Word embeddings and neural language modeling

AACIMP 2015

Sergii Gavrylov

Overview

- Natural language processing
- Word representations
- Statistical language modeling
- Neural models
- Recurrent neural network models
- Long short-term memory rnn models

Natural language processing

- NLP mostly works with text data (but its methods could be applied to music, bioinformatic, speech etc.)
- From the perspective of machine learning NL is a sequence of variable-length sequences of high-dimensional vectors.

Word representation

One-hot encoding

V = {zebra, horse, school, summer}

One-hot encoding

V = {zebra, horse, school, summer}

v(zebra) = [1, 0, 0, 0]
v(horse) = [0, 1, 0, 0]
v(school) = [0, 0, 1, 0]
v(summer) = [0, 0, 0, 1]

One-hot encoding

```
V = {zebra, horse, school, summer}
```

```
v(zebra) = [1, 0, 0, 0]
v(horse) = [0, 1, 0, 0]
v(school) = [0, 0, 1, 0]
v(summer) = [0, 0, 0, 1]
```

(+) Pros:

Simplicity

(-) Cons:

One-hot encoding can be memory inefficient

Notion of word similarity is undefined with one-hot encoding

Distributional representation

Is there a representation that preserves the similarities of word meanings?

$$d(\mathbf{v}(\text{zebra}), \mathbf{v}(\text{horse})) < d(\mathbf{v}(\text{zebra}), \mathbf{v}(\text{summer}))$$

Distributional representation

Is there a representation that preserves the similarities of word meanings?

$$d(v(\text{zebra}), v(\text{horse})) < d(v(\text{zebra}), v(\text{summer}))$$

“You shall know a word by the company it keeps” - John Rupert Firth

Distributional representation

“A cute, hairy wampimuk is sitting on the hands.”

clic.cimec.unitn.it/marco/publications/acl2014/lazaridou-etal-wampimuk-acl2014.pdf

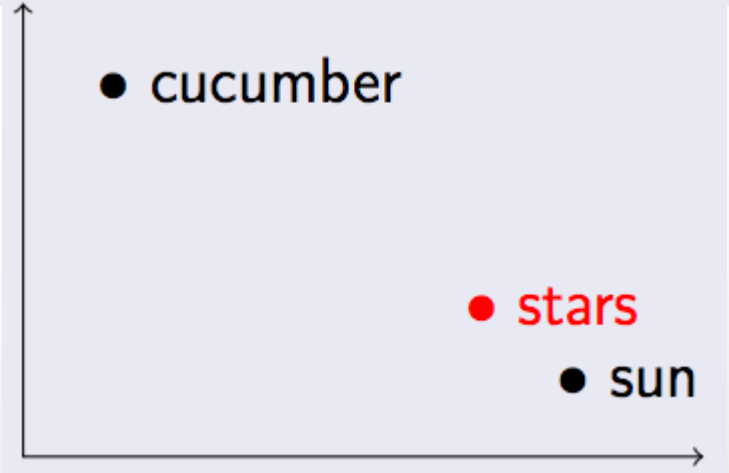
Distributional representation

he curtains open and the stars shining in on the barely
ars and the cold , close stars " . And neither of the w
rough the night with the stars shining so brightly , it
made in the light of the stars . It all boils down , wr
surely under the bright stars , thrilled by ice-white
sun , the seasons of the stars ? Home , alone , Jay pla
m is dazzling snow , the stars have risen full and cold
un and the temple of the stars , driving out of the hug
in the dark and now the stars rise , full and amber a
bird on the shape of the stars over the trees in front
But I could n't see the stars or the moon , only the
they love the sun , the stars and the stars . None of
r the light of the shiny stars . The plash of flowing w
man 's first look at the stars ; various exhibits , aer
rief information on both stars and constellations, inc

www.cs.ox.ac.uk/files/6605/aclVectorTutorial.pdf

Distributional representation

	shining	bright	trees	dark	look
stars	38	45	2	27	12



www.cs.ox.ac.uk/files/6605/aclVectorTutorial.pdf

Distributional representation

(+) Pros:

- Simplicity
- Has notion of word similarity

(-) Cons:

- Distributional representation can be memory inefficient

Distributed representation

V is a vocabulary

$$w_i \in V$$

$$v(w_i) \in \mathbf{R}^n$$

$v(w_i)$ is a low-dimensional, learnable, dense word vector

Distributed representation

colah.github.io/posts/2014-07-NLP-RNNs-Representations

Distributed representation

(+) Pros:

Has notion of word similarity

is memory efficient (low dimensional)

(-) Cons:

is computationally intensive

Distributed representation as a lookup table

W is a matrix whose rows are $v(w_i) \in \mathbf{R}^n$

$v(w_i)$ returns i^{th} row of W

Statistical language modeling

A sentence $\boldsymbol{s} = (x_1, x_2, \ldots, x_T)$

How likely is $\boldsymbol{s}$?
$p(x_1, x_2, \ldots, x_T)$

according to the chain rule (probability)

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

n-gram models

n-th order Markov assumption

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

n-gram models

n-th order Markov assumption

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

bigram model of **s** = (a, cute, wampimuk, is, on, the, tree, .)

1. How likely does ‘a’ follow ‘<S>’?
2. How likely does ‘cute’ follow ‘a’?
3. How likely does ‘wampimuk’ follow ‘cute’?
4. How likely does ‘is’ follow ‘wampimuk’?
5. How likely does ‘on’ follow ‘is’?
6. How likely does ‘the’ follow ‘on’?
7. How likely does ‘tree’ follow ‘the’?
8. How likely does ‘.’ follow ‘tree’?
9. How likely does ‘<\S>’ follow ‘.’?

n-gram models

n-th order Markov assumption

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

bigram model of `s = (a, cute, wampimuk, is, on, the, tree, .)`

$$P(w_i \mid w_{i-(n-1)}, \ldots, w_{i-1}) = \frac{\text{count}(w_{i-(n-1)}, \ldots, w_{i-1}, w_i)}{\text{count}(w_{i-(n-1)}, \ldots, w_{i-1})}$$

the counts are obtained from a training corpus

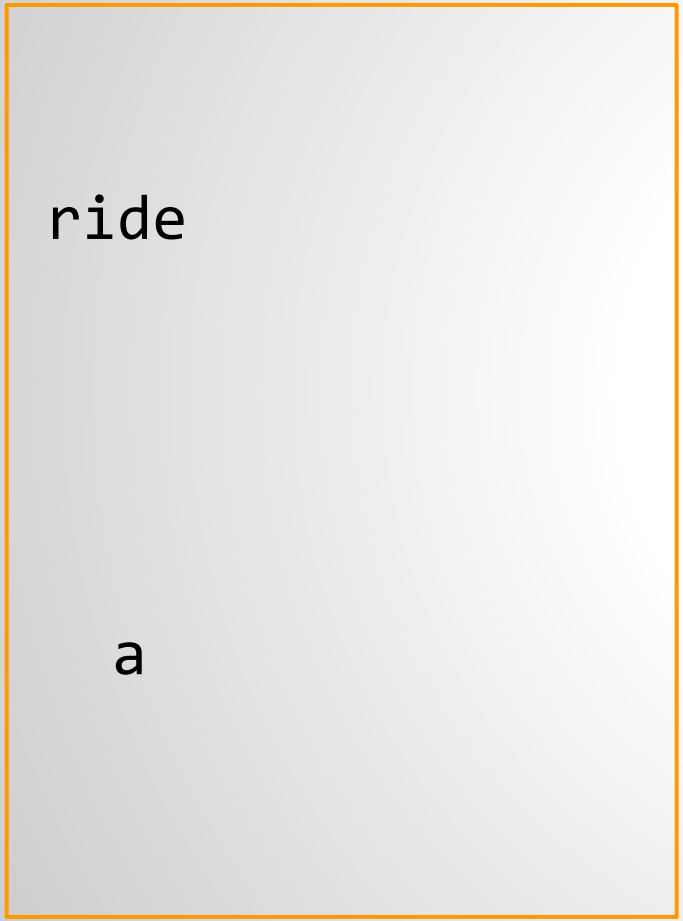
n-gram models

Issues:

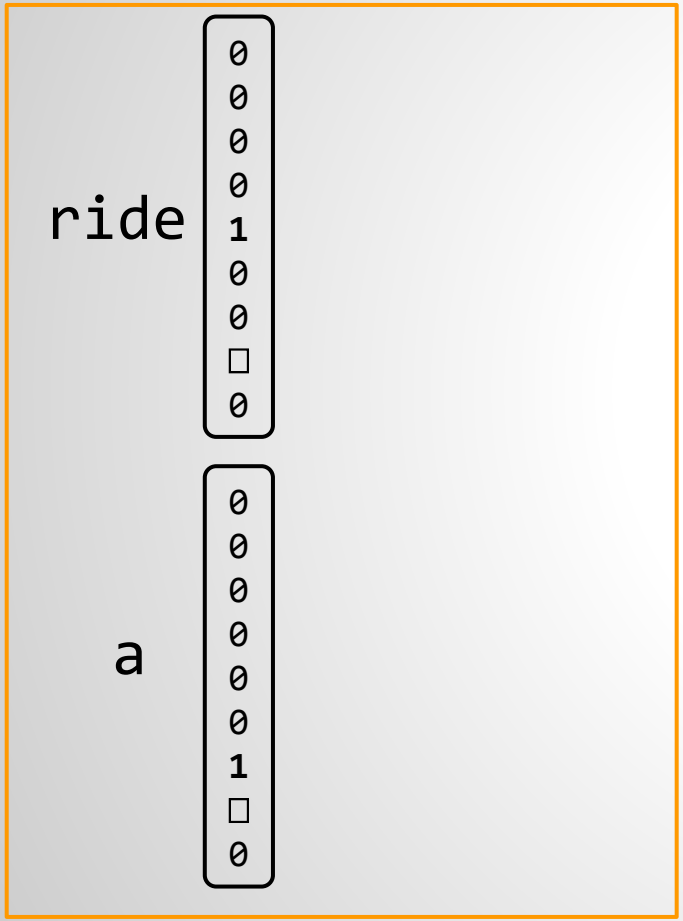
- Data sparsity
- Lack of generalization:
  - [ride a *horse*], [ride a *llama*]
  - [ride a ***zebra***]

