Sequence to Sequence Models

Mausam

(Slides by Yoav Goldberg, Graham Neubig, Prabhakar Raghavan)

Neural Architectures

- Mapping from a sequence to a single decision.
  - with CNN or BiLSTM acceptor.
- Mapping from two sequences to a single decision.
  - with Siamese network.
- Mapping from a sequence to a sequence of same length.
  - with BiLSTM transducer

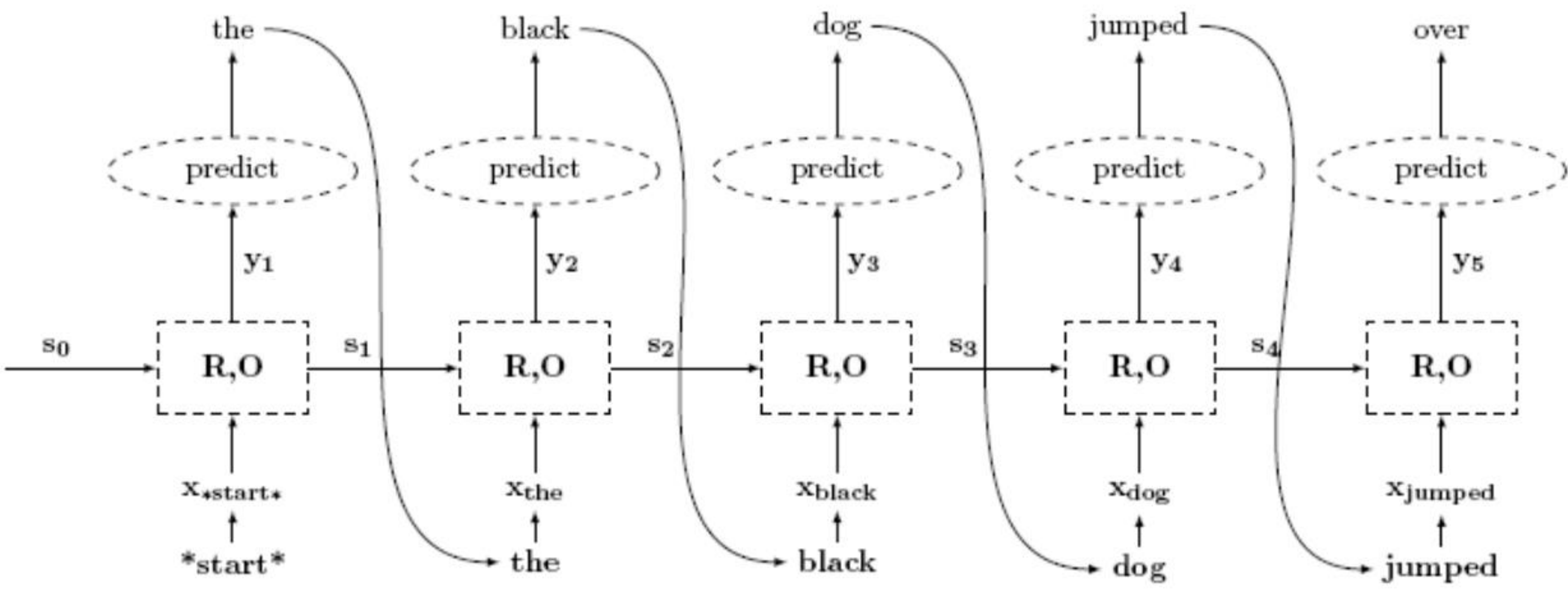
what do we do if the input and output sequences are of **different lengths**?

we already have an architecture from
0 to n mapping.

(sequence generation)

RNN Language Models

- *Training*: an RNN Transducer.
- *Generation*: the output of step i is input to step i+1.



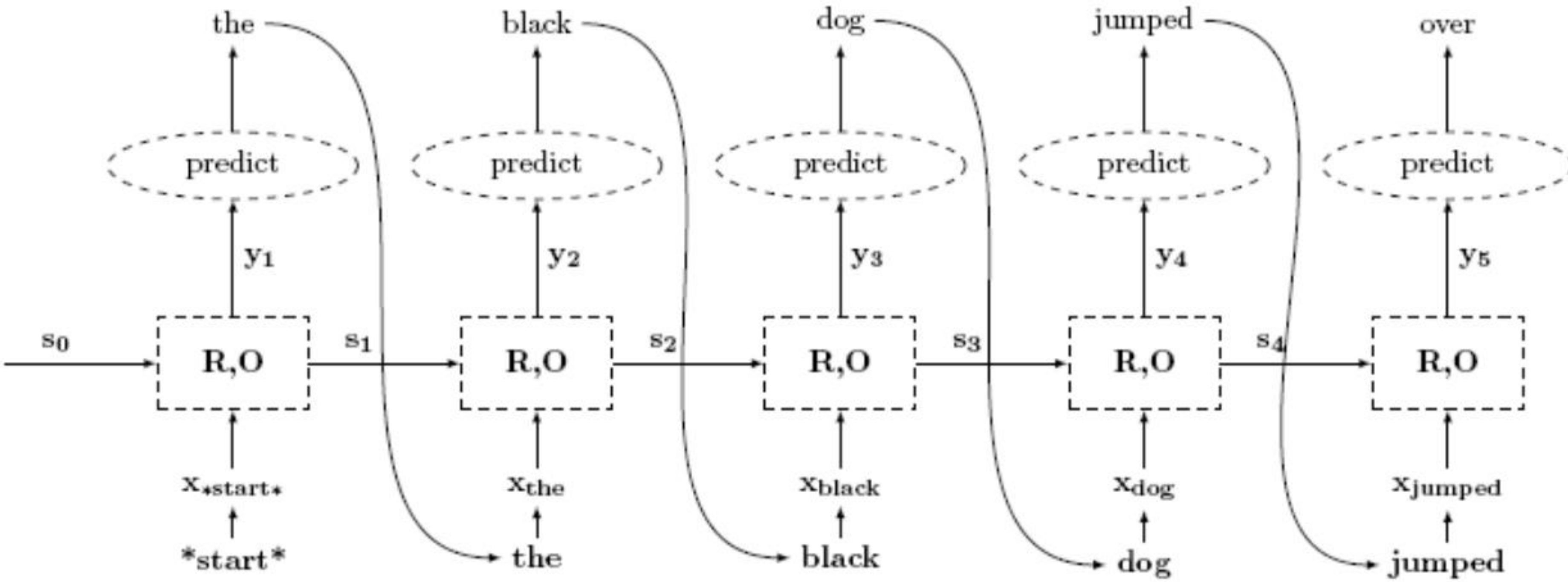
RNN Language Model for generation

- Define the probability distribution over the next item in a sequence (and hence the probability of a sequence).

$$P(w_{1:n}) = P(w_1)P(w_2 \mid w_1)P(w_3 \mid w_{1:2})P(w_4 \mid w_{1:3}) \dots P(w_n \mid w_{1:n-1})$$

$$P(w_1, ..., w_n) = \prod_{i=1}^{n} P(t_i = w_i | w_1, ..., w_{i-1})$$

RNN Language Models



$$p(t_{j+1} = k \mid \hat{t}_{1:j}) = f(\mathrm{RNN}(\hat{\mathbf{t}}_{\mathbf{1:j}}))$$

$$\hat{t}_j \sim p(t_j \mid \hat{t}_{1:j-1})$$

$$p(t_{j+1} = k \mid \hat{t}_{1:j}) = f(O(\mathbf{s_{j+1}}))$$

$$\mathbf{s_{j+1}} = R(\hat{\mathbf{t}}_{\mathbf{j}}, \mathbf{s_j})$$

$$\hat{t}_j \sim p(t_j \mid \hat{t}_{1:j-1})$$

RNN Language Models

Generating sentences is nice, but what if we want to add some additional conditioning contexts?

Conditioned Language Model

- Not just generate text, generate text according to some specification

Input X	Output Y (**Text**)	Task
Structured Data	NL Description	NL Generation
English	Japanese	Translation
Document	Short Description	Summarization
Utterance	Response	Response Generation
Image	Text	Image Captioning
Speech	Transcript	Speech Recognition

RNN Language Model for **Conditioned generation**

Let's add the condition variable to the equation.

$$P(T) = \prod_{i=1}^{I} P(\underbrace{t_i}_{\text{Next Word}} \mid \underbrace{t_1,\dots,t_{i-1}}_{\text{Context}})$$

$$P(T \mid C) = \prod_{j=1}^{J} P(t_i \mid \underbrace{c}_{\text{Added Context!}}, t_1,\dots,t_{i-1})$$

(a vector)

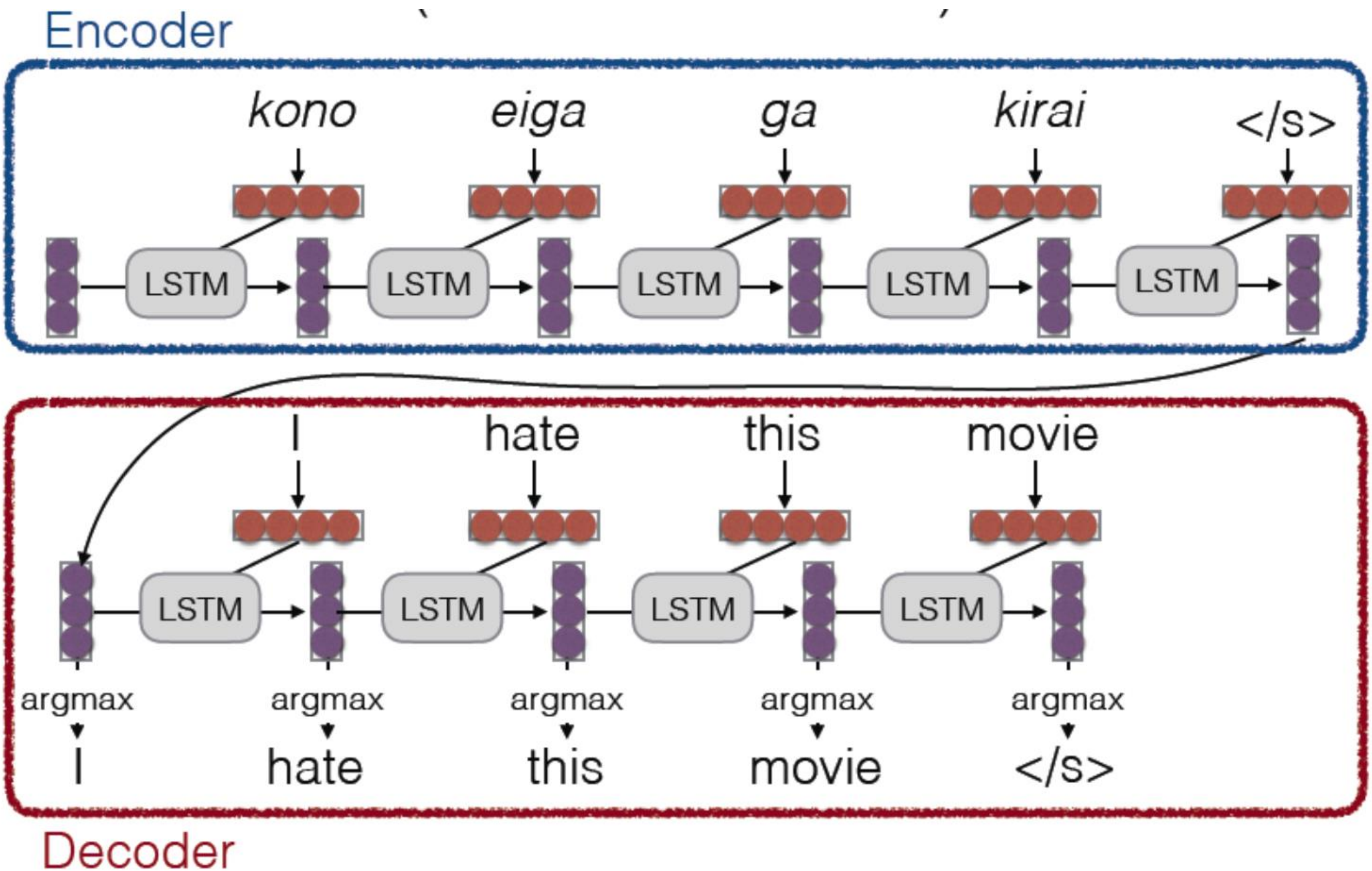
RNN Language Model for **Conditioned generation**

what if we want to condition on an entire sentence?

just encode it as a vector...

$$\mathbf{c} = \mathrm{RNN}^{\mathrm{enc}}(\mathbf{x_{1:n}})$$

A simple Sequence to Sequence conditioned generation

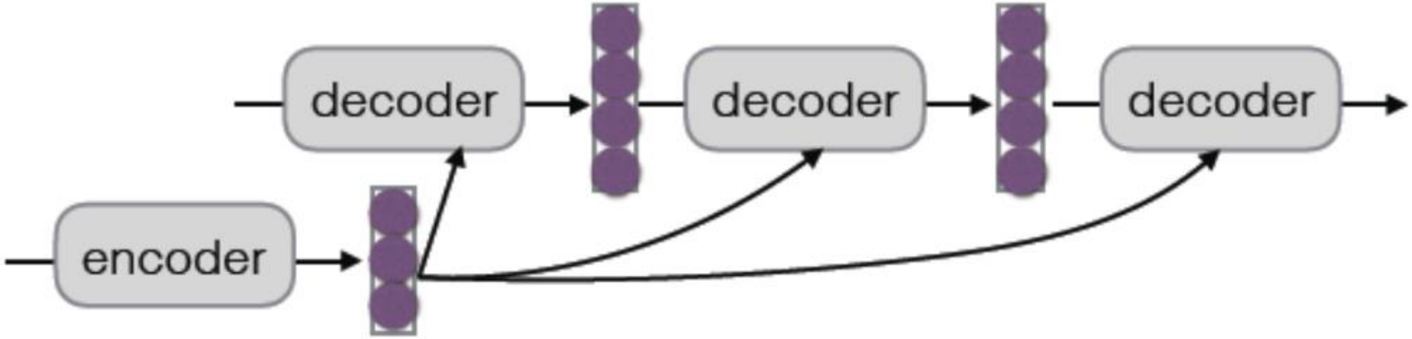


How to Pass Hidden State

- Initialize decoder w/ encoder (Sutskever et al. 2014)

