Natural Language Understanding

Kyunghyun Cho, NYU & U. Montreal

Language Understanding?
Modelling?

LANGUAGE UNDERSTANDING

Topics: Natural Language Understanding

- What does it mean that a machine *understands natural languages*?
- Should we start reading *linguistics*?

> "*Every time I fire a linguist, the performance of the recognizer goes up.*"
>
> - Fred Jelinek (IBM), 1988

LANGUAGE UNDERSTANDING

Topics: Natural Language Understanding

- ***It's all about telling how likely a sentence is..***

- How likely is this sentence as an answer to the question?
  - Q. "*Who is the President of the United States?*"
  - *Likely* answer: "*Obama is the President of the U.S.*"
  - *Unlikely* answer: "*Tsipras is the President of America.*"

LANGUAGE UNDERSTANDING

Topics: Natural Language Understanding

- ***It's all about telling how likely a sentence is..***

- How likely is this sentence given this view?

  - *Likely*: "*Two dolphins are diving*"
  - *Unlikely*: "*Two men are flying*"

LANGUAGE UNDERSTANDING

Topics: Natural Language Understanding

It's all about telling how likely a sentence is..

Language Modelling

HOW LIKELY IS THIS SENTENCE?

Topics: Language Modelling

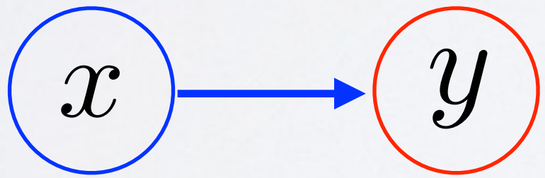
- A sentence $(x_1, x_2, \ldots, x_T)$
  - Ex) (“the”,“cat”,“is”,“eating”,“a”,“sandwich”,“on”,“a”,“couch”)
- How likely is this sentence?
- In other words, what is the probability of $(x_1, x_2, \ldots, x_T)$?
  - i.e., $p(x_1, x_2, \ldots, x_T) = ?$

HOW LIKELY IS THIS SENTENCE?

Topics: Probability 101 - Conditional Probability

- Joint probability $p(x, y)$
- Conditional probability $p(x|y)$
- Marginal probability $p(x)$ and $p(y)$

- They are related by $p(x, y) = p(x|y)p(y) = p(y|x)p(x)$



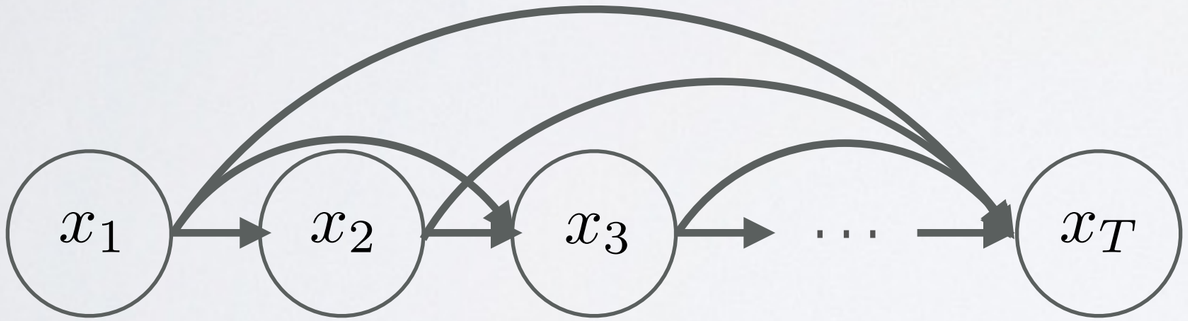
HOW LIKELY IS THIS SENTENCE?

Topics: Language Modelling as a Product of Conditionals

- Rewrite $p(x_1, x_2, \ldots, x_T)$ into

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

- Graphically,



STATISTICAL LM

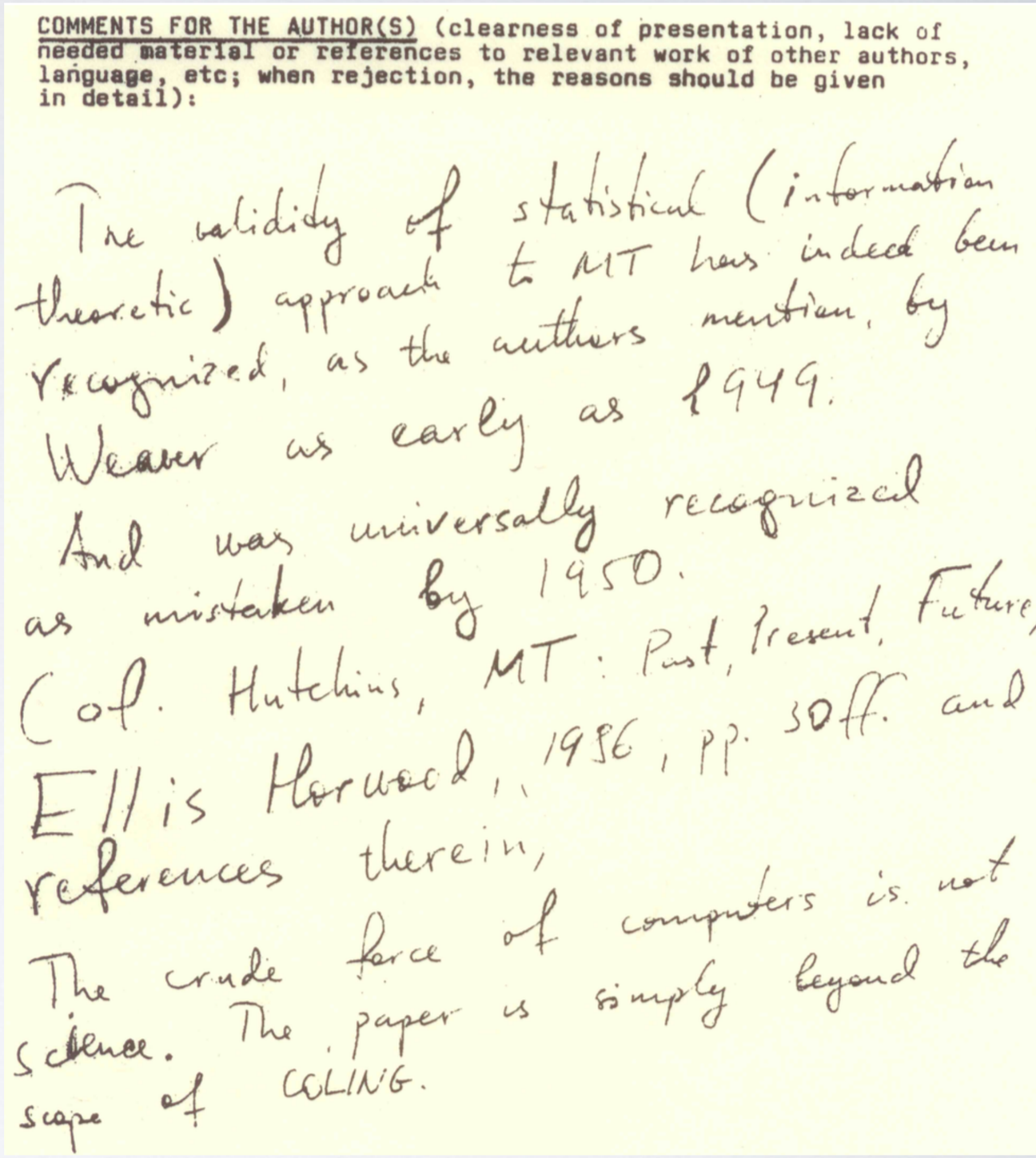
Topics: Statistical Language Modelling

- Maximize the (log-)probabilities of sentences in corpora

$$\max \mathbb{E}_D \left[\log p(x_1, x_2, \ldots, x_T) \right]$$

- Obvious to us, but not to everyone:
  - "*The validity of statistical (information theoretic) approach to MT has indeed been recognized … as early as 1949. And was universally recognized as mistaken [sic] by 1950. … The crude force of computers is not science.*"

(Review of Brown et al. (1990))



COMMENTS FOR THE AUTHOR(S) (clearness of presentation, lack of needed material or references to relevant work of other authors, language, etc; when rejection, the reasons should be given in detail):

The validity of statistical (information theoretic) approach to MT has indeed been recognized, as the authors mention, by Weaver as early as 1949. And was universally recognized as mistaken by 1950. (Cf. Hutchins, MT: Past, Present, Future, Ellis Horwood, 1986, pp. 30ff. and references therein,

The crude force of computers is not science. The paper is simply beyond the scope of COLING.

***n*-gram** Language Modelling

(Blunsom, 2015)

HOW LIKELY IS THIS SENTENCE?

Topics: Non-parametric Approach — n-gram modelling

- n-th order Markov assumption: why?

