COMP 3225

Natural Language Processing

Machine Translation and Encoder-Decoder Models

Stuart E. Middleton

sem03@soton.ac.uk

University of Southampton

Overview

- Machine Translation
- Encoder-Decoder Model

- <break - discussion point>

- Attention
- Beam Search
- MT Systems

Machine Translation

- Machine Translation (MT)
  - Use of computers to translate one language to another
- MT architectures
  - Statistical phrase alignment models <IBM's fast-align> <MOSES>
  - Encoder-Decoder models <this lecture>
  - Transformer models <last lecture>
- Computer aided translation
  - Text >> MT >> Human Correction
  - Post-editing of MT by a human translator

Cite: IBM fast-align https://github.com/clab/fast_align
Cite: MOSES http://www.statmt.org/moses

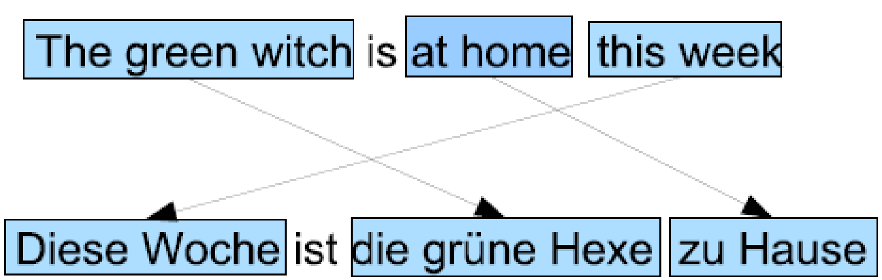
Machine Translation

- Sequence to Sequence (seq2seq) tasks
  - Input >> X >> Sequence of words
  - Output >> Y >> Sequence of words
  - len(X) != len(Y)
- MT can be formulated as a seq2seq task
- seq2seq MT has inspired seq2seq approaches for many NLP tasks
  - Russian text → English text
  - Russian speech → Russian transcript → English transcript
  - Russian Image (e.g. menu) → OCR Russian transcript → English transcript
  - Question → Answer
  - Sentence → Clause (relation + arguments)
  - Document → Abstract

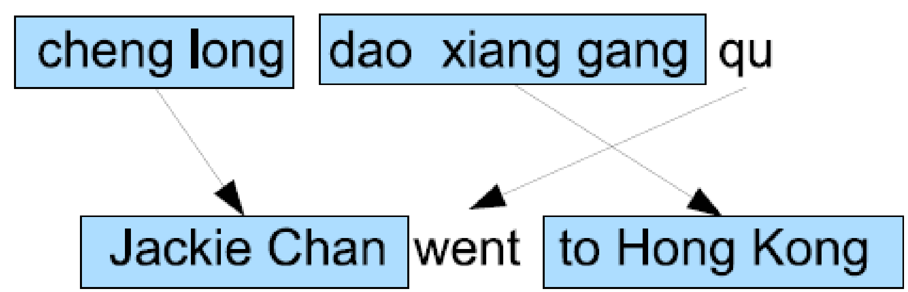
    ... any sequence to sequence problem

Machine Translation

- Human language
  - Universal aspects are true, or statistically mostly true, for all languages
    - noun/verbs
    - greetings
    - polite/rude
  - Translation divergences are where languages differ
    - Idiosyncrasies and lexical differences — e.g. dog translates differently in most languages
    - Systematic differences — e.g. verb → object OR object → verb
    - Linguistic typology studies these differences
- Word Order Typology
  - Subject-Verb-Object (SVO) — English, German, French, Mandarin ...
  - Subject-Object-Verb (SOV) — Japanese, Hindi ...
  - Verb-Subject-Object (VSO) — Arabic, Irish ...



