Sequence to Sequence Learning Using Deep Learning

Anoop Kunchukuttan

Center for Indian Language Technology,
Indian Institute of Technology Bombay

anoopk@cse.iitb.ac.in

https://www.cse.iitb.ac.in/~anoopk

Deep Learning Tutorial.
ICON 2017, Kolkata
21th December 2017

Outline

- Introduction
- Neural Machine Translation
- Summary

Outline

- Introduction
- Neural Machine Translation
- Summary

Introduction

- Typical machine learning approaches deals with fixed input or variable-length input and fixed output
  - Image Classification
  - Sentiment Analysis
  - …
- Certain tasks require the machine learning approach to generate variable-length outputs
  - Summary Generation
  - Machine Translation
  - Image Descriptions
- We will look at Deep Learning Approaches for such tasks, specifically we focus on Machine Translation

What is Machine Translation?

Automatic conversion of text/speech from one natural language to another

Be the change you want to see in the world

वह परिवर्तन बनो जो संसार में देखना चाहते हो

Outline

- Introduction
- Neural Machine Translation
- Summary

Neural Machine Translation

***SMT, Rule-based MT and Example based MT* manipulate symbolic representations of knowledge**

Every word has an atomic representation, which can't be further analyzed

home	0
water	1
house	2
tap	3

1	0	0	0
0	1	0	0
0	0	1	0
0	0	0	1

No notion of similarity or relationship between words

- *Even if we know the translation of* `home`*, we can't translate* `house` *if it an OOV*

Difficult to represent new concepts

- *We cannot say nothing about 'mansion' if it comes up at test time*
- *Creates problems language model as well ⇒ whole are of smoothing exists to overcome this problem*

Neural Network techniques work with distributed representations

Every word is represented by a vector of numbers

- *No element of the vector represents a particular word*
- *The word can be understood with all vector elements*
- *Hence distributed representation*
- *But less interpretable*

home	0.5	0.6	0.7
water	0.2	0.9	0.3
house	0.55	0.58	0.77
tap	0.24	0.6	0.4

Word vectors or embeddings

Can define similarity between words

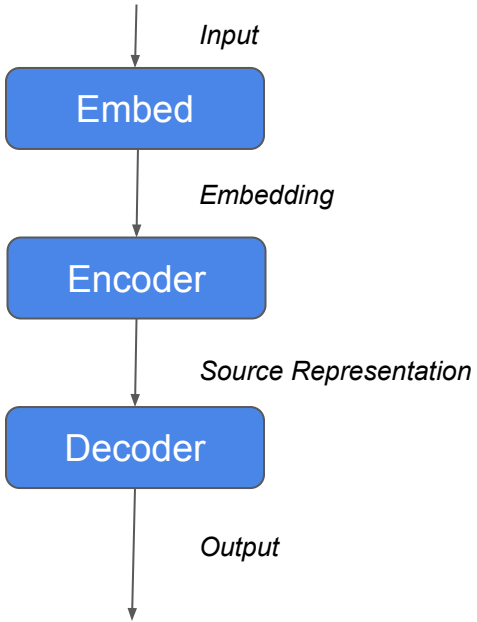
- *Vector similarity measures like cosine similarity*
- *Since* representations of `home` *and* `house`*, we may be able to translate* `house`

New concepts can be represented using a vector with different values

Symbolic representations are ***continuous representations***

- *Generally computationally more efficient to work with continuous values*
- *Especially optimization problems*

