

Predicting Sequential Data with LSTMs augmented with SP2

Chihiro Shibata
Tokyo University of
Technology

Jeffrey Heinz
University of Delaware

Back Ground

- Deep learning technology has developed dramatically.

- Various kinds of applications:

- Image Recognition, natural language processing etc.

- To predict sequential data:

- Recurrent neural networks

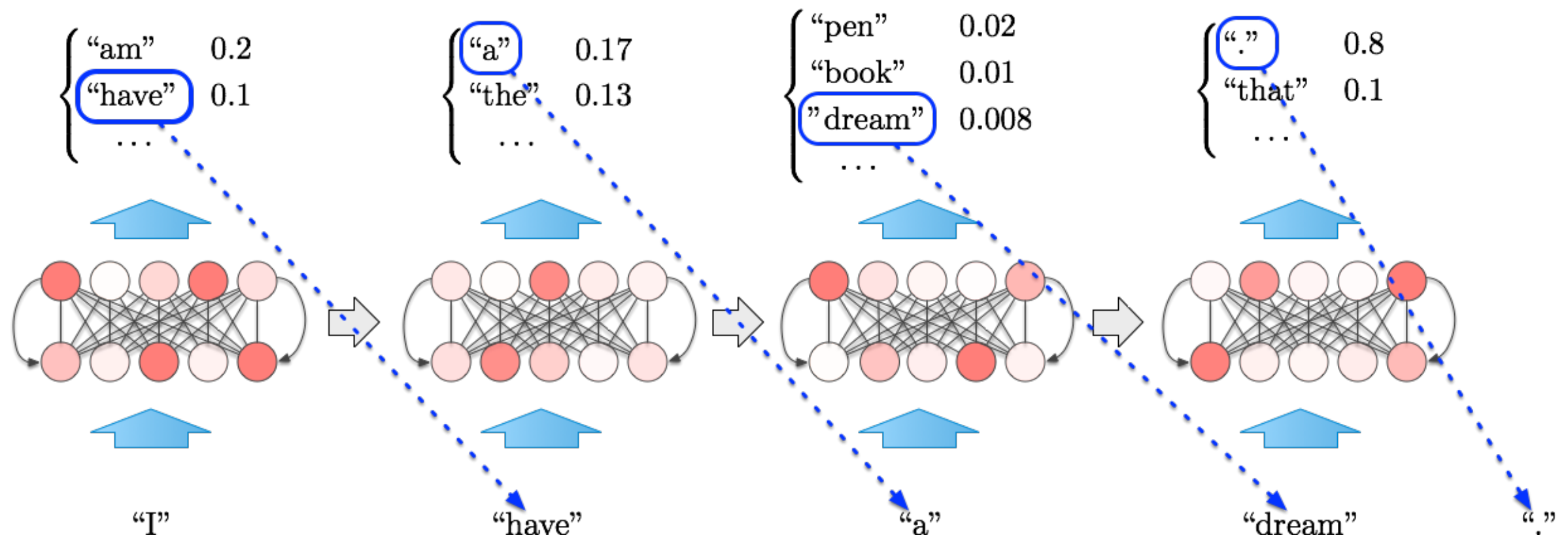
- Long-Short Term Memory (LSTM)

- Gated Recurrent Unit

- Those are carefully constructed so that the models can capture long-distance dependencies in principle.

Recurrent Neural Networks

Recurrent neural networks are empirically known to be useful when next elements in time-series data are required to be predicted. e.g.) sentence generation:

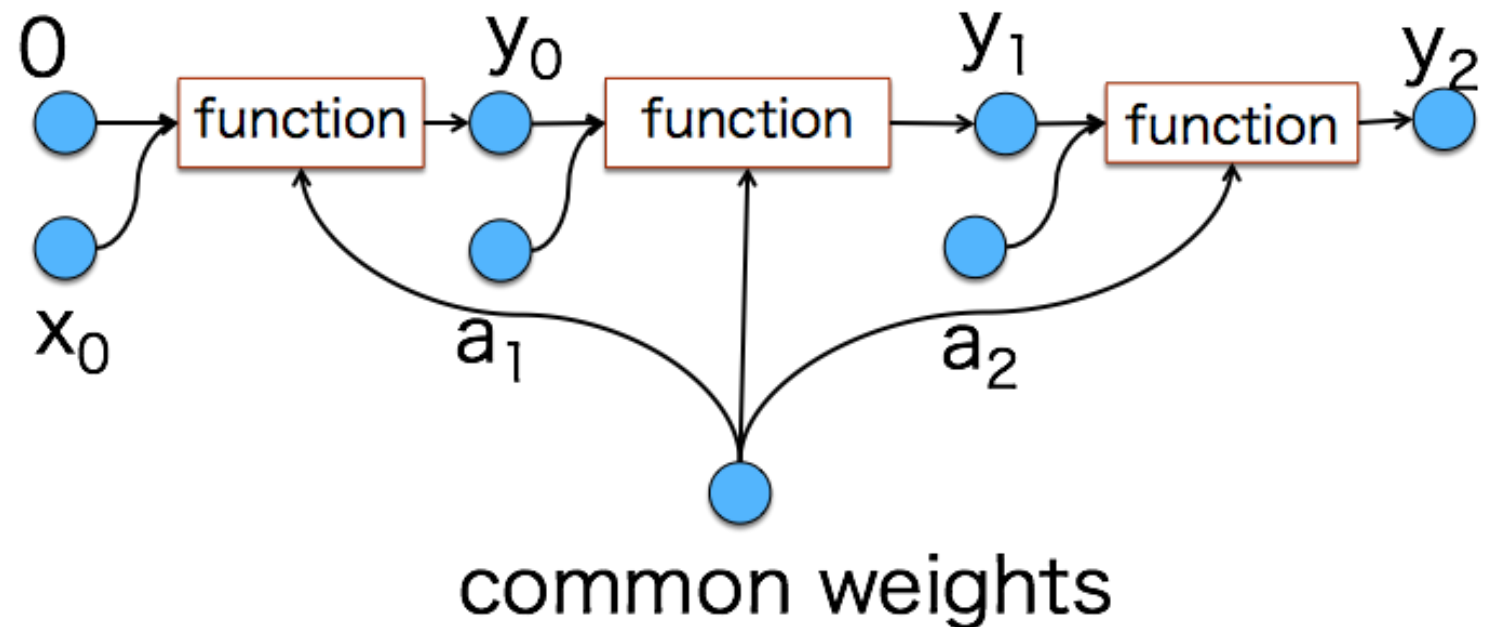
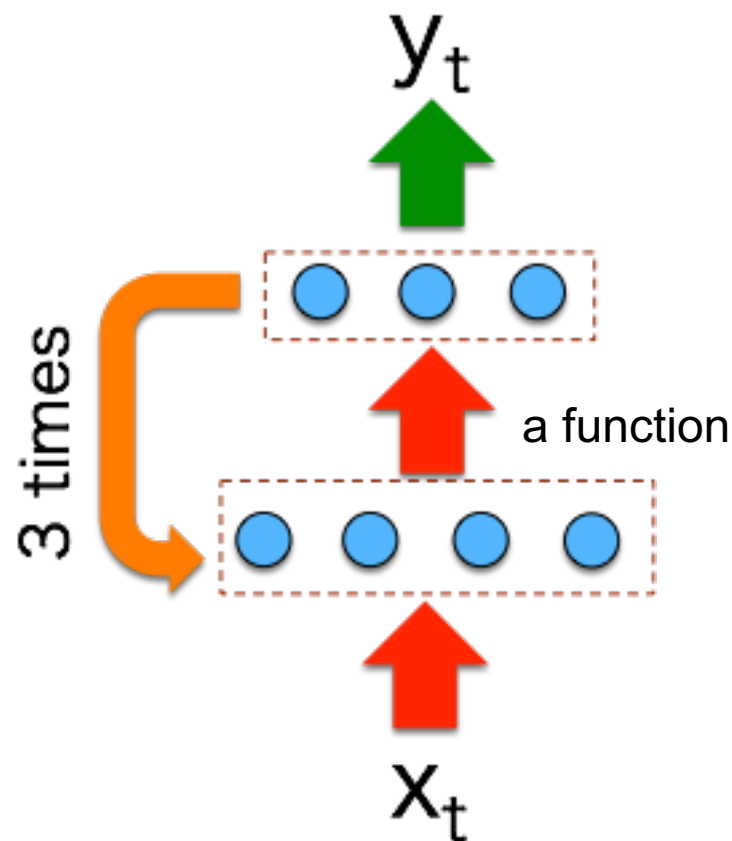