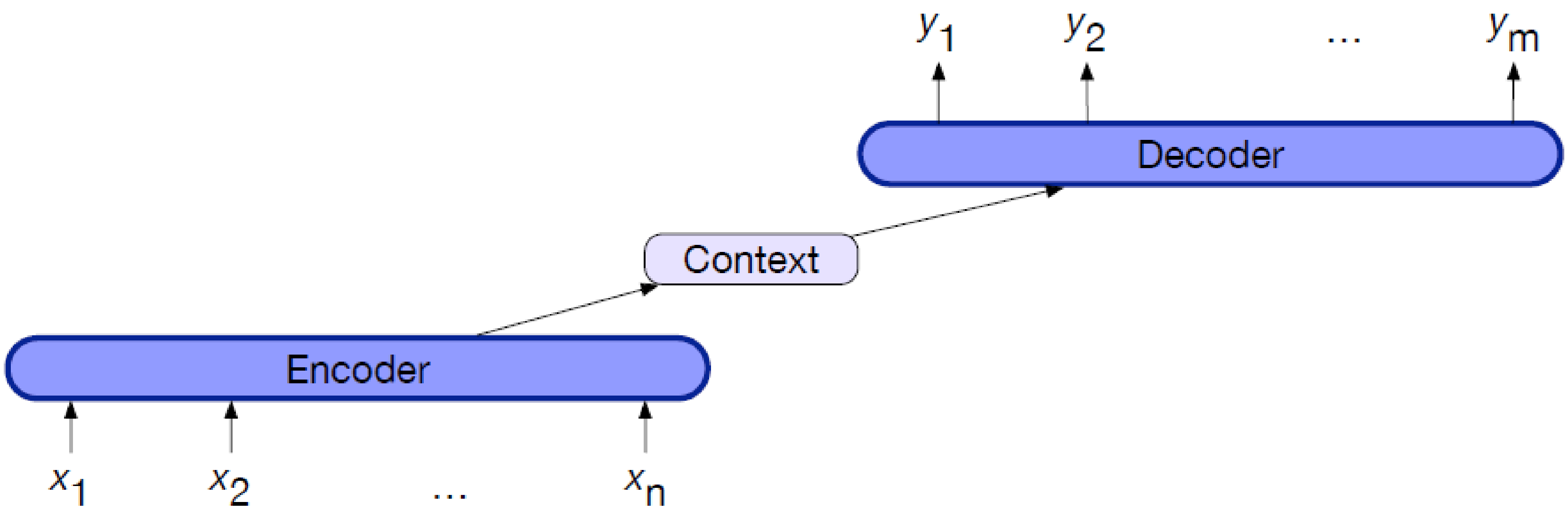
English → German
(adverb before verb in German)



Mandarin → English
(preposition before verb in Mandarin)

Encoder-Decoder Model

- Encoder-decoder model
  - Input sequence X >> encoder >> context vector h
  - context vector h >> decoder >> Output sequence Y
  - encoder = LSTM, GRU, CNN, Transformer ...
  - context vector = last hidden layer of encoder (which may be stacked)