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

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

- Collect *n*-gram statistics from a *large* corpus:

$$p(x_t | x_{t-n}, \ldots, x_{t-1}) = \frac{\text{count}(x_{t-n}, \ldots, x_{t-1}, x_t)}{\text{count}(x_{t-n}, \ldots, x_{t-1})}$$

HOW LIKELY IS THIS SENTENCE?

Topics: Non-parametric Approach — n-gram modelling

- Ex) $p(\mathrm{i}, \mathrm{would}, \mathrm{like}, \mathrm{to}, \ldots, ., \langle/\mathrm{s}\rangle)$
- Unigram Modelling

  $p(\mathrm{i})p(\mathrm{would})p(\mathrm{like}) \cdots p(\langle/\mathrm{s}\rangle)$
- Bigram Modelling

  $p(\mathrm{i})p(\mathrm{would}|\mathrm{i})p(\mathrm{like}|\mathrm{would}) \cdots p(\langle/\mathrm{s}\rangle\,|.)$
- Trigram Modelling

  $p(\mathrm{i})p(\mathrm{would}|\mathrm{i})p(\mathrm{like}|\mathrm{i}, \mathrm{would}) \cdots$

$\vdots$

word	unigram	bigram	trigram	4-gram
i	6.684	3.197	3.197	3.197
would	8.342	2.884	2.791	2.791
like	9.129	2.026	1.031	1.290
to	5.081	0.402	0.144	0.113
commend	15.487	12.335	8.794	8.633
the	3.885	1.402	1.084	0.880
rapporteur	10.840	7.319	2.763	2.350
on	6.765	4.140	4.150	1.862
his	10.678	7.316	2.367	1.978
work	9.993	4.816	3.498	2.394
.	4.896	3.020	1.785	1.510
</s>	4.828	0.005	0.000	0.000
average	8.051	4.072	2.634	2.251
perplexity	265.136	16.817	6.206	4.758

HOW LIKELY IS THIS SENTENCE?

Topics: n-gram modelling — *Two closely-related issues*

- Data Sparsity
  - # of all possible n-grams: $|V|^n$, where $|V|$: size of vocabulary

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

HOW LIKELY IS THIS SENTENCE?

Topics: n-gram modelling — *Two closely-related issues*

- Conventional Solutions to Data Sparsity:
  - Smoothing:

$$p(x_t|x_{t-n},\ldots,x_{t-1}) = \frac{\text{count}(x_{t-n},\ldots,x_{t-1},x_t)+\alpha}{\text{count}(x_{t-n},\ldots,x_{t-1})+\alpha|V|}$$

(add-α smoothing)

  - Backoff:

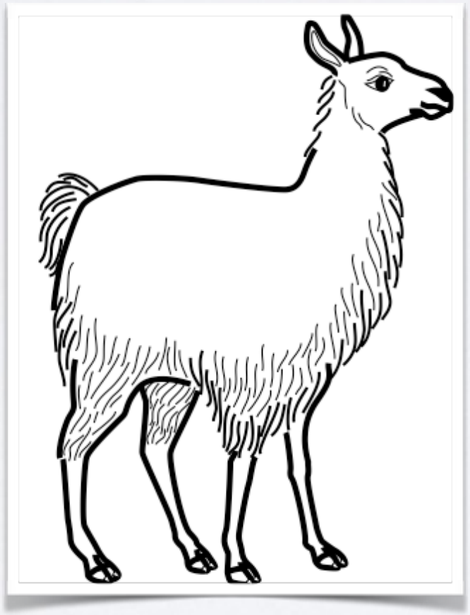
$$p(x_t|x_{t-n},\ldots,x_{t-1}) = \begin{cases} \alpha_n(x_t|x_{t-n},\ldots,x_{t-1}), \\ \quad \text{if } \text{count}_n(x_{t-n},\ldots,x_t) > 0 \\ d_n(x_{t-n},\ldots,x_{t-1})p(x_t|x_{t-n+1}\ldots,x_{t-1}), \\ \quad \text{otherwise} \end{cases}$$

(α_n: adjusted prediction model, d_n: discount factor)

HOW LIKELY IS THIS SENTENCE?

Topics: n-gram modelling — *Two closely-related issues*

- Lack of Generalization
  - (chases, a, **dog**), (chases, a, **cat**), (chases, a, **rabbit**)
  - (chases, a, **llama**)=?



Neural Language Modelling

LANGUAGE MODELLING

Topics: Neural Language Modelling

- ~~Non parametric estimator~~ ⟶ Parametric estimator

$$p(x_t|x_{t-n},\ldots,x_{t-1}) = \frac{\text{count}(x_{t-n},\ldots,x_{t-1},x_t)}{\text{count}(x_{t-n},\ldots,x_{t-1})}$$

$$= f_{x_t}(x_{t-n},\ldots,x_{t-1})$$

LANGUAGE MODELLING

Topics: Neural Language Modelling

$$p(x_t|x_{t-n},\ldots,x_{t-1}) = f_{x_t}(x_{t-n},\ldots,x_{t-1})$$

- Building a neural language model (Bengio et al., 2000)

(1) *1*-of-*K* encoding of each word $x_{t'}$

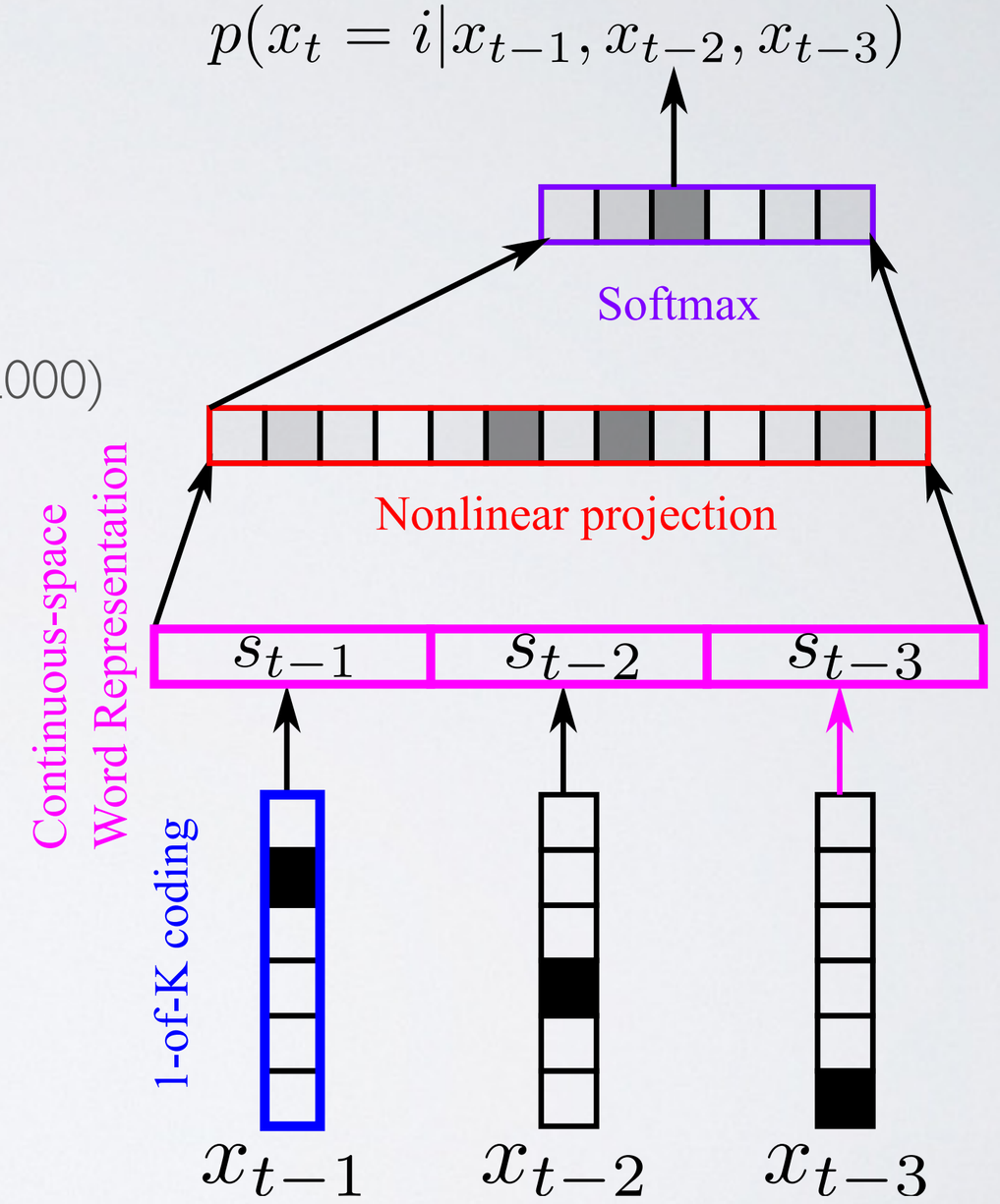
(2) Continuous space word representation

$$s_{t'} = W^\top x_{t'}, \text{ where } W \in \mathbb{R}^{|V|\times d}$$

(3) Nonlinear hidden layer

$$h = \tanh(U^\top [s_{t-1}; s_{t-2}; \cdots; s_{t-n}] + b)$$

$$, \text{ where } U \in \mathbb{R}^{nd\times d'} \text{ and } b \in \mathbb{R}^{d'}$$



