

Natural Language Processing

Lecture : Machine Translation

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

Machine translation



©Dungeons & Dragons

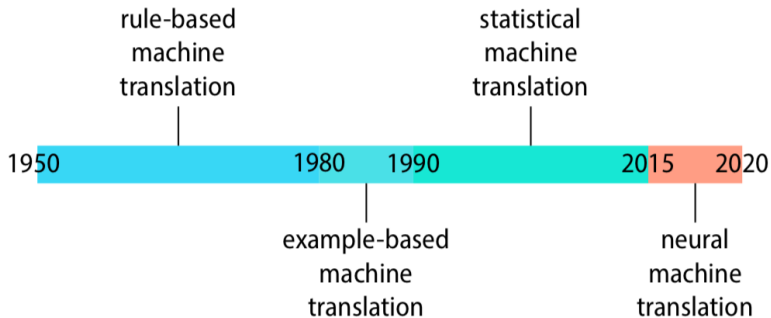
Machine translation (MT) investigates models and algorithms to translate from one language to another.

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

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.

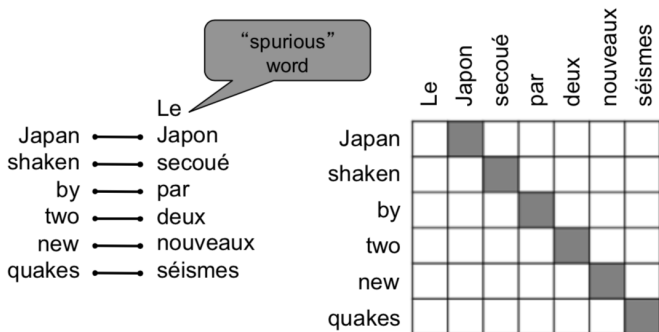
There may be a **lexical gap** between a pair of languages, where no word in the target language can express the exact meaning of a word in the source language.

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

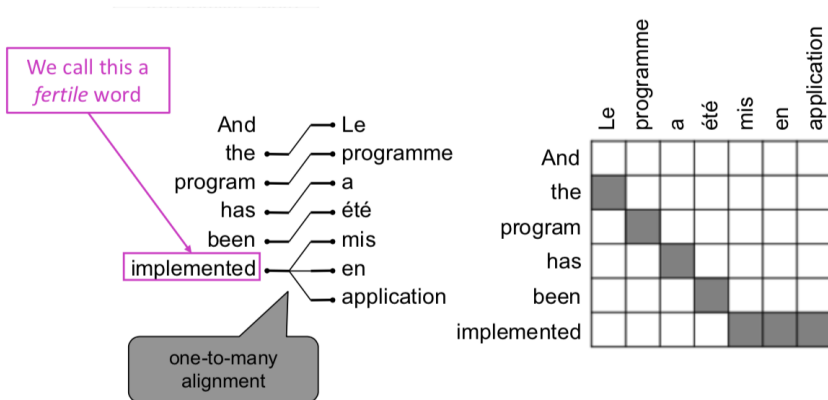
Alignment can be **many-to-one**.



	Le	reste	appartenait	aux	autochtones
The	■				
balance		■			
was			■		
the			■		
territory			■		
of				■	
the				■	
aboriginal					■
people					■

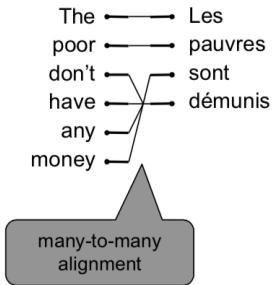
Word alignment

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



Word alignment

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



	Les	pauvres	sont	démunis
The	■			
poor		■		
don't			■	■
have			■	■
any			■	■
money			■	■

phrase alignment

Neural machine translation



©Dungeons & Dragons

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

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 $P(y | x)$, using an approach similar to the one adopted for language modeling

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

Encoder-decoder networks



©Dungeons & Dragons

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 and the output of the decoder generated so far.

Autoregressive = takes its own output as new input.