Encoder-Decoder Model

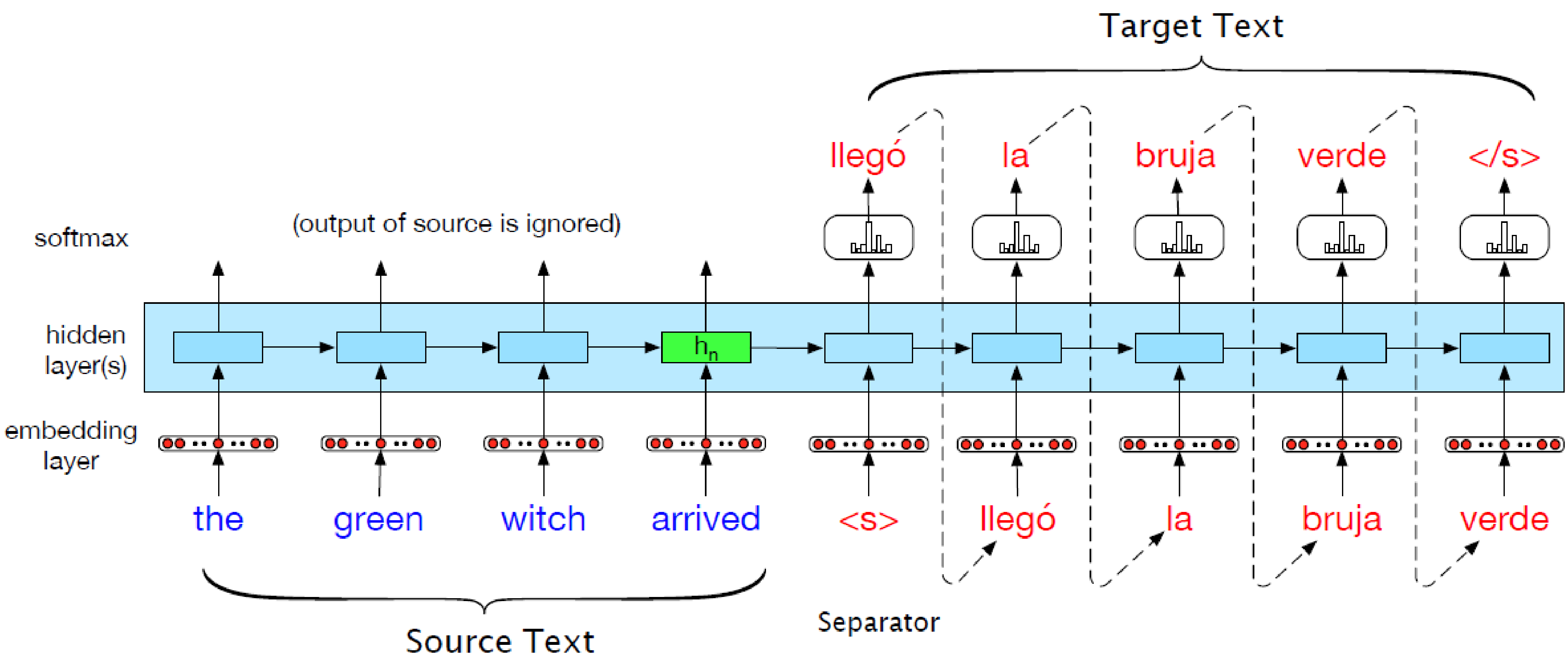
- Encoder-decoder with RNN
    - Language model >> predict next word in sequence Y based on previous word

$$p(y) = p(y_1)p(y_2|y_1)p(y_3|y_1,y_2)...P(y_m|y_1,...,y_{m-1})$$

- Translation model >> predict next target word in sequence Y based on previous target word and the full source sequence X
    - Imagine X consists of source words concatenated with a separator <s> and target words
    - We are interested in text generated after token <s>
    - Separate out sub-sequence X (source text) and sub-sequence Y (target text)

