Introduction to Artificial Intelligence

Lecture 9: Communication

LIÈGE université

Today

- A guided tour of NIPS 2017
- Natural language processing
  - Parsing
  - Semantics
- Machine translation

NIPS 2017

NIPS?

- Research conference and workshops on Neural Information Processing Systems (NIPS).
- #1 conference on machine learning, artificial intelligence and computational neuroscience.
- Pre-proceedings available for download.

Stats

Total Registration	8000 (vs. 5000 last year)
Tracks	2
Submissions	3240 (vs. 2500 last year)
Subject Areas	156 (150% increase)
Top Area:	Algorithms (900), Deep Learning (600), Applications (600)
Unique Authors	7,844
Author demographics	90% men, 10% women
More	Industry 12%, 88% Academic
Reviewers	2093
Area Chairs	183
Reviews	9847
Acceptances	679
Acceptance Rate	21%
Orals	40
Spotlights	112
Posters	527
Paper on arXiv?	43% Yes
Reviewer saw on arXiv?	10% Yes
Not Posted Acceptance?	If not online, 15% accepted
Posted Acceptance?	If online, 29% accepted
Reviewer saw online	If reviewer saw, 35%

Highlights

Highlights

Geometric Deep Learning on Graphs and Manifold

Highlights

Reverse-Engineering Intelligence Using Probabilistic Programs

Highlights

Deep Learning for Robotics

Highlights

AlphaZero

Highlights

The Trouble with Bias

Highlights

Highlights

Deep Probabilistic Modelling with Gaussian Processes

Highlights

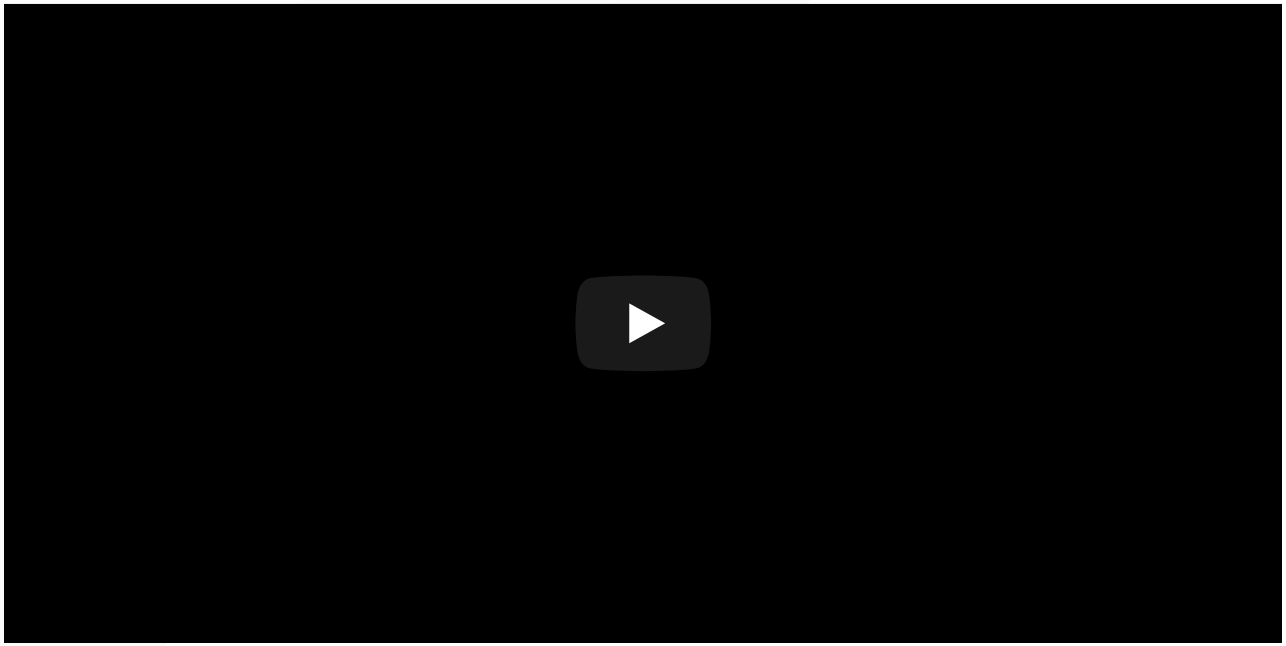
Learning to Run competition

Topics to watch out for

- Generative adversarial networks
- Hierarchical reinforcement learning
- Probabilistic programming
- Meta-learning
- Leveraging simulation
- Bayesian deep learning
- Dealing with data bias
- Optimal transport
- Emergent communication

Academia vs Private labs

- NIPS remains an academic research conference.
- But there is an increasing presence of strong private research labs.
  - Google Brain, Google DeepMind, NVidia, Facebook, IBM, Intel, Microsoft, etc.
  - All are aggressively hiring!



Natural language processing

Natural Language Processing

- Fundamental goal:
  - Analyze and process human language, broadly, robustly, accurately, ...
- End systems that we want to build:
  - Ambitious: speech recognition, machine translation, information extraction, dialog interfaces, question answering, etc.
  - Modest: spelling correction, text categorization, etc.