LANGUAGE MODELLING

Topics: Neural Language Modelling

$$p(x_t|x_{t-n},\ldots,x_{t-1}) = f_{x_t}(x_{t-n},\ldots,x_{t-1})$$

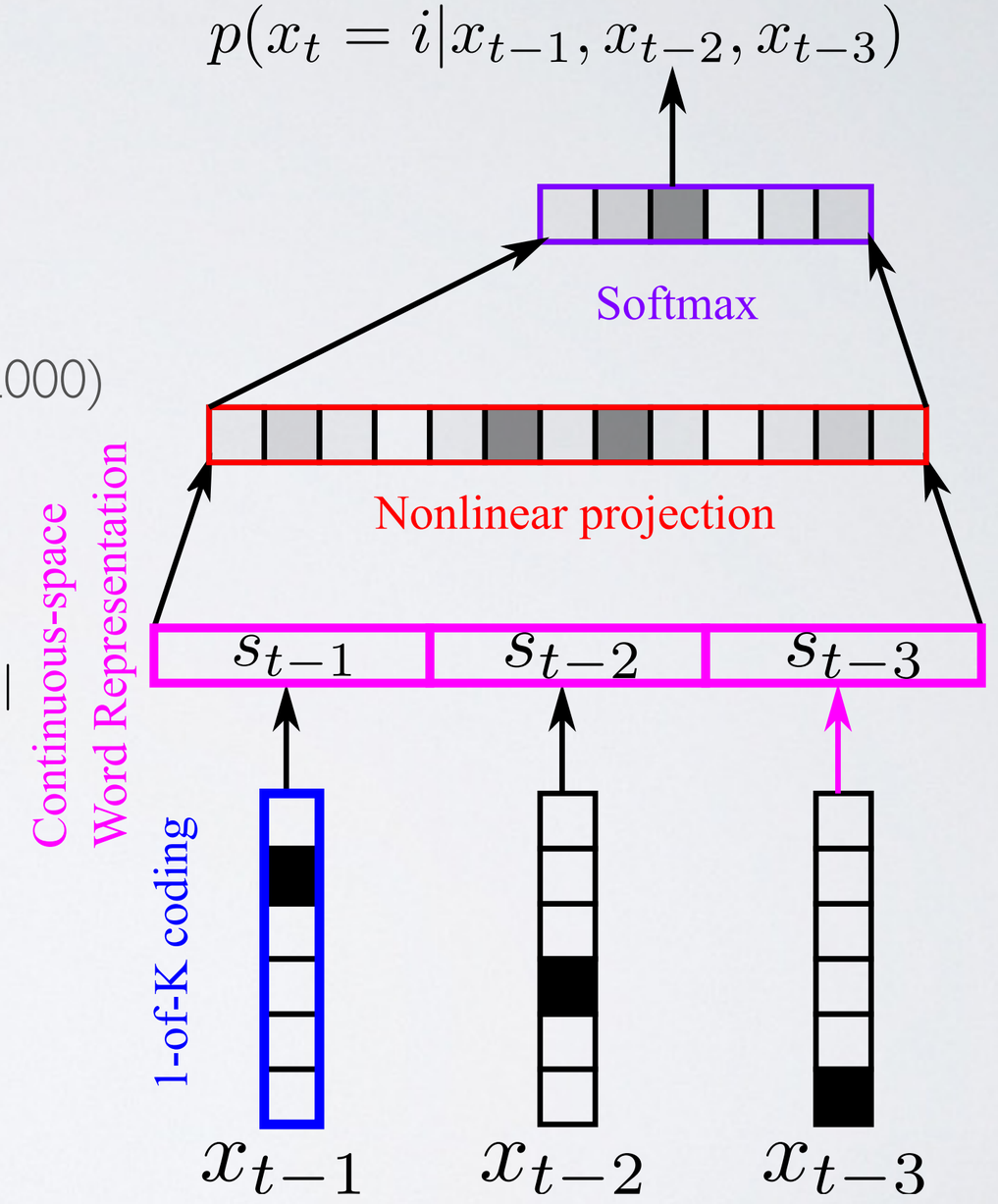
- Building a neural language model (Bengio et al., 2000)

(1)Unnormalized probabilities

$$y = Vh + c, \text{ where } V \in \mathbb{R}^{|V|\times d'} \text{ and } c \in \mathbb{R}^{|V|}$$

(2)Softmax normalization

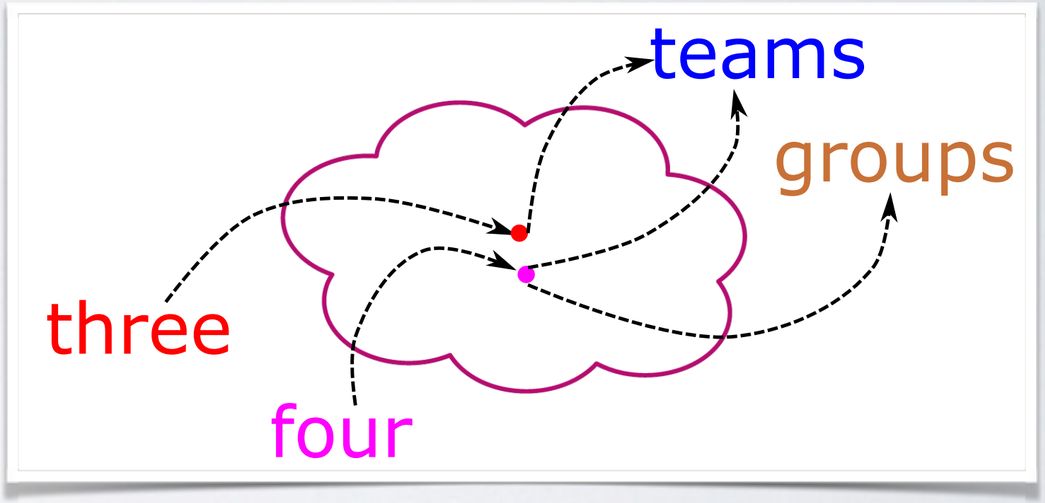
$$p(x_t = i|x_{t-n},\ldots,x_{t-1}) = \frac{\exp(y_i)}{\sum_{j=1}^{|V|}\exp(y_j)}$$



LANGUAGE MODELLING

Topics: Neural LM generalizes to *unseen* n-gram's

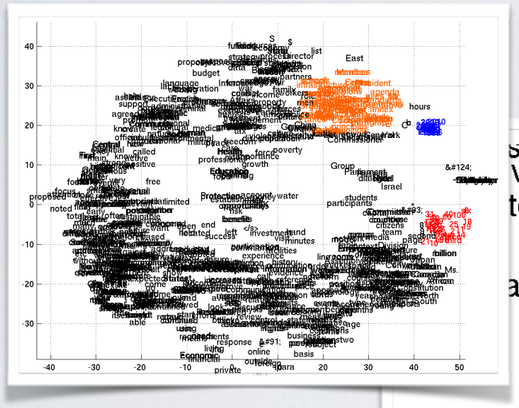
- Example sentences
  - there are three teams left for the qualification.
  - four teams have passed the first round.
  - four groups are playing in the field.