$$p(y|x) = p(y_1|x)p(y_2|y_1,x)p(y_3|y_1,y_2,x)...P(y_m|y_1,...,y_{m-1},x)$$

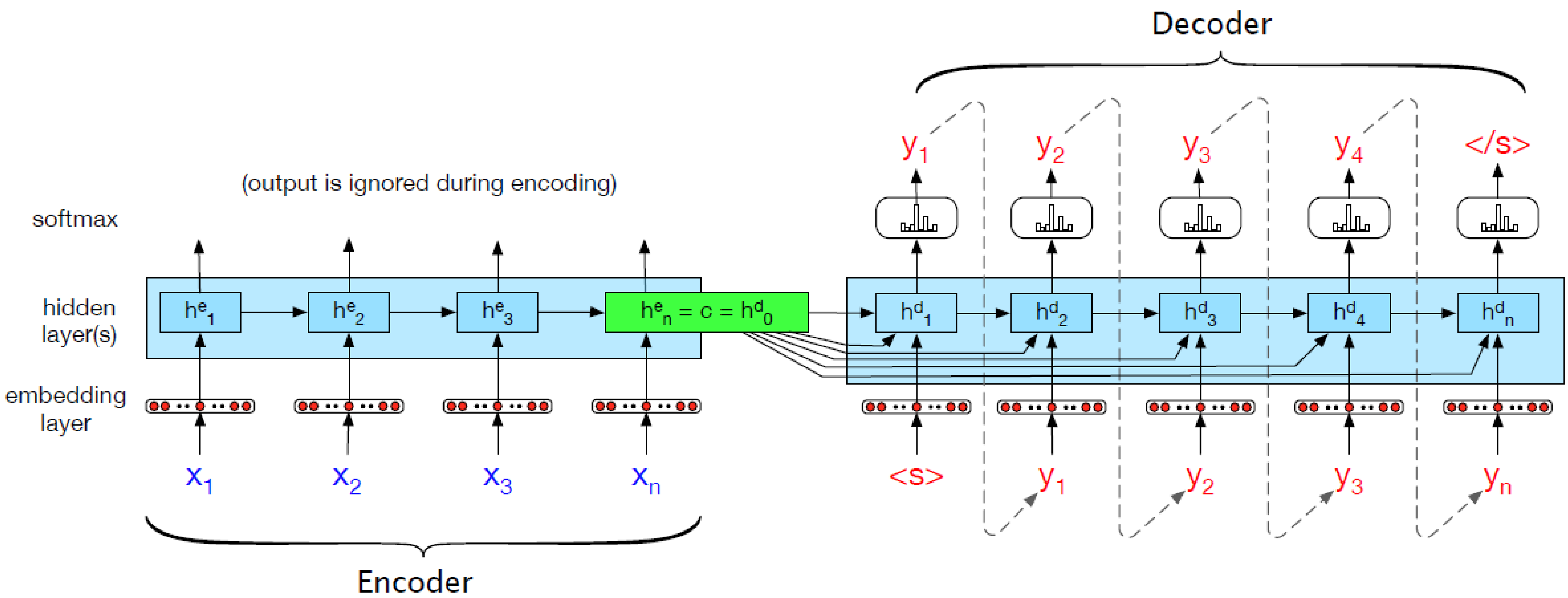
Encoder-Decoder Model



Use final hidden layer h_n (trained with source words X) and the embedding of previous target word Y to predict the next target word

... but to avoid word order typology problems we want to use h_n to guide predictions even after first word Y_1