Recurrent Neural Networks

The main difference from usual neural networks:

Weights are forced to be invariant with respect to time.

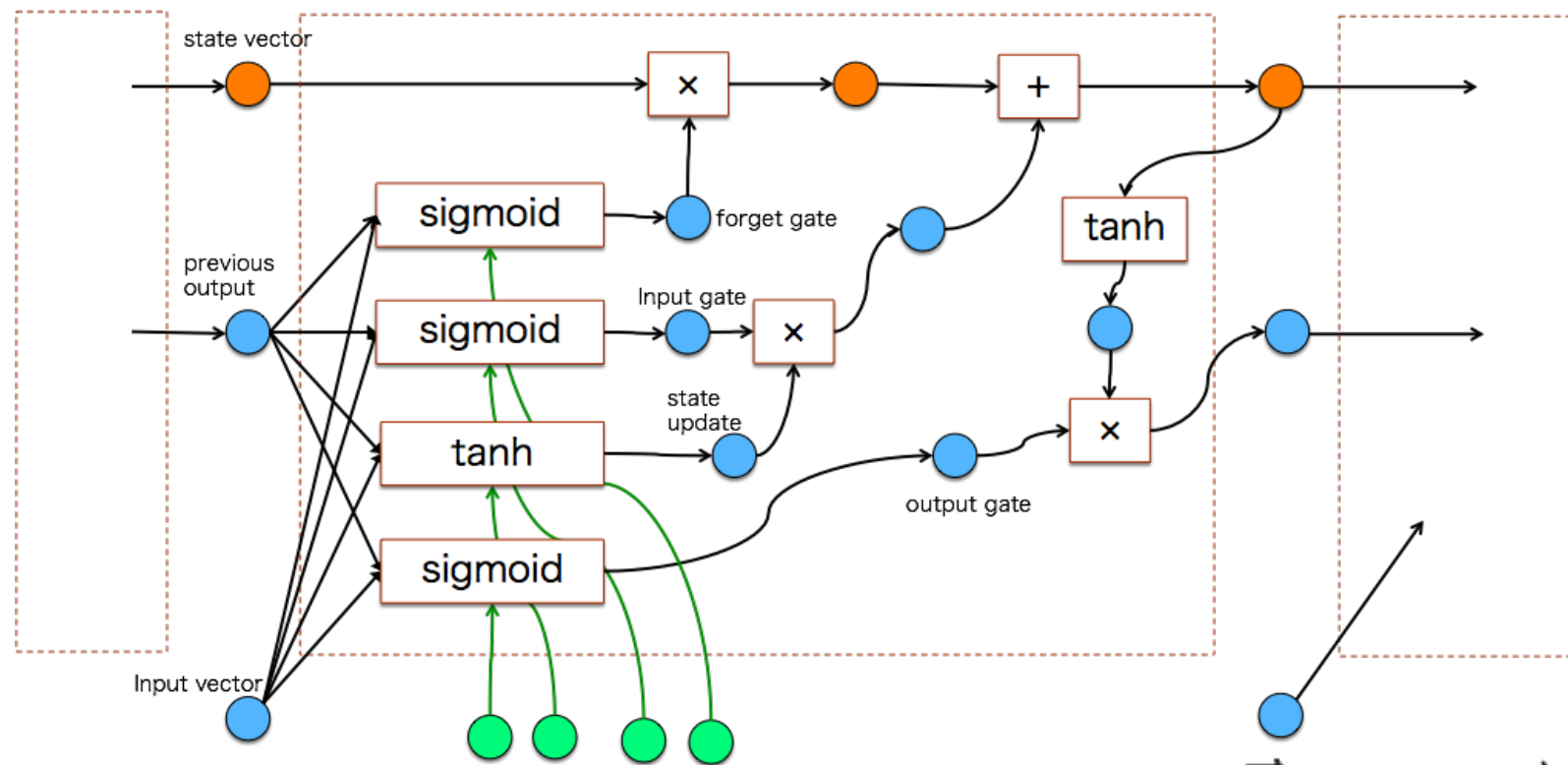
e.g.) a time series (x_0, x_1, x_2) is input to a RNN:



Long Short-term Memory (LSTM)

■ LSTM is a kind of RNN designed to track long-term dependencies.

■ It has a redundant hidden output which is called “the state vector”.



$$\begin{aligned}\vec{f} &= \sigma(W_f[x, h] + b_f) \\ \vec{i} &= \sigma(W_i[x, h] + b_i) \\ \Delta\vec{c} &= \tanh(W_o[x, h] + b_o) \\ \vec{o} &= \sigma(W_o[x, h] + b_o)\end{aligned}$$

$$\begin{aligned}\vec{c}_{\text{new}} &= \vec{f} \odot \vec{c} + \vec{i} \odot \Delta\vec{c} \\ \vec{h}_{\text{new}} &= \vec{o} \odot \tanh(\vec{c}_{\text{new}})\end{aligned}$$

Motivation and Key Idea

Disadvantage of LSTM:

- Long-distance dependencies are implicitly involved in somewhere in the network weights.

Our motivation:

- Knowledge of formal languages and grammars:

- Exploit them to improve neural networks.

Key Idea:

- SP-k languages proposed Heinz et. al describe certain kinds of long-distance dependencies.

- Combine the standard DNN structure using LSTM with SP-k.

Strictly Piecewise Languages

- A subclass of regular languages.



- Capture certain kinds of long-term dependencies.