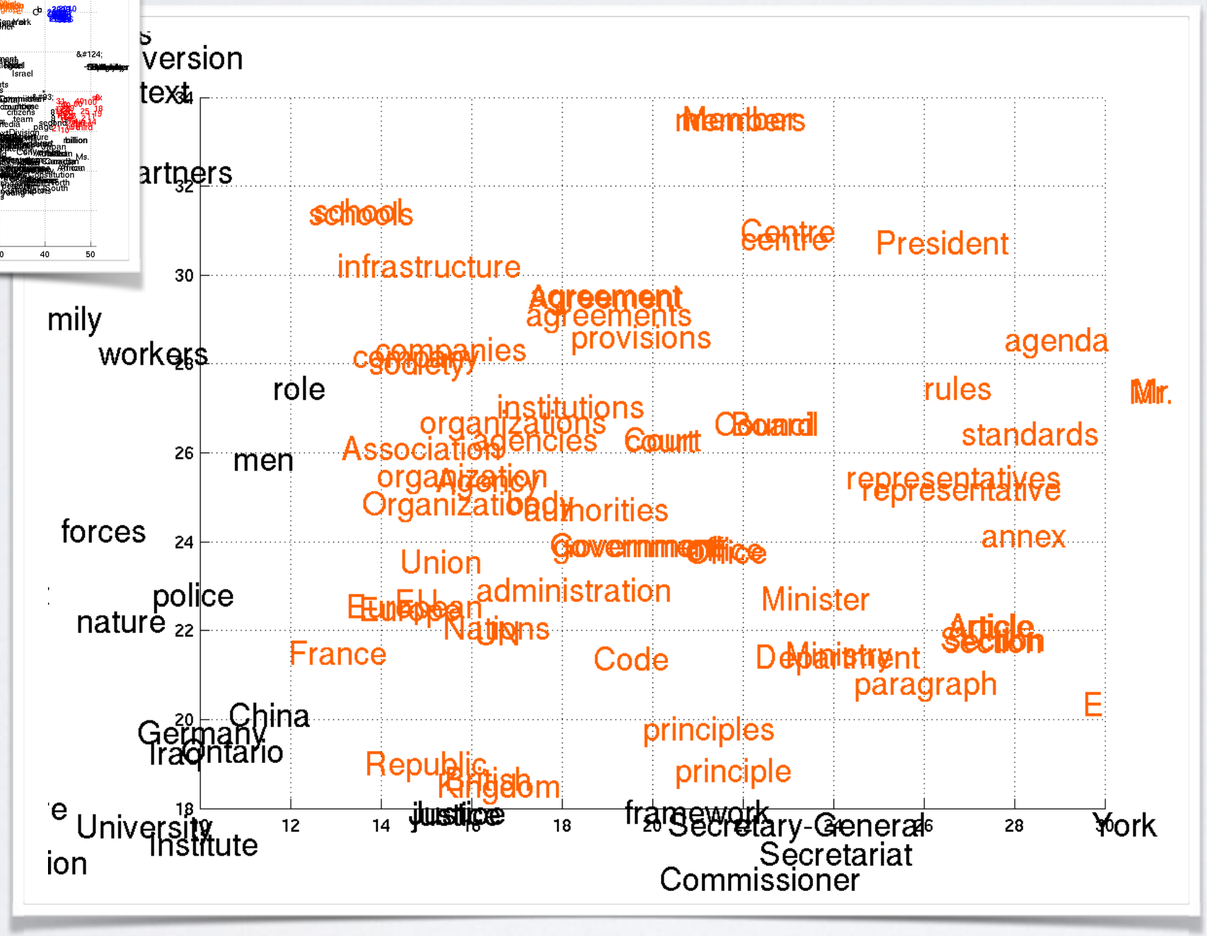


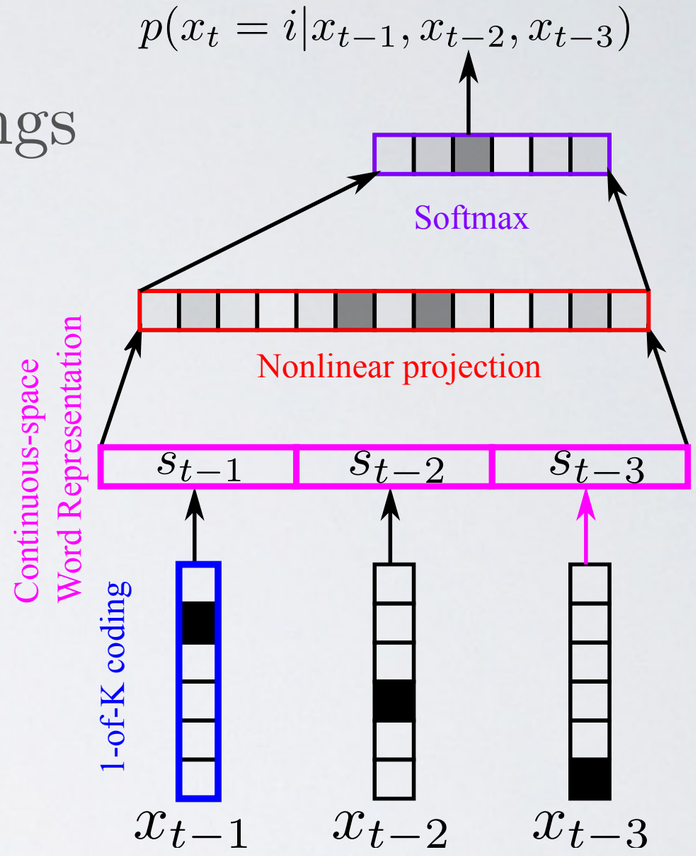
- How likely is groups followed by three?
- *Why?*

LANGUAGE MODELLING

Topics: Continuous-space representation — Embeddings







Q&A

Non-Markovian Language Modelling

LANGUAGE MODELLING

Topics: Markov Assumption

- Markov Assumption in *n*-gram modeling

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

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

- Issue: Dependency beyond the context window is ignored
  - Ex) *the same* ***stump*** *which had impaled the car of many a guest in the past thirty years and which he refused to have* ***removed***

LANGUAGE MODELLING

Topics: Non-Markovian Language Modelling

- Directly model the original conditional probabilities

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

- Feature Extraction + Readout
  - Feature Extraction: $h_t = f(x_1, x_2, \ldots, x_{t-1})$
  - Readout: $p(x_t|x_1, \ldots, x_{t-1}) = g(h_t)$

- *How can we let* f *take variable-length input?*

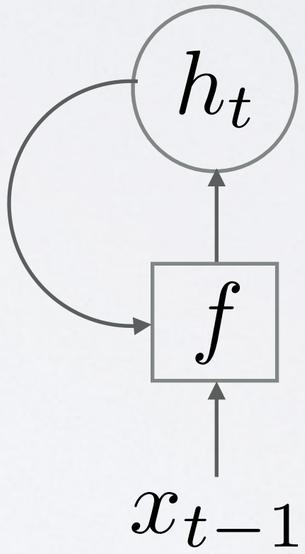
LANGUAGE MODELLING

Topics: Language Modelling via Recursion

- Directly model the original conditional probabilities

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

- Recursive Construction of f
  - Initial Condition: $h_0 = 0$
  - Recursion: $h_t = f(x_{t-1}, h_{t-1})$
- *We call h_t an internal hidden state or* ***memory***
  - h_t summarizes/memorizes the history from x_1 up to x_{t-1}



LANGUAGE MODELLING

Topics: Language Modelling via Recursion

- Example: $p(\text{eating}|\text{the, cat, is})$

  (1) Initialization: $h_0 = 0$

  (2) Recursion

  (1) $h_1 = f(h_0, \text{the})$

  (2) $h_2 = f(h_1, \text{cat})$

  (3) $h_3 = f(h_2, \text{is})$

  (3) Readout: $p(\text{eating}|\text{the, cat, is}) = g(h_3)$

- It works for any number of context words

RNN Language Modelling

LANGUAGE MODELLING

Topics: Recurrent neural network language model

- Example: $p(\text{the}, \text{cat}, \text{is}, \text{eating})$

  (1) Initialization: $h_0 = 0 \longrightarrow p(\text{the}) = g(h_0)$

  (2) Recursion with Readout

  (1) $h_1 = f(h_0, \text{the}) \longrightarrow p(\text{cat}|\text{the}) = g(h_1)$

  (2) $h_2 = f(h_1, \text{cat}) \longrightarrow p(\text{is}|\text{the}, \text{cat}) = g(h_2)$

  (3) $h_3 = f(h_2, \text{is}) \longrightarrow p(\text{eating}|\text{the}, \text{cat}, \text{is}) = g(h_3)$

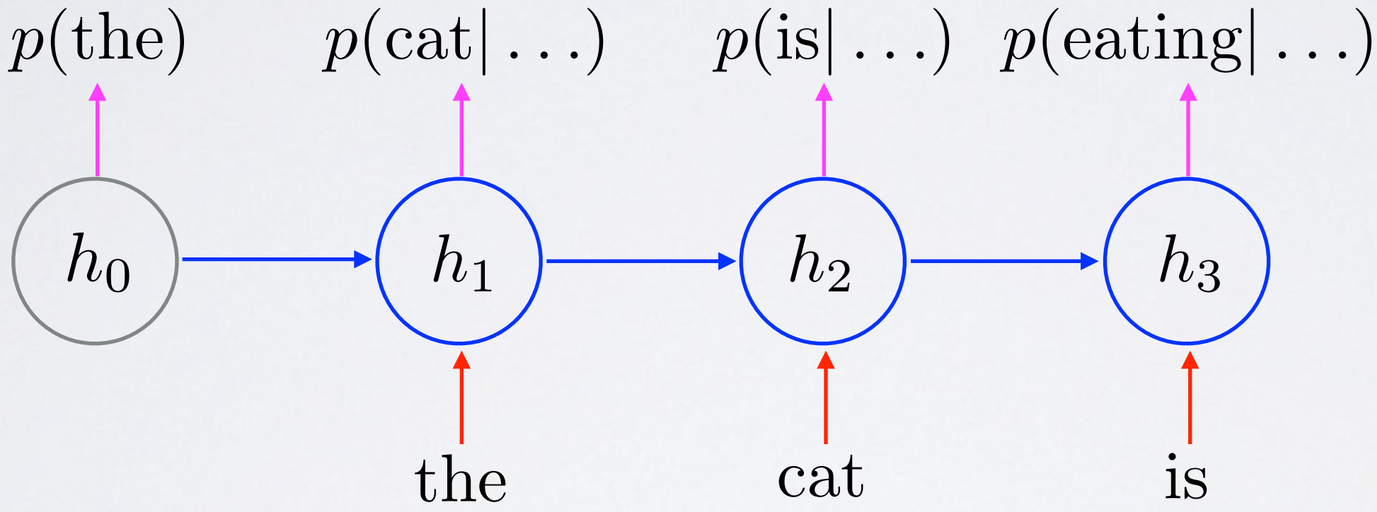
  (3) Combination: $p(\text{the}, \text{cat}, \text{is}, \text{eating}) = g(h_0)g(h_1)g(h_2)g(h_3)$

