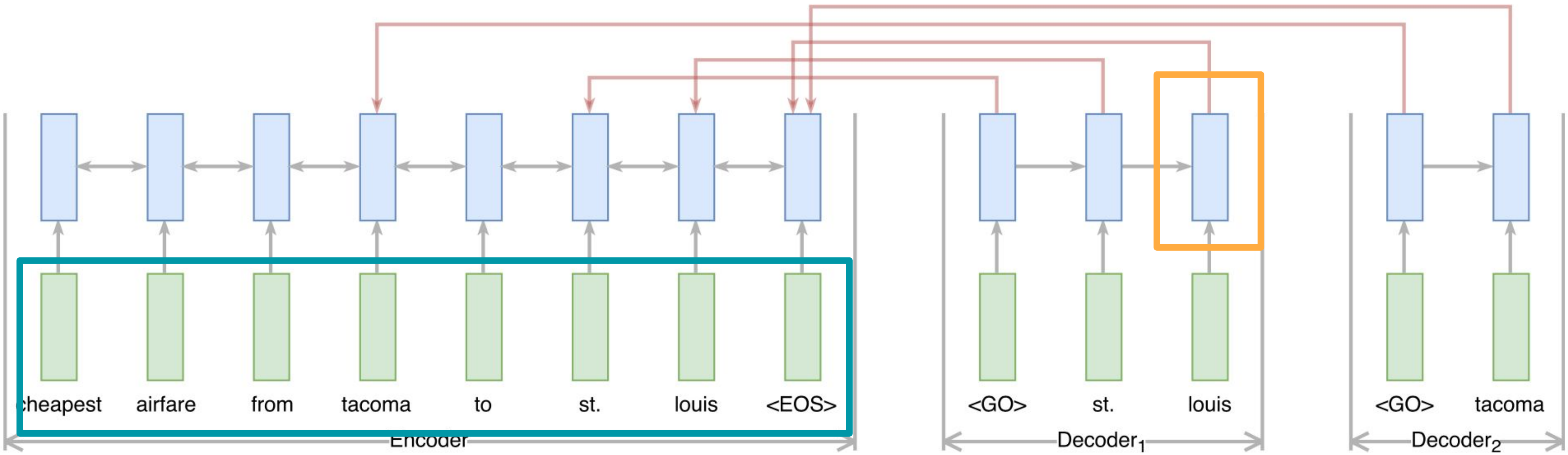
cheapest
airfare
from
tacoma
to
st.
louis
<EOS>
<GO>
st.
louis
<GO>
tacoma
Encoder
Decoder$_1$
Decoder$_2$

Our model

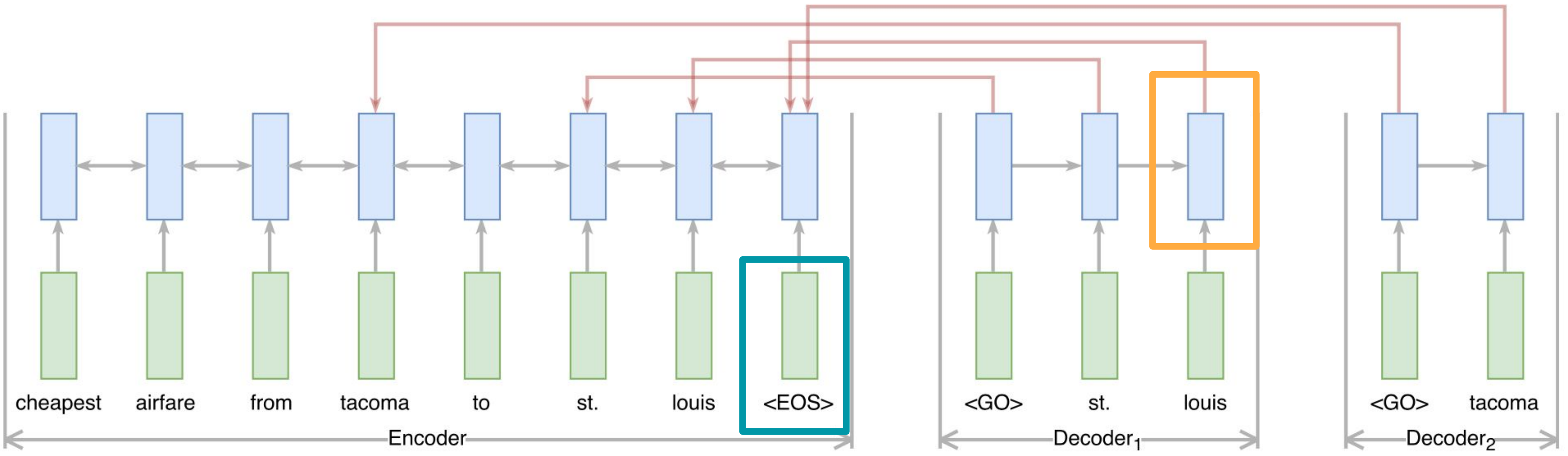


cheapest
airfare
from
tacoma
to
st.
louis
<EOS>
Encoder
<GO>
st.
louis
Decoder$_1$
<GO>
tacoma
Decoder$_2$

