Natural Language Processing

Lecture 12 : Machine Translation

Master Degree in Computer Engineering
University of Padua
Lecturer : Giorgio Satta

Lecture partially based on material originally developed by :
Marco Kuhlman, Linköping University
Cristopher Manning, Stanford University

Machine translation

©Dungeons & Dragons

Machine translation

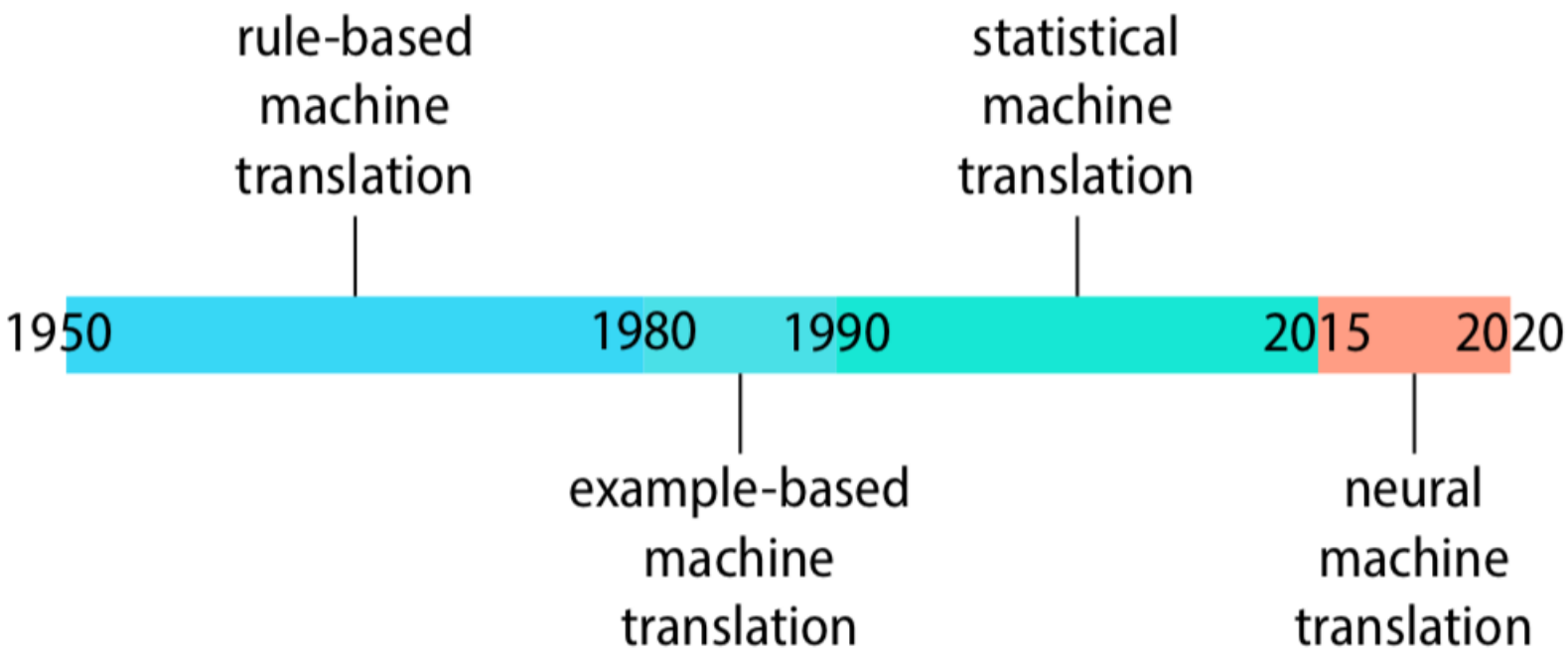
Machine translation (MT) investigates models and algorithms to translate from one language to another.

Nowadays, MT focuses on a number of **practical tasks** such as

- multi-lingual information access
- social networks
- mobile translation

Translation in its full generality is a difficult, fascinating, and intensely human endeavor.

A timeline of machine translation



Translation

©Dungeons & Dragons

Word ordering

Languages differ in the basic word order of verb, subject and object in simple **declarative clauses**.

Example :

- English, French, and Mandarin are SVO (Subject, Verb, Object) order
- Hindi and Japanese are SOV order
- Arabic is VSO order

Word translation

In translating individual words one needs to detect **word sense**, since different word senses map to different words in target language.

Example : English word **bass** appears in Spanish as the word **lubina** (a fish) or as the word **bajo** (a musical instrument).

Sometimes the target language places more grammatical constraints on word choice than the source language.

Example : French and Spanish mark grammatical **gender** on adjectives, while English does not.

Word translation

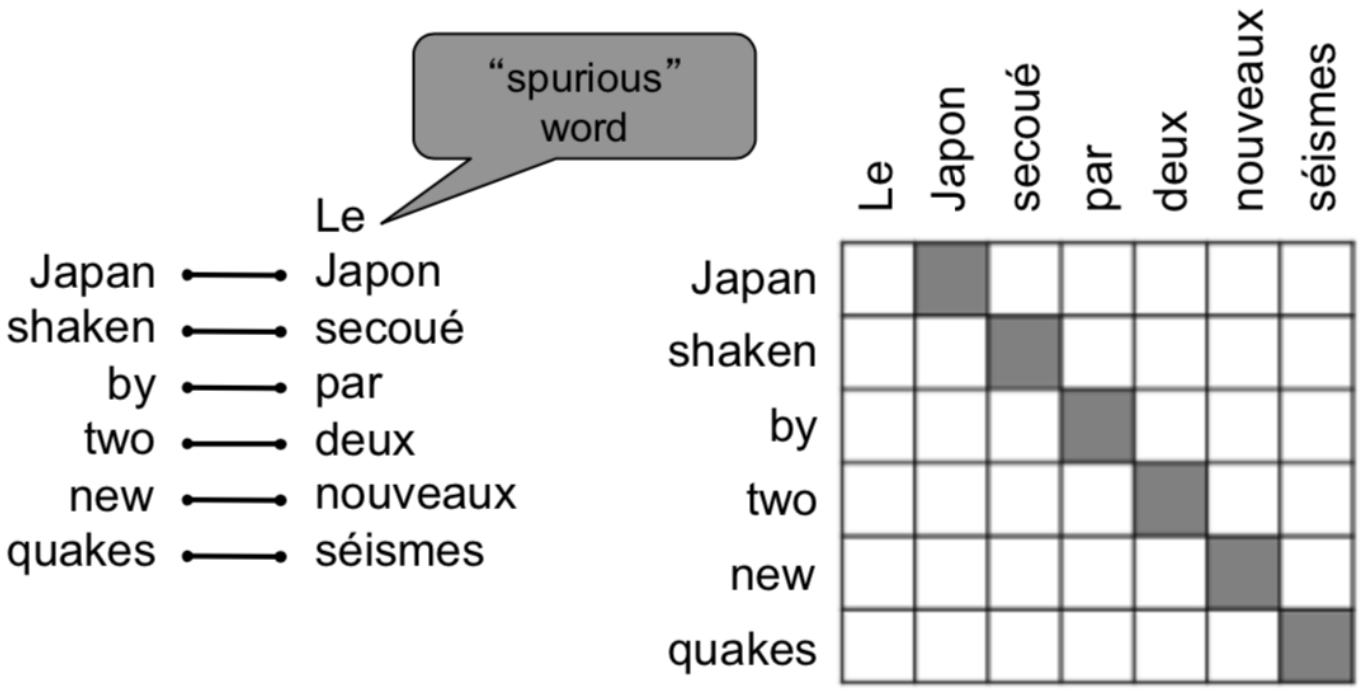
There may be a **lexical gap** between a pair of languages, where no word in the target language can express the exact meaning of a word in the source language.

In these cases one needs to resort to **paraphrasing**.

