Machine Translation 2: Statistical MT: Phrase-Based and Neural

Ondřej Bojar
bojar@ufal.mff.cuni.cz
Institute of Formal and Applied Linguistics
Faculty of Mathematics and Physics
Charles University, Prague

Outline of Lectures on MT

1. Introduction.
   - Why is MT difficult.
   - MT evaluation.
   - Approaches to MT.
   - First peek into phrase-based MT
   - Document, sentence and word alignment.
2. Statistical Machine Translation.
   - Phrase-based: Assumptions, beam search, key issues.
   - Neural MT: Sequence-to-sequence, attention, self-attentive.
3. Advanced Topics.
   - Linguistic Features in SMT and NMT.
   - Multilinguality, Multi-Task, Learned Representations.

Outline of MT Lecture 2

1. What makes MT statistical.
   - Brute-force statistical MT.
   - Noisy channed model.
   - Log-linear model.
2. Phrase-based translation model.
   - Phrase extraction.
   - Decoding (gradual construction of hypotheses).
   - Minimum error-rate training (weight optimization).
3. Neural machine translation (NMT).
   - Sequence-to-sequence, with attention.

Quotes

Warren Weaver (1949):

> I have a text in front of me which is written in Russian but I am going to pretend that it is really written in English and that is has been coded in some strange symbols. All I need to do is strip off the code in order to retrieve the information contained in the text.

Noam Chomsky (1969):

> . . . the notion "probability of a sentence" is an entirely useless one, under any known interpretation of this term.

Frederick Jelinek (80's; IBM; later JHU and sometimes ÚFAL)

> Every time I fire a linguist, the accuracy goes up.

Hermann Ney (RWTH Aachen University):

> MT = Linguistic **M**odelling + Statistical Decision **T**heory

The Statistical Approach

(Statistical = Information-theoretic.)

- Specify a probabilistic model.
  = How is the probability mass distributed among possible outputs given observed inputs.
- Specify the training criterion and procedure.
  = How to learn free parameters from training data.

Notice:

- Linguistics helpful when designing the models:
  - How to divide input into smaller units.
  - Which bits of observations are more informative.

Statistical MT

Given a source (foreign) language sentence $f_1^J = f_1 \dots f_j \dots f_J$,
Produce a target language (English) sentence $e_1^I = e_1 \dots e_j \dots e_I$.

Among all possible target language sentences, choose the sentence with the highest probability:

$$\hat{e}_1^{\hat{I}} = \operatorname*{argmax}_{I, e_1^I} p(e_1^I | f_1^J) \tag{1}$$

We stick to the e_1^I, f_1^J notation despite translating from English to Czech.

Brute-Force MT (1/2)

Translate only sentences listed in a "translation memory" (TM):

Good morning.	=	Dobré ráno.
How are you?	=	Jak se máš?
How are you?	=	Jak se máte?

$$p(e_1^I|f_1^J) = \begin{cases} 1 & \text{if } e_1^I = f_1^J \text{ seen in the TM} \\ 0 & \text{otherwise} \end{cases} \quad (2)$$

Any problems with the definition?

Brute-Force MT (2/2)

Translate only sentences listed in a "translation memory" (TM):

Good morning.	=	Dobré ráno.
How are you?	=	Jak se máš?
How are you?	=	Jak se máte?

$$p(e_1^I|f_1^J) = \begin{cases} 1 & \text{if } e_1^I = f_1^J \text{ seen in the TM} \\ 0 & \text{otherwise} \end{cases} \tag{3}$$

- Not a probability. There may be f_1^J, s.t. $\sum_{e_1^I} p(e_1^I|f_1^J) > 1$.

  $\Rightarrow$ Have to normalize, use $\frac{\text{count}(e_1^I, f_1^J)}{\text{count}(f_1^J)}$ instead of 1.
- Not "smooth", no generalization:

Good morning.	$\Rightarrow$	Dobré ráno.
Good evening.	$\Rightarrow$	$\emptyset$

Bayes' Law

Bayes' law for conditional probabilities: $p(a|b) = \frac{p(b|a)p(a)}{p(b)}$

So in our case:

$$\hat{e}_1^{\hat{I}} = \underset{I,e_1^I}{\mathrm{argmax}}\, p(e_1^I|f_1^J)$$

Apply Bayes' law

$$= \underset{I,e_1^I}{\mathrm{argmax}}\, \frac{p(f_1^J|e_1^I)p(e_1^I)}{p(f_1^J)}$$

$p(f_1^J)$ constant
$\Rightarrow$ irrelevant in maximization

$$= \underset{I,e_1^I}{\mathrm{argmax}}\, p(f_1^J|e_1^I)p(e_1^I)$$

Also called "Noisy Channel" model.

Motivation for Noisy Channel

$$\hat{e}_1^{\hat{I}} = \operatorname*{argmax}_{I,e_1^I} p(f_1^J|e_1^I)p(e_1^I) \quad (4)$$

Bayes' law divided the model into components:

$p(f_1^J|e_1^I)$ Translation model ("reversed", $e_1^I \rightarrow f_1^J$)

. . . is it a likely translation?

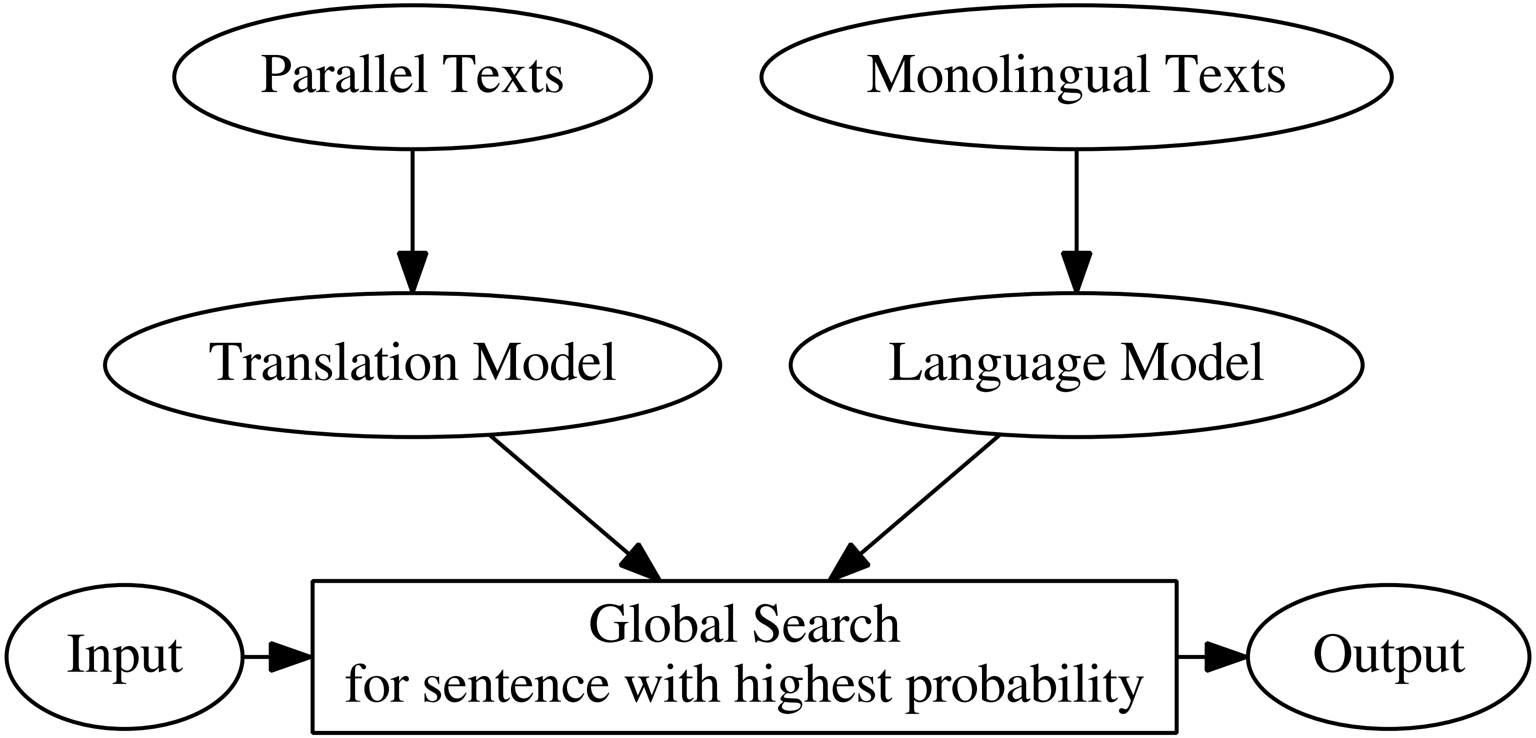
$p(e_1^I)$ Language model (LM)

. . . is the output a likely sentence of the target language?

- The components can be trained on different sources.

  There are far more monolingual data $\Rightarrow$ language model more reliable.

Without Equations



Summary of Language Models

- $p(e_1^I)$ should report how "good" sentence e_1^I is.
- We surely want $p(\text{The the the.}) < p(\text{Hello.})$
- How about $p(\text{The cat was black.}) < p(\text{Hello.})$?

  ... We don't really care in MT. We hope to compare synonymic sentences.

LM is usually a 3-gram language model:

$$p(\Rsh\ \Rsh\ \text{The cat was black .}\ \Lsh\ \Lsh) = p(\text{The}|\Rsh\ \Rsh)\ p(\text{cat}|\Rsh\ \text{The})\ p(\text{was}|\text{The cat})\ p(\text{black}|\text{cat was})\ p(.|\text{was black})\ p(\Lsh\ |\text{black .})\ p(\Lsh\ |.\ \Lsh)$$

Formally, with $n = 3$:

$$p_{\mathrm{LM}}(e_1^I) = \prod_{i=1}^{I} p(e_i|e_{i-n+1}^{i-1}) \quad (5)$$

Estimating and Smoothing LM

$$p(w_1) = \frac{\text{count}(w_1)}{\text{total words observed}}$$ Unigram probabilities.

$$p(w_2|w_1) = \frac{\text{count}(w_1 w_2)}{\text{count}(w_1)}$$ Bigram probabilities.

$$p(w_3|w_2, w_1) = \frac{\text{count}(w_1 w_2 w_3)}{\text{count}(w_1 w_2)}$$ Trigram probabilities.

Unseen ngrams ($p(\textit{ngram}) = 0$) are a big problem, invalidate whole sentence: $p_{\text{LM}}(e_1^I) = \cdots 0 \cdots = 0$

⇒ Back-off with shorter ngrams:

$$p_{\text{LM}}(e_1^I) = \prod_{i=1}^{I} \Big(\begin{array}{l} 0.8 \cdot p(e_i|e_{i-1}, e_{i-2}) + \\ 0.15 \cdot p(e_i|e_{i-1}) + \\ 0.049 \cdot p(e_i) + \\ 0.001 \end{array} \Big) \neq 0 \quad (6)$$

From Bayes to Log-Linear Model

Och (2002) discusses some problems of Equation 19:

- Models estimated unreliably $\Rightarrow$ maybe LM more important:

$$\hat{e}_1^{\hat{I}} = \operatorname*{argmax}_{I, e_1^I} p(f_1^J | e_1^I) (p(e_1^I))^2 \quad (7)$$

- In practice, "direct" translation model equally good:

$$\hat{e}_1^{\hat{I}} = \operatorname*{argmax}_{I, e_1^I} p(e_1^I | f_1^J) p(e_1^I) \quad (8)$$

- Complicated to *correctly* introduce other dependencies.
  $\Rightarrow$ Use log-linear model instead.

Log-Linear Model (1)

- $p(e_1^I|f_1^J)$ is modelled as a weighted combination of models, called "feature functions": $h_1(\cdot,\cdot)\dots h_M(\cdot,\cdot)$

$$p(e_1^I|f_1^J) = \frac{\exp(\sum_{m=1}^{M} \lambda_m h_m(e_1^I, f_1^J))}{\sum_{{e'}_1^{I'}} \exp(\sum_{m=1}^{M} \lambda_m h_m({e'}_1^{I'}, f_1^J))} \quad (9)$$

- Each feature function $h_m(e, f)$ relates source f to target e.
  E.g. the feature for n-gram language model:

$$h_{\mathrm{LM}}(f_1^J, e_1^I) = \log \prod_{i=1}^{I} p(e_i|e_{i-n+1}^{i-1}) \quad (10)$$

- Model weights λ_1^M specify the relative importance of features.

Log-Linear Model (2)

As before, the constant denominator not needed in maximization:

$$\begin{aligned}\hat{e}_1^{\hat{I}} &= \text{argmax}_{I,e_1^I} \frac{\exp(\sum_{m=1}^{M} \lambda_m h_m(e_1^I, f_1^J))}{\sum_{e'^{I'}_1} \exp(\sum_{m=1}^{M} \lambda_m h_m(e'^{I'}_1, f_1^J))} \\ &= \text{argmax}_{I,e_1^I} \exp(\sum_{m=1}^{M} \lambda_m h_m(e_1^I, f_1^J))\end{aligned} \quad (11)$$

Relation to Noisy Channel

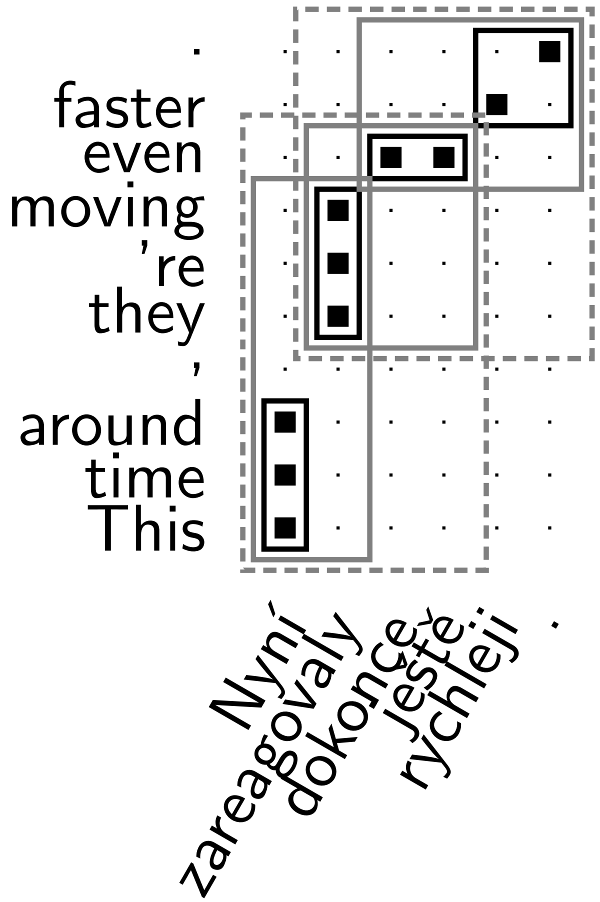
With equal weights and only two features:

- $h_{\mathsf{TM}}(e_1^I, f_1^J) = \log p(f_1^J | e_1^I)$ for the translation model,
- $h_{\mathsf{LM}}(e_1^I, f_1^J) = \log p(e_1^I)$ for the language model,

log-linear model reduces to Noisy Channel:

$$
\begin{aligned}
\hat{e}_1^{\hat{I}} &= \text{argmax}_{I, e_1^I} \exp(\textstyle\sum_{m=1}^{M} \lambda_m h_m(e_1^I, f_1^J)) \\
&= \text{argmax}_{I, e_1^I} \exp(h_{\mathsf{TM}}(e_1^I, f_1^J) + h_{\mathsf{LM}}(e_1^I, f_1^J)) \\
&= \text{argmax}_{I, e_1^I} \exp(\log p(f_1^J | e_1^I) + \log p(e_1^I)) \\
&= \text{argmax}_{I, e_1^I} p(f_1^J | e_1^I) p(e_1^I)
\end{aligned} \tag{12}
$$

Phrase-Based MT Overview



This time around	=	Nyní
they 're moving	=	zareagovaly
even	=	dokonce ještě
. . .	=	. . .
This time around, they 're moving	=	Nyní zareagovaly
even faster	=	dokonce ještě rychleji
. . .	=	. . .

Phrase-based MT: choose such segmentation of input string and such phrase "replacements" to make the output sequence "coherent" (3-grams most probable).

Phrase-Based Translation Model

- Captures the basic assumption of phrase-based MT:
  1. Segment source sentence f_1^J into K phrases $\tilde{f}_1 \dots \tilde{f}_K$.
  2. Translate each phrase independently: $\tilde{f}_k \to \tilde{e}_k$.
  3. Concatenate translated phrases (with possible reordering R): $\tilde{e}_{R(1)} \dots \tilde{e}_{R(K)}$
- In theory, the segmentation s_1^K is a hidden variable in the maximization, we should be summing over all segmentations: (Note the three args in $h_m(\cdot,\cdot,\cdot)$ now.)

$$\hat{e}_1^{\hat{I}} = \operatorname{argmax}_{I,e_1^I} \sum_{s_1^K} \exp(\sum_{m=1}^{M} \lambda_m h_m(e_1^I, f_1^J, s_1^K)) \tag{13}$$

- In practice, the sum is approximated with a max (the biggest element only):

$$\hat{e}_1^{\hat{I}} = \operatorname{argmax}_{I,e_1^I} \max_{s_1^K} \exp(\sum_{m=1}^{M} \lambda_m h_m(e_1^I, f_1^J, s_1^K)) \tag{14}$$

Core Feature: Phrase Trans. Prob.

