From Sequence Modeling to Translation

Kyunghyun Cho

Institut de Montréal des algorithmes d'apprentissage,
Département d'informatique et de recherche opérationnelle,
Facult des arts et des sciences,
Université de Montréal
cho.k.hyun@gmail.com (kyunghyun.cho@umontreal.ca)

Natural Language Processing

In neuropsychology, linguistics and the philosophy of language, a **natural language** *or ordinary language is any language which arises, unpremeditated, in the brains of human beings.*

–Wikipedia

What does it mean to understand natural languages with machine learning?

How likely is a given sentence?

A sentence $\mathbf{x} = (x_1, x_2, \ldots, x_T)$

- x_i: the i-th word
- Example: (A, tenured, professor, likes, drinking, a, lot, of, scottish, whiskey, .)

How likely is this sentence?

How likely is a given sentence?

A sentence $\mathbf{x} = (x_1, x_2, \ldots, x_T)$

- x_i: the i-th word
- Example: (A, tenured, professor, likes, drinking, a, lot, of, scottish, whiskey, .)

How likely is this sentence?

In other words, what is the probability of (A, tenured, professor, likes, drinking, a, lot, of, scottish, whiskey, .)?

How likely is a given sentence?

A sentence $\mathbf{x} = (x_1, x_2, \ldots, x_T)$

- x_i: the i-th word
- Example: (A, tenured, professor, likes, drinking, a, lot, of, scottish, whiskey, .)

How likely is this sentence?

In other words, what is the probability of (A, tenured, professor, likes, drinking, a, lot, of, scottish, whiskey, .)?

In mathematical terms, $p\left((x_1, x_2, \ldots, x_T)\right)$?

How likely is a given sentence? – n-Gram LM

Conditional probability

$$p(X \mid Y) = \frac{p(X, Y)}{p(Y)} \iff p(X, Y) = p(X \mid Y)p(Y)$$

How likely is a given sentence? – n-Gram LM

Conditional probability

$$p(X \mid Y) = \frac{p(X, Y)}{p(Y)} \iff p(X, Y) = p(X \mid Y)p(Y)$$

Factorization of the joint probability

$$\begin{aligned} p(x_1, x_2, \ldots, x_T) &= p(x_1)p(x_2, \ldots, x_T \mid x_1) \\ &= p(x_1)p(x_2 \mid x_1)p(x_3, \ldots, x_T \mid x_2, x_1) \\ &= \prod_{t=1}^{T} p(x_t \mid x_1, \ldots, x_{t-1}) \end{aligned}$$

How likely is a given sentence? – n-Gram LM

n-Gram Language Model

- n-th order Markov assumption

Factorization of the joint probability

$$p(x_1, x_2, \ldots, x_T) = \prod_{t=1}^{T} p(x_t \mid x_1, \ldots, x_{t-1})$$

How likely is a given sentence? – n-Gram LM

n-Gram Language Model

- n-th order Markov assumption

Factorization of the joint probability

$$p(x_1, x_2, \ldots, x_T) = \prod_{t=1}^{T} p(x_t \mid x_1, \ldots, x_{t-1})$$

n-th order Markov assumption

$$p(x_1, x_2, \ldots, x_T) \approx \prod_{t=1}^{T} p(x_t \mid x_{t-n}, \ldots, x_{t-1})$$

How likely is a given sentence? – n-Gram LM

n-Gram Language Model

- n-th order Markov assumption

$$p(x_1, x_2, \ldots, x_T) \approx \prod_{t=1}^{T} p(x_t \mid x_{t-n}, \ldots, x_{t-1})$$

Trigram modeling of (a, tenured, professor, likes, drinking, whiskey, .)

1. How likely is 'a' at the beginning?
2. How likely does 'tenured' follow 'A'?
3. How likely does 'professor' follow 'A tenured'?
4. How likely does 'likes' follow 'tenured professor'?
5. How likely does 'drinking' follow 'professor likes'?
6. How likely does 'whiskey' follow 'likes drinking'?

How likely is a given sentence? – n-Gram LM

n-Gram Language Model

- n-th order Markov assumption
  $p(x_1, x_2, \ldots, x_T) \approx \prod_{t=1}^{T} p(x_t \mid x_{t-n}, \ldots, x_{t-1})$
- Non-parametric Estimator

Trigram modeling of (a, tenured, professor, likes, drinking, whiskey, .)

Context	Word	Probability
_ _	a	-
_ a	tenured	-
a tenured	professor	-
tenured professor	likes	-
professor likes	drinking	-
likes drinking	whiskey	-
drinking whiskey	.	-
whiskey .	_	-

What can go wrong with n-Gram LM?

n-Gram Language Model

- n-th order Markov assumption
  $p(x_1, x_2, \ldots, x_T) \approx \prod_{t=1}^{T} p(x_t \mid x_{t-n}, \ldots, x_{t-1})$
- Non-parametric Estimator

What if "a tenured professor" was not in a training set?

What can go wrong with n-Gram LM?

n-Gram Language Model

- n-th order Markov assumption
  $p(x_1, x_2, \ldots, x_T) \approx \prod_{t=1}^{T} p(x_t \mid x_{t-n}, \ldots, x_{t-1})$
- Non-parametric Estimator

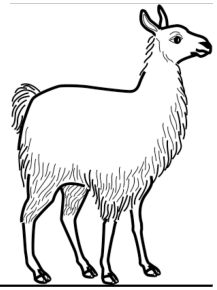
What if "a tenured professor" was not in a training set?

$$p((\text{a, tenured, professor, likes, drinking, whiskey, .})) =$$
$$p(\text{a} \mid _, _) \cdot p(\text{tenured} \mid \text{a}, _) \cdot \underbrace{p(\text{professor} \mid \text{a, tenured})}_{=0} \cdot$$
$$p(\text{likes} \mid \text{tenured, professor}) \cdot \cdots \cdot p(_ \mid \text{whiskey, .})$$
$$= 0$$

What can go wrong with n-Gram LM?

n-Gram Language Model

- n-th order Markov assumption
  $p(x_1, x_2, \ldots, x_T) \approx \prod_{t=1}^{T} p(x_t \mid x_{t-n}, \ldots, x_{t-1})$
- Non-parametric Estimator



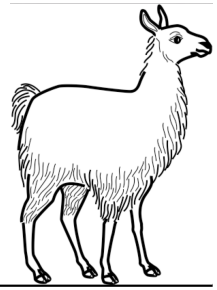
Two closely-related issues

1. Data sparsity
   - # of all possible n-grams: $\|V\|^n$, $\|V\|$: the size of vocabulary
   - Only a fraction of these are present in data

What can go wrong with n-Gram LM?

n-Gram Language Model

- n-th order Markov assumption
  $p(x_1, x_2, \ldots, x_T) \approx \prod_{t=1}^{T} p(x_t \mid x_{t-n}, \ldots, x_{t-1})$
- Non-parametric Estimator



Two closely-related issues

1. Data sparsity
   - # of all possible n-grams: $\|V\|^n$, $\|V\|$: the size of vocabulary
   - Only a fraction of these are present in data
2. Lack of generalization
   - (chases, a, dog), (chases, a, cat), (chases, a, rabbit)
   - (chases, a, **llama**)?

How likely is a given sentence? – Neural Net LM

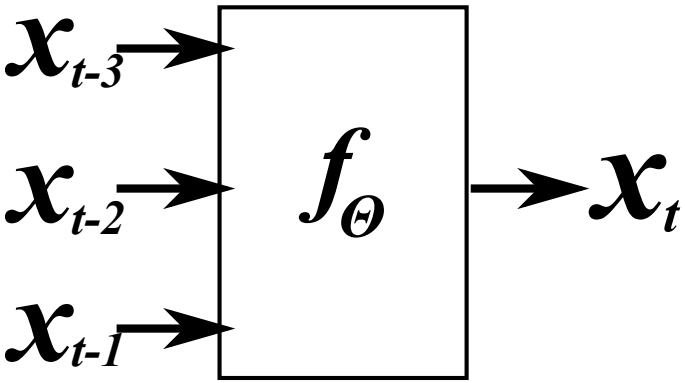
(Feedforward) Neural Net Language Model

- n-th order Markov assumption

$$p(x_1, x_2, \ldots, x_T) \approx \prod_{t=1}^{T} \mathbf{p}(\mathbf{x_t} \mid \mathbf{x_{t-n}}, \ldots, \mathbf{x_{t-1}})$$

- ~~Non-parametric Estimator~~ $\rightarrow$ **Parametric Estimator**

$$\mathbf{p}(\mathbf{x_t} \mid \mathbf{x_{t-n}}, \ldots, \mathbf{x_{t-1}}) \approx f_\theta(x_{t-n}, \ldots, x_{t-1})$$



How likely is a given sentence? – Neural Net LM

(Feedforward) Neural Net Language Model

- ~~Non-parametric Estimator~~ $\rightarrow$ **Parametric Estimator**

$$\mathbf{p}(\mathbf{x_t} \mid \mathbf{x_{t-n}}, \ldots, \mathbf{x_{t-1}}) \approx f_\theta(x_{t-n}, \ldots, x_{t-1})$$

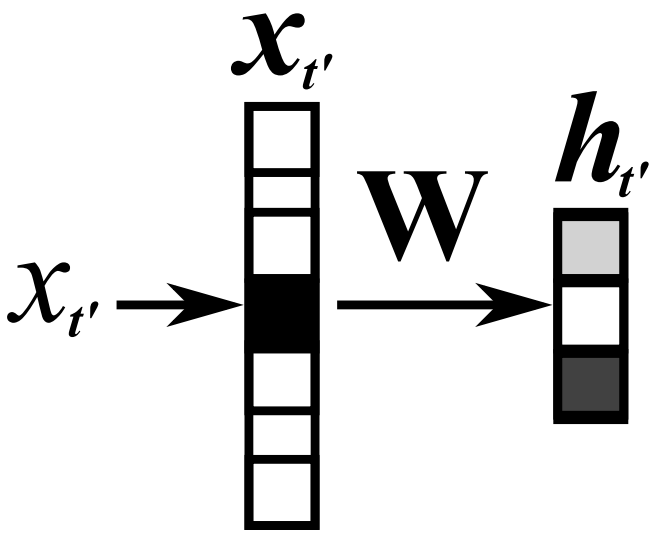
1. One-hot vector representation of $x_{t'}$

$$\mathbf{x}_{t'} = \left[0, \ldots, \underbrace{1}_{x_{t'}\text{-th element}}, \ldots, 0\right]^\top$$

2. Continuous-space representation of $x_{t'}$

$$h_{t'} = \mathbf{W}^\top \mathbf{x}_{t'} \in \mathbb{R}^d,$$

where $\mathbf{W} \in \mathbb{R}^{\|V\| \times d}$ and $d \ll \|V\|$



How likely is a given sentence? – Neural Net LM

(Feedforward) Neural Net Language Model

- ~~Non-parametric Estimator~~ $\rightarrow$ **Parametric Estimator**

$$\mathbf{p}(\mathbf{x_t} \mid \mathbf{x_{t-n}}, \ldots, \mathbf{x_{t-1}}) \approx f_\theta(x_{t-n}, \ldots, x_{t-1})$$

---

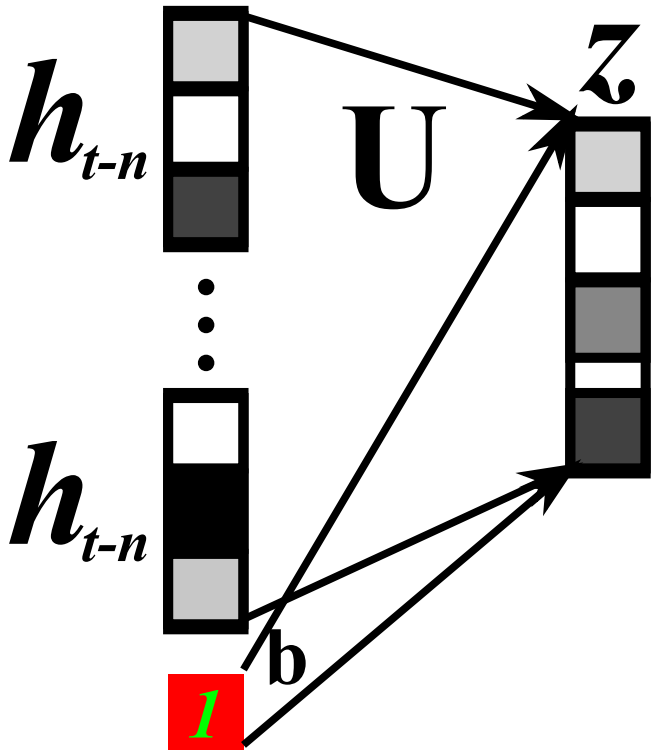
3. Concatenate $\mathbf{h}_{t'}$'s

$$\mathbf{h} = [\mathbf{h}_{t-n}; \ldots; \mathbf{h}_{t-1}]^\top$$

4. Nonlinear projection of $\mathbf{h}_{t'}$

$$\mathbf{z} = \phi\left(\mathbf{U}^\top \mathbf{h} + \mathbf{b}\right),$$

where $\mathbf{U} \in \mathbb{R}^{(n-1)d \times n_h}$ and $\mathbf{b} \in \mathbb{R}^{n_h}$



How likely is a given sentence? – Neural Net LM

(Feedforward) Neural Net Language Model

- ~~Non-parametric Estimator~~ → **Parametric Estimator**

$$\mathbf{p}(\mathbf{x_t} \mid \mathbf{x_{t-n}}, \ldots, \mathbf{x_{t-1}}) \approx f_\theta(x_{t-n}, \ldots, x_{t-1})$$

---

5. Unnormalized Probabilities

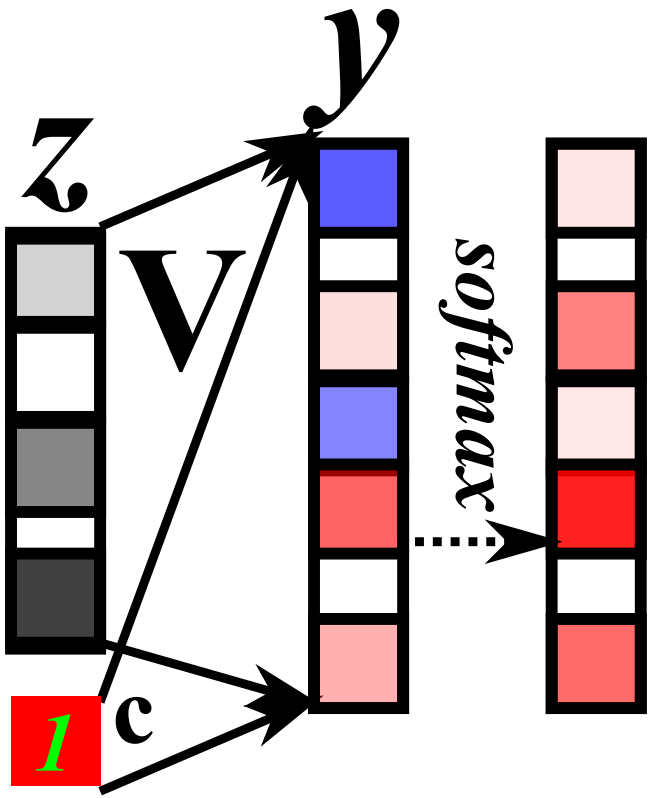
$$\mathbf{y} = \mathbf{V}^\top \mathbf{z} + \mathbf{c},$$

where $\mathbf{V} \in \mathbb{R}^{n_h \times \|V\|}$ and $\mathbf{c} \in \mathbb{R}^{\|V\|}$