Word alignment

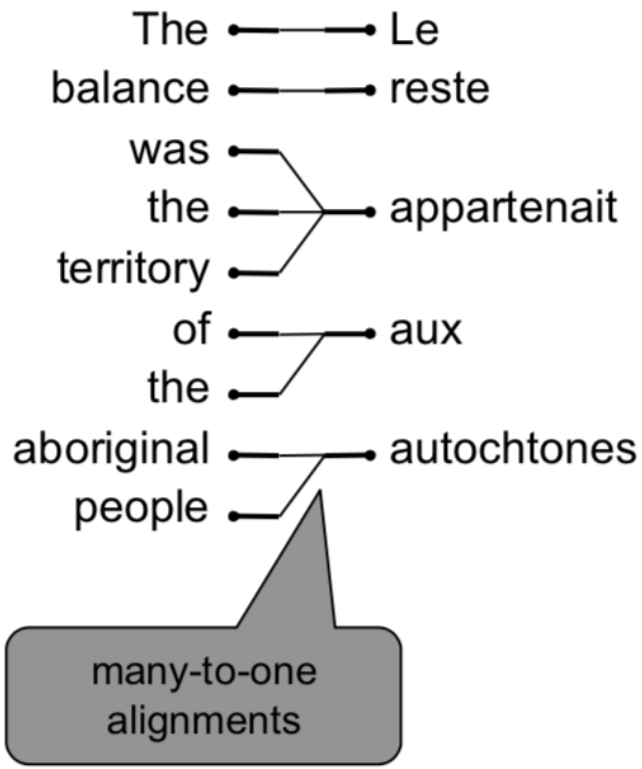
Word alignment is the correspondence between words in the source and target sentences.

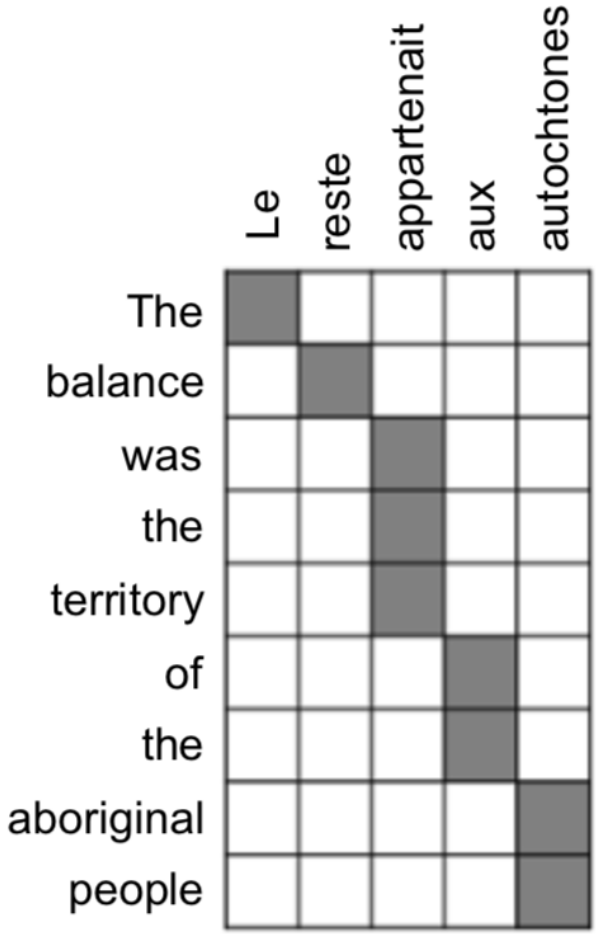
Spurious words have no counterpart in the target language.



Word alignment

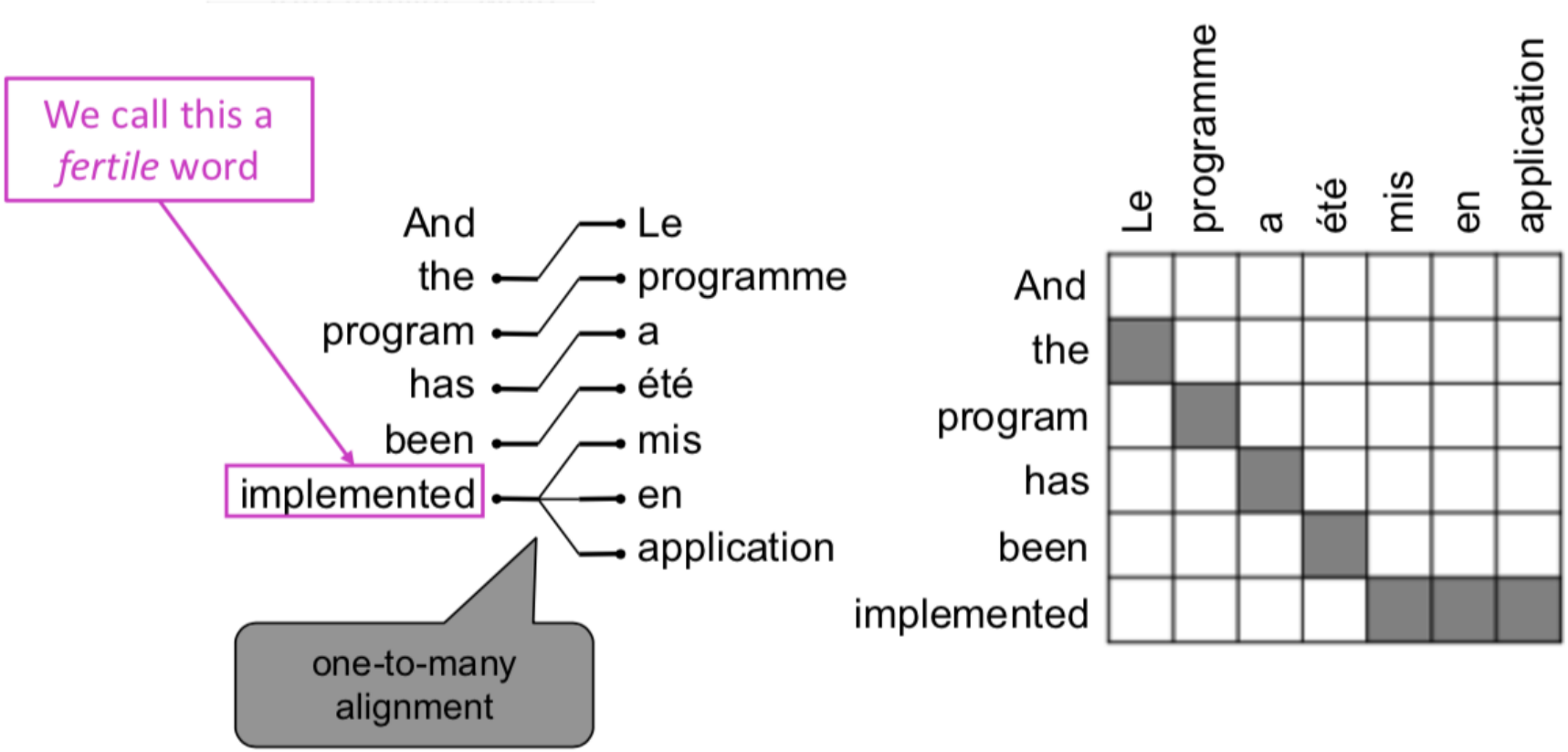
Alignment can be **many-to-one**.





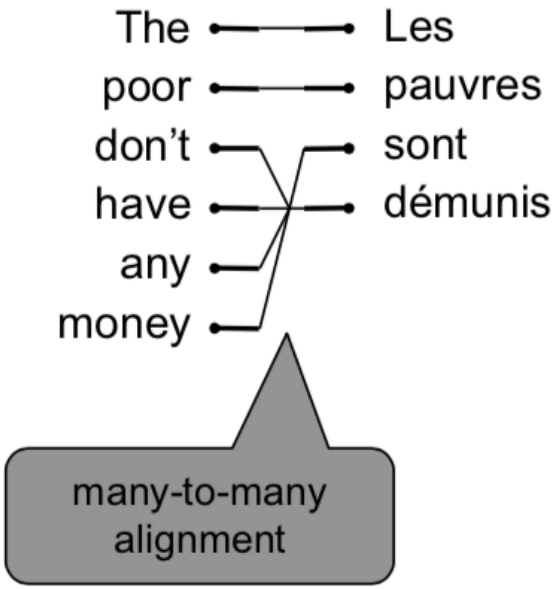
Word alignment

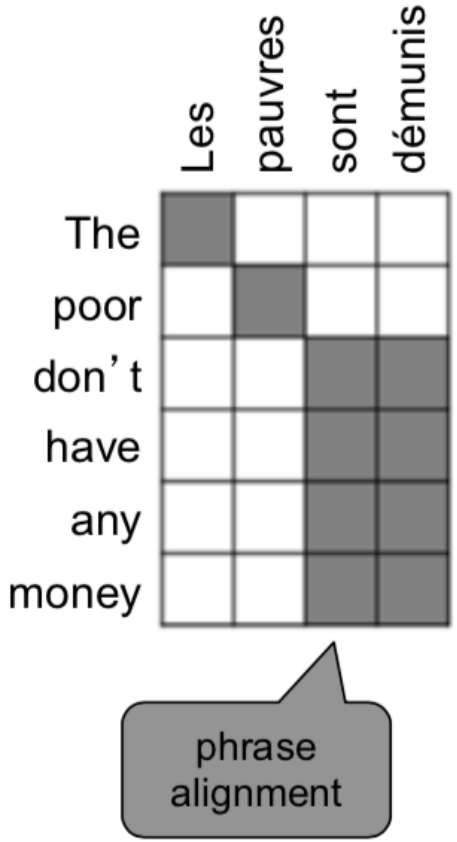
Alignment can be **one-to-many**.