Encode - Decode Paradigm



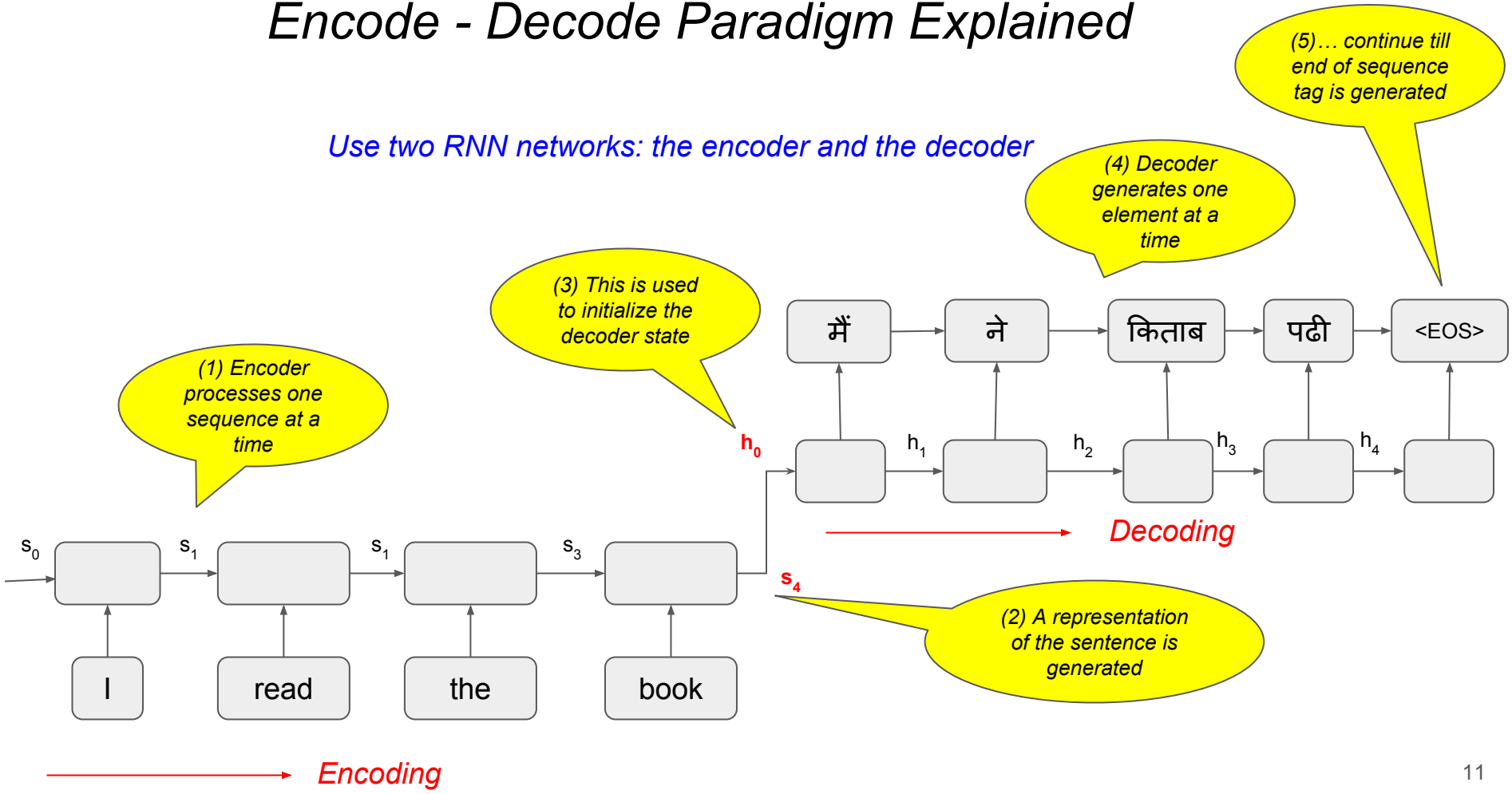
Entire input sequence is processed before generation starts
⇒ In PBSMT, generation was piecewise

The input is a sequence of words, processed one at a time

- *While processing a word, the network needs to know what it has seen so far in the sequence*
- *Meaning, know the history of the sequence processing*
- *Needs a special kind of neural: Recurrent neural network unit which can keep state information*