- Transform (can be different dimensions)

- Input at every time step (Kalchbrenner & Blunsom 2013)



RNN Language Model for **Conditioned generation**

Let's add the condition variable to the equation.

$$p(t_{j+1} = k \mid \hat{t}_{1:j}, c) = f(\mathrm{RNN}(\mathbf{v_{1:j}}))$$

$$\mathbf{v_i} = [\hat{\mathbf{t}}_\mathbf{i}, \mathbf{c}]$$

$$\hat{t}_j \sim p(t_j \mid \hat{t}_{1:j-1}, c)$$

RNN Language Model for **Conditioned generation**

Let's add the condition variable to the equation.

$$p(t_{j+1} = k \mid \hat{t}_{1:j}, c) = f(\mathrm{RNN}(\mathbf{v_{1:j}}))$$

$$\mathbf{v_i} = [\hat{\mathbf{t}}_\mathbf{i}; \mathbf{c}]$$

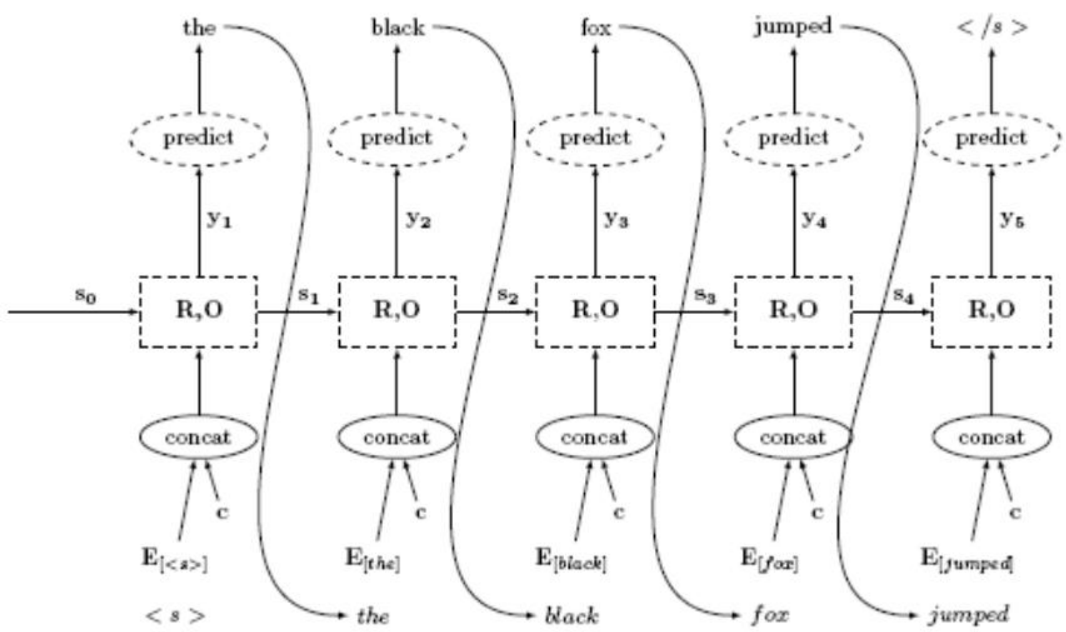
$$\hat{t}_j \sim p(t_j \mid \hat{t}_{1:j-1}, c)$$

$$p(t_{j+1} = k \mid \hat{t}_{1:j}, c) = f(O(\mathbf{s_{j+1}}))$$

$$\mathbf{s_{j+1}} = R(\mathbf{s_j}, [\hat{\mathbf{t}}_\mathbf{j}; \mathbf{c}])$$

$$\hat{t}_j \sim p(t_i \mid \hat{t}_{1:j-1}, c)$$

RNN Language Model for **Conditioned generation**



$$p(t_{j+1} = k \mid \hat{t}_{1:j}, c) = f(O(\mathbf{s_{j+1}}))$$

$$\mathbf{s_{j+1}} = R(\mathbf{s_j}, [\hat{\mathbf{t}}_\mathbf{j}; \mathbf{c}])$$

$$\hat{t}_j \sim p(t_i \mid \hat{t}_{1:j-1}, c)$$

RNN Language Model for **Conditioned generation**

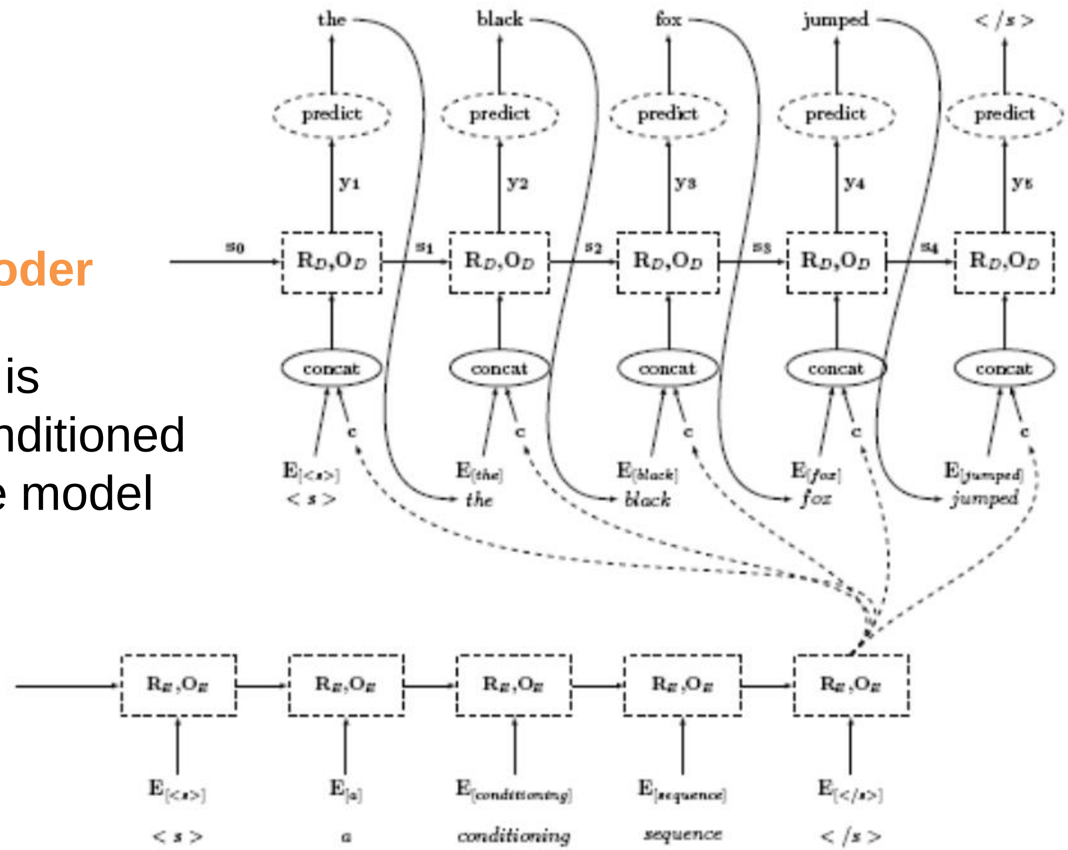
what if we want to condition on an entire sentence?

Sequence to Sequence conditioned generation

This is also called
 "Encoder Decoder"
architecture.

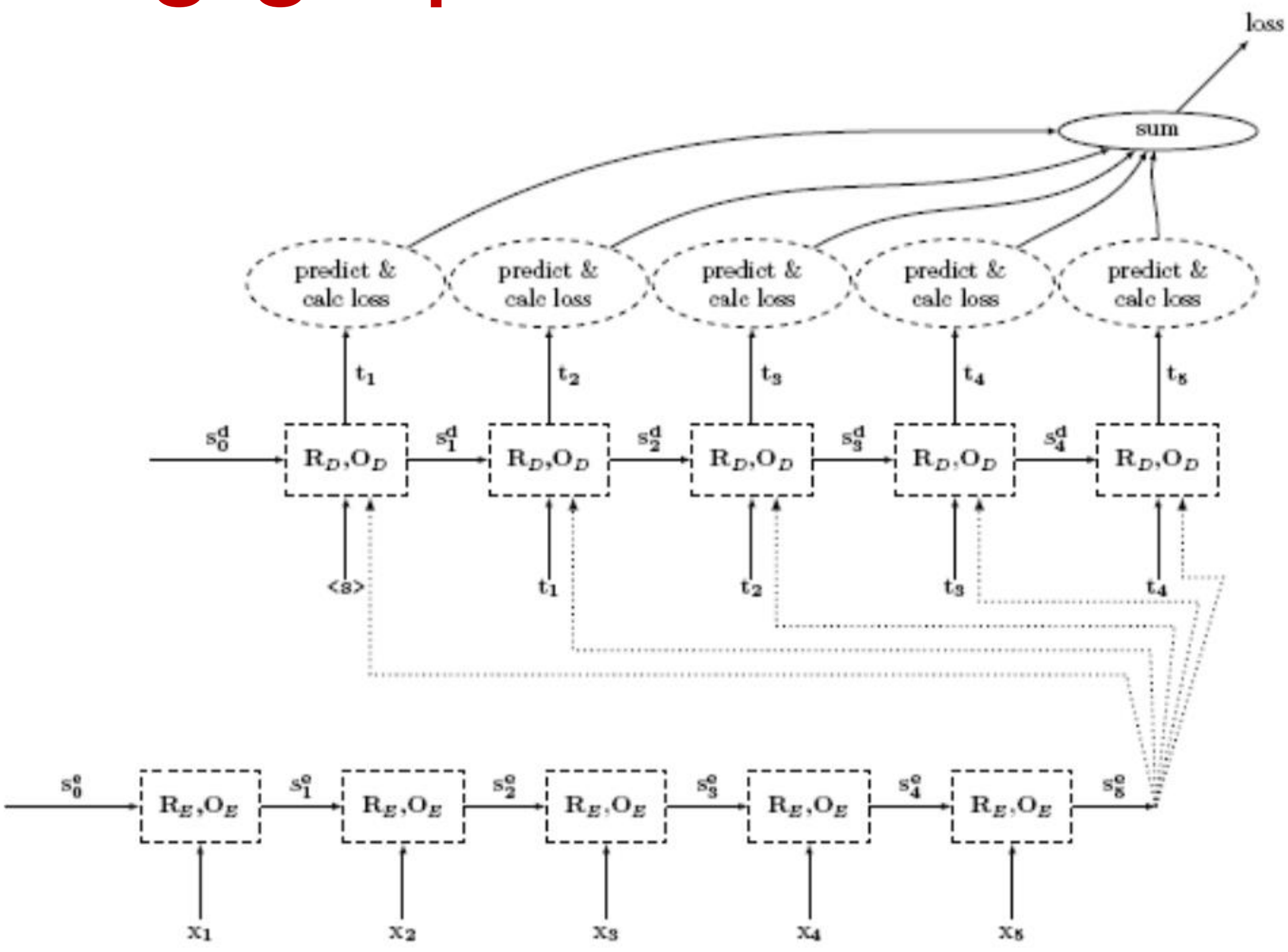
Decoder

Decoder is
just a conditioned
language model



Encoder

Sequence to Sequence training graph



The Generation Problem

We have a probability model, how do we use it to generate a sentence?

Two methods:

- **Sampling:** Try to generate a *random* sentence according to the probability distribution.
- **Argmax:** Try to generate the sentence with the *highest* probability.

Ancestral Sampling

Randomly generate words one-by-one.

$$\begin{array}{|l|}\hline \text{while } y_{j-1} \mathrel{!}= \text{“</s>”:} \\ \quad y_j \sim P(y_j \mid X, y_1, \ldots, y_{j-1}) \\ \hline \end{array}$$

An **exact method** for sampling from P(X), no further work needed.