Word alignment

Alignment can be **many-to-many**.





Statistical machine translation

©depositphotos

Statistical machine translation

Machine translation can be formulated as a **structured prediction** task. Given a source sentence x, find the most probable target sentence $\widehat{y}$:

$$\widehat{y} = \underset{y}{\text{argmax}} \; P(y \mid x)$$

Statistical machine translation (SMT) uses Bayes' rule to decompose the posterior probability into two components that can be learned separately:

$$\widehat{y} = \underset{y}{\text{argmax}} \; P(x \mid y)P(y)$$

where

- $P(x \mid y)$ is a **translation model**
- $P(y)$ is a language model

Statistical machine translation

$$P(y \mid x) = P(x \mid y)P(y)$$

Why two models rather than one?

- $P(y \mid x)$ needs to account for **both** the word translation and the ordering of the output y
- $P(x \mid y)$ assigns large probability to strings that have the necessary words (roughly at the right places)
- $P(y)$ assigns large probability to well-formed strings y, regardless of the connection to x.

$P(x \mid y)$ and $P(y)$ collaborate to produce a large probability for well-formed translation pairs.

Statistical machine translation

$$\widehat{y} = \underset{y}{\text{argmax}} \; P(x \mid y)P(y)$$

Our statistical model consists of three separate tasks:

- estimating the language model
- estimating the translation model
- devising an efficient **search** for the string y that maximizes the product, for given x

Translation model

If variable a ranges over all possible **alignment relations** for x and y, we can write

$$P(x \mid y) = \sum_a P(x, a \mid y)$$

Different alignments may contribute to the same translation.

Several models for $P(x, a \mid y)$ of increasing degree of complexity, based on

- specific independence assumptions
- constraint on word-to-word translation
- constraint on aligment

Research papers

Title: The Mathematics of Statistical Machine Translation: Parameter Estimation

Authors: Peter F. Brown, Stephen A. Della Pietra, Vincent J. Della Pietra, Robert L. Mercer

Journal: Computational Linguistics, Volume 19, Number 2, June 1993

Content: This article describes a series of five statistical models for the translation process and provides algorithms for estimating the parameters of these models, given a set of pairs of sentences that are translations of one another.

https://aclanthology.org/J93-2003/

Neural machine translation

©Dungeons & Dragons

Neural machine translation

Neural machine translation (NMT) models the translation task through a single artificial neural network.

SMT systems, built by hundreds of engineers over many years, outperformed by NMT systems trained by a handful of engineers in a few months.

2014: First NMT paper published; 2016: Google Translate switches from SMT to NMT. This meant replacing 500,000 lines of phrase-based MT code with a 500-line neural network model.

NMT research has pioneered many of the recent innovations of NLP deep learning

Neural machine translation

Let x be the source text and $y = y_1 \cdots y_m$ be the target text.

Subword tokenization always used; see the lecture on text normalization.

In contrast to SMT, in NMT we **directly** model $P(y \mid x)$, using an approach similar to the one adopted for language modeling

$$\begin{aligned} P(y \mid x) &= P(y_1 \mid x) \cdot P(y_2 \mid y_1, x) \cdot P(y_3 \mid y_1, y_2, x) \cdot \\ &\quad \cdots \cdot P(y_m \mid y_1, \ldots, y_{m-1}, x) \\ &= \prod_{t=1}^{m} P(y_t \mid y_1, \ldots, y_{t-1}, x) \end{aligned}$$

Encoder-decoder networks

©Dungeons & Dragons

Encoder-decoder networks

Encoder-decoder networks, also called **sequence-to-sequence** (seq2seq) networks, are models capable of generating contextually appropriate, arbitrary length sequences.

Encoder-decoder networks have been applied to a wide range of NLP applications

- machine translation
- summarization
- question answering
- dialogue

Recently, also syntactic and semantic parsing.

Encoder-decoder networks

The encoder-decoder model consists of two components.

The **encoder** is a neural network that produces a representation of the source sentence.

The **decoder** is an **autoregressive** language model that generates the target sentence, conditioned on the output of the encoder.

Autoregressive = takes its own output as new input.

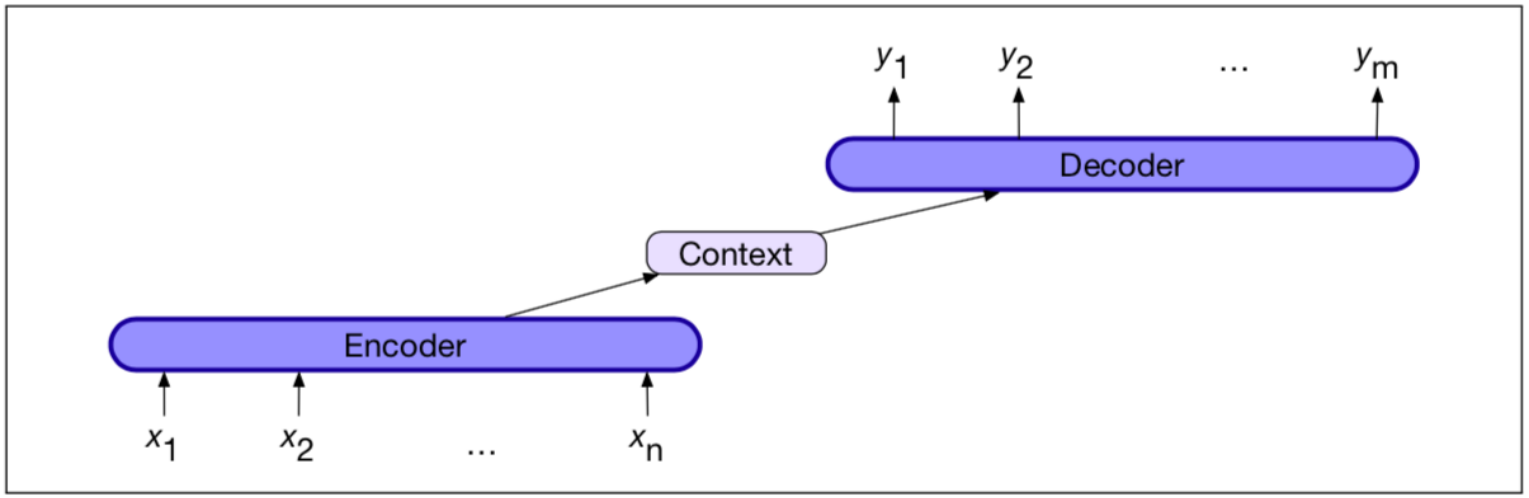
The encoder-decoder model can be implemented in two ways, through recurrent networks or else through the tranformer.

In general, use of **attention** techniques improves model performance.

Encoder-decoder networks

Encoder-decoder models based on RNN use the following **main idea**.

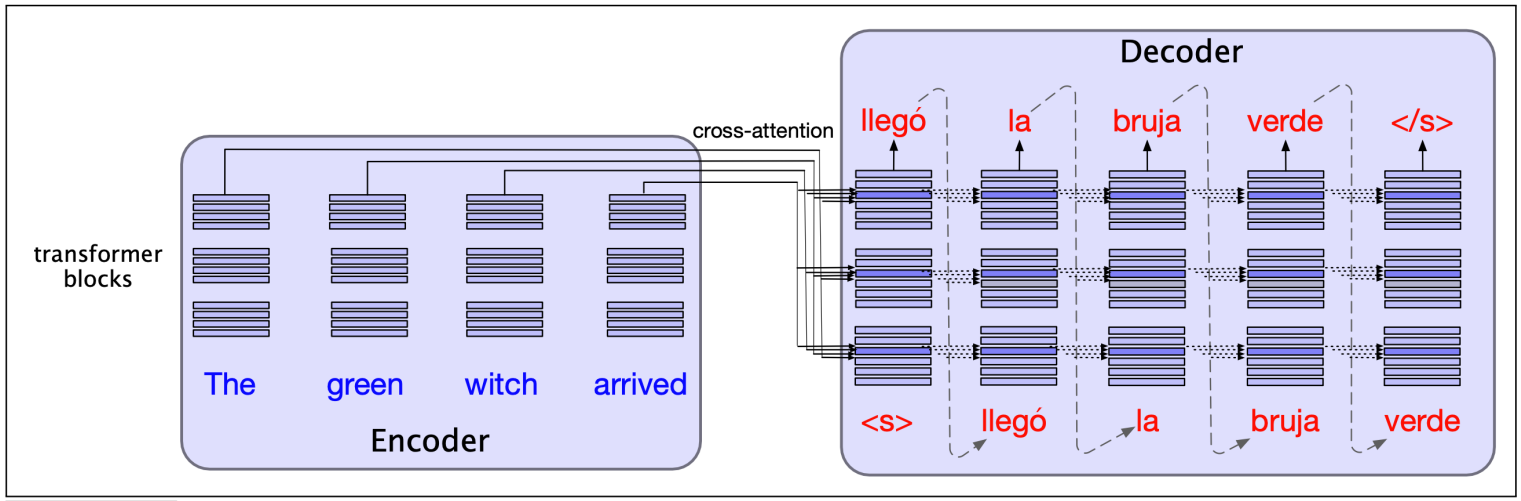
The output of the encoder is called **context** and drives the translation, together with the decoder output.