Credits: UC Berkeley, CS188

Probabilistic context-free grammar

- A grammar is a collection of rules that defines a language as a set of allowable strings of words.
- Probabilistic context-free grammars are grammars such that:
  - production rules do not depend on context (context-free);
  - a probability is assigned to every string (probabilistic).
- Example:
  - $VP \rightarrow Verb[0.7] | VP\ NP[0.3]$
- Let ξ_0 be a language suitable for communication between agents exploring the Wumpus world.

Lexicon of ξ_0

Noun	→	**stench** [0.05] | **breeze** [0.10] | **wumpus** [0.15] | **pits** [0.05] | ...
Verb	→	**is** [0.10] | **feel** [0.10] | **smells** [0.10] | **stinks** [0.05] | ...
Adjective	→	**right** [0.10] | **dead** [0.05] | **smelly** [0.02] | **breezy** [0.02] ...
Adverb	→	**here** [0.05] | **ahead** [0.05] | **nearby** [0.02] | ...
Pronoun	→	**me** [0.10] | **you** [0.03] | **I** [0.10] | **it** [0.10] | ...
RelPro	→	**that** [0.40] | **which** [0.15] | **who** [0.20] | **whom** [0.02] ∨ ...
Name	→	**John** [0.01] | **Mary** [0.01] | **Boston** [0.01] | ...
Article	→	**the** [0.40] | **a** [0.30] | **an** [0.10] | **every** [0.05] | ...
Prep	→	**to** [0.20] | **in** [0.10] | **on** [0.05] | **near** [0.10] | ...
Conj	→	**and** [0.50] | **or** [0.10] | **but** [0.20] | **yet** [0.02] ∨ ...
Digit	→	**0** [0.20] | **1** [0.20] | **2** [0.20] | **3** [0.20] | **4** [0.20] | ...

Figure 23.1 The lexicon for $\mathcal{E}_0$. *RelPro* is short for relative pronoun, *Prep* for preposition, and *Conj* for conjunction. The sum of the probabilities for each category is 1.

Grammar of ξ_0

$\mathcal{E}_0$:	S	$\rightarrow$	$NP\ VP$	[0.90]	I + feel a breeze
		|	$S\ Conj\ S$	[0.10]	I feel a breeze + and + It stinks
	NP	$\rightarrow$	$Pronoun$	[0.30]	I
		|	$Name$	[0.10]	John
		|	$Noun$	[0.10]	pits
		|	$Article\ Noun$	[0.25]	the + wumpus
		|	$Article\ Adjs\ Noun$	[0.05]	the + smelly dead + wumpus
		|	$Digit\ Digit$	[0.05]	3 4
		|	$NP\ PP$	[0.10]	the wumpus + in 1 3
		|	$NP\ RelClause$	[0.05]	the wumpus + that is smelly
	VP	$\rightarrow$	$Verb$	[0.40]	stinks
		|	$VP\ NP$	[0.35]	feel + a breeze
		|	$VP\ Adjective$	[0.05]	smells + dead
		|	$VP\ PP$	[0.10]	is + in 1 3
		|	$VP\ Adverb$	[0.10]	go + ahead
	$Adjs$	$\rightarrow$	$Adjective$	[0.80]	smelly
		|	$Adjective\ Adjs$	[0.20]	smelly + dead
	PP	$\rightarrow$	$Prep\ NP$	[1.00]	to + the east
	$RelClause$	$\rightarrow$	$RelPro\ VP$	[1.00]	that + is smelly

Figure 23.2 The grammar for $\mathcal{E}_0$, with example phrases for each rule. The syntactic categories are sentence (S), noun phrase (NP), verb phrase (VP), list of adjectives ($Adjs$), prepositional phrase (PP), and relative clause ($RelClause$).

- Unfortunately, this grammar overgenerates:
  - It generates sentences that are not grammatical (in English).
  - e.g., "Me go Boston" or "I smell pits wumpus John"
- It also undergenerates:
  - Many English sentences are rejected.
  - e.g., "I think the wumpus is smelly."

Parse tree

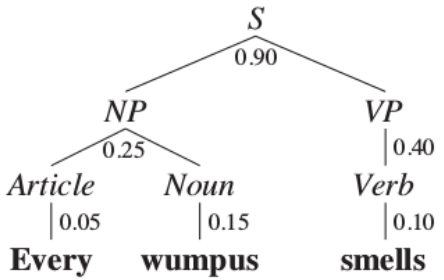


Figure 23.3 Parse tree for the sentence "Every wumpus smells" according to the grammar $\mathcal{E}_0$. Each interior node of the tree is labeled with its probability. The probability of the tree as a whole is $0.9 \times 0.25 \times 0.05 \times 0.15 \times 0.40 \times 0.10 = 0.0000675$. Since this tree is the only parse of the sentence, that number is also the probability of the sentence. The tree can also be written in linear form as [*S* [*NP* [*Article* **every**] [*Noun* **wumpus**]][*VP* [*Verb* **smells**]]].