Greedy Search

One by one, pick the single highest-probability word

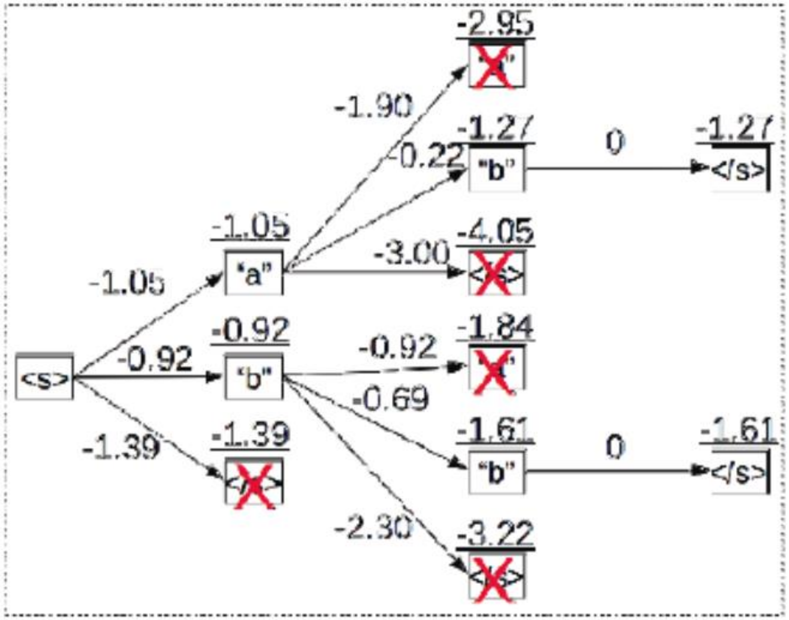
```
while y_{j-1} != "</s>":
  y_j = argmax P(y_j | X, y_1, …, y_{j-1})
```

Not exact, real problems:

- Will often generate the "easy" words first
- Will prefer multiple common words to one rare word

Beam Search

Instead of picking one high-probability word, maintain several paths

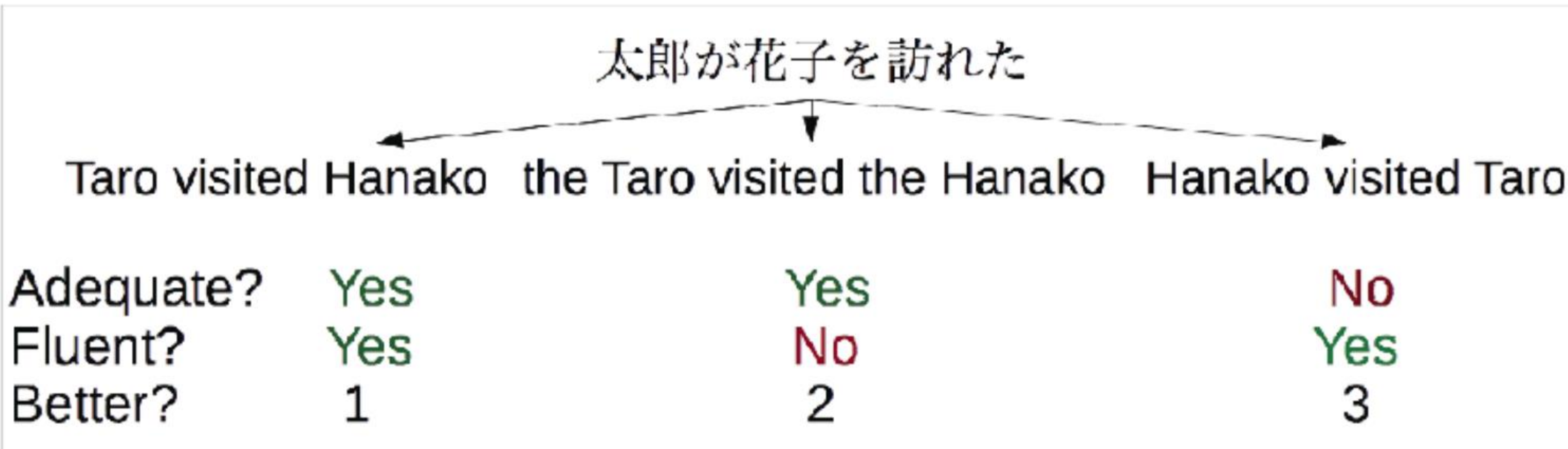


How to evaluate?

- Basic Paradigm

- Use parallel test set
- Use system to generate translations
- Compare target translations w/ reference

Human Evaluation



- Final goal, but slow, expensive, and sometimes inconsistent

BLEU

- Works by comparing n-gram overlap w/ reference

Reference: Taro visited Hanako

System: the Taro visited the Hanako

1-gram: 3/5
2-gram: 1/4

Brevity: min(1, |System|/|Reference|) = min(1, 5/3)   brevity penalty = 1.0

BLEU-2 = $(3/5*1/4)^{1/2}$ * 1.0
= 0.387

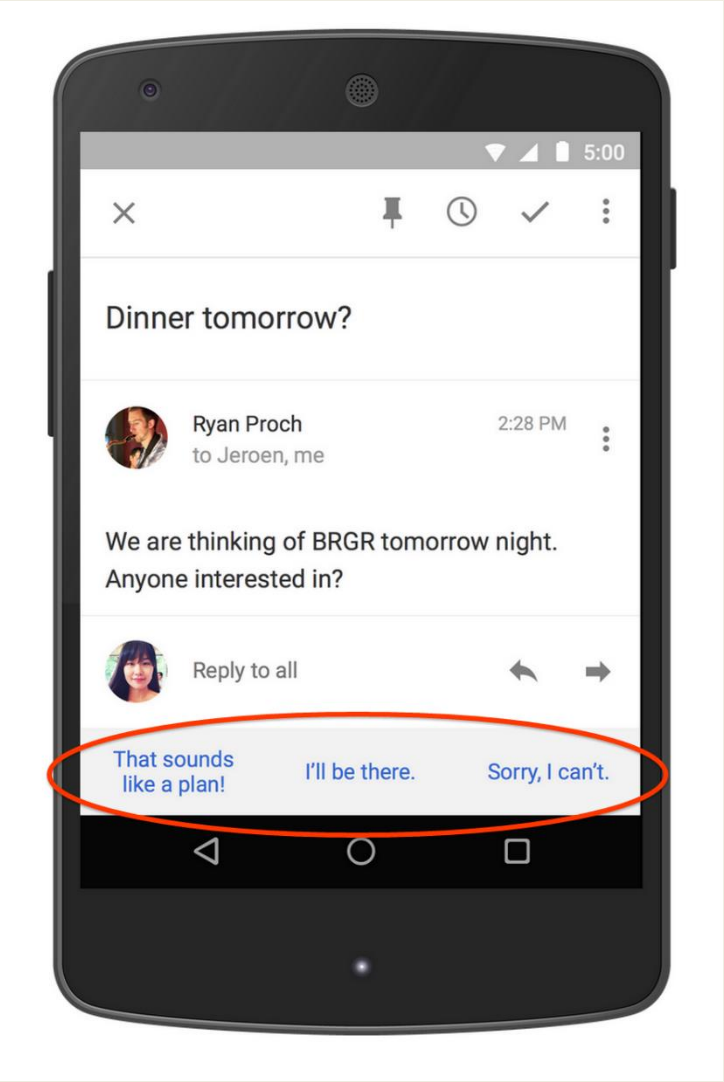
- **Pros:** Easy to use, good for measuring system improvement
- **Cons:** Often doesn't match human eval, bad for comparing very different systems

METEOR

- Like BLEU in overall principle, with many other tricks: consider paraphrases, reordering, and function word/content word difference
- **Pros:** Generally significantly better than BLEU, esp. for high-resource languages
- **Cons:** Requires extra resources for new languages (although these can be made automatically), and more complicated

Perplexity

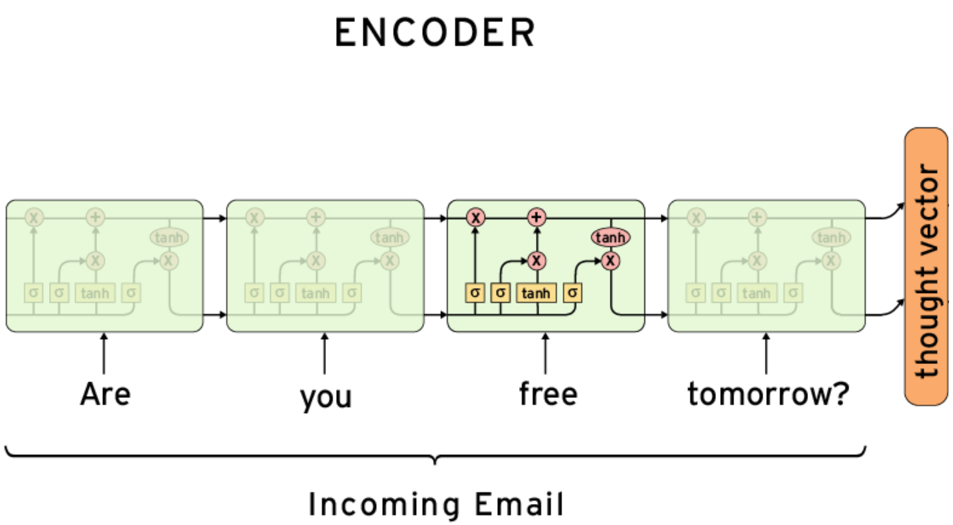
- Calculate the perplexity of the words in the held-out set *without* doing generation
- **Pros:** Naturally solves multiple-reference problem!
- **Cons:** Doesn't consider decoding or actually generating output. May be reasonable for problems with lots of ambiguity.



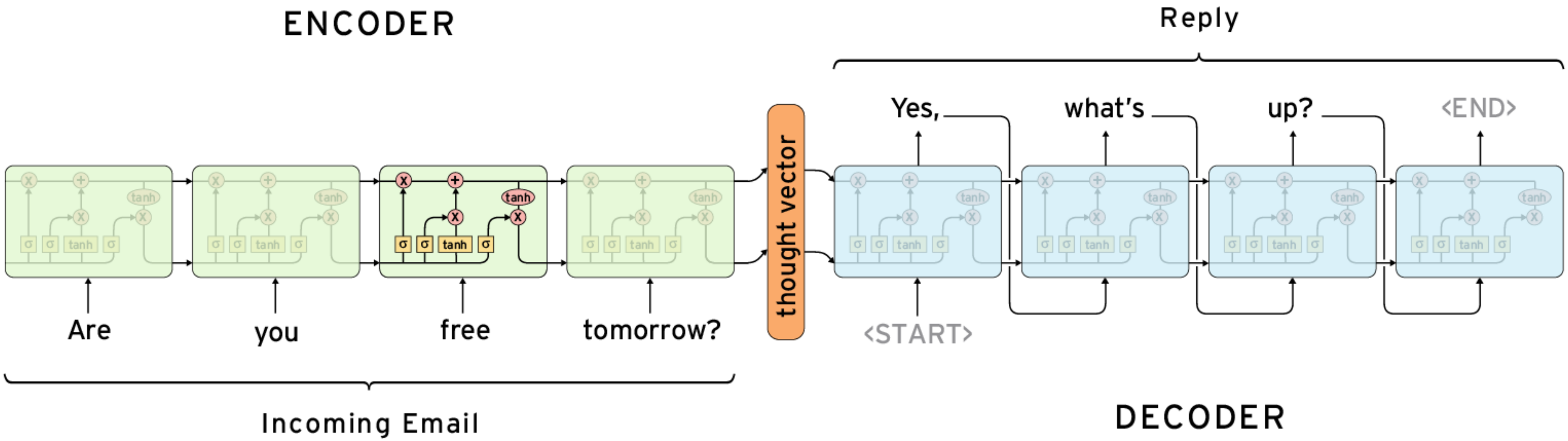
Case Study: Smart Reply in Gmail

Preprocessing an incoming email

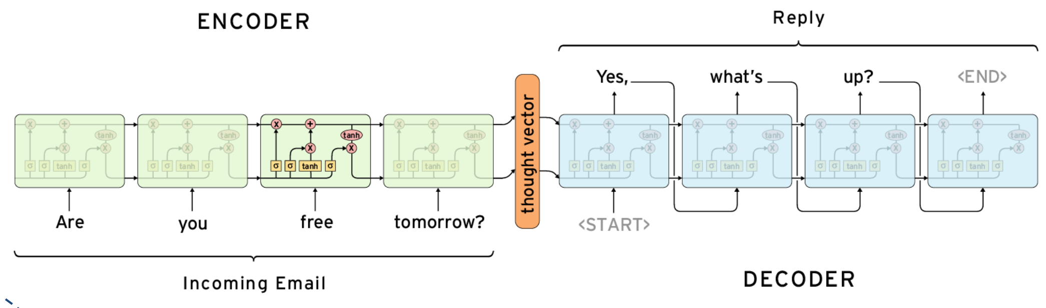
- Language detection
  - Currently handle English, Portuguese, Spanish … a few more languages are in preparation
- Tokenization of subject and message body
- Sentence segmentation
- Normalization of infrequent words and entities – replaced by special tokens
- Removal of quoted and forward email portions
- Removal of greeting and closing phrases (“Hi John”,… “Regards, Mary”)


ENCODER
tanh
σ
tanh
thought vector
Are
you
free
tomorrow?
Incoming Email

LSTM translation



Vinyals & Le, 2015



Pick the best suggestions (LSTM)

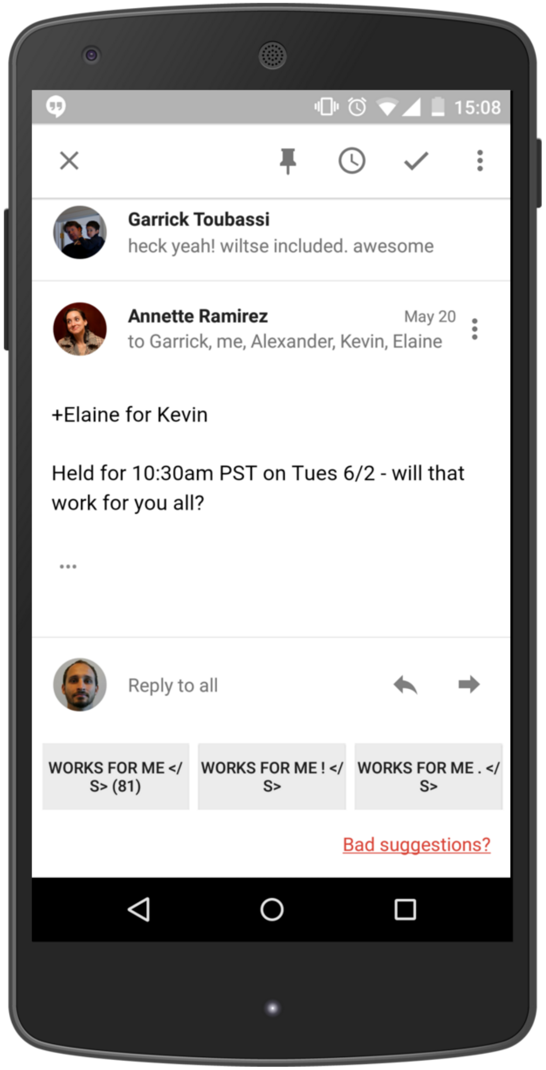
Is it worth it?