The most important feature: phrase-to-phrase translation:

$$h_{\mathrm{Phr}}(f_1^J, e_1^I, s_1^K) = \log \prod_{k=1}^{K} p(\tilde{f}_k | \tilde{e}_k) \tag{15}$$

The conditional probability of phrase $\tilde{f}_k$ given phrase $\tilde{e}_k$ is estimated from relative frequencies:

$$p(\tilde{f}_k | \tilde{e}_k) = \frac{\mathrm{count}(\tilde{f}, \tilde{e})}{\mathrm{count}(\tilde{e})} \tag{16}$$

- $\mathrm{count}(\tilde{f}, \tilde{e})$ is the number of co-occurrences of a phrase pair $(\tilde{f}, \tilde{e})$ that are consistent with the word alignment
- $\mathrm{count}(\tilde{e})$ is the number of occurrences of the target phrase $\tilde{e}$ in the training corpus.
- h_{Phr} usually used twice, in both directions: $p(\tilde{f}_k | \tilde{e}_k)$ and $p(\tilde{e}_k | \tilde{f}_k)$

Phrase-Based Features in Moses

Given parallel training corpus, phrases are extracted and scored:

```
in europa ||| in europe ||| 0.829007 0.207955 0.801493 0.492402
europas ||| in europe ||| 0.0251019 0.066211 0.0342506 0.0079563
in der europaeischen union ||| in europe ||| 0.018451 0.00100126 0.0319584 0
```

The scores are: $(\phi(\cdot) = \log p(\cdot))$

- phrase translation probabilities: $\phi_{\text{phr}}(f|e)$ and $\phi_{\text{phr}}(e|f)$
- lexical weighting: $\phi_{\text{lex}}(f|e)$ and $\phi_{\text{lex}}(e|f)$ (Koehn, 2003)

$$\phi_{\text{lex}}(f|e) = \log \max_{\substack{a \in \text{alignments} \\ \text{of } (f,e)}} \prod_{i=1}^{|f|} \frac{1}{|\{j|(i,j) \in a|} \sum_{\forall (i,j) \in a} p(f_i|e_j) \tag{17}$$

Other Features Used in PBMT

- Word count/penalty: $h_{\text{wp}}(e_1^I, \cdot, \cdot) = I$

  ⇒ Do we prefer longer or shorter output?
- Phrase count/penalty: $h_{\text{pp}}(\cdot, \cdot, s_1^K) = K$

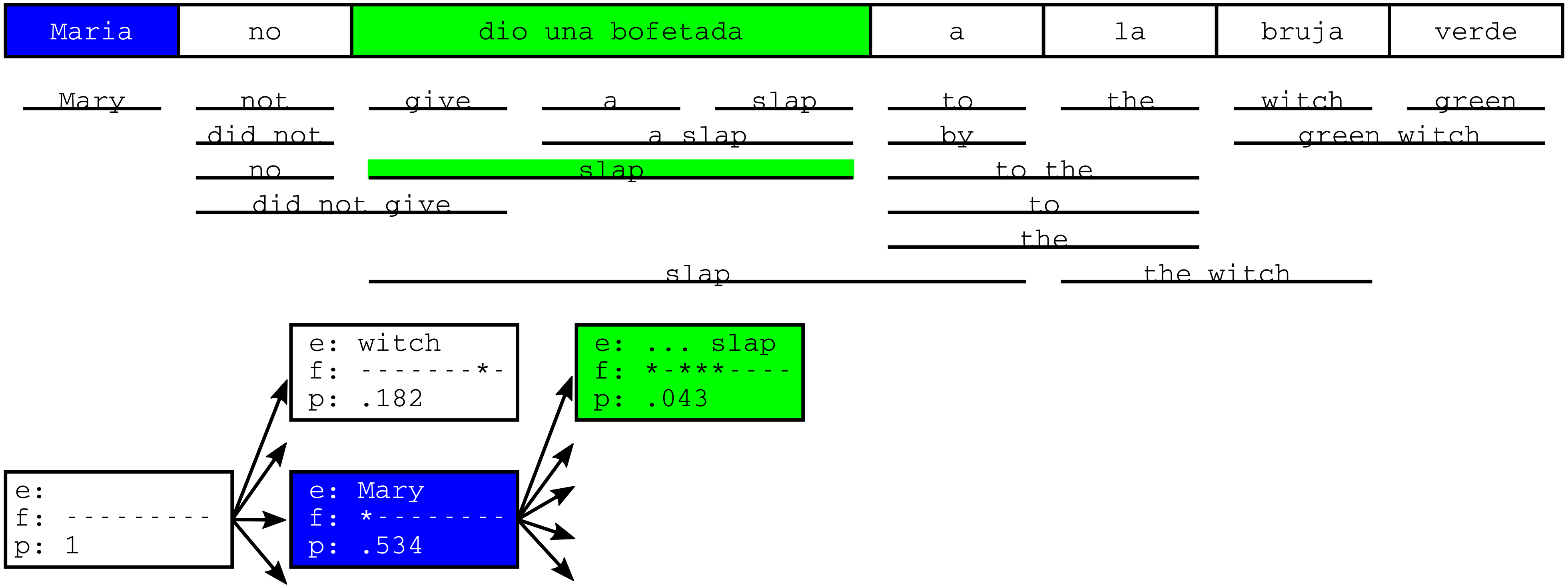
  ⇒ Do we prefer translation in more or fewer less-dependent bits?
- Reordering model: different basic strategies (Lopez, 2009)

  ⇒ Which source spans can provide continuation at a moment?
- n-gram LM:

$$h_{\text{LM}}(\cdot, e_1^I, \cdot) = \log \prod_{i=1}^{I} p(e_i | e_{i-n+1}^{i-1}) \tag{18}$$

⇒ Is output n-gram-wise coherent?

Decoding in Phrase-Based MT



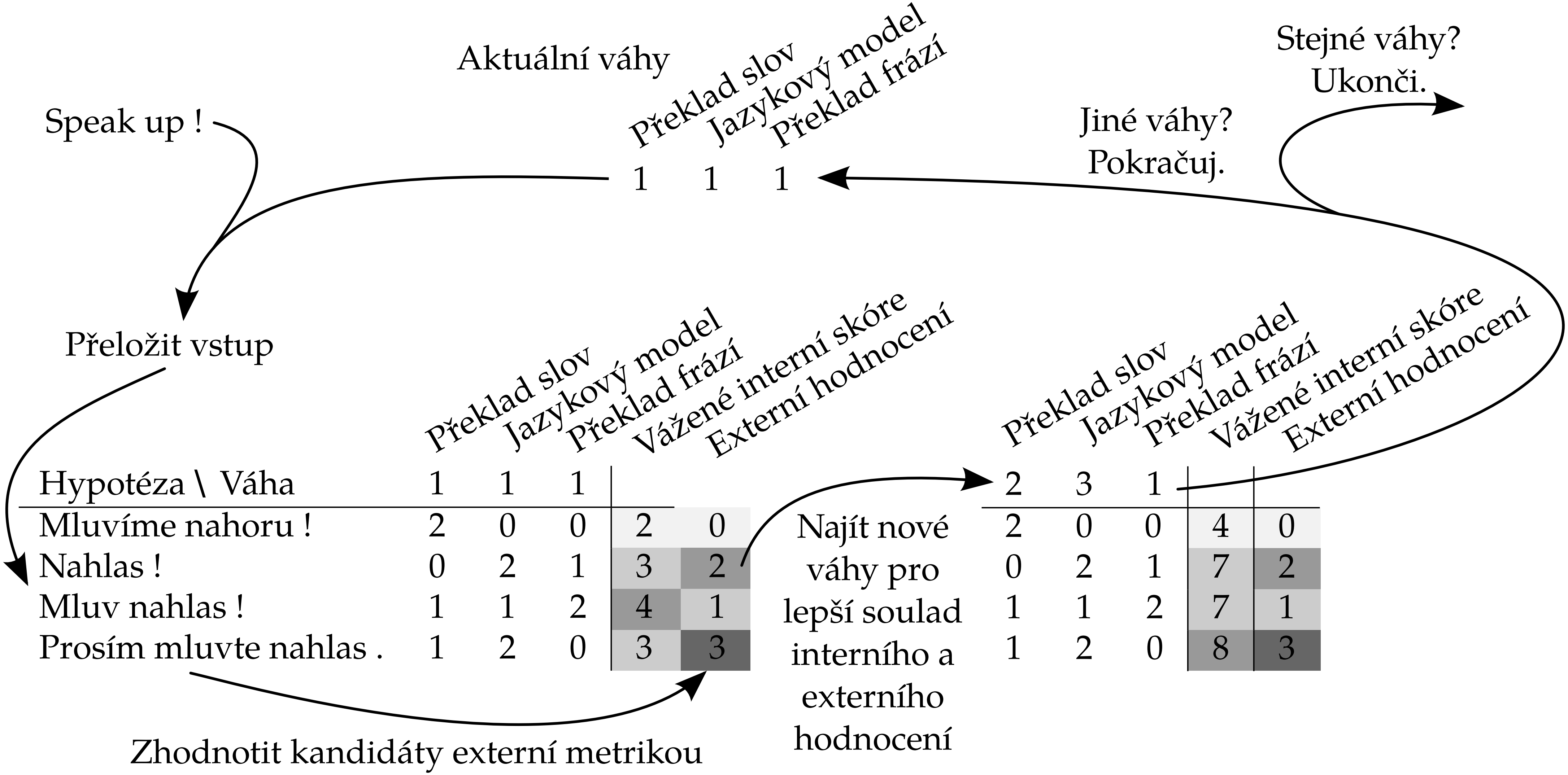
1. Collect **translation options** (all possible translations per span).
2. Gradually **expand partial hypotheses** until all input covered.
3. **Prune** less promising hypotheses.
4. When all input covered, trace back the best path.

Local and Non-Local Features

							Total	Weight	Weighted
Phrase log. prob.		0,0	-0,69	-1,39			-2,08	2,0	-4,16
Phrase penalty		1,0	1,0	1,0			3,0	-1,0	-3,0
Word penalty		1,0	2,0	1,0			4,0	-0,5	-2,0
		Peter	left for	home	.				
	▷	Petr	odešel	domů	.	◁			
Bigram log. prob.		-4,02	-2,50	-3,61	-0,39	-0,08	-10,59	1,0	-10,59
								Total	-19,75

- Local features decompose along hypothesis construction.
  - Phrase- and word-based features.
- Non-local features span the boundaries (e.g. LM).

Weight Optimization: MERT Loop

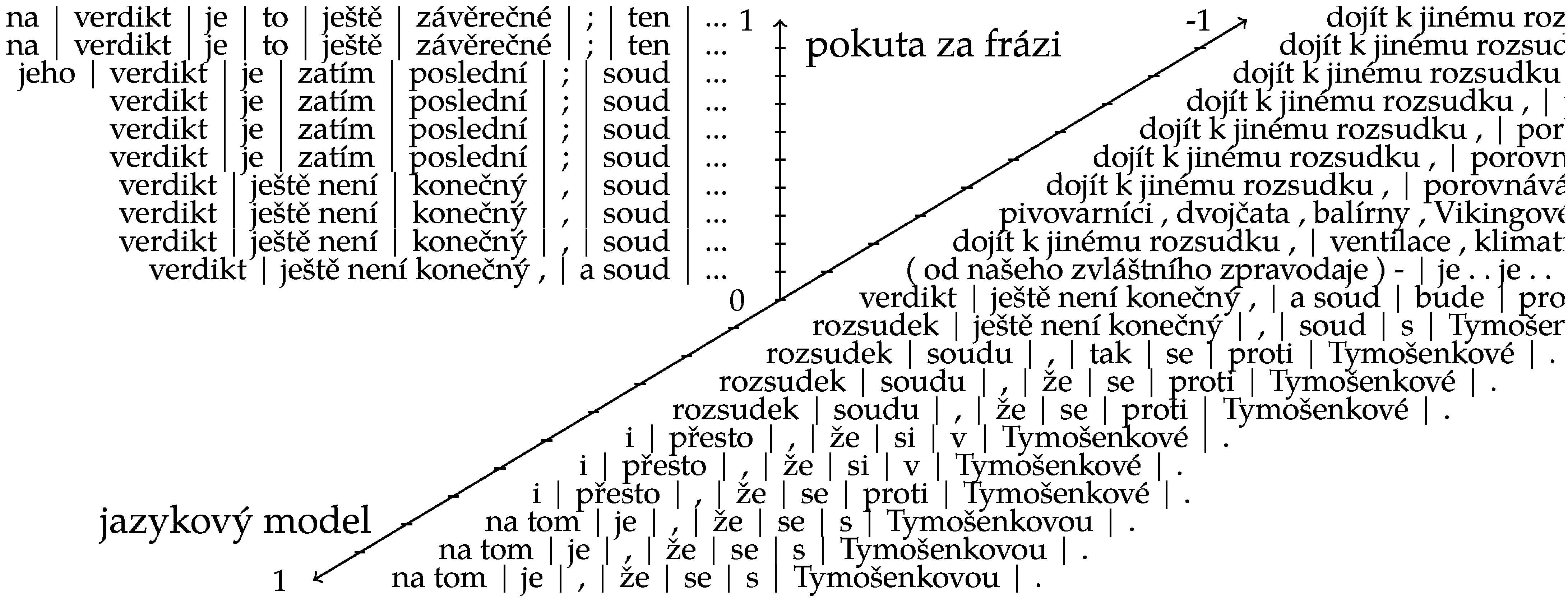


Hypotéza \ Váha	Překlad slov	Jazykový model	Překlad frází	Vážené interní skóre	Externí hodnocení
	1	1	1		
Mluvíme nahoru !	2	0	0	2	0
Nahlas !	0	2	1	3	2
Mluv nahlas !	1	1	2	4	1
Prosím mluvte nahlas .	1	2	0	3	3

Překlad slov	Jazykový model	Překlad frází	Vážené interní skóre	Externí hodnocení
2	3	1		
2	0	0	4	0
0	2	1	7	2
1	1	2	7	1
1	2	0	8	3

Minimum Error Rate Training (Och, 2003)

Effects of Weights



- Higher phrase penalty chops sentence into more segments.
- Too strong LM weight leads to words dropped.
- Negative LM weight leads to obscure wordings.

Summary of PBMT

Phrase-based MT:

- is a log-linear model
- assumes phrases relatively independent of each other
- decomposes sentence into contiguous phrases
- search has two parts:
  - lookup of all relevant translation options
  - stack-based beam search, gradually expanding hypotheses

To train a PBMT system:

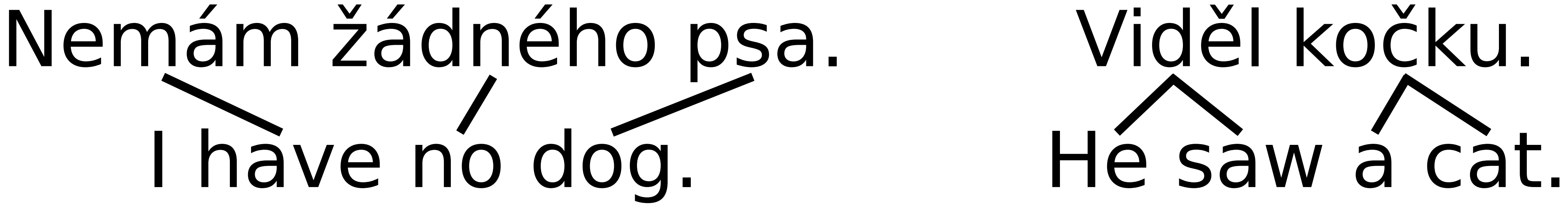
1. Align words.
2. Extract (and score) phrases consistent with word alignment.
3. Optimize weights (MERT).

1: Align Training Sentences

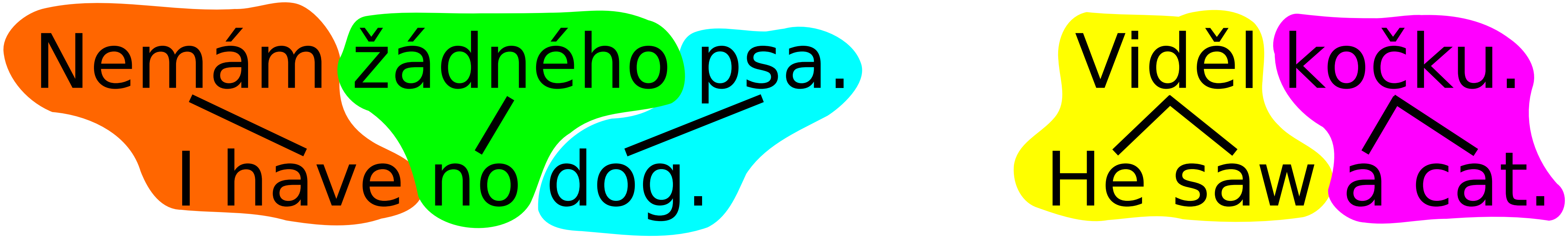
Nemám žádného psa. Viděl kočku.