Syntactic analysis

S	
$NP\ VP$	$S \rightarrow NP\ VP$
$NP\ VP\ Adjective$	$VP \rightarrow VP\ Adjective$
$NP\ Verb\ Adjective$	$VP \rightarrow Verb$
$NP\ Verb$ **dead**	$Adjective \rightarrow$ **dead**
NP **is dead**	$Verb \rightarrow$ **is**
$Article\ Noun$ **is dead**	$NP \rightarrow Article\ Noun$
$Article$ **wumpus is dead**	$Noun \rightarrow$ **wumpus**
the wumpus is dead	$Article \rightarrow$ **the**

Figure 23.4 Trace of the process of finding a parse for the string "The wumpus is dead" as a sentence, according to the grammar $\mathcal{E}_0$. Viewed as a top-down parse, we start with the list of items being S and, on each step, match an item X with a rule of the form $(X \rightarrow \ldots)$ and replace X in the list of items with $(\ldots)$. Viewed as a bottom-up parse, we start with the list of items being the words of the sentence, and, on each step, match a string of tokens $(\ldots)$ in the list against a rule of the form $(X \rightarrow \ldots)$ and replace $(\ldots)$ with X.

- Parsing is the process of analyzing a string of words to uncover its phrase structure.
- This process can be carried out efficiently using the CYK algorithm:
  - Imagine a state-action space where actions correspond to production rules.
  - Use A* to search the space efficiently until the string has been compressed to a single item S.

(demo)

Ambiguity

- There may be several distinct parse trees for a given sentence.
- E.g., the sentence "Fall leaves fall and spring leaves spring." admit 4 parse trees:

$$
\begin{array}{l}
[S\ [S\ [NP\ \text{Fall leaves}]\ \text{fall}]\ \text{and}\ [S\ [NP\ \text{spring leaves}]\ \text{spring}] \\
[S\ [S\ [NP\ \text{Fall leaves}]\ \text{fall}]\ \text{and}\ [S\ \text{spring}\ [VP\ \text{leaves spring}]] \\
[S\ [S\ \text{Fall}\ [VP\ \text{leaves fall}]]\ \text{and}\ [S\ [NP\ \text{spring leaves}]\ \text{spring}] \\
[S\ [S\ \text{Fall}\ [VP\ \text{leaves fall}]]\ \text{and}\ [S\ \text{spring}\ [VP\ \text{leaves spring}]]
\end{array}
$$

- With A*:
  - Define the cost of a state as the inverse of its probability as defined by the rules applied so far.
  - Estimate the remaining distance using machine learning.
  - With very high probability, the procedure yields the most probable tree.

Learning PCFGs

- A PCFG has many rules, with a probability for each rule.
- Learning the grammar might be better than a knowledge engineering approach.
- If we are given a corpus of correctly parsed sentences, the rules and their probability can be estimated directly from this data.
    - E.g., count nodes S and nodes $[S[NP...][VP...]]$ over all trees to estimate the probability of $S \rightarrow NPVP$.
- If we dont have labeled sentences, then we have to learn both the rules and their probability.
    - This is more complicated, but several algorithms exist:
        - Assume (or cross-validate) a number of categories $X, Y, Z, ...$
        - Assume the grammar includes every possible rule $X \rightarrow YZ$ or $X \rightarrow word$.
        - Use an expectation-minimization algorithm estimate the probabilities.

Semantic interpretation

$$
\begin{aligned}
Exp(x) &\rightarrow Exp(x_1)\ Operator(op)\ Exp(x_2)\ \{x = Apply(op, x_1, x_2)\} \\
Exp(x) &\rightarrow (\ Exp(x)\ ) \\
Exp(x) &\rightarrow Number(x) \\
Number(x) &\rightarrow Digit(x) \\
Number(x) &\rightarrow Number(x_1)\ Digit(x_2)\ \{x = 10 \times x_1 + x_2\} \\
Digit(x) &\rightarrow x\ \{0 \leq x \leq 9\} \\
Operator(x) &\rightarrow x\ \{x \in \{+, -, \div, \times\}\}
\end{aligned}
$$

Figure 23.8 A grammar for arithmetic expressions, augmented with semantics. Each variable x_i represents the semantics of a constituent. Note the use of the $\{test\}$ notation to define logical predicates that must be satisfied, but that are not constituents.

- Semantics can be added to each rule of a grammar.
- The rules obey the principle of compositional semantics:
  - The semantics of a phrase is a function of the semantics of the subphrases.
- In general, semantics can be properly defined with first-order logic, where nodes are either associated to a logical term, a logical sentence or a predicate.