- Precision/accuracy - how well can we guess good replies?
  - Self-reinforcing behavior - often machine predictions are “good enough”
  - Machines learn from humans, and vice versa
- Coverage - do most emails have simple, predictable responses?
  - Do a small number of utterances cover a large fraction of responses?
  - Language/cultural variations? Linguistic entropy

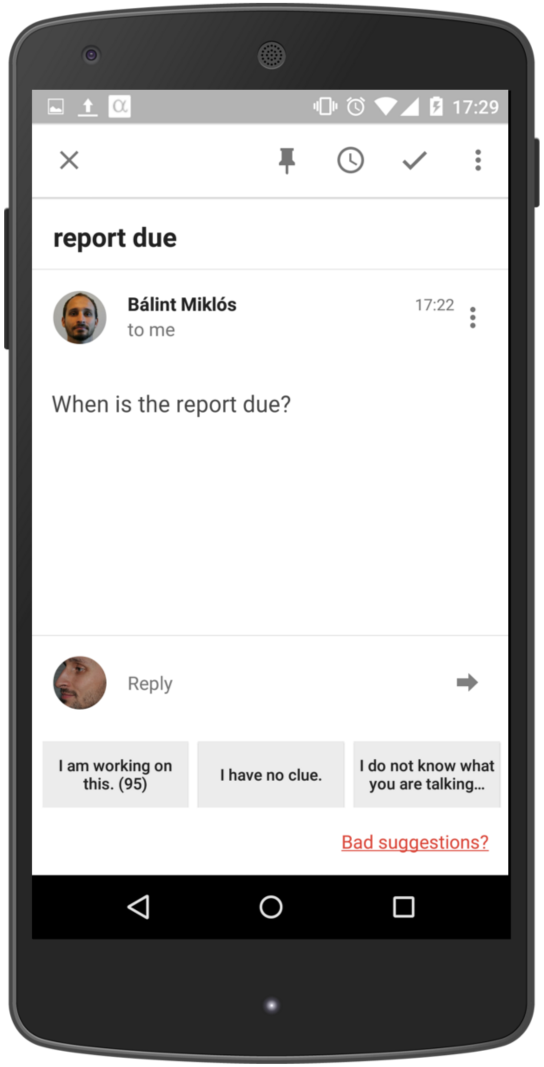
Metric

- What fraction of the time do users select a suggested reply?
  - How many replies do we suggest? 3
  - Constraint based on user interface, but also users' ability to quickly process choices
- We get a boost from allowing users to edit responses before sending
  - In early studies, users were nervous that choosing a response would instantly send
  - Careful tuning of this UI gave us bigger gains than a lot of ML tuning

Some early observations



Some early observations



A scoring algorithm doesn't make a product

- Semantic variation: doesn't help if all three suggestions say the same thing …
  - Can't simply take the 3 highest scoring suggestions
- The "I love you" problem
  - Some responses are unhelpful and a human can say them, but not a computer …*
  - A lot of responses in the training corpus have "I love you"
  - In many cases this isn't appropriate
  - "Family friendliness"
- Sensitivity
  - There are many incoming emails where you don't want the computer to guess replies - Bad news, etc
- * in general our expectations of "working" AI are higher than of humans

Michael Gadberry @michaelgadberry · 13h
Google Inbox's automated suggested replies are mind-blowingly awesome and accurate. #CheckOutDatStuff #GoogleInbox @inboxbygmail

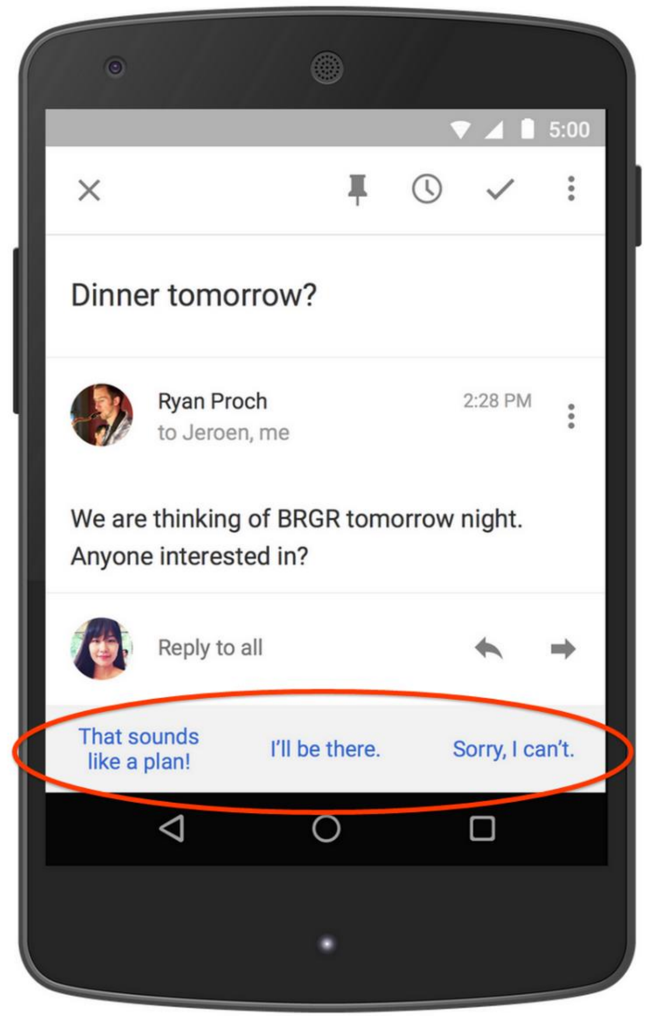
Simon Dingle @SimonDingle · Nov 12
It's like @inboxbygmail has telepathy with its automated responses.

Tatiana King Jones @TatianaKing · Nov 12
The new @inboxbygmail auto response choices have been pretty good so far. Have been using them maybe 50% of the time.

>10%

of Gmail responses are Smart Replies.

(Users accept computer-generated replies.)



Encoder-Decoder with different modalities

The encoded conditioning context need not be text, or even a sequence.

Encoder-Decoder
with different modalities

Show and Tell: A Neural Image Caption Generator

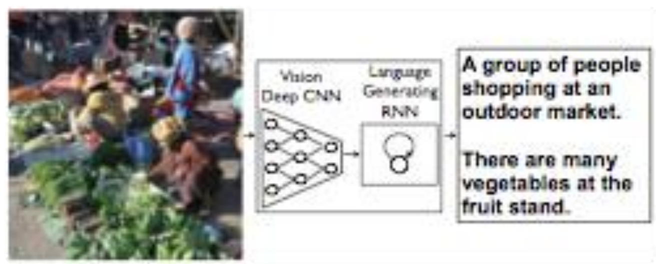
Oriol Vinyals
Google
vinyals@google.com

Alexander Toshev
Google
toshev@google.com

Samy Bengio
Google
bengio@google.com

Dumitru Erhan
Google
dumitru@google.com

- Encode: **image** to vector.
  Decode: a sentence describing the image.



This sort-of works.
In my opinion, looks more impressive than really is.

I think it's a man in a business suit standing on a bench.

I am not really confident, but I think it's a man standing on a beach near the water.

I think it's a group of people sitting in front of a crowd.

I am not really confident, but I think it's a close up of a sheep.

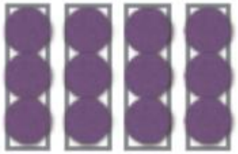
Sentence Representation

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

But what if we could use multiple vectors, based on the length of the sentence.

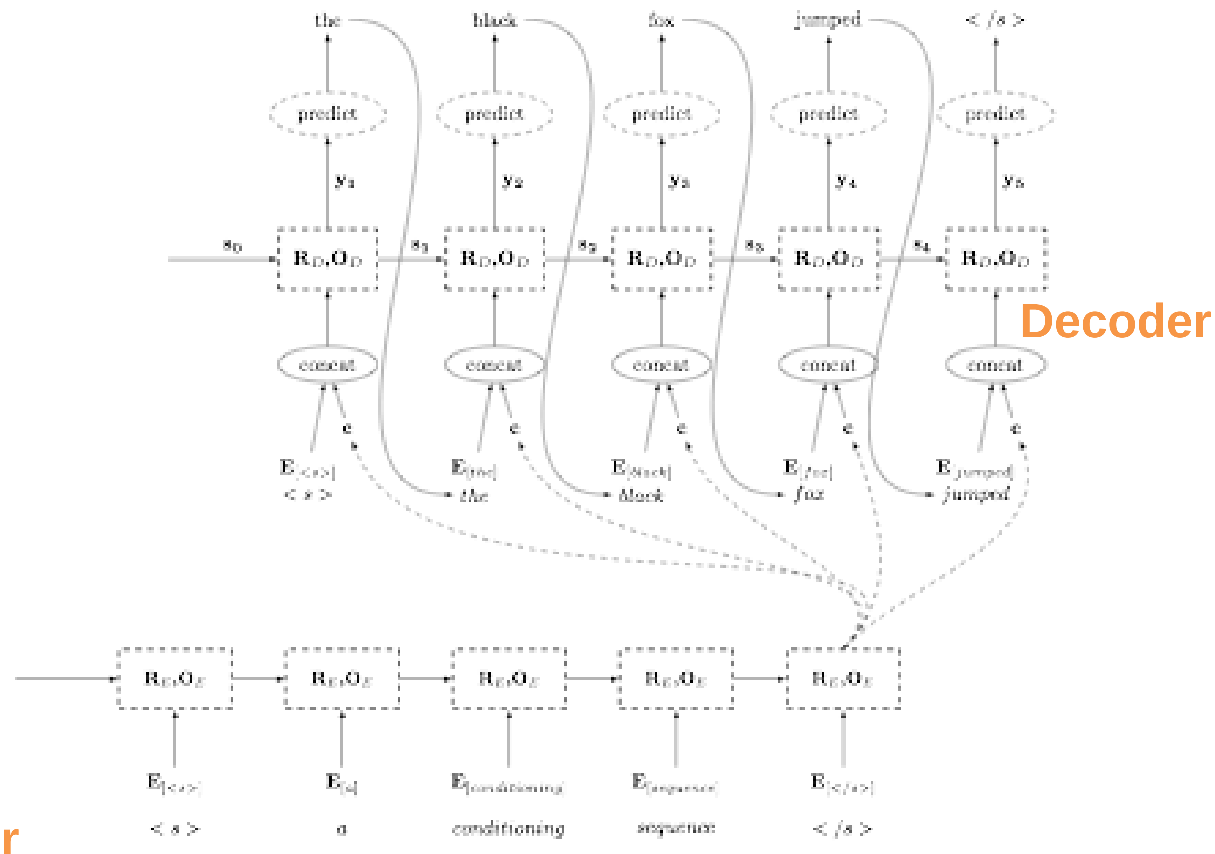
this is an example →

this is an example →



Sequence to Sequence conditioned generation

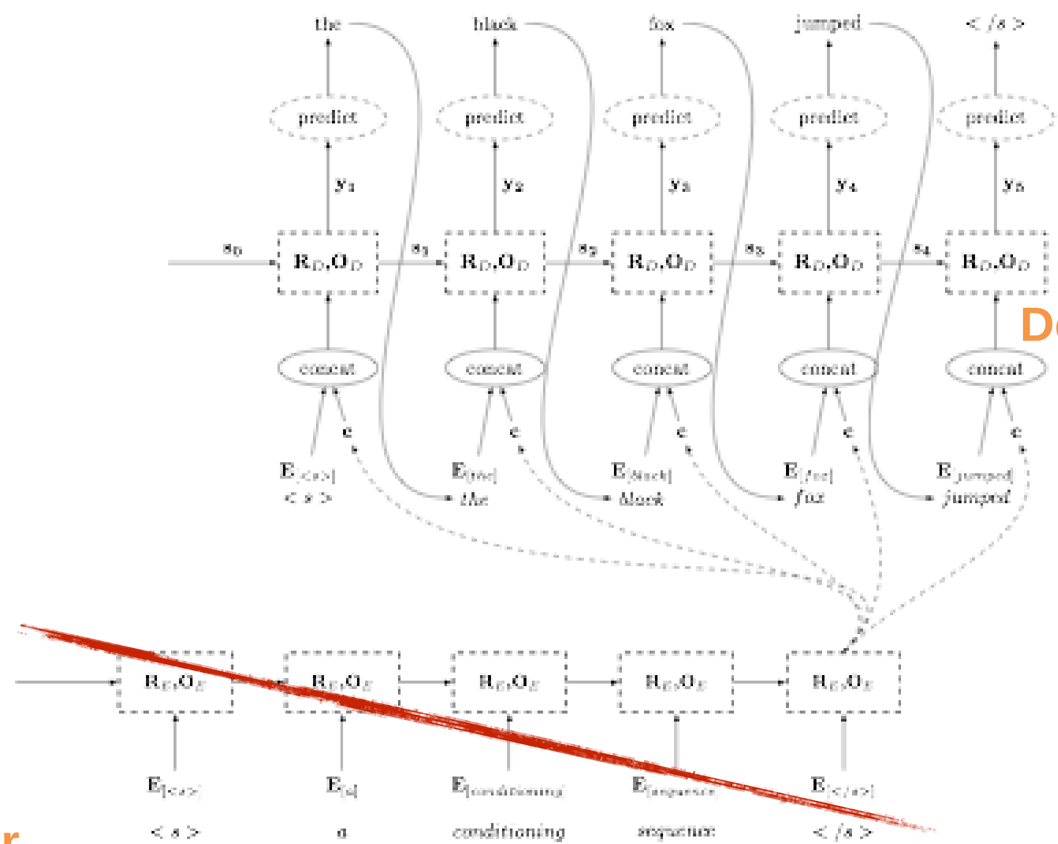
main idea:
encoding
a **single vector** is
too restrictive.