- Read, Update and Predict

LANGUAGE MODELLING

Topics: Recurrent neural network language model

- Example: $p(\text{the}, \text{cat}, \text{is}, \text{eating})$

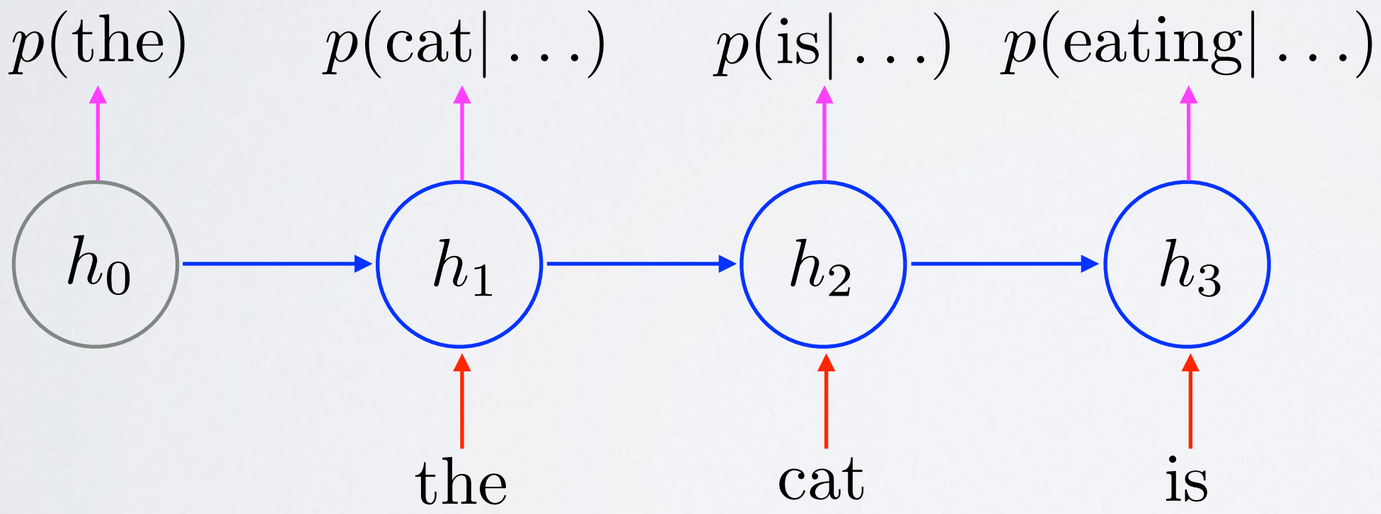


- Read, Update and Predict

LANGUAGE MODELLING

Topics: Building an RNN Language Model

- What do we need?
  - Transition Function $h_t = f(h_{t-1}, x_{t-1})$
  - Output/Readout Function $p(x_t = w | x_1, \ldots, x_{t-1}) = g_w(h_t)$



LANGUAGE MODELLING

Topics: Building an RNN Language Model - Transition Function

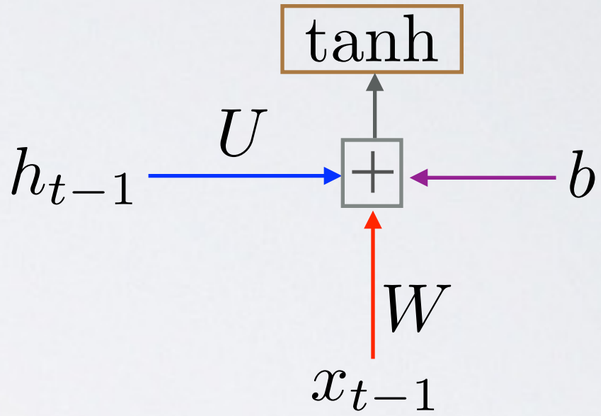
- **Inputs**
  - Input $x_{t-1} \in \{0,1\}^{|V|}$: one-hot vector, i.e., $x_{t-1} = w \in \{1, \ldots, |V|\}$
  - Hidden state $h_{t-1} \in \mathbb{R}^d$
- **Parameters**
  - Input weight matrix $W \in \mathbb{R}^{d \times |V|}$ *(often called word embeddings)*
  - Transition weight matrix $U \in \mathbb{R}^{d \times d}$
  - Bias vector $b \in \mathbb{R}^d$

LANGUAGE MODELLING

Topics: Building an RNN Language Model - Transition Function

- **Inputs**: $x_{t-1} \in \{0,1\}^{|V|}$, $h_{t-1} \in \mathbb{R}^d$
- **Parameters**: $W \in \mathbb{R}^{d \times |V|}$, $U \in \mathbb{R}^{d \times d}$, $b \in \mathbb{R}^d$
- Naive **Transition Function**

$$h_t = \tanh(Wx_{t-1} + Uh_{t-1} + b)$$



(1) Continuous-space Representation of word: Wx_{t-1}

(2) Linear Transformation of the Previous Hidden State: Uh_{t-1}

(3) Additive combination of x_{t-1} and h_{t-1} together with b

(4) Point-wise nonlinear transformation

LANGUAGE MODELLING

Topics: Building an RNN Language Model - Readout Function

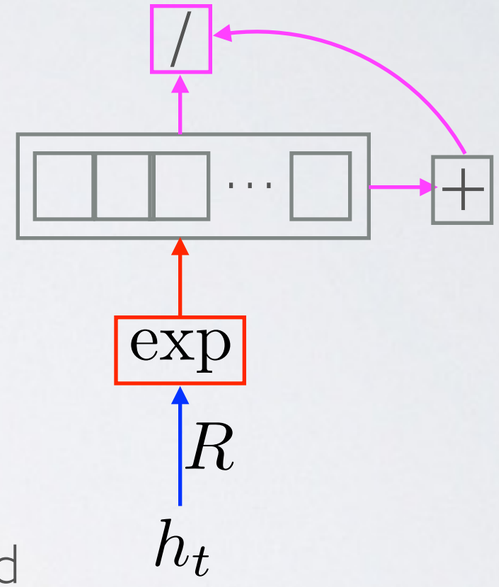
- **Inputs**
  - (Current) Hidden State $h_t \in \mathbb{R}^d$
- **Parameters**
  - Output matrix $R \in \mathbb{R}^{|V| \times d}$ *(often called **target** word embeddings)*
  - Bias vector $c \in \mathbb{R}^{|V|}$

LANGUAGE MODELLING

Topics: Building an RNN Language Model - Readout Function

- **Inputs** $h_t \in \mathbb{R}^d$
- **Parameters** $R \in \mathbb{R}^{|V| \times d}$, $c \in \mathbb{R}^{|V|}$
- *Softmax* **Readout Function**

$$p(x_t = w | x_{<t}) = g_w(h_t) = \frac{\exp(R_w^\top h_{t-1} + c_w)}{\sum_{i=1}^{|V|} \exp(R_i^\top h_{t-1} + c_i)}$$



(1) Linear projection of the hidden state for each possible target word

$$v_i = R_i^\top h_{t-1} \text{ for all } i = 1, \ldots, |V|$$

(3) Transform each projected vector v_i to be *positive* $\tilde{p}_i = \exp(v_i)$

(4) Normalize $\tilde{p}_i$'s to make them into *probabilities* of the i-th target words

LANGUAGE MODELLING

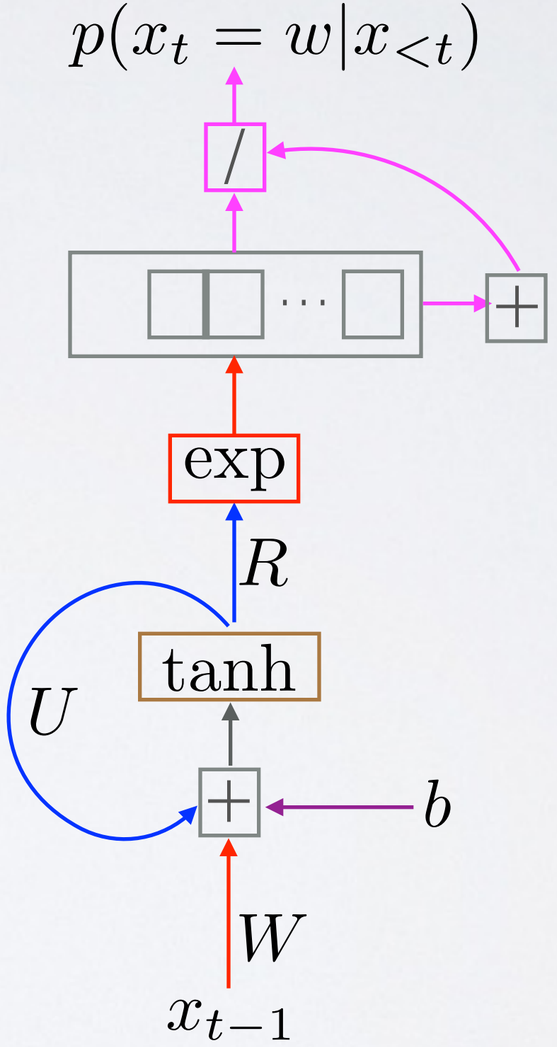
Topics: Building an RNN Language Model

- Recursion and Readout:
  - Recursion

$$h_t = \tanh(W x_{t-1} + U h_{t-1} + b)$$

  - Readout/Output

$$p(x_t = w | x_{<t}) = \frac{\exp(R_w^\top h_{t-1})}{\sum_{i=1}^{|V|} \exp(R_i^\top h_{t-1})}$$