Encoder-decoder networks

Encoder-decoder models based on Transformers use the following **main idea**.

The self-attention in the encoder is allowed to **look ahead** at the entire source language text. The decoder is augmented with a special new layer called the **cross-attention layer**.



Encoder-decoder with RNN

©Wizard of the Coast

Encoder-decoder with RNN: inference

We present a **greedy** inference algorithm.

Later we also discuss a beam search inference algorithm.

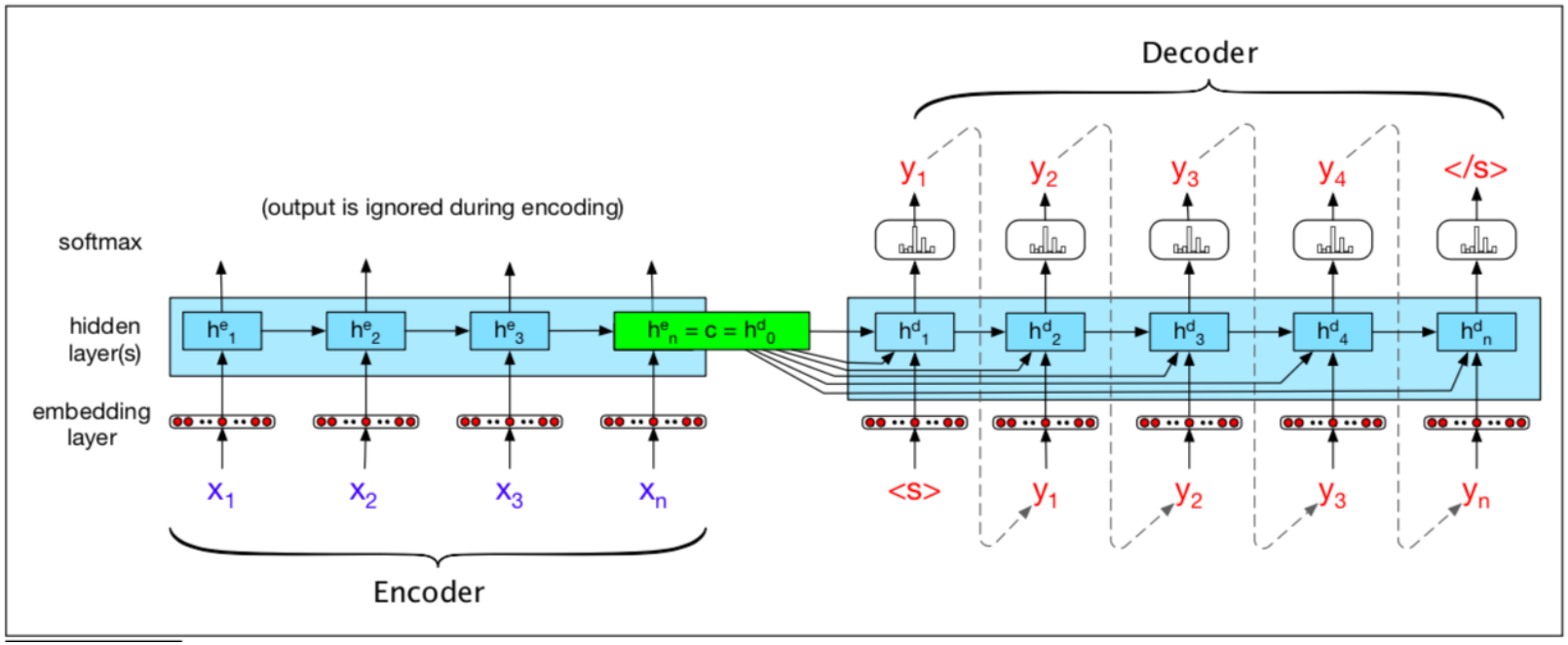
$P(y \mid x)$ can be computed as follows

- run an RNN **encoder** through $x = x_1 \cdots x_n$ performing forward inference, generating hidden states $\mathbf{h}_t^e$, $t \in [1..n]$
- run an RNN **decoder** performing autoregressive generation; to generate y_t, $t \in [1..m]$, use
  - encoder hidden state $\mathbf{h}_n^e$
  - decoder hidden state $\mathbf{h}_{t-1}^d$
  - embedding of word y_{t-1}

  Stop when the end-of-sentence marker is predicted.

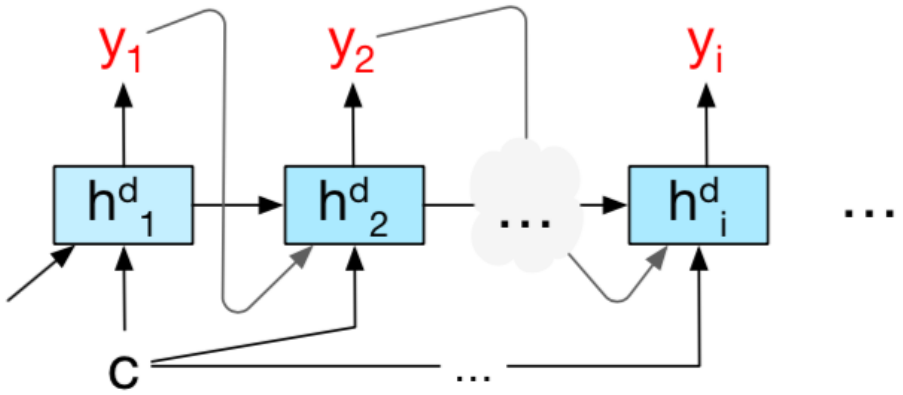
Encoder-decoder with RNN: inference

Overview of inference process



Encoder-decoder with RNN: inference

The final hidden state of the encoder $\mathbf{h}_n^e$ is a representation of the entire input sentence x, and is also called the **context** $\mathbf{c}$.



Each hidden state $\mathbf{h}_t^d$ of the decoder is computed from $\mathbf{c}$, the previous decoder hidden state, and the output generated in the previous state.

Early models did not use $\mathbf{c}$ in computation of $\mathbf{h}_t^d$, and were affected by vanishing gradient problem.

Encoder-decoder with RNN: inference

Assume

- $\mathbf{b}(u)$ is an embedding of word u from the vocabulary V
- g affine transformation combined with non-linear component; f affine transformation

The model equations are

$$
\begin{aligned}
\mathbf{h}_t^e &= \text{RNN}(\mathbf{h}_{t-1}^e, \mathbf{b}(x_t)) \\
\mathbf{c} &= \mathbf{h}_0^d = \mathbf{h}_n^e \\
\mathbf{h}_t^d &= g(\mathbf{c}, \mathbf{h}_{t-1}^d, \mathbf{b}(\widehat{y}_{t-1})) \\
\mathbf{z}_t &= f(\mathbf{h}_t^d) \\
\mathbf{s}_t &= \text{softmax}(\mathbf{z}_t) = P(u \mid \widehat{y}_1, \ldots, \widehat{y}_{t-1}, x) \\
\widehat{y}_t &= \underset{u \in V}{\text{argmax}} \; P(u \mid \widehat{y}_1, \ldots, \widehat{y}_{t-1}, x)
\end{aligned}
$$

Practical issues

For the encoder, **stacked** (multi-layer) architectures are the norm, where the output states from the top layer of the stack are taken as the final representation.