Encode - Decode Paradigm Explained

Use two RNN networks: the encoder and the decoder



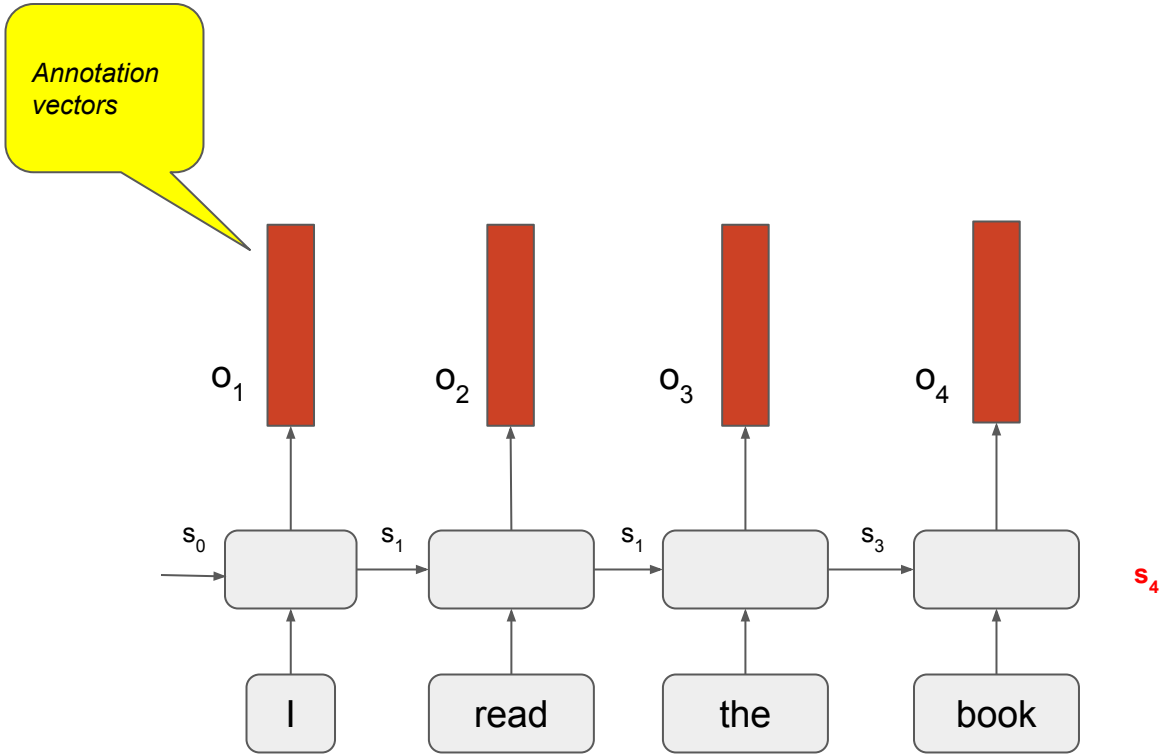
This approach reduces the entire sentence representation to a single vector

Two problems with this design choice:

- A single vector is not sufficient to represent to capture all the syntactic and semantic complexities of a sentence
  - *Solution: Use a richer representation for the sentences*
- Problem of capturing long term dependencies: The decoder RNN will not be able to make use of source sentence representation after a few time steps
  - *Solution: Make source sentence information when making the next prediction*
  - *Even better, make **RELEVANT** source sentence information available*

These solutions motivate the next paradigm

Encode - Attend - Decode Paradigm



Represent the source sentence by the **set of output vectors** from the encoder

Each output vector at time *t* is a contextual representation of the input at time *t*