Attention

- Instead of the encoder producing a single vector for the sentence, it will produce a one vector **for each word**.

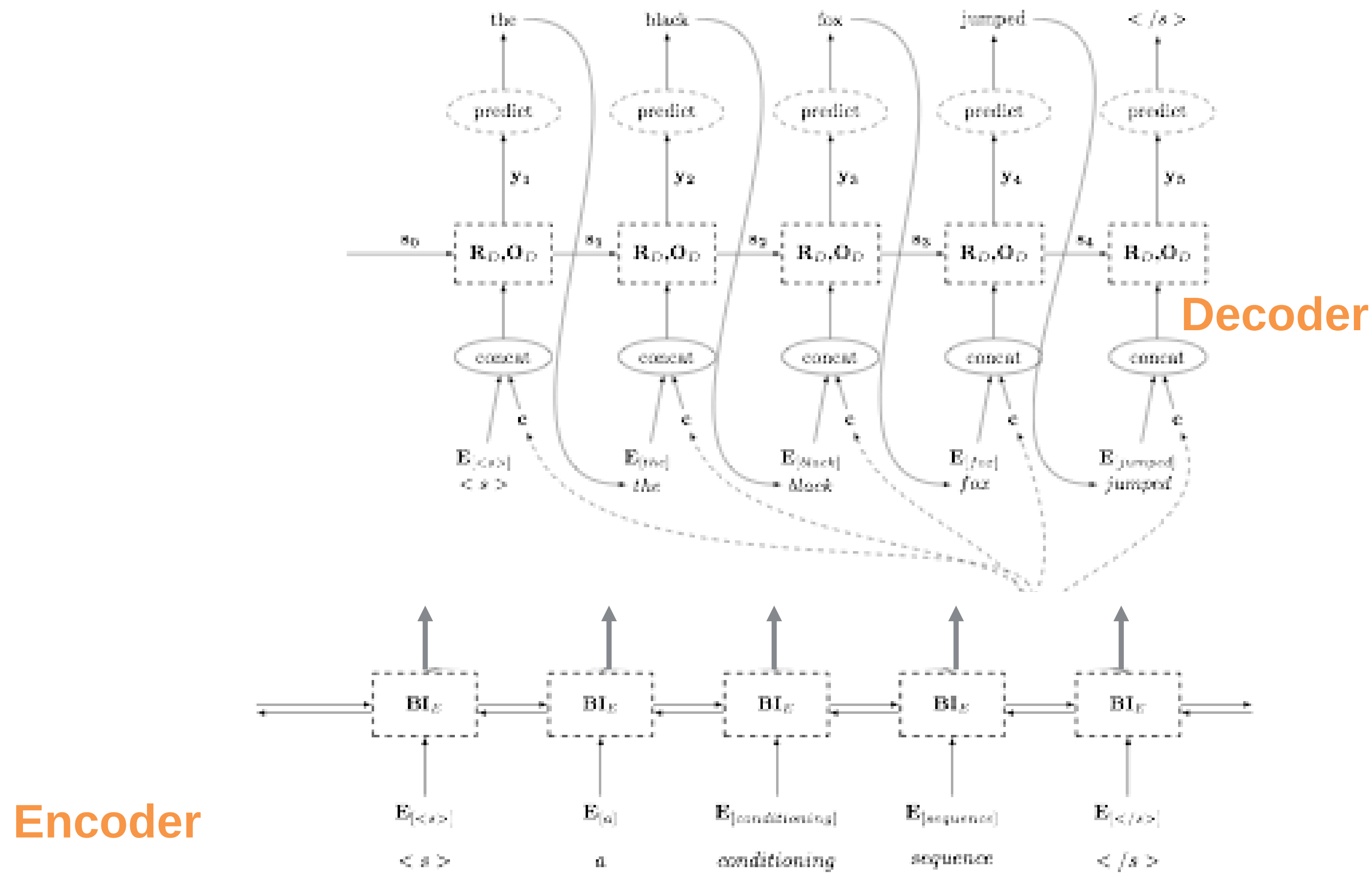
Sequence to Sequence conditioned generation



Decoder

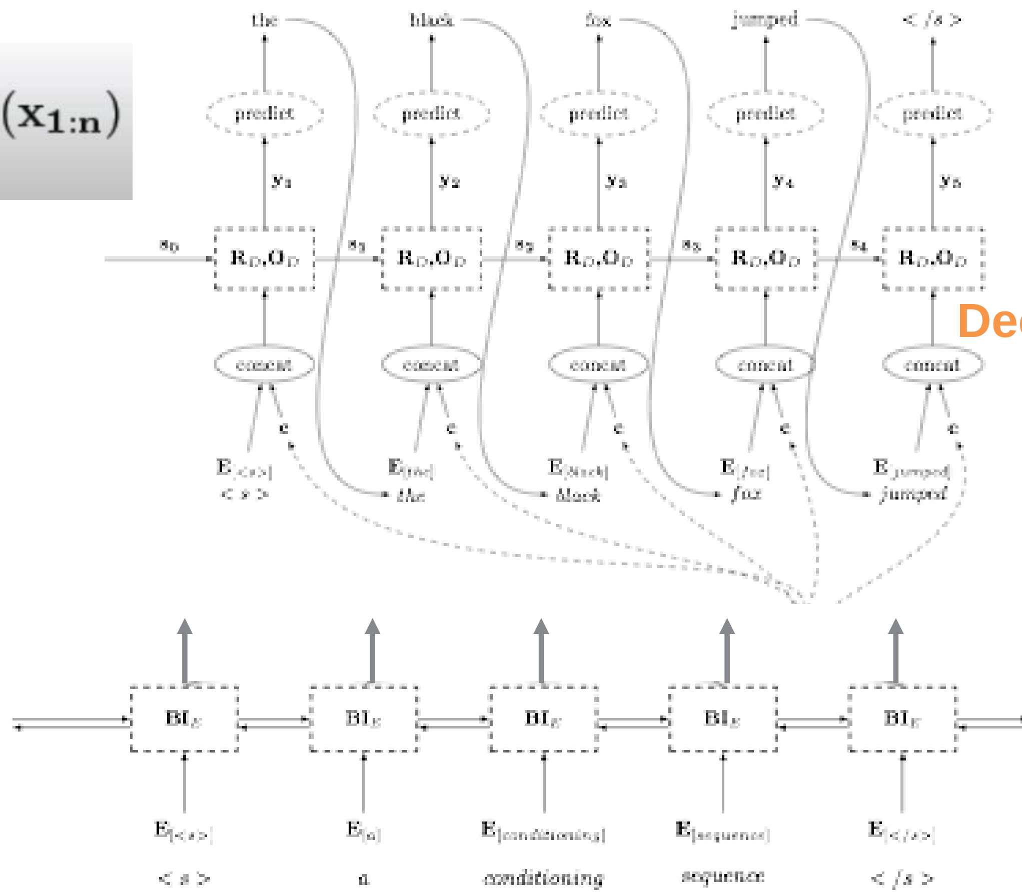
Encoder

Sequence to Sequence conditioned generation



Sequence to Sequence conditioned generation

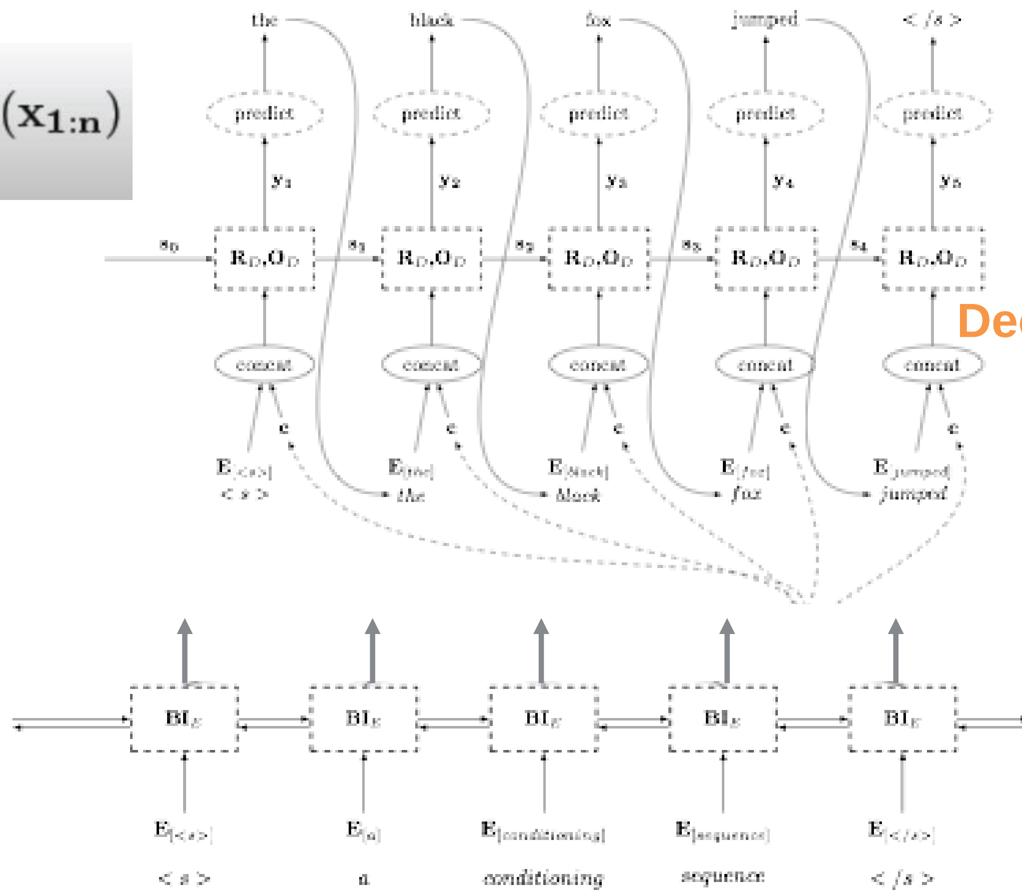
$$\mathbf{c_{1:n}} = \textsc{Enc}(\mathbf{x_{1:n}}) = \text{biRNN}^{\star}(\mathbf{x_{1:n}})$$



Sequence to Sequence conditioned generation

$$\mathbf{c_{1:n}} = \textsc{Enc}(\mathbf{x_{1:n}}) = \text{biRNN}^{\star}(\mathbf{x_{1:n}})$$

but how do we feed this sequence to the decoder?

