CMPE 297, INSTRUCTOR: JORJETA JETCHEVA

# MACHINE TRANSLATION

# MACHINE TRANSLATION OVERVIEW

- Typically focuses on practical tasks
  - Information access (articles, news, Wikipedia, instructions) - Google translate is used on billions of words per day across 100+ languages
  - Localization (adapting content to different locations), e.g., HR policies translated across all countries where a multi-national has employees
    - Typically computer-aided translation (CAT) is employed to facilitate human translators by providing an initial draft
  - Real-time translation - on-the-fly speech, street signs, menus (latter two combined with OCR)
- Translation of literature and poetry is very difficult, even for humans!

# MACHINE TRANSLATION OVERVIEW

- Machine translation involves coming up with an output sequence based on an input sequence, but where the number or order of words may change from one language to another

大会/General Assembly 在/on 1982年/1982 12月/December 10日/10 通过了/adopted 第37号/37th 决议/resolution， 核准了/approved 第二次/second 探索/exploration 及/and 和平peaceful 利用/using 外层空间/outer space 会议/conference 的/of 各项/various 建议/suggestions 。

On 10 December 1982 , the General Assembly adopted resolution 37 in which it endorsed the recommendations of the Second United Nations Conference on the Exploration and Peaceful Uses of Outer Space .

| | | | |
|---|---|---|---|
| English: | *He wrote a letter to a friend* | | |
| Japanese: | *tomodachi ni* | *tegami-o* | *kaita* |
| | friend | to letter | wrote |

**In the example above, we can see that**

- The order of the words is different between the two languages
- Some words (e.g., "various") are present in Chinese but not English; "the" is present in English but not Chinese
- Chinese does not mark the plurality of nouns ("various" is used below to indicate plurality)

# LANGUAGE DIVERGENCIES AND TYPOLOGY

- Lexical and one off (idiosyncratic) differences between languages, e.g., word for dog is different in different languages, need to be dealt with individually
- Linguistic typology deals with systematic cross-language similarities and differences, e.g., verb comes before or after direct object depending on the language

# WORD ORDER TYPOLOGY

- SVO (Subject-Verb-Object) languages, e.g., English, Mandarin (verb usually comes before the object)
- SOV languages, e.g., Hindi and Japanese (verb comes at the end of basic clauses)
- VSO languages, Irish and Arabic (verb comes first)
- VO languages have prepositions, whereas OV languages have postpositions
  - E.g., the “preposition” is after the noun (postposition) as in “friend to” in the Japanese example

English: *He wrote a letter to a friend*

Japanese: *tomodachi ni tegami-o kaita*
friend to letter wrote

Arabic: *katabt risāla li ṡadq*
wrote letter to friend

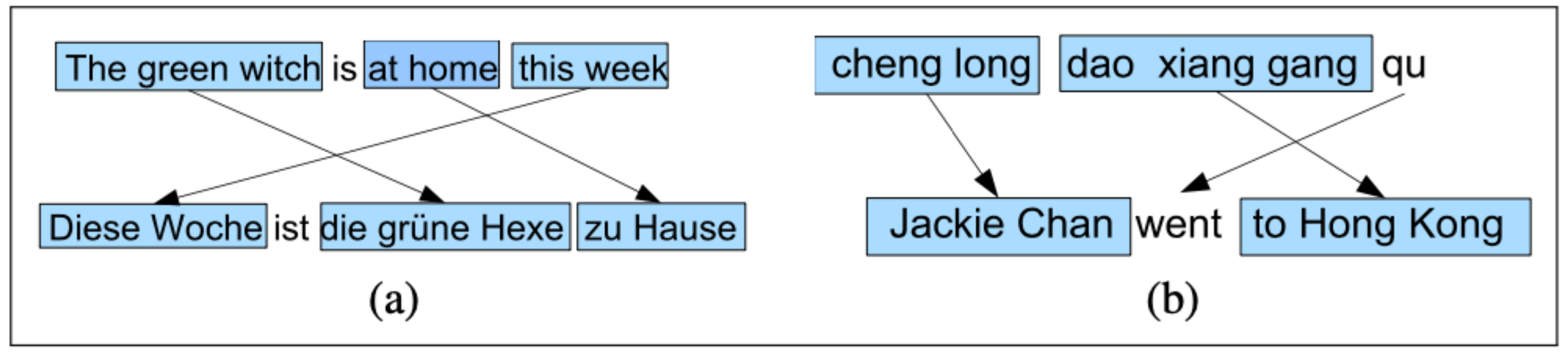



**Figure 10.1** Examples of other word order differences: (a) In German, adverbs occur in initial position that in English are more natural later, and tensed verbs occur in second position. (b) In Mandarin, preposition phrases expressing goals often occur pre-verbally, unlike in English.

# LEXICAL DIVERGENCIES

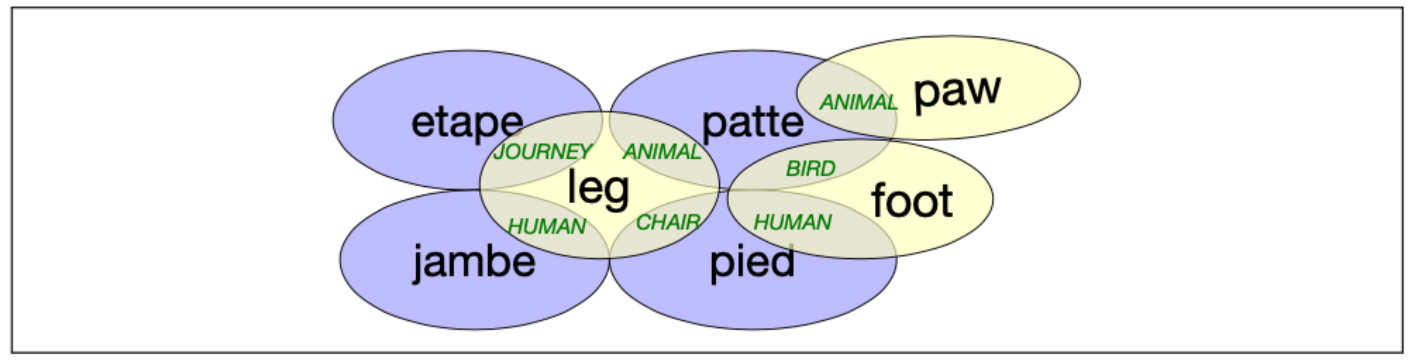



**Figure 10.2** The complex overlap between English *leg*, *foot*, etc., and various French translations as discussed by Hutchins and Somers (1992).

- Examples of lexical divergencies
  - Some languages have multiple words for the same thing used in different contexts, e.g., wall inside a building vs. outside has a different word in German
  - Leg of a journey vs. leg of a person or animal may have different words
  - Older sister vs. younger sister may have different words in some languages, and not in others
  - Some languages mark (change) nouns based on whether they are plural or singular vs. others don't
  - In some languages adjectives have an associated gender based on the noun they modify
- There may be lexical gaps: a word in one language may not have a corresponding word in another language
- Verb-framed vs. satellite-framed languages (whether the verb indicates the direction or motion or some of the other (satellite) words indicate the direction of motion)
  - English is satellite-framed, e.g., direction of motion is marked on the particle "out" in the example to the right

English: *The bottle floated out.*

Spanish: La botella salió flotando.
The bottle exited floating.

# MORPHOLOGICAL TYPOLOGY

- Morphologically rich languages may have multiple forms of each word
  - Extracting the morpheme (root) from each word (e.g., to map to the same token) can be difficult esp. since some words may contain multiple morphemes
- Translation which involves a morphologically rich language requires the use of subword tokenization

# REFERENTIAL DENSITY

[El jefe]$_i$ dio con un libro. $\emptyset_i$ Mostró a un descifrador ambulante.
[The boss] came upon a book. [He] showed it to a wandering decoder.

- Referential density refers to the extent to which a language allows omissions, e.g., the omission of pronouns
- Pro-drop languages are ones that can omit pronouns
  - Some languages are more aggressive about dropping pronouns than others (e.g., Japanese and Chinese omit more than Spanish)
  - Languages that tend to use more pronounce are said to be more referentially dense
- Languages that are referential sparse are called cold (vs. hot for those with high referential density)
- Sparse languages require the person hearing/reading to perform a lot more inference in order to figure out what's going on
- Translating from pro-drop languages to non-pro-drop languages is challenging because we first need to identify that something is missing, and then figure out what it would refer to if it was present

# ENCODER-DECODER MODEL

# THE ENCODER-DECODER MODEL - REVIEW FROM PREVIOUS LECTURE

- Encoder-decoder networks (aka sequence-to-sequence networks) are models that generate output sequences
- Example applications:
  - Machine translation
  - Summarization
  - Question Answering

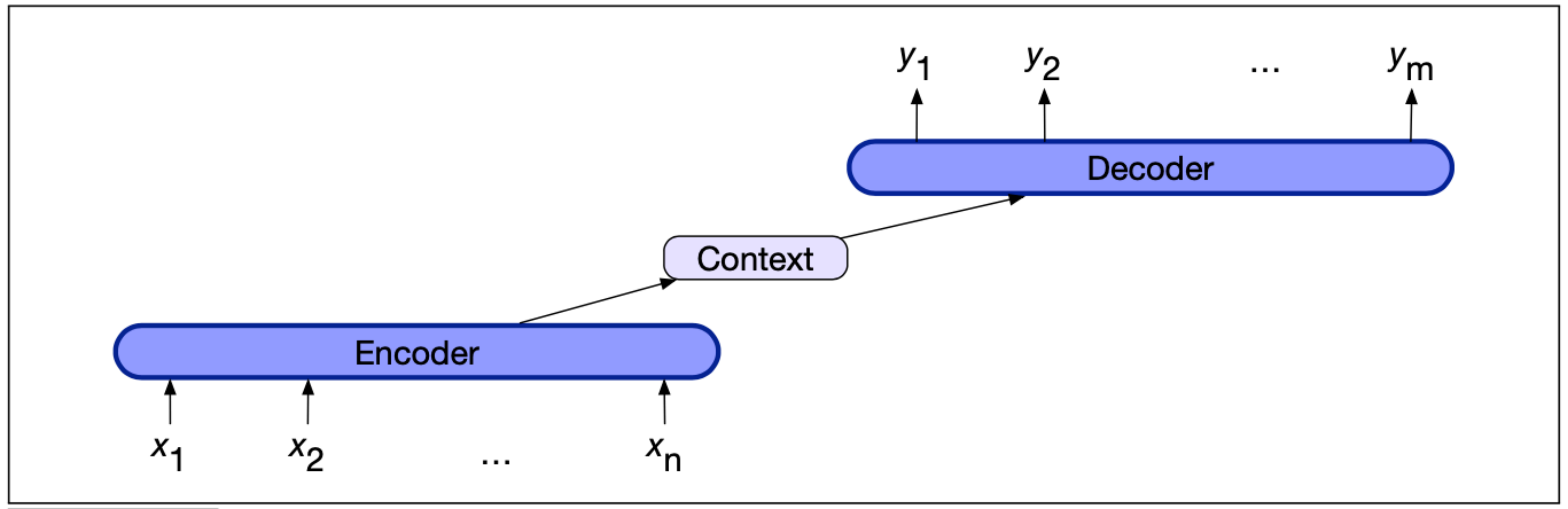



**Figure 10.3** The encoder-decoder architecture. The context is a function of the hidden representations of the input, and may be used by the decoder in a variety of ways.