A widely used approach for the encoder is stacked **biLSTMs**, where the hidden states from top layers from the forward and backward passes are concatenated.

Encoder-decoder with RNN: training

©Pinterest

Encoder-decoder with RNN: training

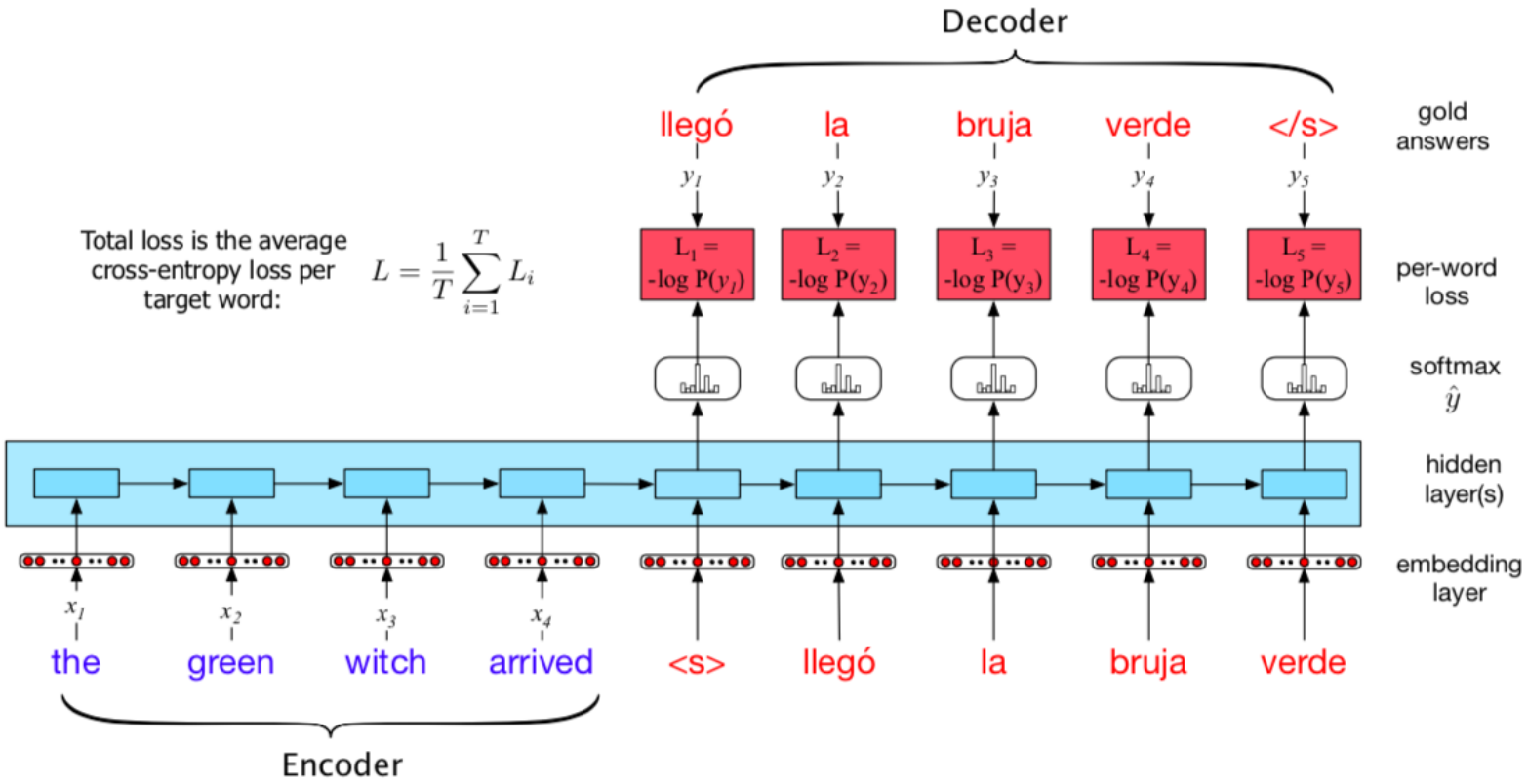
An encoder-decoder is optimized as a single system, backpropagation operates **end-to-end**.

Training proceeds as with any RNN-based language model.

Given the source text and the gold translation, we compute the **average loss** w.r.t. our predictions on next word in the translation.

Encoder-decoder with RNN: training

Schematic representation for computing the average loss during training.



Teacher forcing

During training, the decoder uses gold translation words as the input for the next step prediction. This is called **teacher forcing**.

During inference, the decoder uses its own estimated output as the input for the next step prediction.

Thus the decoder will tend to **deviate** more and more from the gold target sentence as it keeps generating more tokens.

The decoder is not used to deal with possibly erroneous output words, a scenario that is quite common at testing time.

Attention

©Dungeons & Dragons

Attention

The context vector $\mathbf{c}$ must represent the whole source sentence in one fixed-length vector. This is called the **bottleneck problem**.

In addition, which information from $\mathbf{c}$ do we use at each step in the translation?

The **attention** mechanism allows the decoder to get information from **all the hidden states** of the encoder.

The idea is to compute context $\mathbf{c}_i$ at each decoding step i, as a **weighted sum** of all the encoder hidden states $\mathbf{h}_j^e$.

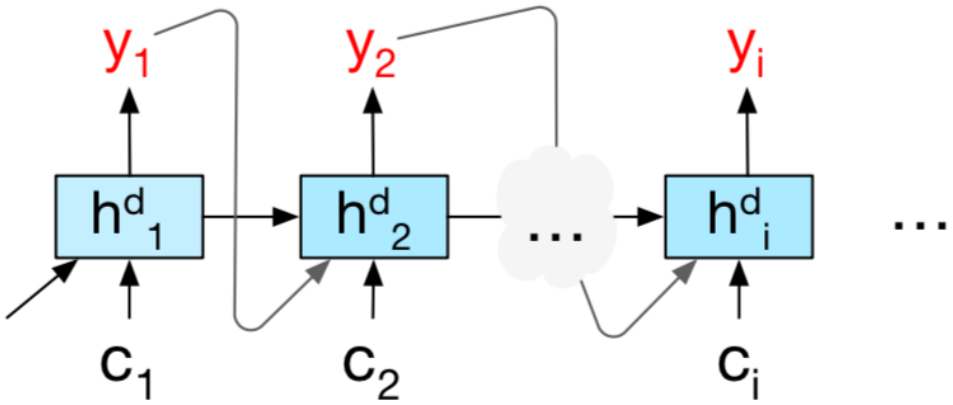
Weights are a function of i.

The weights are used to focus on a particular part of the source text that is relevant for the token being predicted at step i.

Dynamic context vector

Attention thus replaces the static context $\mathbf{c}$ with a context $\mathbf{c}_i$ dynamically computed from all encoder hidden states.

$$\mathbf{h}_i^d = g(\mathbf{c}_i, \mathbf{h}_{i-1}^d, y_{i-1})$$



Dynamic context vector

At each step i during decoding, we compute **relevance scores** $\text{score}(\mathbf{h}^d_{i-1}, \mathbf{h}^e_j)$, for each encoder state $\mathbf{h}^e_j$.

The simplest such score, called **dot-product attention**, implements relevance as similarity through dot-product

$$\text{score}(\mathbf{h}^d_{i-1}, \mathbf{h}^e_j) \quad = \quad \mathbf{h}^d_{i-1} \cdot \mathbf{h}^e_j$$

More sophisticated scoring functions presented later.

Dynamic context vector

We **normalize** the scores to create weights α_{ij}, $j \in [1..n]$

$$\begin{aligned} \alpha_{ij} &= \text{softmax}(\text{score}(\mathbf{h}^d_{i-1}, \mathbf{h}^e_j)) \\ &= \frac{\exp \text{score}(\mathbf{h}^d_{i-1}, \mathbf{h}^e_j)}{\sum_k \exp \text{score}(\mathbf{h}^d_{i-1}, \mathbf{h}^e_k)} \end{aligned}$$

Quantity α_{ij} tells us the proportional **relevance** of encoder hidden state $\mathbf{h}^e_j$ at step i.

We can also view α_{ij}, $j \in [1..n]$ as a **probability distribution**.