Neural language model

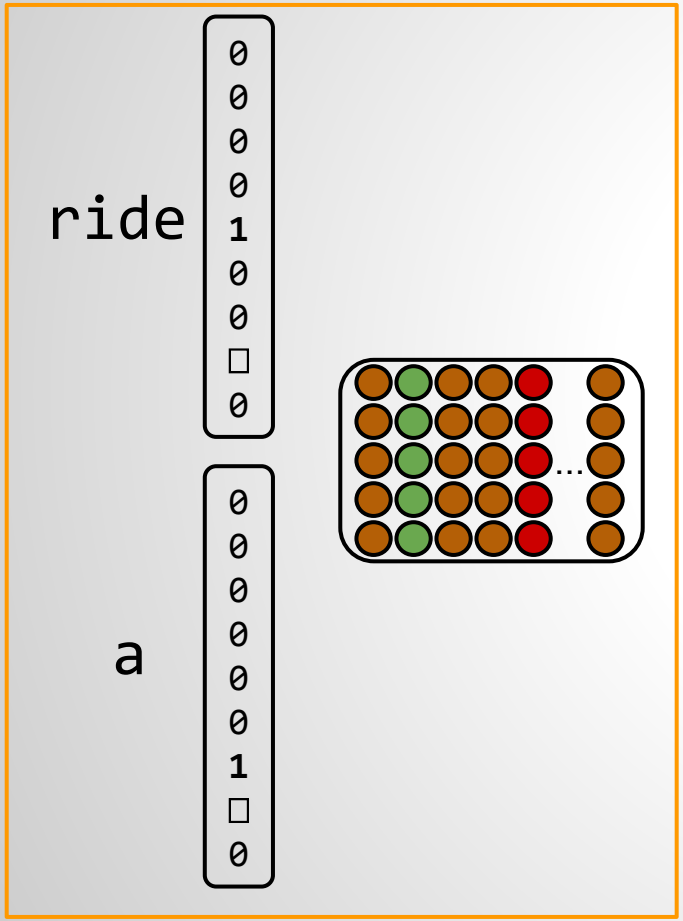
lookup table



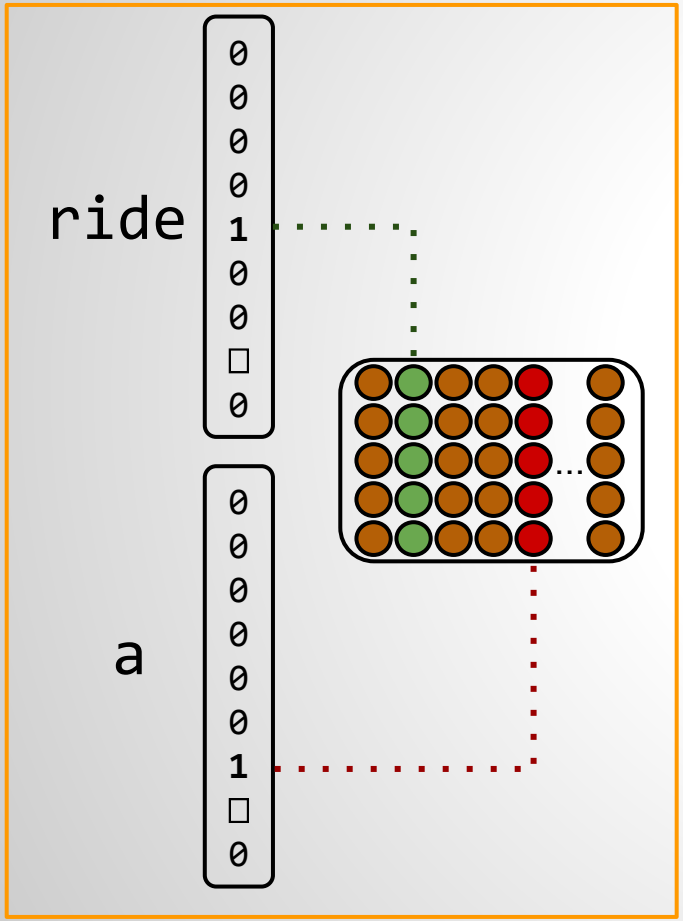
Neural language model



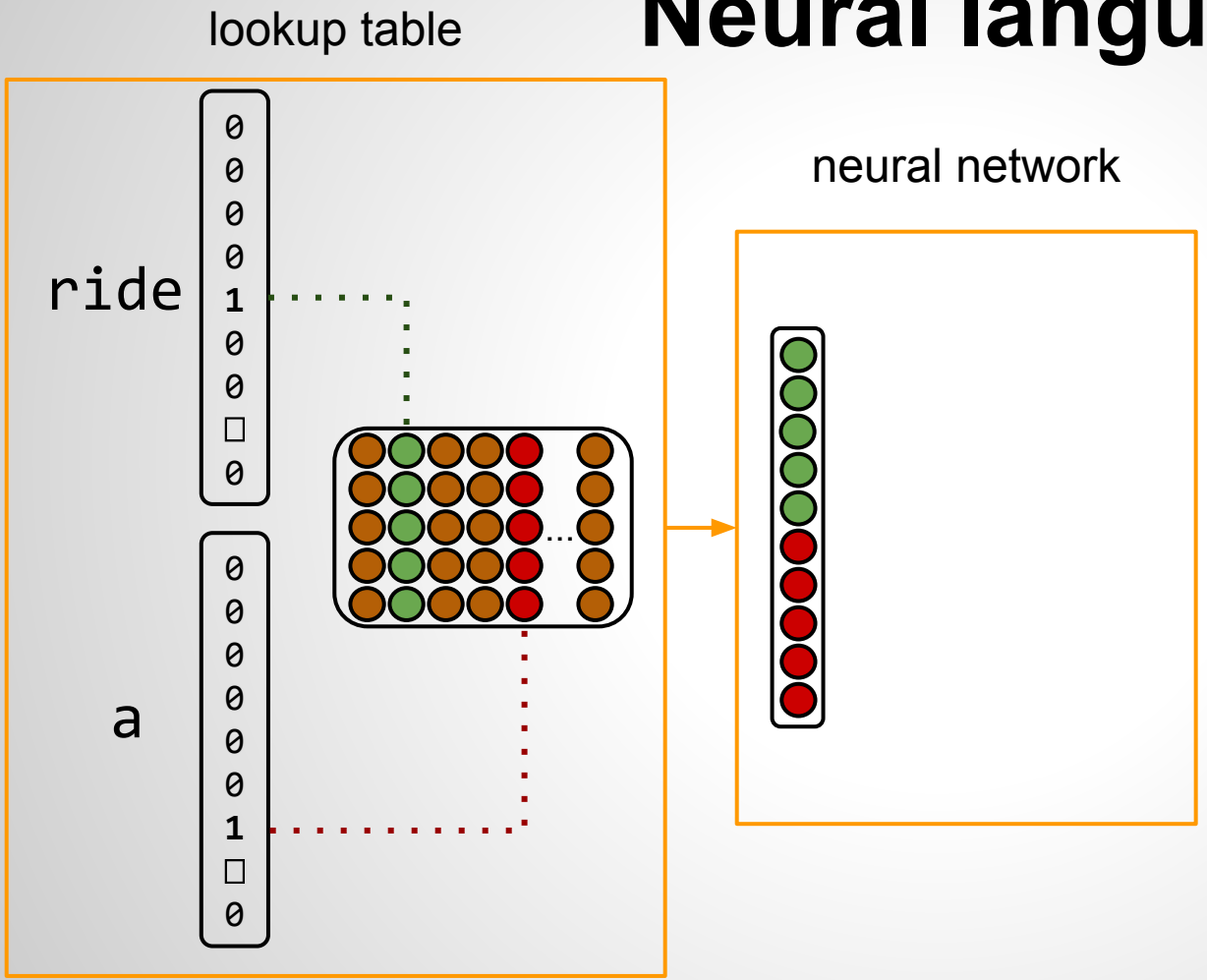
Neural language model



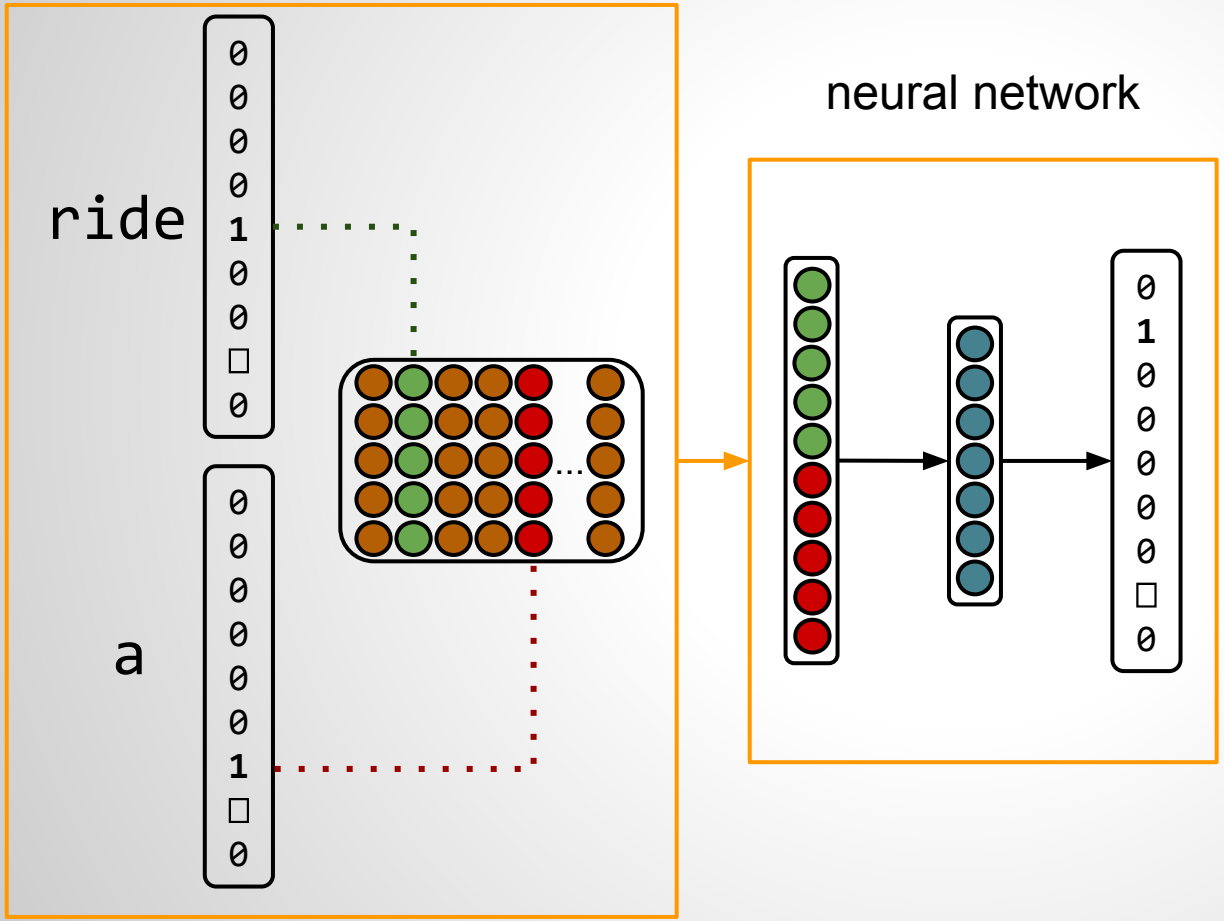
Neural language model