- L is SP iff there is a finite set S such that no w in L contains any string in S as a *subsequence* (so finitely many forbidden subsequences)

- Long-term dependency in phonotactic is known to be captured:

- e.g.: Sarcee language (one of the native american languages)

- [-anterior] sibilants like [ʃ] and [ʒ] regressively require [+anterior] sibilants like [s] and [z], but not vice versa. *

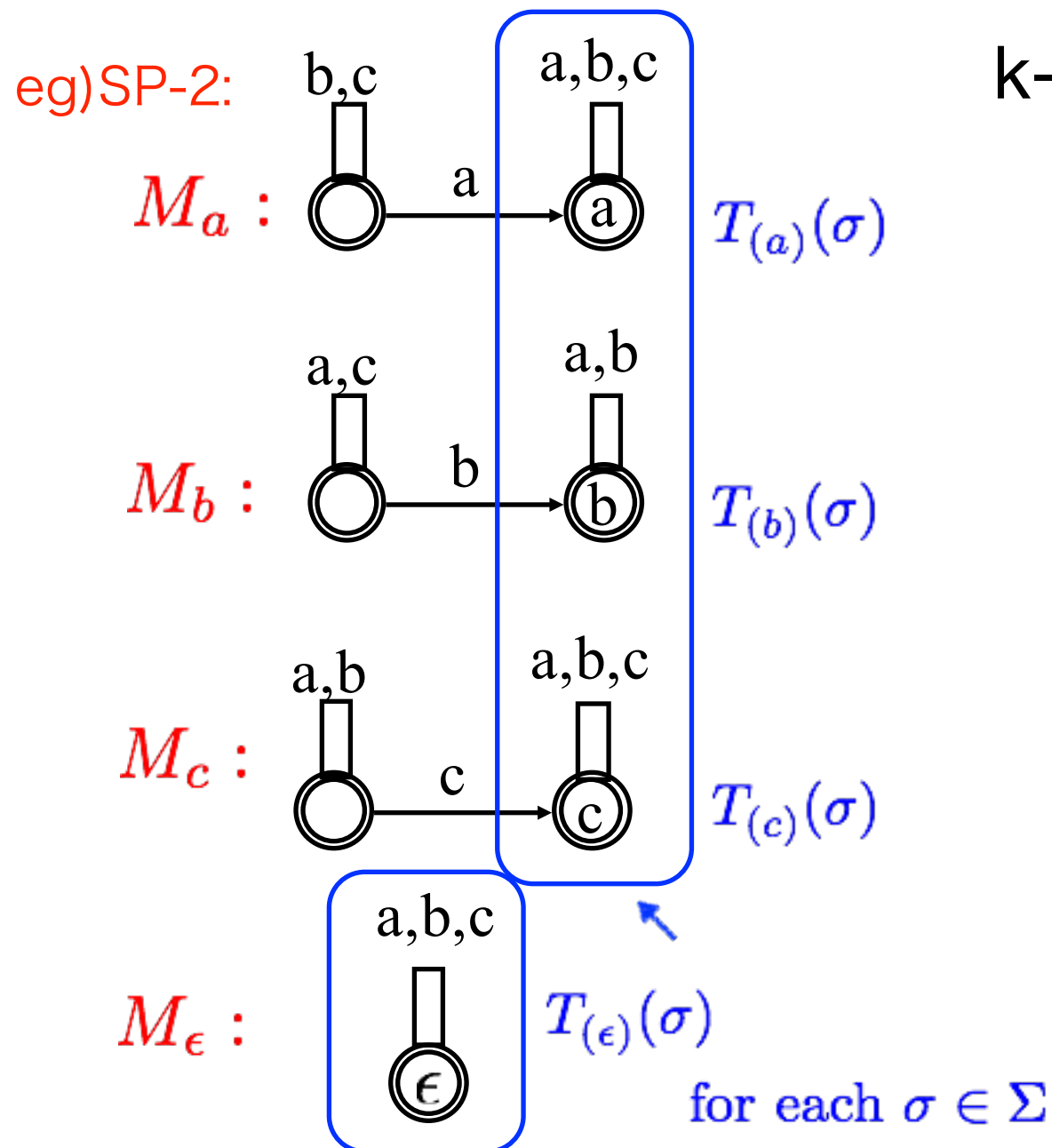
OK  a. /si-tʃiz-aʔ/ → ʃítʃídʒàʔ ‘my duck’
b. /na-s-ʏatʃ/ → nāʃʏátʃ ‘I killed them again’ 

NG  c. cf. *sítʃídʒàʔ 

* James Rogers, Jeffrey Heinz, Gil Bailey, Matt Edlefsen, Molly Visscher, David Wellcome, and Sean Wibel. On Languages Piecewise Testable in the Strict Sense. The Mathematics of Language, volume 6149 of Lecture Notes in Artificial Intelligence, pages 255-265. Springer, 2010.

Strictly Piecewise-k (SP-k)

SP-k language L can be represented by a list of k -SP-DFAs, whose DFA product equals L



k -SP-DFAs : $\{ M_1, \dots, M_k \}$

This representation can be exponentially smaller than the minimal DFA representation of L .

This representation can also be used to define distributions via **co-emission**.

E.g.: co-emission for SP-2

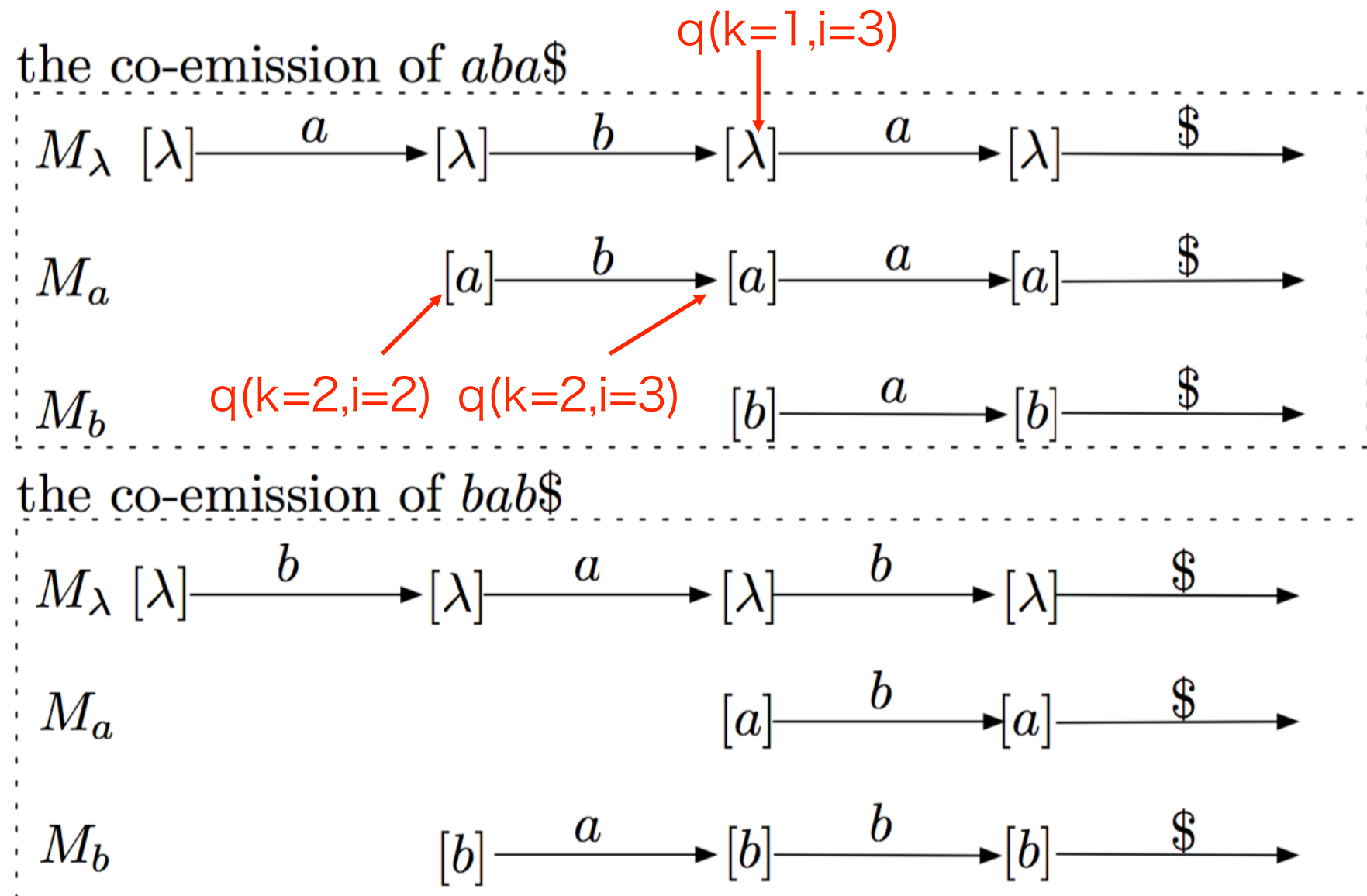
Alphabet :