I have no dog. He saw a cat.

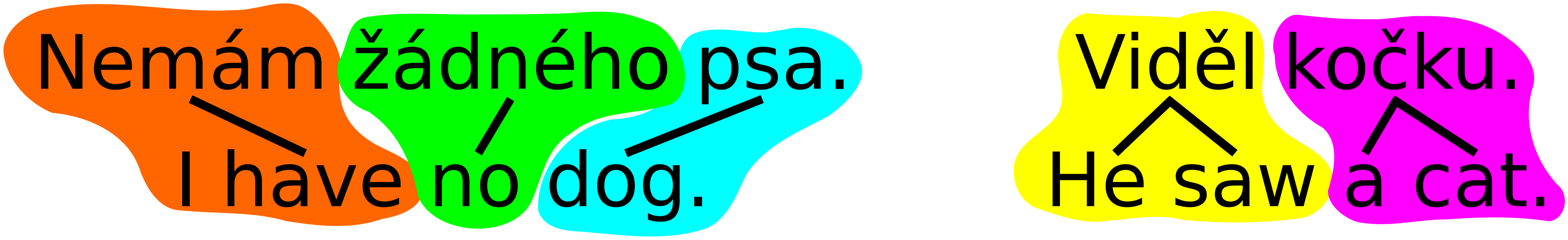
2: Align Words



3: Extract Phrase Pairs (MTUs)

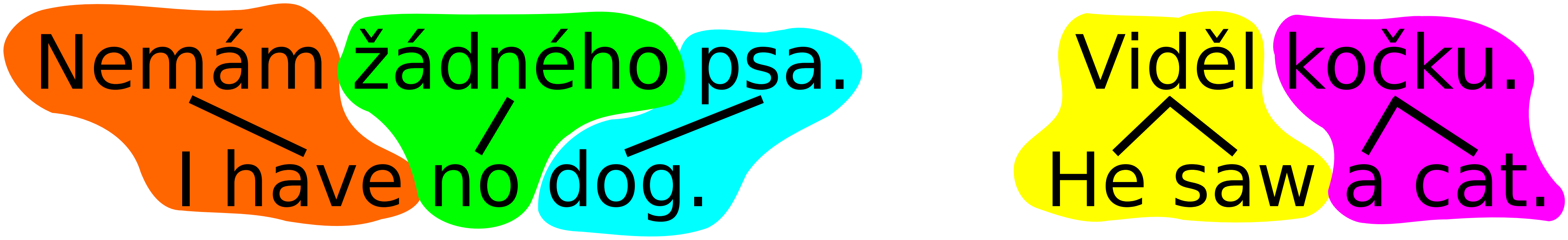


4: New Input



New input: Nemám kočku.

4: New Input



... I don't have cat.

New input: Nemám kočku.

5: Pick Probable Phrase Pairs (TM)

6: So That n-Grams Probable (LM)

Meaning Got Reversed!

What Went Wrong?

$$\hat{e}_1^{\hat{I}} = \operatorname*{argmax}_{I,e_1^I} p(f_1^J|e_1^I)p(e_1^I) = \operatorname*{argmax}_{I,e_1^I} \prod_{(\hat{f},\hat{e})\in\text{phrase pairs of } f_1^J,e_1^I} p(\hat{f}|\hat{e})p(e_1^I) \quad (19)$$

- Too strong phrase-independence assumption.
  - Phrases do depend on each other.
    Here "nemám" and "žádného" jointly express one negation.
  - Word alignments ignored that dependence.
    But adding it would increase data sparseness.
- Language model is a separate unit.
  - $p(e_1^I)$ models the target sentence independently of f_1^J.

Redefining $p(e_1^I|f_1^J)$

What if we modelled $p(e_1^I|f_1^J)$ directly, word by word:

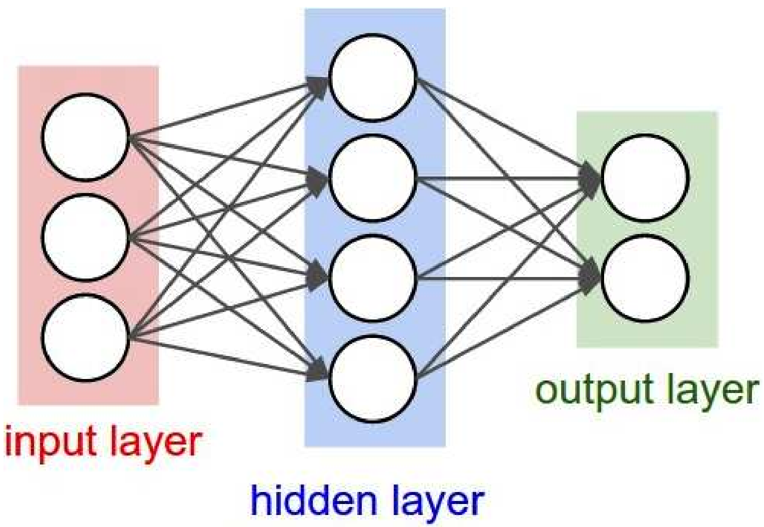
$$
\begin{aligned}
p(e_1^I|f_1^J) &= p(e_1, e_2, \ldots e_I|f_1^J) \\
&= p(e_1|f_1^J) \cdot p(e_2|e_1, f_1^J) \cdot p(e_3|e_2, e_1, f_1^J) \ldots \\
&= \prod_{i=1}^{I} p(e_i|e_1, \ldots e_{i-1}, f_1^J)
\end{aligned} \tag{20}
$$

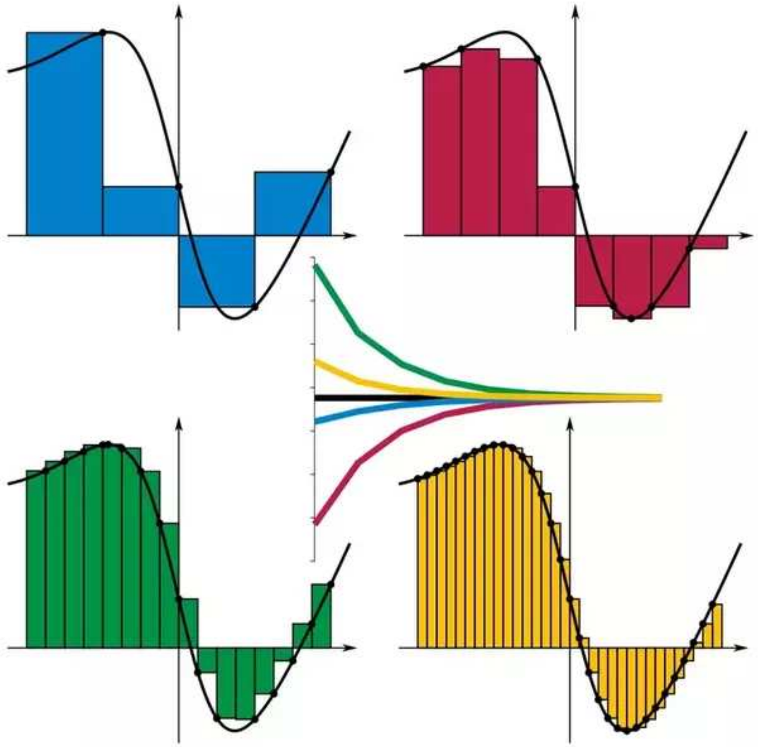
. . . this is "just a cleverer language model:" $p(e_1^I) = \prod_{i=1}^{I} p(e_i|e_1, \ldots e_{i-1})$

Main Benefit: All dependencies available.

But what technical device can learn this?

NNs: Universal Approximators

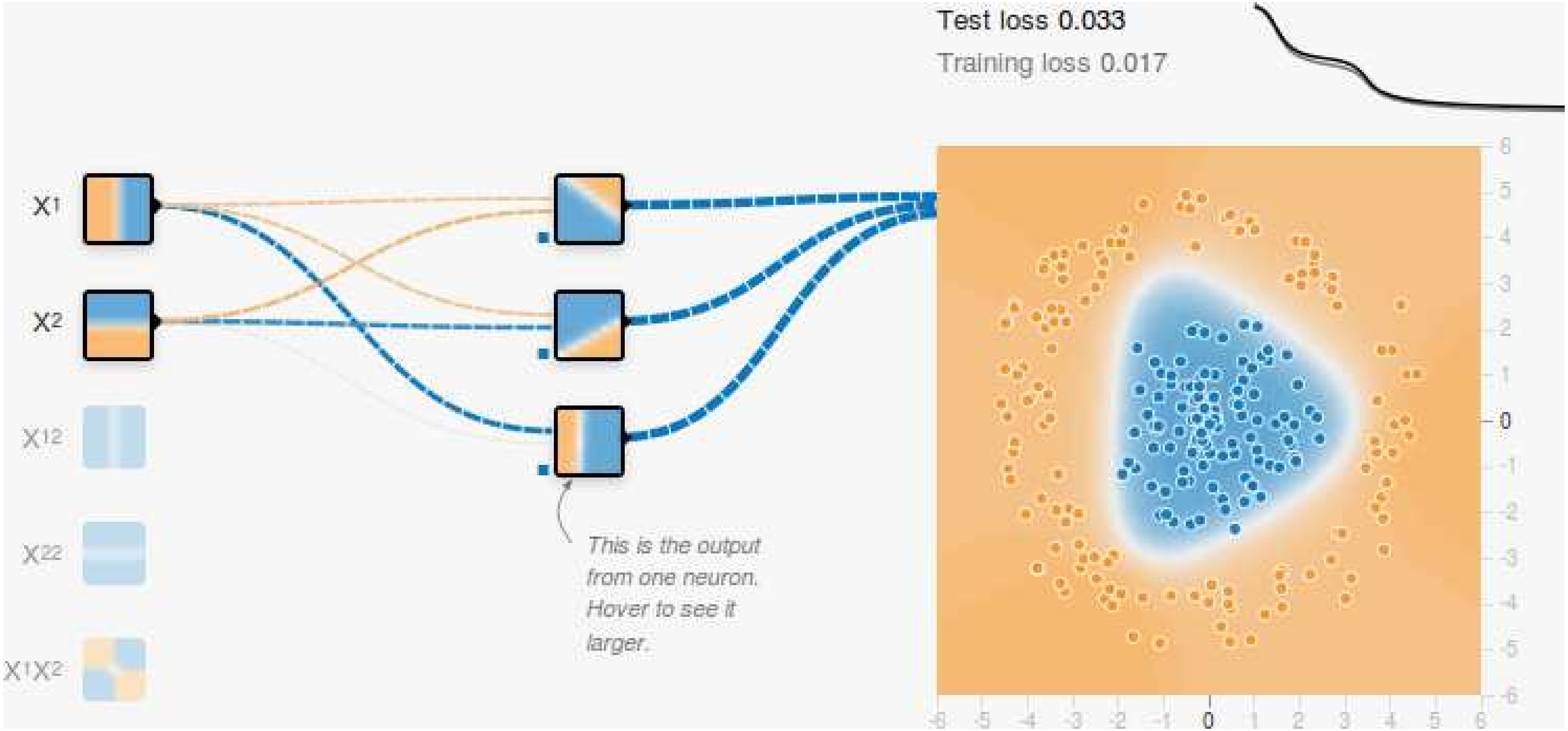




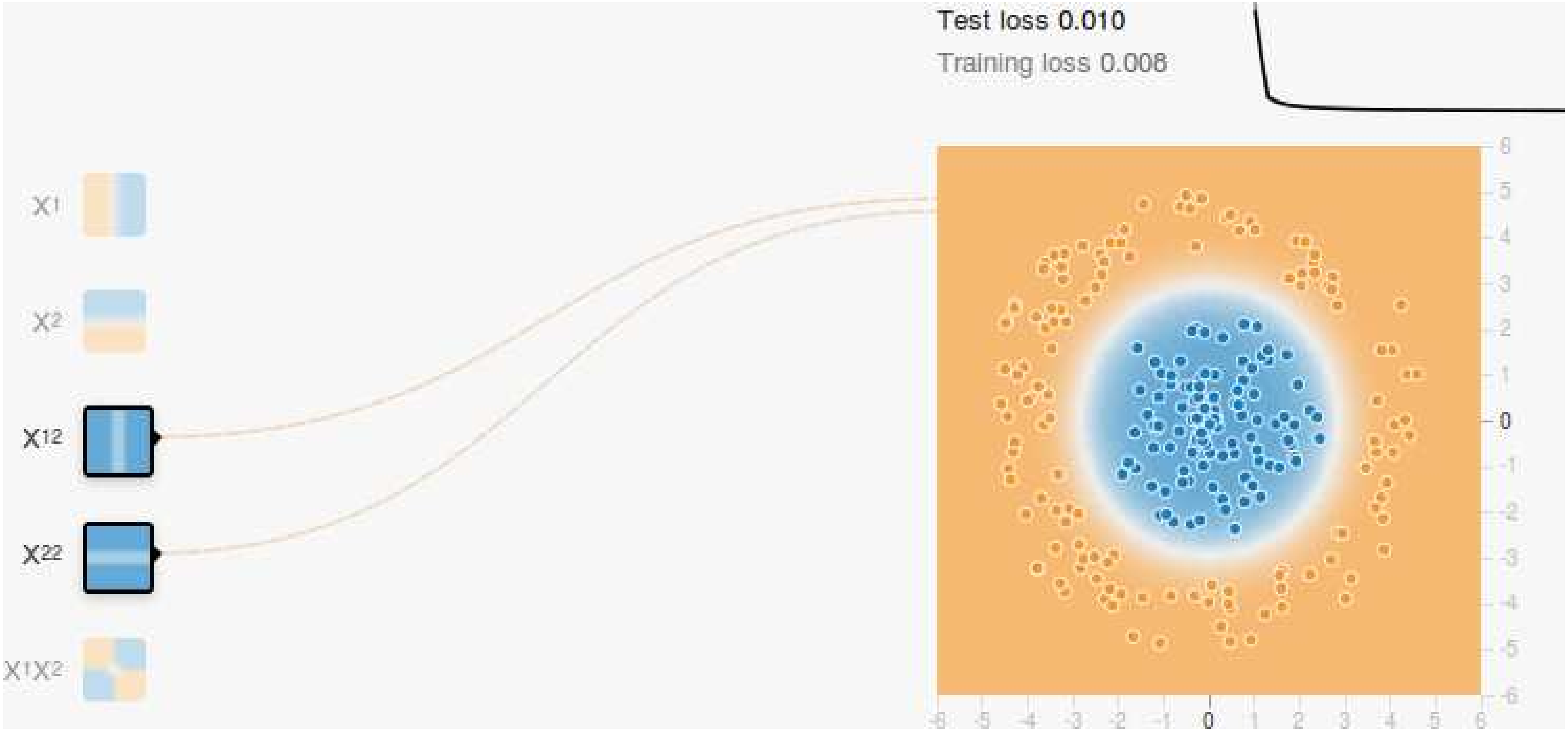
- A neural network with a single hidden layer (possibly huge) can approximate any continuous function to any precision.
- (Nothing claimed about learnability.)

https://www.quora.com/How-can-a-deep-neural-network-with-ReLU-activations-in-its-hidden-layers-approximate-an