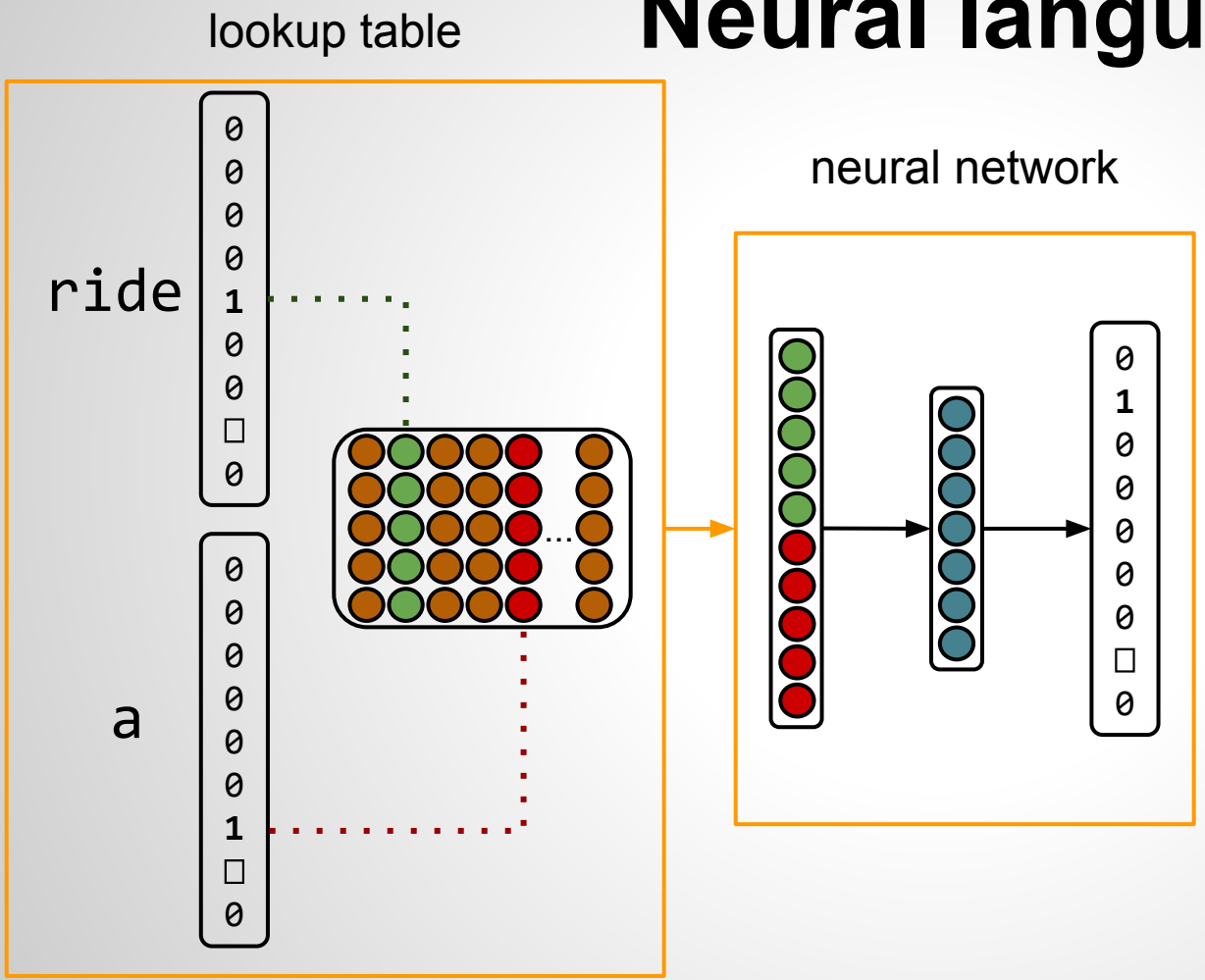
Neural language model



Neural language model



Neural language model



zebra should have representation similar to *horse* and *llama*

Now we can generalize to the unseen n-grams

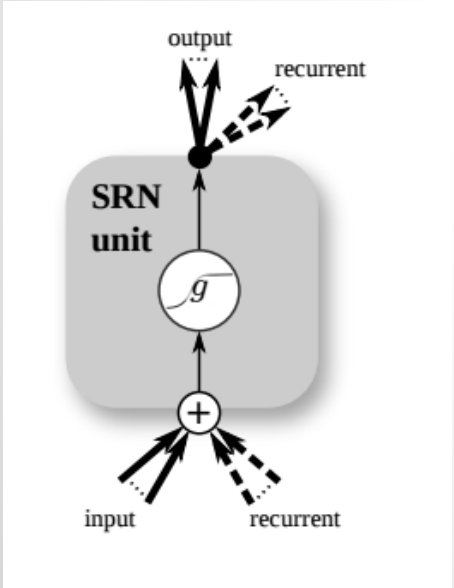
[ride a ***zebra***]