Training RNN-LM

LANGUAGE MODELLING

Topics: Cost Function $J(\Theta)$

- Log-Probability of a sentence $(x_1, x_2, \ldots, x_T)$

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

- Train an RNN LM to maximize the log-prob's of *training* sentences
- Given a training set of N sentences: $\{(x_1^1, \ldots, x_{T_1}^1), \ldots, (x_1^N, \ldots, x_{T_N}^N)\}$

$$\text{maximize}_\Theta \frac{1}{N} \sum_{n=1}^{N} \log p(x_1^n, \ldots, x_{T_n}^n)$$

$$\iff \text{minimize}_\Theta J(\Theta) = -\frac{1}{N} \sum_{n=1}^{N} \sum_{t=1}^{T_n} \log p(x_t^n | x_1^n \ldots, x_{t-1}^n)$$

LANGUAGE MODELLING

Topics: Minibatch Stochastic Gradient Descent - Recap

(1)Randomly select a minibatch of N' sentences: $D = \left\{x^1, \ldots, x^{N'}\right\}$

(2)Compute the gradient of per-sample cost w.r.t. Θ : $\nabla J(\Theta, x^n)$

(3)Compute the minibatch gradient:

$$\nabla J(\Theta, D) = \frac{1}{N'} \sum_{n=1}^{N'} \nabla J(\Theta, x^n)$$

(4)Update the parameters Θ

$$\Theta \leftarrow \Theta + \eta \nabla J(\Theta, D)$$

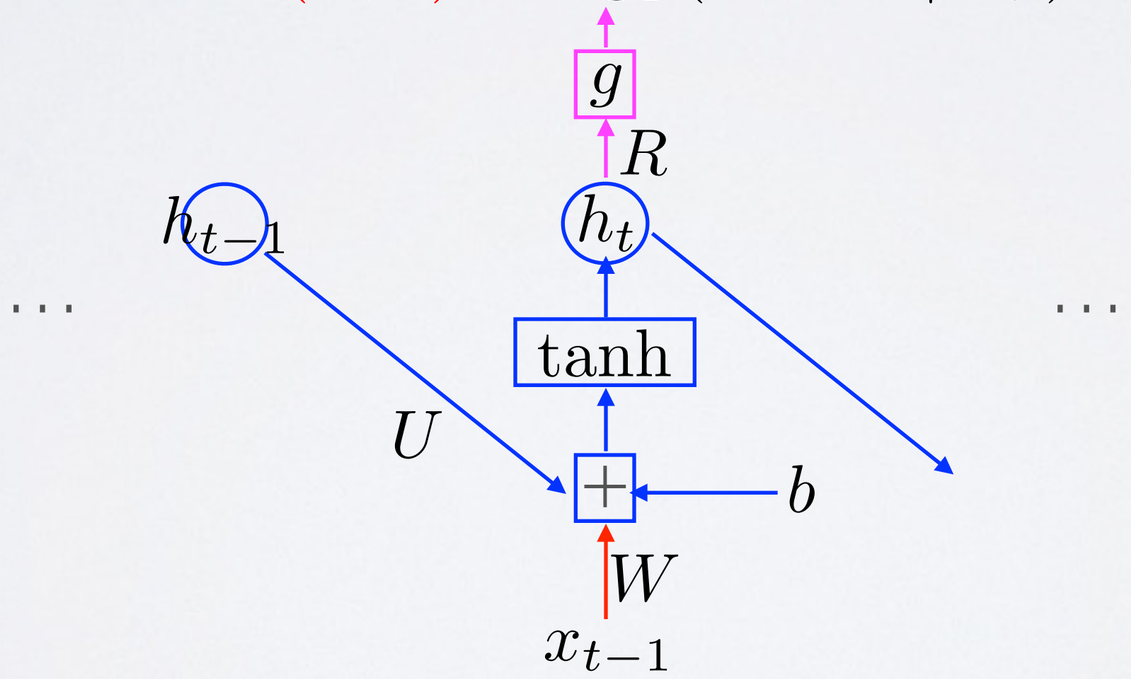
(5)Repeat until *convergence*

LANGUAGE MODELLING

Topics: Backpropagation through time

- Decomposition of a per-sample cost function $J(\Theta, x) = -\sum_{t=1}^{T} J_t(\Theta, x_t)$
- Unrolled Computational Graph

$$J_t(\Theta, \hat{x}) = \log p(x_t = \hat{x}_t | x_{<t})$$



LANGUAGE MODELLING

Topics: Backpropagation through time

(1)Initialize $\nabla_R, \nabla_U, \nabla_W, \nabla_b$ and $t = T$

(1)The per-step cost derivative: $\frac{\partial J_t}{\partial g}$

(2)Gradient w.r.t. R : $\frac{\partial J_t}{\partial g} \frac{\partial g}{\partial R}$

(3)Gradient w.r.t. h_t : $\frac{\partial J_t}{\partial g} \frac{\partial g}{\partial h_t} + \frac{\partial J_{>t}}{\partial h_{t+1}} \frac{\partial h_{t+1}}{\partial h_t}$

(4)Gradient w.r.t. U : $\frac{\partial J_{\geq t}}{\partial h_t} \frac{\partial h_t}{\partial U}$

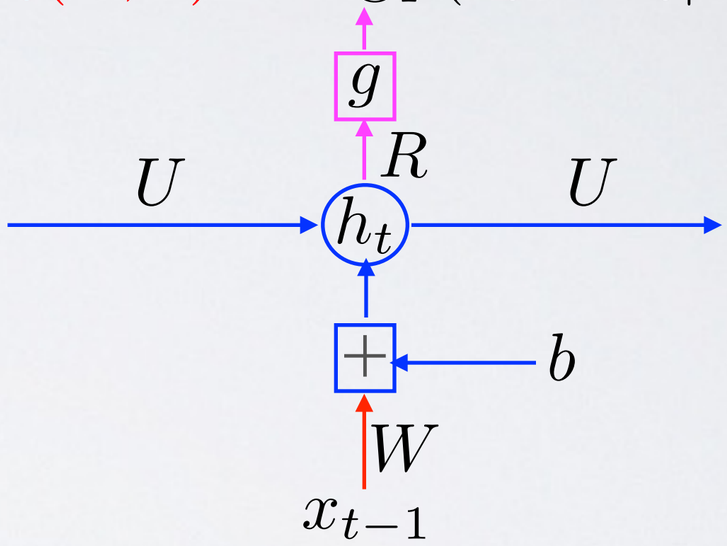
(5)Gradient w.r.t W and b : $\frac{\partial J_{\geq t}}{\partial h_t} \frac{\partial h_t}{\partial W}$, $\frac{\partial J_{\geq t}}{\partial h_t} \frac{\partial h_t}{\partial b}$

(2)Update the parameter gradient and repeat until $t = 1$

$$\nabla_R \leftarrow \nabla_R + \frac{\partial J_t}{\partial R}, \nabla_U \leftarrow \nabla_U + \frac{\partial J_{\geq t}}{\partial U}$$

$$\nabla_W \leftarrow \nabla_W + \frac{\partial J_{\geq t}}{\partial W}, \nabla_b \leftarrow \nabla_b + \frac{\partial J_{\geq t}}{\partial b}$$

$$J_t(\Theta, \hat{x}) = \log p(x_t = \hat{x}_t | x_{<t})$$



Note: I'm abusing math a lot here!!

Q&A

Code: https://github.com/kyunghyuncho/dl4mt-material/tree/master/session0

Gated Recurrent Units

GATED RECURRENT UNITS

Topics: Temporal Dependency and Vanishing Gradient

- How much influence does h_t have on $\log p(x_{t+n}|x_{<t+n})$?

$$\frac{\partial J_{t+n}}{\partial h_t} = \frac{\partial J_{t+n}}{\partial g} \frac{\partial g}{\partial h_{t+N}} \frac{\partial h_{t+N}}{\partial h_{t+N-1}} \cdots \frac{\partial h_{t+1}}{\partial h_t}$$

Note: I'm abusing math a lot here!!