$$\Sigma = \{a, b, \$\}$$

There are three 2-SD-DFAs:

$$\{M_\lambda, M_a, M_b\}$$

$q(k,i)$: the state of
k-th DFA after fed
the prefix
whose length is i-1.



SP-k vector representation:

e.g.

$\Sigma = \{a, b, c, d, e\}$

prefix: $x = a d a c d$

The SP-2 vector for prefix x :

a	b	c	d	e
1	0	1	1	0

The SP-3 vector (or tensor) for x :

	a	b	c	d	e
a	1	b	1	1	d
b	a	b	c	d	d
c	a	b	c	d	d
d	1	b	1	1	d
e	a	b	c	d	d

The num of 1s:

SP-2: $\mathcal{O}(\max\{|\Sigma|, |w|\})$

SP-3: $\mathcal{O}(\max\{|\Sigma|^2, |w|^2/2\})$

Preliminary Experiments

Purpose:

Check whether the SP-k vector can capture long-term dependencies and is effective for the next-element prediction.

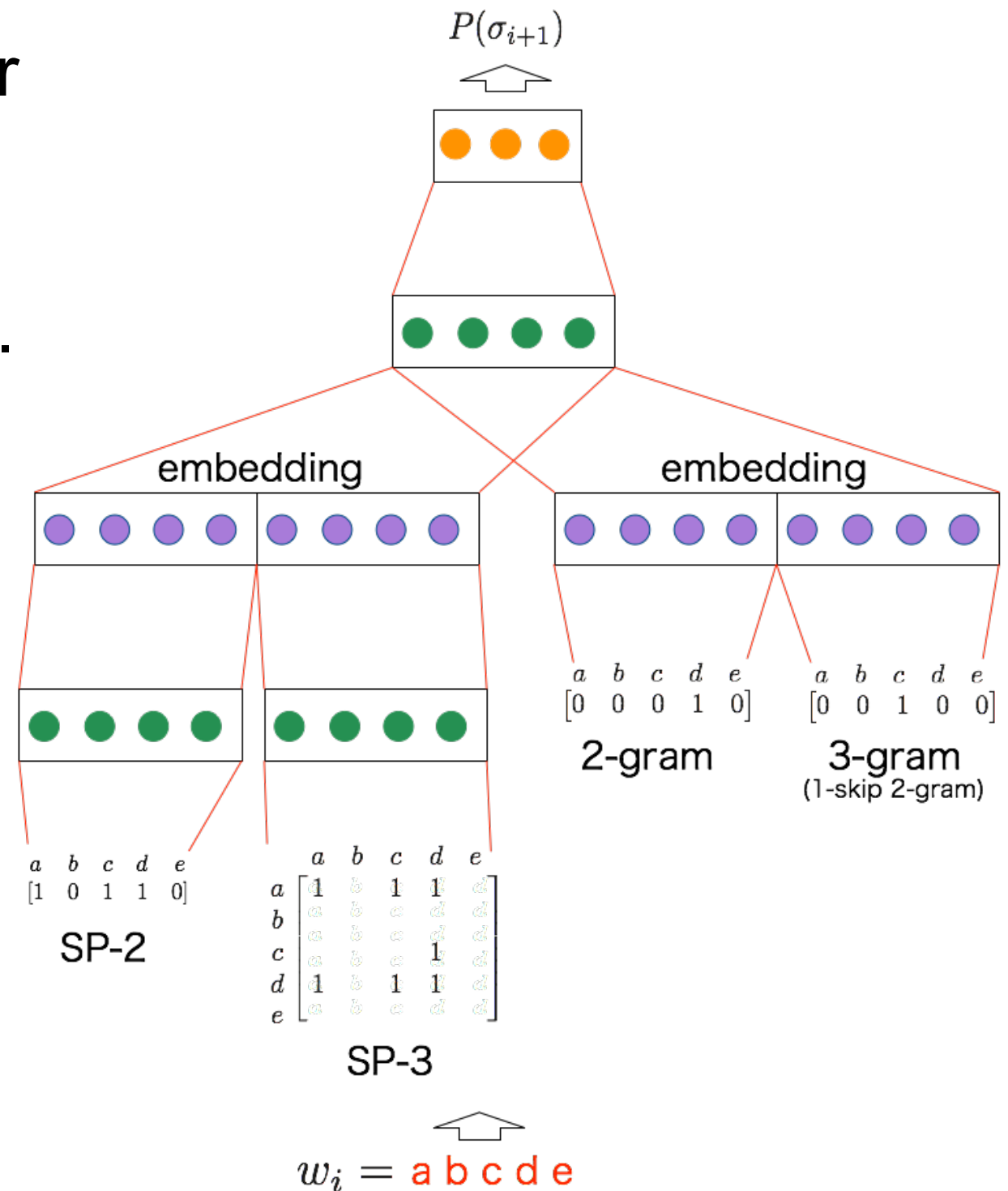
Two types of Input:

n-gram :

adjacent n letters.