Recurrent neural network models

There is no Markov assumption

Recurrent neural network models

There is no Markov assumption



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

arxiv.org/pdf/1503.04069v1.pdf

Recurrent neural network models

yesterday

Recurrent neural network models

yesterday

Recurrent neural network models

yesterday

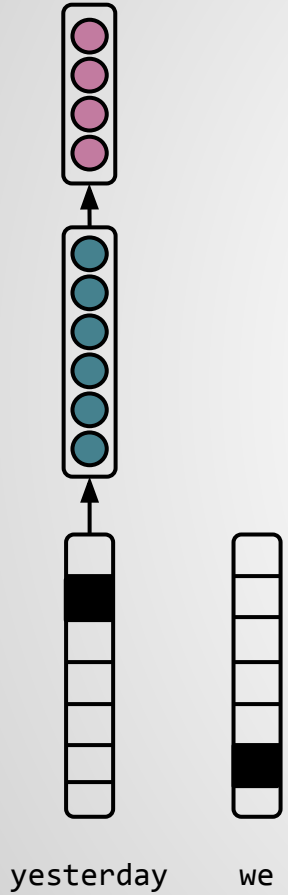
Recurrent neural network models

yesterday

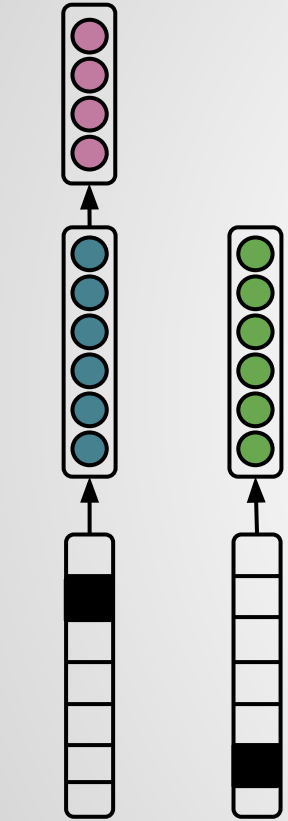
Recurrent neural network models

yesterday we

Recurrent neural network models

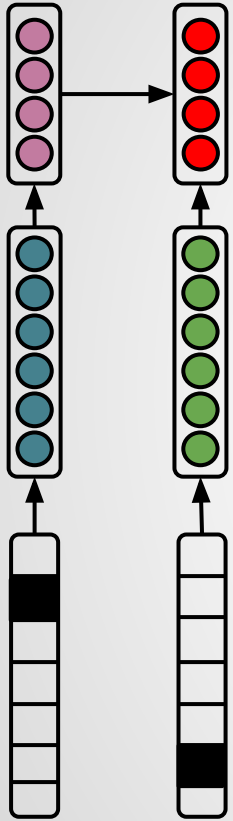


Recurrent neural network models



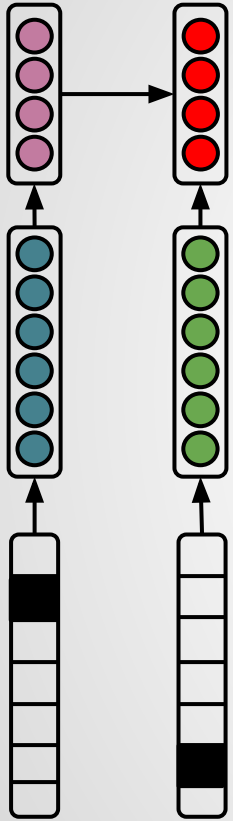
yesterday we

Recurrent neural network models