Note: in the encoder-decode paradigm, we ignore the encoder outputs

Let's call these encoder output vectors ***annotation vectors***

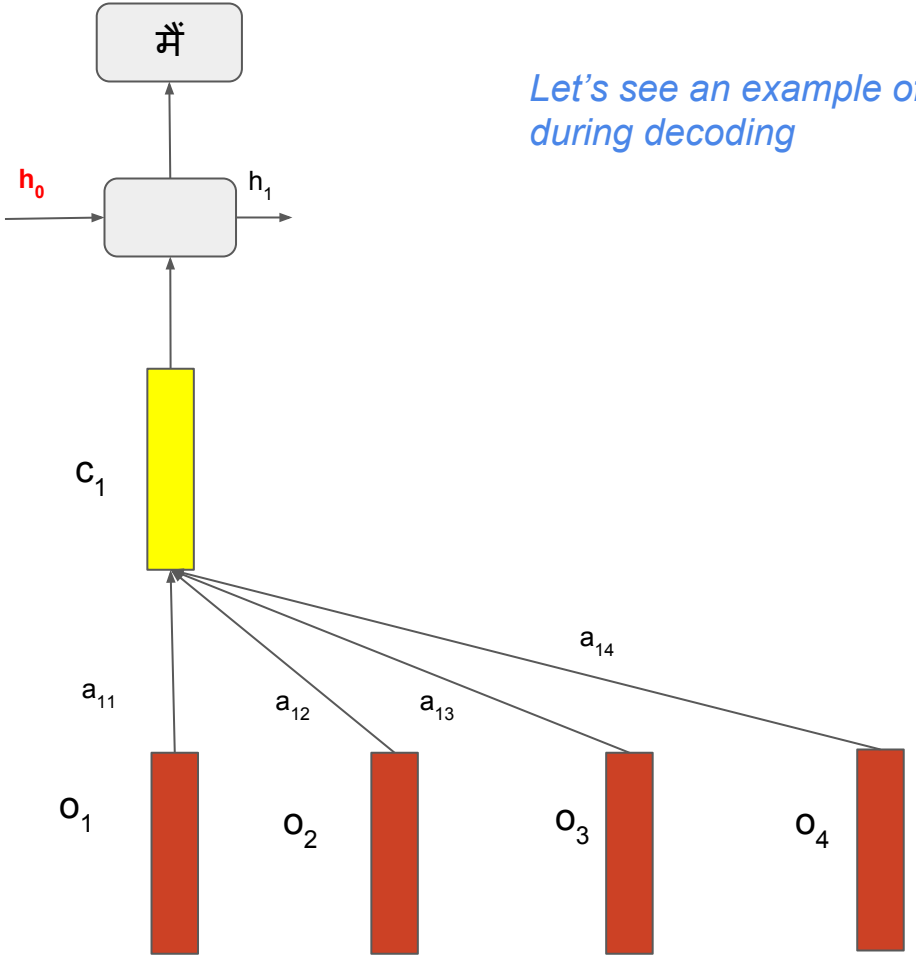
How should the decoder use the set of annotation vectors while predicting the next character?

Key Insight:
(1) Not all annotation vectors are equally important for prediction of the next element
(2) The annotation vector to use next depends on what has been generated so far by the decoder

eg. To generate the 3rd target word, the 3rd annotation vector (hence 3rd source word) is most important

One way to achieve this:
Take a weighted average of the annotation vectors, with more weight to annotation vectors which need more **focus or attention**