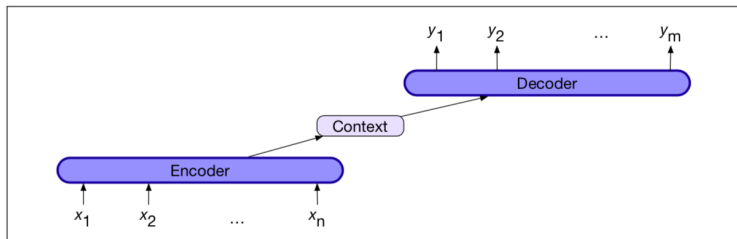
The encoder-decoder model can be implemented in two ways, through recurrent networks or else through the transformer.

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

Encoder-decoder networks

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

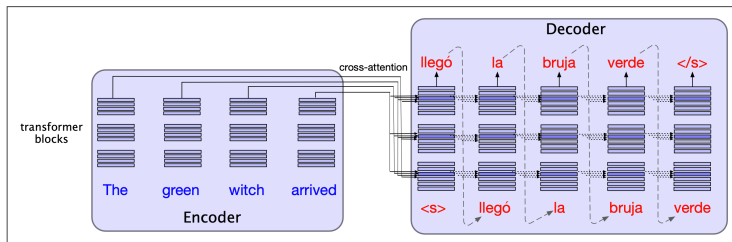
The output of the encoder is called **context** and drives the translation, together with the decoder output.



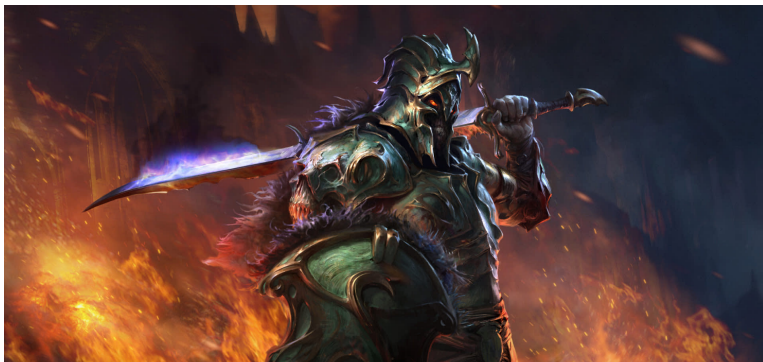
Encoder-decoder networks

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.

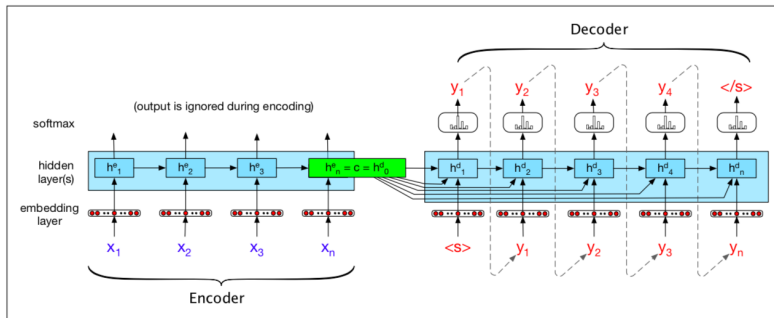
$P(y | x)$ can be computed as follows

- 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.

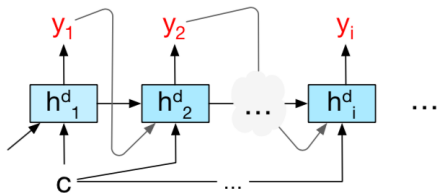
Encoder-decoder with RNN: inference

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

$$\mathbf{h}_t^e = \text{RNN}(\mathbf{h}_{t-1}^e, \mathbf{b}(x_t))$$

$$\mathbf{c} = \mathbf{h}_0^d = \mathbf{h}_n^e$$

$$\mathbf{h}_t^d = g(\mathbf{c}, \mathbf{h}_{t-1}^d, \mathbf{b}(\hat{y}_{t-1}))$$

$$\mathbf{z}_t = f(\mathbf{h}_t^d)$$

$$\mathbf{s}_t = \text{softmax}(\mathbf{z}_t) = P(\cdot \mid \hat{y}_1, \dots, \hat{y}_{t-1}, x)$$

$$\hat{y}_t = \underset{u \in V}{\text{argmax}} P(u \mid \hat{y}_1, \dots, \hat{y}_{t-1}, x)$$

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 of the forward and backward passes are concatenated.