- Encoder-decoder components:
  - An encoder: accepts an input sequence $x_1^n$, and generates a corresponding sequence of contextualized representations, $h_1^n$
  - A context vector, $c$, which is a function of $h_1^n$, and passes the extracted input information to the decoder
  - A decoder which accepts the context as input, and then generates some task-specific (arbitrary length) output sequence
- The encoder and decoder can be based on any kind of sequence-based architecture, e.g., RNN, LSTM or Transformer for example (the encoder and decoder can also use different architectures from each other)

# ENCODER-DECODER BASED ON RNNS

- In a typical RNN, we compute the value of a hidden state at time $t$, by applying the activation function to the hidden state from the previous time step, $t-1$, and the current input, $x_t$:

  $\mathbf{h}_t = g(\mathbf{h}_{t-1}, \mathbf{x}_t)$, where $g$ can be ReLU, $tanh$, etc.

  Then we apply a softmax over the hidden layer output, $\mathbf{h}_t$, to derive the output $\mathbf{y}_t$ at the current time $t$:

  $\mathbf{y}_t = f(\mathbf{h}_t)$

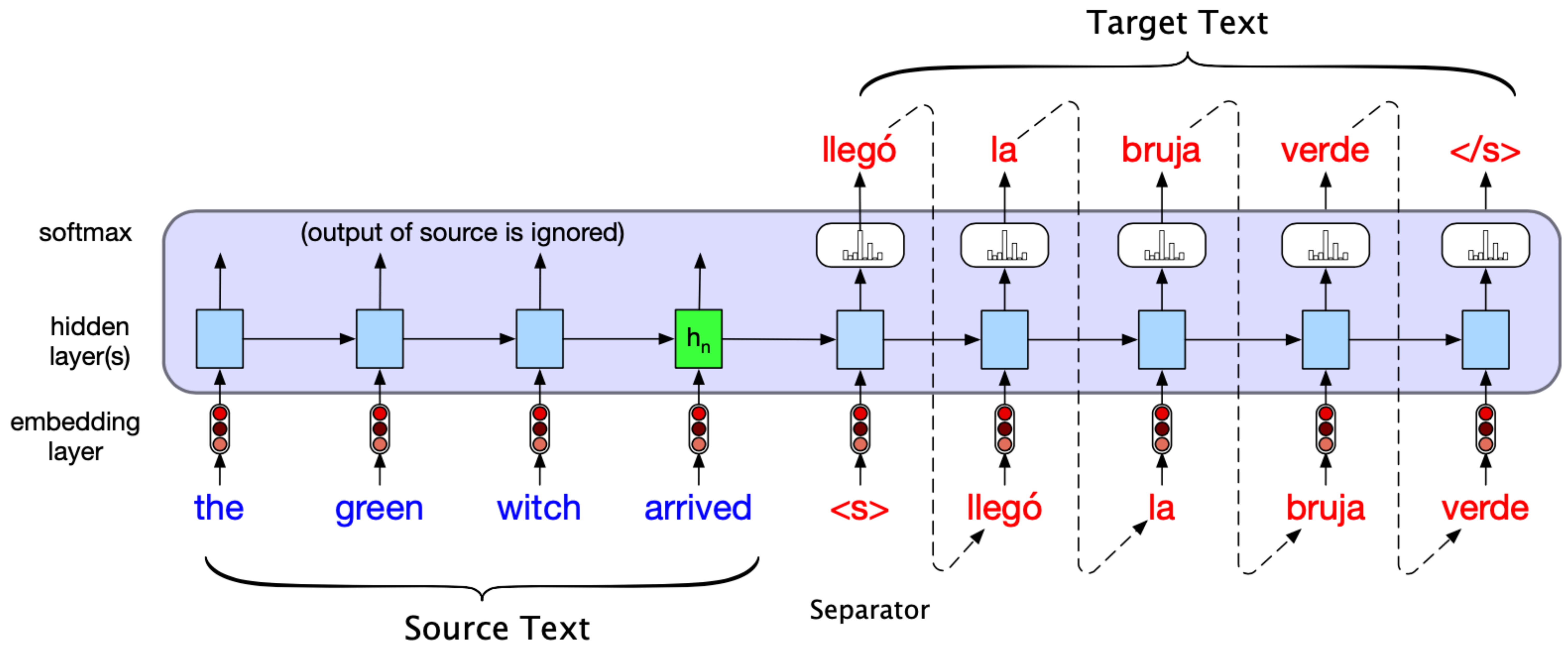



**Figure 10.4** Translating a single sentence (inference time) in the basic RNN version of encoder-decoder approach to machine translation. Source and target sentences are concatenated with a separator token in between and the decoder uses context information from the encoder's last hidden state.

# ENCODER-DECODER BASED ON RNNS (2)

- Input: source text  as input, followed by a separator token, followed by the target text
- Below superscript $e$ indicates encoder, and superscript $d$ indicates decoder

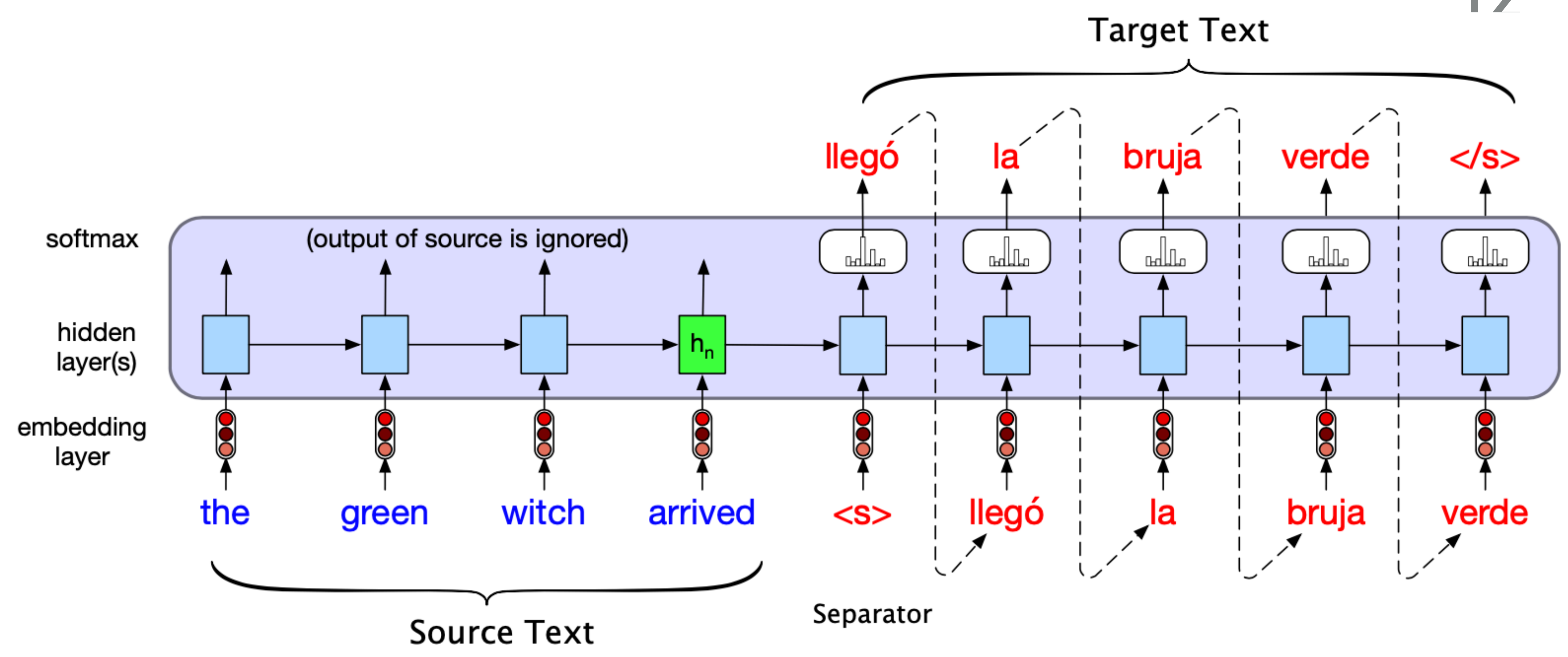
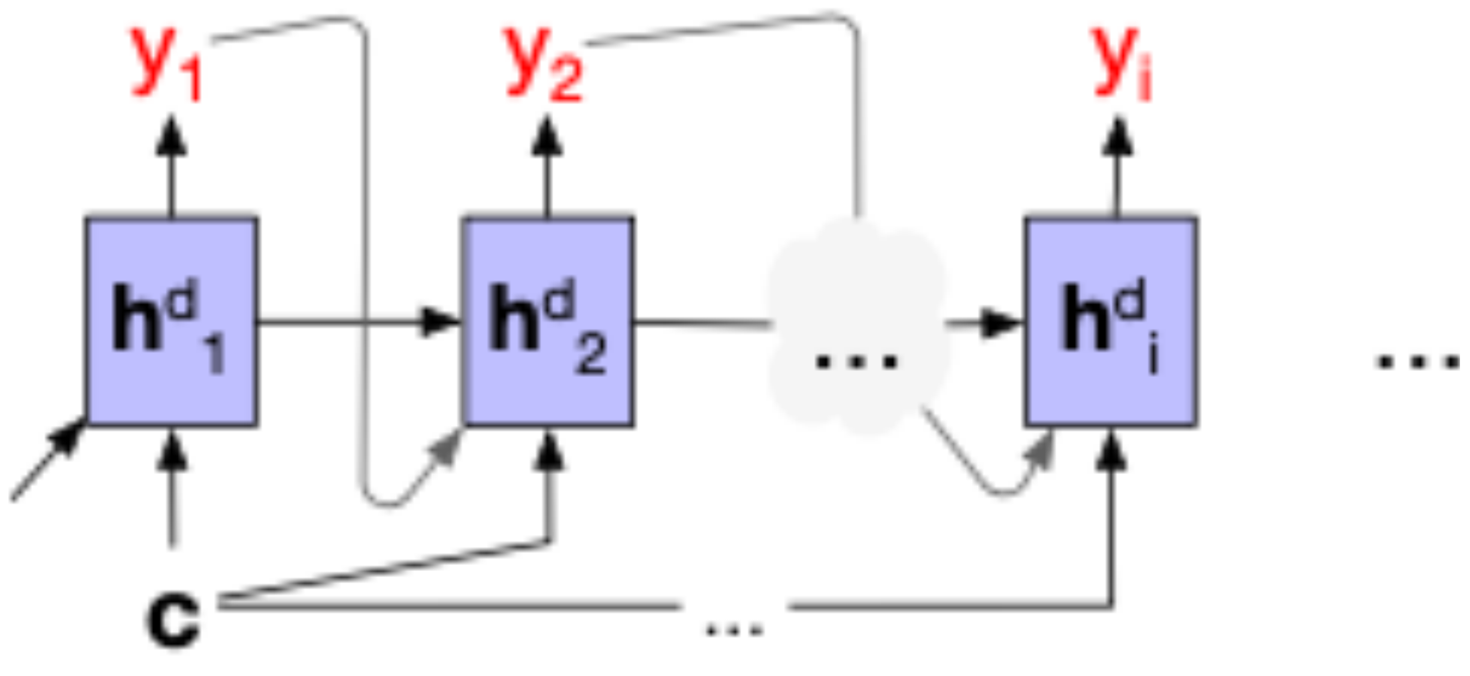