6. Normalize for each word (*softmax*)

$$p(x_t = i \mid \ldots) = \frac{\exp\{y_i\}}{\sum_{j=1}^{\|V\|} \exp\{y_j\}}$$

where y_i is the i-th element of $\mathbf{y}$

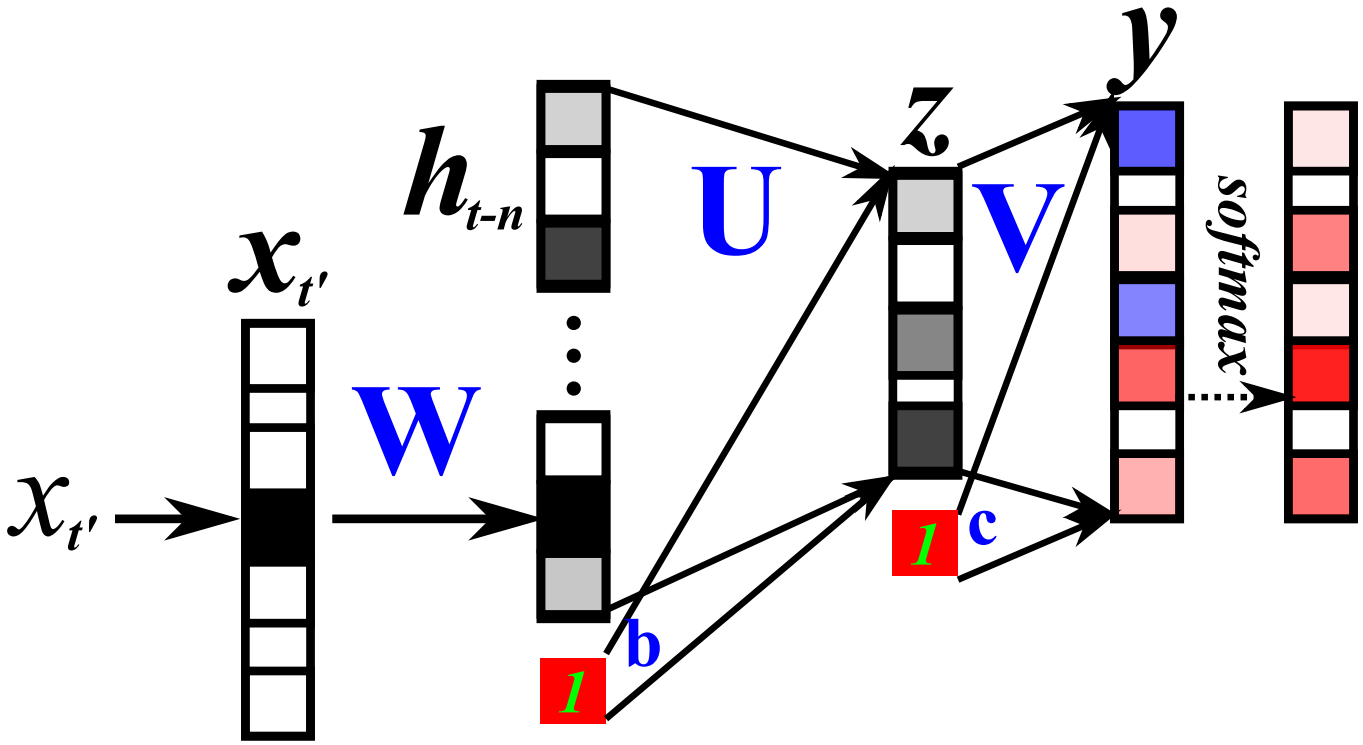


How likely is a given sentence? – Neural Net LM

Maximum Likelihood Estimation *with Backprop*

$$\theta = \arg\max_{\theta} \frac{1}{N} \sum_{n=1}^{N} \sum_{t=1}^{T_n} \log p_\theta(x_t \mid x_{t-n}, \ldots, x_{t-1}),$$

where there are n training sentences and $\theta = \{\mathbf{W}, \mathbf{U}, \mathbf{b}, \mathbf{V}, \mathbf{c}\}$



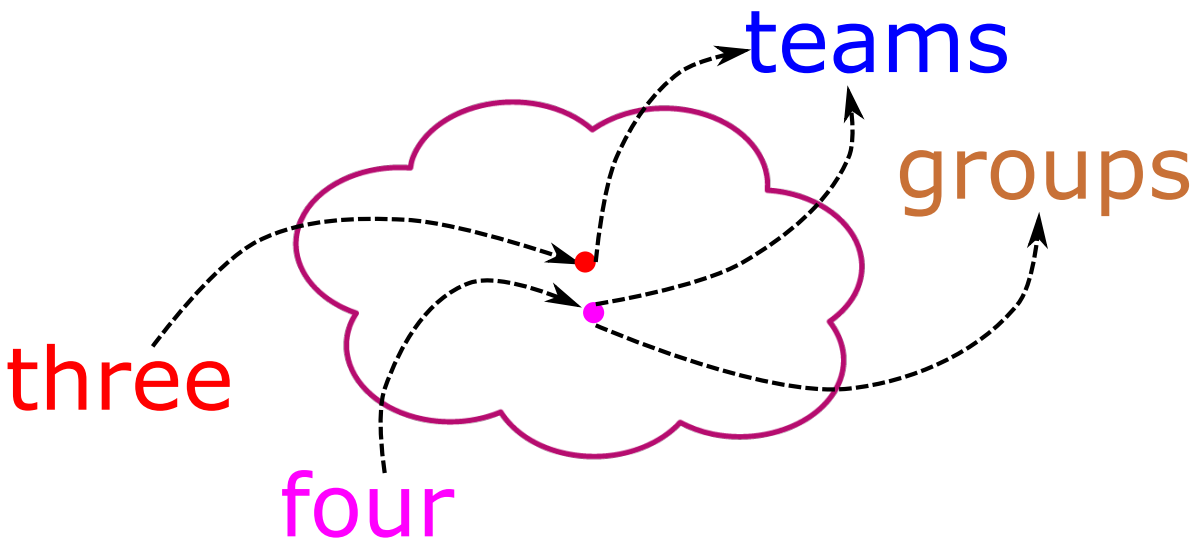
Hopefully, you all remember what backprop is..

Continuous-Space Representation – Neural Net LM Generalizes to Unseen n-Grams

Sample sentences:

1. There are three teams left for the qualification.
2. four teams have passed the first round.
3. four groups are playing in the field.

Neural Net LM: *compress* the corpus into a continuous function



Continuous-Space Representation – Neural Net LM Generalizes to Unseen n-Grams

Sample sentences:

1. There are three teams left for the qualification.
2. four teams have passed the first round.
3. four groups are playing in the field.

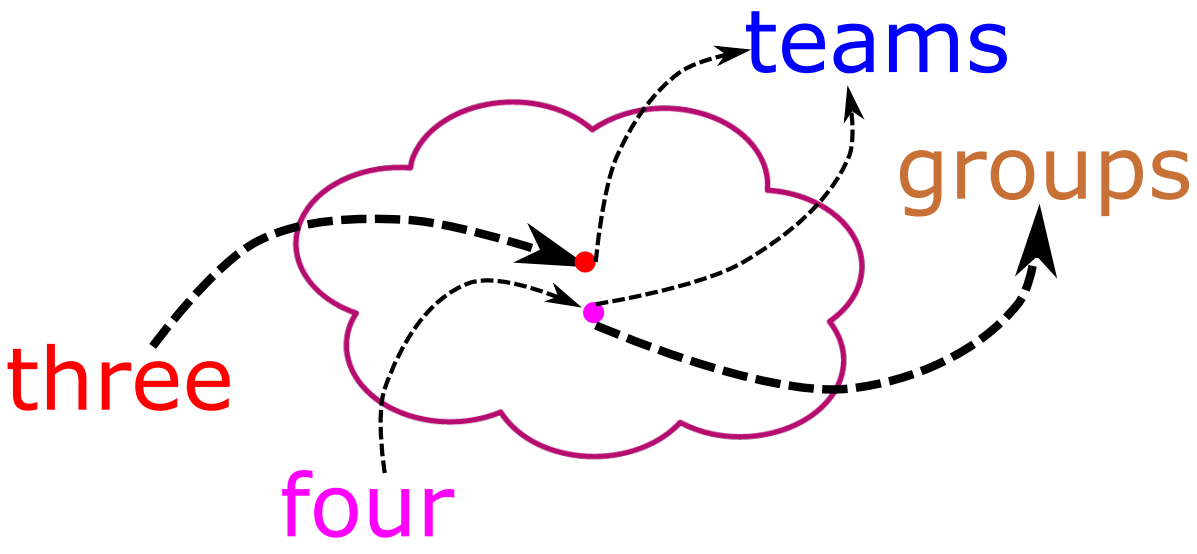
Question: How likely is groups followed by three?

Continuous-Space Representation – Neural Net LM Generalizes to Unseen n-Grams

Sample sentences:

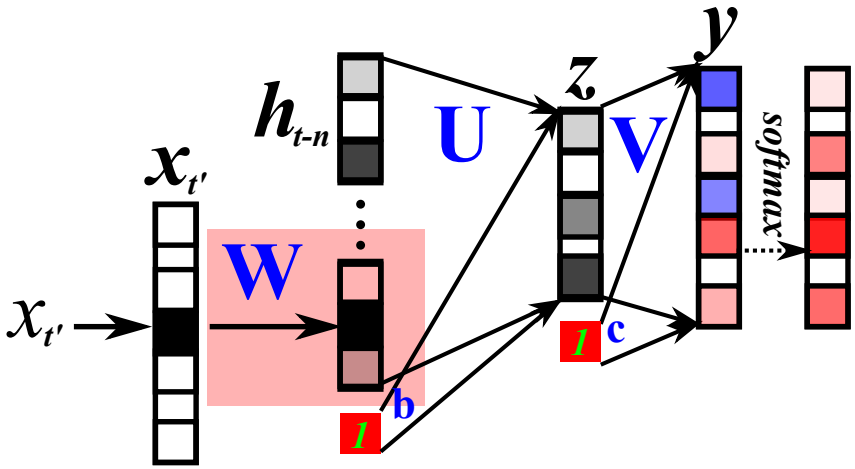
1. There are three teams left for the qualification.
2. four teams have passed the first round.
3. four groups are playing in the field.

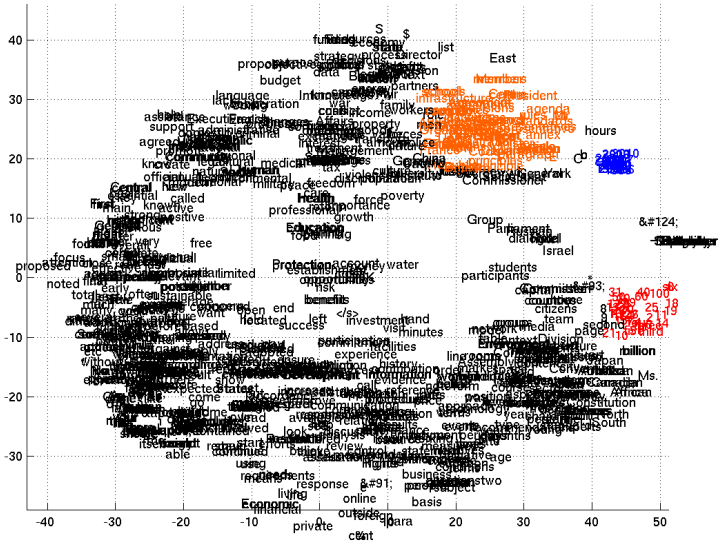
Question: How likely is groups followed by three?

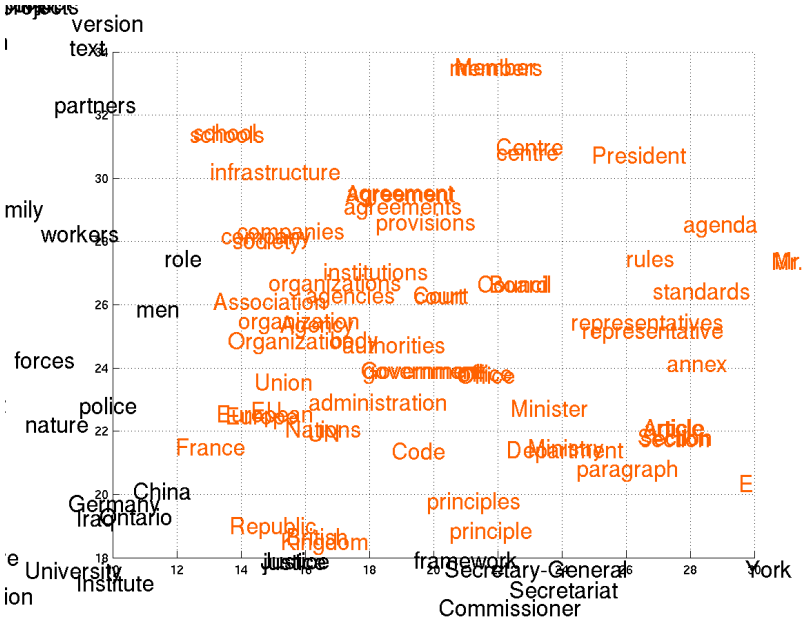


Continuous-Space Representation – *so-called* Word Embeddings

- **W**: a lookup table of word embeddings
- **z**: a phrase representation?







10-min Break

Can we **not** assume Markov property?

RNN Language Model

- ~~n-th order Markov assumption~~ $\rightarrow$ **Non-Markovian Model**
  $p(x_1, x_2, \ldots, x_T) \approx \prod_{t=1}^{T} p(x_t \mid x_1, \ldots, x_{t-n}, \ldots, x_{t-1})$
- ~~Non-parametric Estimator~~ $\rightarrow$ Parametric Estimator

---

Big Issue: *T is not bounded* – Variable-length input

Can we **not** assume Markov property?

RNN Language Model

- ~~n-th order Markov assumption~~ $\rightarrow$ **Non-Markovian Model**
  $p(x_1, x_2, \ldots, x_T) \approx \prod_{t=1}^{T} p(x_t \mid x_1, \ldots, x_{t-n}, \ldots, x_{t-1})$
- ~~Non-parametric Estimator~~ $\rightarrow$ Parametric Estimator

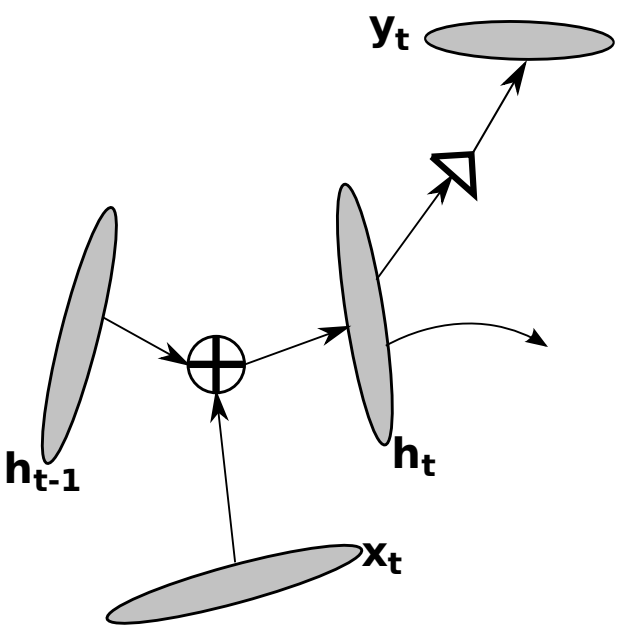
---

Big Issue: *T is not bounded* – Variable-length input

Solution: History summarization

1. Summary vector $\mathbf{h}$
2. Addition: $\mathbf{h} \leftarrow \mathbf{h} \oplus \mathbf{x}$
3. Predict the next word x_{t+1} given the summary $\mathbf{h}_t$

Recurrent Neural Network



Operators

1. Addition: $\mathbf{h}' = \mathbf{x} \oplus \mathbf{h}$
2. Pop: $\mathbf{x}' = \triangleright \mathbf{h}$

These are *learned* operators that do not have fixed meaning.

$\Longleftrightarrow$

The meaning of each operator is defined by the neural network architecture and *data*.