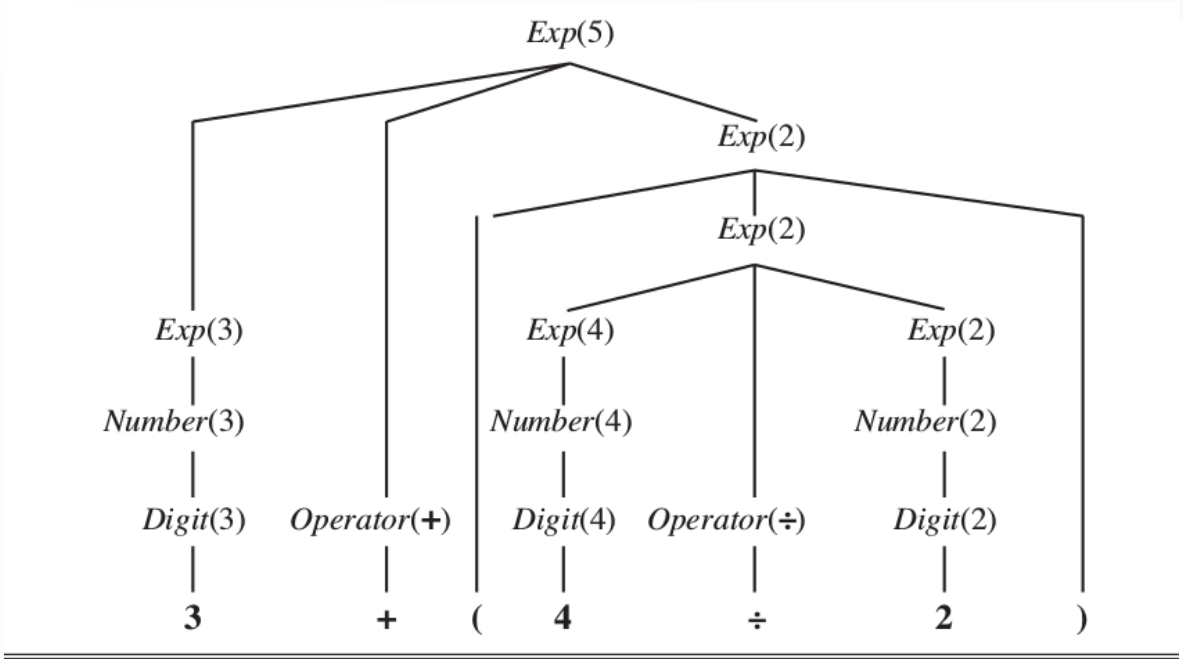


Figure 23.9 Parse tree with semantic interpretations for the string "$3 + (4 \div 2)$".

Real language

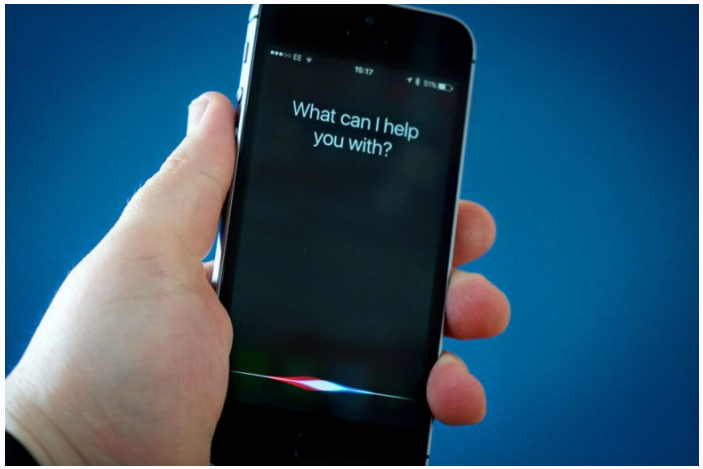
- A real language like English is difficult to map to a grammar and to a semantics, because of the following reasons:
  - Long-distance dependencies
  - Ambiguities (lexical, syntax, semantic)
  - Metonymies (figure of speech)
  - Metaphors
- Recovering the most probable intended meaning is called disambiguation. To do disambiguation properly, we need to combine:
  - The world model: the likelihood that a proposition occurs in the world.
  - The mental model: the likelihood that the speaker forms the intention of communicating some fact to the hearer.
  - The language model: the likelihood that a certain string of words will be chosen.
  - The acoustic model: the likelihood that a certain sequence of sounds will be generated.
- Combining perfectly all these pieces together remains an open problem!