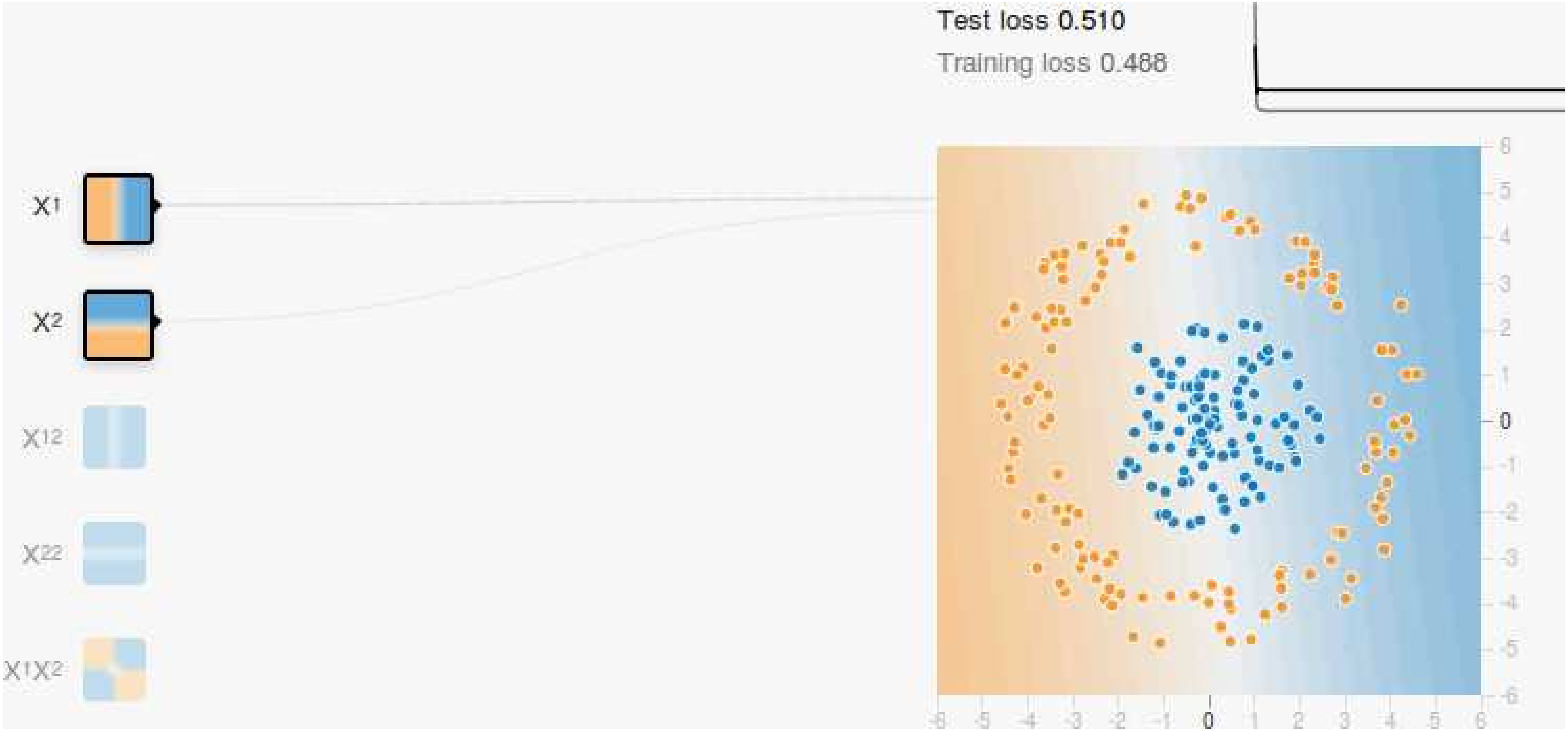
playground.tensorflow.org



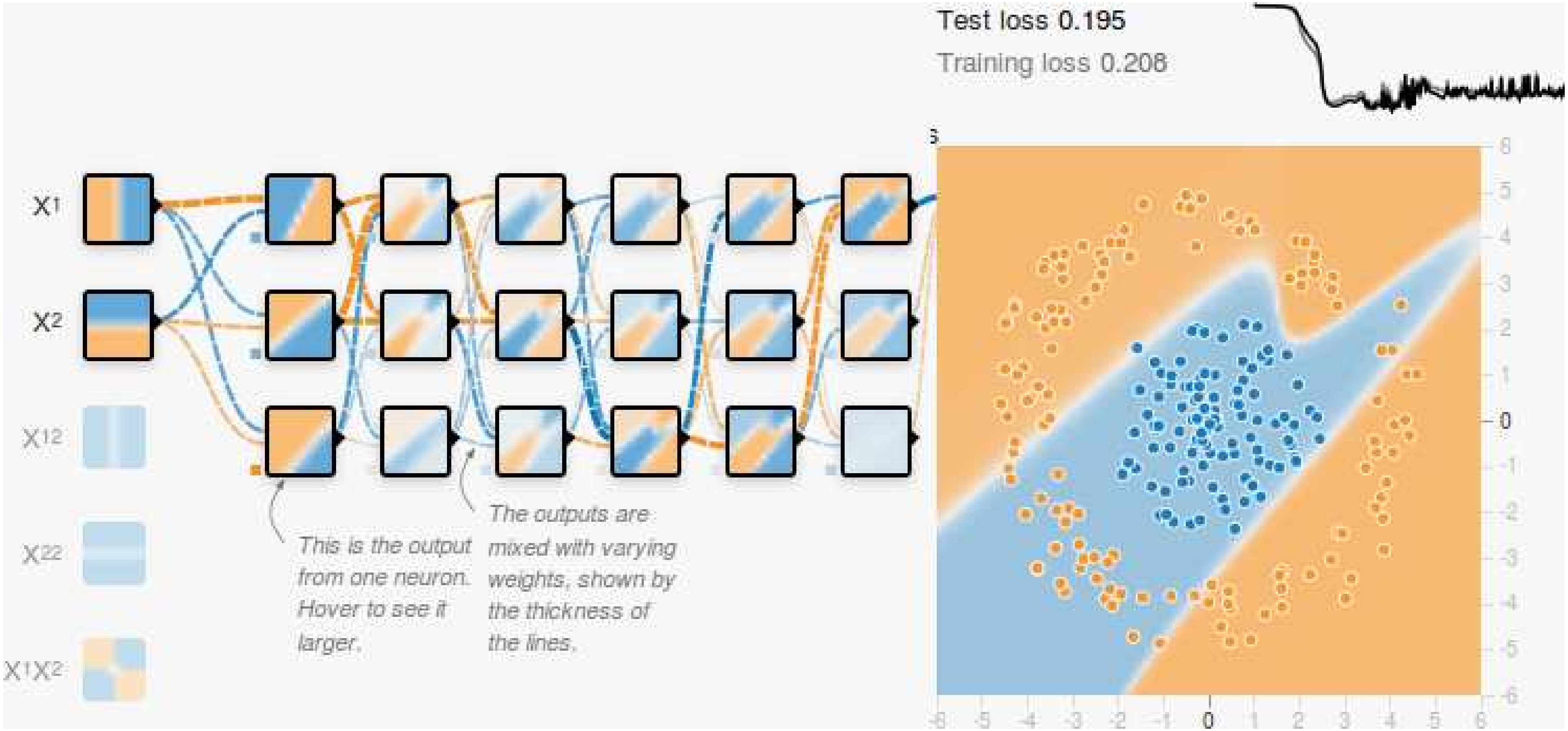
Perfect Features



Bad Features & Low Depth

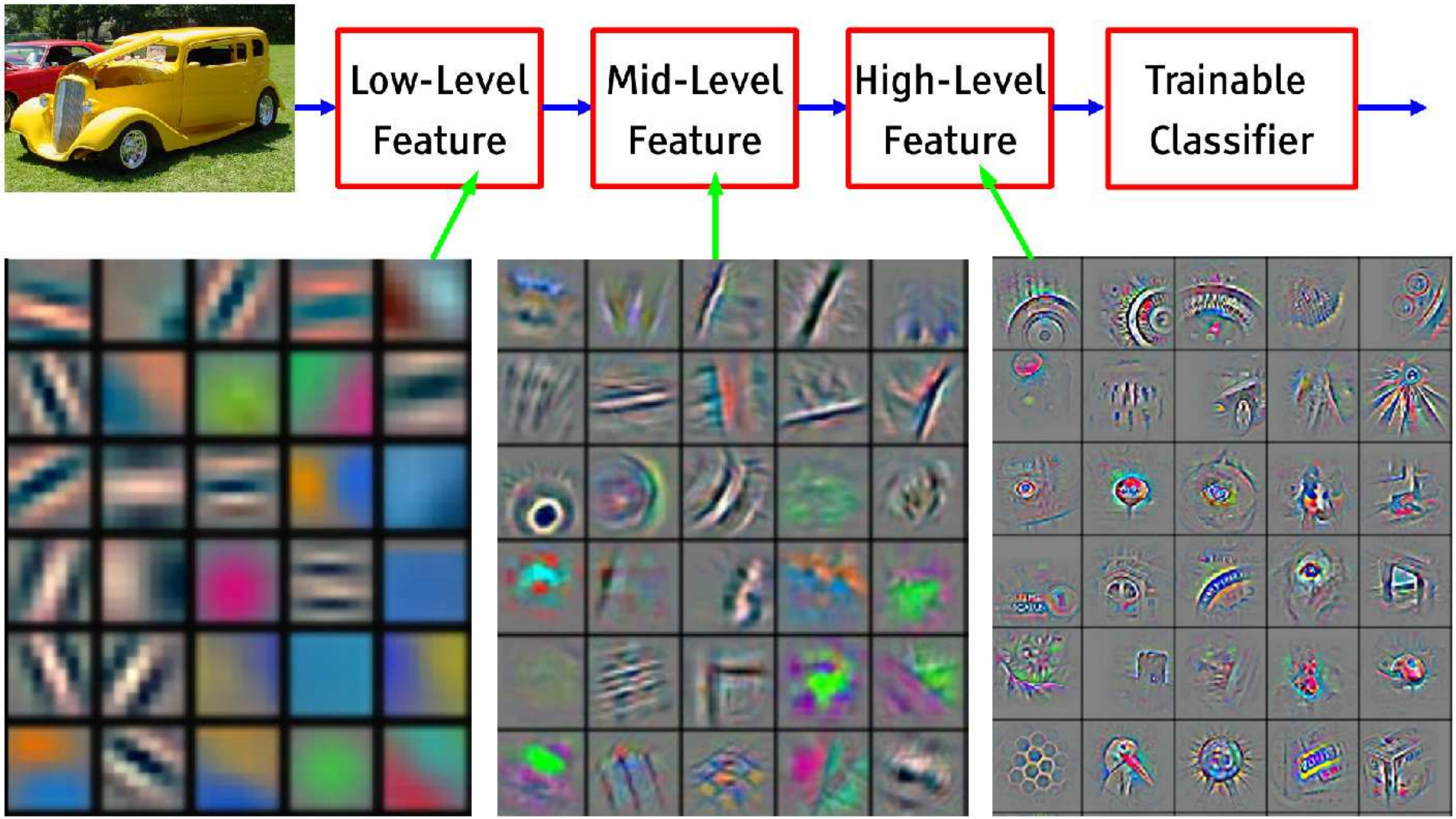


Too Complex NN Fails to Learn



Deep NNs for Image Classification

- It's deep if it has more than one stage of non-linear feature transformation



Feature visualization of convolutional net trained on ImageNet from [Zeiler & Fergus 2013]

Representation Learning

- Based on training data (sample inputs and expected outputs)
- the neural network learns by itself
- what is important in the inputs
- to predict the outputs best.

A "**representation**" is a new set of axes.

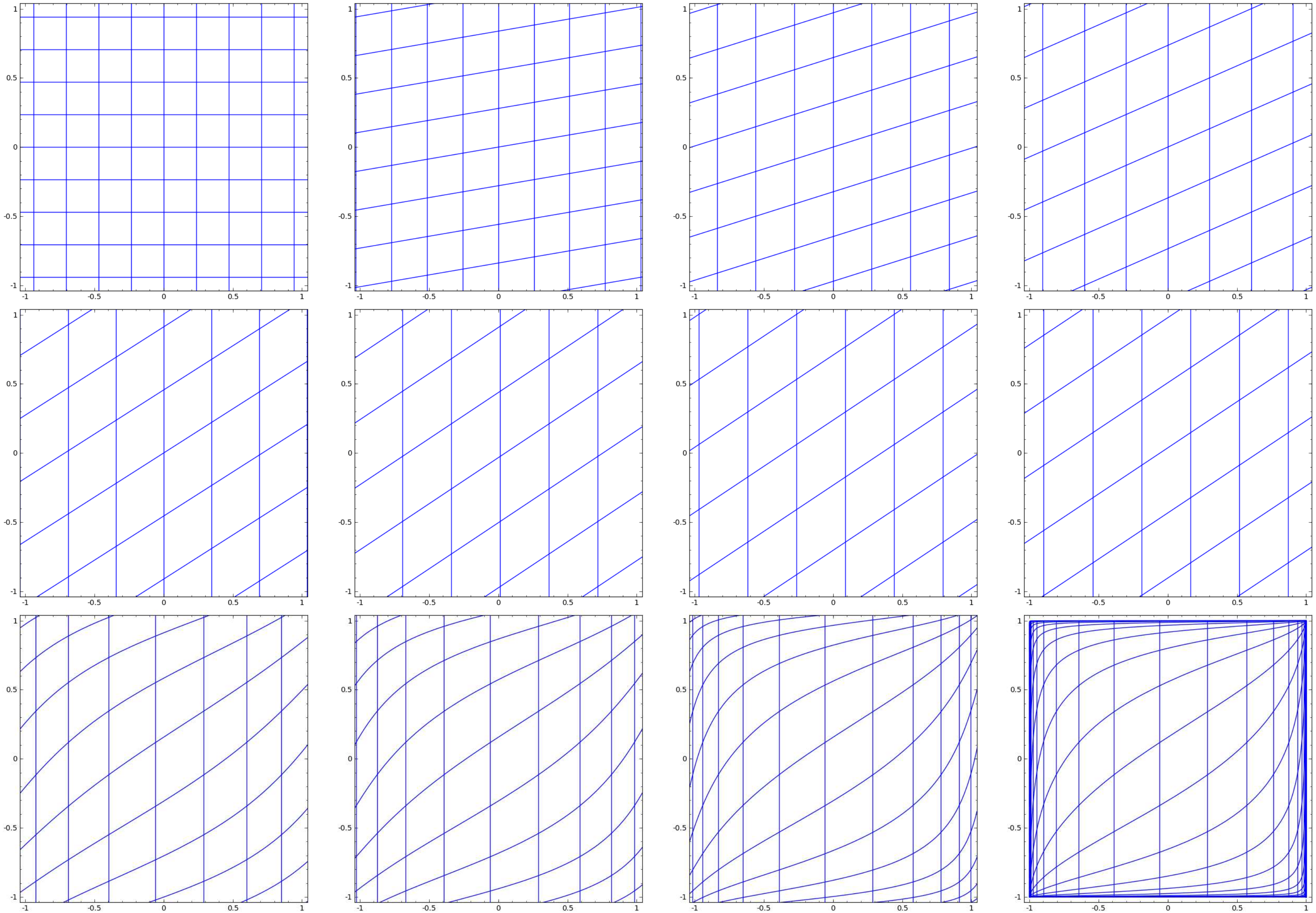
- Instead of 3 dimensions (x, y, color), we get
- 2000 dimensions: (elephantity, number of storks, blueness, . . .)
- designed automatically to help in best prediction of the output