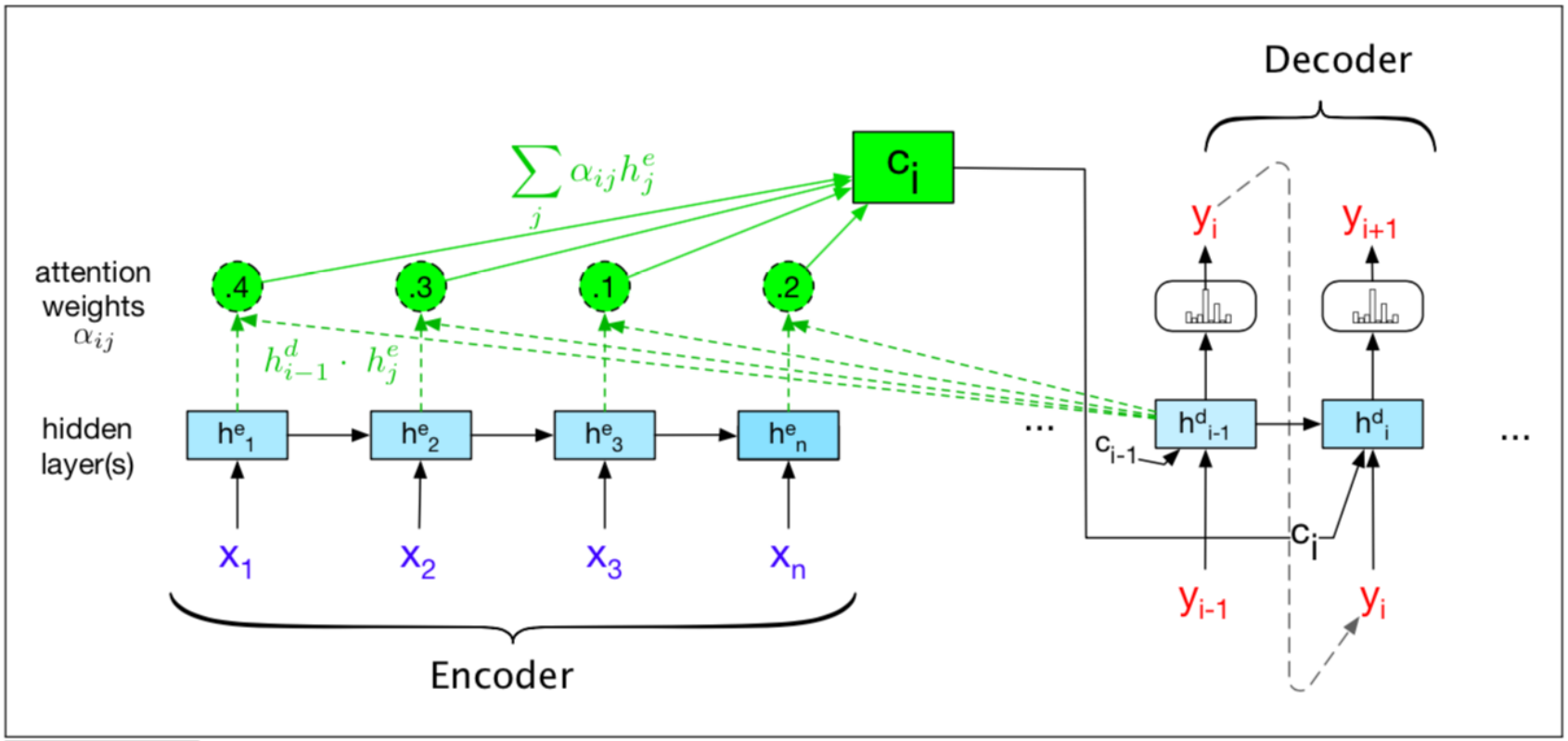
Dynamic context vector

We finally compute a fixed-length context vector that takes into account information from all of the encoder hidden states and that is **dynamically** updated

$$\mathbf{c}_i = \sum_j \alpha_{ij} \mathbf{h}_j^e$$

We can view $\mathbf{c}_i$ as the **expected value** of the encoder hidden states with respect to distribution α_{ij}.

Attention


Decoder
$\sum_j \alpha_{ij} h_j^e$
c_i
attention weights α_{ij}
.4
.3
.1
.2
$h_{i-1}^d \cdot h_j^e$
hidden layer(s)
h^e_1
h^e_2
h^e_3
h^e_n
x_1
x_2
x_3
x_n
Encoder
y_i
y_{i+1}
h^d_{i-1}
h^d_i
c_{i-1}
c_i
y_{i-1}
y_i

Attention

More sophisticated scoring functions for attention have been proposed, as the **bilinear model**

$$\text{score}(\mathbf{h}^d_{i-1}, \mathbf{h}^e_j) \quad = \quad \mathbf{h}^d_{i-1} \mathbf{W}_s \mathbf{h}^e_j$$

Learnable parameters $\mathbf{W}_s$ weight all feature combinations from decoder and encoder hidden states.

This score allows the encoder and decoder to use different dimensions for their hidden states.

Encoder-decoder with Transformer

©Wizard of the Coast

Encoder-decoder with Transformer

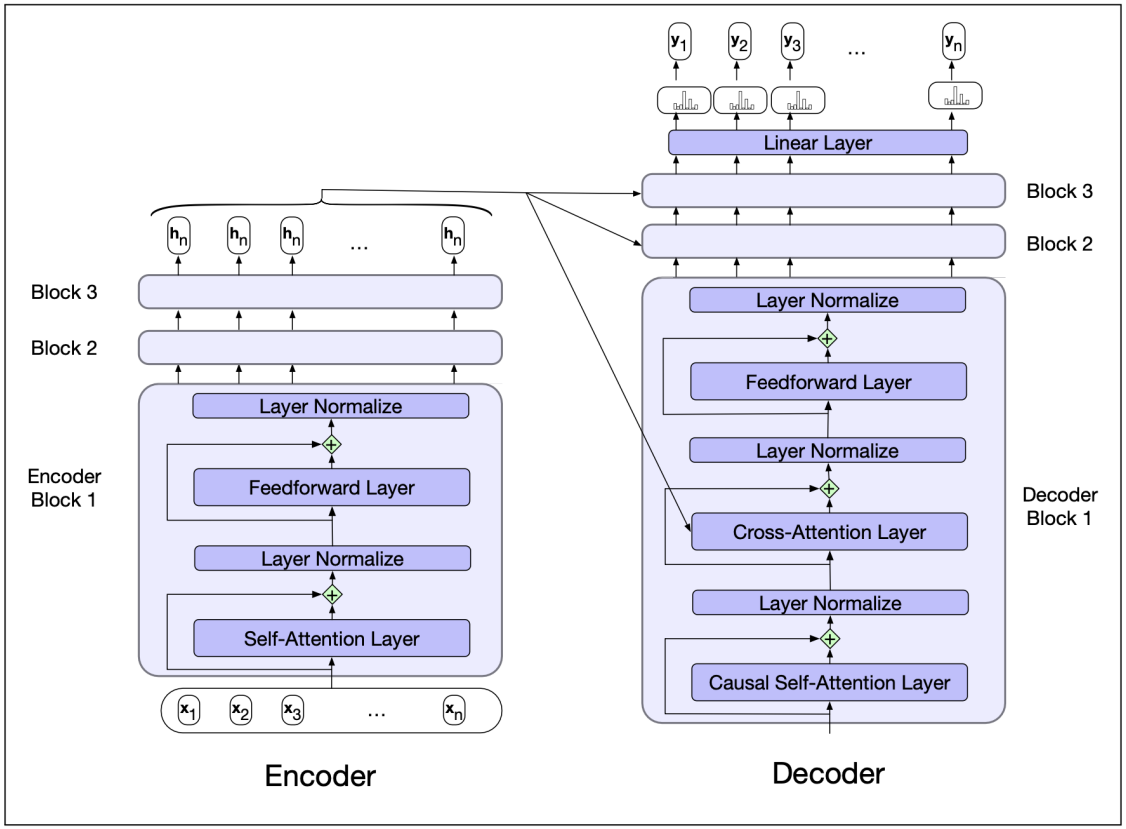
The standard architecture for neural MT is the transformer.

Recall that the **transformer block in the encoder** consists of

- a self-attention layer that attends to the input from the previous layer
- a normalization layer
- a feed forward layer
- another normalization layer

Encoder-decoder with Transformer

In order to attend to the source language, each **transformer block in the decoder** has an extra layer implementing cross-attention.