**Figure 10.4** Translating a single sentence (inference time) in the basic RNN version of encoder-decoder approach to machine translation. Source and target sentences are concatenated with a separator token in between and the decoder uses context information from the encoder's last hidden state.

To perform translation, we do the following:

- We perform the encoder computations for all the input tokens until we get to the end of the source text
- Then we start auto-generating text based on the last hidden layer state computed by the encoder $\mathbf{h}_n^e$ (aka context $c$), which is a compressed representation of the input and is passed to the decoder;
- The decoder uses $\mathbf{h}_n^e$ to initialize its first hidden state ($\mathbf{h}_0^d$)
- Each generated word is conditioned on the previous hidden state $\mathbf{h}_{t-1}^d$ , and the output of the previous hidden state (output $\hat{y}_{t-1}$: the embedding of the last generated word) {in some cases also conditioned on the context the context $c$ as well!)
- Text generation stops once we generate an end of sentence marker

# ENCODER-DECODER BASED ON RNNS (CONT.)




- The influence of the context vector is diluted more, the longer the sequence of generated words gets
- As a result, in some architectures, the context is passed into each step of the decoding process as an extra parameter: $\mathbf{h}_t^d = g(\hat{y}_{t-1}, \mathbf{h}_{t-1}^d, \mathbf{c})$
- The full computation looks as follows, where $\hat{y}_t$ is the most likely output at each time step:

$$
\begin{aligned}
\mathbf{c} &= \mathbf{h}_n^e \\
\mathbf{h}_0^d &= \mathbf{c} \\
\mathbf{h}_t^d &= g(\hat{y}_{t-1}, \mathbf{h}_{t-1}^d, \mathbf{c}) \\
\mathbf{z}_t &= f(\mathbf{h}_t^d) \\
y_t &= \text{softmax}(\mathbf{z}_t) \\
\hat{y}_t &= \text{argmax}_{w \in V} P(w|x, y_1 \ldots y_{t-1})
\end{aligned}
$$

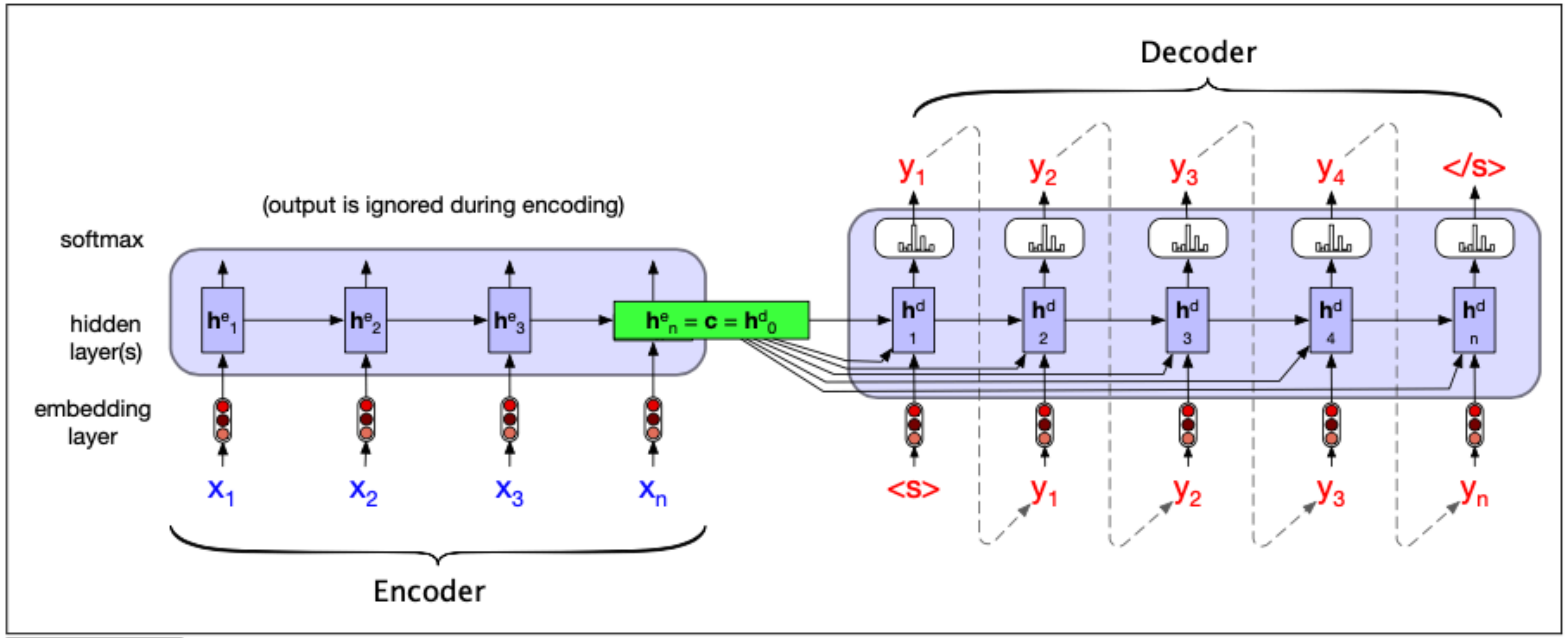



**Figure 10.5** A more formal version of translating a sentence at inference time in the basic RNN-based encoder-decoder architecture. The final hidden state of the encoder RNN, $h_n^e$, serves as the context for the decoder in its role as $h_0^d$ in the decoder RNN.

# TRAINING THE ENCODER-DECODER

- We use text sequences and their corresponding translations, separated with a separator token
- The networks is trained to predict the next word using teacher forcing in the decoder
  - We use the ground truth previous word rather than the previously predicted word by the model)
- Note that during inference, the decoder does use its previously predicted word as input into the next time step

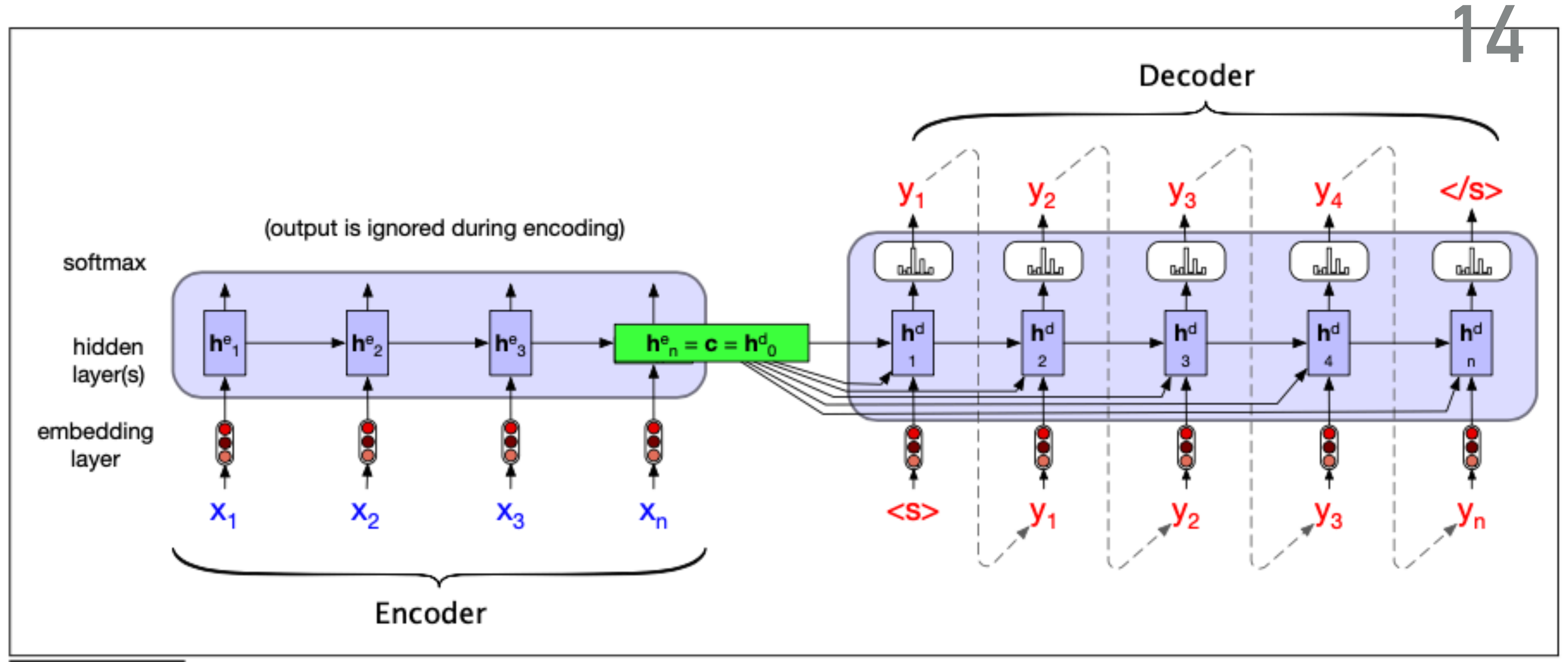



**Figure 10.5** A more formal version of translating a sentence at inference time in the basic RNN-based encoder-decoder architecture. The final hidden state of the encoder RNN, $h^e_n$, serves as the context for the decoder in its role as $h^d_0$ in the decoder RNN.