One Layer $\tanh(Wx + b)$, 2D→2D

Skew: W

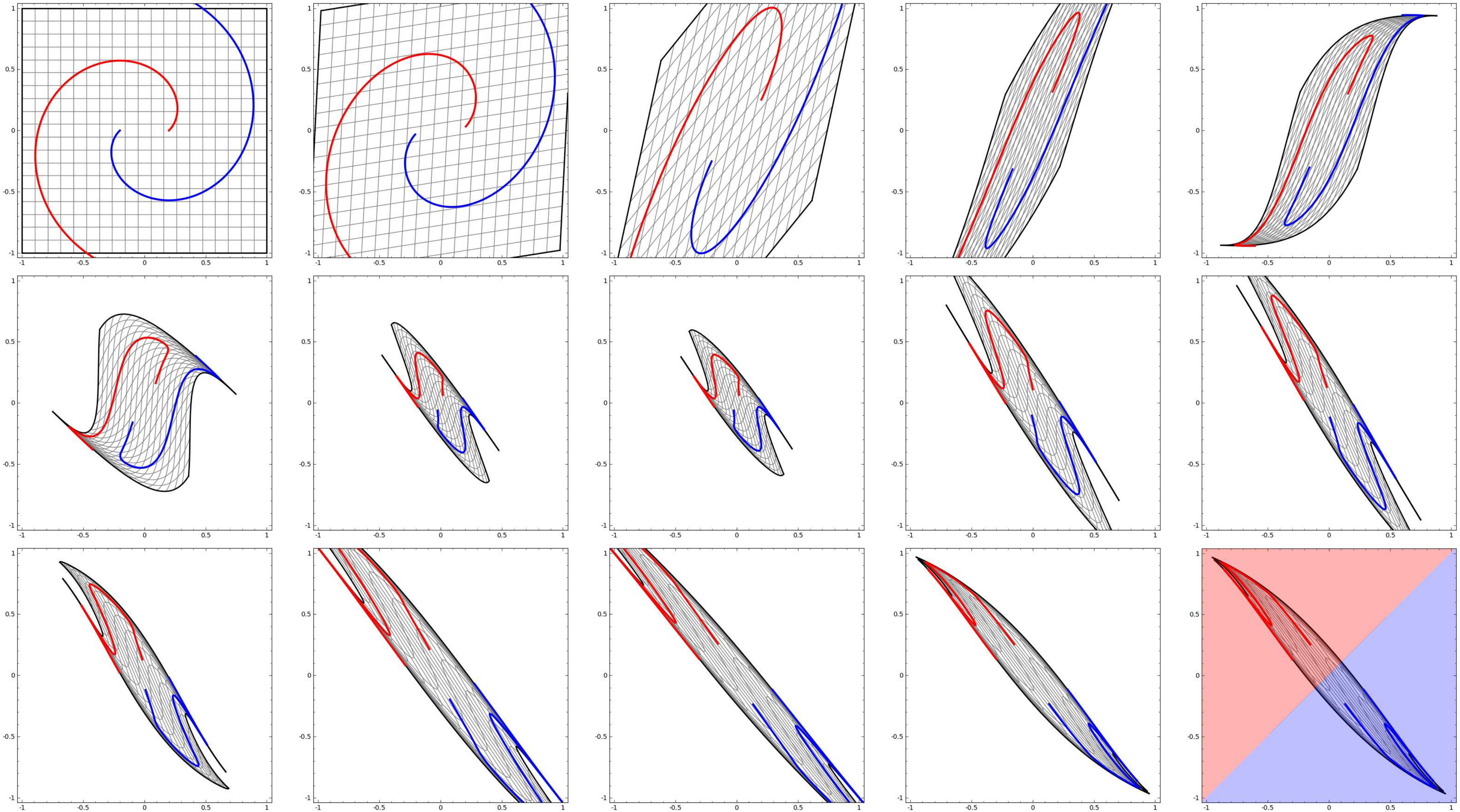
Transpose: b

Non-lin.: $\tanh$



Animation by `http://colah.github.io/posts/2014-03-NN-Manifolds-Topology/`

Four Layers, Disentagling Spirals



Animation by `http://colah.github.io/posts/2014-03-NN-Manifolds-Topology/`

Processing Text with NNs

- Map each word to a vector of 0s and 1s ("1-hot repr."):

$$\text{cat} \mapsto (0, 0, \ldots, 0, 1, 0, \ldots, 0)$$

- Sentence is then a matrix:

		the	cat	is	on	the	mat
↑	a	0	0	0	0	0	0
	about	0	0	0	0	0	0
	...	...	...	...	...	...	...
	cat	0	**1**	0	0	0	0
	...	...	...	...	...	...	...
	is	0	0	**1**	0	0	0
	...	...	...	...	...	...	...
	on	0	0	0	**1**	0	0
	...	...	...	...	...	...	...
	the	**1**	0	0	0	**1**	0
	...	...	...	...	...	...	...
↓	zebra	0	0	0	0	0	0

Processing Text with NNs

- Map each word to a vector of 0s and 1s ("1-hot repr."):

$$\text{cat} \mapsto (0, 0, \ldots, 0, 1, 0, \ldots, 0)$$

- Sentence is then a matrix:

		the	cat	is	on	the	mat
↑	a	0	0	0	0	0	0
	about	0	0	0	0	0	0
	...	...	...	...	...	...	...
	cat	0	**1**	0	0	0	0
Vocabulary size:	...	...	...	...	...	...	...
1.3M English	is	0	0	**1**	0	0	0
2.2M Czech	...	...	...	...	...	...	...
	on	0	0	0	**1**	0	0
	...	...	...	...	...	...	...
	the	**1**	0	0	0	**1**	0
	...	...	...	...	...	...	...
↓	zebra	0	0	0	0	0	0

Processing Text with NNs

- Map each word to a vector of 0s and 1s ("1-hot repr."):

$$\text{cat} \mapsto (0, 0, \ldots, 0, 1, 0, \ldots, 0)$$

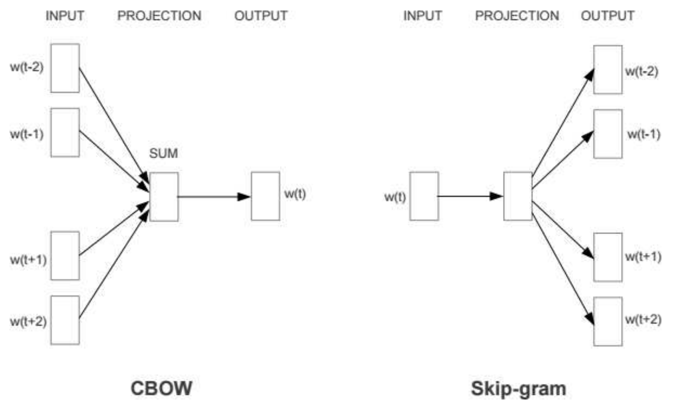
- Sentence is then a matrix:

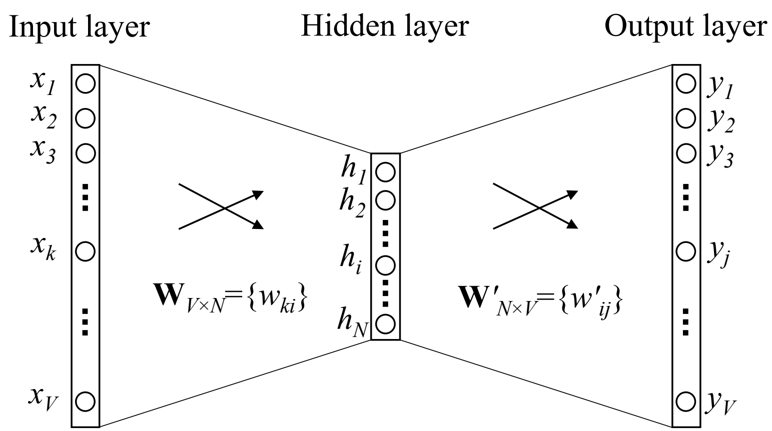
		the	cat	is	on	the	mat
↑	a	0	0	0	0	0	0
	about	0	0	0	0	0	0
	...	...	...	...	...	...	...
	cat	0	**1**	0	0	0	0
Vocabulary size:	...	...	...	...	...	...	...
1.3M English	is	0	0	**1**	0	0	0
2.2M Czech	...	...	...	...	...	...	...
	on	0	0	0	**1**	0	0
	...	...	...	...	...	...	...
	the	**1**	0	0	0	**1**	0
	...	...	...	...	...	...	...
↓	zebra	0	0	0	0	0	0

Main drawback: No relations, all words equally close/far.

Solution: Word Embeddings

- Map each word to a dense vector.
- In practice 300–2000 dimensions are used, not 1–2M.
  - The dimensions have no clear interpretation.
- Embeddings are trained for each particular task.
  - NNs: The matrix that maps 1-hot input to the first layer.
- The famous word2vec (Mikolov et al., 2013):
  - CBOW: Predict the word from its four neighbours.
  - Skip-gram: Predict likely neighbours given the word.



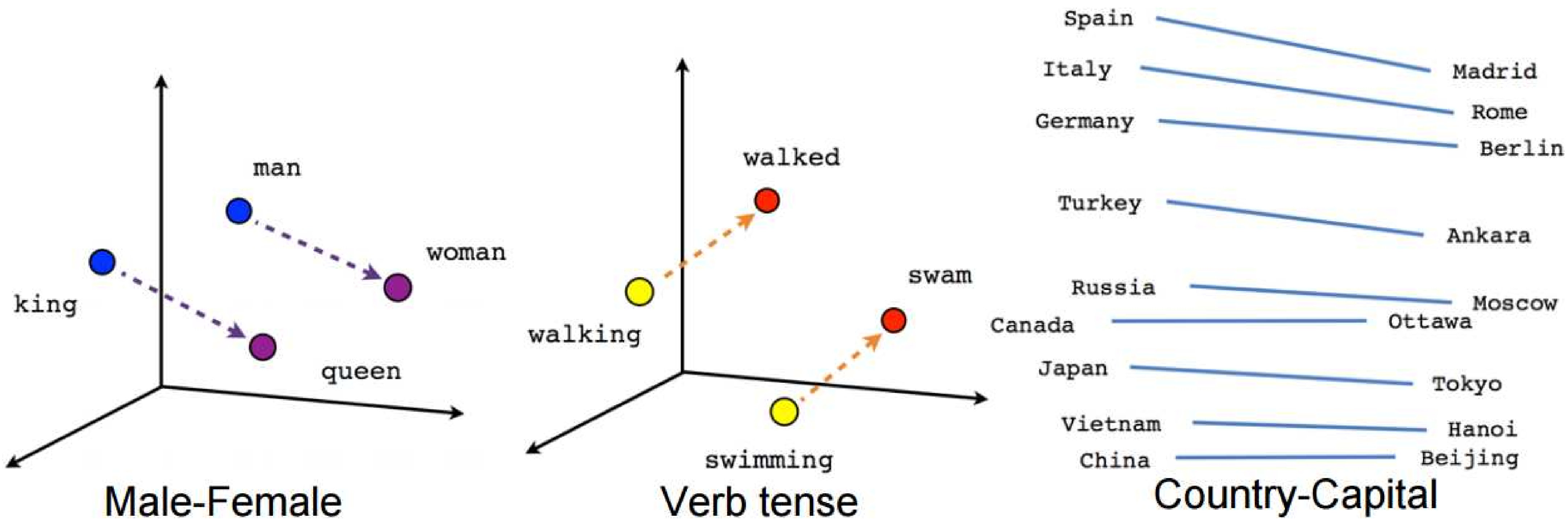


Right: CBOW with just a single-word context (`http://www-personal.umich.edu/~ronxin/pdf/w2vexp.pdf`)

Continuous Space of Words

Word2vec embeddings show interesting properties:

$$v(\text{king}) - v(\text{man}) + v(\text{woman}) \approx v(\text{queen}) \quad (21)$$



Illustrations from `https://www.tensorflow.org/tutorials/word2vec`

Further Compression: Sub-Words

- SMT struggled with productive morphology (>1M wordforms).

  nejneobhodpodařovávatelnějšími, Donaudampfschifffahrtsgesellschaftskapitän
- NMT can handle only 30–80k dictionaries.

⇒ Resort to sub-word units.

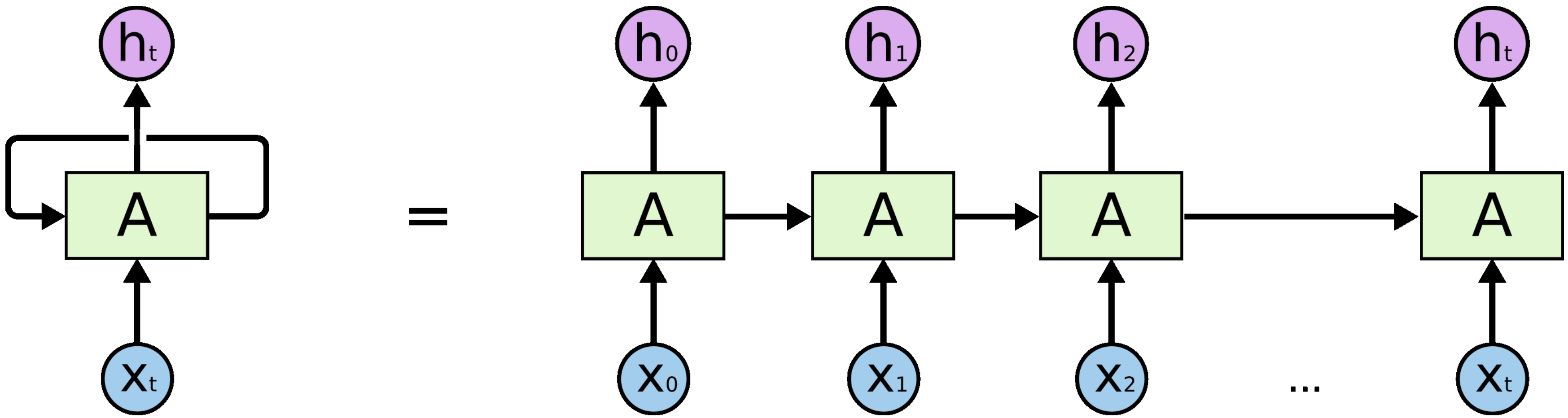
Orig	český politik svezl migranty
Syllables	čes ký ⊔ po li tik ⊔ sve zl ⊔ mig ran ty
Morphemes	česk ý ⊔ politik ⊔ s vez l ⊔ migrant y
Char Pairs	če sk ý ⊔ po li ti k ⊔ sv ez l ⊔ mi gr an ty
Chars	č e s k ý ⊔ p o l i t i k ⊔ s v e z l ⊔ m i g r a n t y
BPE 30k	český politik s@@ vez@@ l mi@@ granty

BPE (Byte-Pair Encoding) uses n most common substrings (incl. frequent words).

Variable-Length Inputs

Variable-length input can be handled by recurrent NNs:

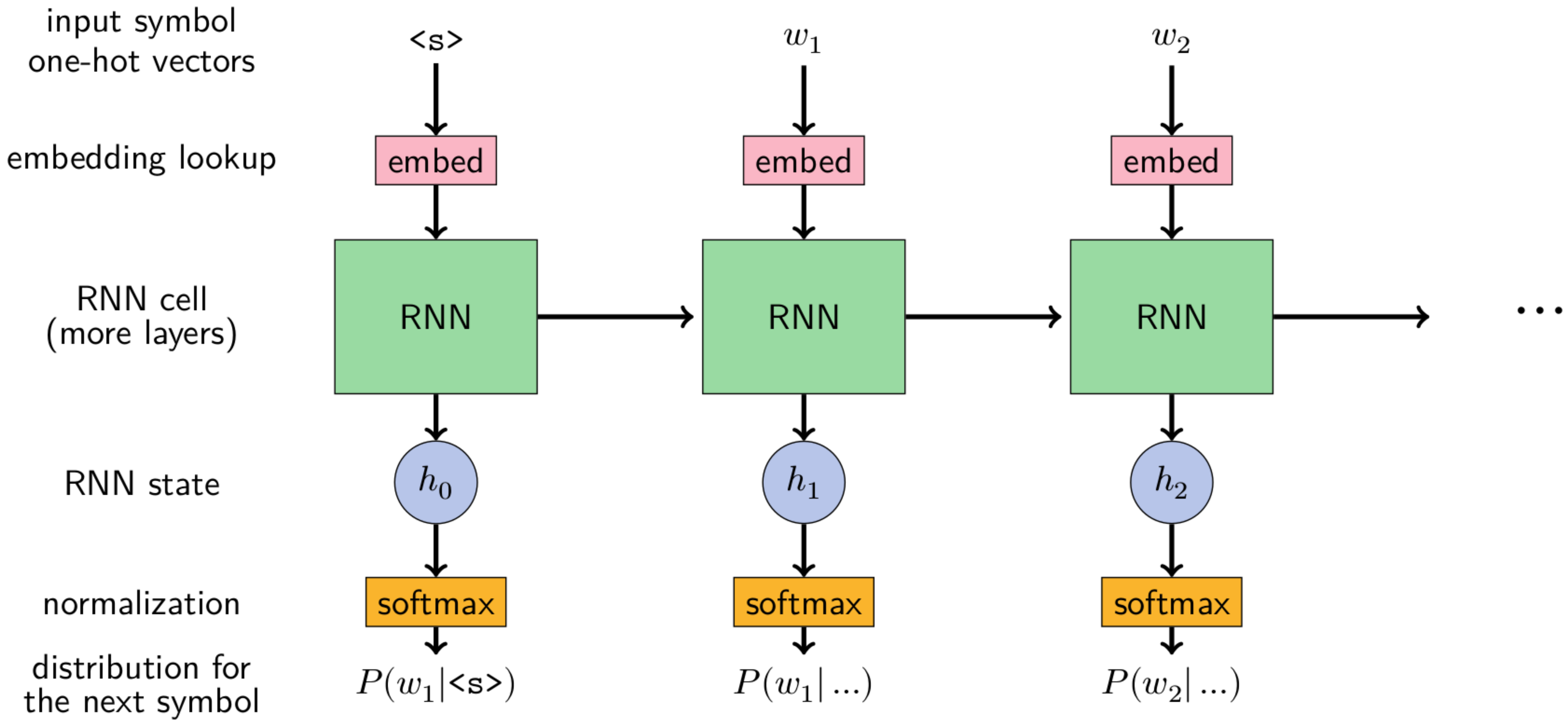
- Reading one input symbol at a time.
  - The same (trained) transformation A used every time.
- Unroll in time (up to a fixed length limit).