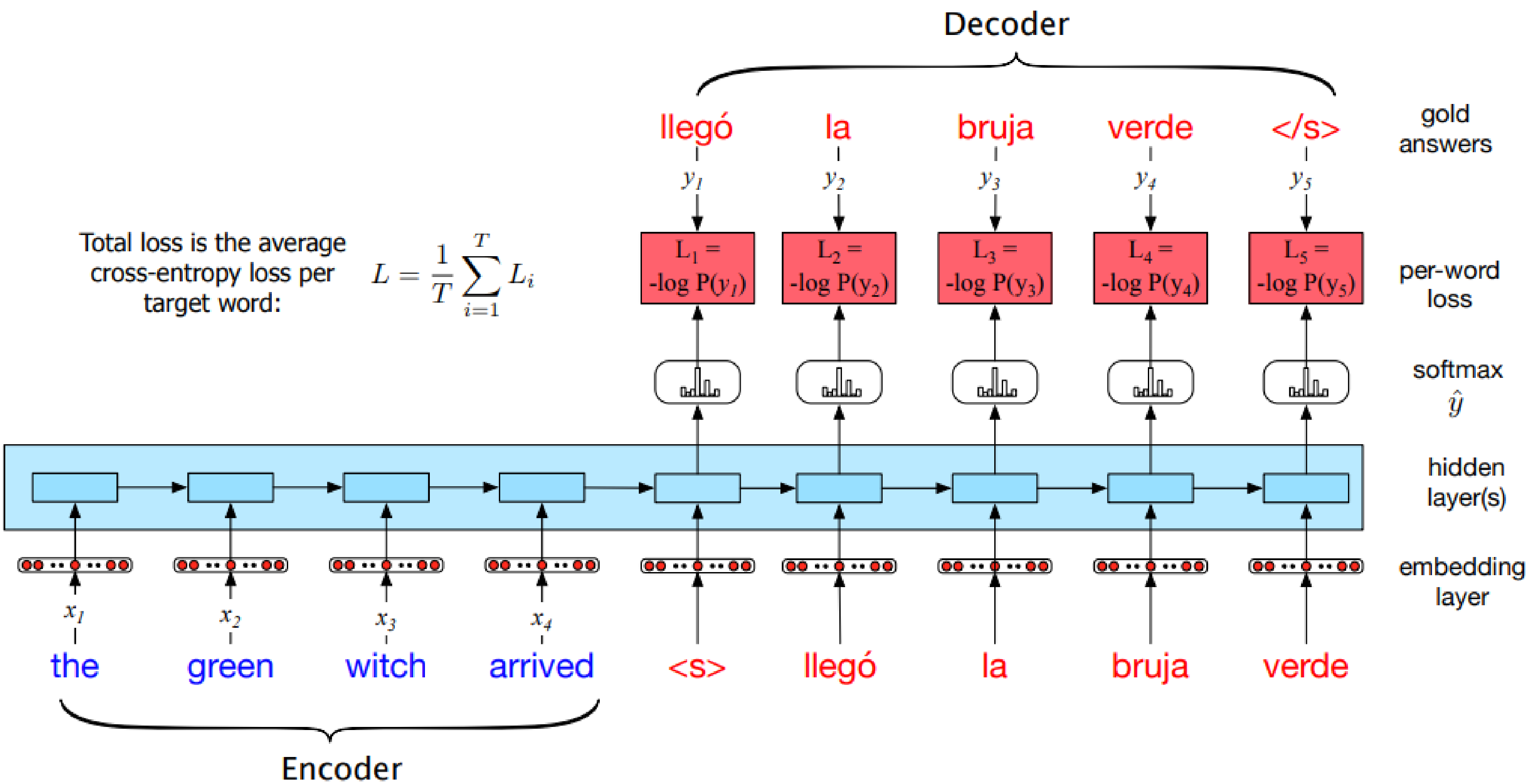
Encoder-Decoder Model



Train encoder hidden layers using X (current word)
encoder hidden layer $h_t = g(h_{t-1},x_t)$, g = activation function (e.g. ReLU)
Final encoder state h_n = context vector c

Train decoder hidden layers using c (<u>at every step</u>) and Y (previous word generated)
decoder hidden layer $h_t = g(c,h_{t-1},y_{t-1})$
... continue until end of sequence predicted

Encoder-Decoder Model



y_t = softmax(c, h_{t-1}, y_{t-1})

During training use teacher forcing (replace predictions with actual target words)

L_i = cross-entropy loss function

Total loss per sentence L is the averaged loss across all target words

Break

- Panopto Quiz - discussion point

- Why do we add a <s> token to target sequence Y?

To indicate the start of the target sentence
Because the decoder needs a previous word embedding to compute a prediction
The decoder associates a special value to the <s> token
To ensure len(X) == len(Y)