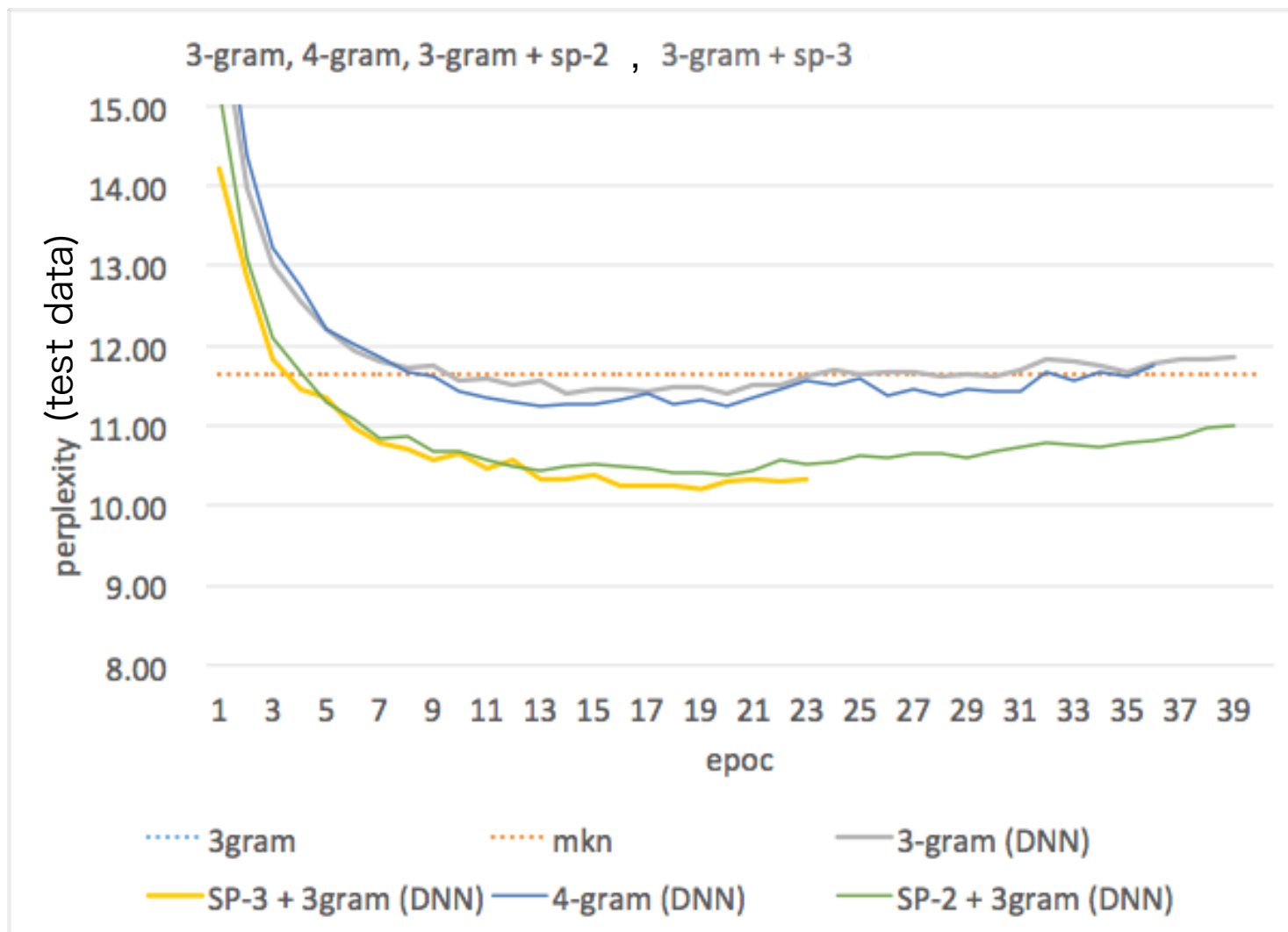
SP-k :

long-term dependency.



Result of Preliminary Experiments

Data: Brown corpus (1000 sentences).



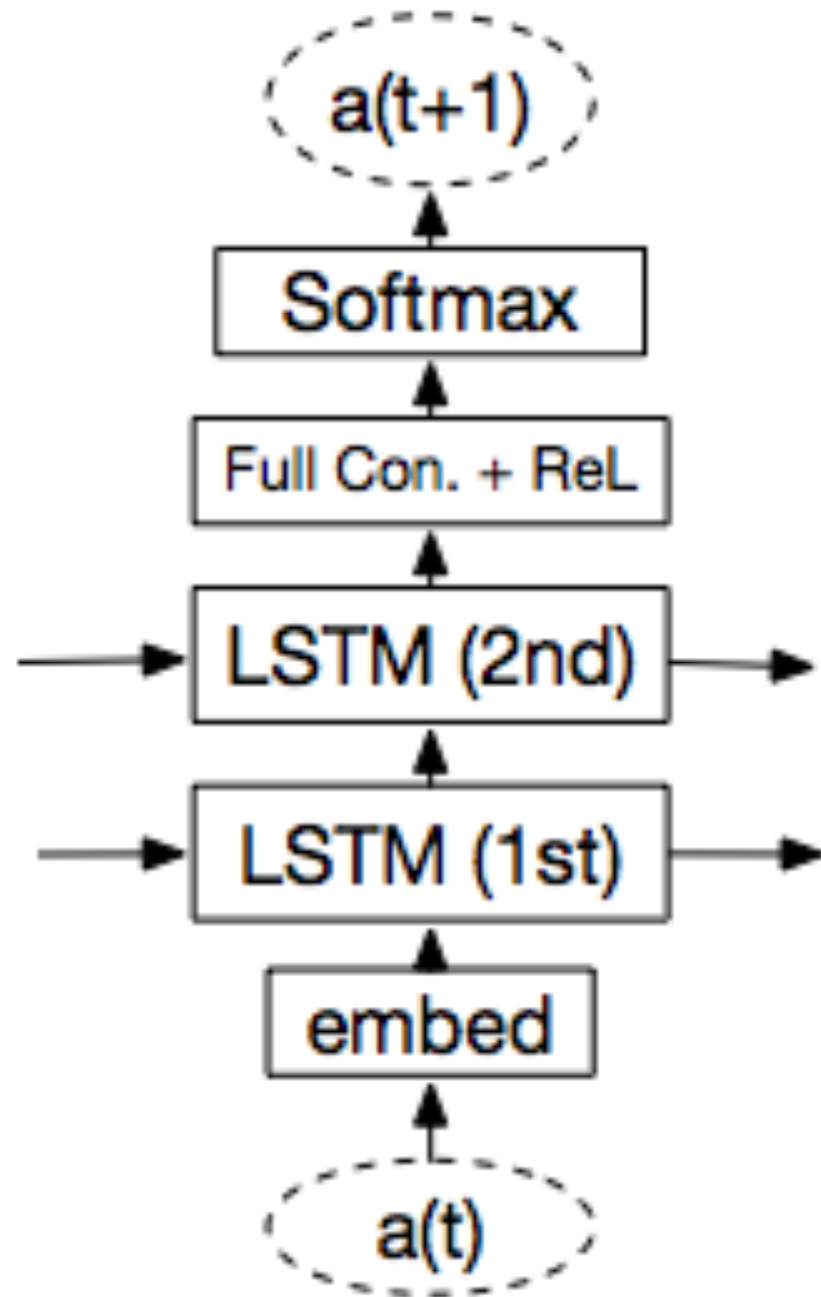
■ SP-2 + n-grams was better than MKN as well as n-grams.

■ SP-k vectors was clearly effective to have better prediction.

mkn	3-gram	4-gram	SP-2 + 3-gram	SP-2 + 4-gram	SP-3 + 3-gram
11.64	11.39	11.24	10.37	10.40	10.20

SP-3 + 3-gram

Basic model: Two-layered LSTM



■ The network predicts the next letter of the sequence ($a(t+1)$).

■ Each input letter is converted to a real valued vector in the embed layer.