Vanilla RNN:

$$h_t = \tanh\left(W[h_{t-1}; x_t] + b\right) \quad (22)$$

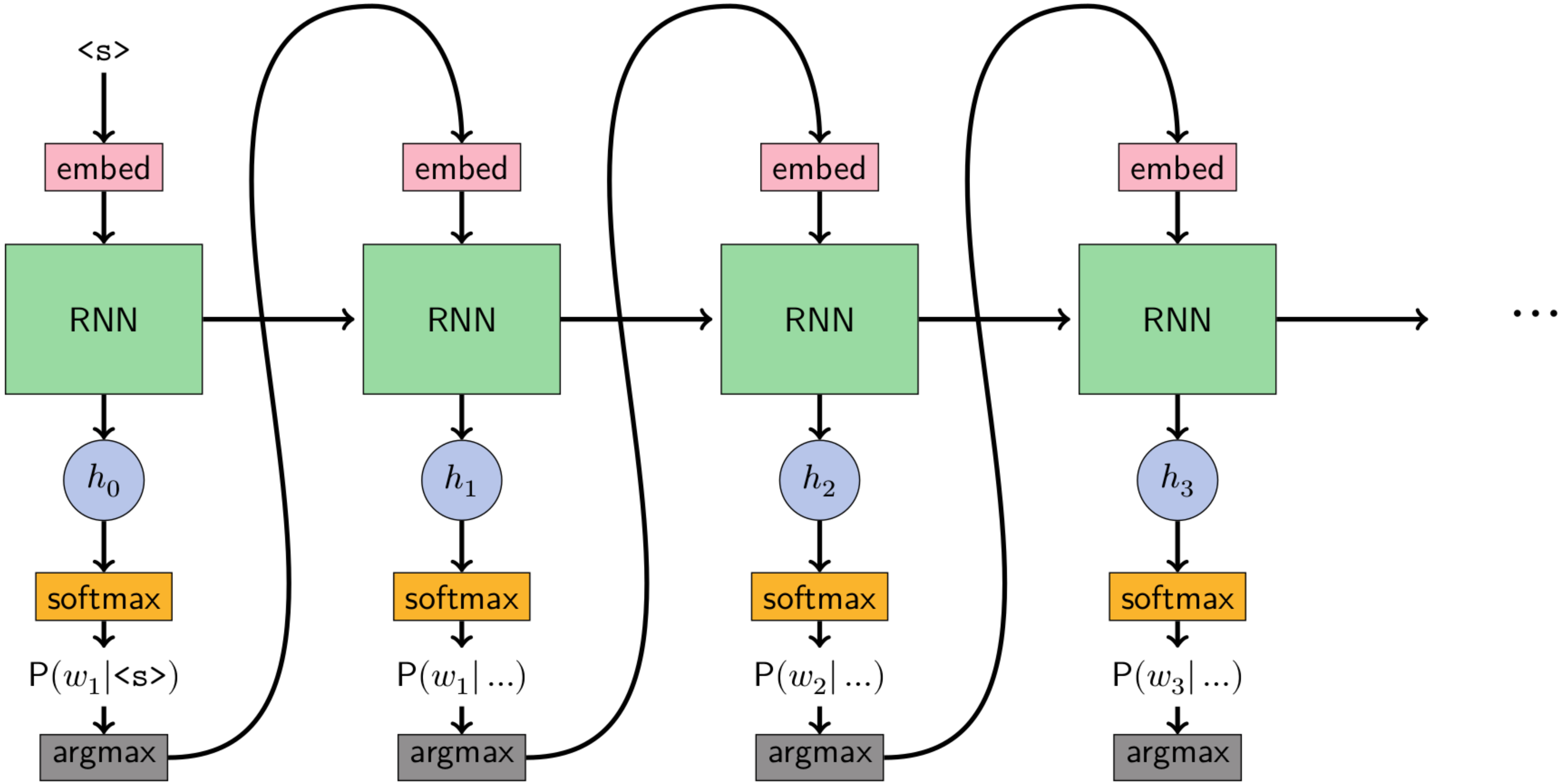
Neural Language Model



- estimate probability of a sentence using the chain rule
- output distributions can be used for sampling

Thanks to Jindřich Libovický for the slides.

Sampling from a LM



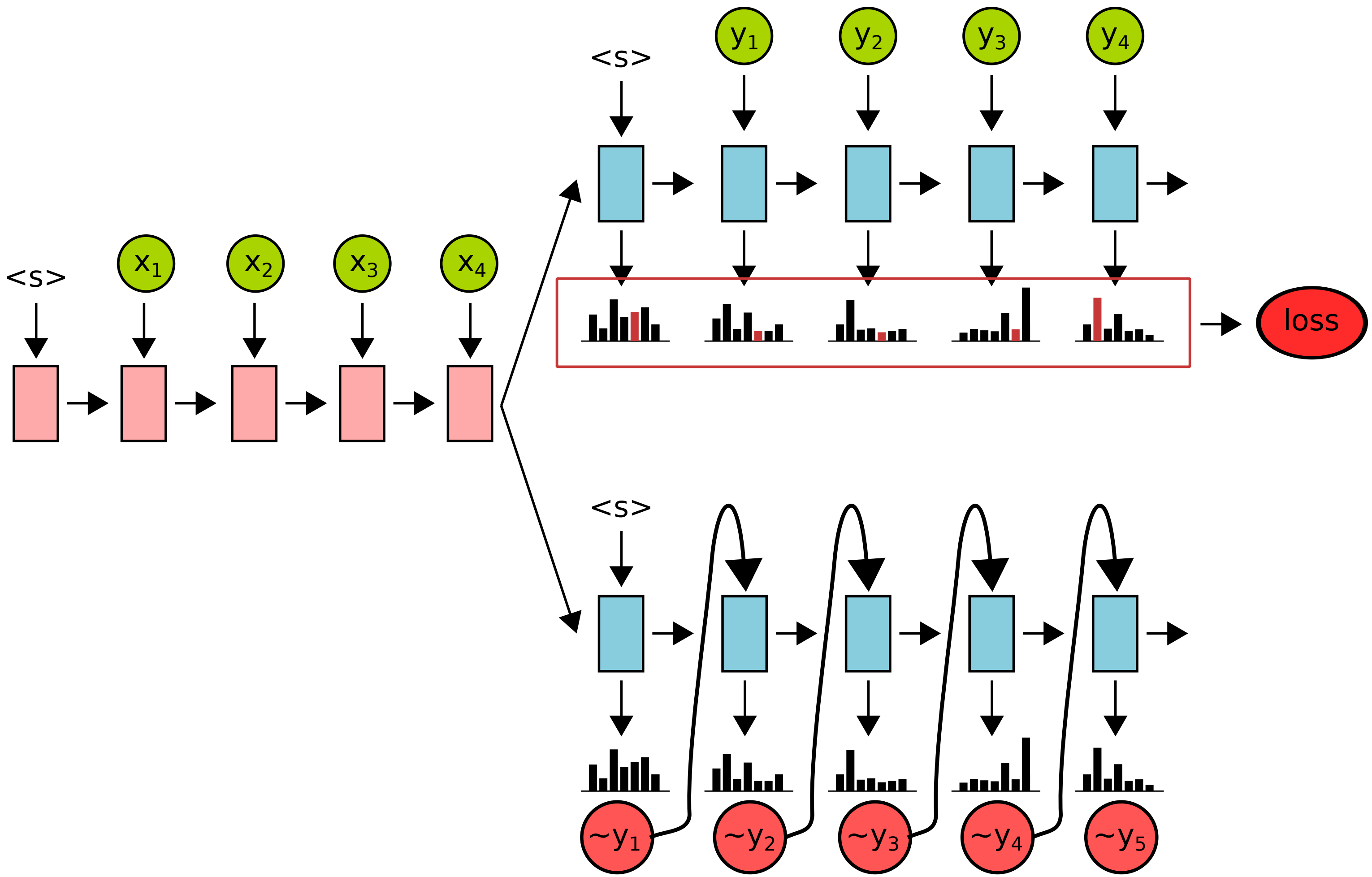
- "Autoregressive decoder" = conditioned on its preceding output.

Autoregressive Decoding

```
last_w = "<s>"
while last_w != "</s>":
    last_w_embedding = target_embeddings[last_w]
    state, dec_output = dec_cell(state,
                                 last_w_embedding)
    logits = output_projection(dec_output)
    last_w = np.argmax(logits)
    yield last_w
```

RNN Training vs. Runtime

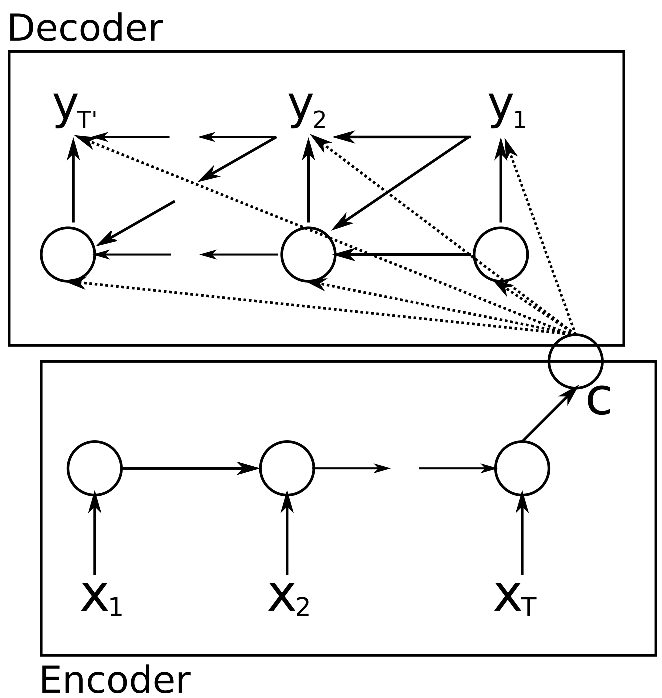
runtime: $\hat{y}_j$ (decoded) $\times$ **training:** y_j (ground truth)



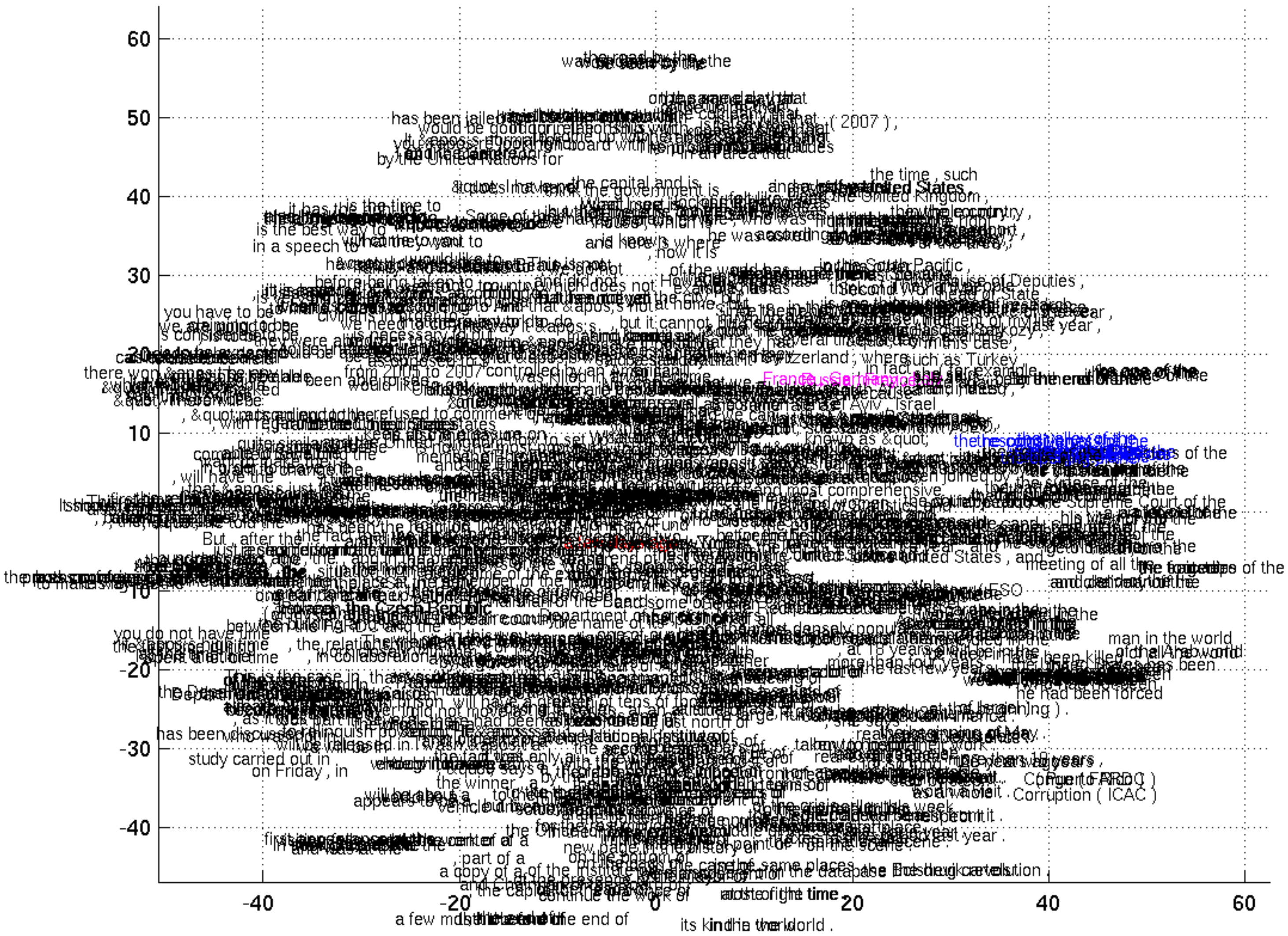
NNs as Translation Model in SMT

Cho et al. (2014) proposed:

- encoder-decoder architecture and
- GRU unit (name given later by Chung et al. (2014))
- to score variable-length phrase pairs in PBMT.

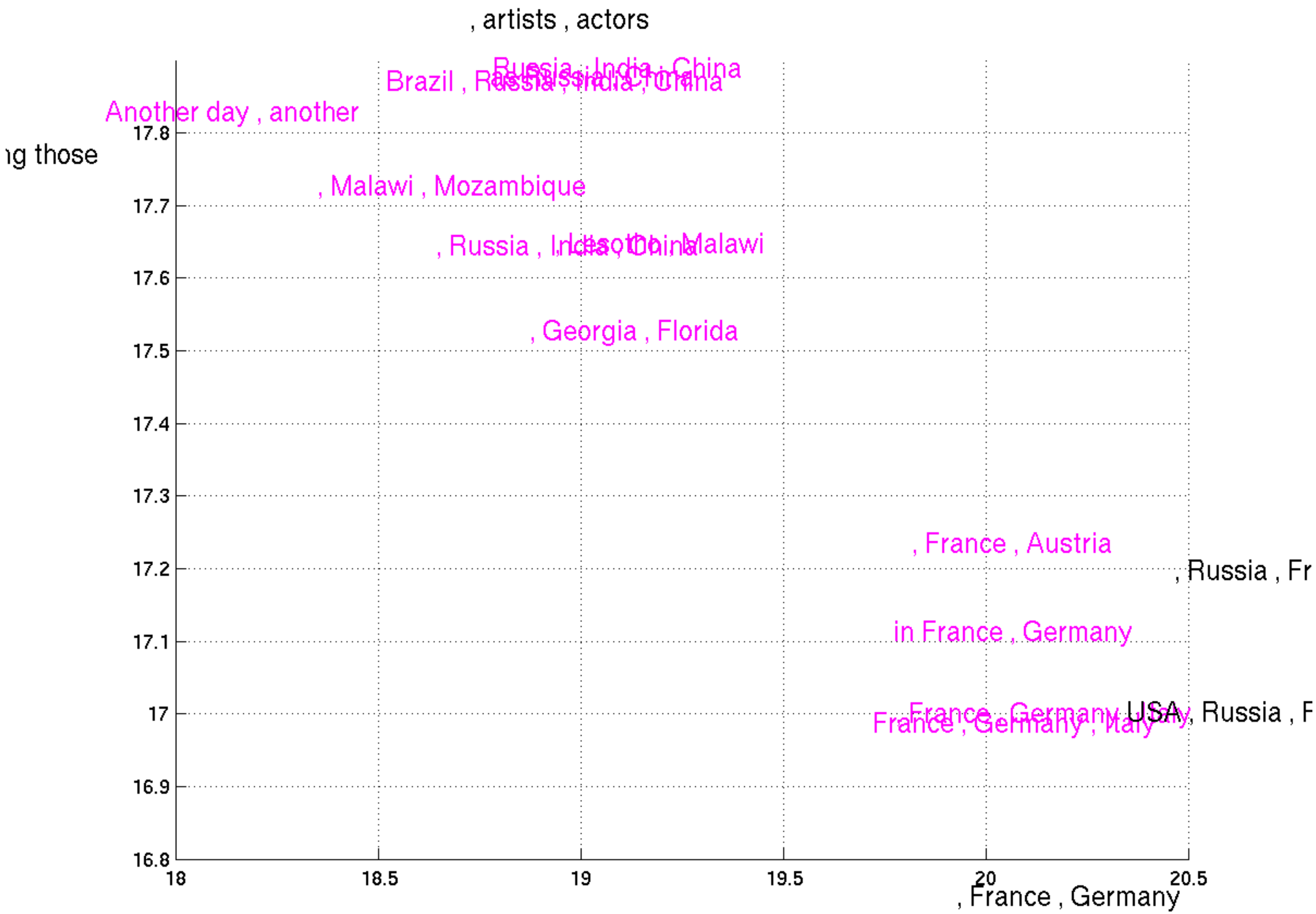


⇒ Embeddings of Phrases



⇒ Syntactic Similarity (“of the”)