Break

- Panopto Quiz - discussion point

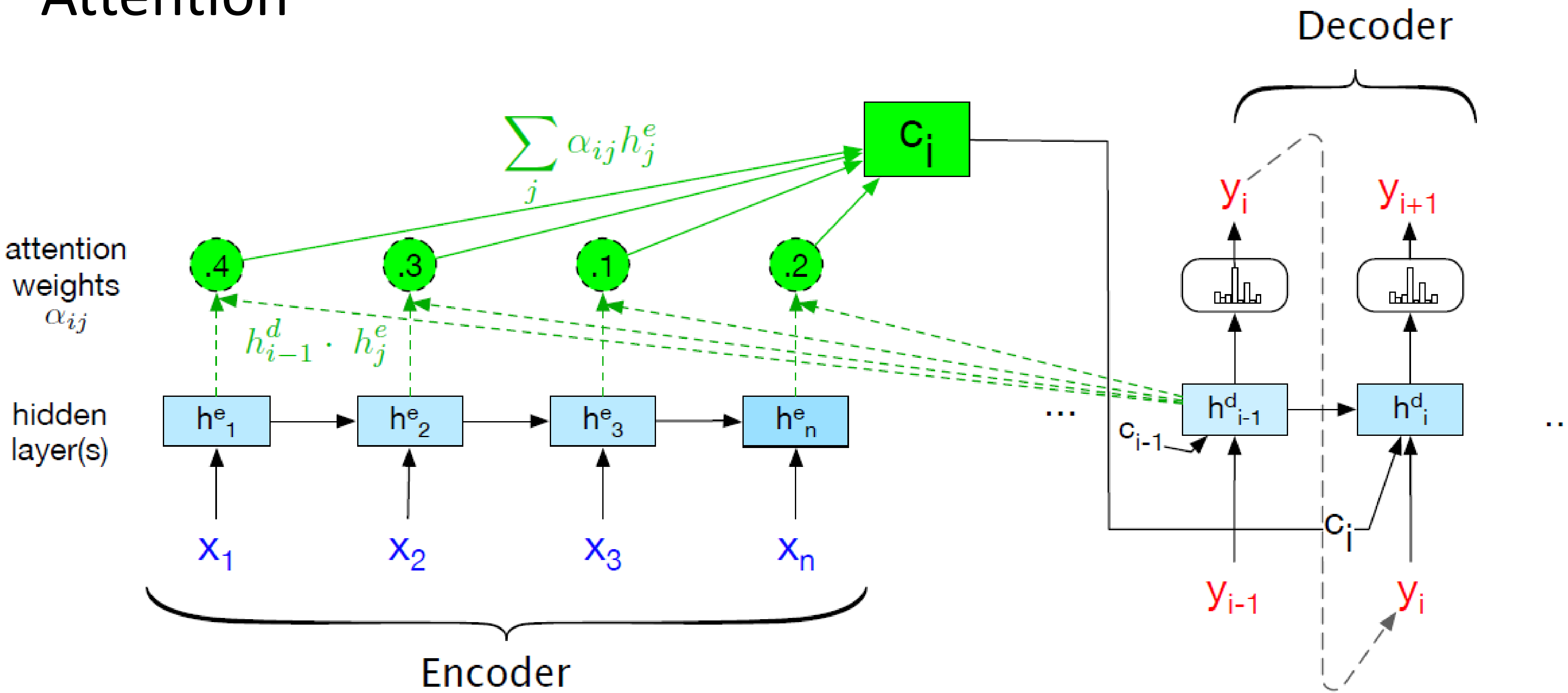
- Why do we add a <s> token to target sequence Y?

To indicate the start of the target sentence >> Y_0 tells us that so why need <s>?
Because the decoder needs a previous word embedding to compute a prediction
>> we do not know the first target word unless we are training!
The decoder associates a special value to the <s> token >> all tokens are treated the same
To ensure len(X) == len(Y) >> No!! the whole point of seq2seq is X and Y can vary in length

Attention

- The static context vector (i.e. last encoder hidden state h_n) needs to represent everything about the full sequence X that might be needed to predict Y to do a good job
  - The contribution to weights of token x_t decays as sequence is processed
  - So, impact of x_{t-1} is not as much as x_t
- Attention mechanisms allow access to all hidden states
  - Attention uses a fixed length vector c calculated from a weighted sum of all encoder hidden states
  - Attention vector replaces the static context vector
- There are various attention mechanisms
  - Dot-product attention (simplest, also called Luong attention)
  - Additive attention (also called Bahdanau attention)
  - Self-attention (also called multi-head attention)
  - <see last lecture>

Attention



$$c_i = \sum_j \alpha_{ij} h^e_j$$

Context vector c is computed using a weighted sum of encoder hidden states

α_{ij} is the attention weight for decoder state i and encoder state j

α is defined by the chosen attention mechanism

Beam Search

- So far we have used greedy decoding using argmax
- At each step t choose the output word y_t that is most likely