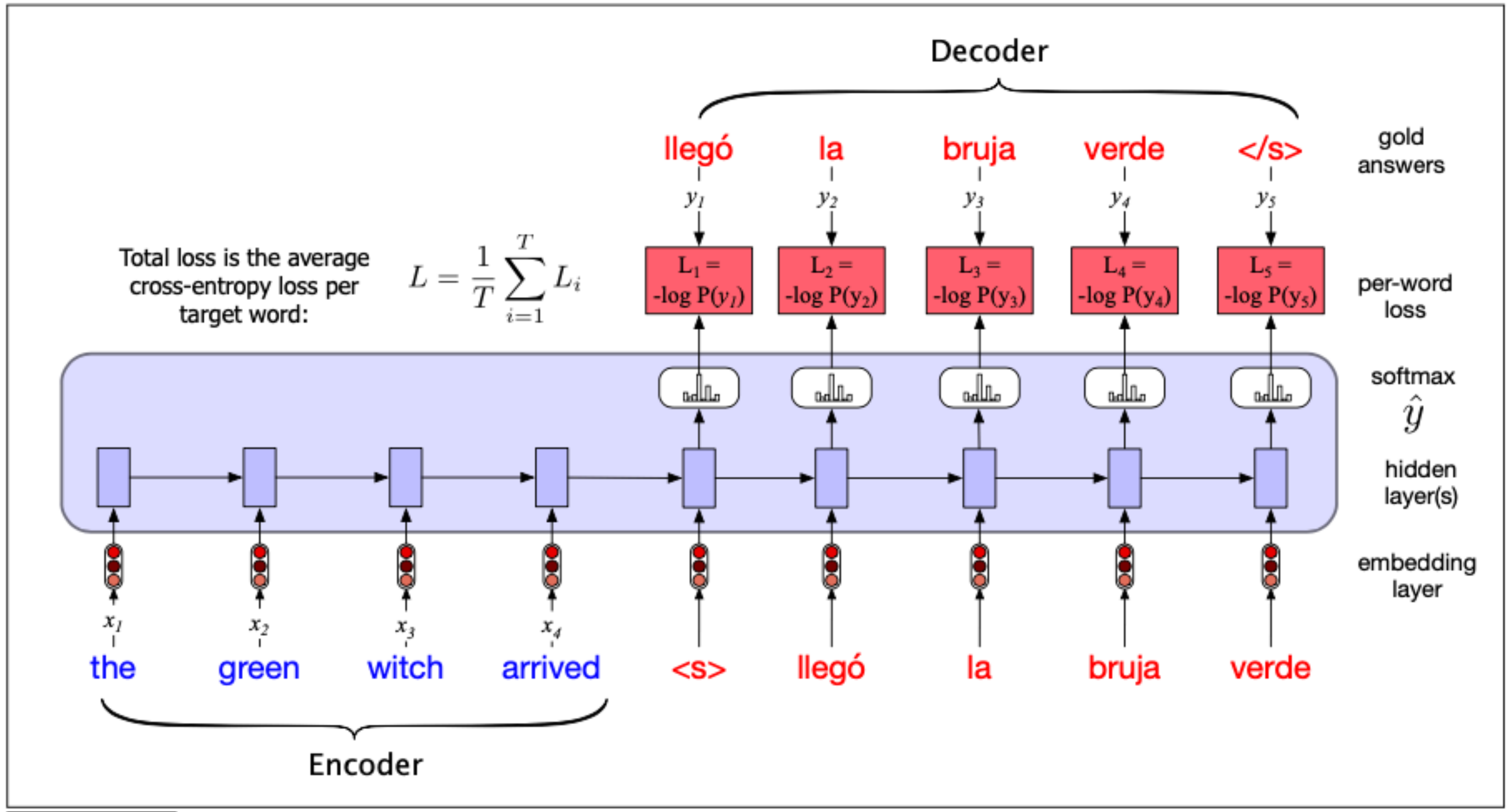



**Figure 10.7** Training the basic RNN encoder-decoder approach to machine translation. Note that in the decoder we usually don't propagate the model's softmax outputs $\hat{y}_t$, but use **teacher forcing** to force each input to the correct gold value for training. We compute the softmax output distribution over $\hat{y}$ in the decoder in order to compute the loss at each token, which can then be averaged to compute a loss for the sentence.

# ATTENTION IN A TRANSLATION TASK

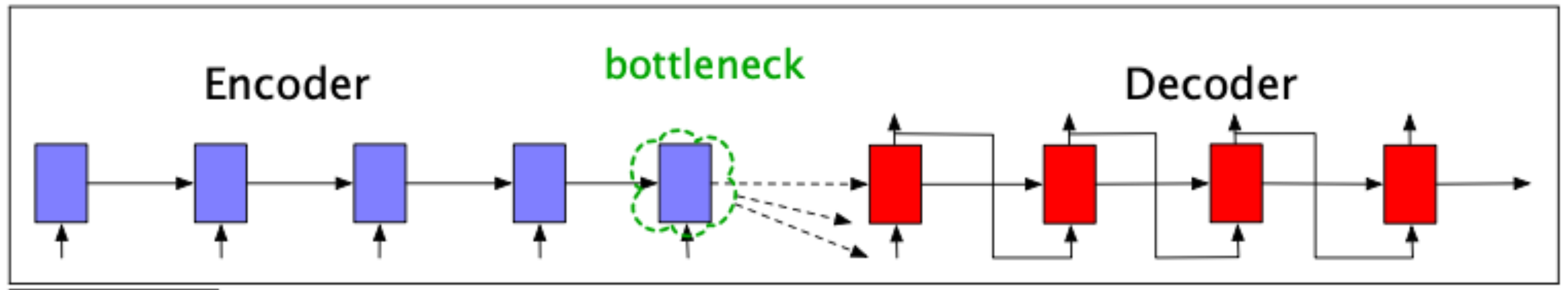



**Figure 10.8** Requiring the context $c$ to be only the encoder's final hidden state forces all the information from the entire source sentence to pass through this representational bottleneck.

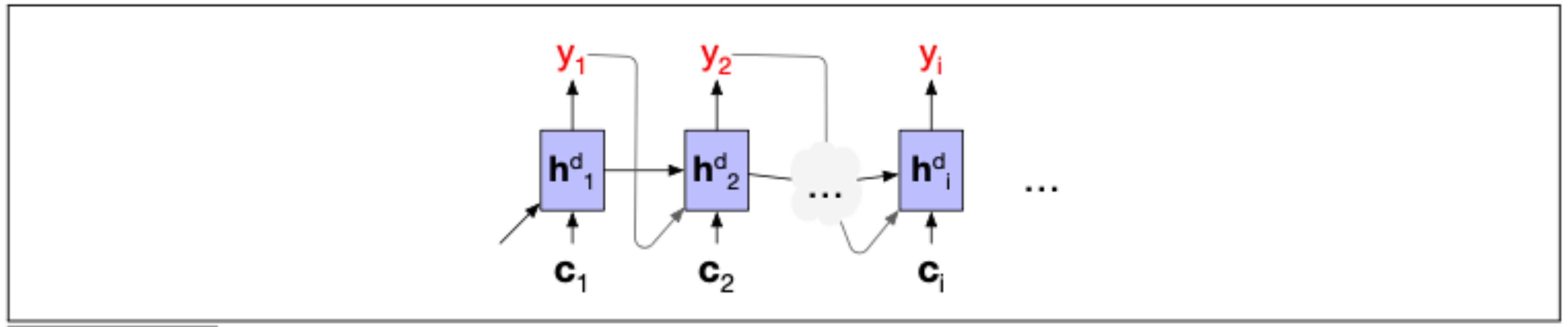



**Figure 10.9** The attention mechanism allows each hidden state of the decoder to see a different, dynamic, context, which is a function of all the encoder hidden states.

- The final hidden state in the encoder ($\mathbf{h}^e_n$) is a bottleneck because it limits the amount of information about the input that is passed to the decoder: $\mathbf{c} = \mathbf{h}^e_n$
- The attention mechanism allows information from all hidden state of the encoder to be passed to the decoder, i.e., the context now can be derived as follows: $\mathbf{c} = f(\mathbf{h}^e_1 \ldots \mathbf{h}^e_n)$
- We use a weighted sum of the hidden state vectors to arrive at a fixed size context, $\mathbf{c}_i$, for each time step $i$
  - The number of hidden states varies (based on the length of the input) so don't want to just concatenate them to form the context
- The overall computation of each hidden state at the decoder is as follows:

$$\mathbf{h}^d_i = g(\hat{y}_{i-1}, \mathbf{h}^d_{i-1}, \mathbf{c}_i)$$

(we are conditioning the computation on the output of the previous time step, the previous hidden state, and the context vector)