- With the naive transition function?

$$\frac{\partial h_{t+1}}{\partial h_t} = U^\top \frac{\partial \tanh(a)}{\partial a}, \text{ where } a = Wx_t + Uh_t + b$$

- Let's rewrite it

$$\frac{\partial J_{t+n}}{\partial h_t} = \frac{\partial J_{t+n}}{\partial g} \frac{\partial g}{\partial h_{t+N}} \underbrace{\prod_{n=1}^{N} U^\top \text{diag}\left(\frac{\partial \tanh(a_{t+n})}{\partial a_{t+n}}\right)}$$

Problematic! Bengio et al. (1994)

GATED RECURRENT UNITS

Topics: Temporal Dependency and Vanishing Gradient

- Upper bound on the **norm of the gradient** w.r.t. h_t ?

$$\left\| \prod_{n=1}^{N} U^\top \operatorname{diag}\left(\frac{\partial \tanh(a_{t+n})}{\partial a_{t+n}} \right) \right\| \leq \prod_{n=1}^{N} \|U^\top\| \prod_{n=1}^{N} \left\| \frac{\partial \tanh(a_{t+n})}{\partial a_{t+n}} \right\|$$

- **Observations**

  (1) *Vanishing gradient* when $\lambda_{\max}(U) < 1 : \prod_{n=1}^{N} \|U^\top\| \to 0$

  (2) *Vanishing gradient* when the units are *saturated*: $\frac{\partial \tanh(a_{t+n})}{\partial a_{t+n}} \to 0$

  (3) Potentially, *exploding gradient* when $\lambda_{\max}(U) > 1$

- **Problem**: *It's likely that there's no learning signal!*

GATED RECURRENT UNITS

Topics: Exploding gradient is *less* problematic

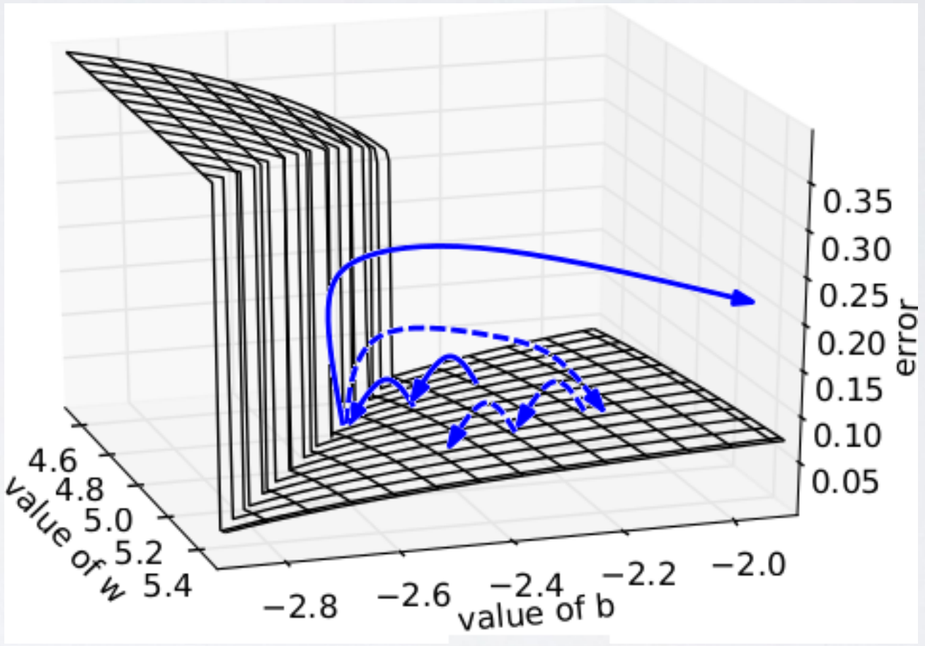
- "*when gradients explode so does the curvature along v, leading to a wall in the error surface*"
- **Solution**: Gradient Clipping

(1) Gradient norm clipping

$$\tilde{\nabla} \leftarrow \begin{cases} \frac{c}{\|\nabla\|}\nabla & \text{,if } \|\nabla\| \geq c \\ \nabla & \text{,otherwise} \end{cases}$$

(2) Element-wise gradient clipping

$$\nabla_i \leftarrow \min(c, \nabla_i), \text{ for all } i \in \{1, \ldots, \dim\nabla\}$$



Pascanu et al. (2013)

GATED RECURRENT UNITS

Topics: But, vanishing gradient is very problematic

- Why does the gradient vanish?

$$\left\| \frac{\partial h_{t+N}}{\partial h_t} \right\| = \left\| \prod_{n=1}^{N} U^\top \operatorname{diag}\left(\frac{\partial \tanh(a_{t+n})}{\partial a_{t+n}} \right) \right\| \to 0$$

- Can we simply "maximize" $\left\| \frac{\partial h_{t+N}}{\partial h_t} \right\|$?
  - "*we need to force the network to increase the norm of* $\frac{\partial h_{t+N}}{\partial h_t}$ *at the expense of larger errors*"
- Pascanu et al. (2013)
  - Regularize $\Omega = \sum_k \Omega_k = \sum_k \left(\frac{\left\| \frac{\partial \mathcal{E}}{\partial \mathbf{x}_{k+1}} \frac{\partial \mathbf{x}_{k+1}}{\partial \mathbf{x}_k} \right\|}{\left\| \frac{\partial \mathcal{E}}{\partial \mathbf{x}_{k+1}} \right\|} - 1 \right)^2$

GATED RECURRENT UNITS

Topics: But, vanishing gradient is very problematic

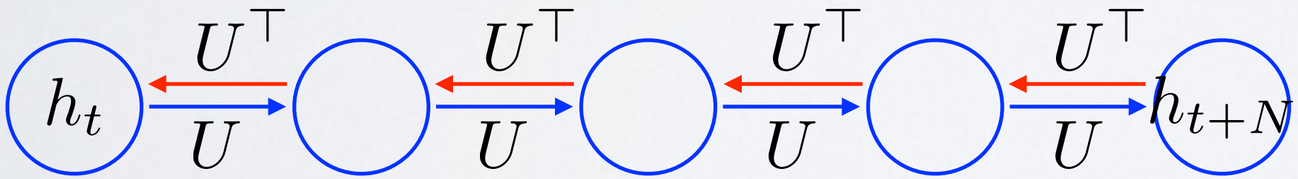
- Why does the gradient vanish?

$$\left\| \frac{\partial h_{t+N}}{\partial h_t} \right\| = \left\| \prod_{n=1}^{N} U^\top \mathrm{diag}\left(\frac{\partial \tanh(a_{t+n})}{\partial a_{t+n}} \right) \right\| \to 0$$

- Perhaps, it is a problem with the naive transition function…

$$h_t = \tanh(W x_{t-1} + U h_{t-1} + b)$$

- Error is backpropagated through *every intermediate node*



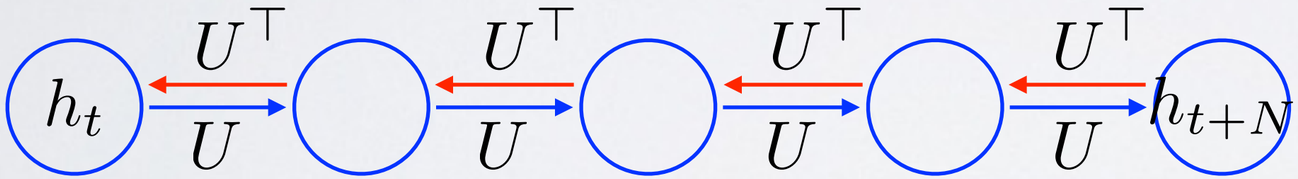
GATED RECURRENT UNITS

Topics: But, vanishing gradient is very problematic

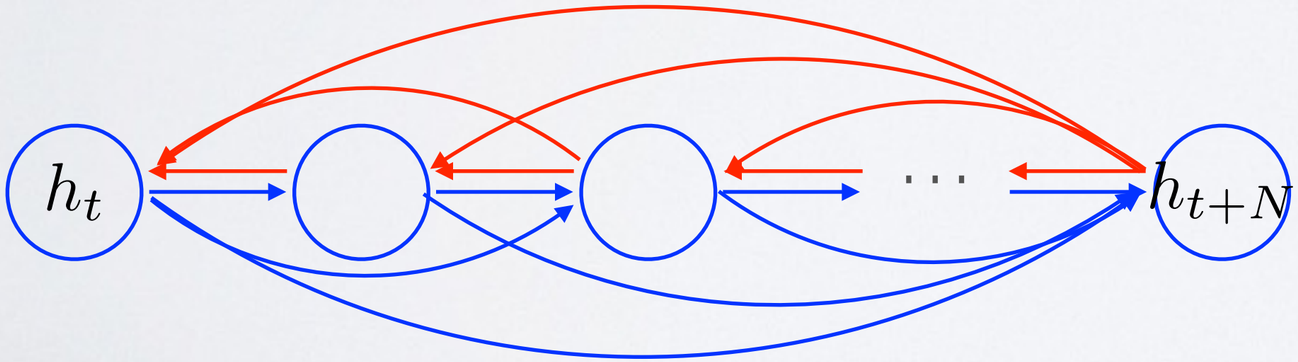
- Perhaps, it is a problem with the naive transition function…