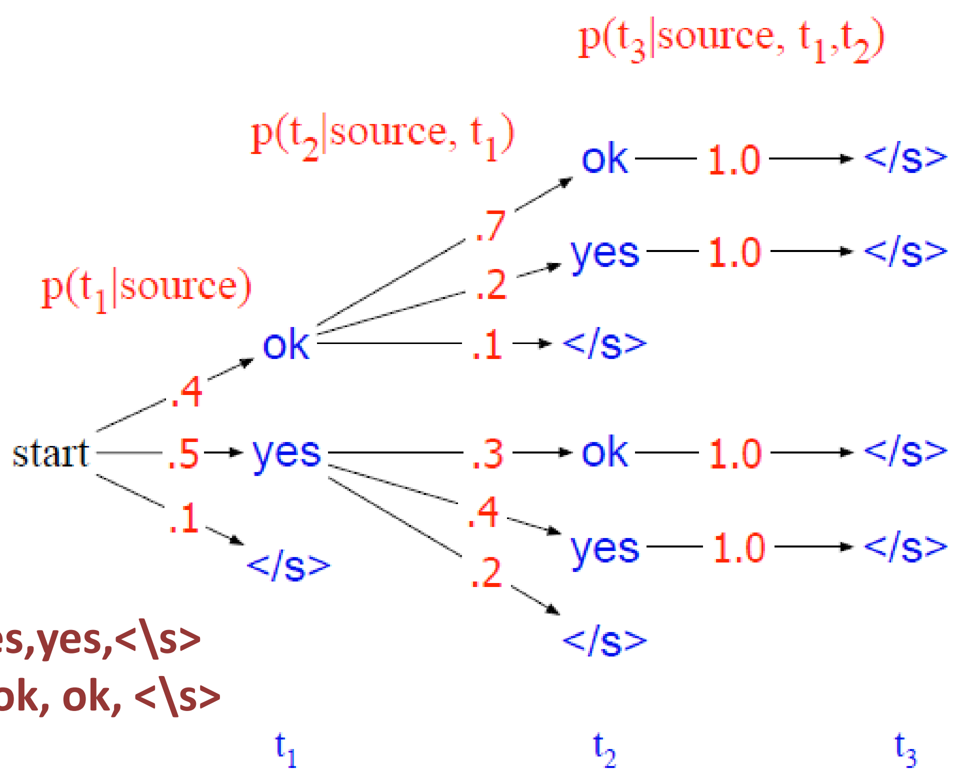
$$\hat{y}_t = \text{argmax}_{\text{w} \in \text{V}} P(w|x, y_1 \ldots y_{t-1})$$

- But ... what if we get it wrong at step t? there is no hindsight to allow us to revisit choices at t-1, t-2 ...
- Dynamic programming (Viterbi algorithm) cannot handle long-distance dependencies between output decisions either

Beam Search

- So far we have used greedy decoding using argmax
- At each step t choose the output word y_t that is most likely