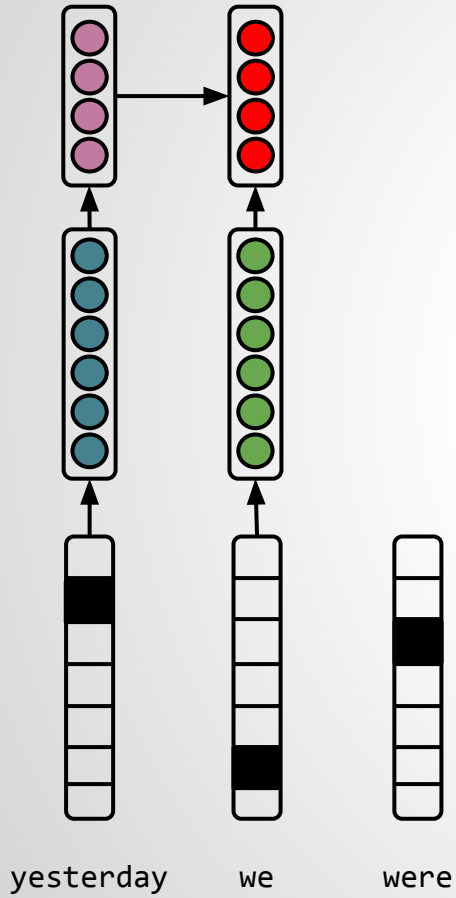
yesterday we

Recurrent neural network models

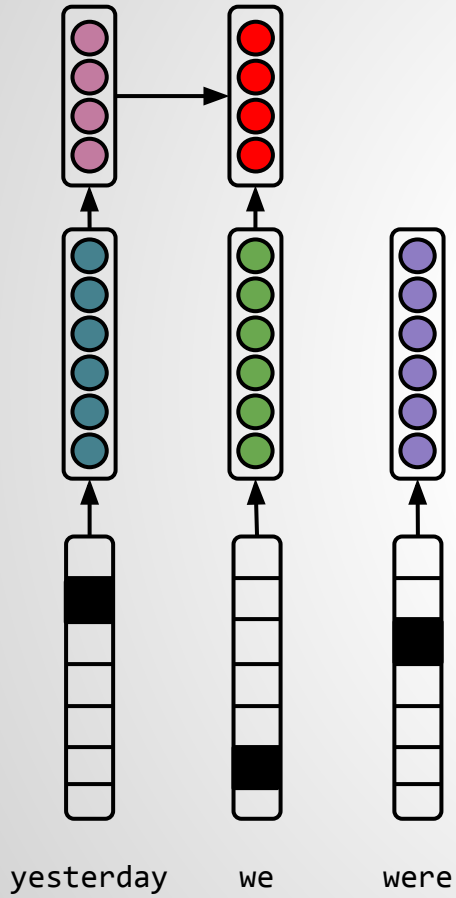


yesterday we were

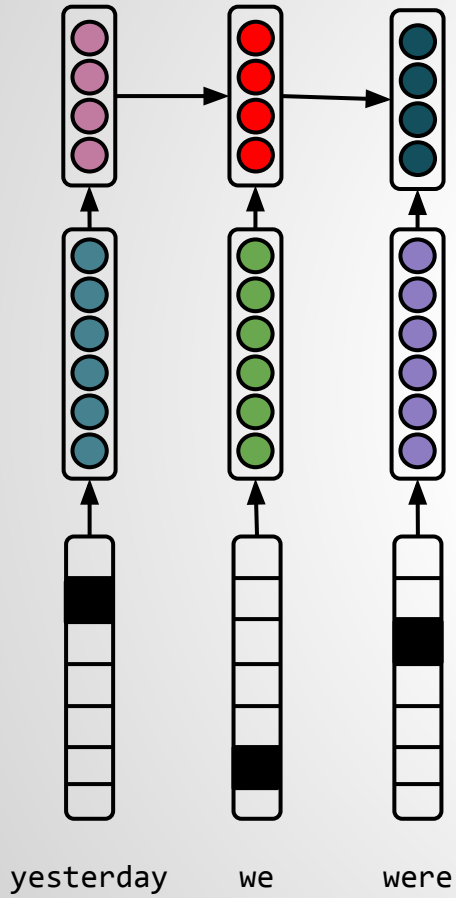
Recurrent neural network models



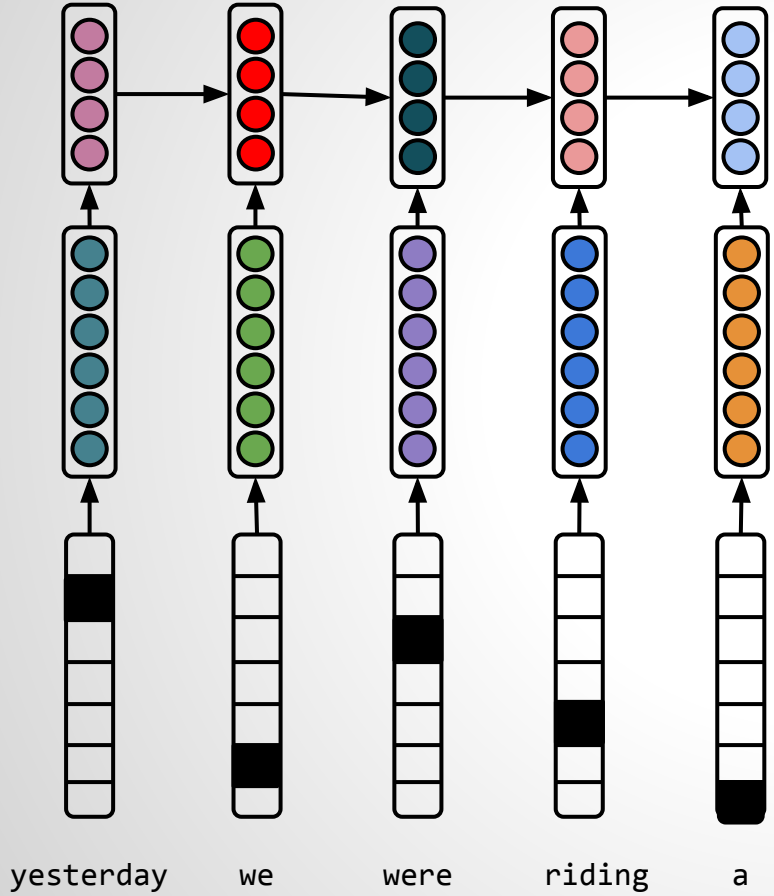
Recurrent neural network models



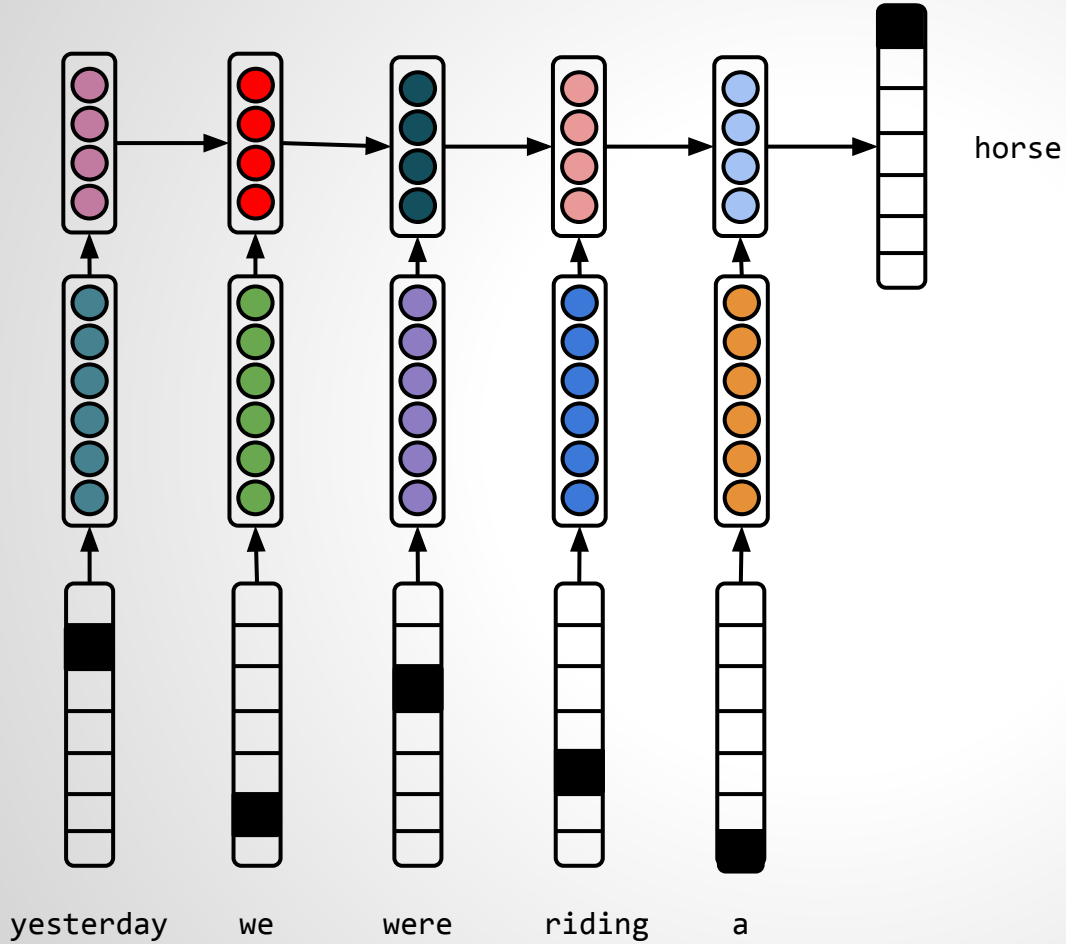
Recurrent neural network models



Recurrent neural network models



Recurrent neural network models

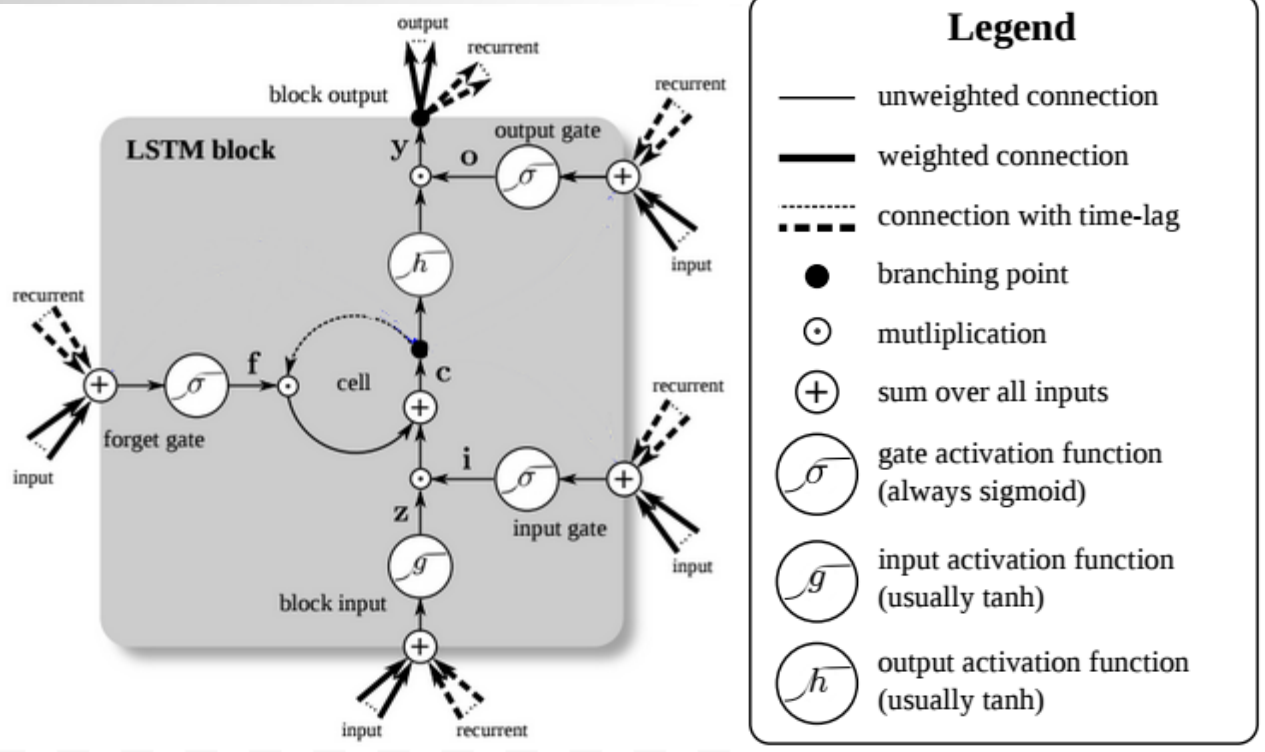


Recurrent neural network models

Vanishing/exploding gradient problem
www.jmlr.org/proceedings/papers/v28/pascanu13.pdf

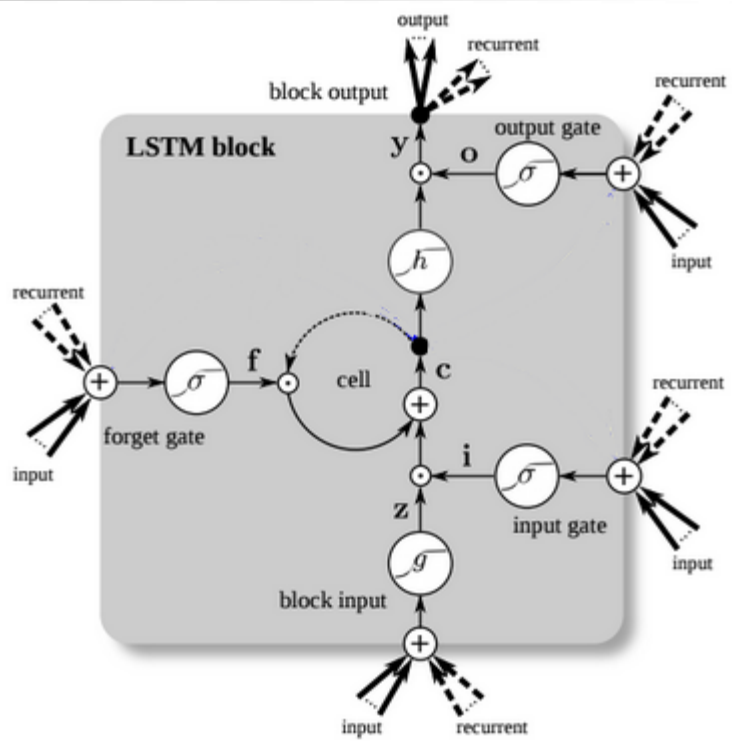
Naïve transition has difficulty in handling long-term dependencies