Encoder-decoder with Transformer

Cross-attention in the decoder is a variant of the self-attention in the encoder

- the queries are computed from the previous layer of the decoder, as in the encoder
- the keys and values are computed from the output of the encoder

Encoder-decoder with Transformer

Assume that

- $\mathbf{H}^{enc}$ is the output from the topmost encoder layer
- $\mathbf{H}^{dec[i-1]}$ is the output from the previous decoder layer
- $\mathbf{W^Q}$ is the cross-attention layer's query weights
- $\mathbf{W^K}, \mathbf{W^V}$ are the cross-attention key and value weights

The model equations are

$$
\begin{aligned}
\mathbf{Q} &= \mathbf{W^Q H}^{dec[i-1]} \\
\mathbf{K} &= \mathbf{W^K H}^{enc} \\
\mathbf{V} &= \mathbf{W^V H}^{enc} \\
\mathit{CrossAttention}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) &= \mathit{SoftMax}\left(\frac{\mathbf{Q}\mathbf{K}^\top}{\sqrt{d_k}}\right)\mathbf{V}
\end{aligned}
$$

Encoder-decoder with Transformer

In summary

- the self-attention layer in the encoder is allowed to look ahead at the entire source language text (no masking)
- the self-attention layer in the decoder is causal (left-to-right, use masking)
- the cross-attention layer in the decoder attends to each of the source language words

Encoder-decoder with Transformer

During training, the network is given the source text and then, starting at the separator token, it is trained **autoregressively** to predict the next token using cross-entropy loss.

We use **teacher forcing**: at each time step in decoding we force the system to use the gold target token rather than the decoder output.

Already introduced in this lecture.

Beam search

Search tree

We have seen a **greedy algorithm** for inference, choosing the single most probable token at each step in decoding.

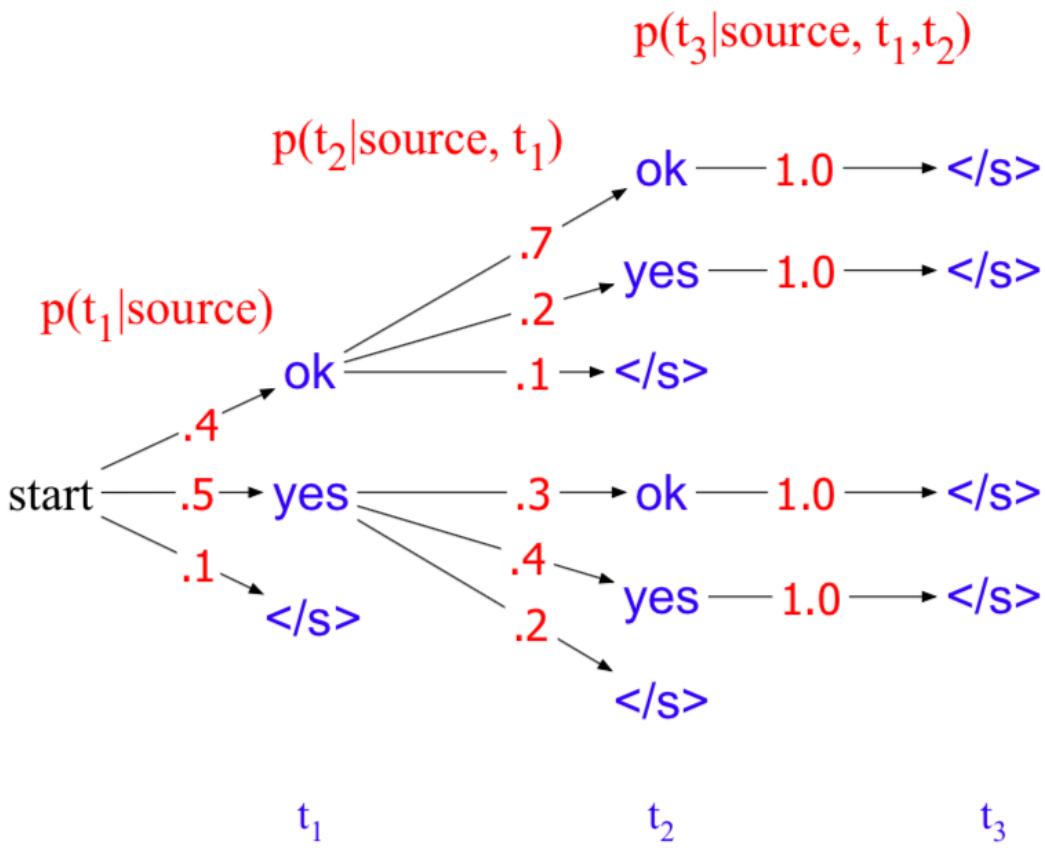
A greedy algorithm makes choices that are locally optimal, but this may not find the highest probability translation.

We define the **search tree** for the decoder

- the nodes are the states, representing the generated prefix of the translation
- the branches are the actions of generating the next token

Search tree

Search tree for the **decoder** with trace of the optimal sequence.



Beam search

Unfortunately, **dynamic programming** is not applicable to this search tree, because of long-distance dependencies between the output decisions.

The only method guaranteed to find the best solution is **exhaustive search**, which is unfeasible in practice.

In **beam search** we keep K possible hypotheses at each step; parameter K is called the **beam width**.

Already seen for syntactic parsing.

Beam search

Beam search algorithm

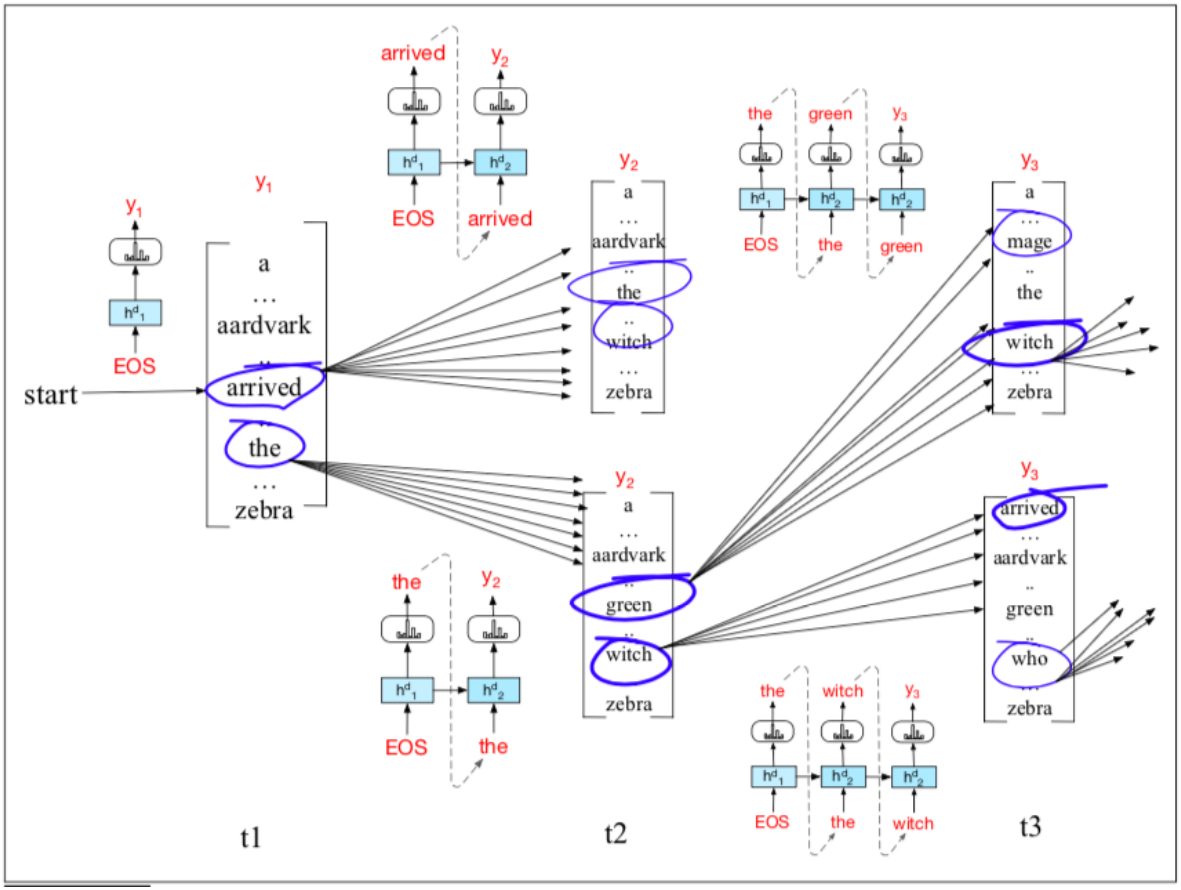
- start with K initial best hypotheses
- at each step, expand each of the K hypotheses resulting in $|V| \times K$ new hypotheses, which are all scored

  V is the vocabulary.
- prune the $|V| \times K$ hypotheses down to the K best hypotheses
- when a complete hypothesis is found, remove it from the frontier and reduce by one the size of the beam, stopping at $K = 0$

Greedy algorithm is beam search with $K = 1$.