Recurrent Neural Network

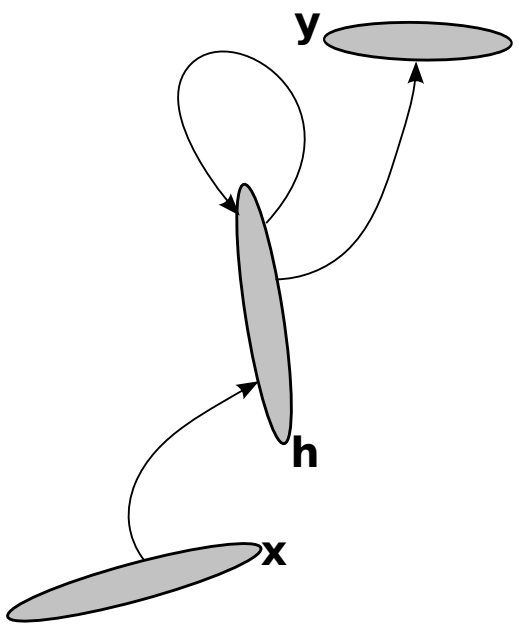
Natural way to deal with a variable-length input $(x_1, \ldots, x_T)$

- Addition

$$h_t = \begin{cases} 0, & t = 0 \\ \phi\left(h_{t-1}, x_t\left[, y_{t-1}\right]\right), & \text{otherwise} \end{cases}$$

- Pop

$$y_t = f(h_t, \ldots)$$



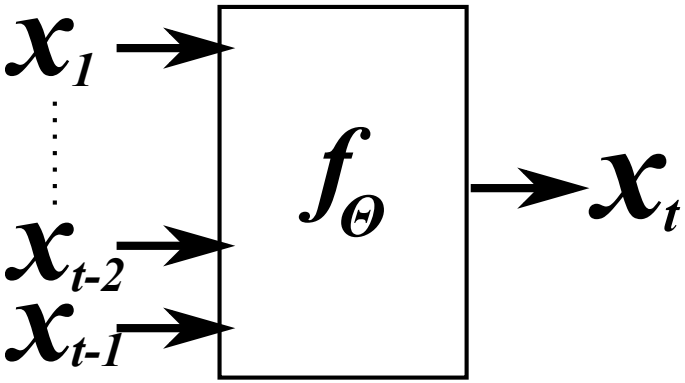
Language Modeling with Recurrent Neural Network

RNN Language Model

- ~~n-th order Markov assumption~~ $\rightarrow$ **Non-Markovian Model**
  $p(x_1, x_2, \ldots, x_T) \approx \prod_{t=1}^{T} p(x_t \mid x_1, \ldots, x_{t-n}, \ldots, x_{t-1})$
- ~~Non-parametric Estimator~~ $\rightarrow$ Parametric Estimator

---

$$\mathbf{p}(\mathbf{x_t} \mid \mathbf{x_1}, \ldots, \mathbf{x_{t-1}}) \approx f_\theta(x_1, \ldots, x_{t-1})$$



Language Modeling with Recurrent Neural Network

RNN Language Model

- ~~n-th order Markov assumption~~ $\rightarrow$ **Non-Markovian Model**
  $p(x_1, x_2, \ldots, x_T) \approx \prod_{t=1}^{T} p(x_t \mid x_1, \ldots, x_{t-n}, \ldots, x_{t-1})$
- ~~Non-parametric Estimator~~ $\rightarrow$ Parametric Estimator

---

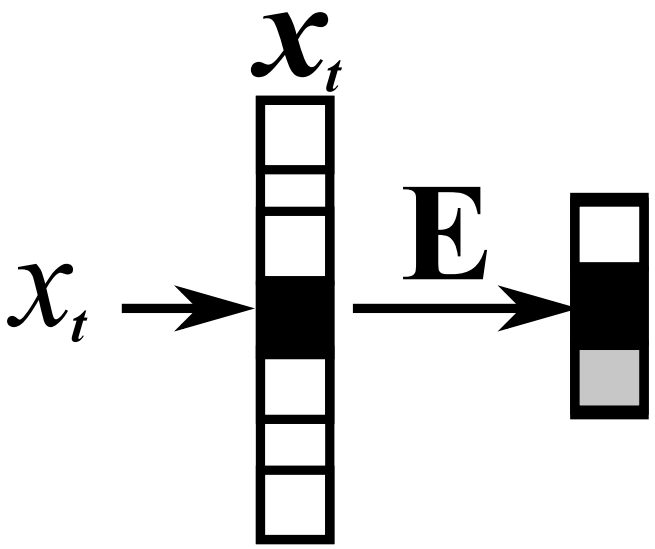
1. One-hot vector representation of x_t

$$\mathbf{x}_t = \left[0, \ldots, \underbrace{1}_{x_t\text{-th element}}, \ldots, 0\right]^\top$$

2. Continuous-space representation of x_t

$$\tilde{\mathbf{x}}_t = \mathbf{E}^\top \mathbf{x}_t \in \mathbb{R}^d,$$

where $\mathbf{E} \in \mathbb{R}^{\|V\| \times d}$ and $d \ll \|V\|$



Language Modeling with Recurrent Neural Network

RNN Language Model

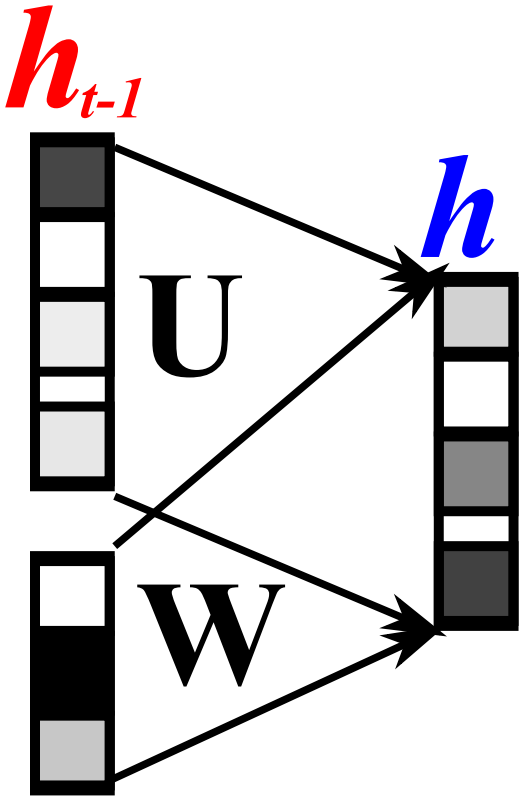
- ~~n-th order Markov assumption~~ $\rightarrow$ **Non-Markovian Model**
  $p(x_1, x_2, \ldots, x_T) \approx \prod_{t=1}^{T} p(x_t \mid x_1, \ldots, x_{t-n}, \ldots, x_{t-1})$
- ~~Non-parametric Estimator~~ $\rightarrow$ Parametric Estimator

---

3. Update Internal Memory

$$\mathbf{h}_t = \phi\left(\mathbf{W}^\top \tilde{\mathbf{x}}_t + \mathbf{U}^\top \mathbf{h}_{t-1}\right),$$

where $\mathbf{U} \in \mathbb{R}^{n_h \times n_h}$ and $\mathbf{W} \in \mathbb{R}^{d \times n_h}$.



Language Modeling with Recurrent Neural Network

RNN Language Model

- ~~n-th order Markov assumption~~ $\rightarrow$ **Non-Markovian Model**
  $p(x_1, x_2, \ldots, x_T) \approx \prod_{t=1}^{T} p(x_t \mid x_1, \ldots, x_{t-n}, \ldots, x_{t-1})$
- ~~Non-parametric Estimator~~ $\rightarrow$ Parametric Estimator

---

5. Unnormalized Probabilities

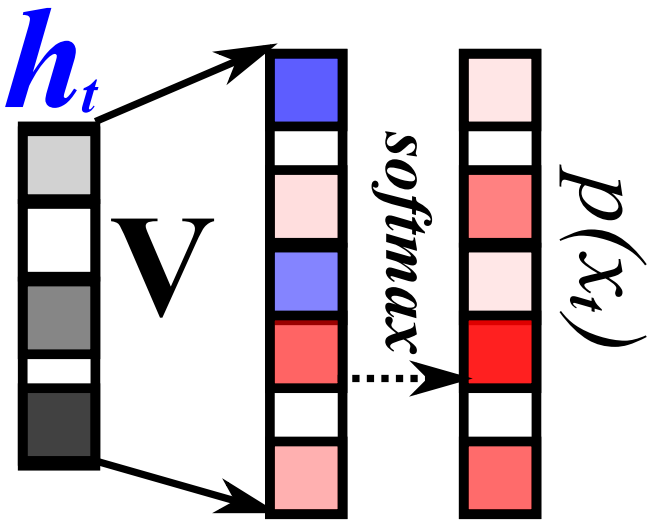
$$\mathbf{y}_t = \mathbf{V}^\top \mathbf{h}_t + \mathbf{c},$$

where $\mathbf{V} \in \mathbb{R}^{n_h \times \|V\|}$ and $\mathbf{c} \in \mathbb{R}^{\|V\|}$

6. Normalize for each word (*softmax*)

$$p(x_t = i \mid x_{<t}) = \frac{\exp\{y_i\}}{\sum_{j=1}^{\|V\|} \exp\{y_j\}}$$

where y_i is the i-th element of $\mathbf{y}$

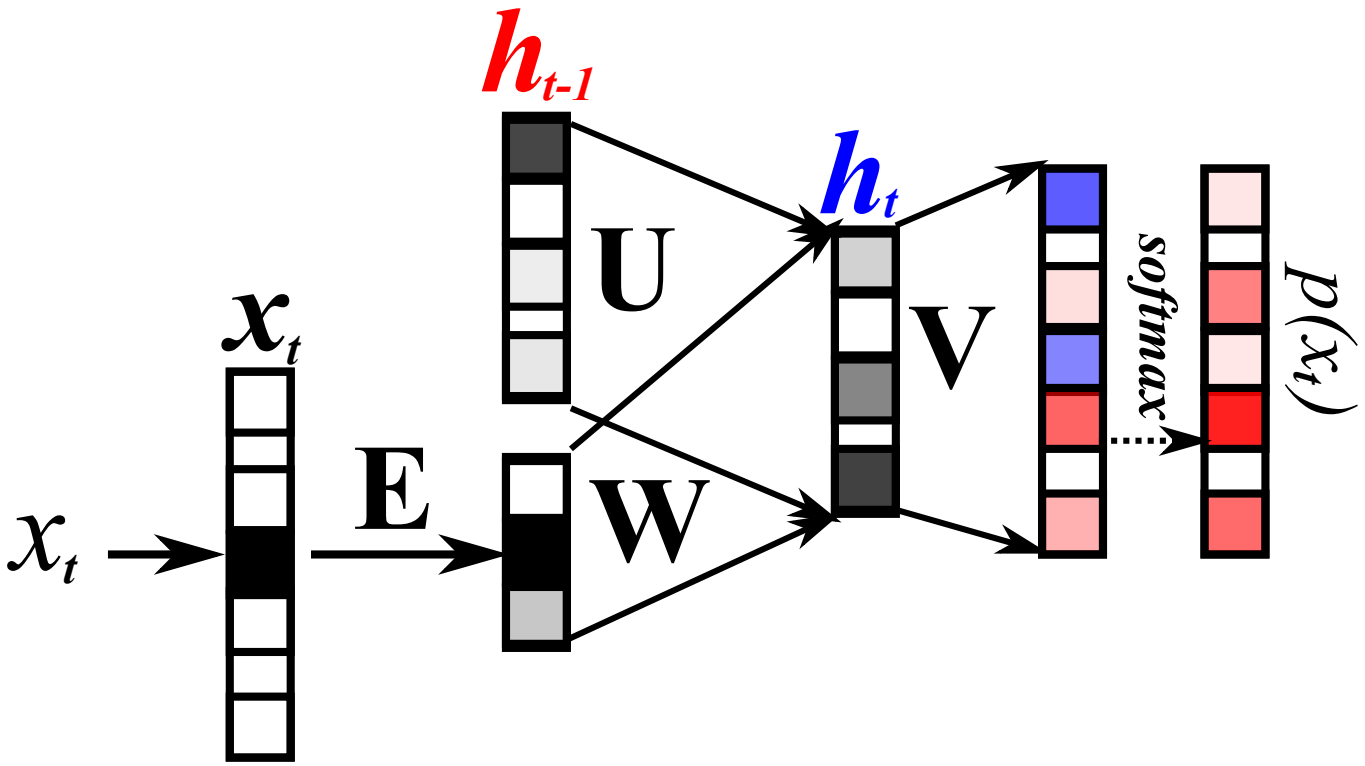


LM with RNN – Training

Maximum Likelihood Estimation *with Backprop Through Time*

$$\theta = \arg\max_{\theta} \frac{1}{N} \sum_{n=1}^{N} \sum_{t=1}^{T_n} \log p_\theta(x_t \mid x_{t-n}, \ldots, x_{t-1}),$$

where there are n training sentences and $\theta = \{\mathbf{W}, \mathbf{U}, \mathbf{b}, \mathbf{V}, \mathbf{c}\}$



Backpropagation Through Time and Vanishing Gradient

Maximum Likelihood Estimator: Next-step Prediction Maximization

$$\arg\max_{\boldsymbol{\theta}} \mathbb{E}_{x \in D} \left[\log p((x_1, \ldots, x_T) \mid \boldsymbol{\theta}) \right] \approx \arg\max_{\boldsymbol{\theta}} \frac{1}{N} \sum_{n=1}^{N} \sum_{t=1}^{T_n} \log p(x_t^n \mid x_{<t}^n)$$

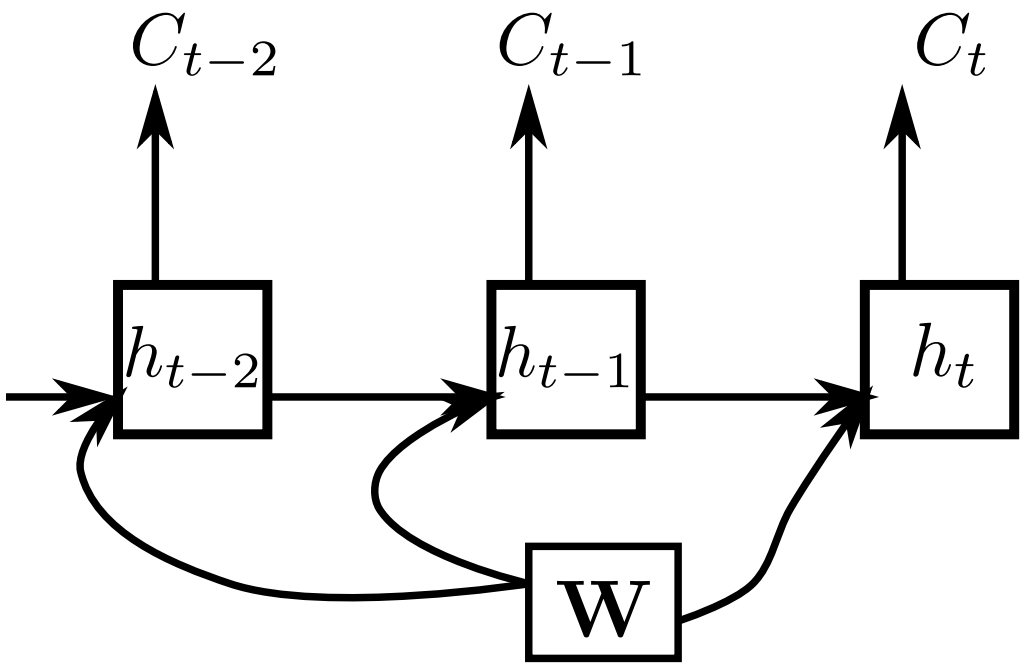
Stochastic gradient descent with backpropagation

Backpropagation Through Time and Vanishing Gradient

Maximum Likelihood Estimator: Next-step Prediction Maximization

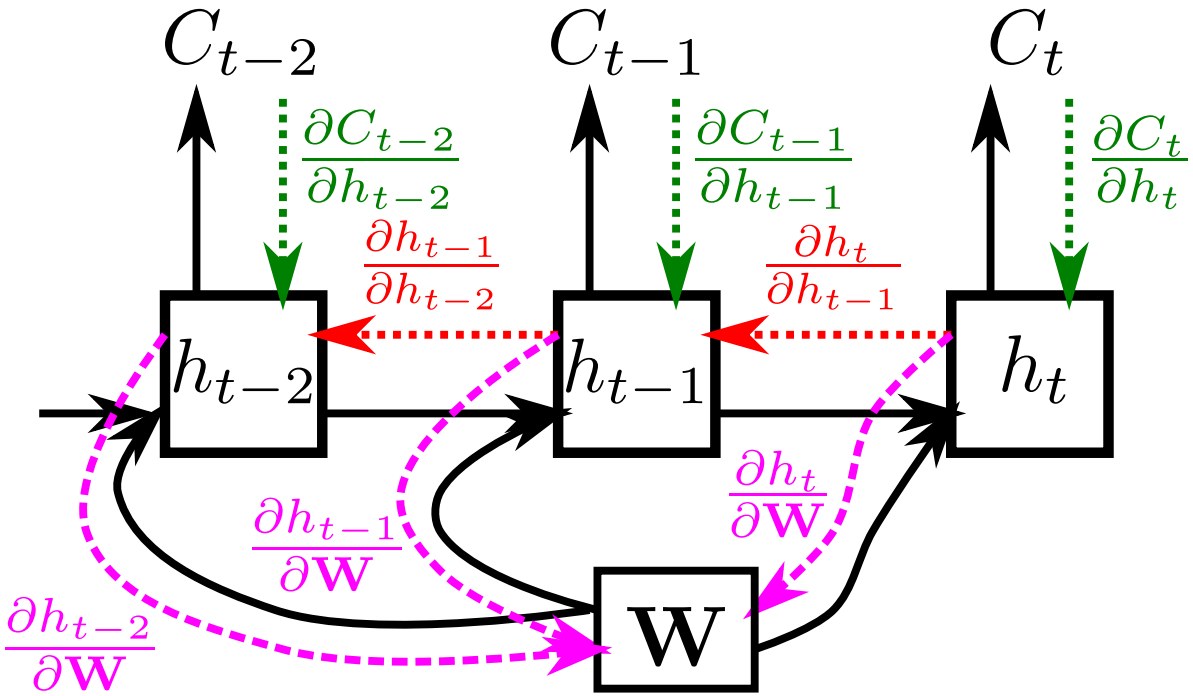
$$\arg\max_{\boldsymbol{\theta}} \mathbb{E}_{x \in D} \left[\log p((x_1, \ldots, x_T) \mid \boldsymbol{\theta})\right] \approx \arg\max_{\boldsymbol{\theta}} \frac{1}{N} \sum_{n=1}^{N} \sum_{t=1}^{T_n} \log p(x_t^n \mid x_{<t}^n)$$