Encoder-decoder with RNN: training



©Pinterest

Encoder-decoder with RNN: training

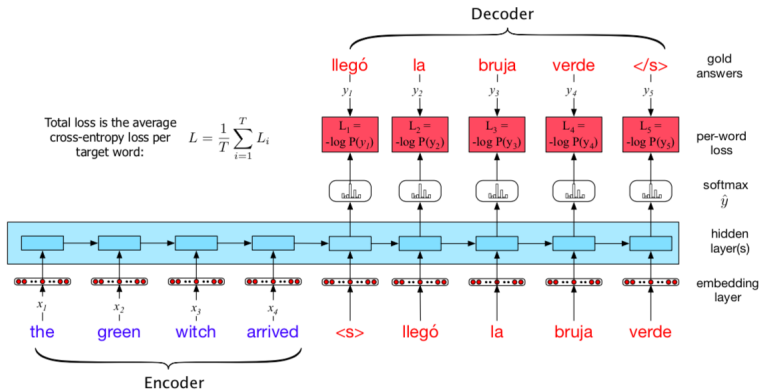
An encoder-decoder is optimized as a single system, backpropagation operates **end-to-end**.

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 .

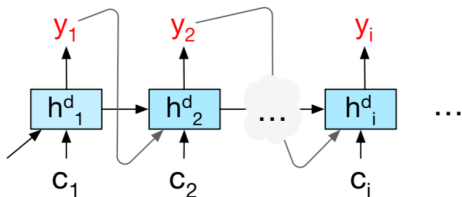
Weights are a function of i .

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.

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



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

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

$$\text{score}(\mathbf{h}_{i-1}^d, \mathbf{h}_j^e) = \mathbf{h}_{i-1}^d \cdot \mathbf{h}_j^e$$

More sophisticated scoring functions presented later.

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

$$\begin{aligned}\alpha_{ij} &= \text{softmax}(\text{score}(\mathbf{h}_{i-1}^d, \mathbf{h}_j^e)) \\ &= \frac{\exp \text{score}(\mathbf{h}_{i-1}^d, \mathbf{h}_j^e)}{\sum_k \exp \text{score}(\mathbf{h}_{i-1}^d, \mathbf{h}_k^e)}\end{aligned}$$

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

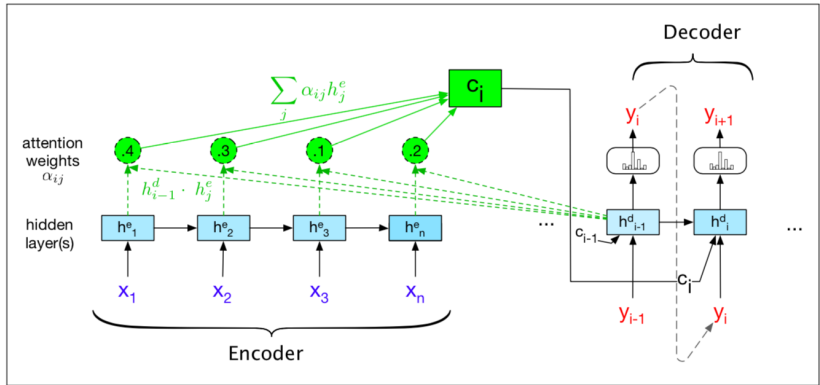
We can also view α_{ij} , $j \in [1..n]$ as a **probability distribution**.

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



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

$$\text{score}(\mathbf{h}_{i-1}^d, \mathbf{h}_j^e) = (\mathbf{h}_{i-1}^d)^\top \mathbf{W}_s \mathbf{h}_j^e$$

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.

Observe that this is also equivalent to the application of two separate learnable linear transformations to hidden states \mathbf{h}_{i-1}^d and \mathbf{h}_j^e .