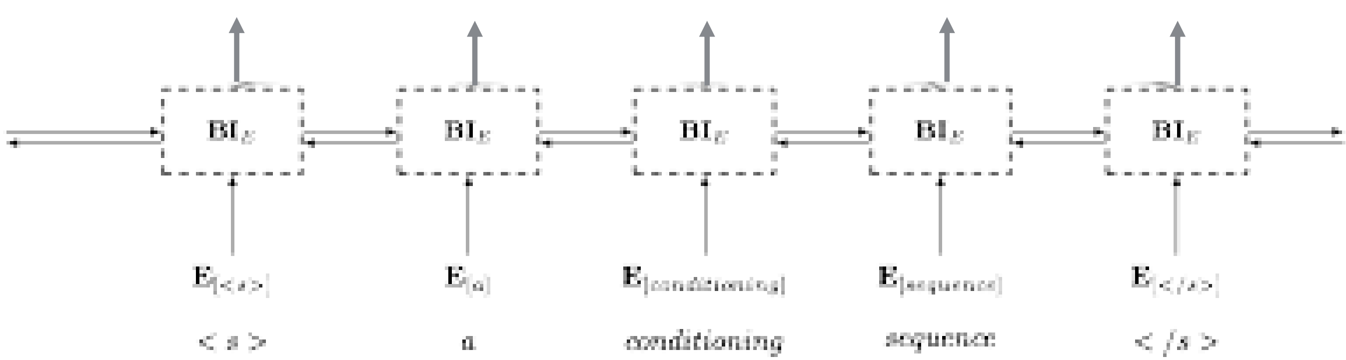


Decoder

Encoder

Sequence to Sequence conditioned generation

we can combine the different outputs
into a single vector (attended summary)

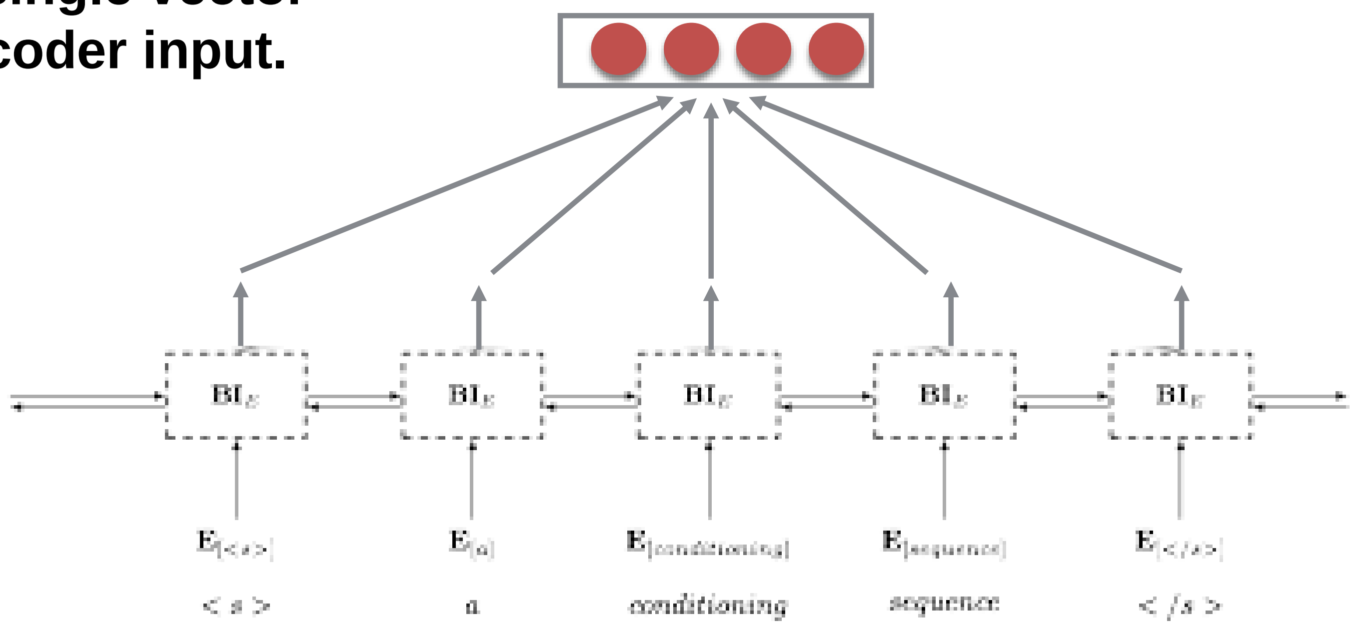


Encoder

Sequence to Sequence conditioned generation

we can combine the different outputs
into a single vector (attended summary)

a different single vector at each encoder input.



Encoder

$$
\begin{aligned}
p(t_{j+1} = k \mid \hat{t}_{1:j}, \mathbf{x_{1:n}}) &= f(O(\mathbf{s_{j+1}})) \\
\mathbf{s_{j+1}} &= R(\mathbf{s_j}, [\mathbf{\hat{t}_j}, \mathbf{c^j}]) \\
\mathbf{c^j} &= \mathrm{attend}(\mathbf{c_{1:n}}, \hat{t}_{1:j}) \\
\hat{t}_j &\sim p(t_j \mid \hat{t}_{1:j-1}, \mathbf{x_{1:n}})
\end{aligned}
$$

$$p(t_{j+1} = k \mid \hat{t}_{1:j}, \mathbf{x_{1:n}}) = f(O(\mathbf{s_{j+1}}))$$

$$\mathbf{s_{j+1}} = R(\mathbf{s_j}, [\hat{\mathbf{t}}_\mathbf{j}; \mathbf{c^j}])$$

$$\mathbf{c^j} = \mathrm{attend}(\mathbf{c_{1:n}}, \hat{t}_{1:j})$$

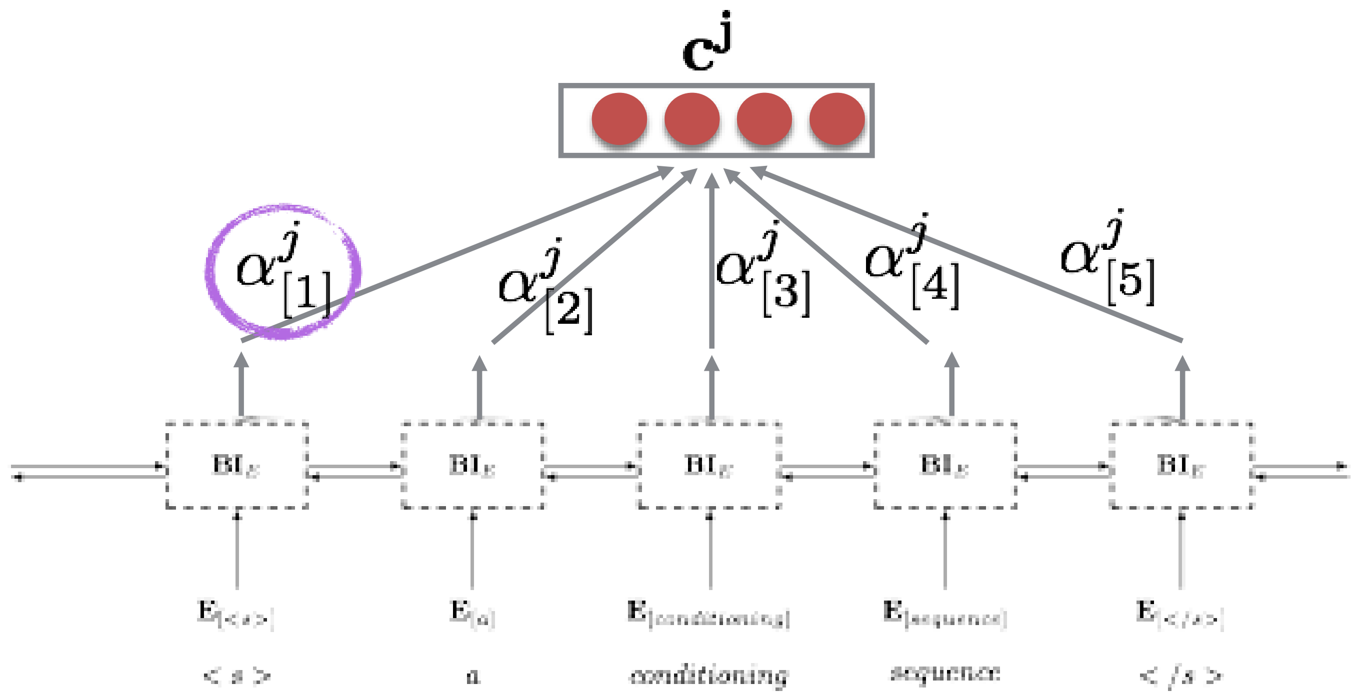
$$\hat{t}_j \sim p(t_j \mid \hat{t}_{1:j-1}, \mathbf{x_{1:n}})$$

$$\mathbf{c^j} = \sum_{i=1}^{n} \alpha^{\mathbf{j}}_{[\mathbf{i}]} \cdot \mathbf{c_i}$$

Sequence to Sequence conditioned generation

$$\alpha^{\mathbf{j}} = \text{softmax}(\bar{\alpha}^{\mathbf{j}}_{[\mathbf{1}]}, \ldots, \bar{\alpha}^{\mathbf{j}}_{[\mathbf{n}]})$$

$$\mathbf{c}^{\mathbf{j}} = \sum_{i=1}^{n} \alpha^{\mathbf{j}}_{[\mathbf{i}]} \, \mathbf{c}_{\mathbf{i}}$$



Encoder

Sequence to Sequence conditioned generation

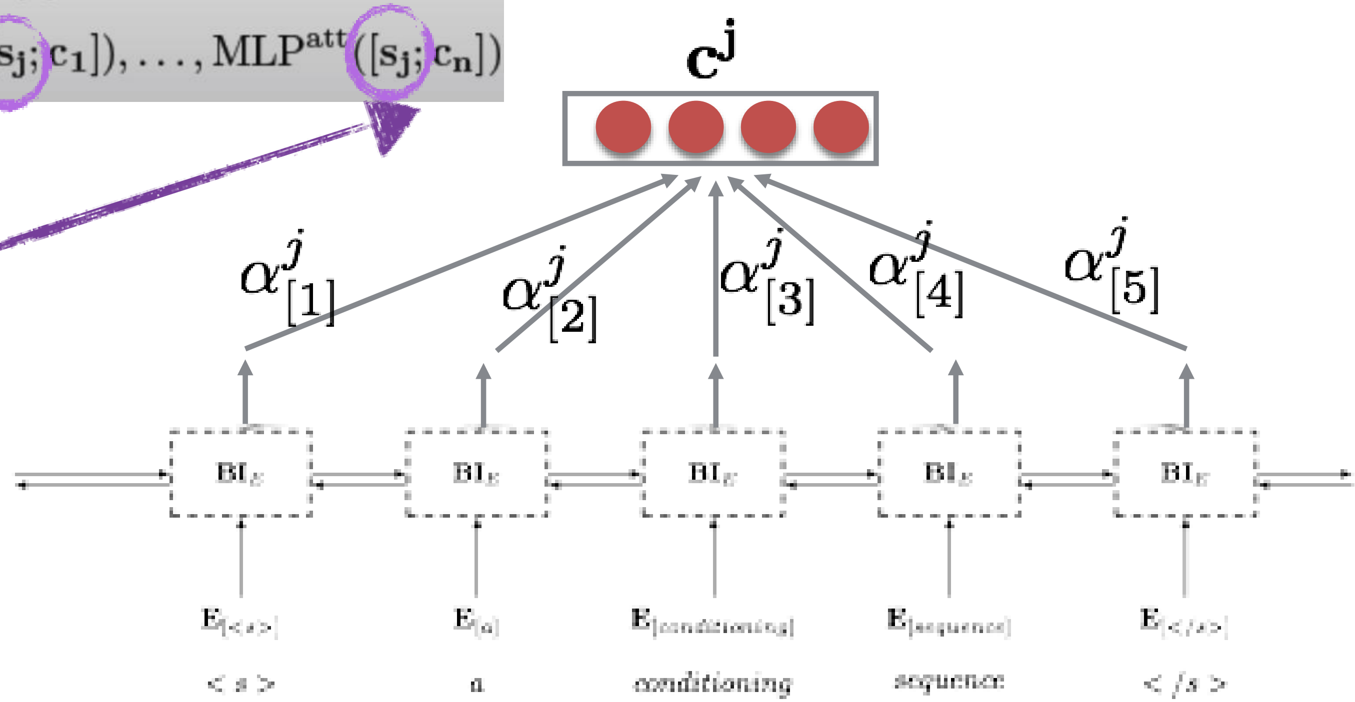
$$\alpha^{\mathbf{j}} = \mathrm{softmax}(\bar{\alpha}^{\mathbf{j}}_{[\mathbf{1}]}, \ldots, \bar{\alpha}^{\mathbf{j}}_{[\mathbf{n}]})$$

$$\bar{\alpha}^{\mathbf{j}} = \bar{\alpha}^{\mathbf{j}}_{[\mathbf{1}]}, \ldots, \bar{\alpha}^{\mathbf{j}}_{[\mathbf{n}]} =$$

$$= \mathrm{MLP}^{\mathrm{att}}([\mathbf{s_j}; \mathbf{c_1}]), \ldots, \mathrm{MLP}^{\mathrm{att}}([\mathbf{s_j}; \mathbf{c_n}])$$

$$\mathbf{c}^{\mathbf{j}} = \sum_{i=1}^{n} \alpha^{\mathbf{j}}_{[\mathbf{i}]} \cdot \mathbf{c_i}$$

decoder state



Encoder

encoder-decoder with attention

$$p(t_{j+1} = k \mid \hat{t}_{1:j}, \mathbf{x_{1:n}}) = f(O_{\text{dec}}(\mathbf{s_{j+1}}))$$

$$\mathbf{s_{j+1}} = R_{\text{dec}}(\mathbf{s_j}, [\hat{\mathbf{t}}_\mathbf{j}; \mathbf{c^j}])$$

$$\mathbf{c^j} = \sum_{i=1}^{n} \alpha^{\mathbf{j}}_{[\mathbf{i}]} \cdot \mathbf{c_i}$$

$$\mathbf{c_{1:n}} = \text{biRNN}^{\star}_{\text{enc}}(\mathbf{x_{1:n}})$$

$$\alpha^{\mathbf{j}} = \text{softmax}(\bar{\alpha}^{\mathbf{j}}_{[\mathbf{1}]}, \ldots, \bar{\alpha}^{\mathbf{j}}_{[\mathbf{n}]})$$

$$\bar{\alpha}^{\mathbf{j}}_{[\mathbf{i}]} = \text{MLP}^{\text{att}}([\mathbf{s_j}; \mathbf{c_i}])$$

$$\hat{t}_j \sim p(t_j \mid \hat{t}_{1:j-1}, \mathbf{x_{1:n}})$$

$$f(\mathbf{z}) = \text{softmax}(\text{MLP}^{\text{out}}(\mathbf{z}))$$

$$\text{MLP}^{\text{att}}([\mathbf{s_j}; \mathbf{c_i}]) =$$

encoder-decoder with attention

$$p(t_{j+1} = k \mid \hat{t}_{1:j}, \mathbf{x_{1:n}}) = f(O_{\text{dec}}(\mathbf{s_{j+1}}))$$