Stochastic gradient descent with backpropagation



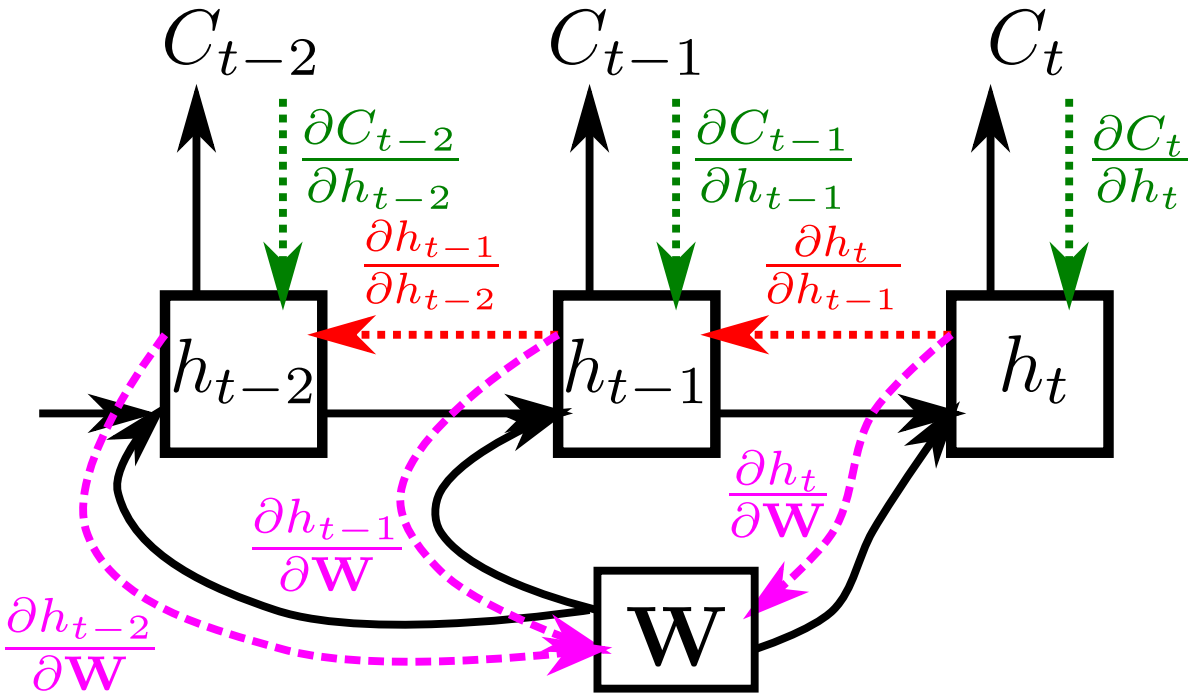
$$C_t = -\log p(x_t \mid x_{<t}).$$

Recurrent Neural Networks – Curse of Vanishing Gradient



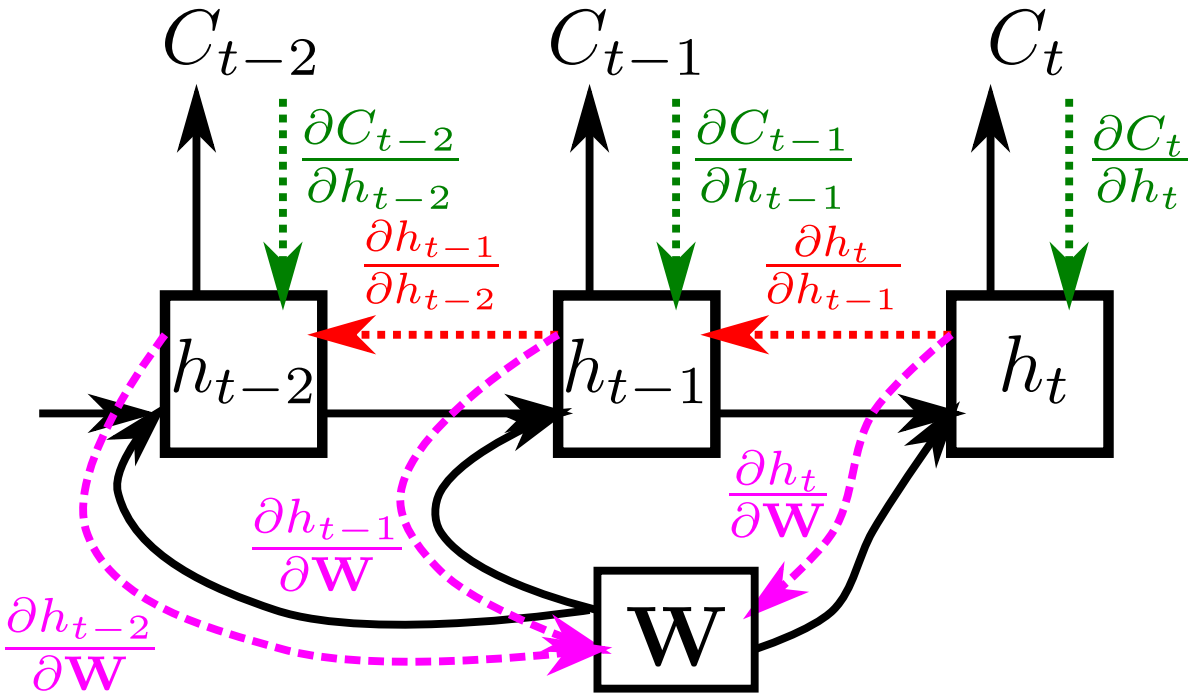
$$\frac{\partial C_t}{\partial \mathbf{W}} = \sum_{t'=1}^{t} \frac{\partial C_t}{\partial h_t} \frac{\partial h_t}{\partial h_{t'}} \frac{\partial h_{t'}}{\partial \mathbf{W}}, \text{ where } \frac{\partial h_t}{\partial h_{t'}} = \prod_{k=t'+1}^{t} \frac{\partial h_k}{\partial h_{k-1}}$$

Recurrent Neural Networks – Curse of Vanishing Gradient



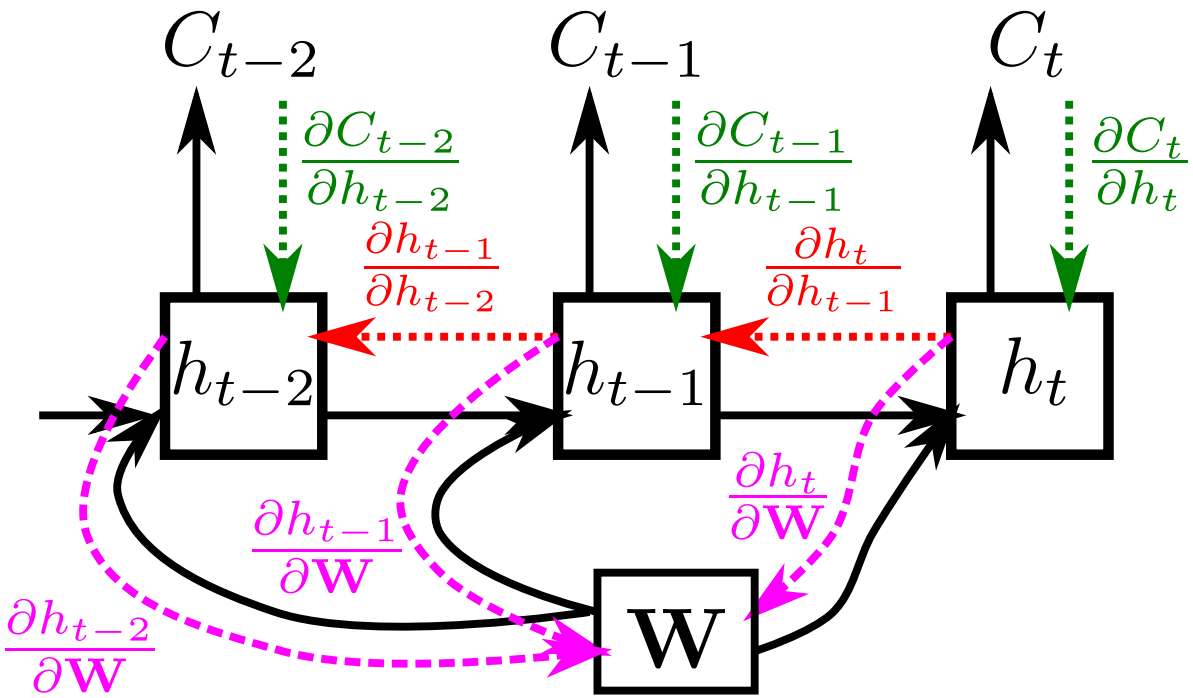
$$\frac{\partial h_t}{\partial h_{t'}} = \prod_{k=t'+1}^{t} \frac{\partial h_k}{\partial h_{k-1}}, \text{where } \frac{\partial h_k}{\partial h_{k-1}} = \mathbf{W}^\top \operatorname{diag}\left(\frac{\partial \phi(h_{k-1})}{\partial h_{k-1}}\right)$$

Recurrent Neural Networks – Curse of Vanishing Gradient



$$\left\| \frac{\partial h_t}{\partial h_{t'}} \right\| \leq \left\| \prod_{k=t'+1}^{t} \mathbf{W}^\top \right\|, \text{if } \left[\frac{\partial \phi(h_{k-1})}{\partial h_{k-1}} \right]_i \leq 1$$

Recurrent Neural Networks – Curse of Vanishing Gradient



Vanish or Explode, that is the problem!

$$\lim_{t\to\infty} \left\| \prod_{k=t'+1}^{t} \mathbf{W}^\top \right\| = \begin{cases} \infty & \text{if} \quad \lambda_{\max}(\mathbf{W}) > 1 \\ 0 & \text{if} \quad \lambda_{\max}(\mathbf{W}) < 1 \end{cases}$$

Is RNN doomed?

Long-Term Dependencies – (1)

RNN does *not* assume *Markov* property

Is it a good thing? – Yes! :)

1. Dependency across long time steps can be captured
   - Language modeling for German or Korean

Long-Term Dependencies – (2)

RNN does *not* assume *Markov* property

Does it make our lives difficult? – Yes! :(

1. Error vanishes over a chain of (bounded) nonlinearities
2. Naïve transition has difficulty in *remembering* things for long

$$\mathbf{h}_t = g\left(W\mathbf{x}_t + U\mathbf{h}_{t-1}\right),$$

Gated Recurrent Units – (1)

What can we do?

1. Let the unit carry the information without *update*
2. Let the unit *reset*

IF *it wants.*

Gated Recurrent Units – (2)

The new hidden state at time t

$$h_t = z_t \odot \tilde{h}_t + (1 - z_t) \odot h_{t-1},$$

where the candidate hidden state

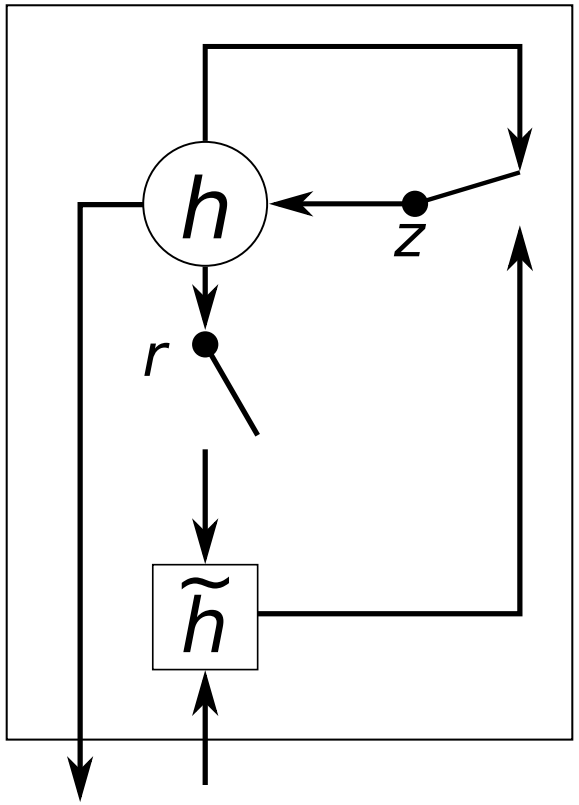
$$\tilde{h}_t = \phi\left(Wx_t + U\left(r_t \odot h_{t-1}\right)\right),$$

the *update* gate

$$z_t = \sigma\left(W_z x_t + U_z h_{t-1}\right),$$

and the *reset* gate

$$r_t = \sigma\left(W_r x_t + U_r h_{t-1}\right).$$



Both help, why?

Common feature

between Gated Recurrent Unit (GRU) and LSTM

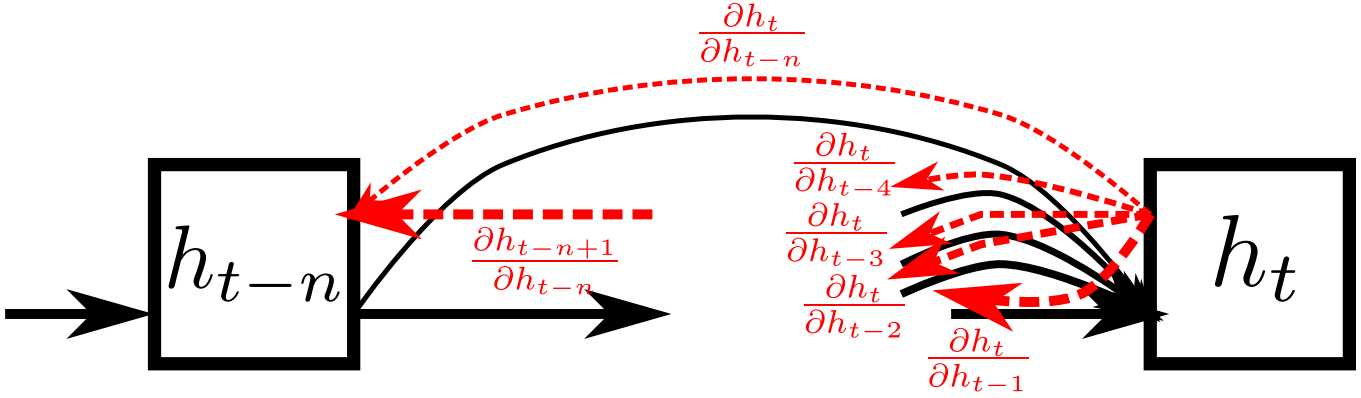
- Additive update: preserves the previous activation level

$$h_t = \tanh\left(Uh_{t-1} + Wx_t\,[+Vy_{t-1}] + b\right) \qquad \text{Overwrite}$$

vs.

$$h_t = z_t \odot \tilde{h}_t + (1 - z_t) \odot h_{t-1} \qquad \text{Addition}$$

Consequence: Multiple shortcuts across time



Some (*controversial*) tips on training recurrent neural nets

1. Use GRU or LSTM
2. Use adaptive learning rate algorithm (Adagrad, Adadelta, Adam, ..)
3. Initialize recurrent weight matrices to be orthogonal
4. With GRU or LSTM, momentum does not seem to matter too much
5. Clip the norm of the gradient
   - GRU/LSTM addresses vanishing gradient not exploding one
6. Use Theano: automatic differentiation!

And, with these, we can train large recurrent neural nets!

Beyond Sequence/Language Modelling

What can recurrent neural nets do?

1. Learn a probability distribution over a variable-length sequence
   - Score a sequence
   - Generate a sequence
2. Summarize a variable-length sequence

What can recurrent neural nets do?