⇒ Semantic Similarity (Countries)

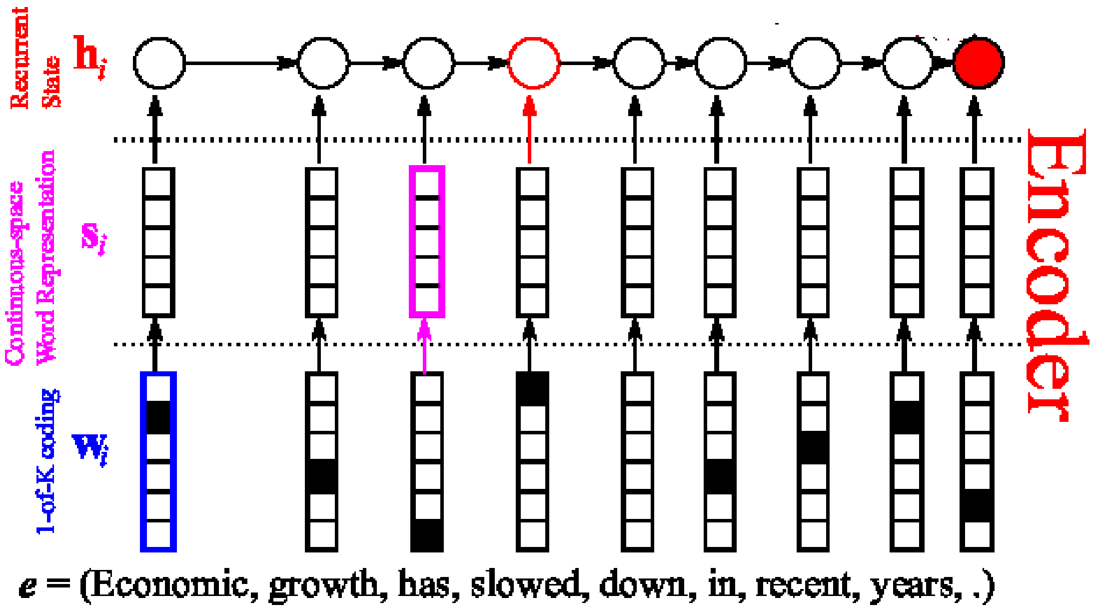


NMT: Sequence to Sequence

Sutskever et al. (2014) use:

- LSTM RNN encoder-decoder
- to consume
  and produce variable-length sentences.

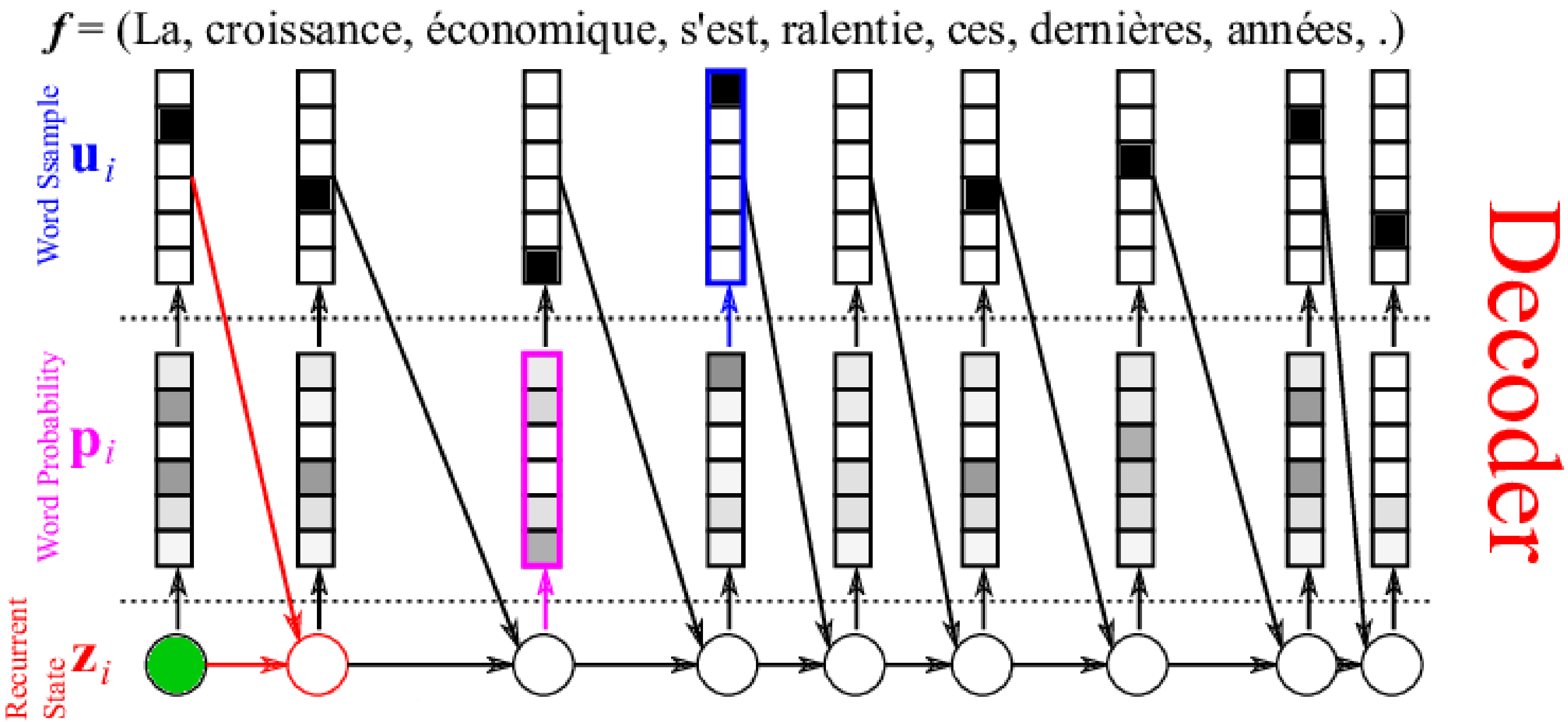
First the Encoder:



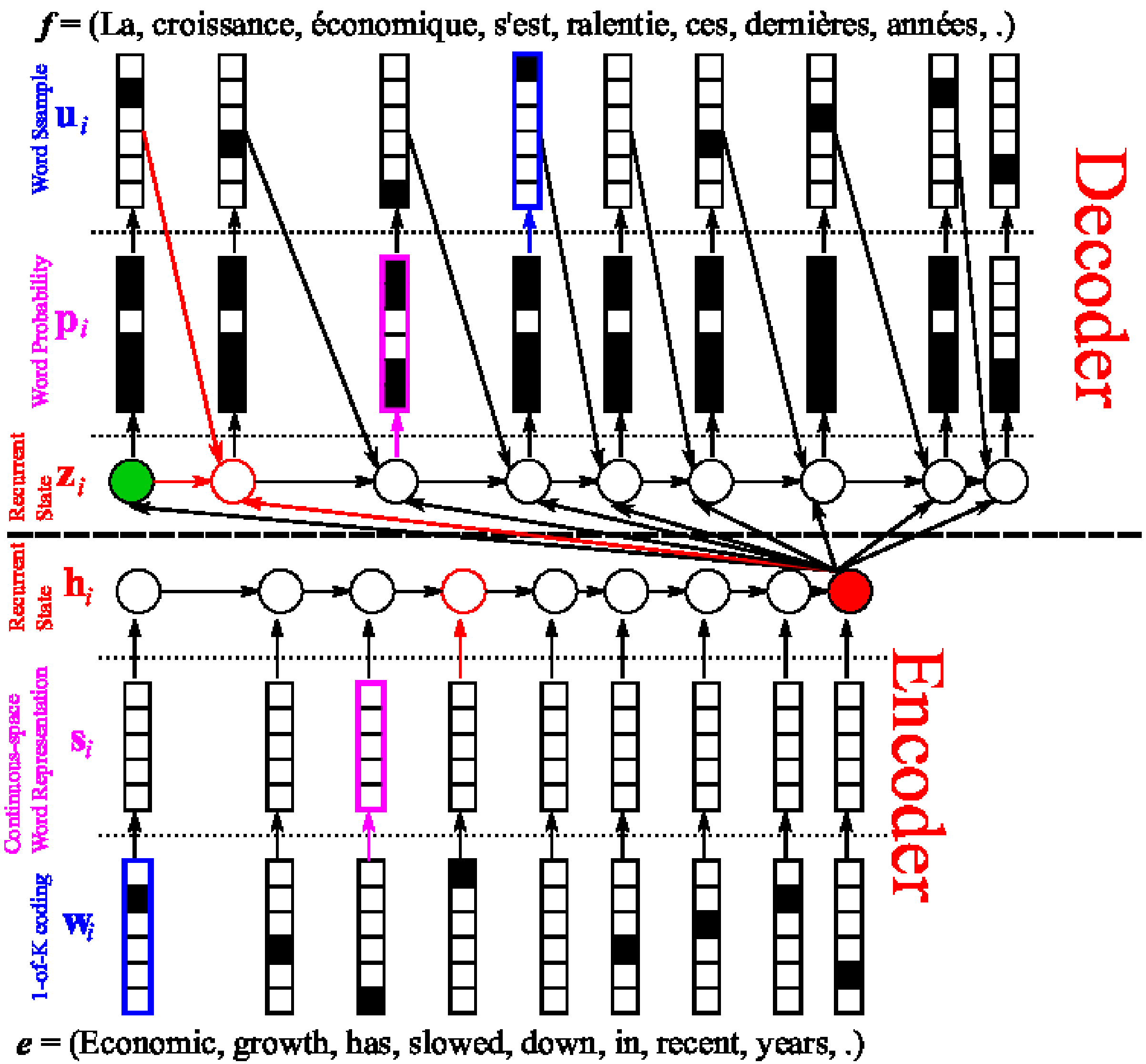
Then the Decoder

Remember: $p(e_1^I|f_1^J) = p(e_1|f_1^J) \cdot p(e_2|e_1, f_1^J) \cdot p(e_3|e_2, e_1, f_1^J) \dots$

- Again RNN, producing one word at a time.
- The produced word fed back into the network.
  - (Word embeddings in the target language used here.)

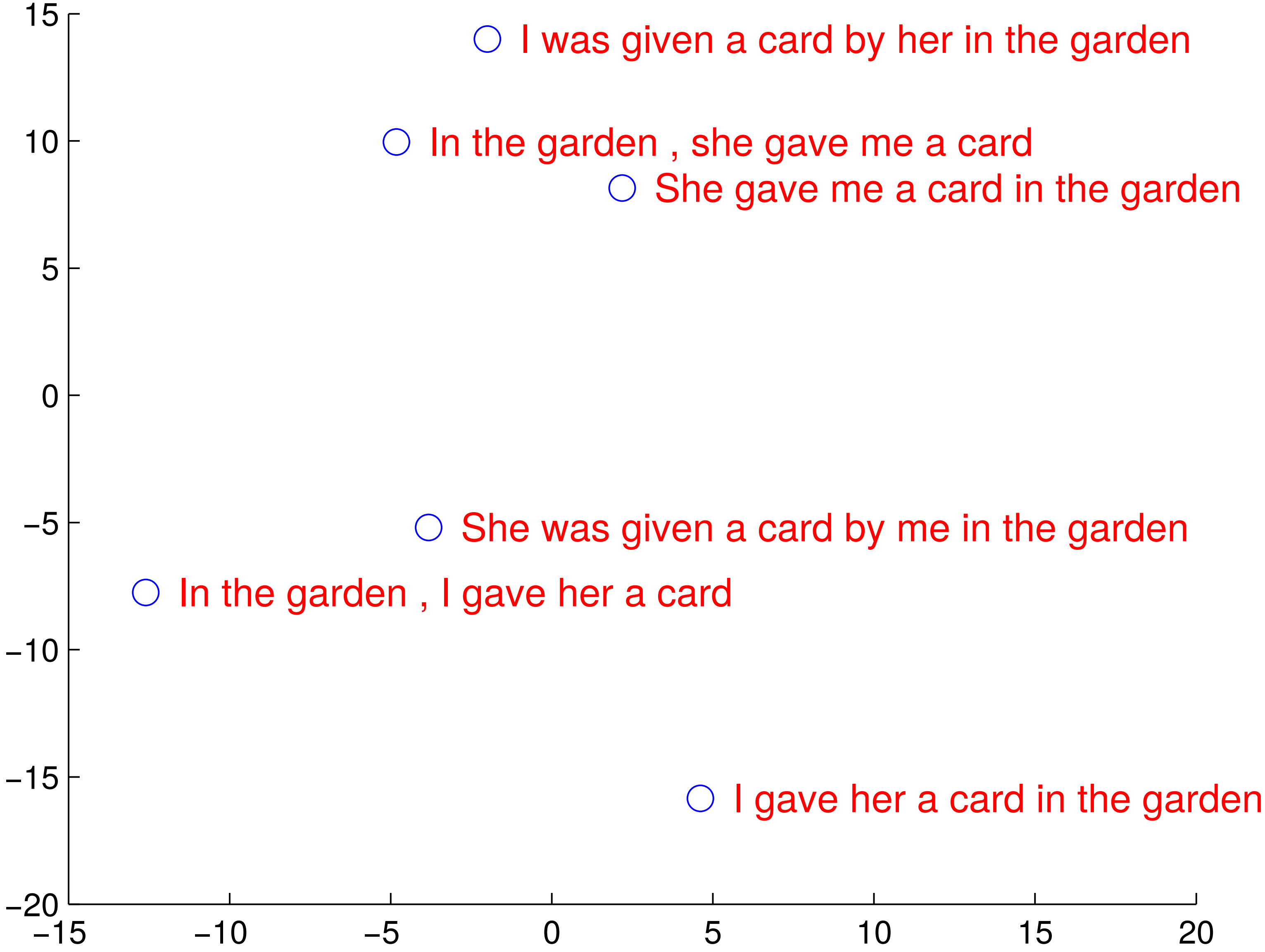


Encoder-Decoder Architecture



https://devblogs.nvidia.com/parallelforall/introduction-neural-machine-translation-gpus-part-2/

Continuous Space of Sentences



2-D PCA projection of 8000-D space representing sentences (Sutskever et al., 2014).

Architectures in the Decoder

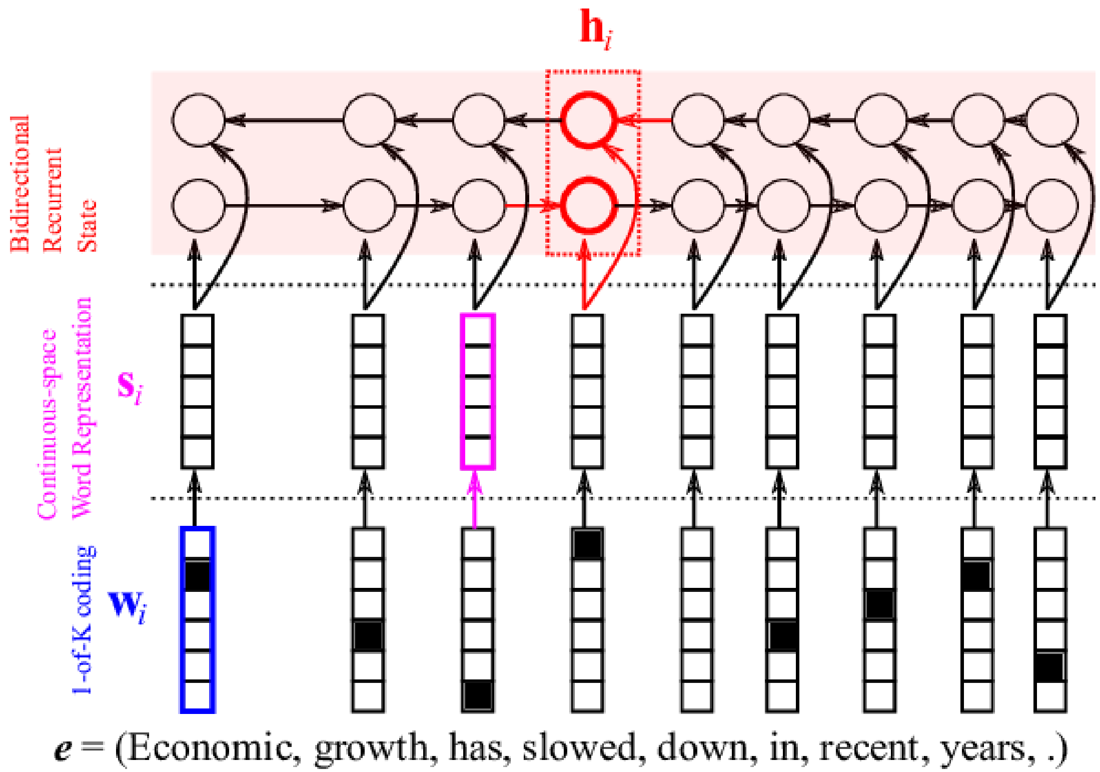
- RNN – original sequence-to-sequence learning (2015)
  - principle known since 2014 (University of Montreal)
  - made usable in 2016 (University of Edinburgh)
- CNN – convolution sequence-to-sequence by Facebook (2017)
- Self-attention (so called Transformer) by Google (2017)

Attention (1/3)

- Arbitrary-length sentences fit badly into a fixed vector.
- Reading input backward works better.