$$\mathbf{s_{j+1}} = R_{\text{dec}}(\mathbf{s_j}, [\hat{\mathbf{t}}_\mathbf{j}; \mathbf{c^j}])$$

$$\mathbf{c^j} = \sum_{i=1}^{n} \alpha^{\mathbf{j}}_{[\mathbf{i}]} \cdot \mathbf{c_i}$$

$$\mathbf{c_{1:n}} = \text{biRNN}^{\star}_{\text{enc}}(\mathbf{x_{1:n}})$$

$$\alpha^{\mathbf{j}} = \text{softmax}(\bar{\alpha}^{\mathbf{j}}_{[\mathbf{1}]}, \ldots, \bar{\alpha}^{\mathbf{j}}_{[\mathbf{n}]})$$

$$\bar{\alpha}^{\mathbf{j}}_{[\mathbf{i}]} = \text{MLP}^{\text{att}}([\mathbf{s_j}; \mathbf{c_i}])$$

$$\hat{t}_j \sim p(t_j \mid \hat{t}_{1:j-1}, \mathbf{x_{1:n}})$$

$$f(\mathbf{z}) = \text{softmax}(\text{MLP}^{\text{out}}(\mathbf{z}))$$

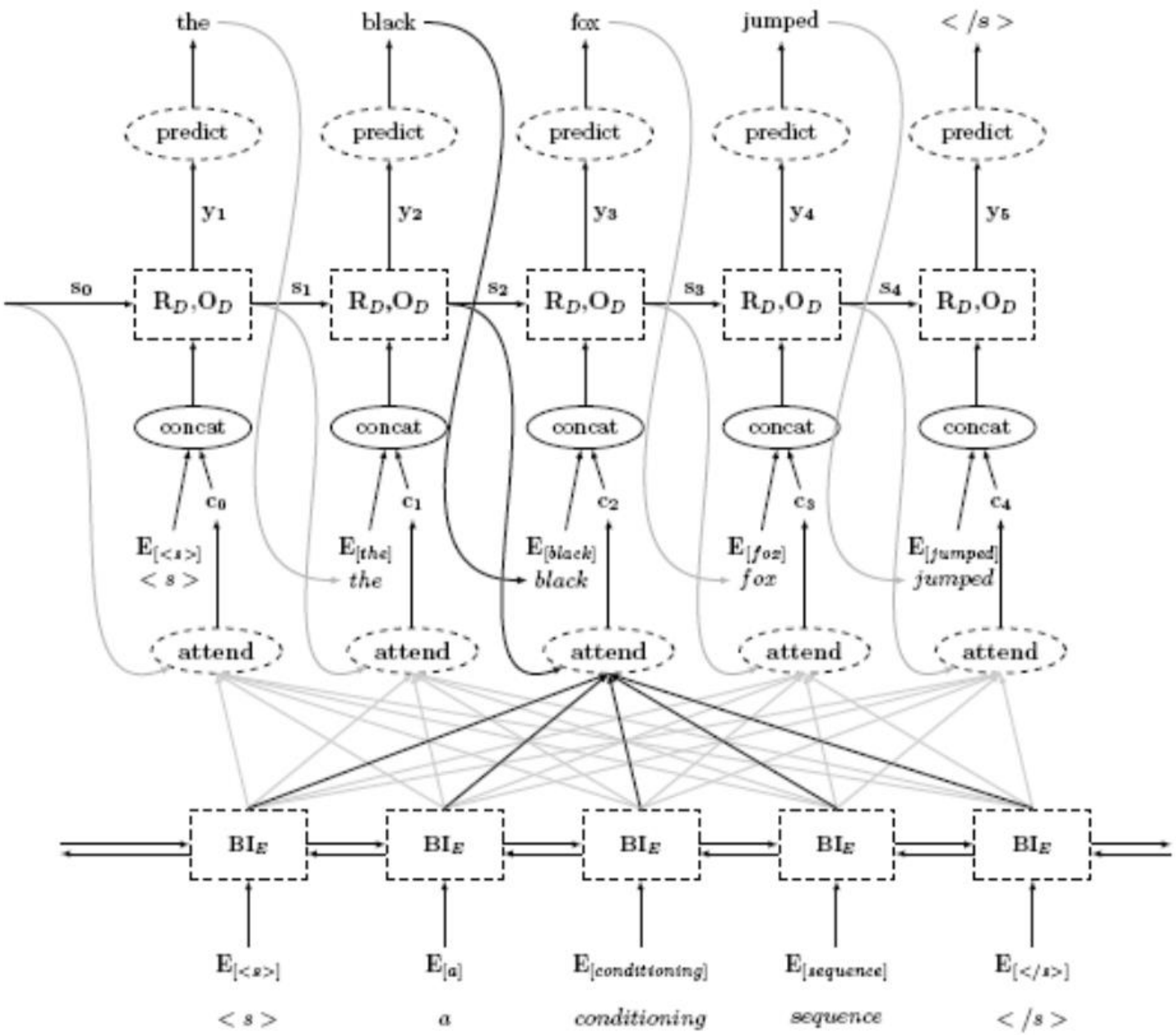
$$\text{MLP}^{\text{att}}([\mathbf{s_j}; \mathbf{c_i}]) =$$

encoder-decoder with attention

$$
\begin{aligned}
p(t_{j+1} = k \mid \hat{t}_{1:j}, \mathbf{x_{1:n}}) &= f(O_{\text{dec}}(\mathbf{s_{j+1}})) \\
\mathbf{s_{j+1}} &= R_{\text{dec}}(\mathbf{s_j}, [\hat{\mathbf{t}}_{\mathbf{j}}; \mathbf{c^j}]) \\
\mathbf{c^j} &= \sum_{i=1}^{n} \alpha^{\mathbf{j}}_{[\mathbf{i}]} \cdot \mathbf{c_i} \\
\mathbf{c_{1:n}} &= \text{biRNN}^{\star}_{\text{enc}}(\mathbf{x_{1:n}}) \\
\alpha^{\mathbf{j}} &= \text{softmax}(\bar{\alpha}^{\mathbf{j}}_{[\mathbf{1}]}, \dots, \bar{\alpha}^{\mathbf{j}}_{[\mathbf{n}]}) \\
\bar{\alpha}^{\mathbf{j}}_{[\mathbf{i}]} &= \text{MLP}^{\text{att}}([\mathbf{s_j}; \mathbf{c_i}]) \\
\hat{t}_j &\sim p(t_j \mid \hat{t}_{1:j-1}, \mathbf{x_{1:n}}) \\
f(\mathbf{z}) &= \text{softmax}(\text{MLP}^{\text{out}}(\mathbf{z}))
\end{aligned}
$$

$$\text{MLP}^{\text{att}}([\mathbf{s_j}; \mathbf{c_i}]) =$$

encoder-decoder with attention



encoder-decoder with attention

- Encoder encodes a sequence of vectors, $c_1,\ldots,c_n$
- At each decoding stage, an MLP assigns a relevance score to each Encoder vector.
- The relevance score is based on c_i and the state s_j
- Weighted-sum (based on relevance) is used to produce the conditioning context for decoder step j.

encoder-decoder with attention

- Decoder "pays attention" to different parts of the encoded sequence at each stage.
- The attention mechanism is "soft" -- it is a mixture of encoder states.

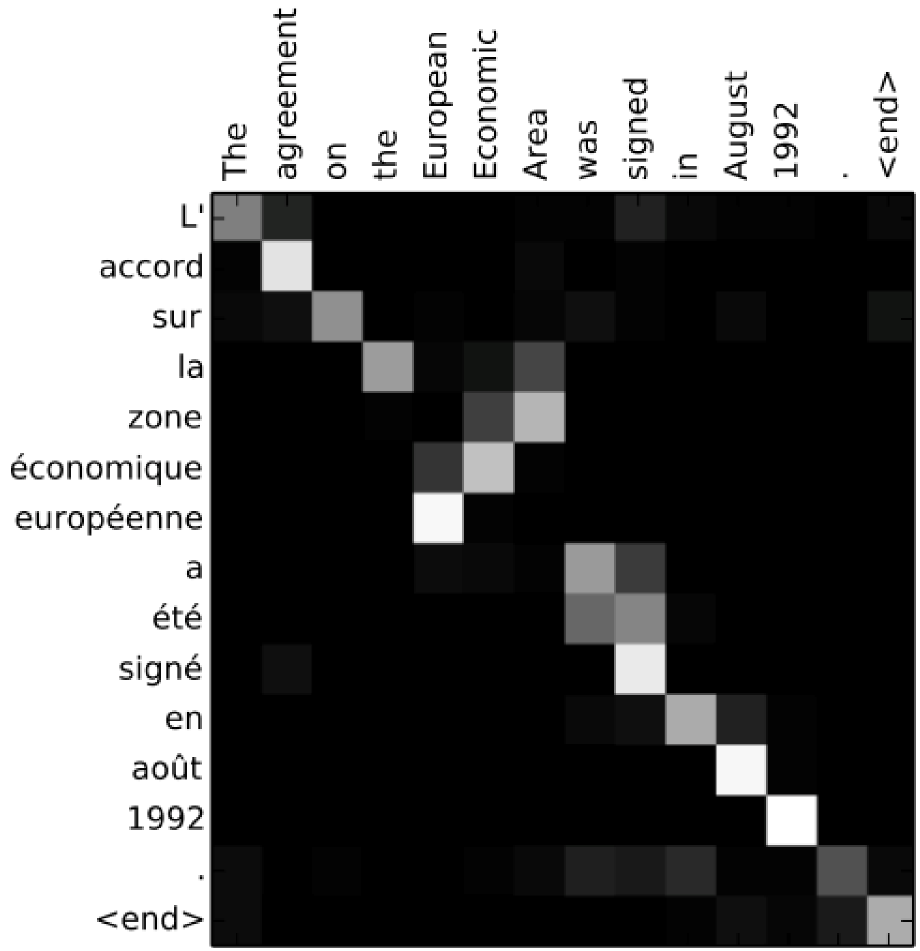
- The encoder acts as a read-only memory for the decoder
- The decoder chooses what to read at each stage

Attention

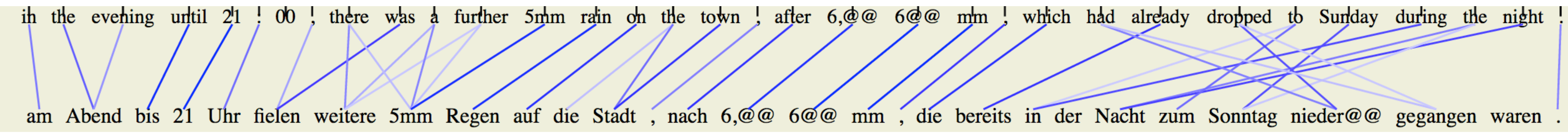
- Attention is very effective for sequence-to-sequence tasks.
- Current state-of-the-art systems all use attention.
  (this is basically how Machine Translation works)

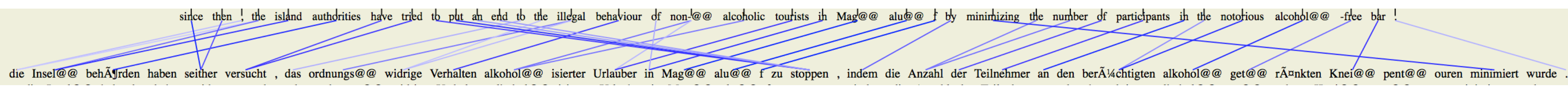
- Attention also makes models somewhat more interpretable.
- (we can see where the model is "looking" at each stage of the prediction process)

Attention


The
agreement
on
the
European
Economic
Area
was
signed
in
August
1992
.
<end>
L'
accord
sur
la
zone
économique
européenne
a
été
signé
en
août
1992
.
<end>

Attention


in the evening until 21 : 00 , there was a further 5mm rain on the town , after 6,@@ 6@@ mm , which had already dropped to Sunday during the night .
am Abend bis 21 Uhr fielen weitere 5mm Regen auf die Stadt , nach 6,@@ 6@@ mm , die bereits in der Nacht zum Sonntag nieder@@ gegangen waren .


since then , the island authorities have tried to put an end to the illegal behaviour of non-@@ alcoholic tourists in Mag@@ alu@@ f by minimizing the number of participants in the notorious alcohol@@ -free bar .
die Insel@@ behÃ¶rden haben seither versucht , das ordnungs@@ widrige Verhalten alkohol@@ isierter Urlauber in Mag@@ alu@@ f zu stoppen , indem die Anzahl der Teilnehmer an den berÃ¼chtigten alkohol@@ get@@ rÃ¤nkten Knei@@ pent@@ ouren minimiert wurde .

Complexity

- Encoder decoder:

- Encoder-decoder with attention:

Complexity

- Encoder decoder: $O(n+m)$

- Encoder-decoder with attention: $O(nm)$

Beyond Seq2Seq

- Can think of a general design pattern in neural nets:
  - **Input**: sequence, query
    - **Encode** the input into a sequence of vectors
    - **Attend** to the encoded vectors, based on query (weighted sum, determined by query)
    - **Predict** based on the attended vector

Attention Functions

v: attended vec, q: query vec
MLP^{att}(q;v)=

- Additive Attention: $\mathbf{u}g(\mathbf{W^1v} + \mathbf{W^2q})$

- Dot Product: $\mathbf{v} \cdot \mathbf{q}$

- Multiplicative Attention: $\mathbf{v}^\top \mathbf{Wq}$

Additive vs Multiplicative

While the two are similar in theoretical complexity, dot-product attention is much faster and more space-efficient in practice, since it can be implemented using highly optimized matrix multiplication code.