# COMPUTING THE CONTEXT VECTOR WITH ATTENTION

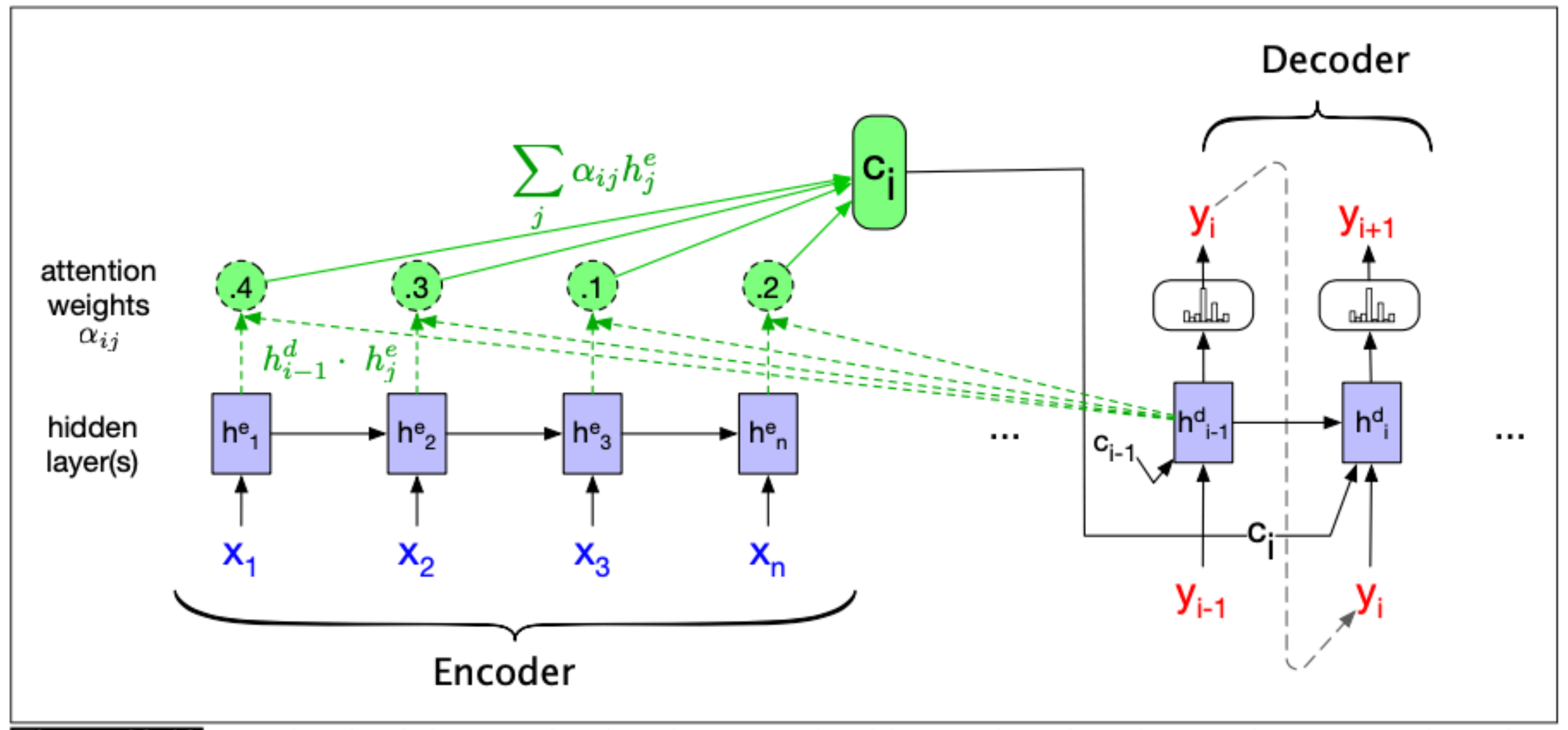



**Figure 10.10** A sketch of the encoder-decoder network with attention, focusing on the computation of $\mathbf{c}_i$. The context value $\mathbf{c}_i$ is one of the inputs to the computation of $\mathbf{h}^d_i$. It is computed by taking the weighted sum

- The context vector is a weighted sum of the hidden state vectors which allows us to arrive at a fixed size context, $\mathbf{c}_i$, for each time step $i$, i.e.:

  $\mathbf{c}_i = \sum_j \alpha_{ij} \mathbf{h}_j^e$ computed over all the hidden state vectors in the encoder

  The weights $\alpha_{ij}$ indicate how relevant each encoder state is to the current state of the decoder (in the current time step $i$), which we compute using vector similarity (dot product):

  $score(\mathbf{h}_{i-1}^d, \mathbf{h}_j^e) = \mathbf{h}_{i-1}^d \cdot \mathbf{h}_j^e$ we compare the decoder state in $\mathbf{h}_{i-1}^d$ to each of the hidden states in the encoder

  We then softmax the scores in order to convert the scores to a probability format (weights sum up to 1):

$$\alpha_{ij} = softmax(score(\mathbf{h}_{i-1}^d, \mathbf{h}_j^e)) \qquad \forall j \in e$$
$$= \frac{exp(score(\mathbf{h}_{i-1}^d, \mathbf{h}_j^e))}{\sum_k exp(score(\mathbf{h}_{i-1}^d, \mathbf{h}_j^e))}$$

- Some models use a mode sophisticated scoring function, where the score has its own set of learnable weights, $\mathbf{W}_s$, which allow it to adapt the score to the specific application and to have a different number of dimensions in the encoder vs. the decoder: $score(\mathbf{h}_{i-1}^d, \mathbf{h}_j^e) = \mathbf{h}_{i-1}^d \mathbf{W}_s \mathbf{h}_j^e$

# ENCODER-DECODER WITH TRANSFORMERS FOR TRANSLATION

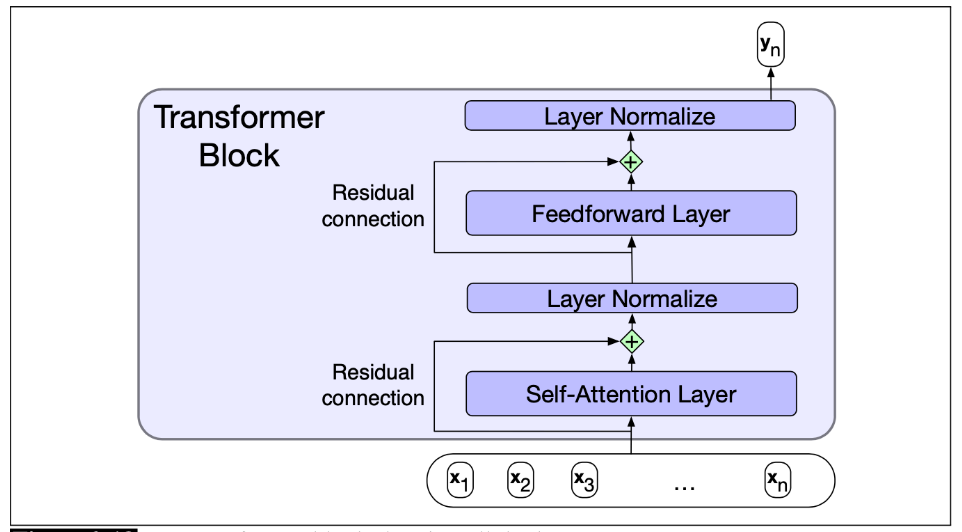



**Figure 9.18** A transformer block showing all the layers.

There are some differences from the original transformer architecture we discussed earlier (e.g., Fig. 9.18)

- Cross-attention layer (aka encoder-decoder attention)
  - Multi-headed self-attention where the queries come from the decoder, and the keys and values come from the output of the encoder

$$\mathbf{Q} = \mathbf{W^Q}\mathbf{H}^{dec[i-1]}; \;\; \mathbf{K} = \mathbf{W^K}\mathbf{H}^{enc}; \;\; \mathbf{V} = \mathbf{W^V}\mathbf{H}^{enc}$$

$$CrossAttention(\mathbf{Q},\mathbf{K},\mathbf{V}) \;=\; \text{softmax}\left(\frac{\mathbf{Q}\mathbf{K}^\intercal}{\sqrt{d_k}}\right)\mathbf{V}$$

  - Cross-attention allows the decoder to compute attention relative to each of the source word representations as encoded by the Encoder

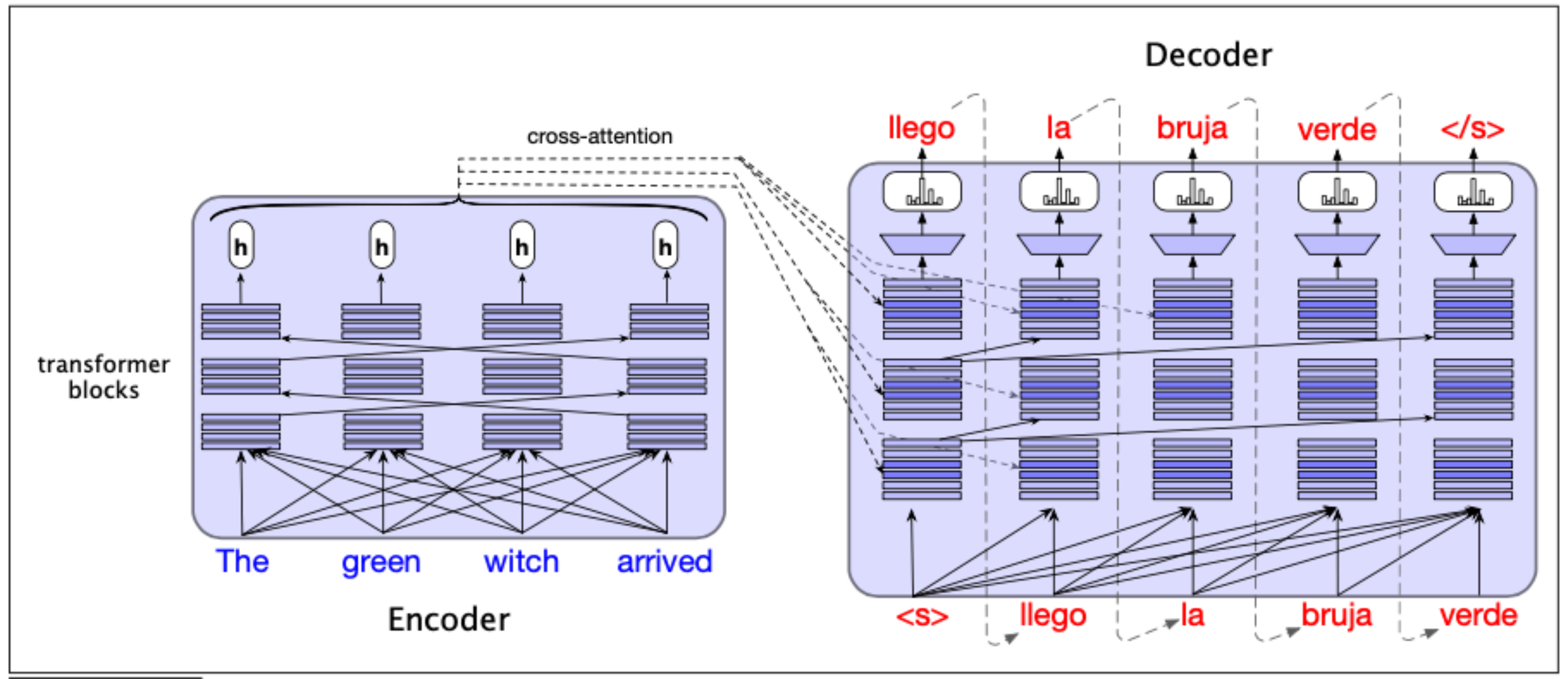



**Figure 10.15** The encoder-decoder architecture using transformer components. The encoder uses the transformer blocks we saw in Chapter 9, while the decoder uses a more powerful block with an extra encoder-decoder attention layer. The final output of the encoder $\mathbf{H}^{enc} = \mathbf{h}_1,...,\mathbf{h}_T$ is used to form the $\mathbf{K}$ and $\mathbf{V}$ inputs to the cross-attention layer in each decoder block.