1. Learn a probability distribution over a variable-length sequence
   - Score a sequence
   - Generate a sequence
2. Summarize a variable-length sequence

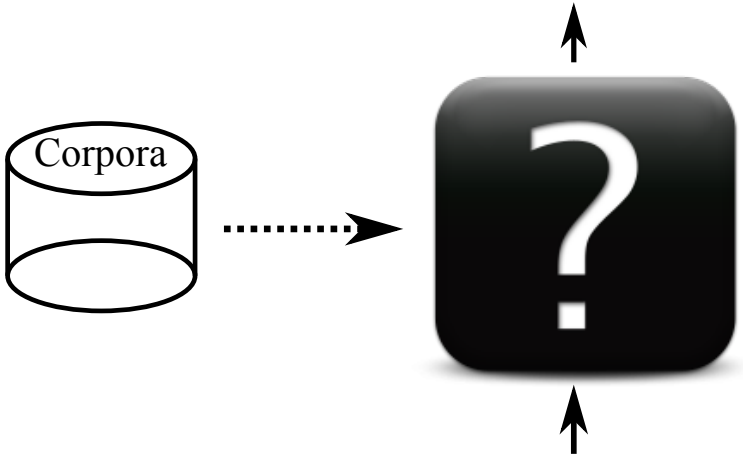
What would be an ultimate challenge with RNNs?

10-min Break

Neural Machine Translation

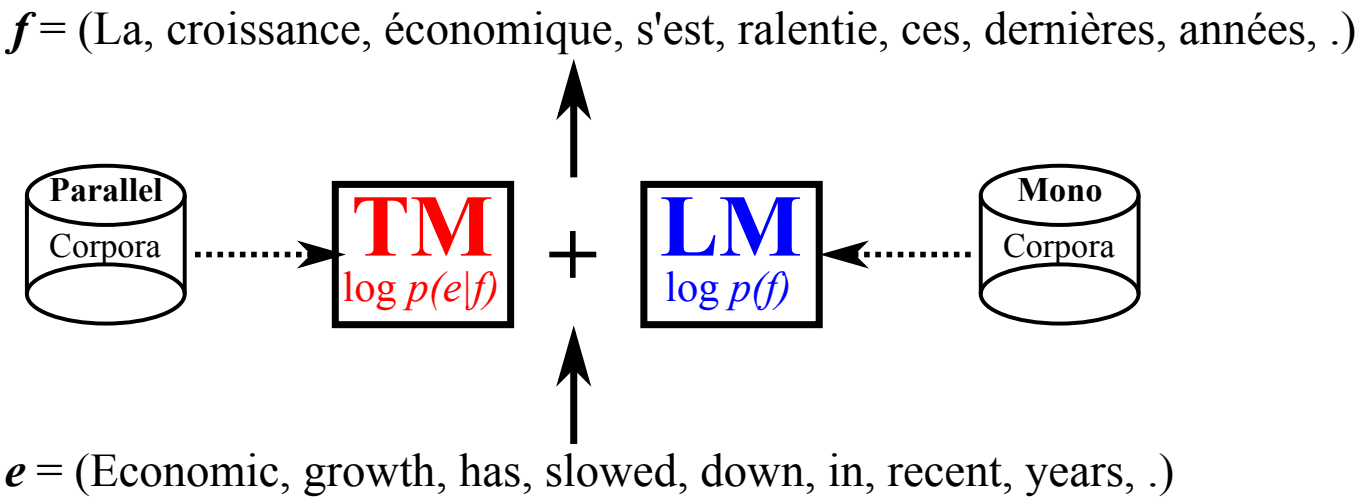
Statistical Machine Translation

$\boldsymbol{f}$ = (La, croissance, économique, s'est, ralentie, ces, dernières, années, .)



$\boldsymbol{e}$ = (Economic, growth, has, slowed, down, in, recent, years, .)

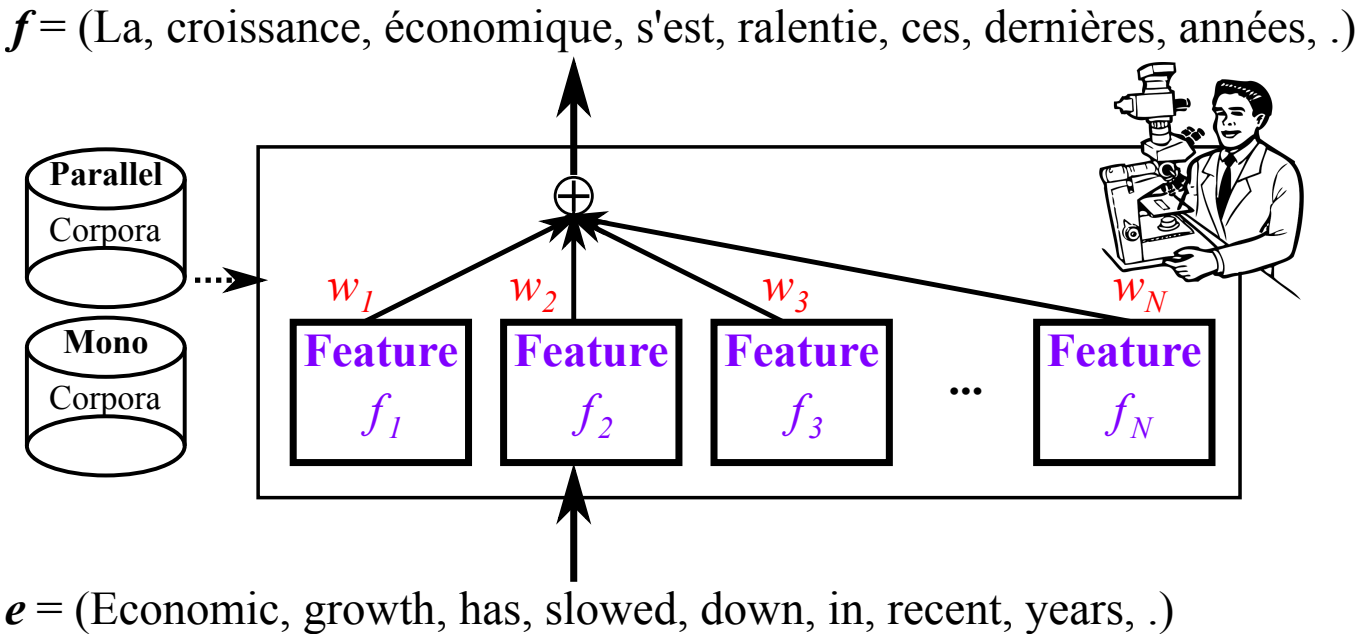
Statistical Machine Translation – Ideal World



$$\log p(f \mid e) = \log p(e \mid f) + \log p(f) + C$$

- $p(f \mid e)$: probability of a target sentence f given a source sentence e
- $p(e \mid f)$: *so-called* translation model
- $p(f)$: (target) language model
- C: constant

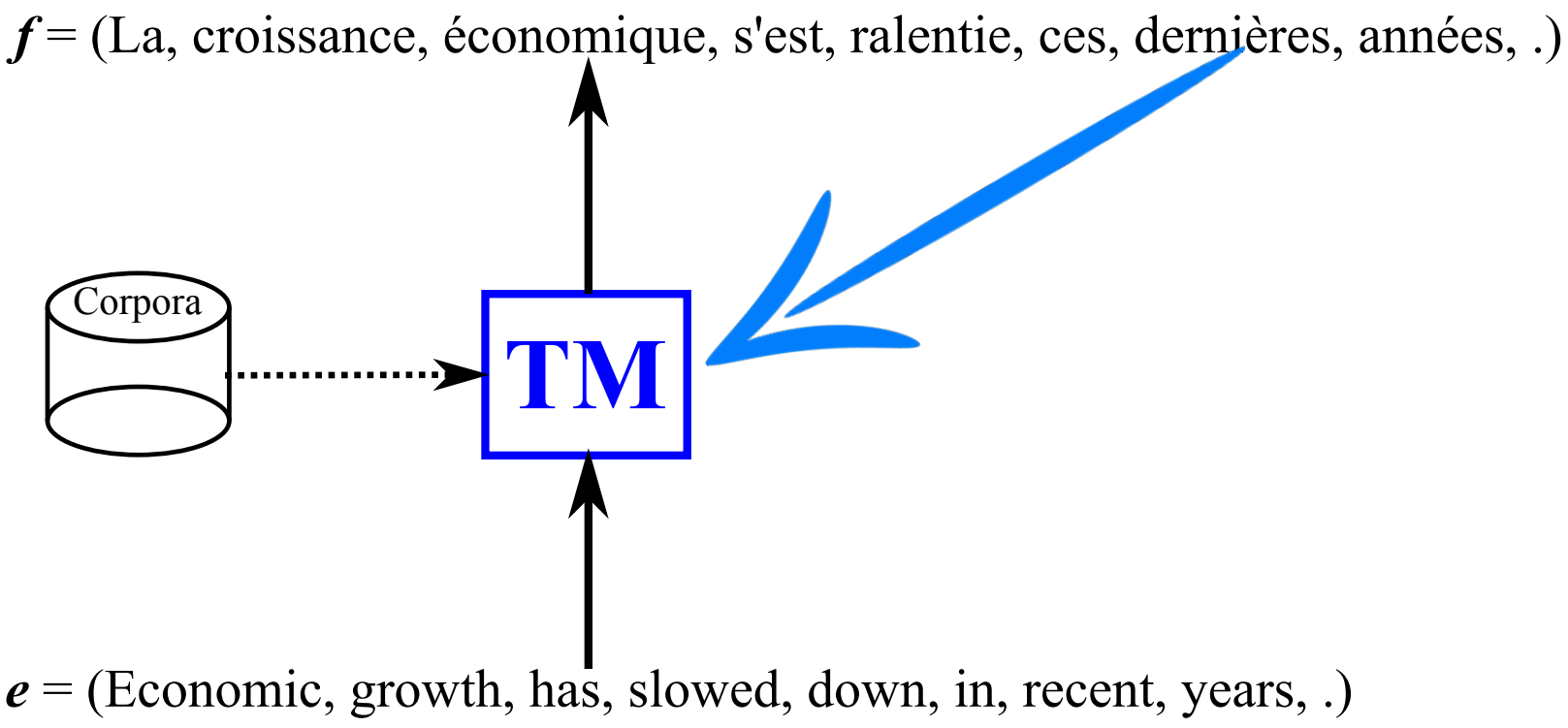
Statistical Machine Translation – Reality



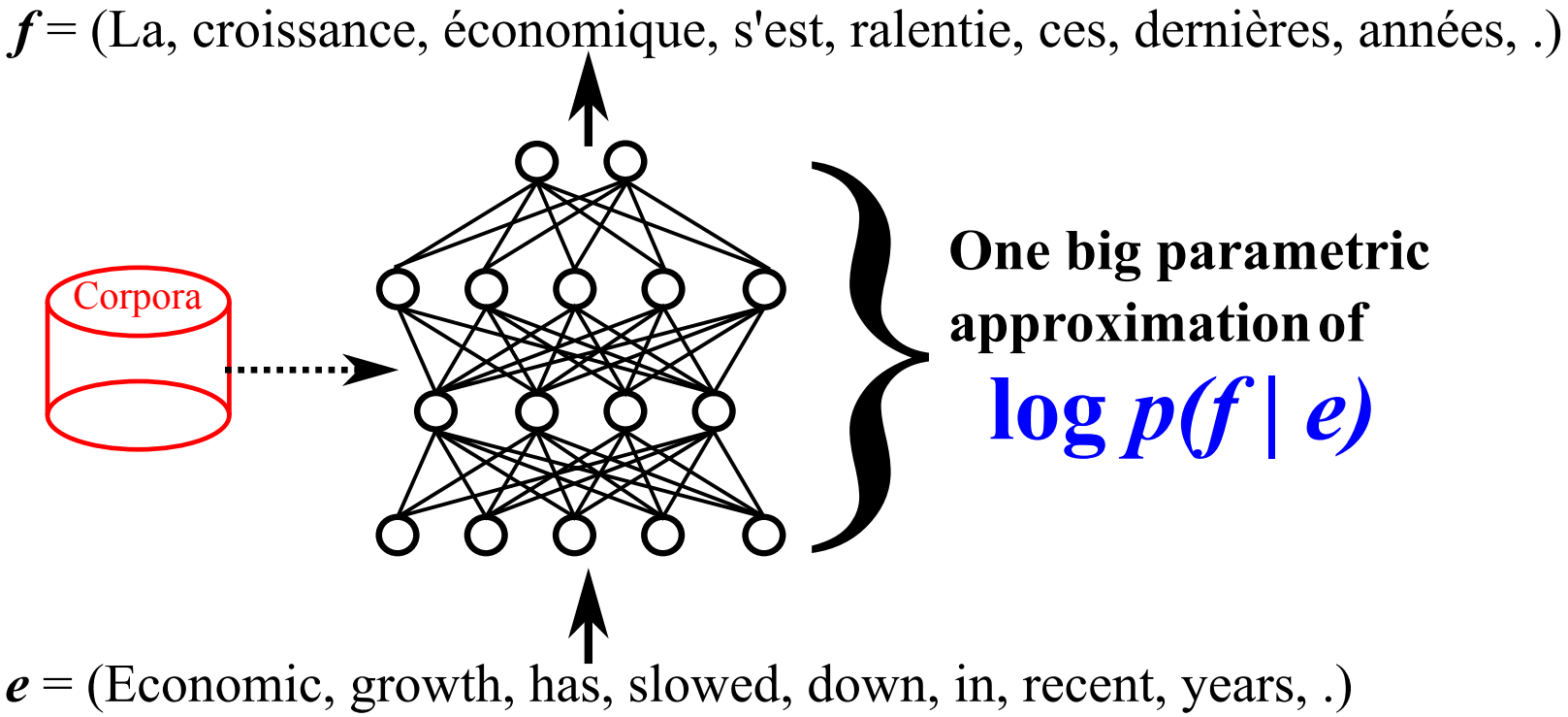
$$\log p(f \mid e) \approx \sum_i \omega_i f_i(e, f) + C$$

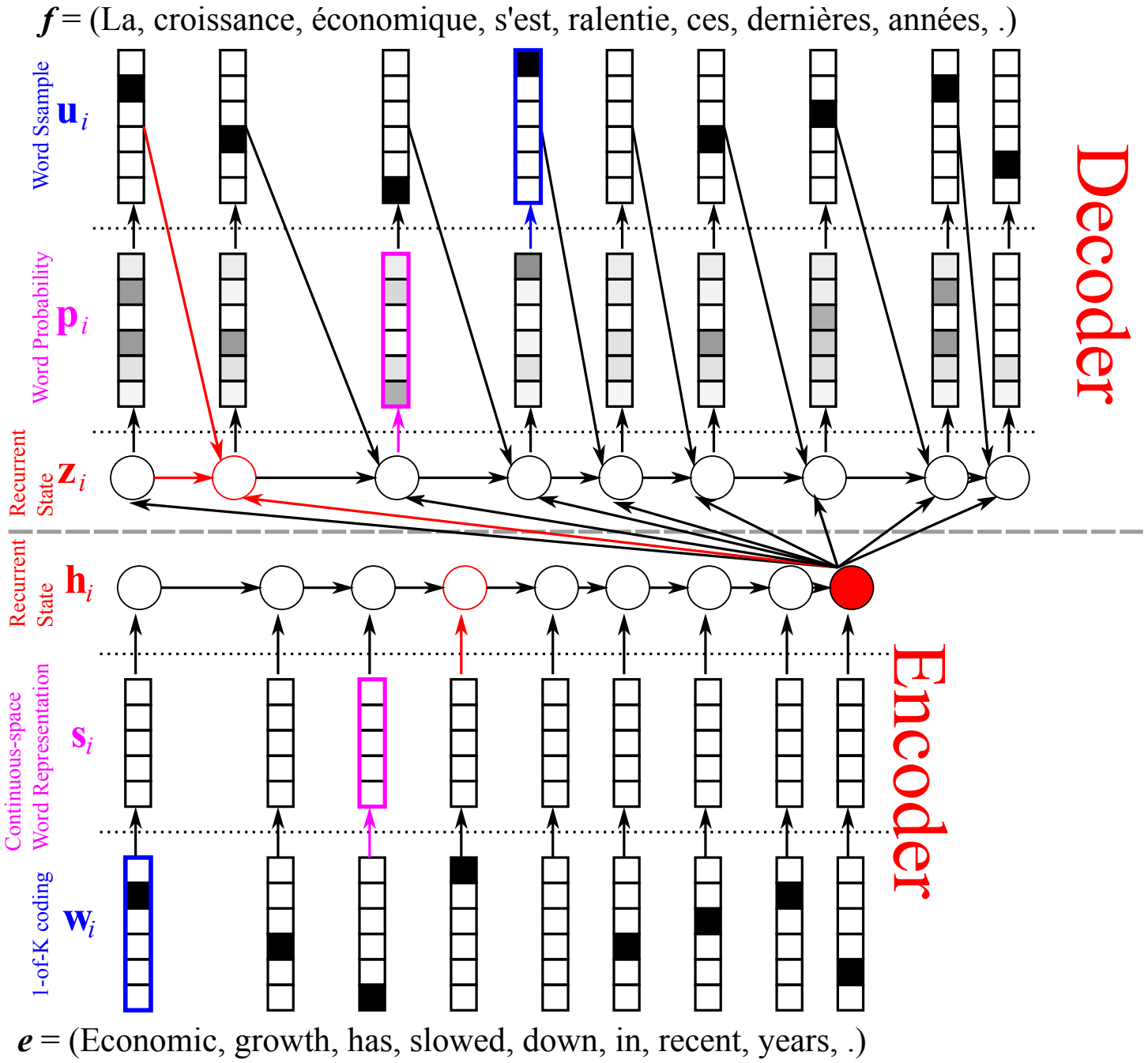
1. *Experts* engineer useful features
2. Use a simple model (log-linear model) from machine learning
3. (Maximize another metric, e.g., BLEU, w.r.t. ω)
4. (Use a strong language model)

Why don't we model the whole **TM** directly?