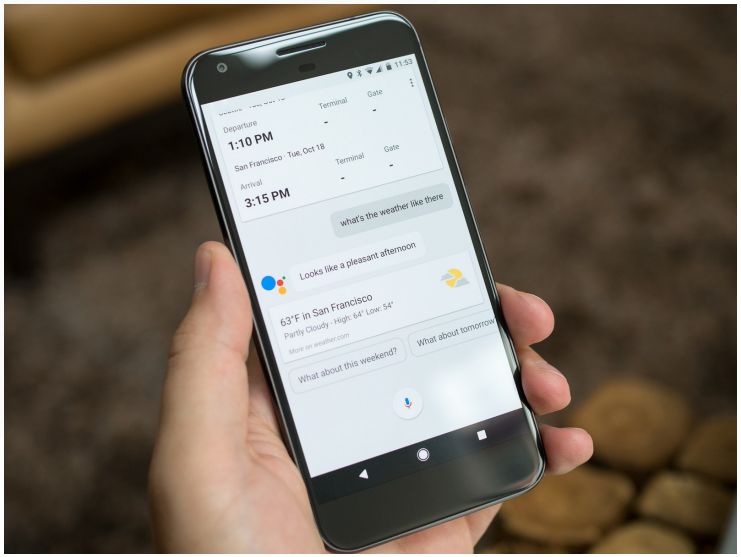
Eliza

- Eliza is one of the earliest instances of a chatterbot (Weizenbaum, 1964).
- Led to a long line of chatterbots.
- How does it work?
  - Trivial NLP: string match and substitution.
  - Trivial knowledge: tiny script / response database.
  - Example: matching "I remember " results in "Do you often think of ?"

Credits: UC Berkeley, CS188

Modern chatbots





Siri, Google Assistant, Alexa, etc

Machine translation

- Translate text from one language to another, while preserving the intended meaning.
- Recombines fragments of example translations.
- Challenges:
  - What fragments? [Learning to translate]
  - How to make efficient? [Fast translation search]

Credits: UC Berkeley, CS188

Issue of dictionary lookups

顶部	/**top**/roof/
顶端	/summit/peak/**top**/apex/
顶头	/coming directly towards one/**top**/end/
盖	/lid/**top**/cover/canopy/build/Gai/
盖帽	/surpass/**top**/
极	/extremely/pole/utmost/**top**/collect/receive/
尖峰	/peak/**top**/
面	/fade/side/surface/aspect/**top**/face/flour/
摘心	/**top**/topping/

Credits: UC Berkeley, CS188

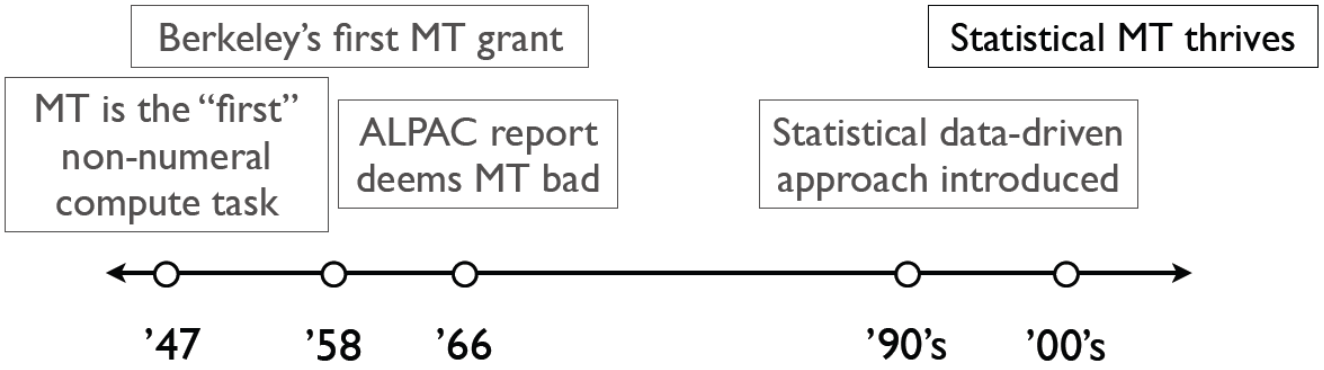
History

When I look at an article in Russian, I say: "This is really written in English, but it has been coded in some strange symbols. I will now proceed to decode."