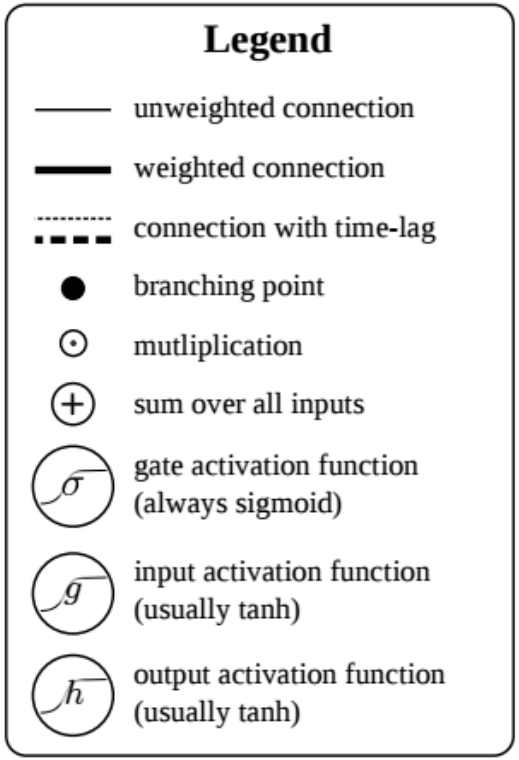
Long short-term memory rnn models



arxiv.org/pdf/1503.04069v1.pdf

Long short-term memory rnn models

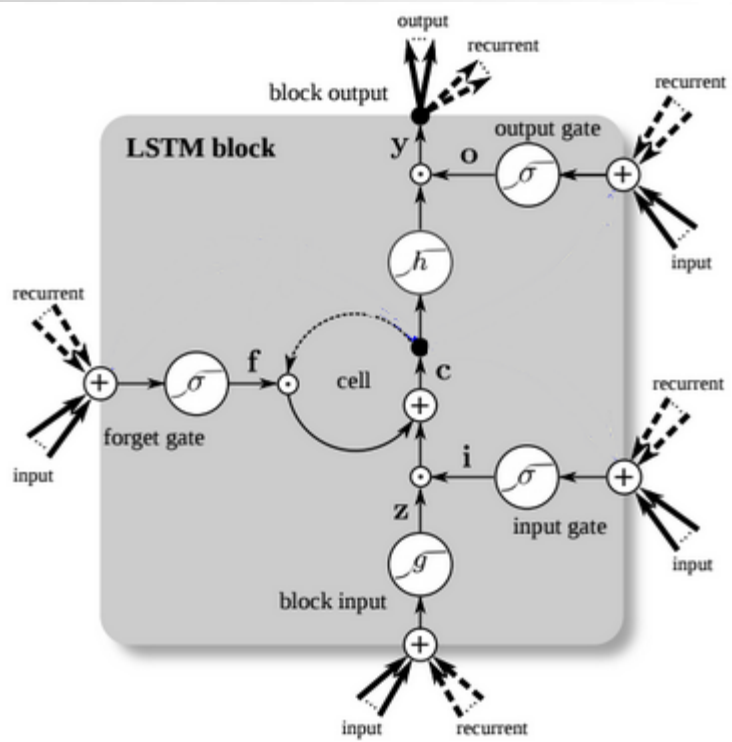


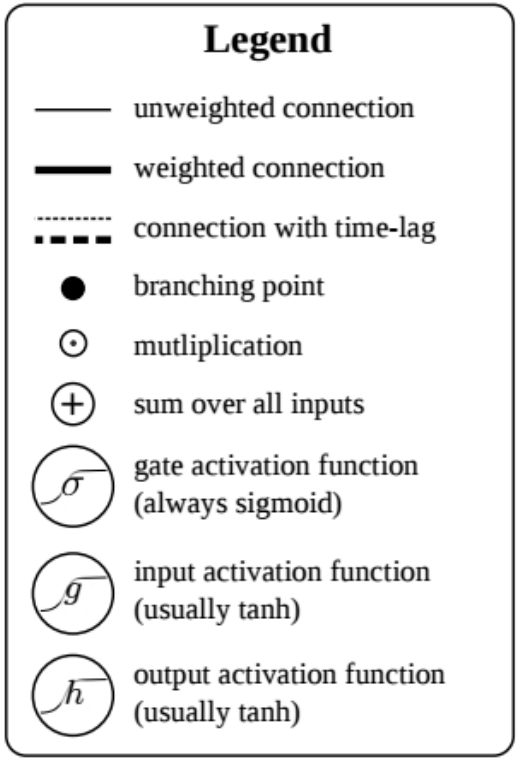


$$\mathbf{z}^t = g(\mathbf{W}_z \mathbf{x}^t + \mathbf{R}_z \mathbf{y}^{t-1})$$

arxiv.org/pdf/1503.04069v1.pdf

Long short-term memory rnn models



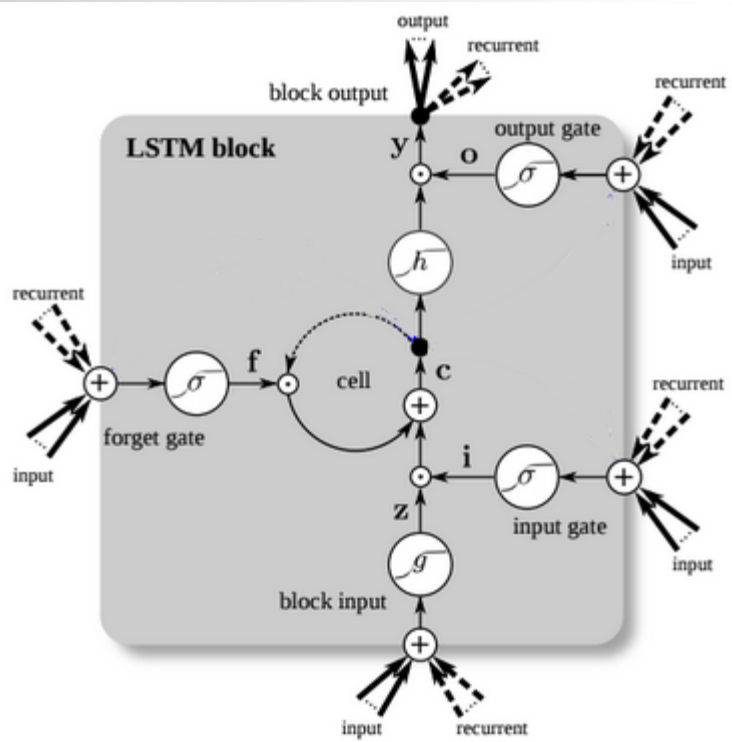


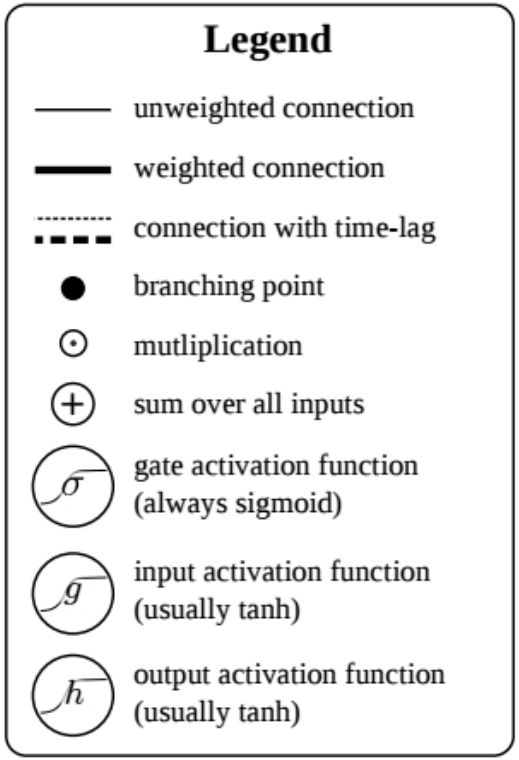
$$\mathbf{z}^t = g(\mathbf{W}_z\mathbf{x}^t + \mathbf{R}_z\mathbf{y}^{t-1})$$

$$\mathbf{i}^t = \sigma(\mathbf{W}_i\mathbf{x}^t + \mathbf{R}_i\mathbf{y}^{t-1})$$

arxiv.org/pdf/1503.04069v1.pdf

Long short-term memory rnn models

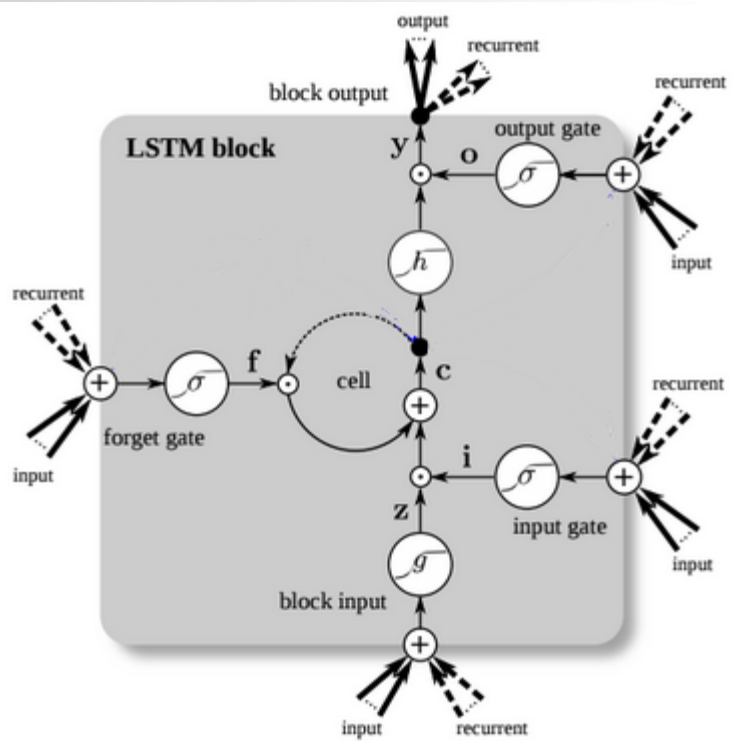




$$\mathbf{z}^t = g(\mathbf{W}_z\mathbf{x}^t + \mathbf{R}_z\mathbf{y}^{t-1})$$
$$\mathbf{i}^t = \sigma(\mathbf{W}_i\mathbf{x}^t + \mathbf{R}_i\mathbf{y}^{t-1})$$
$$\mathbf{f}^t = \sigma(\mathbf{W}_f\mathbf{x}^t + \mathbf{R}_f\mathbf{y}^{t-1})$$