Our model

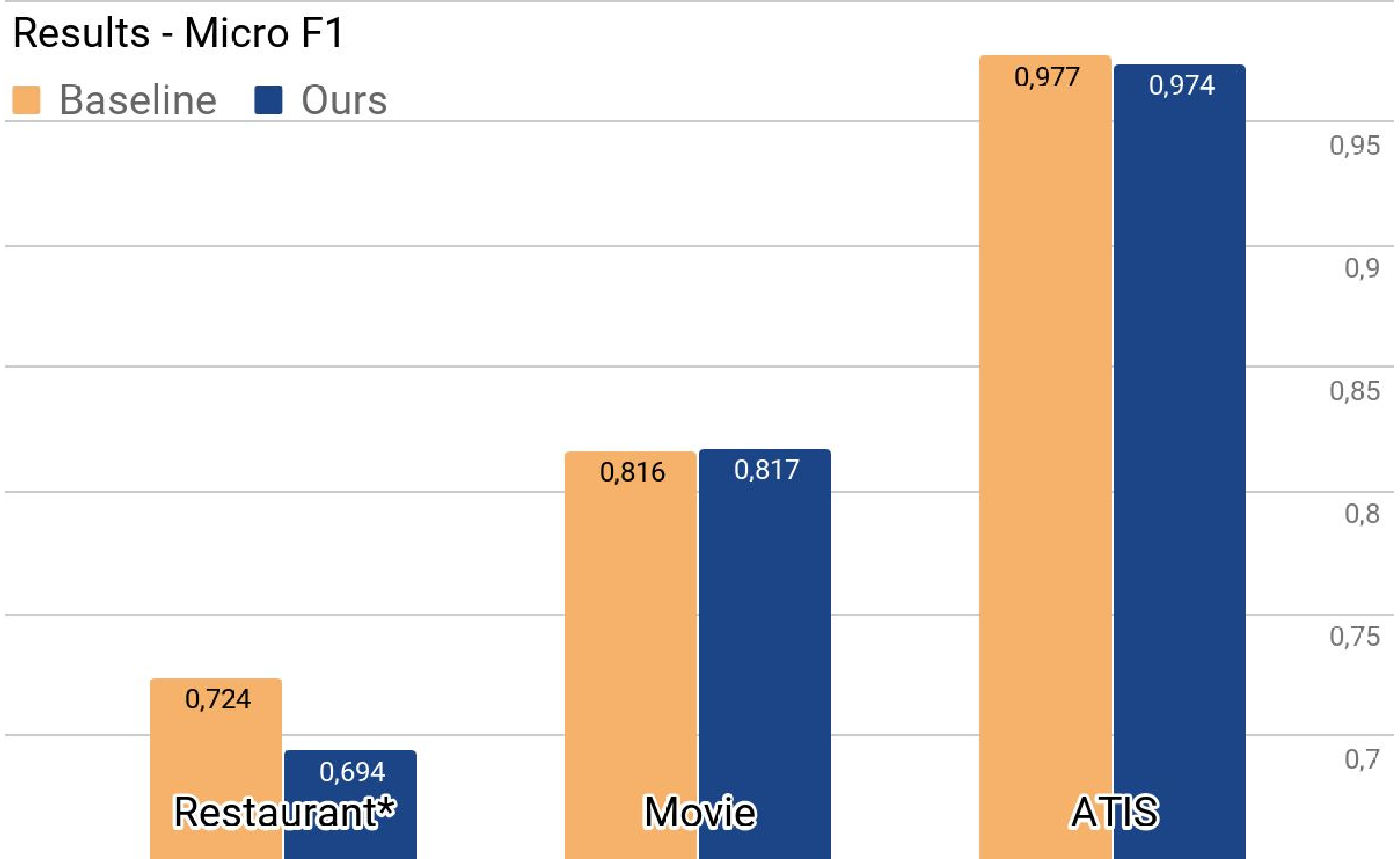

cheapest
airfare
from
tacoma
to
st.
louis
<EOS>
Encoder
<GO>
st.
louis
Decoder1
<GO>
tacoma
Decoder2

Our model


cheapest
airfare
from
tacoma
to
st.
louis
<EOS>
Encoder
<GO>
st.
louis
Decoder1
<GO>
tacoma
Decoder2

Code is available at

github.com/rasmusbergpalm/e2e-ie-release


Results - Micro F1
Baseline
Ours
0,977
0,974
0,95
0,9
0,85
0,816
0,817
0,8
0,75
0,724
0,7
0,694
Restaurant*
Movie
ATIS

There’s one major limitation

Normalization

‘17 Jan 2012’ → ‘2012-01-17’