While for small values of d_k the two mechanisms perform similarly, additive attention outperforms dot product attention without scaling for larger values of d_k [3]. We suspect that for large values of d_k, the dot products grow large in magnitude, pushing the softmax function into regions where it has extremely small gradients [4]. To counteract this effect, we scale the dot products by $\frac{1}{\sqrt{d_k}}$.

$$\frac{\mathbf{v} \cdot \mathbf{q}}{\sqrt{\boldsymbol{d_k}}}$$

d_k is the dimensionality of q and v

Attention Is All You Need

Ashish Vaswani*
Google Brain
avaswani@google.com

Noam Shazeer*
Google Brain
noam@google.com

Niki Parmar*
Google Research
nikip@google.com

Jakob Uszkoreit*
Google Research
usz@google.com

Llion Jones*
Google Research
llion@google.com

Aidan N. Gomez* †
University of Toronto
aidan@cs.toronto.edu

Łukasz Kaiser*
Google Brain
lukaszkaiser@google.com

Illia Polosukhin* ‡
illia.polosukhin@gmail.com

Key-Value Attention

- Split v into two vectors $v=[v_k;v_v]$
  - v_k: key vector
  - v_v: value vector

- Use key vector for computing attention

  $\mathrm{MLP}^{att}(q;v) = u\,g(\mathbf{W^1}v_k + \mathbf{W^2}q)$ //additive

- Use value vector for computing attended summary

$$\mathbf{v^j} = \sum_{i=1}^{n} \alpha^{j}_{[i]} \cdot (v_v)_i$$

Multi-head Key-Value Attention

- For each head
  - Learn different projection matrices $\mathbf{W_q}$, $\mathbf{W_k}$, $\mathbf{W_v}$
- $MLP^{att}(q;v) = [(v_k\mathbf{W_k}).(q\mathbf{W_q})]/sqrt(d_k)$
- For summary use $v_v\mathbf{W_v}$ (instead of v_v)

- Train many such heads and
  - use aggr(all such attended summaries)

Hard Attention

Instead of a soft interpolation, make a **zero-one decision** about where to attend (Xu et al. 2015)

- Harder to train, requires methods such as reinforcement learning (see later classes)

Perhaps this helps interpretability? (Lei et al. 2016)

Review

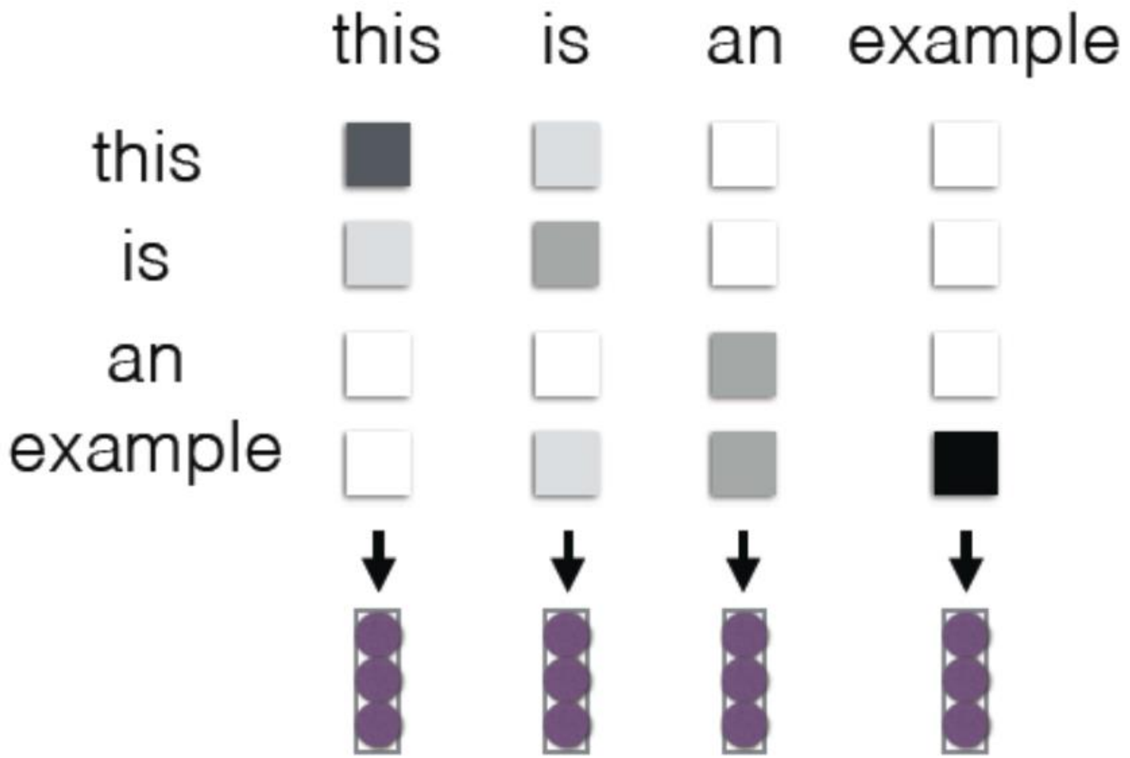
the beer was n't what I expected, and I'm not sure it's "true to style", but i thought it was delicious. **a very pleasant ruby red-amber color** with a relatively brilliant finish, but a limited amount of carbonation, from the look of it. aroma is what i think an amber ale should be - a nice blend of caramel and happiness bound together.

Ratings

Look: 5 stars | *Smell:* 4 stars

Self-attention/Intra-attention

Each element in the sentence attends to other elements → context sensitive encodings!



Recall the attended Enc-dec

$$p(t_{j+1} = k \mid \hat{t}_{1:j}, \mathbf{x_{1:n}}) = f(O_{\text{dec}}(\mathbf{s_{j+1}}))$$

$$\mathbf{s_{j+1}} = R_{\text{dec}}(\mathbf{s_j}, [\hat{\mathbf{t}}_\mathbf{j}; \mathbf{c^j}])$$

$$\mathbf{c^j} = \sum_{i=1}^{n} \alpha^{\mathbf{j}}_{[\mathbf{i}]} \cdot \mathbf{c_i}$$

$$\mathbf{c_{1:n}} = \text{biRNN}^{\star}_{\text{enc}}(\mathbf{x_{1:n}})$$

$$\alpha^{\mathbf{j}} = \text{softmax}(\bar{\alpha}^{\mathbf{j}}_{[\mathbf{1}]}, \ldots, \bar{\alpha}^{\mathbf{j}}_{[\mathbf{n}]})$$

$$\bar{\alpha}^{\mathbf{j}}_{[\mathbf{i}]} = \text{MLP}^{\text{att}}([\mathbf{s_j}; \mathbf{c_i}])$$

$$\hat{t}_j \sim p(t_j \mid \hat{t}_{1:j-1}, \mathbf{x_{1:n}})$$

$$f(\mathbf{z}) = \text{softmax}(\text{MLP}^{\text{out}}(\mathbf{z}))$$

$$\text{MLP}^{\text{att}}([\mathbf{s_j}; \mathbf{c_i}]) = \mathbf{v} \tanh([\mathbf{s_j}; \mathbf{c_i}]\mathbf{U} + \mathbf{b})$$

Self attention with LSTM

- c (in prev slide) = h (in this slide)
- h (hidden state); x (input); h~ (attended summary)

$$a_i^t = v^{\mathrm{T}} \tanh(W_h h_i + W_x x_t + W_{\tilde{h}} \tilde{h}_{t-1})$$

$$s_i^t = \mathrm{softmax}(a_i^t)$$

- (Attended) Hidden state/Cell State

$$\begin{bmatrix} \tilde{h}_t \\ \tilde{c}_t \end{bmatrix} = \sum_{i=1}^{t-1} s_i^t \cdot \begin{bmatrix} h_i \\ c_i \end{bmatrix}$$

- Rest of LSTM

$$\begin{bmatrix} i_t \\ f_t \\ o_t \\ \hat{c}_t \end{bmatrix} = \begin{bmatrix} \sigma \\ \sigma \\ \sigma \\ \tanh \end{bmatrix} W \cdot [\tilde{h}_t, x_t]$$

$$c_t = f_t \odot \tilde{c}_t + i_t \odot \hat{c}_t$$

$$h_t = o_t \odot \tanh(c_t)$$

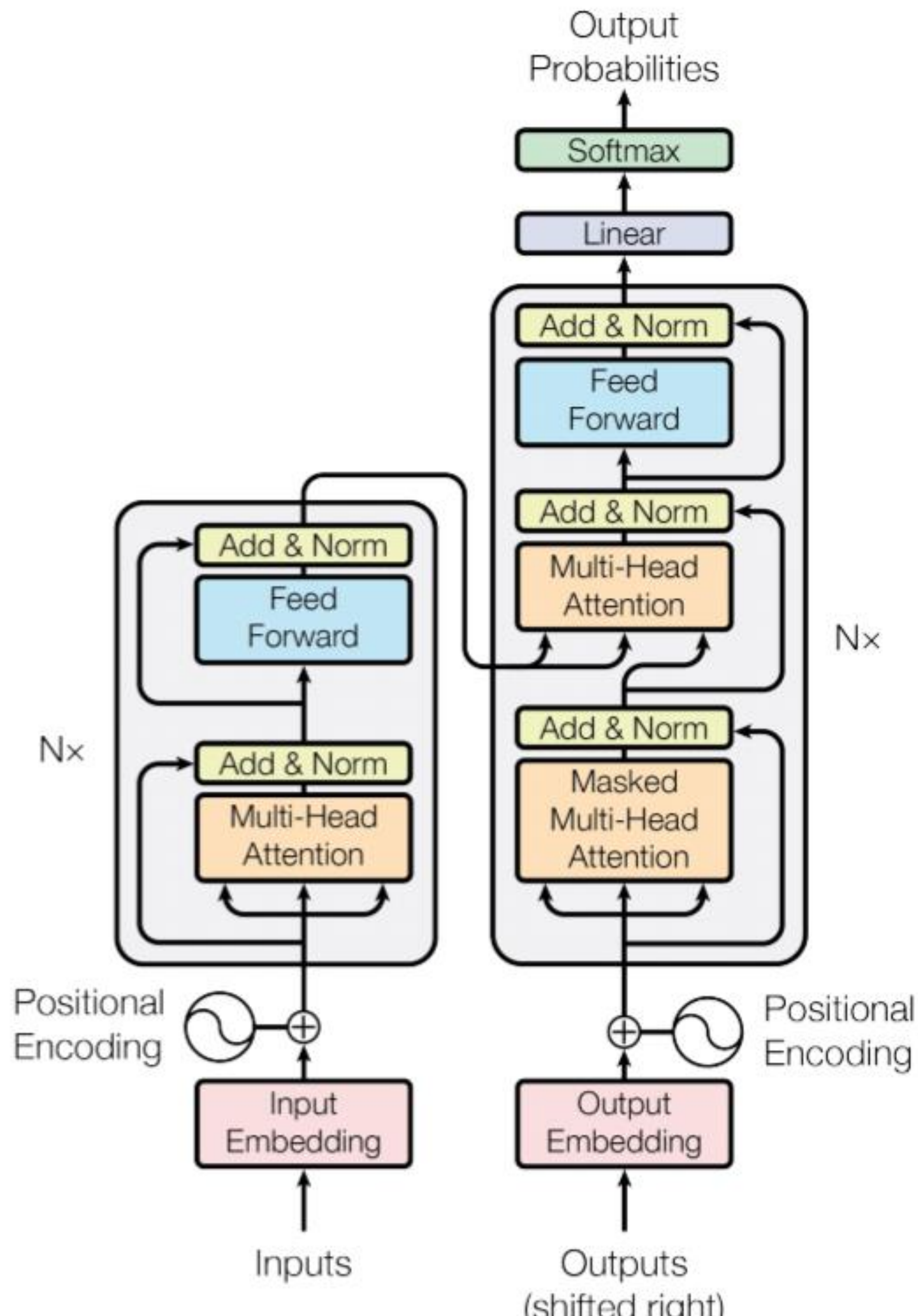
Do we "need" an LSTM?

Objective

- RNN is slow; can't be parallelized
- Reduce sequential computation

- Self-attention encoder (Transformer)
  - creatively combines layers of attention
  - with other bells and whistles

- Self-attention decoder!!


Output
Probabilities
Softmax
Linear
Add & Norm
Feed
Forward
Add & Norm
Multi-Head
Attention
Nx
Add & Norm
Masked
Multi-Head
Attention
Add & Norm
Feed
Forward
Nx
Add & Norm
Multi-Head
Attention
Positional
Encoding
Positional
Encoding
Input
Embedding
Output
Embedding
Inputs
Outputs
(shifted right)

Layer Type	Complexity per Layer	Sequential Operations	Maximum Path Length
Self-Attention	$O(n^2 \cdot d)$	$O(1)$	$O(1)$
Recurrent	$O(n \cdot d^2)$	$O(n)$	$O(n)$
Convolutional	$O(k \cdot n \cdot d^2)$	$O(1)$	$O(log_k(n))$
Self-Attention (restricted)	$O(r \cdot n \cdot d)$	$O(1)$	$O(n/r)$

Summary

- RNNs are very capable learners of sequential data.
- n -> 1: (bi)RNN acceptor
- n -> n : biRNN (transducer)
- 1 -> m : conditioned generation (conditioned LM)
- n -> m : conditioned generation (encoder-decoder)
- n -> m : encoder-decoder with attention