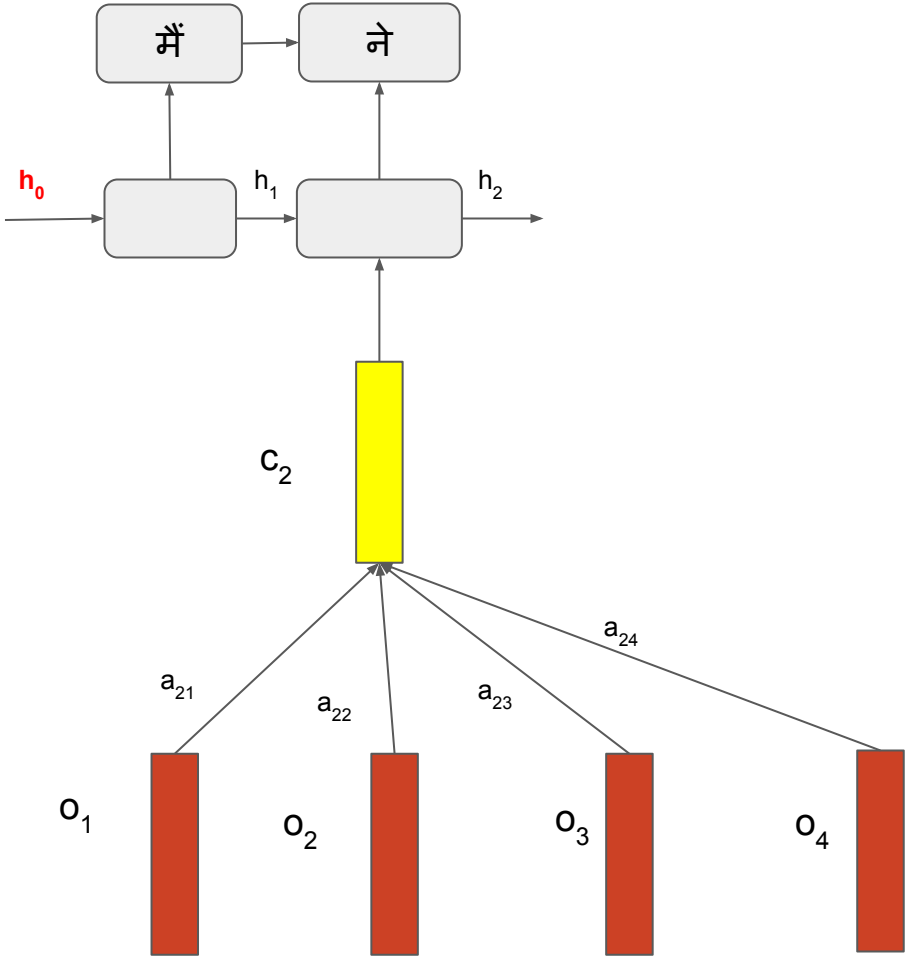
This averaged ***context vector*** is an input to the decoder