Beam search

Beam search with beam width $K = 2$.



Beam search

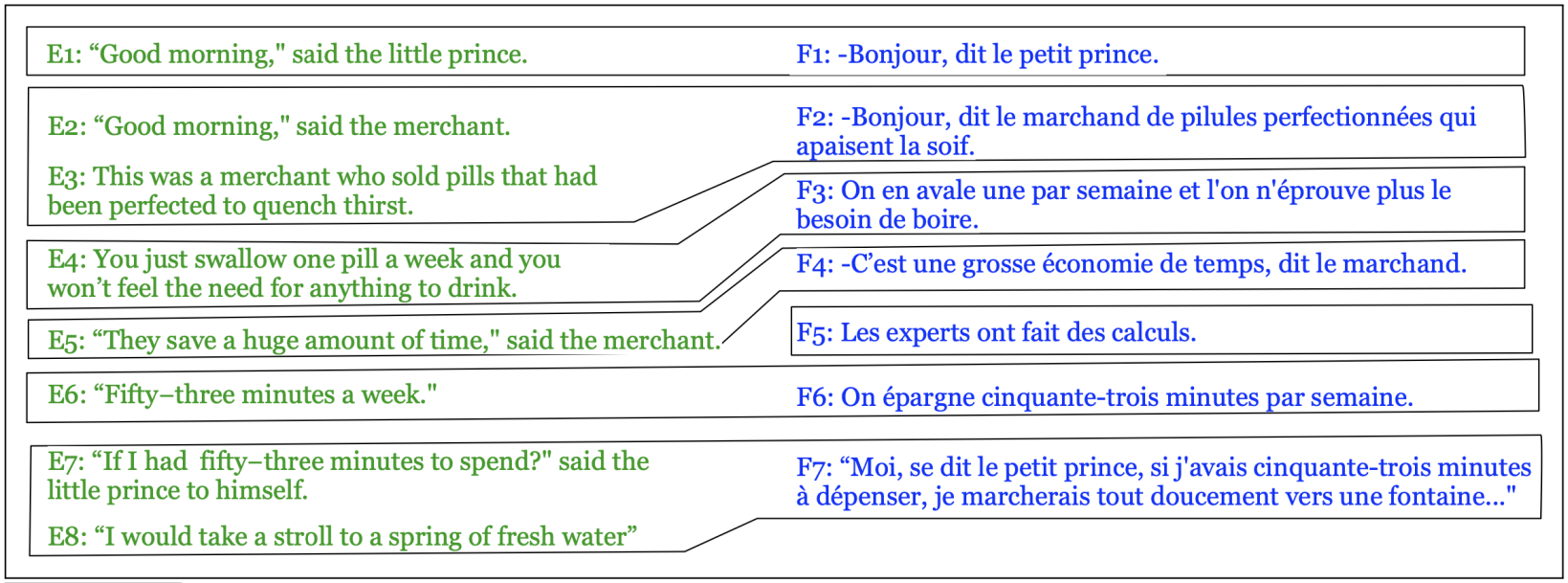
Problem: Because models generally assign lower probabilities to longer strings, the algorithm favours completed hypotheses which are shorter.

Apply some form of length **normalization** to hypotheses based on number of words.

Parallel corpora

Parallel corpora

Machine translation models are trained on **parallel corpora**, which are texts in two or more languages with **aligned** sentences.



Parallel corpora

Given two documents that are translations of each other, we can **automatically align** sentences using

- a score function measuring how likely a source span and a target span are translations
- an alignment algorithm that finds a good alignment between the documents using the scores

Score function usually exploits multi-lingual word embeddings and cosine similarity (see previous lectures).

Alignment algorithm exploits dynamic programming and are simple extension of the minimum edit distance algorithm.

Parallel corpora

Popular parallel corpora

- Canadian Hansard (English-French): extracted from the proceedings of the Canadian Parliament.
- Europarl (21 languages): 30.32M parallel sentences extracted from the proceedings of the European parliament.

  `http://www.statmt.org/europarl/`
- OPUS (several languages): growing collection of translated texts, automatically preprocessed and aligned.

  `https://opus.nlpl.eu`

Evaluation

©Wizard of the Coast

Evaluation

The most popular automatic metric for MT systems is **BLEU**, for BiLingual Evaluation Understudy.

The ***N*-gram precision** for a candidate translation is the percentage of N-grams in the target sentence that also occur in the reference translation.

N-gram precision is naturally extended to a whole translated document.

BLEU combines N-gram precisions for $N \in [1..4]$ using the geometric mean. A brevity penalty for too-short translations is also added.

Brevity penalty has the function of balancing the overall recall.

Evaluation

BLEU score ranges in 0-100%

- 50% up is a very good score
- 15% and below means bad translation quality, high level of post-editing will be required.

BLEU has been criticised for not correlating well with human judgement.

Evaluation

An aternative metric for MT systems is **METEOR**, for Metric for Evaluation of Translation with Explicit ORdering.

The metric is based on the harmonic mean of unigram precision and recall, with recall weighted higher than precision.

Along with the standard exact word matching, this metric also uses **stemming** and **synonymy** matching.

The metric was designed to fix some of the problems found in BLEU, producing better correlation with human judgement at the sentence or segment level.

Leaderboard

Leaderboard

Model	BLEU	Paper / Source
DeepL	45.9	DeepL Press release
Transformer Big + BT (Edunov et al., 2018)	45.6	Understanding Back-Translation at Scale
Admin (Liu et al., 2020)	43.8	Understand the Difficulty of Training Transformers
MUSE (Zhao et al., 2019)	43.5	MUSE: Parallel Multi-Scale Attention for Sequence to Sequence Learning
TaLK Convolutions (Lioutas et al., 2020)	43.2	Time-aware Large Kernel Convolutions
DynamicConv (Wu et al., 2019)	43.2	Pay Less Attention With Lightweight and Dynamic Convolutions

http://nlpprogress.com

English-French dataset, Ninth Workshop on Statistical Machine Translation (WMT 2014 EN-FR).

Miscellanea

©Wizard of the Coast

Libraries

Many practical implementations today are based on the OpenNMT ecosystem for neural machine translation.

`https://opennmt.net`

Research papers

©nerdarchy.com

Research papers

Title: Neural Machine Translation by Jointly Learning to Align and Translate
Authors: Dzmitry Bahdanau, Kyunghyun Cho, Yoshua Bengio
Repository: arXiv:1409.0473 19 May 2016 (version v7)
Content: The models proposed recently for neural machine translation belong to a family of encoder-decoders. In this work, we conjecture that the use of a fixed-length vector is a bottleneck, and propose to extend this by allowing a model to automatically (soft-)search for parts of a source sentence that are relevant to predicting a target word.

https://arxiv.org/abs/1409.0473

Research papers

Title: Massive Exploration of Neural Machine Translation Architectures
Authors: Denny Britz, Anna Goldie, Minh-Thang Luong, Quoc Le
Repository: arXiv.org.cs.Computation and Language, 21 Mar 2017
Content: This work presents a large-scale analysis of NMT architecture hyperparameters. Empirical results and variance numbers are reported for several hundred experimental runs, leading to novel insights and practical advice for building and extending NMT architectures.

https://arxiv.org/abs/1703.03906

Research papers

Title: Neural Machine Translation of Rare Words with Subword Units
Authors: Rico Sennrich, Barry Haddow, Alexandra Birch
Conference: ACL 2016
Content: Neural machine translation models typically operate with a fixed vocabulary, but translation is an open-vocabulary problem. We introduce an effective approach, making the NMT model capable of open-vocabulary translation by encoding rare and unknown words as sequences of subword units.

https://aclanthology.org/P16-1162/

Research papers

Title: Improved Neural Machine Translation with a Syntax-Aware Encoder and Decoder

Authors: Huadong Chen, Shujian Huang, David Chiang, Jiajun Chen

Conference: ACL 2017

Content: This paper improves the encoder-decoder model by explicitly incorporating source-side syntactic trees. Experiments on Chinese-English translation demonstrate that the model outperforms the sequential attentional model.

https://aclanthology.org/P17-1177/

Research papers

Title: Best-First Beam Search
Authors: Clara Meister, Tim Vieira, Ryan Cotterell
Journal: Transactions of the Association for Computational Linguistics, Volume 8
Content: This work shows that the standard implementation of beam search can be made up to 10x faster in practice.

https://aclanthology.org/2020.tacl-1.51/