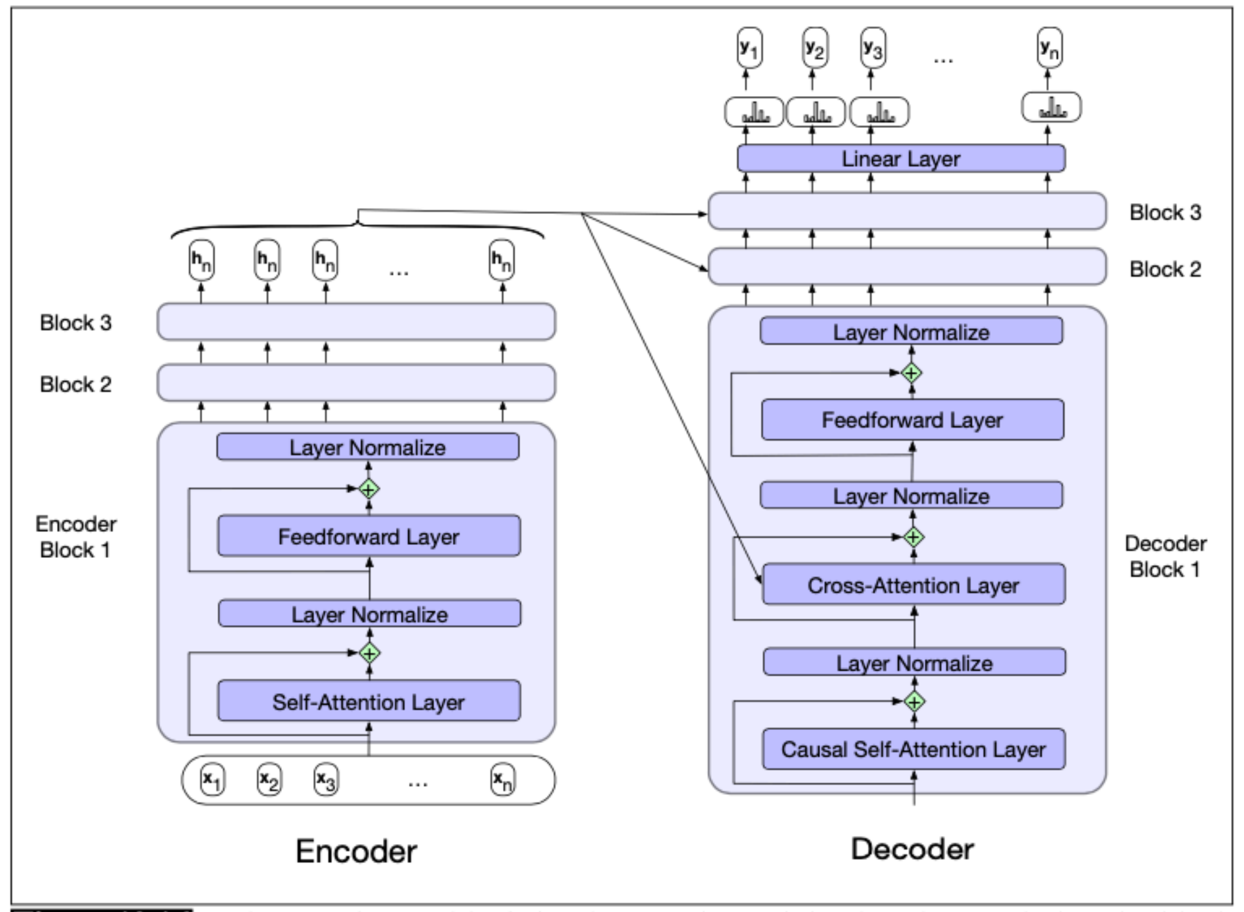



**Figure 10.16** The transformer block for the encoder and the decoder. Each decoder block has an extra cross-attention layer, which uses the output of the final encoder layer $H^{enc} = h_1,...,h_t$ to produce its key and value vectors.

# BEAM SEARCH

# GREEDYNESS OF LANGUAGE MODELS AND THE ENCODER-DECODER APPROACH

- When we generate the next word, in an encoder-decoder model, we typically output the word with the highest probability:

$$\hat{y}_t = argmax_{w \in V} P(w \mid x, y_1 \ldots y_{t-1})$$

- This "greedy" choice is the best in the moment, but not necessarily optimal overall

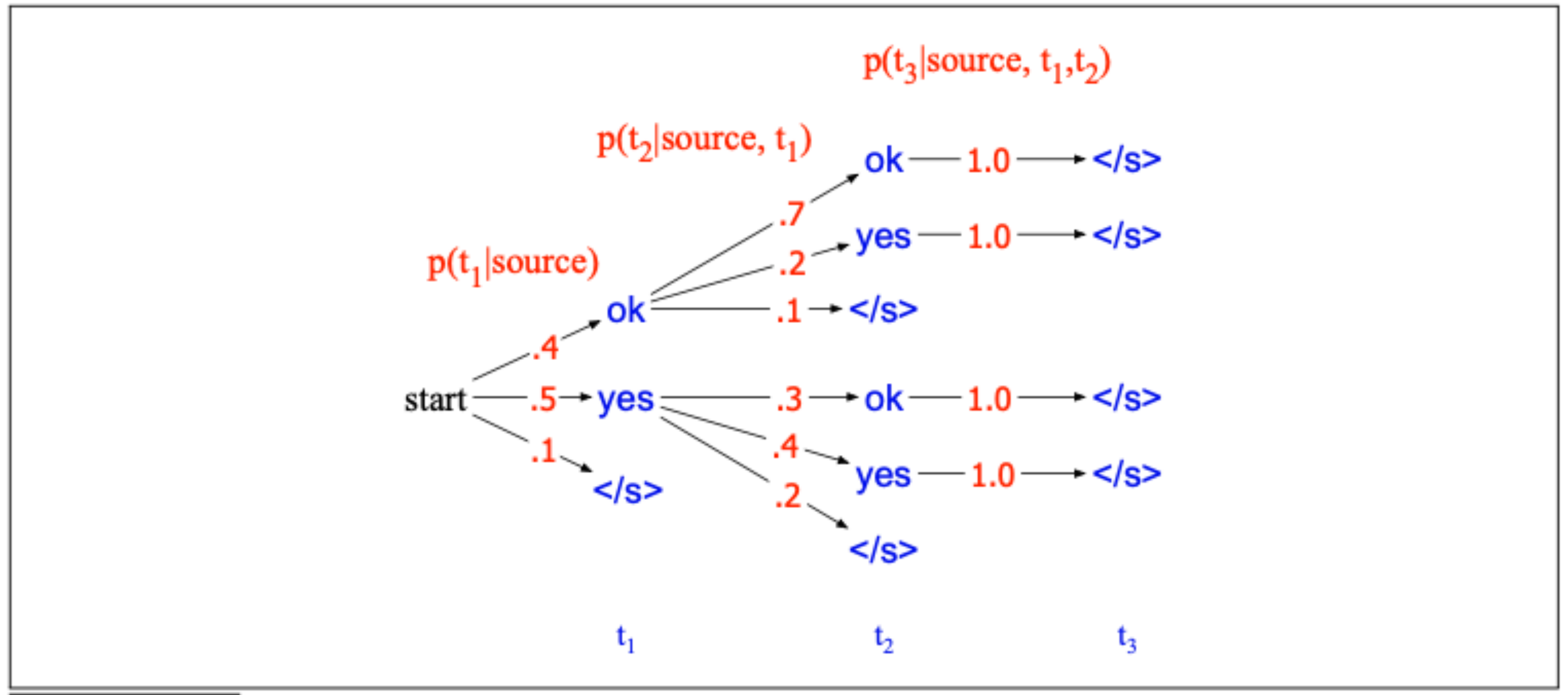



**Figure 10.11** A search tree for generating the target string $T = t_1, t_2, \ldots$ from the vocabulary $V = \{$yes, ok, <s>$\}$, given the source string, showing the probability of generating each token from that state. Greedy search would choose *yes* at the first time step followed by *yes*, instead of the globally most probable sequence *ok ok*.