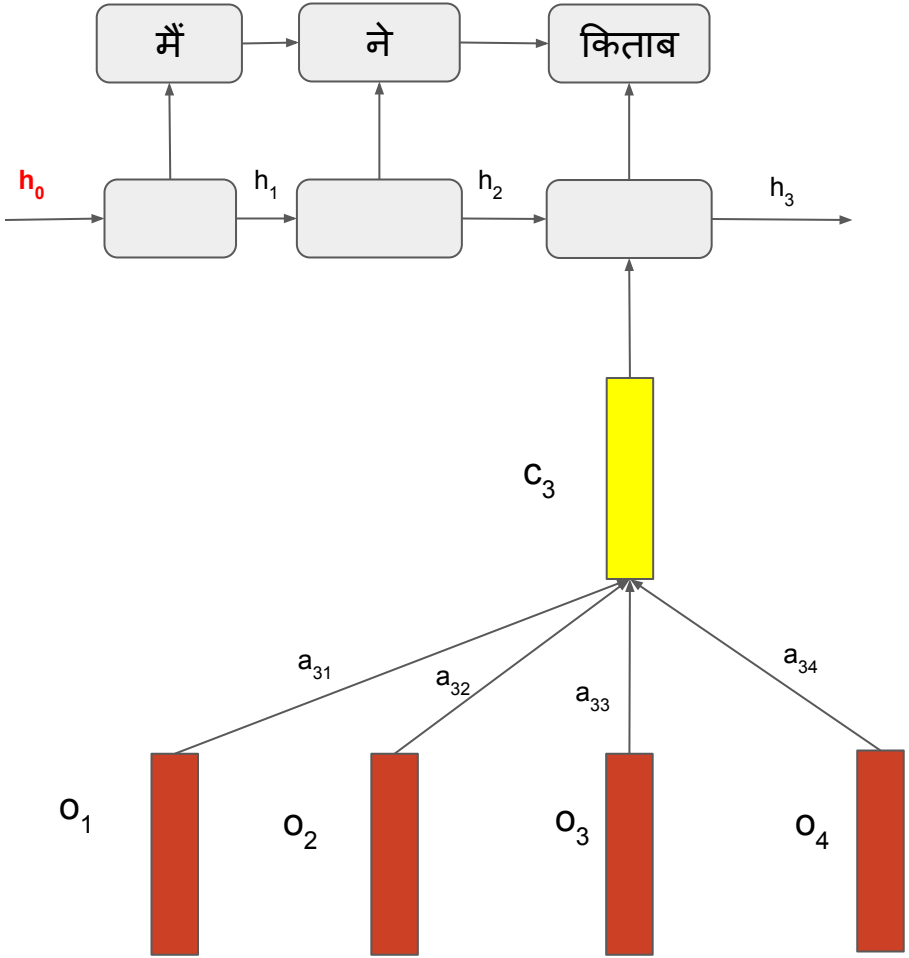


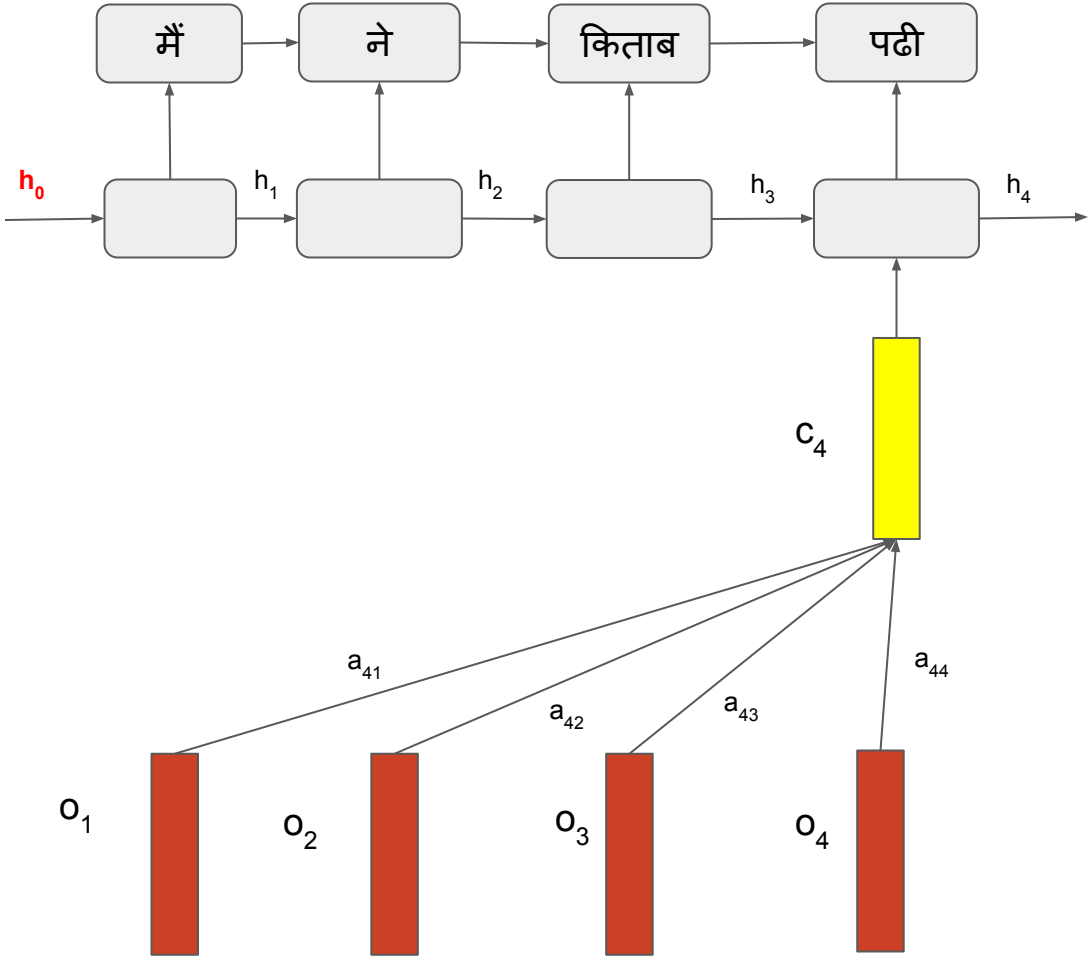
*Let's see an example of how the **attention mechanism** works during decoding*