arxiv.org/pdf/1503.04069v1.pdf

Long short-term memory rnn models



Legend

- unweighted connection
- weighted connection
- connection with time-lag
- branching point
- $\odot$ mutliplication
- $+$ sum over all inputs
- σ gate activation function (always sigmoid)
- g input activation function (usually tanh)
- h output activation function (usually tanh)

$$\mathbf{z}^t = g(\mathbf{W}_z\mathbf{x}^t + \mathbf{R}_z\mathbf{y}^{t-1})$$

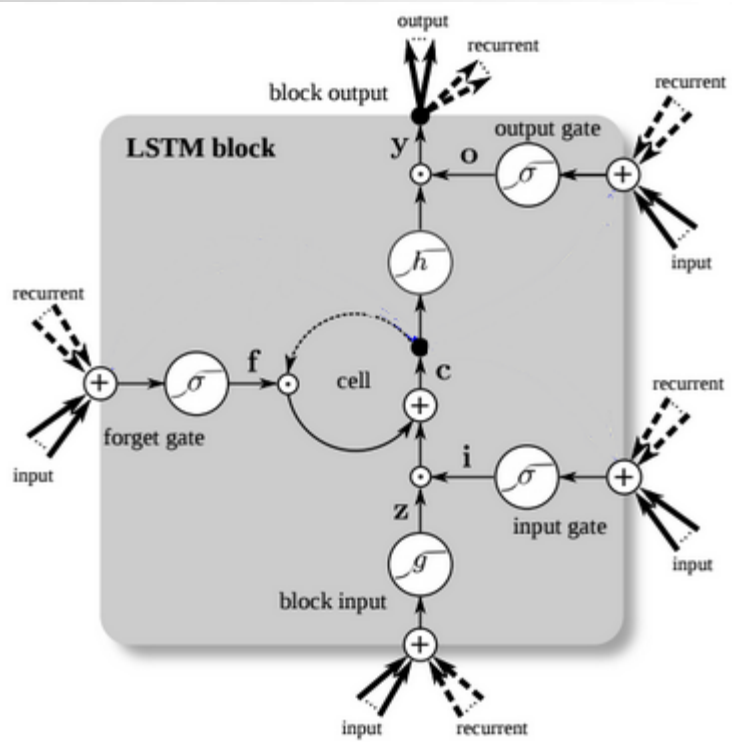
$$\mathbf{i}^t = \sigma(\mathbf{W}_i\mathbf{x}^t + \mathbf{R}_i\mathbf{y}^{t-1})$$

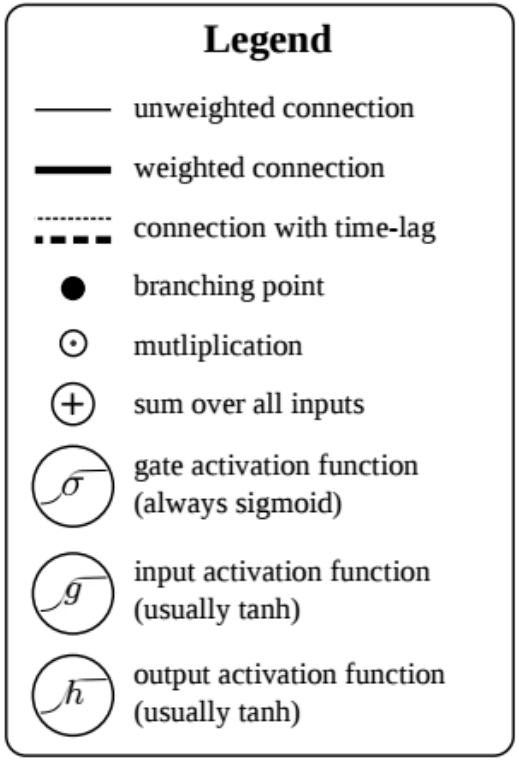
$$\mathbf{f}^t = \sigma(\mathbf{W}_f\mathbf{x}^t + \mathbf{R}_f\mathbf{y}^{t-1})$$

$$\mathbf{c}^t = \mathbf{i}^t \odot \mathbf{z}^t + \mathbf{f}^t \odot \mathbf{c}^{t-1}$$

arxiv.org/pdf/1503.04069v1.pdf

Long short-term memory rnn models

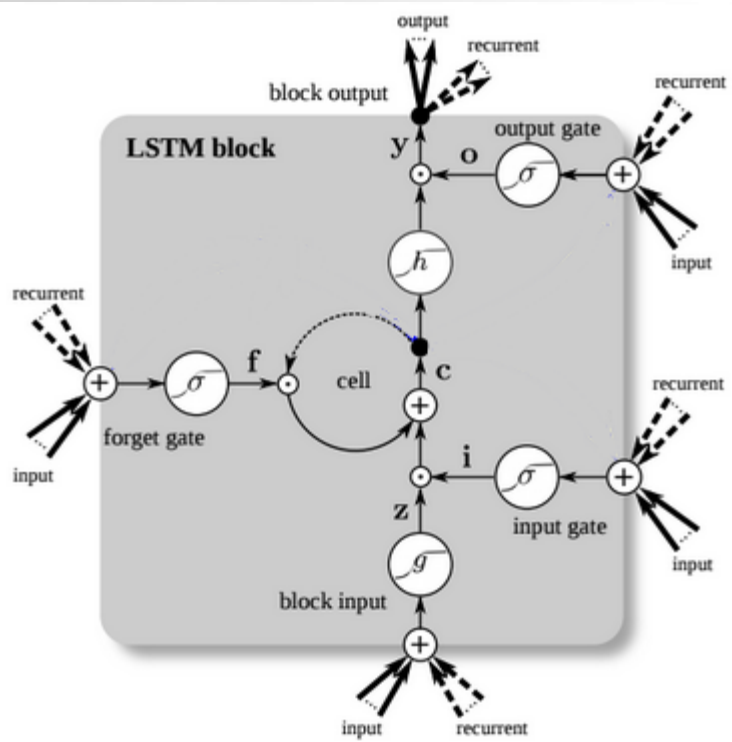


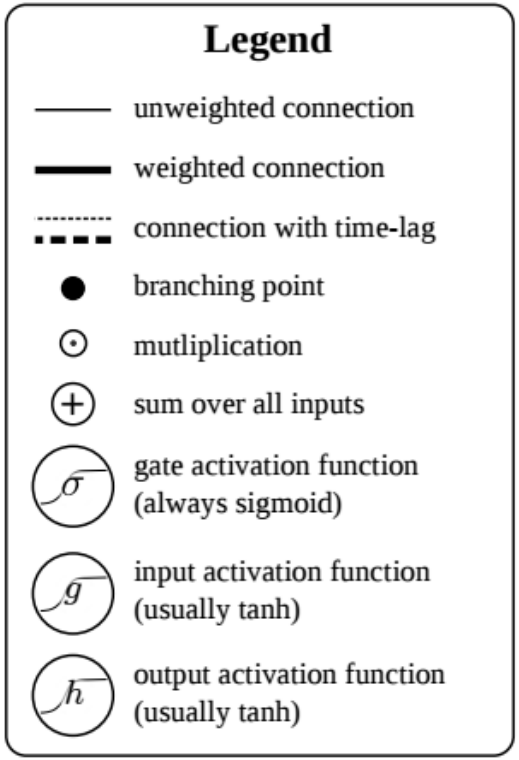


$$\mathbf{z}^t = g(\mathbf{W}_z\mathbf{x}^t + \mathbf{R}_z\mathbf{y}^{t-1})$$
$$\mathbf{i}^t = \sigma(\mathbf{W}_i\mathbf{x}^t + \mathbf{R}_i\mathbf{y}^{t-1})$$
$$\mathbf{f}^t = \sigma(\mathbf{W}_f\mathbf{x}^t + \mathbf{R}_f\mathbf{y}^{t-1})$$
$$\mathbf{c}^t = \mathbf{i}^t \odot \mathbf{z}^t + \mathbf{f}^t \odot \mathbf{c}^{t-1}$$
$$\mathbf{o}^t = \sigma(\mathbf{W}_o\mathbf{x}^t + \mathbf{R}_o\mathbf{y}^{t-1})$$

arxiv.org/pdf/1503.04069v1.pdf

Long short-term memory rnn models





$$\mathbf{z}^t = g(\mathbf{W}_z\mathbf{x}^t + \mathbf{R}_z\mathbf{y}^{t-1})$$

$$\mathbf{i}^t = \sigma(\mathbf{W}_i\mathbf{x}^t + \mathbf{R}_i\mathbf{y}^{t-1})$$