# BEAM SEARCH OVERVIEW

- Beam Search addresses this heuristically by keeping the top $k$ hypothesis (options for the output sequence at each step, along with their probabilities)
- At each step, we evaluate the probability of extending the sequence so far with each possible word in the vocabulary ($kV$ hypothesis) and
  - Multiply the probability of the sequence generated so far, aka the prefix (which is a product of the probabilities of the individual words in the sequence) by the probability of each possible word (per the softmax computation)
  - The log format of the computation to evaluate hypothesis (sequence) $y$ is as shown below, where $P(y_t | y_1, \ldots . y, y_{t-i}, x)$ is the softmax of the vocabulary word under consideration, and the other probabilities form the prefix probability (with each word conditioned on its prior context):

$$score(y) = logP(y|x) = log(P(y_1|x)P(y_2|y_1,x)\ldots P(y_t|y_1,\ldots y_{t-1},x)) = \sum_{i=1}^{t} logP(y_i|y_1,\ldots,y_{t-1},x)$$

- And choose the top $k$ sequences among the options that were generated
- When the end of sequence symbol is generated, we remove the hypothesis from consideration, reduce $k$ by 1, until the beam has been reduced to 0 (and we thus have $k$ completed hypotheses/sequences)
- Because at the end, we have sequences with different lengths and longer sequences have a lower probability (longer multiplication of small numbers), we typically normalize the output sequences

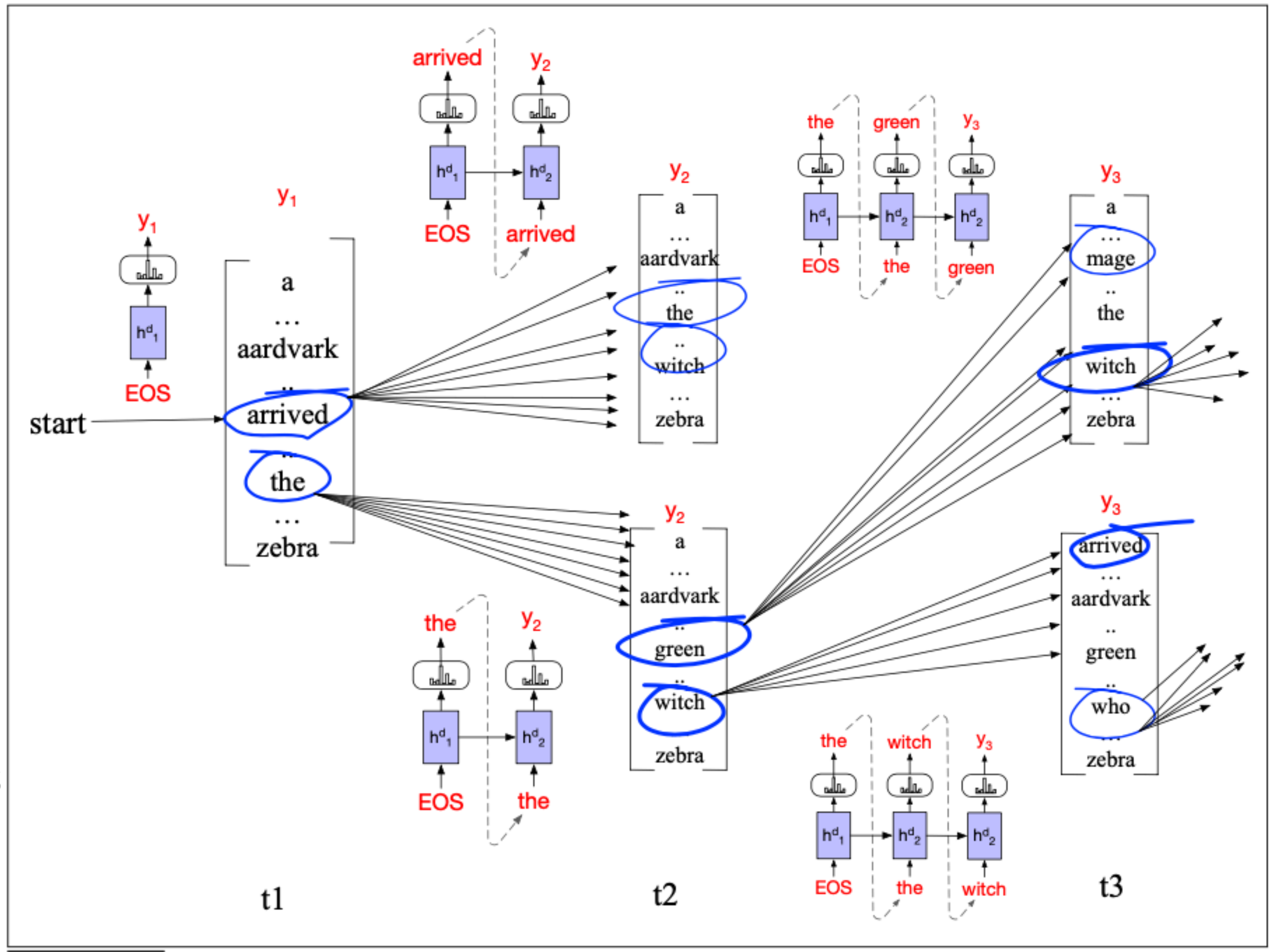



**Figure 10.12** Beam search decoding with a beam width of $k = 2$. At each time step, we choose the $k$ best

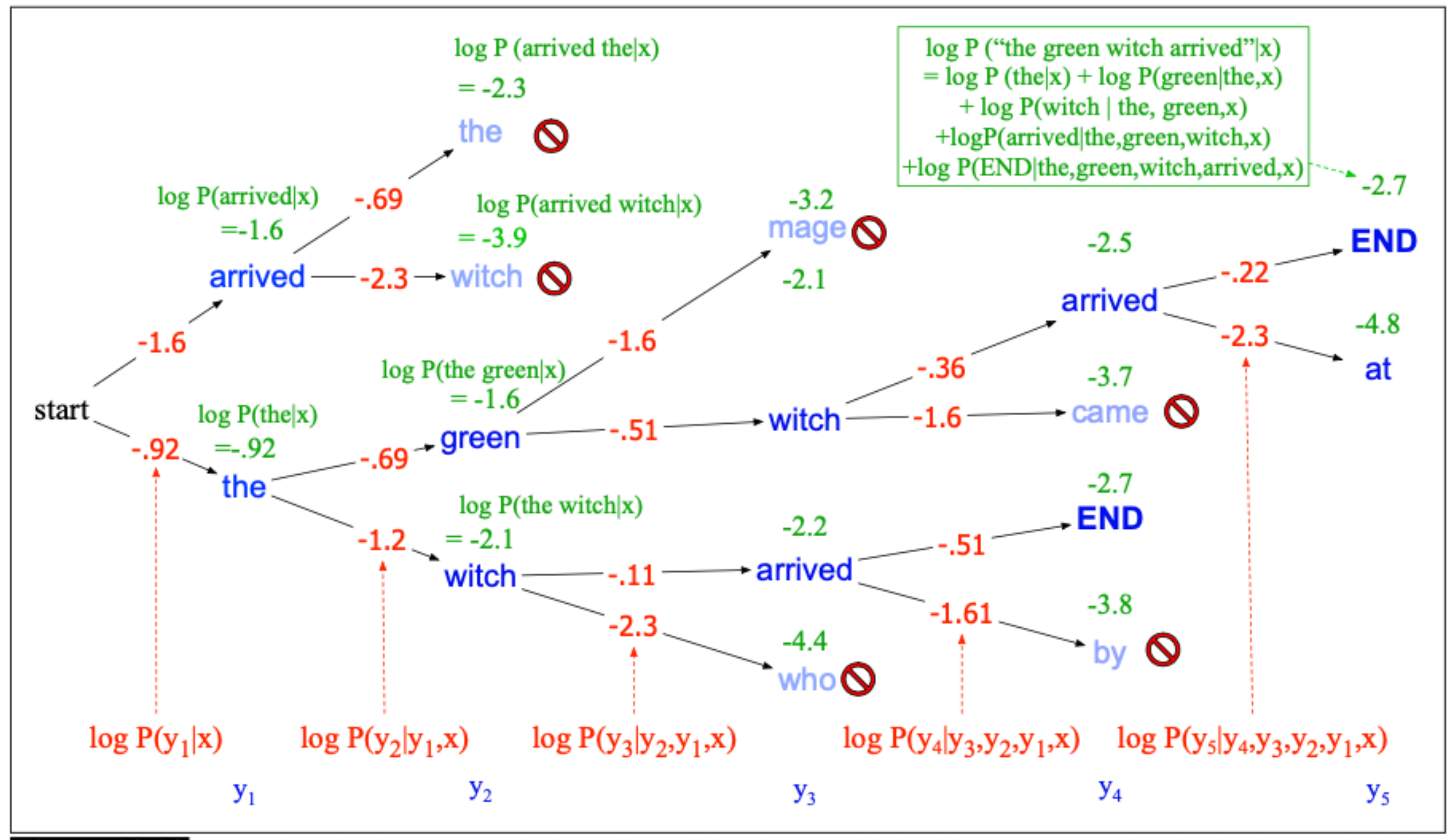



**Figure 10.13** Scoring for beam search decoding with a beam width of $k = 2$. We maintain the log probability of each hypothesis in the beam by incrementally adding the logprob of generating each next token. Only the top $k$ paths are extended to the next step.

# BEAM SEARCH IN PRACTICE

- Typically $k$ is set to 5-10
- The final $k$ hypotheses can be ranked, or a random one can be chosen among them
- Or can all be passed to the downstream application along with their scores and some further processing can happen

# PRACTICAL CONSIDERATIONS IN MACHINE TRANSLATION

# TOKENIZATION

- Common approach to tokenization is to use the BPE or wordpiece algorithms, with 8k to 32k word pieces commonly used
- A shared vocabulary is used for both languages

BPE example:

- Initialize the wordpiece lexicon with characters
- Repeat until there are $V$ word pieces:
  - Train an $n$-gram language model on training corpus, using current set of word pieces
  - Consider the set of possible new word pieces made by concatenating two word pieces from the current lexicon
    - Choose the one new word piece to add to $V$ that most increases the language model probability of the training corpus

# MT CORPORA

- MT training typically uses a parallel corpus (or bitext) which is available in multiple languages, e.g.,
  - E.g., European Parliament proceedings (~2 million sentences in 21 European languages)
  - United Nations Parallel Corpus (10 million sentences in the 6 official UN languages: Arabic, Chinese, English, French, Russian, Spanish)
  - OpenSubtitles Corpus (movie and TV subtitles)
  - ParaCrawl contains sentence pairs from general web text extracted from https://commoncrawl.org/ – 223 million sentence pairs between 23 EU languages and English

# PARALLEL CORPUS TRAINING REQUIRES SENTENCE ALIGNMENT

To be used for training Bitext needs to be sentence aligned

- Need a cost function to compute a score that indicates how likely spans of two sentences are likely to be translations
  - Cosine similarity of multilingual sentence embeddings can be used as a basis for the score computation
    - Multilingual embedding model encodes text from multiple languages into a shared embedding space
- Need an algorithm to find good alignment based on the scores
  - Minimum edit distance algorithm can be adapted for this where we use each word as a token (as opposed to characters)