Encoder-decoder with Transformer



©Wizard of the Coast

Encoder-decoder with Transformer

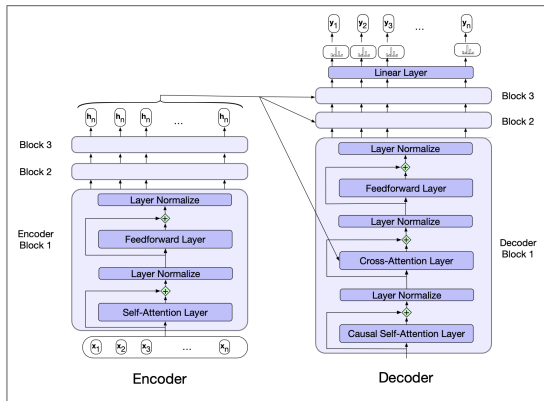
The standard architecture for neural MT is the transformer.

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.



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
- 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

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

$$\mathbf{K} = \mathbf{W}^K \mathbf{H}^{enc}$$

$$\mathbf{V} = \mathbf{W}^V \mathbf{H}^{enc}$$

$$CrossAttention(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = SoftMax\left(\frac{\mathbf{QK}^T}{\sqrt{d_k}}\right) \mathbf{V}$$

In summary

- the self-attention layer in the encoder is allowed to look 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



©Wizard of the Coast

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

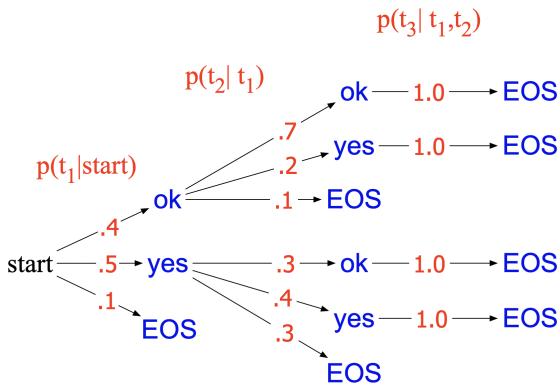
A greedy algorithm makes choices that are locally optimal, but this may not find the highest probability translation.

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 vocabulary $V = \{\text{yes}, \text{ok}, \text{EOS}\}$.



Greedy search and global search provide different translations.

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.

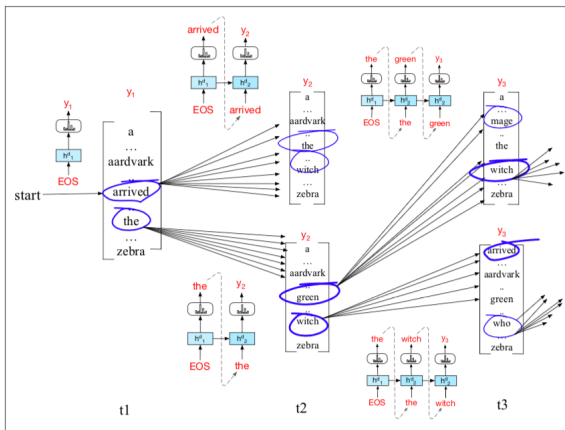
Beam search algorithm

- 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 $K = 2$.



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



©Wizard of the Coast

Parallel corpora

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

E1: "Good morning," said the little prince.	F1: -Bonjour, dit le petit prince.
E2: "Good morning," said the merchant.	F2: -Bonjour, dit le marchand de pilules perfectionnées qui apaisent la soif.
E3: This was a merchant who sold pills that had been perfected to quench thirst.	F3: On en avale une par semaine et l'on n'éprouve plus le besoin de boire.
E4: You just swallow one pill a week and you won't feel the need for anything to drink.	F4: -C'est une grosse économie de temps, dit le marchand.
E5: "They save a huge amount of time," said the merchant.	F5: Les experts ont fait des calculs.
E6: "Fifty-three minutes a week."	F6: On épargne cinquante-trois minutes par semaine.
E7: "If I had fifty-three minutes to spend?" said the little prince to himself.	F7: "Moi, se dit le petit prince, si j'avais cinquante-trois minutes à dépenser, je marcherais tout doucement vers une fontaine..."
E8: "I would take a stroll to a spring of fresh water"	

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.

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

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.

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.

An alternative 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



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).



©Wizard of the Coast

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

<https://opennmt.net>