Warren Weaver

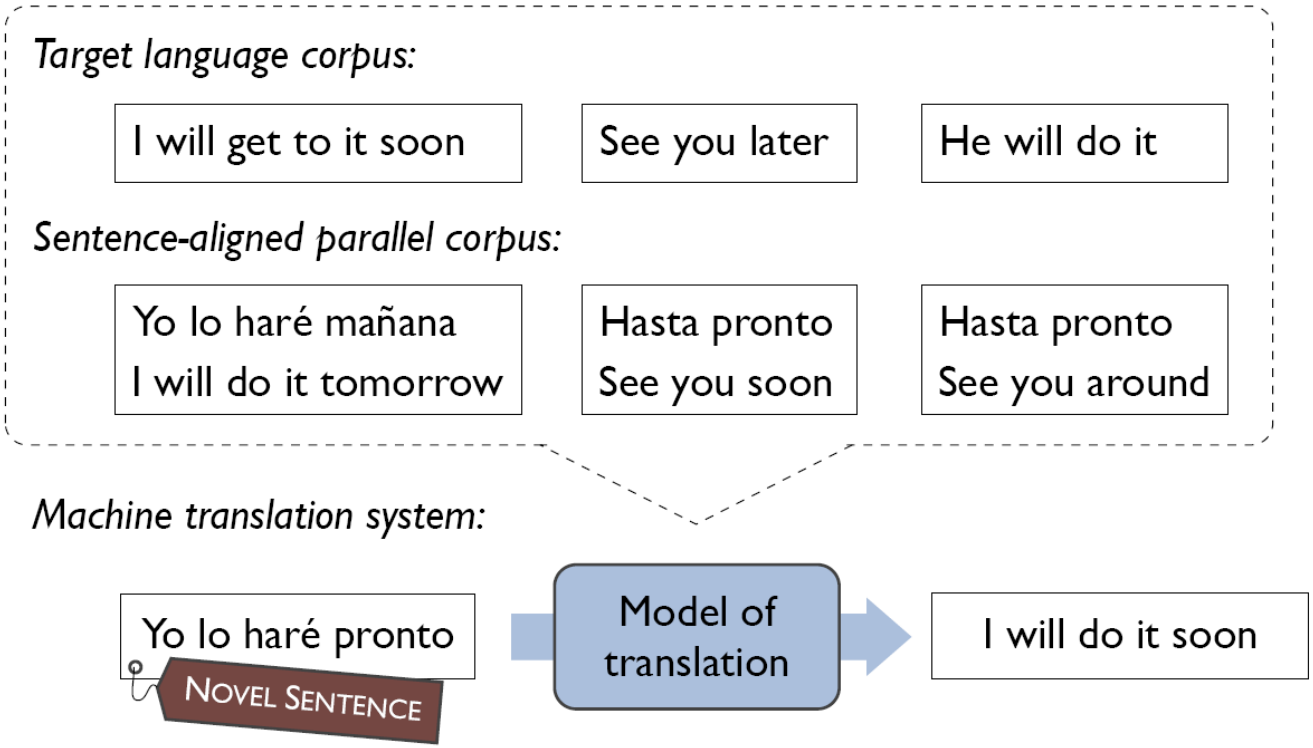
"Machine Translation" presumably means going by algorithm from machine-readable source text to useful target text... In this context, there has been no machine translation...

John Pierce



Credits: UC Berkeley, CS188

Data-driven machine translation



Credits: UC Berkeley, CS188

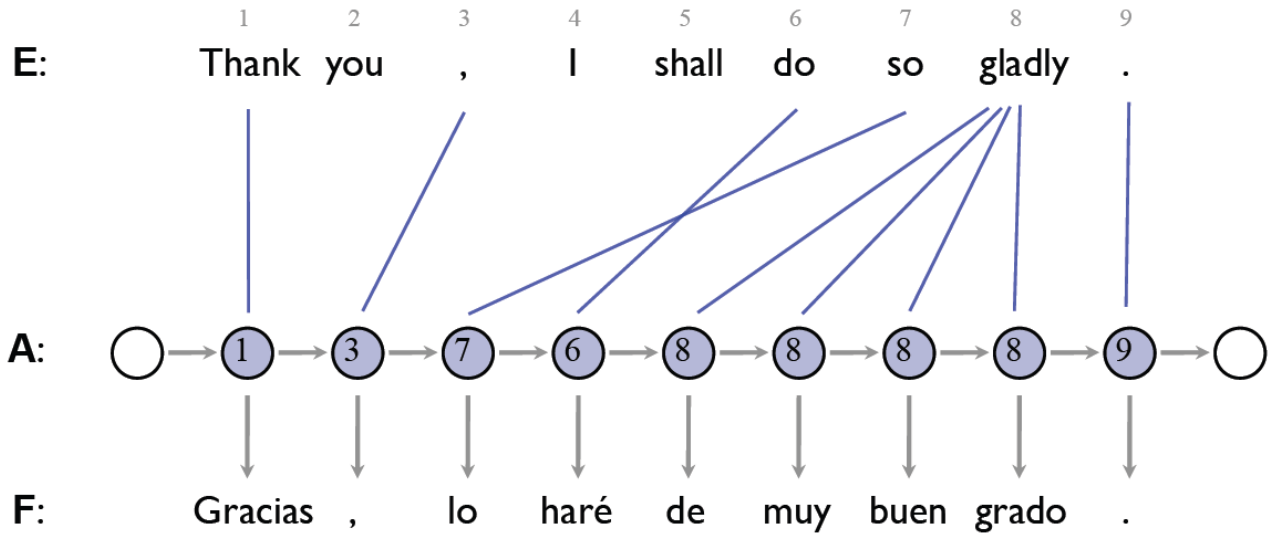
Statistical machine translation

To translate a sentence in English (e) into French (f), we seek the strings of words f^* that maximizes

$$f^* = \arg\max_f P(f|e) = \arg\max_f P(e|f)P(f)$$

- $P(f)$ is the language model
- $P(e|f)$ is the translation model (from French to English)