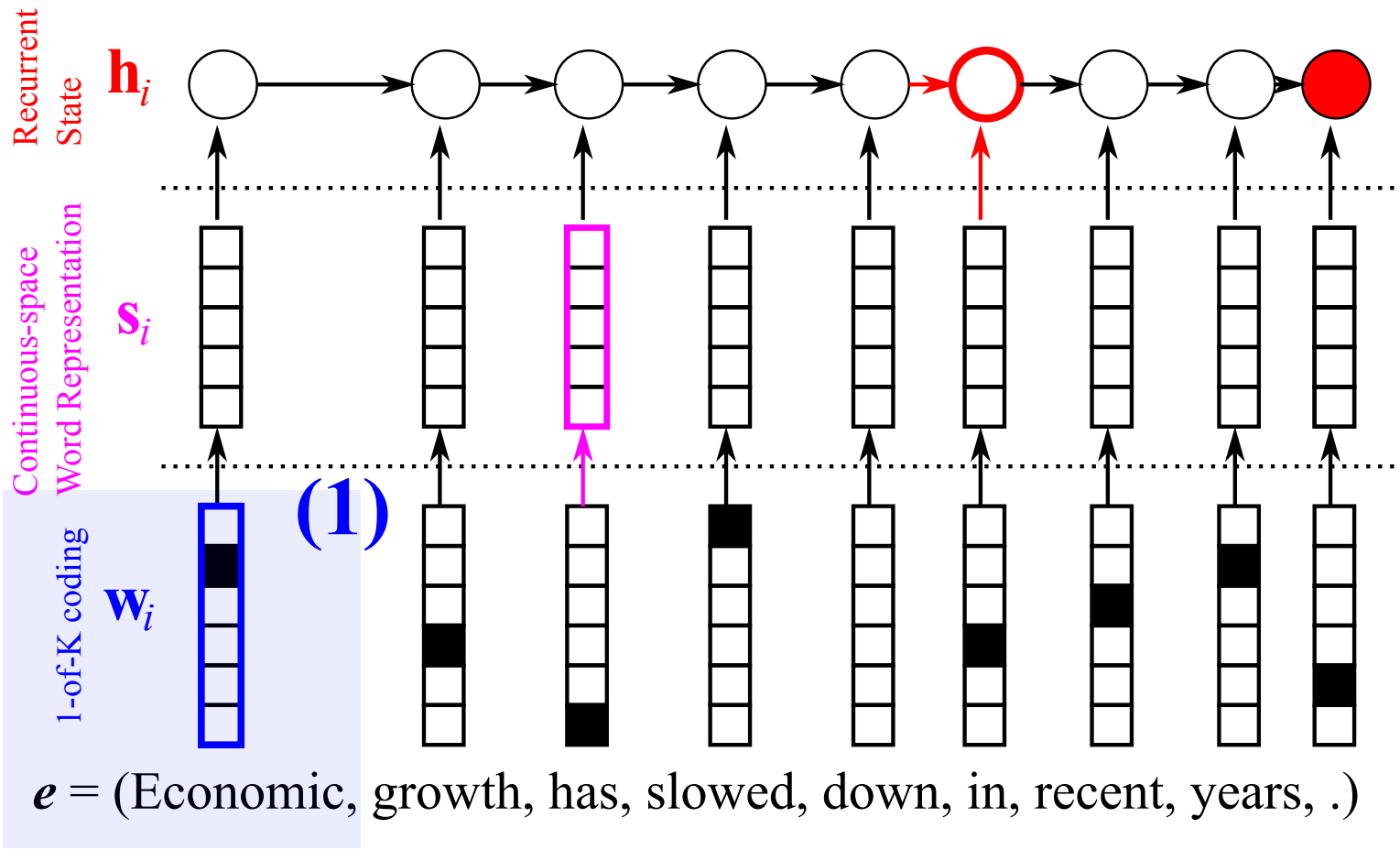
How about . . .


f = (La, croissance, économique, s'est, ralentie, ces, dernières, années, .)
Corpora
One big parametric approximation of
log p(f | e)
e = (Economic, growth, has, slowed, down, in, recent, years, .)



(Forcada&Ñeco, 1997, Kalchbrenner&Blunsom, 2013; Cho et al., 2014; Sutskever

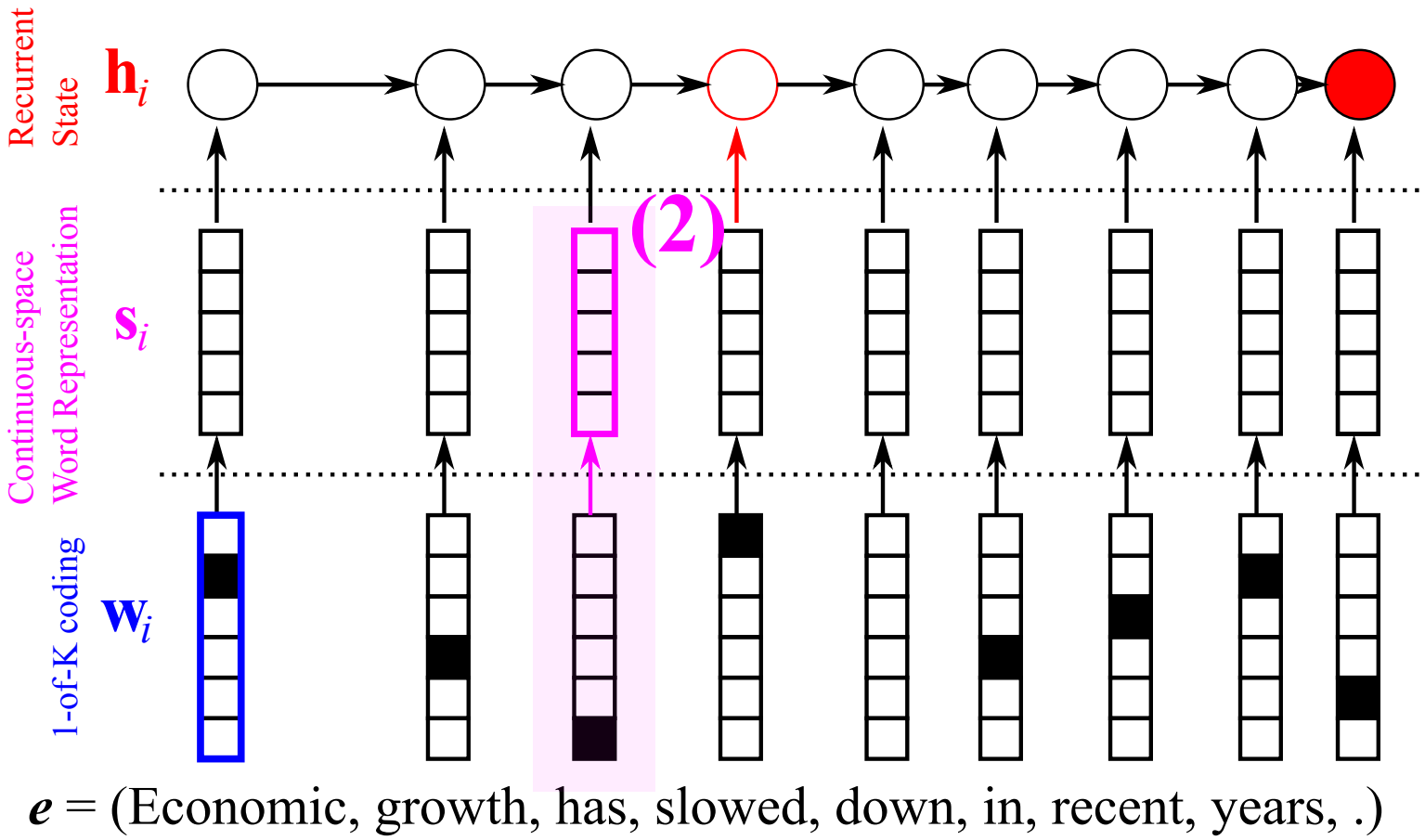
Neural Machine Translation – Encoder (1)



(1) Encode each word as a 1-of-K coded vector $\mathbf{w}_i$

(Cho et al., 2014; Sutskever et al., 2014)

Neural Machine Translation – Encoder (2)

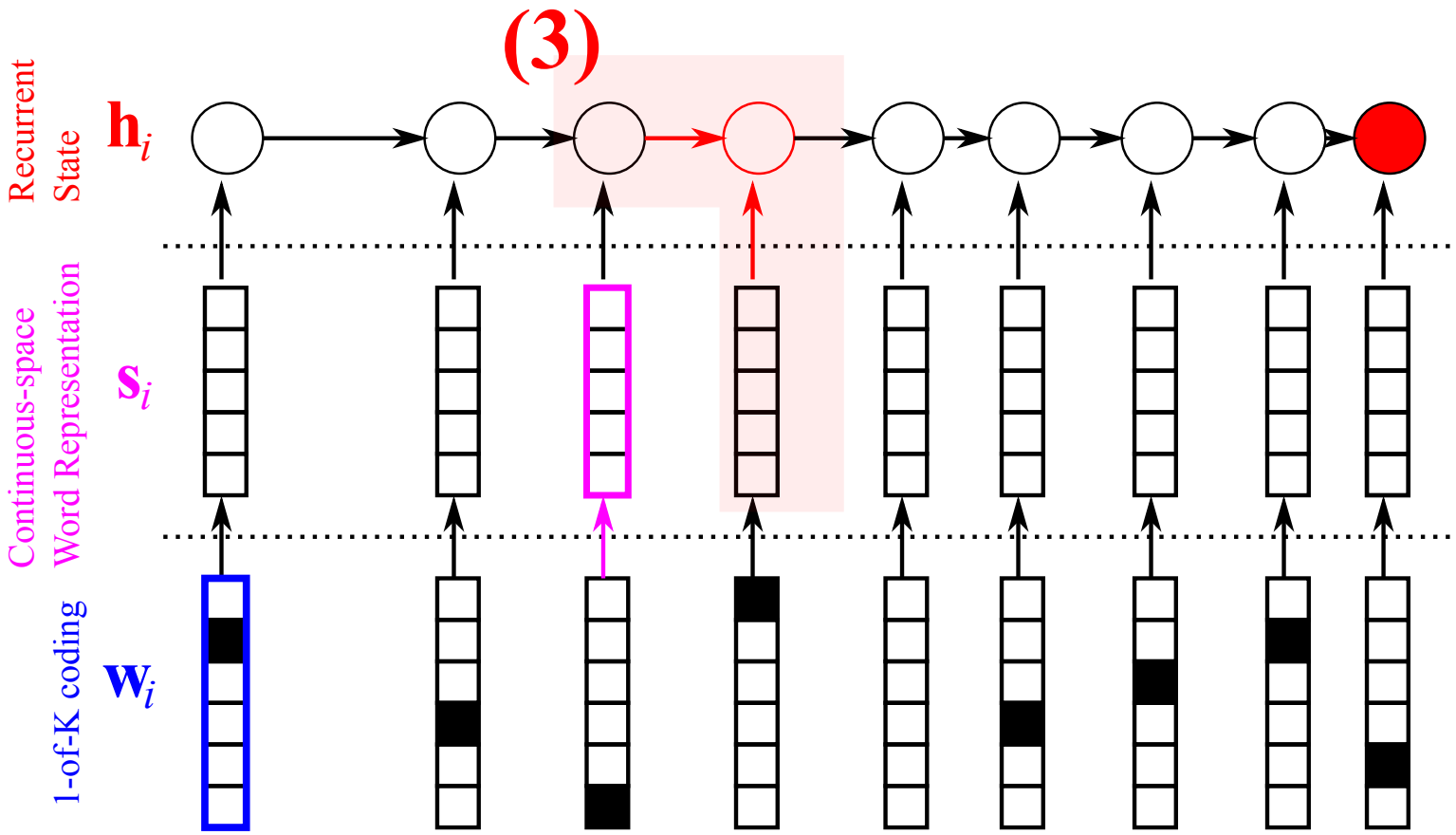


(2) Project $\mathbf{w}_i$ into a continuous space: $\mathbf{s}_i = \mathbf{U}\mathbf{w}_i \in \mathbb{R}^d$

- *Preserves the underlying meaning of a word*

(Cho et al., 2014; Sutskever et al., 2014)

Neural Machine Translation – Encoder (3)



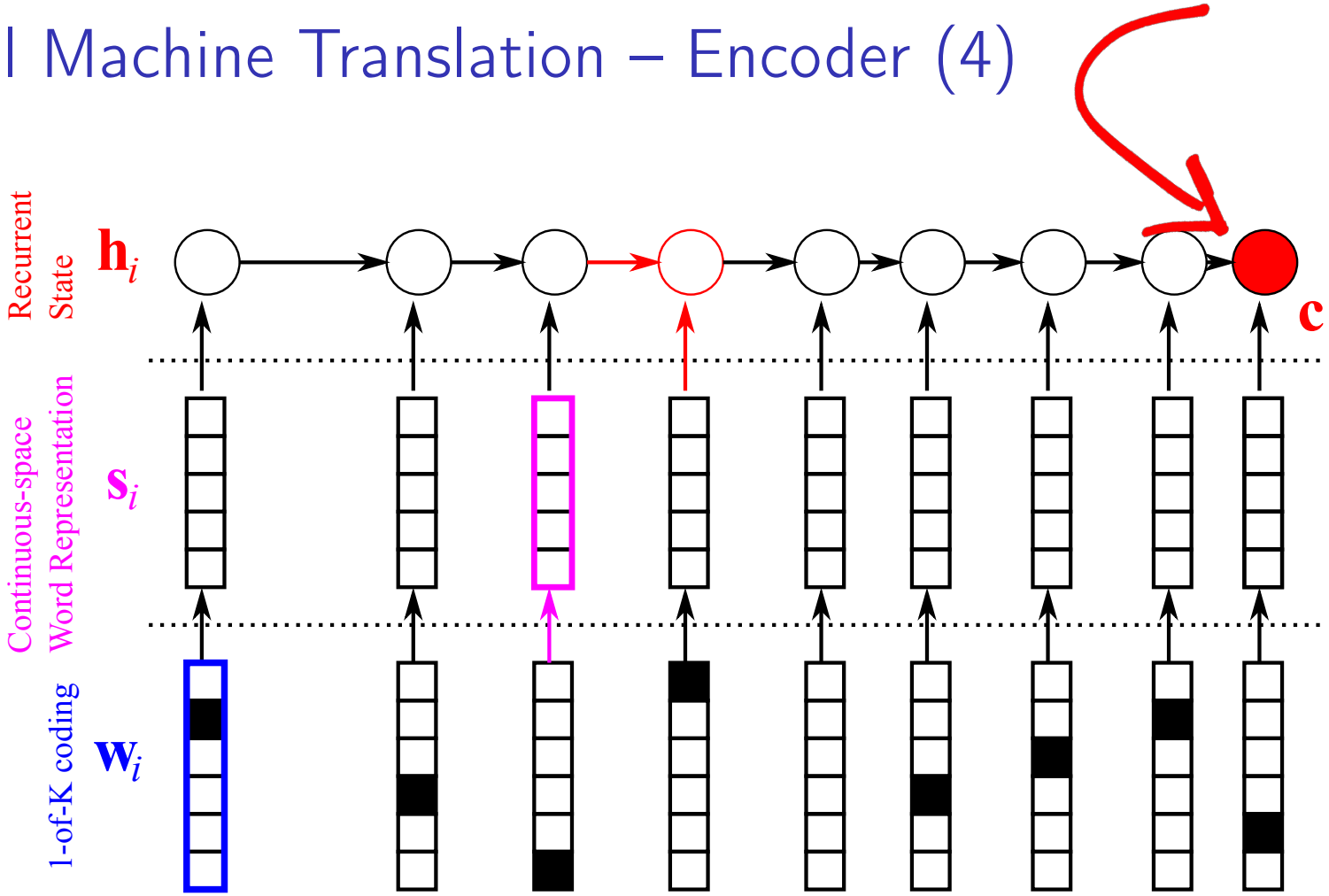
$\boldsymbol{e}$ = (Economic, growth, has, slowed, down, in, recent, years, .)