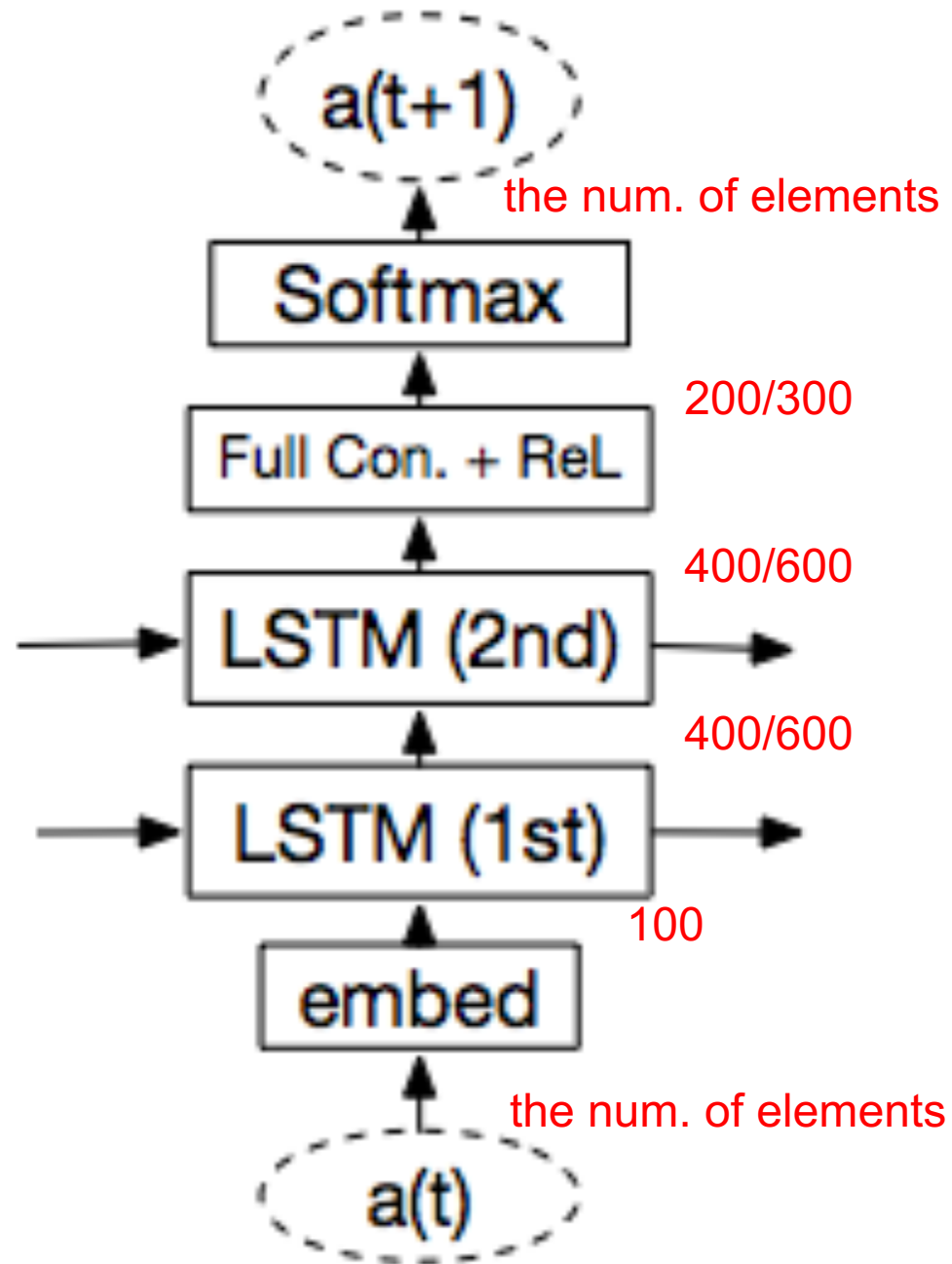
■ The output is the probabilities of the next letters.

■ Two-layered LSTM network.

■ Information of the prefix is kept through this layer.

■ Non-linear layer on top of it.

Basic model: details of the network



■ The number of LSTM layers were decided through experiments.

■ Using single layer LSTM decrease the prediction score.

■ Three layers did not improve the score so much, but worsen the computational cost.

■ The number of the dimension of each layer was also decided with experimental validation.

■ Two sets of numbers of the dimension are compared.

Strictly Piecewise Model

■ We concatenate the output of the LSTM network and the SP-2 vector.

■ SP-2 vector:

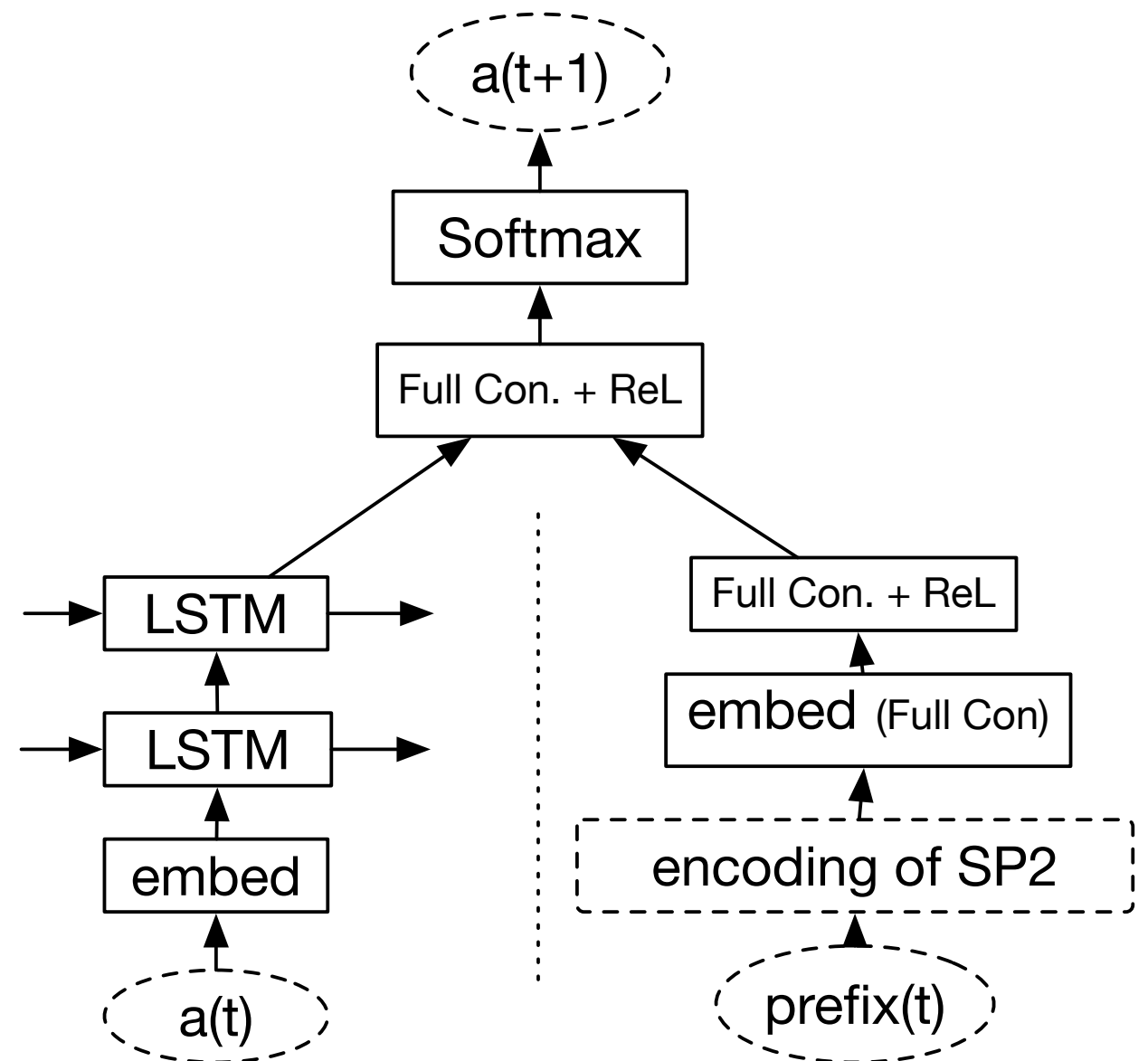
■ Encode the prefix into the SP-2 zero-one vector.