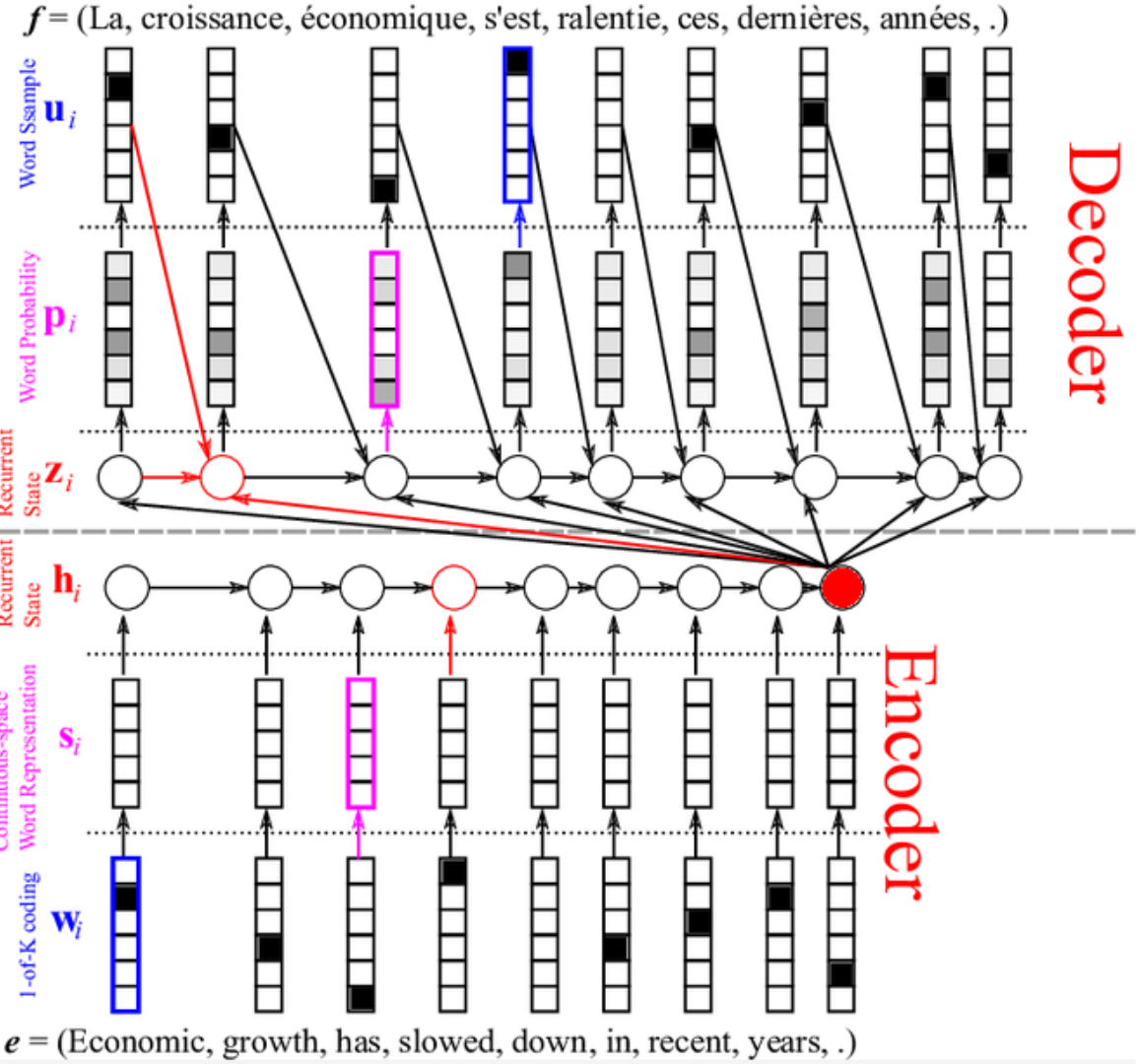
$$\mathbf{f}^t = \sigma(\mathbf{W}_f\mathbf{x}^t + \mathbf{R}_f\mathbf{y}^{t-1})$$

$$\mathbf{c}^t = \mathbf{i}^t \odot \mathbf{z}^t + \mathbf{f}^t \odot \mathbf{c}^{t-1}$$

$$\mathbf{o}^t = \sigma(\mathbf{W}_o\mathbf{x}^t + \mathbf{R}_o\mathbf{y}^{t-1})$$

$$\mathbf{y}^t = \mathbf{o}^t \odot h(\mathbf{c}^t)$$

arxiv.org/pdf/1503.04069v1.pdf



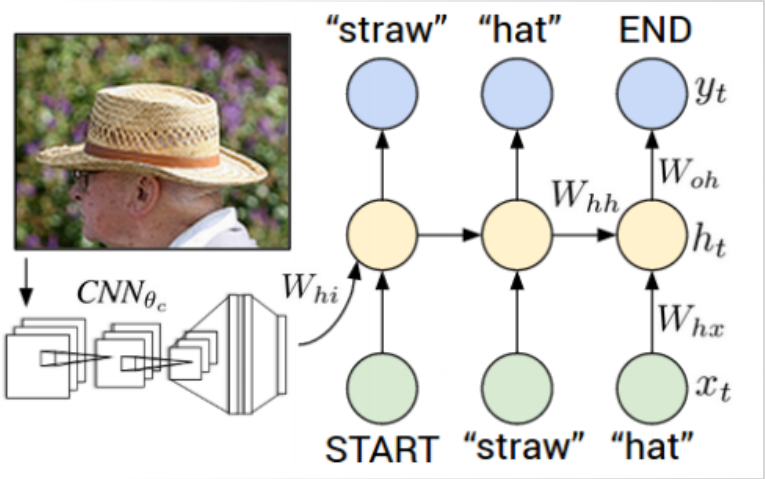
Neural machine translation

devblogs.nvidia.com/parallelforall/introduction-neural-machine-translation-with-gpus

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

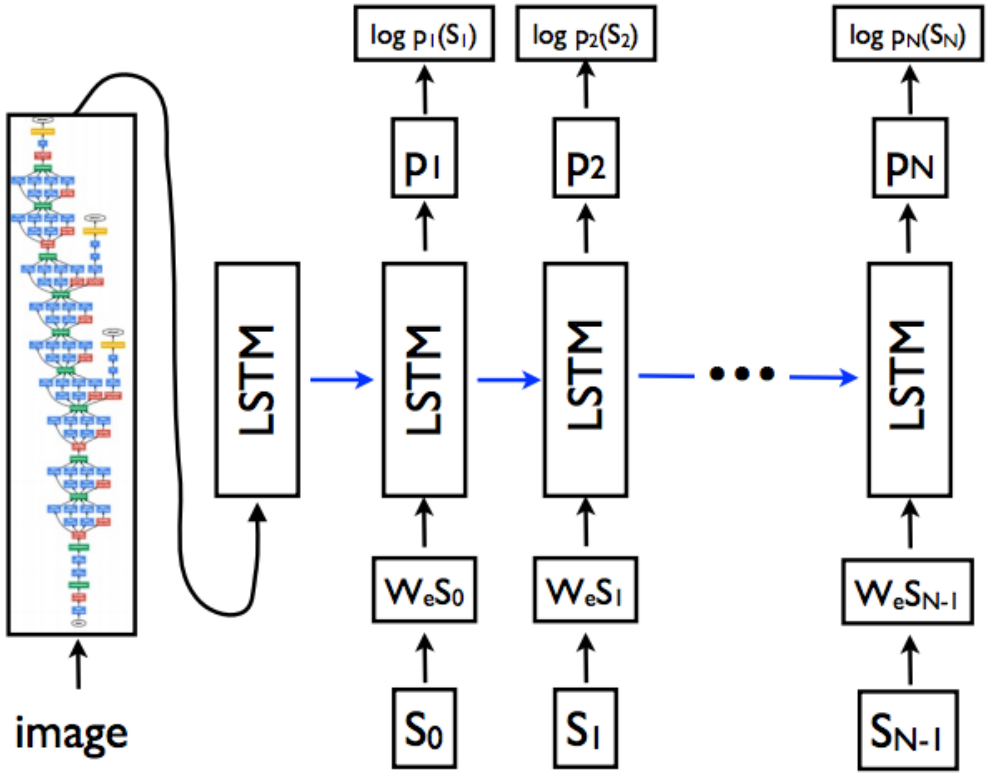
devblogs.nvidia.com/parallelforall/introduction-neural-machine-translation-gpus-part-3

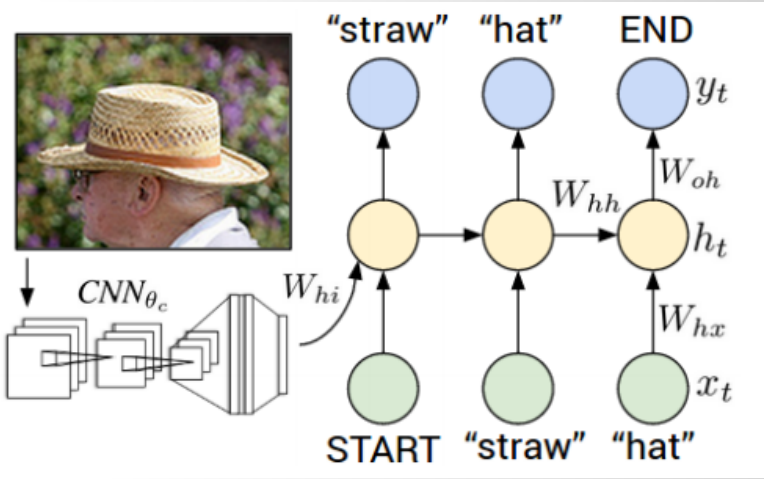
Image captioning



cs.stanford.edu/people/karpathy/cvpr2015.pdf

Image captioning





cs.stanford.edu/people/karpathy/cvpr2015.pdf
arxiv.org/pdf/1411.4555v2.pdf

Conclusion

- Neural methods provide us with a powerful set of tools for embedding language.
- They provide better ways of tying language learning to extra-linguistic contexts (images, knowledge-bases, cross-lingual data).

CS224d: Deep Learning for Natural Language Processing
cs224d.stanford.edu