HMM translation model

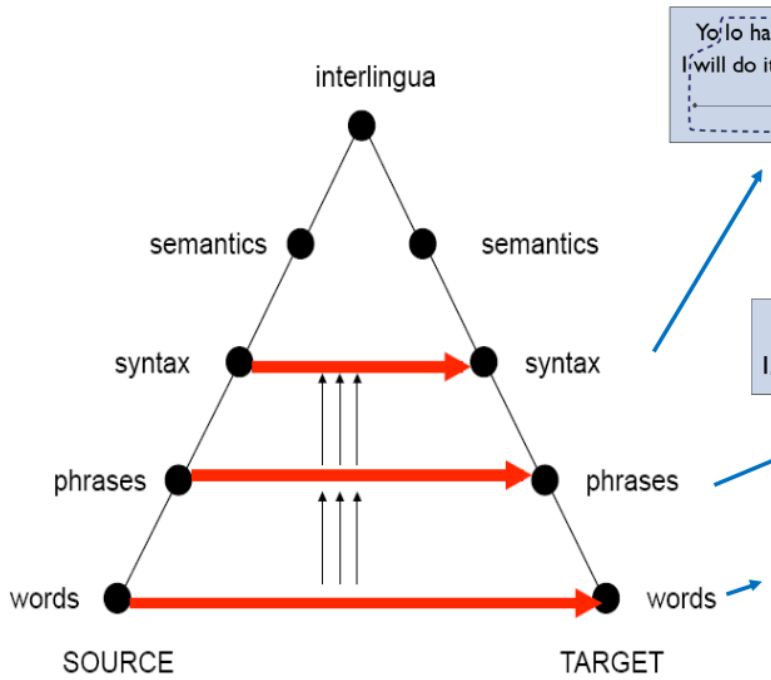


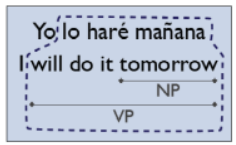
Model Parameters

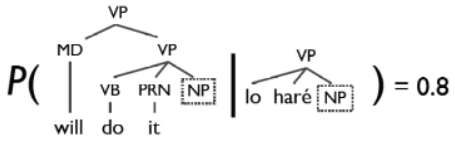
Emissions: $P(F_1 = \text{Gracias} \mid E_{A1} = \text{Thank})$ *Transitions:* $P(A_2 = 3 \mid A_1 = 1)$

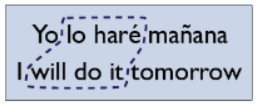
Credits: UC Berkeley, CS188

Levels of transfer

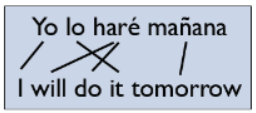








English (E)	*P*(E | lo haré)
will do it	0.8
will do so	0.2

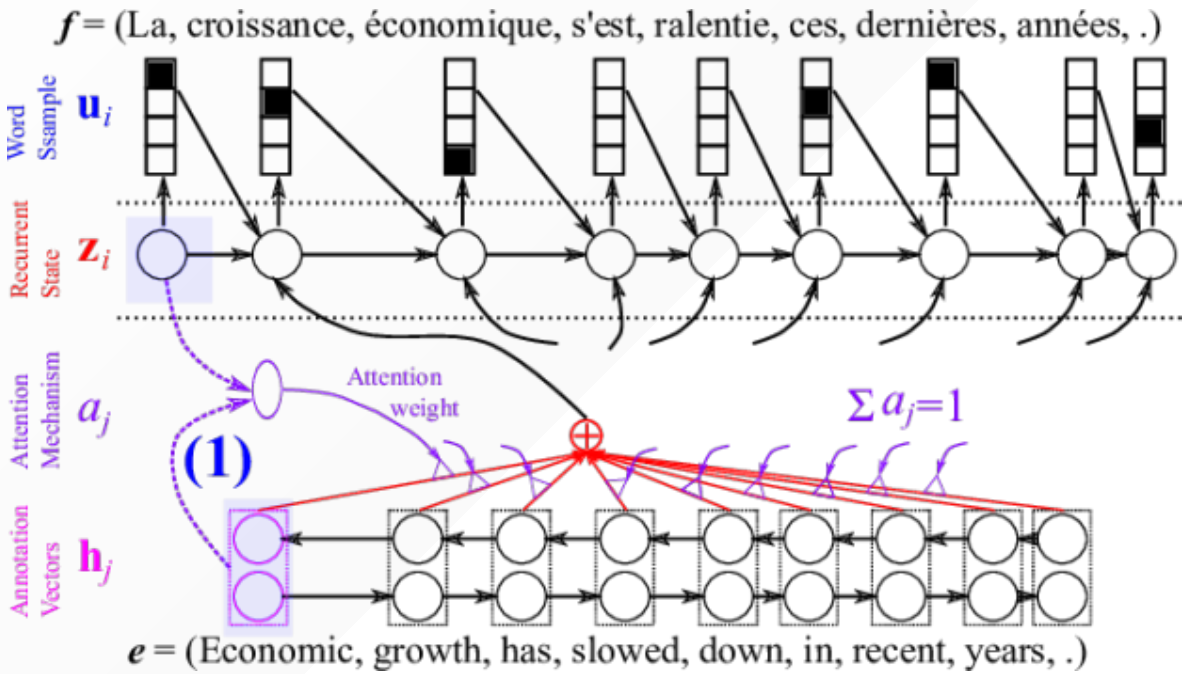


English (E)	*P*(E | mañana)
tomorrow	0.7
morning	0.3

Credits: UC Berkeley, CS188

Neural translation

- Modern machine translation systems are all based on neural networks.