(3) Recursively update the recurrent hidden state:

$$\mathbf{h}_i = f(\mathbf{h}_{i-1}, \mathbf{s}_i)$$

(Cho et al., 2014; Sutskever et al., 2014)

Neural Machine Translation – Encoder (4)

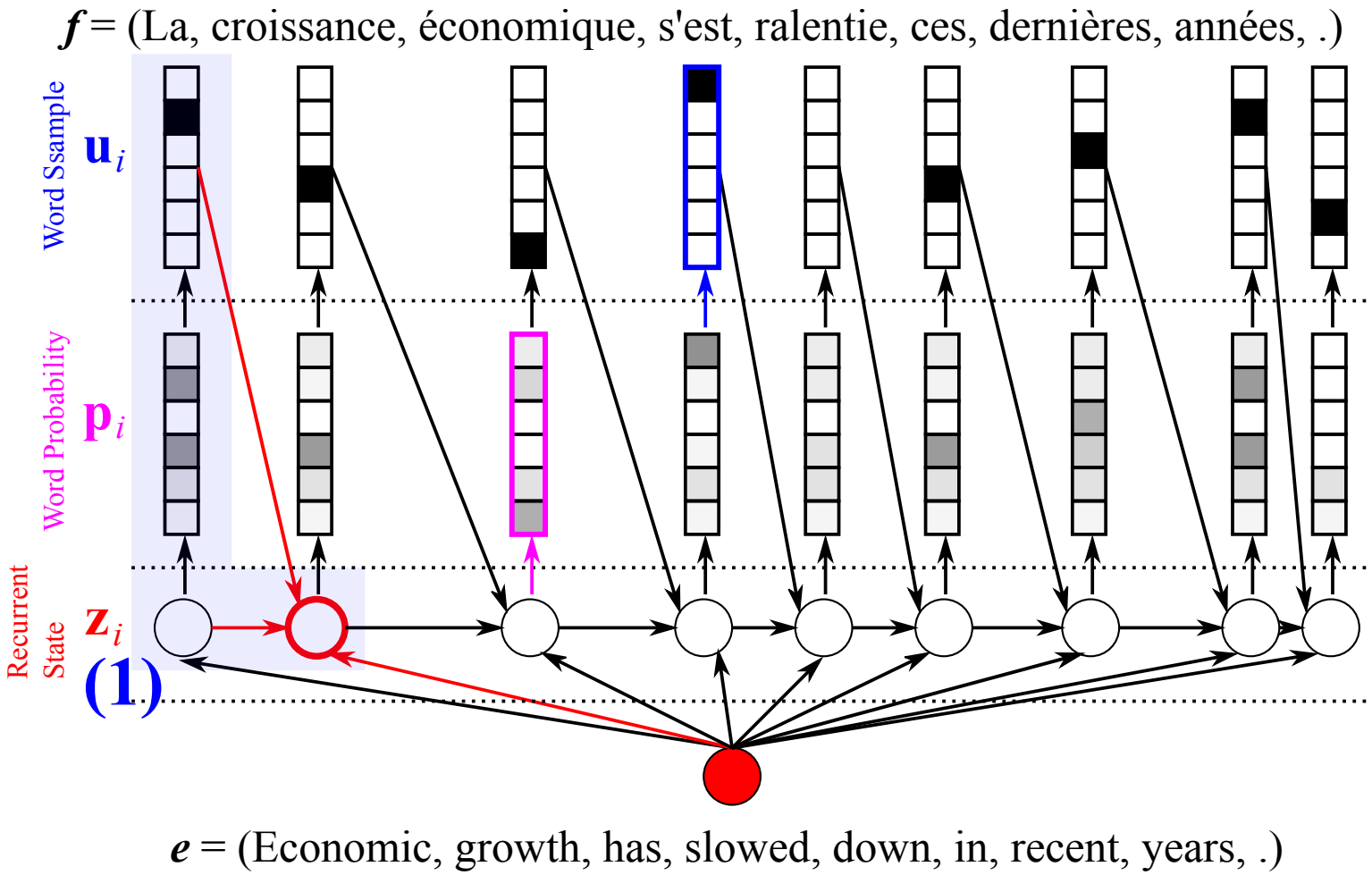


$\boldsymbol{e}$ = (Economic, growth, has, slowed, down, in, recent, years, .)

- $\mathbf{c} = \mathbf{h}_T$: A fixed-size representation of a variable-length sentence

(Cho et al., 2014; Sutskever et al., 2014)

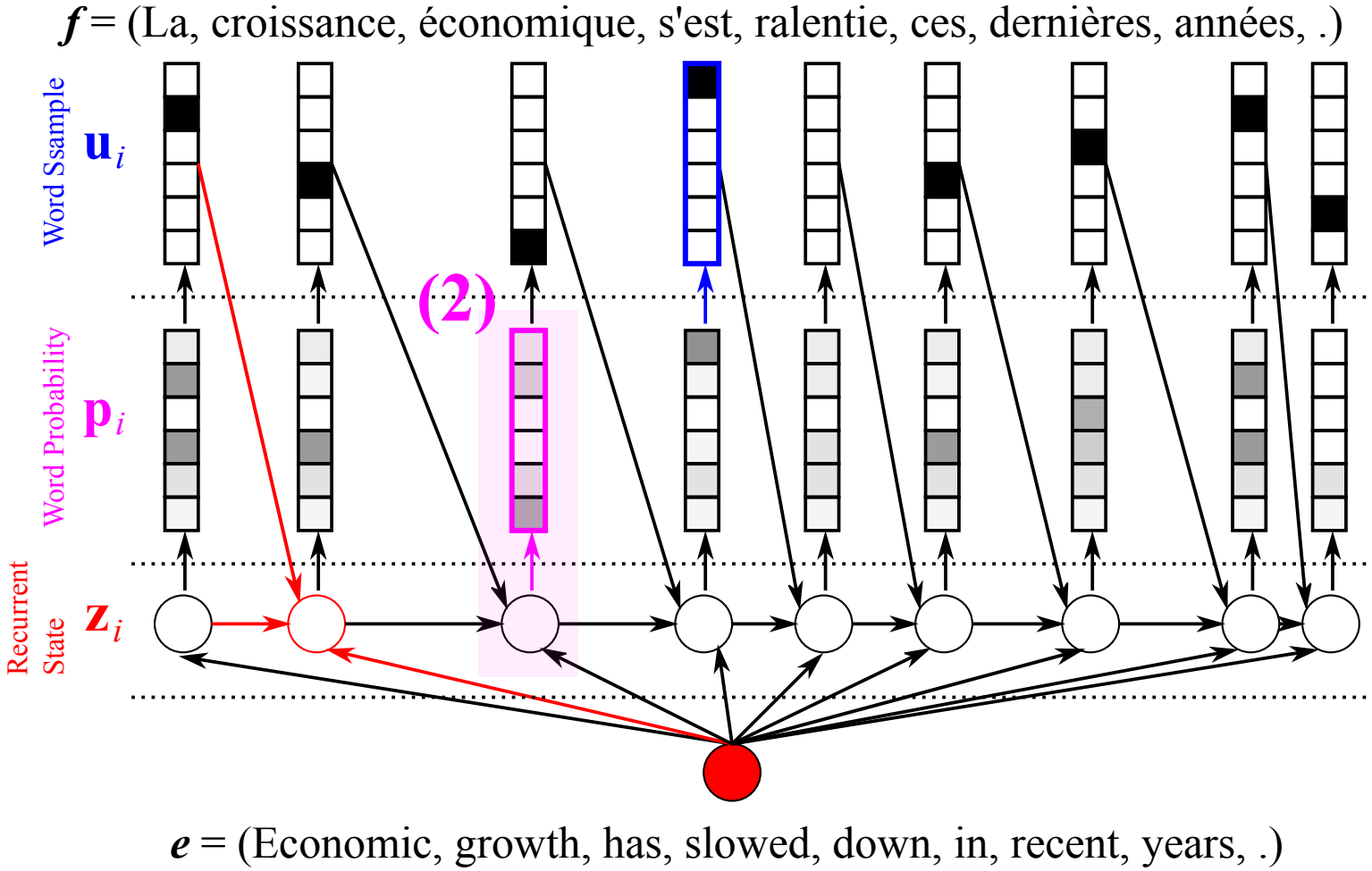
Neural Machine Translation – Decoder (1)



1 Update the recurrent hidden state: $\mathbf{z}_i = f(\mathbf{z}_{i-1}, \mathbf{u}_{i-1}, \mathbf{c})$

(Cho et al., 2014; Sutskever et al., 2014)

Neural Machine Translation – Decoder (2)

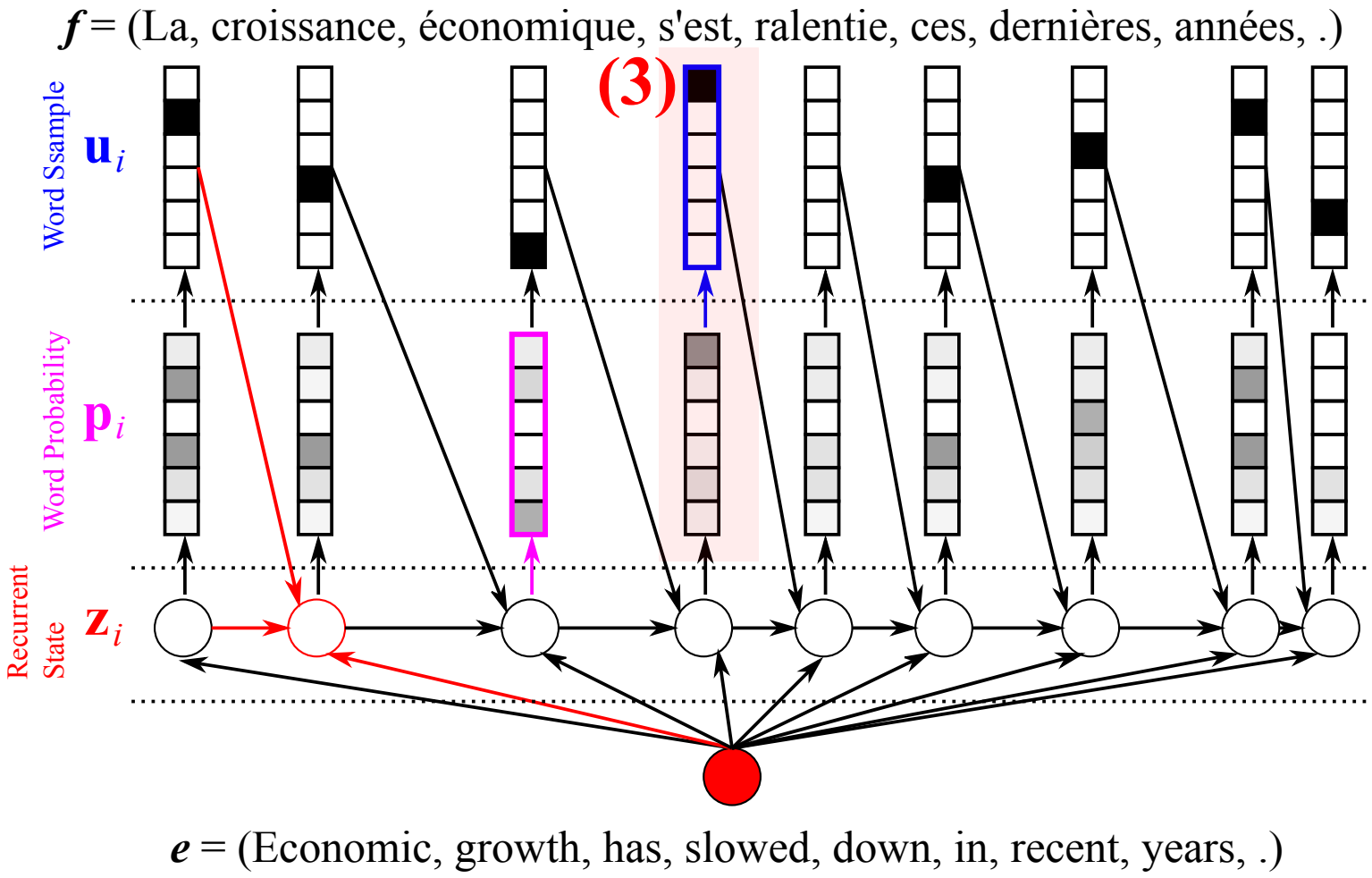


2 Compute the next word probability:

$$p(u_i = k \mid \mathbf{u}_1, \ldots \mathbf{u}_{i-1}) = \frac{1}{Z} \exp\left(\mathbf{p}_k^\top \mathbf{z}_i\right)$$

(Cho et al., 2014; Sutskever et al., 2014)

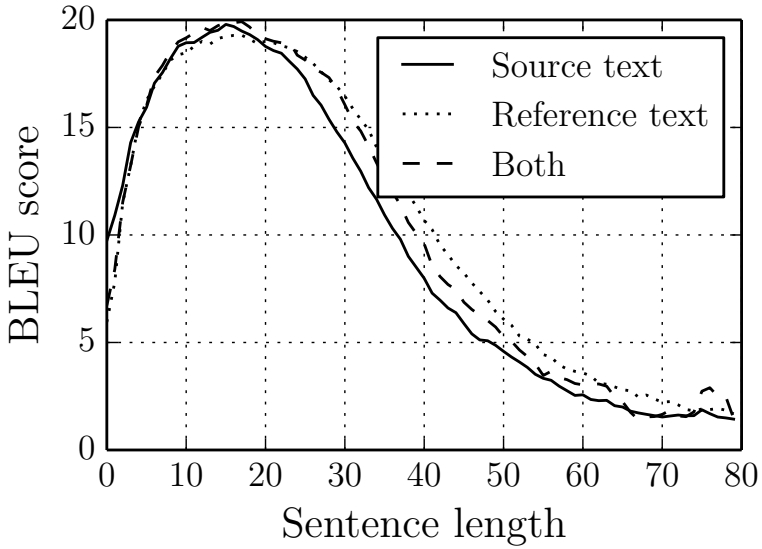
Neural Machine Translation – Decoder (3)

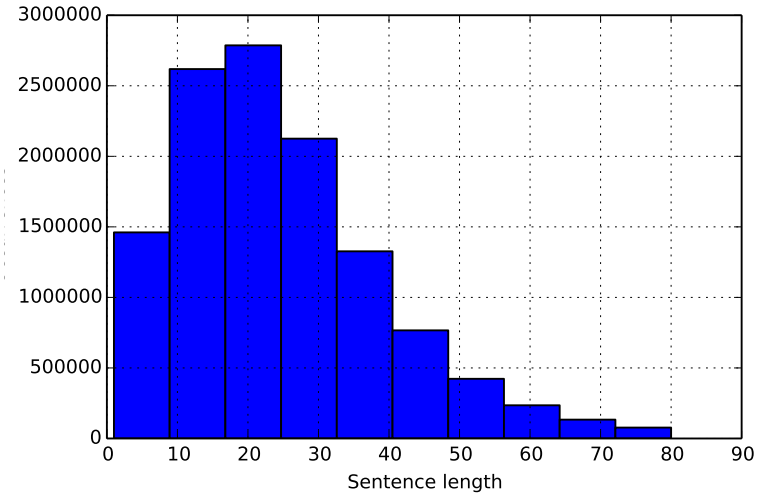


3 Sample the next word $\mathbf{u}_i$

(Cho et al., 2014; Sutskever et al., 2014)