$$h_t = \tanh(W x_{t-1} + U h_{t-1} + b)$$

- Error is backpropagated through *every intermediate node*



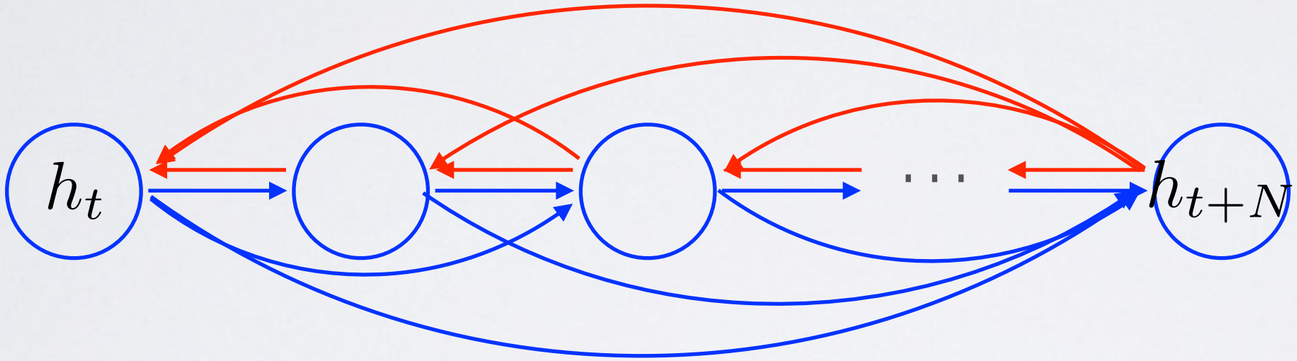
- Temporal *shortcut* connections



GATED RECURRENT UNITS

Topics: Gated Recurrent Units (GRU)

- Temporal shortcut connections



- *Adaptive* Leaky integration

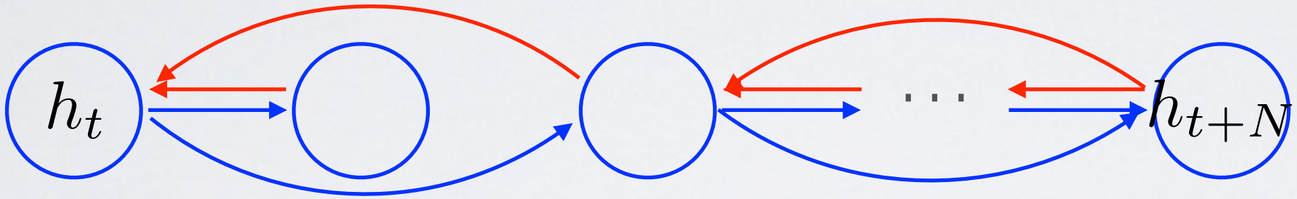
$$h_t = (1 - u_t) \odot h_{t-1} + u_t \odot \tilde{h}_t$$

- *Update* gate $u_t = \sigma(W_u x_{t-1} + U_u h_{t-1} + b_u)$
- Candidate state $\tilde{h}_t = \tanh(W x_{t-1} + U h_{t-1} + b)$

GATED RECURRENT UNITS

Topics: Gated Recurrent Units (GRU)

- Pruning connections: avoids the diffusion of signal



- *Adaptive* Reset

$$\tilde{h}_t = \tanh(W x_{t-1} + U(r_t \odot h_{t-1}) + b)$$

- *Reset* gate

$$r_t = \sigma(W_r x_{t-1} + U_r h_{t-1} + b_r)$$

GATED RECURRENT UNITS

Topics: Gated Recurrent Units (GRU)

- *Update* and *Reset* gates

$$u_t = \sigma(W_u x_{t-1} + U_u h_{t-1} + b_u)$$

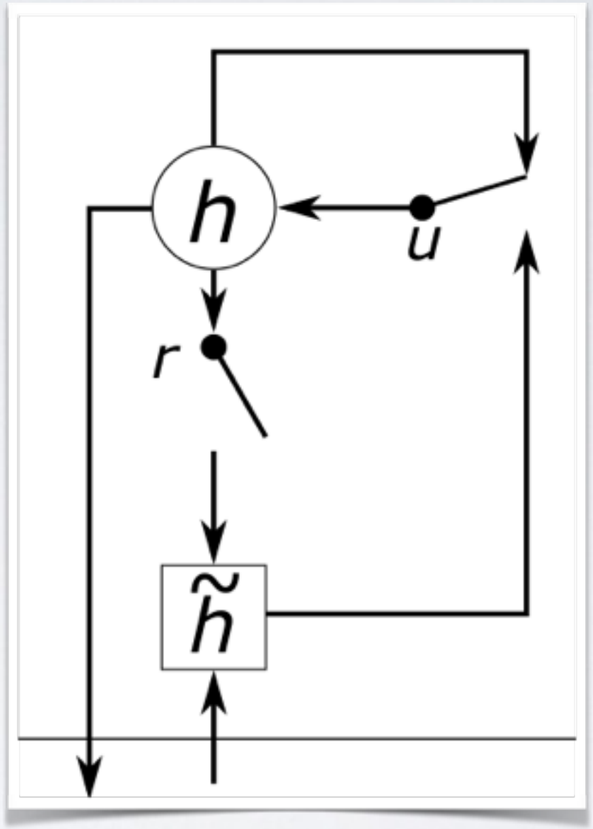
$$r_t = \sigma(W_r x_{t-1} + U_r h_{t-1} + b_r)$$

- *Candidate* hidden state

$$\tilde{h}_t = \tanh(W x_{t-1} + U(r_t \odot h_{t-1}) + b)$$

- *Adaptive* Leaky Integration

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



Cho et al. (2014)

GATED RECURRENT UNITS

Topics: Long Short-Term Memory (LSTM)

- *Input*, *Forget* and *Output* gates

$$i_t = \sigma(W_i x_{t-1} + U_i h_{t-1} + b_i)$$

$$f_t = \sigma(W_f x_{t-1} + U_f h_{t-1} + b_f)$$

$$o_t = \sigma(W_o x_{t-1} + U_o h_{t-1} + b_o)$$

- *Candidate memory* cell state

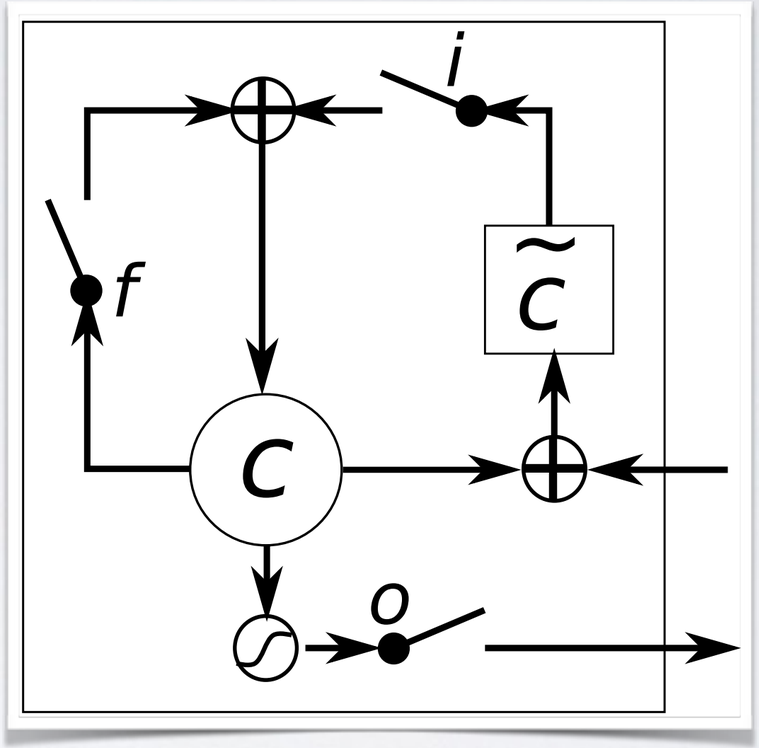
$$\tilde{c}_t = \tanh(W x_{t-1} + U h_{t-1} + b)$$

- *Adaptive* Leaky Integration

$$c_t = f_t \odot c_{t-1} + i_t \odot \tilde{c}_t$$

- Output

$$h_t = o_t \odot \tanh(c_t)$$



Hochreiter&Schmidhuber (1999), Gers et al. (2001)

Q&A

Code: https://github.com/kyunghyuncho/dl4mt-material/tree/master/session0

Next Lecture: **Neural** Machine Translation