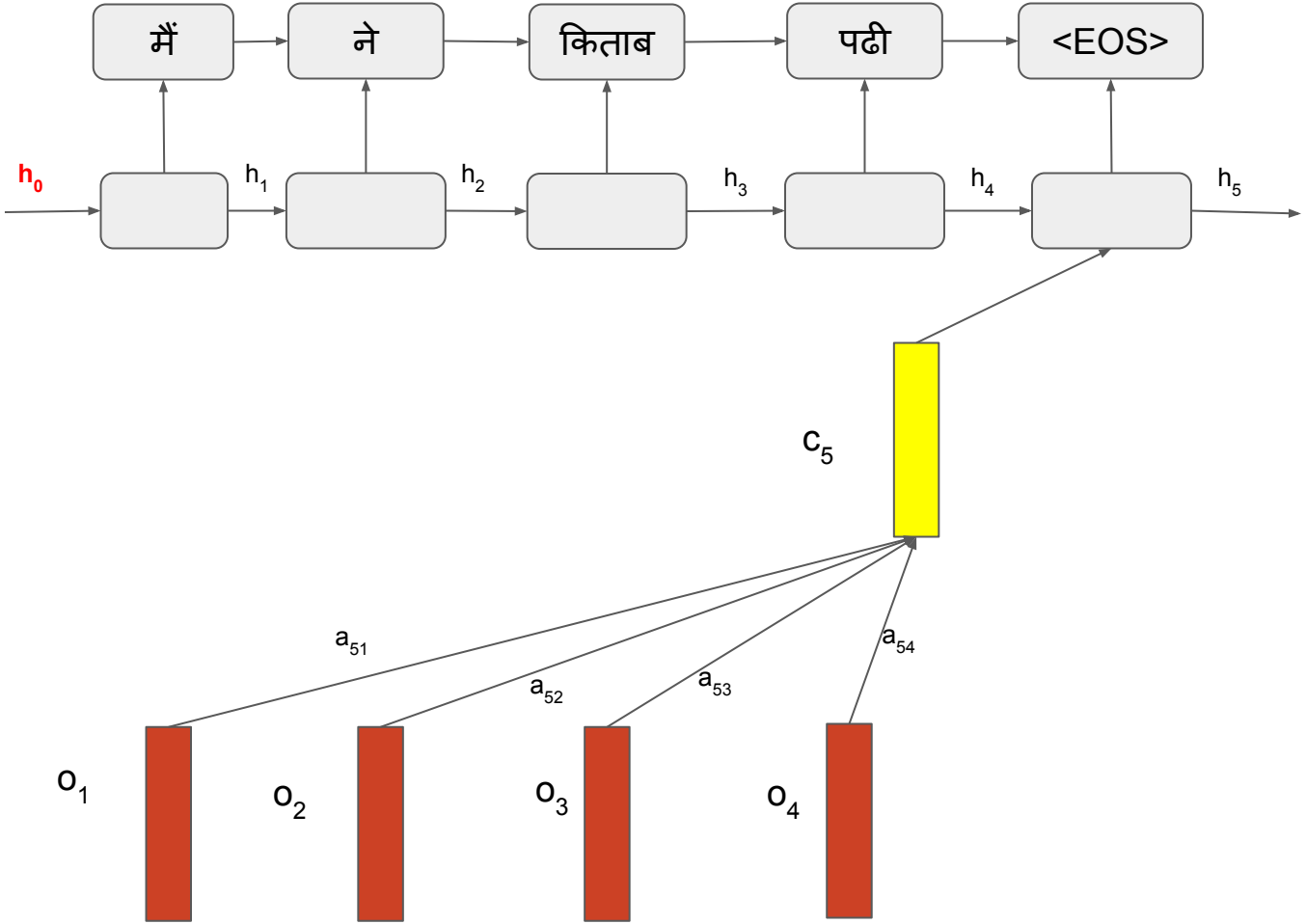
$$c_i = \sum_{j=1}^{n} a_{ij} o_j$$

For generation of i^{th} output character:
c_i : context vector
a_{ij} : annotation weight for the j^{th} annotation vector
o_j: j^{th} annotation vector