But, it didn't work well for us – Curse of Sentence Length



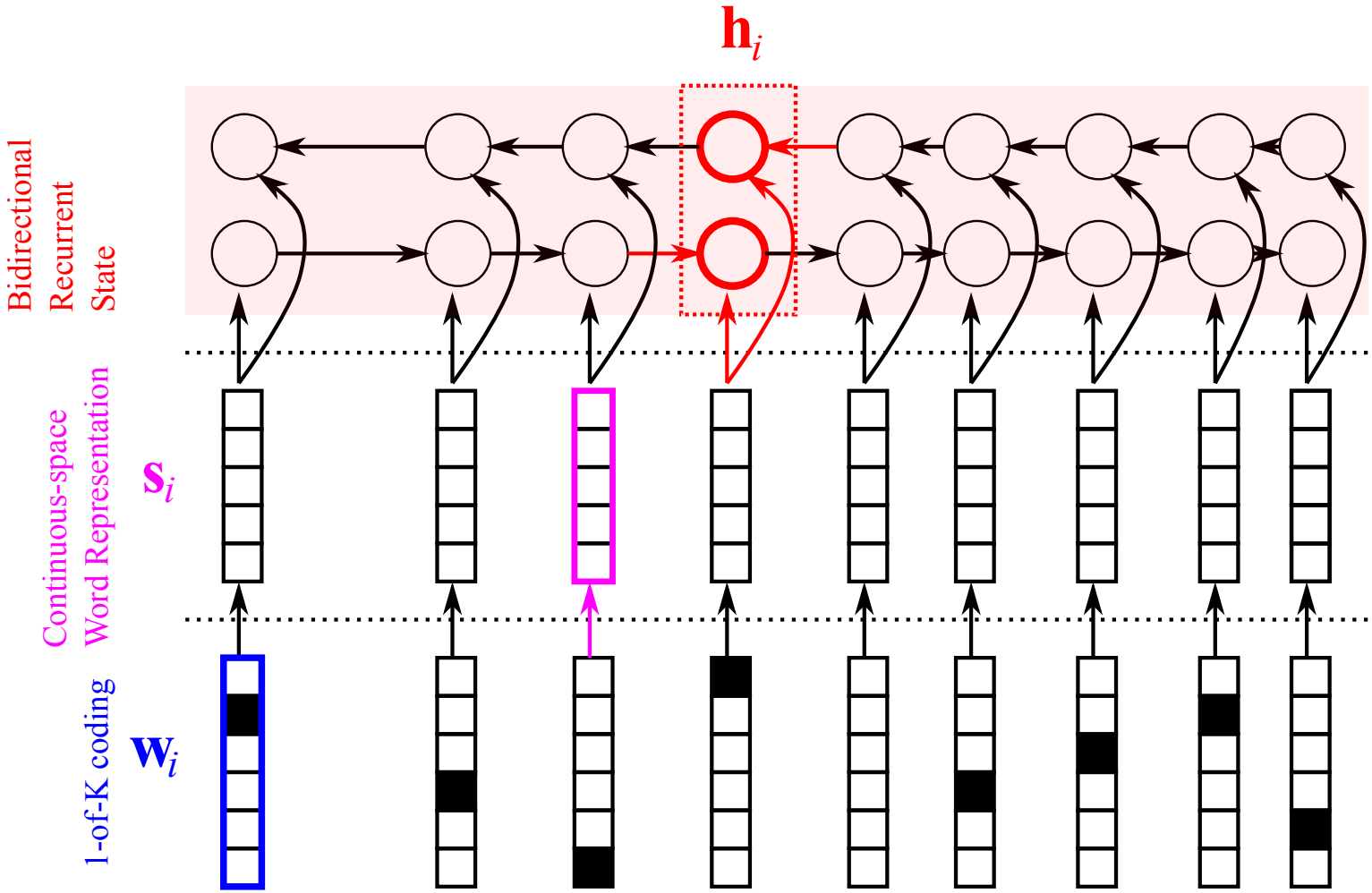


Potential Reasons:

1. **Difficult to encode a long sentence into a fixed-length vector**
2. ~~Only few long sentences in the training corpus~~
3. ~~Difficult to generate a coherent long sentence~~

Cho et al. (2014a,b), Pouget-Abadie et al. (2014)

Neural Machine Translation with Attention – Encoder

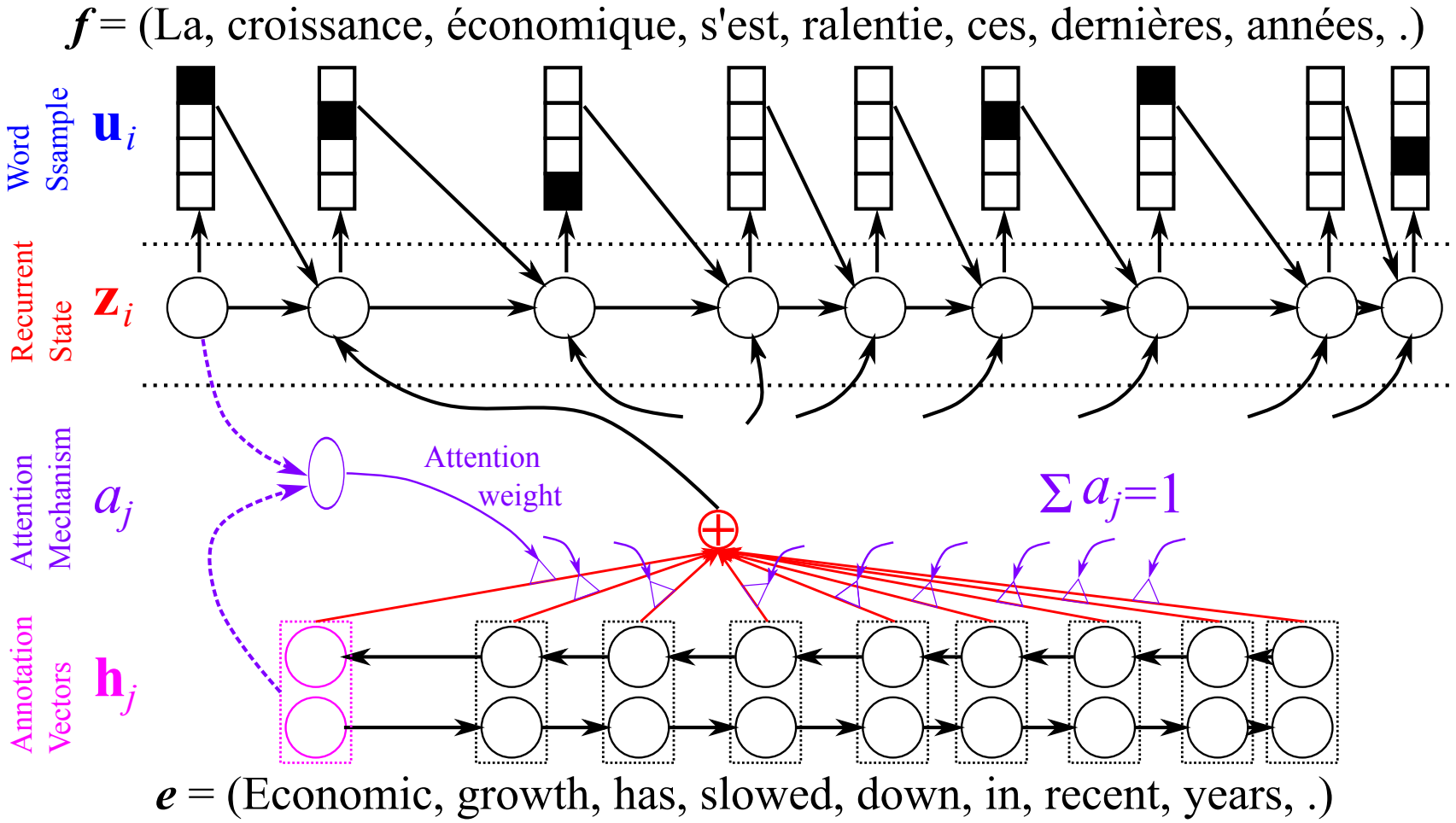


$\boldsymbol{e}$ = (Economic, growth, has, slowed, down, in, recent, years, .)

- Bi-directional recurrent neural network

(Bahdanau et al., 2014)

Neural Machine Translation with Attention – Decoder



At each timestep in the decoder:

1. Computes a *relevance* score of each annotation
2. Use the weighted sum of the annotations as a *context*

(Bahdanau et al., 2014)

(a) **English→French**

	NMT(A)	Google	P-SMT
NMT	32.68	30.6*	**37.03$^{\bullet}$**
+Cand	33.28	–	
+UNK	33.99	32.7°	
+Ens	**36.71**	**36.9°**	

(b) **English→German**

NMT(A)	P-SMT
16.95	**20.67$^{\diamond}$**
17.51	
18.87	
20.98	

(c) **Chinese→English$^{\odot}$ (SMS/Chat)**

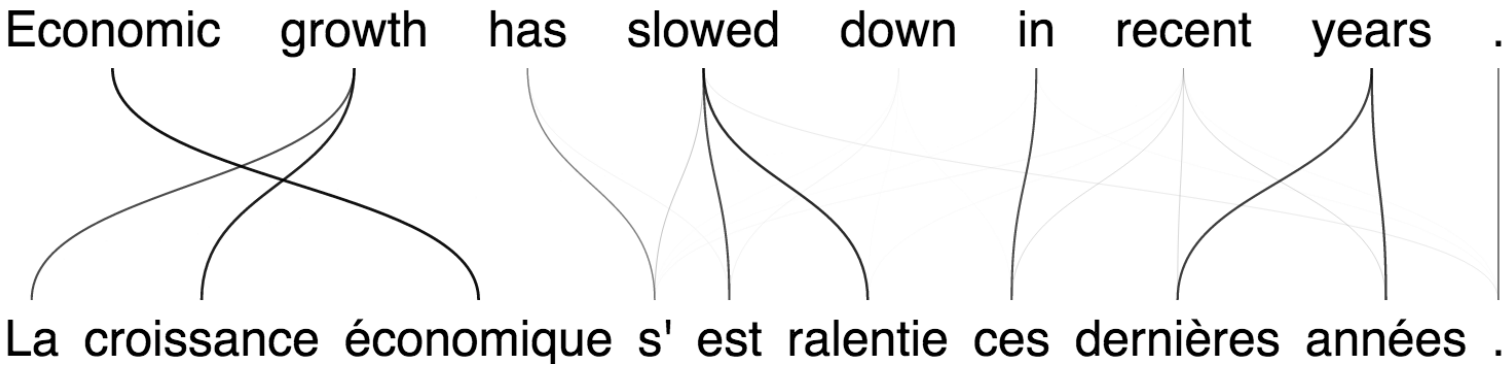
	NMT(A)	P-SMT	HP-SMT
NMT	17.36	14.73	14.71
+LM^{Sh}	–	**15.25**	**15.8**
+LM^{De}	**17.64**	–	–

(d) **Turkish→English$^{\odot}$ (TED Subtitles, tst2013)**

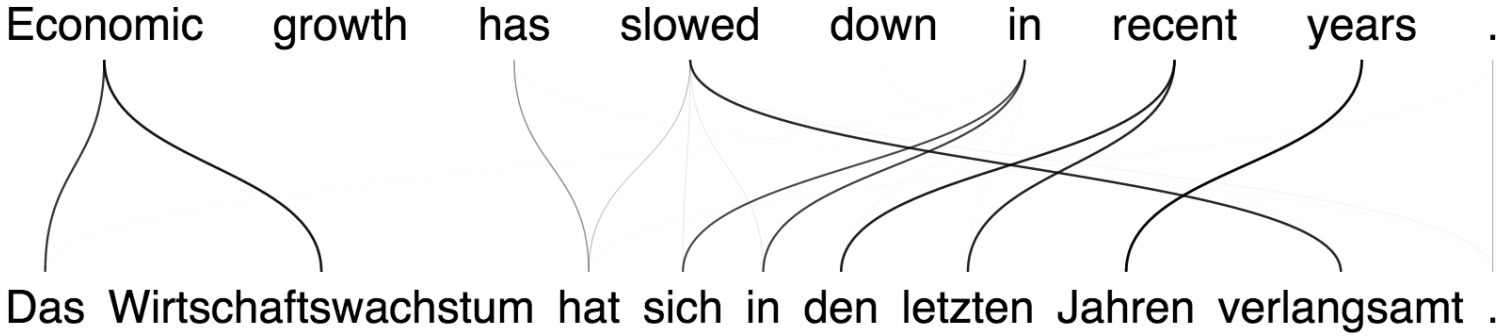
NMT(A)	P-SMT
20.13	**18.88**
–	–
21.34	–

NMT(A): **(Bahdanau et al., 2014; Jean et al., 2014)**,
($\star$): (Sutskever et al., 2014), ($\circ$): (Luong et al., 2014),
($\bullet$): (Durrani et al., 2014), ($\diamond$): (Buck et al., 2014),
($\odot$): **(Gulcehre et al., Under review)**

English-French



English-German

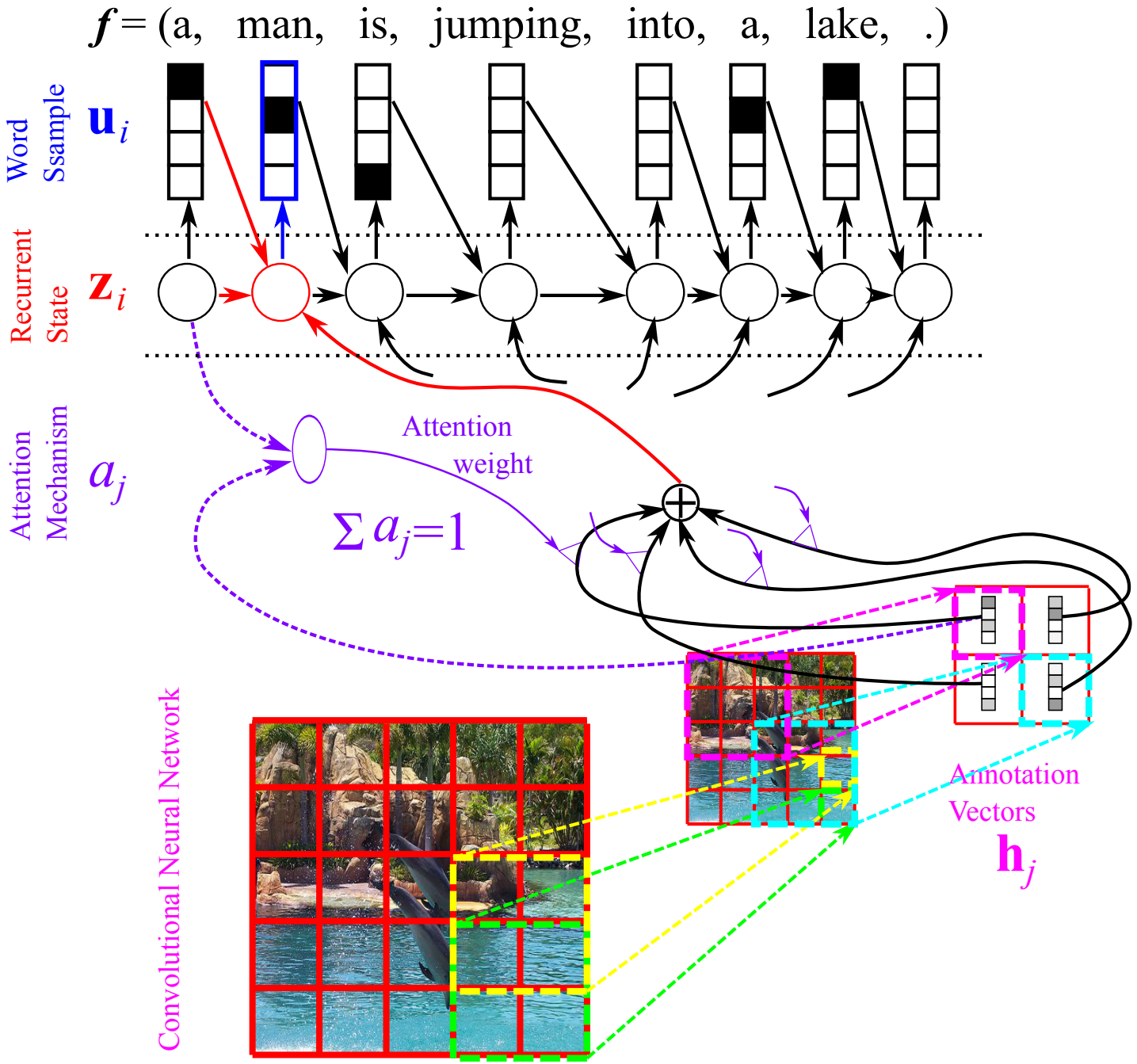


I confess that I speak neither French nor German. Sorry!

Demo at

HTTP://LISA.IRO.UMONTREAL.CA/MT-DEMO

Beyond Natural Languages



(Xu et al., 2015), (Yao et al., 2015)

A(0.98)

zebra(0.23)

standing(0.20)

in(0.14)

a(0.17)

field(0.24)

of(0.24)

tall(0.19)

grass(0.22)

.(0.18)

A(0.97)

stop(0.36)

sign(0.19)

is(0.22)

on(0.25)

a(0.21)

road(0.26)

with(0.28)

a(0.30)

mountain(0.44)

in(0.37)

the(0.21)

background(0.11)

.(0.13)

More on neural nets for natural language processing..

CS224d: Deep Learning for Natural Language Processing @ Stanford

http://cs224d.stanford.edu/