■ Embed into a real-valued vector through a fully connected layer

■ As an intermediate layer, use another fully connected layer with a non-linear activation function.

■ LSTM network:

■ the same as in the basic model.



Strictly Piecewise Model: details of the network

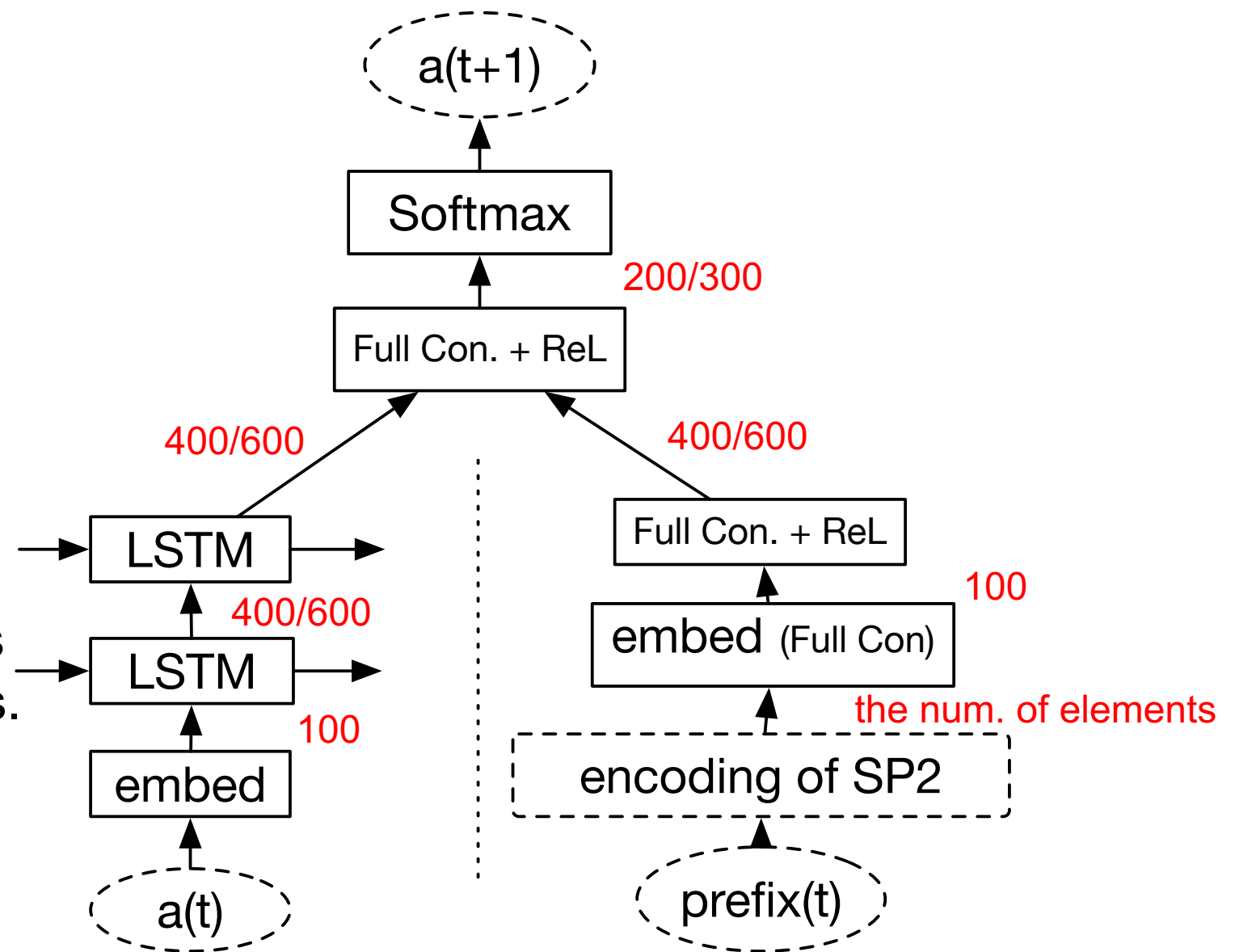
■ For SP-2 vector,
■ layers and
■ the number of the dimension
of each layer
was decided in similar ways.

➡ There may be better network
structures.