$$\hat{y}_t = \text{argmax}_{w \in V} P(w|x, y_1 \ldots y_{t-1})$$

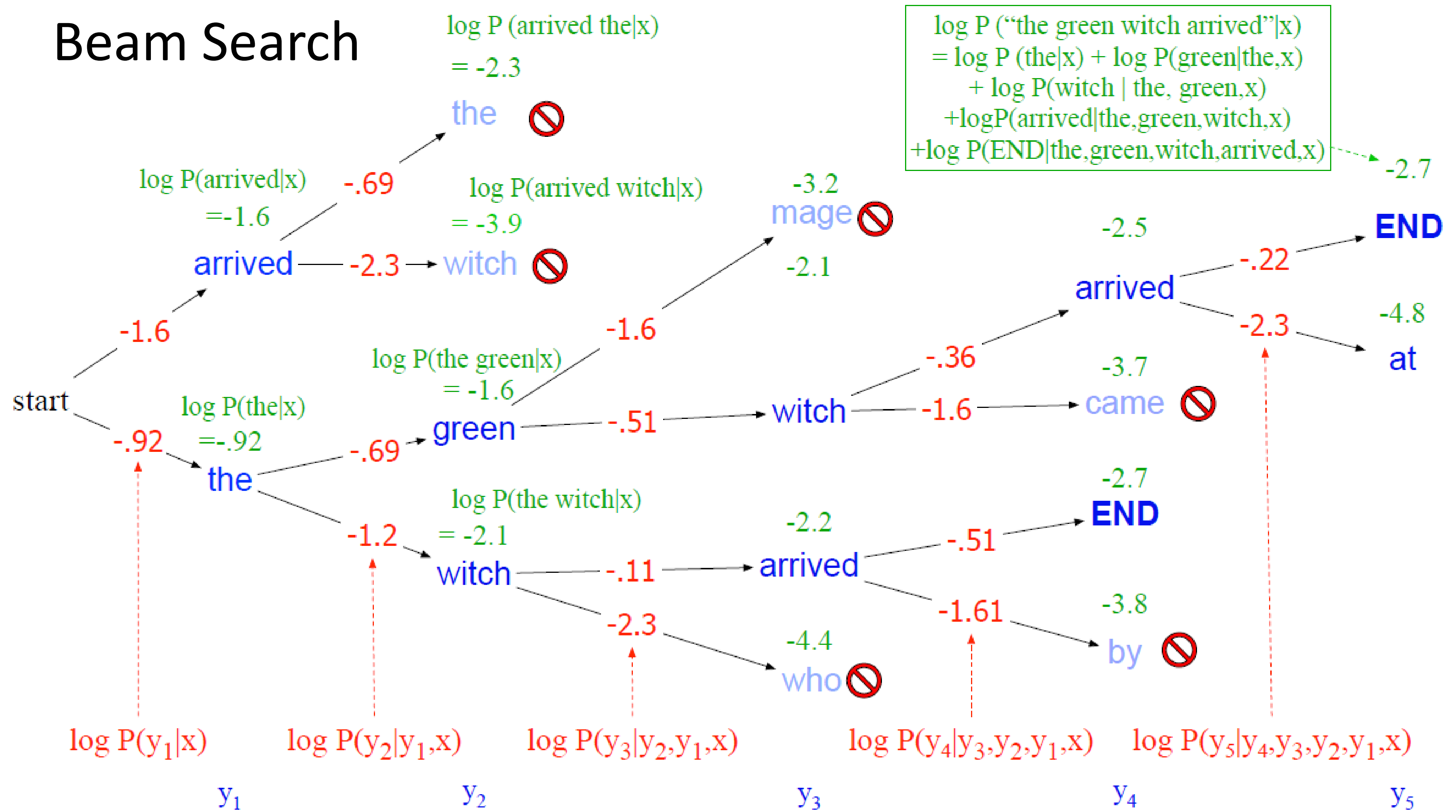


V = yes, ok, </s>
Greedy argmax >> yes,yes,<\s>
Optimal solution >> ok, ok, <\s>

Beam Search

- The beam search decoder keeps a memory of the k-best sequence options (called hypotheses) at any decoding step
  - The memory size k is called the beam width
  - At each step (target word prediction) all k hypotheses are extended by V predicted tokens
  - The best k sequences from k x V hypotheses are then selected for memory

Beam Search



k = beam width = 2

V = the, arrived, green, witch, mage, who, came, by, at, END

At each step the log probability computed for each new sequence, keeping best k

MT Systems

- Tokenization
  - seq2seq usually has a fixed vocabulary (e.g. 50k limited by GPU memory)
  - Use Byte Pair Encoding (BPE) or WordPiece
- Encoder-decoder model
  - seq2seq + basic attention
  - Transformer + self-attention (better results)
  - Fast results >> Greedy decoder
  - Best results >> Beam decoder
- Training data
  - Parallel corpus
    - Bitext >> manually translated >> Limited in size (1,000's to 100,000's of sentences)
    - Bitext >> automatic alignment >> Large but noisy (1,000,000's of sentences)
  - Monolingual corpus
    - Very large corpus but not suitable for seq2seq
    - Backtranslation >> create very large synthetic bitext from monolingual dataset
      - high quality small bitext (target) → train target to source MT#1
      - very large target corpus → MT#1 → very large bitext → train source to target MT#2
    - Variable Auto-Encoders (VAE)

MT Systems

- Evaluation of MT
  - Human assessment
  - BLEU very popular (function of n-gram precision over whole sentence)
  - Alternatives are P, R, NIST, TER, METEOR
  - see Conference on Machine Translation (WMT) - e.g. WMT'19

Required Reading

- Machine Translation
  - Jurafsky and Martin, Speech and Language Processing, 3rd edition (online)
    >> chapter 11
- Encoder-Decoder (optional)
  - Aurelien Geron, Hands-On Machine Learning with Scikit-Learn and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems, O’Reilly , 2017
    >> Chapter 16 'an encoder-decoder network for neural machine translation'

Questions

- Panopto Quiz - 1 minute brainstorm for interactive questions

    Please write down in Panopto quiz in **1 minute** two or three questions that you would like to have answered at the next interactive session.

    Do it **right now** while its fresh.

    Take a screen shot of your questions and **bring them with you** at the interactive session so you have something to ask.