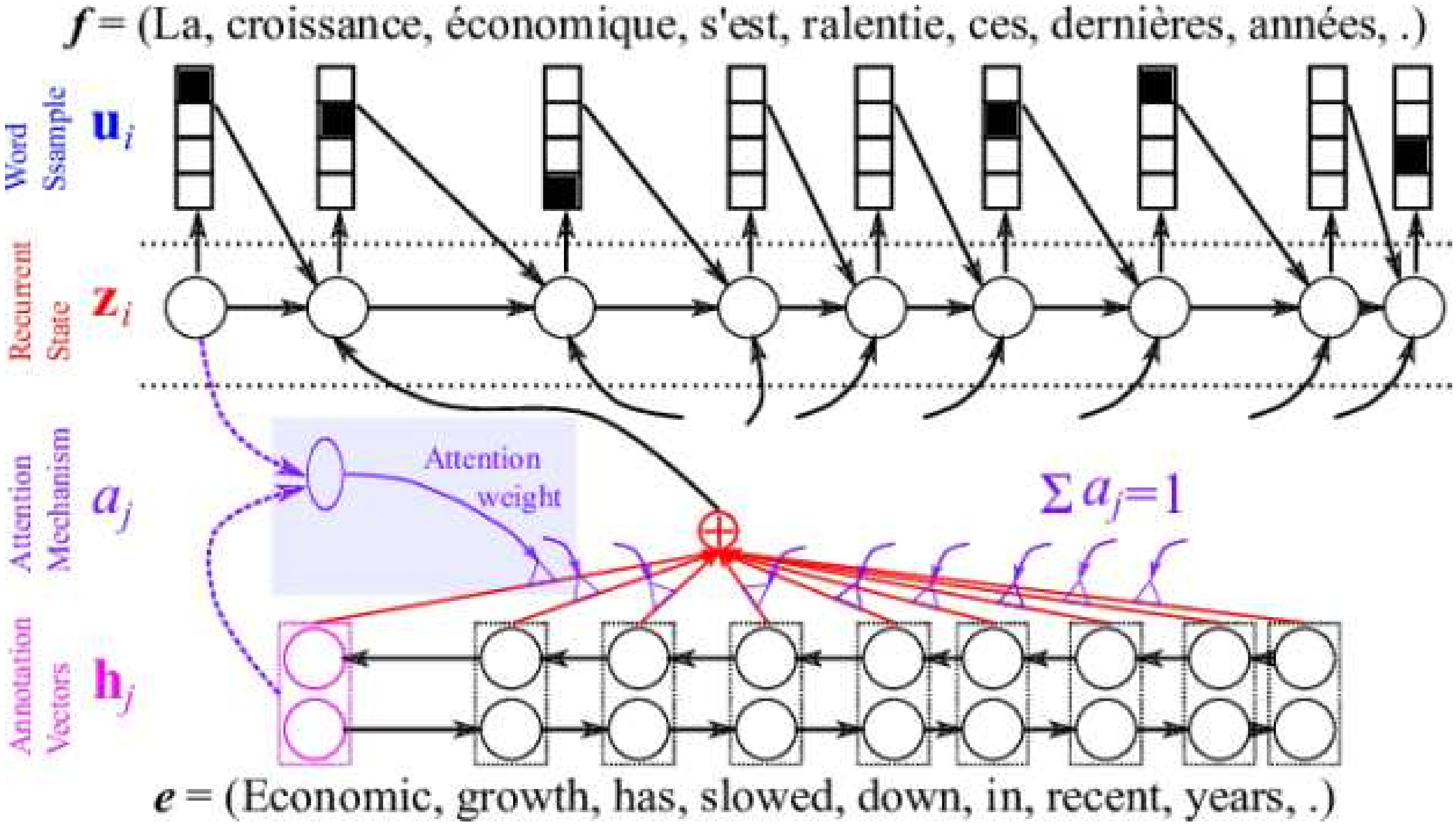
... because early words will be more salient.

⇒ Use Bi-directional RNN and "attend" to all states h_i.

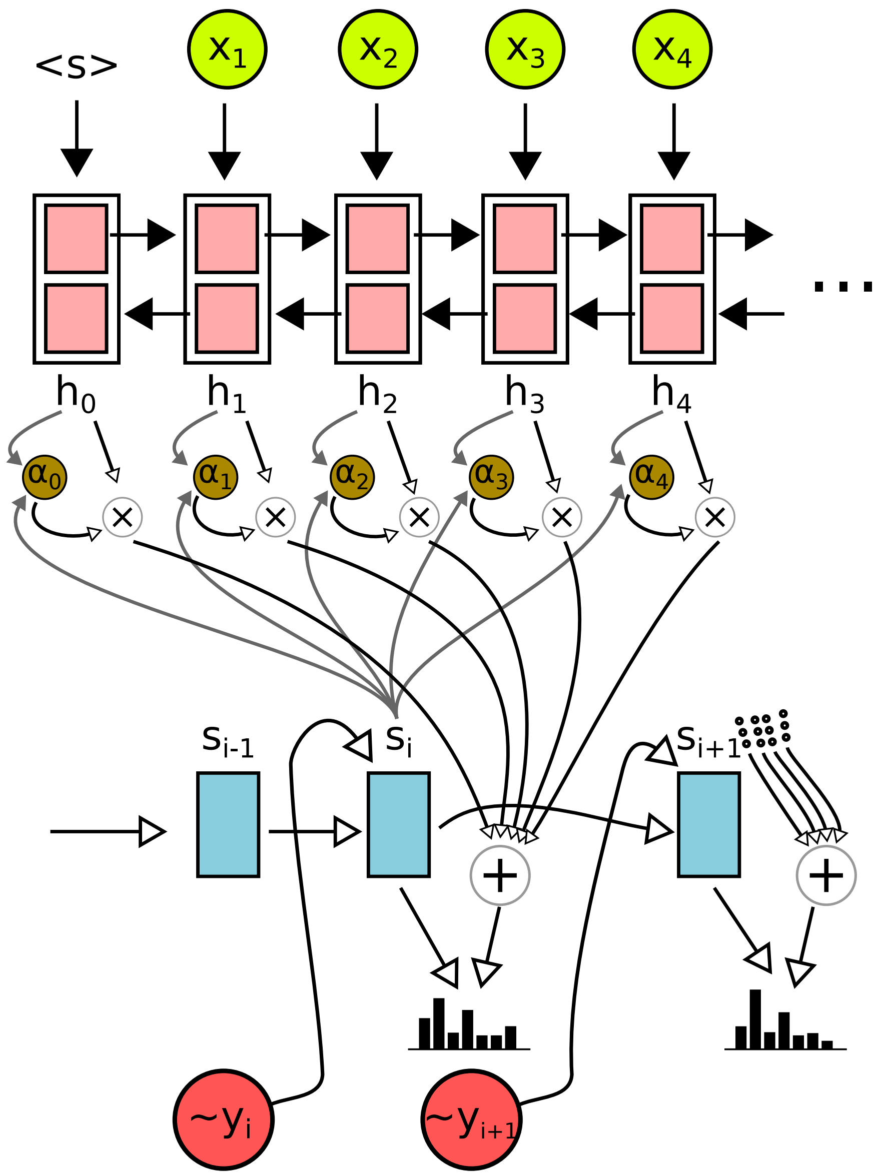


Attention (2/3)

- Add a sub-network prediciing importance of source states at each step.



Attention (3/3)



Attention Model in Equations (1)

Inputs:

decoder state: s_i, encoder states: $h_j = \left[\overrightarrow{h_j}; \overleftarrow{h_j}\right] \quad \forall i = 1 \ldots T_x$

Attention energies:

$$e_{ij} = v_a^\top \tanh\left(W_a s_{i-1} + U_a h_j + b_a\right)$$

Attention distribution: $\alpha_{ij} = \frac{\exp\left(e_{ij}\right)}{\sum_{k=1}^{Tx} \exp(e_{ik})}$

Context vector: $c_i = \sum_{j=1}^{T_x} \alpha_{ij} h_j$

Attention Model in Equations (2)

Output projection:

$$t_i = \mathrm{MLP}\left(U_o s_{i-1} + V_o E y_{i-1} + C_o c_i + b_o\right)$$

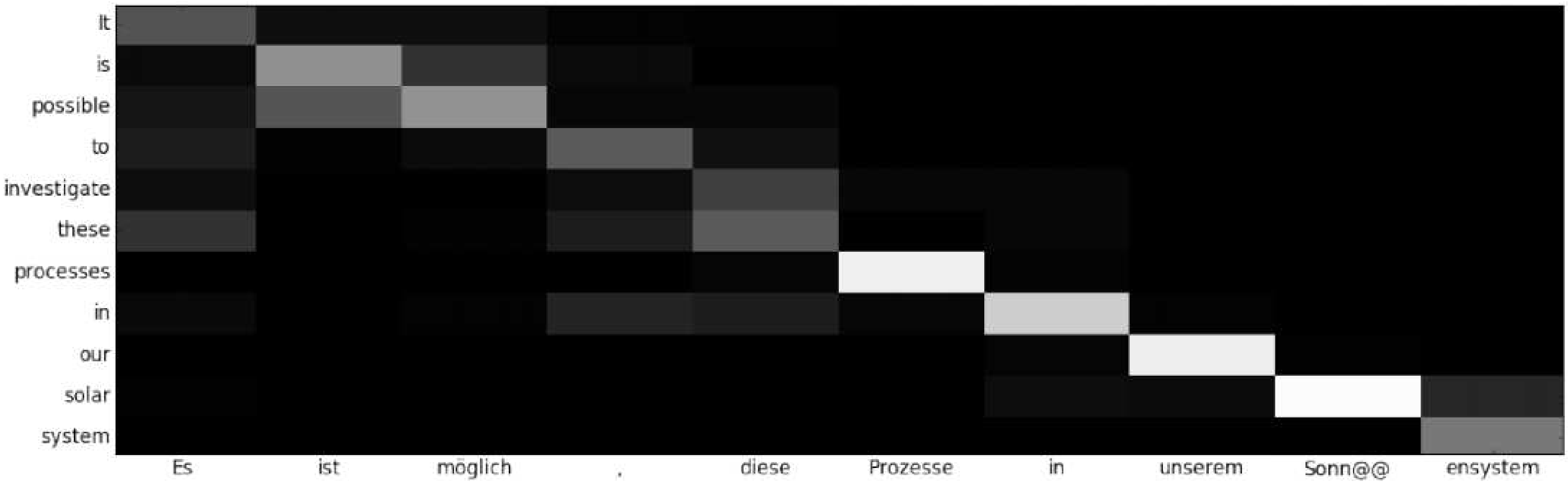
. . . attention is mixed with the hidden state

Output distribution:

$$p\left(y_i = k \mid s_i, y_{i-1}, c_i\right) \propto \exp\left(W_o t_i\right)_k + b_k$$

Attention ≈ Alignment

- We can collect the attention across time.
- Each column corresponds to one decoder time step.
- Source tokens correspond to rows.



Ultimate Goal of SMT vs. NMT

Goal of "classical" SMT:

Find **minimum translation units** $\sim$ graph partitions:

- such that they are frequent across many sentence pairs.
- without imposing (too hard) constraints on reordering.
- in an unsupervised fashion.

Goal of neural MT:

Avoid minimum translation units. Find NN architecture that

- Reads input in as original form as possible.
- Produces output in as final form as possible.
- Can be optimized end-to-end in practice.

Is NMT That Much Better?

The outputs of this year's best system: `http://matrix.statmt.org/`

SRC	A 28-year-old chef who had recently moved to San Francisco was found dead in the stairwell of a local mall this week.
	Osmadvacetiletý kuchař, který se nedávno přestěhoval do San Francisca, byl tento týden nalezen mrtvý na schodišti místního obchodního centra.
	Osmadvacetiletý šéfkuchař, který se nedávno přistěhoval do San Franciska, byl tento týden $\emptyset$ schodech místního obchodu.

SRC	There were creative differences on the set and a disagreement.
	Došlo ke vzniku kreativních rozdílů na scéně a k neshodám.
	Na place byly tvůrčí rozdíly a neshody.

Is NMT That Much Better?

The outputs of this year's best system: `http://matrix.statmt.org/`

SRC	A 28-year-old chef who had recently moved to San Francisco was found dead in the stairwell of a local mall this week.
MT	Osmadvacetiletý kuchař, který se nedávno přestěhoval do San Francisca, byl tento týden nalezen mrtvý na schodišti místního obchodního centra.
REF	Osmadvacetiletý šéfkuchař, který se nedávno přistěhoval do San Franciska, byl tento týden ∅ schodech místního obchodu.

SRC	There were creative differences on the set and a disagreement.
REF	Došlo ke vzniku kreativních rozdílů na scéně a k neshodám.
MT	Na place byly tvůrčí rozdíly a neshody.

Luckily ;-) Bad Errors Happen

SRC ... said Frank initially stayed in hostels...
MT ... řekl, že Frank původně zůstal v **Budějovicích**...

SRC Most of the Clintons' income...
MT Většinu příjmů **Kliniky**...

SRC The 63-year-old has now been made a special representativ
MT 63letý **mladík** se nyní stal zvláštním zástupcem...

SRC He listened to the moving stories of the women.
MT Naslouchal **pohyblivým** příběhům žen.

Catastrophic Errors

SRC	Criminal Minds star Thomas Gibson sacked after hitting producer
REF	Thomas Gibson, hvězda seriálu Myšlenky zločince, byl propuštěn po té, co uhodil režiséra
MT	**Kriminalisté Minsku** hvězdu Thomase Gibsona **vyhostili** po **zásahu** producenta

SRC	...add to that its long-standing grudge...
REF	...přidejte k tomu svou dlouholetou nenávist...
MT	...přidejte k tomu svou dlouholetou **záštitu**... (grudge $\rightarrow$ zášť $\rightarrow$ záštita)

German→Czech SMT vs. NMT

- A smaller dataset, very first (but comparable) results.
- NMT performs better on average, but occasionally:

SRC	Das Spektakel ähnelt dem Eurovision Song Contest.
REF	Je to jako pěvecká soutěž Eurovision.
SMT	Podívan**ou** připomíná hudební soutěž Eurovize.
NMT	Divadlo se podobá Eurovizi **Conview**.

SRC	Erderwärmung oder Zusammenstoß mit Killerasteroid.
REF	Globální oteplení nebo kolize se zabijáckým asteroidem.
SMT	Globální oteplování, nebo srážka s **Killerasteroid**.
NMT	Globální oteplování, nebo střet **s zabijákem**.

SRC	Zu viele verletzte Gefühle.
REF	Příliš mnoho nepřátelských pocitů.
SMT	Příliš mnoho zraněných pocit**y**.
NMT	Příliš mnoho zraněných $\emptyset$.

Summary

- What makes MT statistical.

Two crucially different models covered:

- Phrase-based: contiguous but independent phrases.
  - Bayes Law as a special case of Log-Linear Model.
  - Hand-crafted features (scoring functions); local vs. non-local.
  - Decoding as search, expanding partial hypotheses.
- Neural: unit-less, continuous space.
  - NMT as a fancy Language Model.
  - Word embeddings, subwords.
  - RNNs for variable-length input and output.
  - Attention model.

References

Kyunghyun Cho, Bart van Merrienboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using rnn encoder–decoder for statistical machine translation. In Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), pages 1724–1734, Doha, Qatar, October. Association for Computational Linguistics.

Junyoung Chung, Çaglar Gülçehre, KyungHyun Cho, and Yoshua Bengio. 2014. Empirical evaluation of gated recurrent neural networks on sequence modeling. CoRR, abs/1412.3555.

Philipp Koehn. 2003. Noun Phrase Translation. Ph.D. thesis, University of Southern California.

Adam Lopez. 2009. Translation as weighted deduction. In Proceedings of the 12th Conference of the European Chapter of the ACL (EACL 2009), pages 532–540, Athens, Greece, March. Association for Computational Linguistics.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. CoRR, abs/1301.3781.

Franz Joseph Och. 2002. Statistical Machine Translation: From Single-Word Models to Alignment Templates. Ph.D. thesis, RWTH Aachen University.

Franz Josef Och. 2003. Minimum Error Rate Training in Statistical Machine Translation. In Proc. of the Association for Computational Linguistics, Sapporo, Japan, July 6-7.

Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. 2014. Sequence to Sequence Learning with Neural Networks. pages 3104–3112.