Unsupervised machine translation (1)

- The latest approaches (e.g., arXiv:1711.00043) do not even need to have a bilingual corpus!
- Machine translation can be learned in a fully unsupervised way with unsupervised alignment.

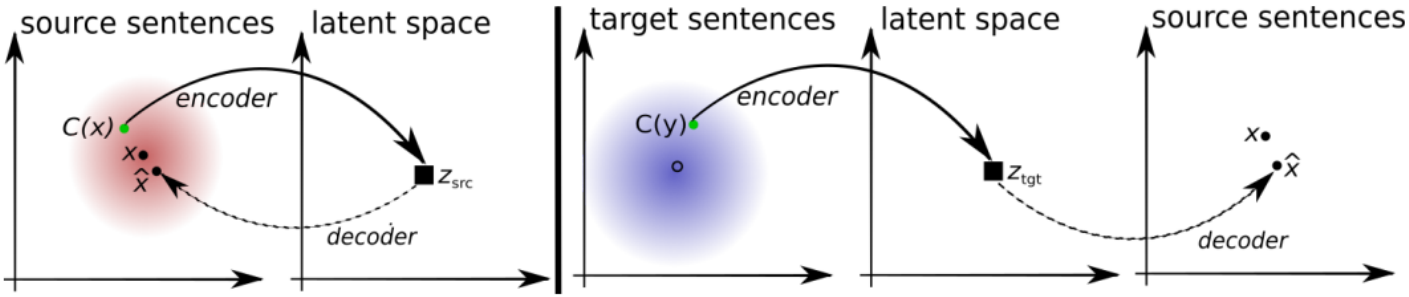


Figure 1: Toy illustration of the principles guiding the design of our objective function. Left (auto-encoding): the model is trained to reconstruct a sentence from a noisy version of it. x is the target, $C(x)$ is the noisy input, $\hat{x}$ is the reconstruction. Right (translation): the model is trained to translate a sentence in the other domain. The input is a noisy translation (in this case, from source-to-target) produced by the model itself, M, at the previous iteration (t), $y = M^{(t)}(x)$. The model is symmetric, and we repeat the same process in the other language. See text for more details.

Unsupervised machine translation (2)

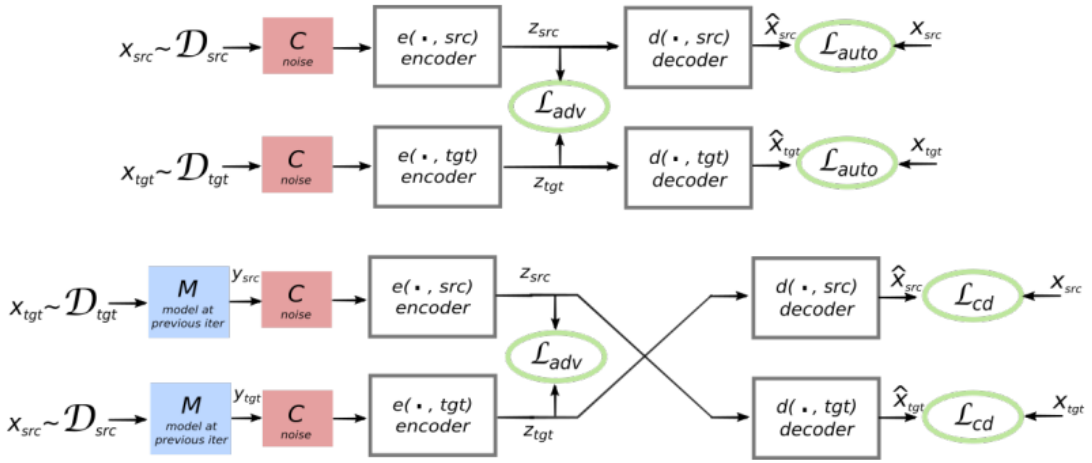


Figure 2: Illustration of the proposed architecture and training objectives. The architecture is a sequence to sequence model, with both encoder and decoder operating on two languages depending on an input language identifier that swaps lookup tables. Top (auto-encoding): the model learns to denoise sentences in each domain. Bottom (translation): like before, except that we encode from another language, using as input the translation produced by the model at the previous iteration (light blue box). The green ellipses indicate terms in the loss function.

References

- Vogel, S. et al. "HMM-based word alignment in statistical translation.", 1996.
- Bahdanau, Dzmitry, Kyunghyun Cho, and Yoshua Bengio. "Neural machine translation by jointly learning to align and translate." arXiv preprint arXiv:1409.0473 (2014).
- Lample, Guillaume, Ludovic Denoyer, and Marc'Aurelio Ranzato. "Unsupervised Machine Translation Using Monolingual Corpora Only." arXiv preprint arXiv:1711.00043 (2017).