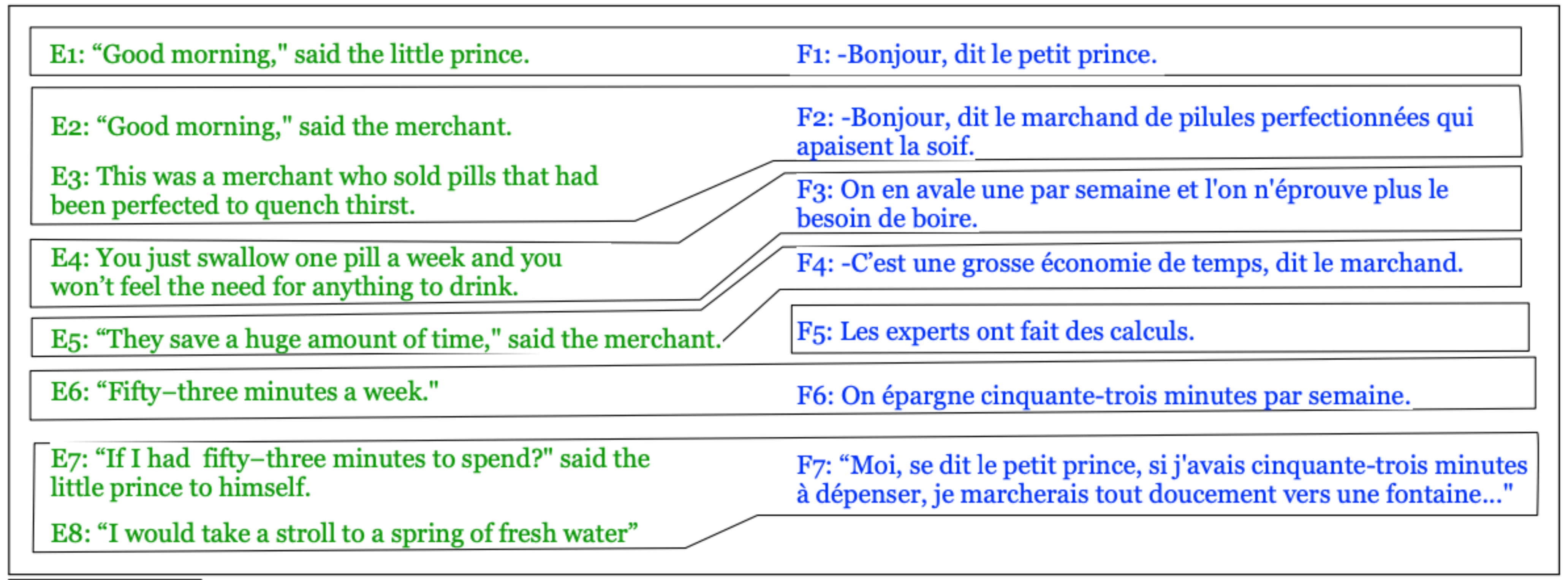



**Figure 10.17** A sample alignment between sentences in English and French, with sentences extracted from Antoine de Saint-Exupery's *Le Petit Prince* and a hypothetical translation. Sentence alignment takes sentences $e_1, \dots, e_n$, and $f_1, \dots, f_n$ and finds minimal sets of sentences that are translations of each other, including single sentence mappings like ($e_1$,$f_1$), ($e_4$,$f_3$), ($e_5$,$f_4$), ($e_6$,$f_6$) as well as 2-1 alignments ($e_2$/$e_3$,$f_2$), ($e_7$/$e_8$,$f_7$), and null alignments ($f_5$).

# BACKTRANSLATION

- When we don’t have a Bitext for a particular target, we can create a synthetic Bitext using *backtranslation* from a monolingual target corpus
  - We train a model on a small Bitext (much easier to obtain/construct) and then use that model to translate a large target language corpus from English to the target language (e.g., Navajo)
  - We use the translated corpus as if it was a human translation for further machine translation model training
- This approach is estimated to work 2/3 as well as using a natural human translated bitext

# EVALUATING MACHINE TRANSLATION

# EVALUATING TRANSLATIONS

- Adequacy/faithfulness/fidelity - how well the translation captures the meaning of the source text
- Fluency - how readable/natural the translated text is

## HUMAN EVALUATION

- Evaluate the translated text only (if crowd workers only speak the target language) - can use a numerical scale (e.g., 5-point or 100-point)
- Compare the translated text with a gold standard translation – rate how much information is preserved
- Compare two translations - rank translations relative to each other
- Human evaluation is not trivial
    - Training of the crowd workers is typically required
    - Need a way to consolidate differing ratings among different workers (e.g., removing outliers, normalizing the ratings)

# AUTOMATED EVALUATION

- Based on character overlap
- Based on word overlap
- Based on embedding similarity

# AUTOMATED EVALUATION BASED ON CHARACTER OVERLAP

- chrF stands for character F-score (2015) and is based on computing the level of overlap of character n-grams between the human and machine translations
- The human translation is referred to as "reference" and the machine translation is referred to as "hypothesis"

chrP - percentage of character 1-grams, 2-grams, …, k-grams in the hypothesis that occur in the reference, averaged over the number of n-gram types

chrR - percentage of character 1-grams, 2-grams, …, k-grams in the reference that occur in the hypothesis, averaged over the number of n-gram types

$chrF\beta = (1+\beta^2)\dfrac{chrP \cdot chrR}{\beta^2 \cdot chrR + chrR}$, where $\beta$ weights precision vs. recall, eg., $\beta = 2$ values recall twice as much as precision

- Example with $k = 2$ and $\beta = 2$:

| | |
|---|---|
| REF: `witness for the past,` | |
| HYP1: `witness of the past,` | chrF2,2 = .86 |
| HYP2: `past witness` | chrF2,2 = .62 |

unigrams that match: `w i t n e s s f o t h e p a s t ,` (17 unigrams)
bigrams that match: `wi it tn ne es ss th he ep pa as st t,` (13 bigrams)

| | | | |
|---|---|---|---|
| unigram P: | 17/17 = 1 | unigram R: | 17/18 = .944 |
| bigram P: | 13/16 = .813 | bigram R: | 13/17 = .765 |

$$\begin{aligned} \text{chrP} &= (17/17+13/16)/2 = .906 \\ \text{chrR} &= (17/18+13/17)/2 = .855 \\ \text{chrF2,2} &= 5\frac{\text{chrP} * \text{chrR}}{4\text{chrP}+\text{chrR}} = .86 \end{aligned}$$

# AUTOMATED EVALUATION BASED ON CHARACTER OVERLAP

- chrF is robust and correlates well with human judgement
- It has superseded the use of some of the previously common word-overlap metrics such as BLEU which was purely precision-based
  - Word-based metrics are sensitive to word tokenization (which makes some comparisons difficult), and are especially challenged in languages with complex morphology
- Shortcomings:
  - Approaches based on n-grams or words are very local:
    - Movement of a phrase across a sentence does not affect the metric much
    - Coherence of the overall translation is not captured (cross-sentence properties of a translation)
  - chrF is generally good at evaluating improvements to a single system rather than comparing very different translation systems (e.g., different automated models)

# AUTOMATIC EVALUATION USING EMBEDDING SIMILARITY

- Embedding-based methods are not constrained to exact character n-grams and are thus able to capture synonymous/alternative words or phrases
- For example, we can use BERT to compute embeddings for the tokens in reference ($x$) and the hypothesis/candidate ($\tilde{x}$) sequences, and compare each possible pair of tokens using cosine similarity
- Then we compute precision and recall over the pairs that have highest similarity - we match each token in $x$ to the token in $\tilde{x}$ with which it has the highest similarity to compute recall, and we match each token in $\tilde{x}$ to a token in $x$ with which it has the highest similarity to compute precision

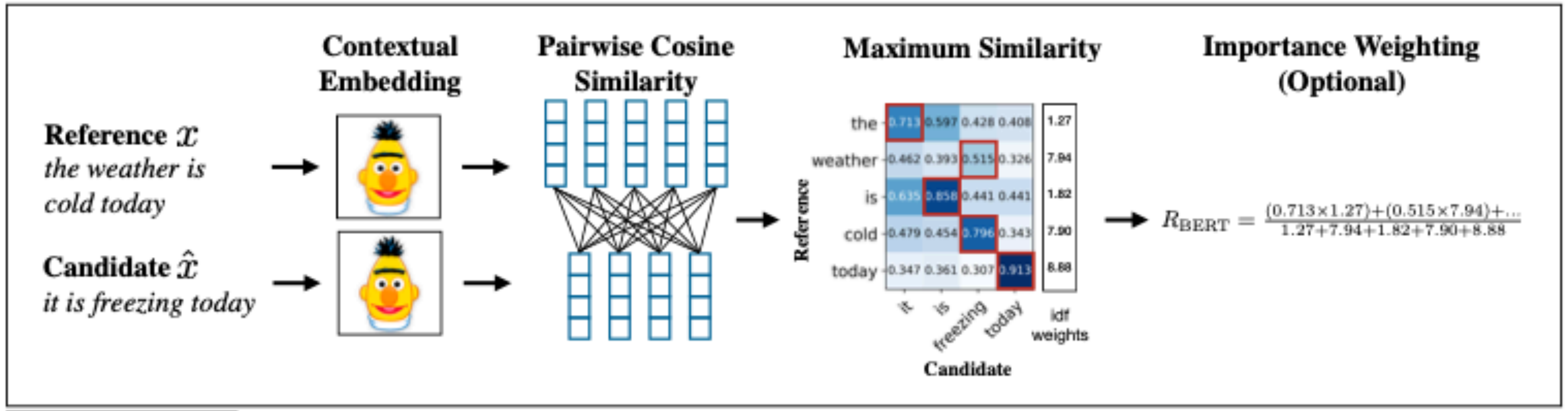



**Figure 10.18** The computation of BERTScore recall from reference $x$ and candidate $\hat{x}$, from Figure 1 in Zhang et al. (2020). This version shows an extended version of the metric in which tokens are also weighted by their idf values.

# ETHICAL ISSUES AND BIAS

# BIAS IN MACHINE TRANSLATION

- MT systems amplify gender biases beyond actual labor employment statistics and do worse when having to translate text that includes non-stereotypical gender roles
- Current focus is on trying to surface better when the system is not sure through assigning confidence values to translations ("do no harm" directive, especially in medical and legal domains)
- Another major effort is to improve translation in low resourced languages

| Hungarian (gender neutral) source | English MT output |
|---|---|
| ő egy ápoló | she is a nurse |
| ő egy tudós | he is a scientist |
| ő egy mérnök | he is an engineer |
| ő egy pék | he is a baker |
| ő egy tanár | she is a teacher |
| ő egy esküvőszervező | she is a wedding organizer |
| ő egy vezérigazgató | he is a CEO |

**Figure 10.19** When translating from gender-neutral languages like Hungarian into English, current MT systems interpret people from traditionally male-dominated occupations as male, and traditionally female-dominated occupations as female (Prates et al., 2019).