मैं
ने
h0
h1
h2
c2
a21
a22
a23
a24
o1
o2
o3
o4


मैं
ने
किताब
h0
h1
h2
h3
c3
a31
a32
a33
a34
o1
o2
o3
o4


मैं
ने
किताब
पढी
h0
h1
h2
h3
h4
c4
a41
a42
a43
a44
o1
o2
o3
o4


मैं
ने
किताब
पढी
<EOS>
h0
h1
h2
h3
h4
h5
c5
a51
a52
a53
a54
o1
o2
o3
o4

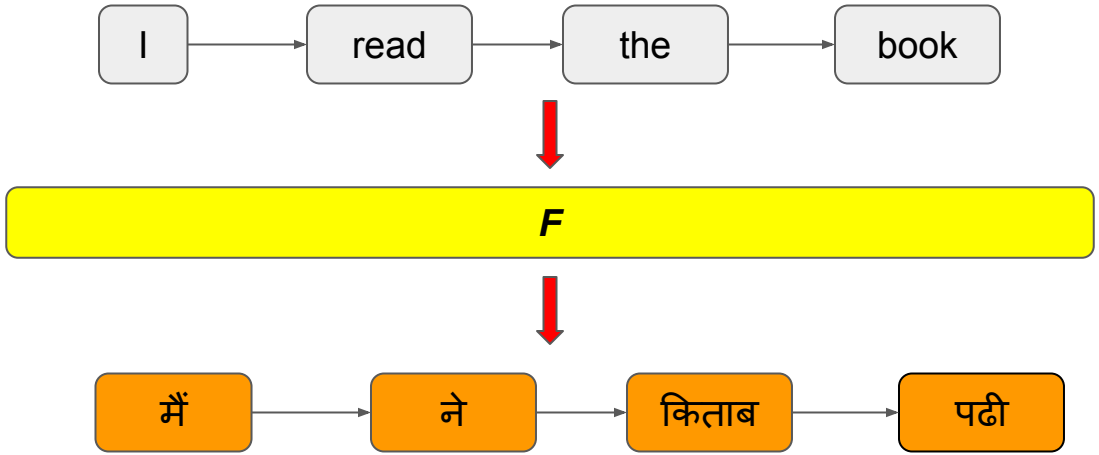
But we do not know the attention weights?
How do we find them?

Let the training data help you decide!!

Idea: Pick the attention weights that maximize the translation accuracy
(more precisely, decrease training data loss)

- *Note ⇒ no separate language model*
- *Neural MT generates fluent sentences*
- *Quality of word order is better*
- *No combinatorial search required for evaluating different word orders:*
  - *Decoding is very efficient compared to PBSMT*
- *Exciting times ahead!*

*Read the entire sequence and predict the output sequence (using function **F**)*



- Length of output sequence need not be the same as input sequence
- Prediction at any time step *t* has access to the entire input
- A very general framework

Sequence to Sequence transformation is a very general framework

Many other problems can be expressed as sequence to sequence transformation

- *Summarization: Article ⇒ Summary*
- *Question answering: Question ⇒ Answer*
- *Image labelling: Image ⇒ Label*
- *Transliteration: character sequence ⇒ character sequence*

Resources for Reading

Books & Articles

- Machine Translation. Pushpak Bhattacharyya (book)
- Neural Machine Translation. Kyunghyun Cho (online)
  - https://devblogs.nvidia.com/parallelforall/introduction-neural-machine-translation-with-gpus/

Thank You!

https://www.cse.iitb.ac.in/~anoopk