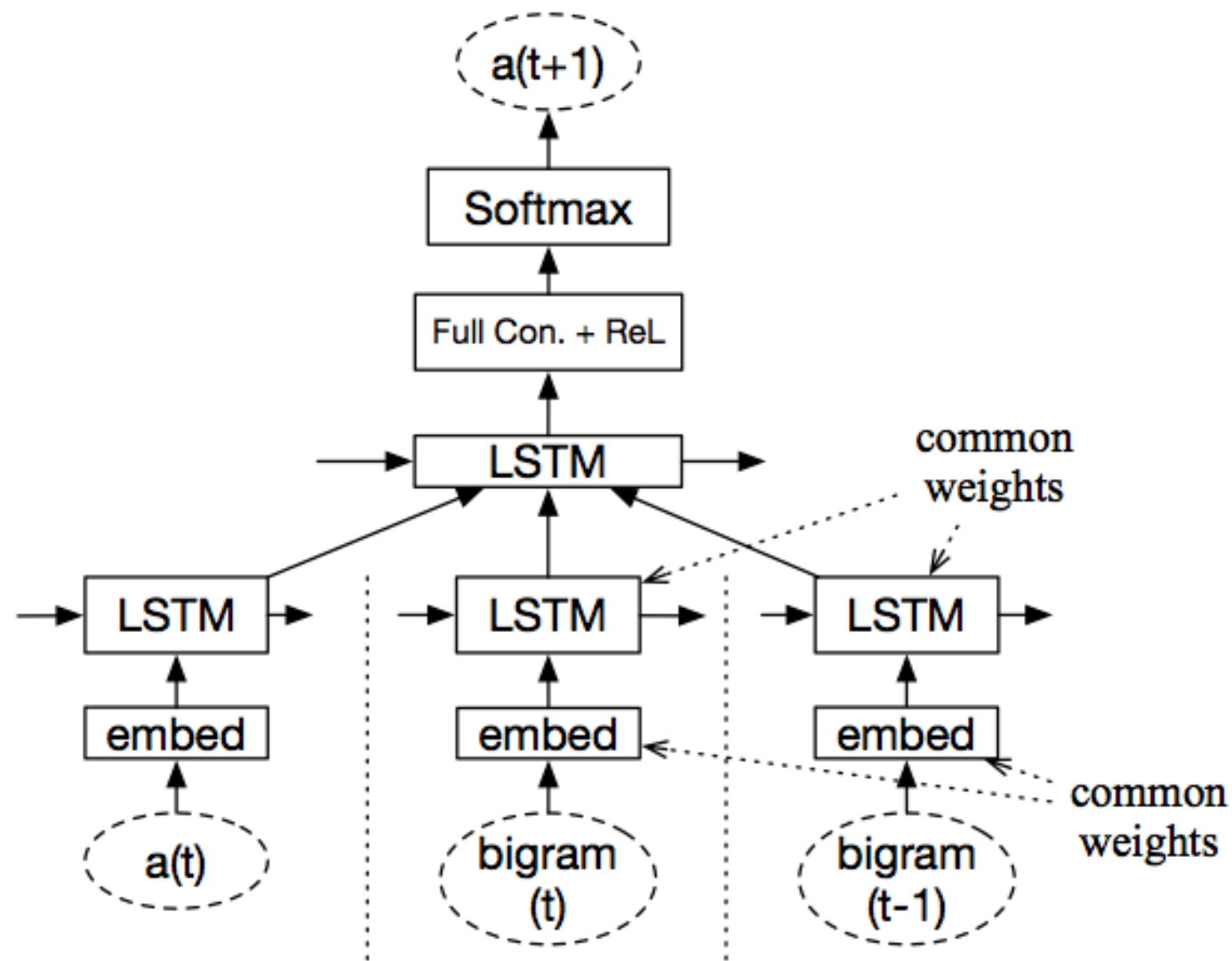
As we discuss later, the SP-2 vector has
the positive impact for several problems.



This suggest that there is room
to improve both
the performance and
our understanding of RNNs.

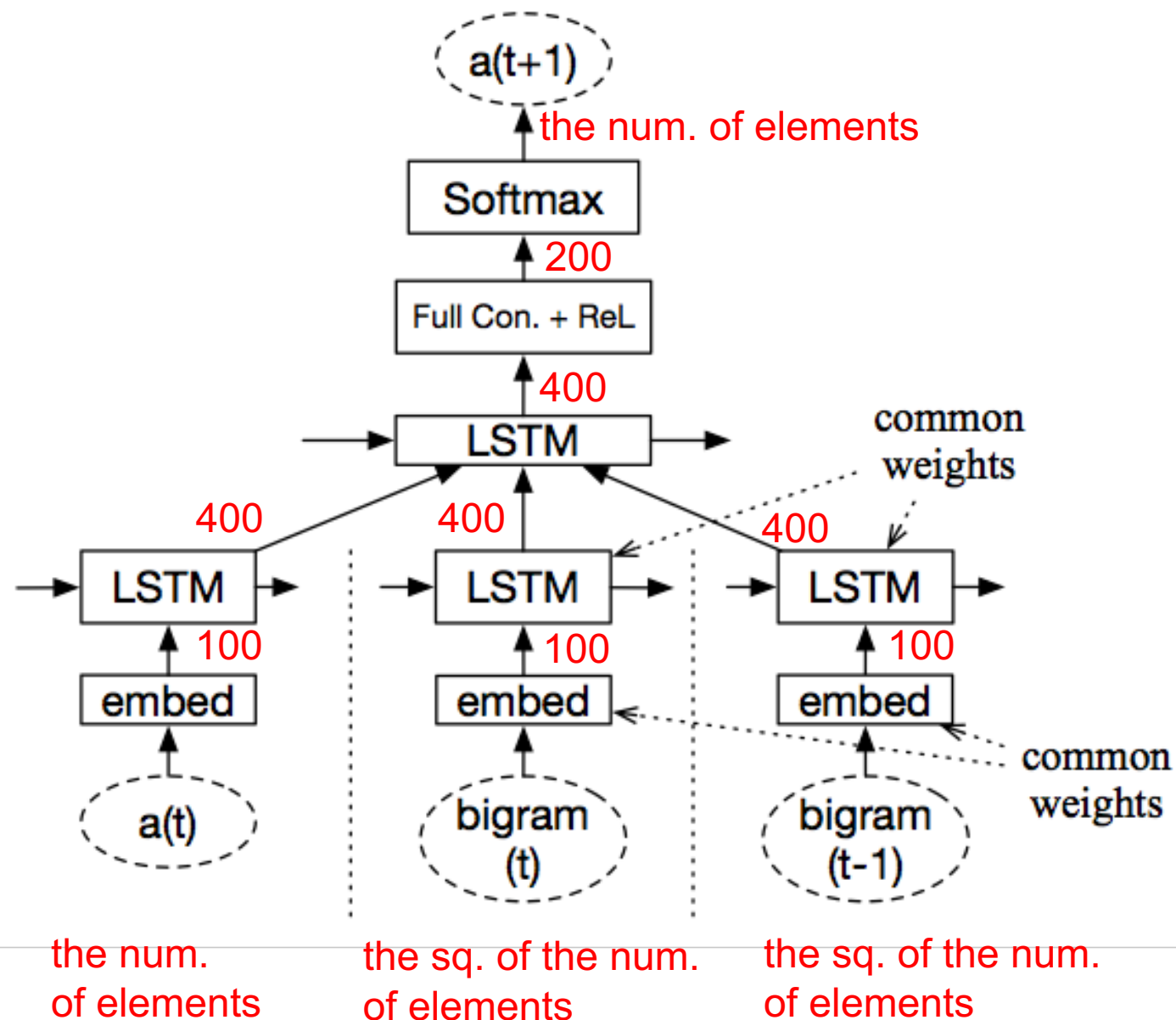


Alternately-taken Bigram Model



- We divide a sentence
■ $(a_0 a_1 a_2 a_3 \dots)$ into:
 - even bigrams
■ $(a_{01} a_{23} \dots)$
 - and the odd bigrams
■ $(a_{12} a_{34} \dots)$.
- These three sequences are fed into the LSTMs separately.
- Note that bigram models are essentially stochastic versions of Strictly 2-Local formal languages.

Alternately-taken Bigram Model: details of the network



Although even and odd bigrams are fed into the LSTMs separately, the LSTMs themselves are required to have common weights.

In this way, the number of samples for learning the weights is doubled.

Implementation

- We used a python library for the deep learning called Chainer.
 - Back propagation is done automatically like as other libraries such as TensorFlow , Theano and Torch etc.
- Learning:
 - the momentum stochastic gradient descent (SGD) with momentum 0.9.
 - Step size: decreased from 0.1 to 0.001.
 - Iterations: 45 epochs.
 - Batch size: 32
 - Dropout is applied to the penultimate layer.
- Execution time
 - for each problem, 10 min. -- several hours
on a commodity machine with a high-end GPU.
- The source code is available at
https://github.com/cshib/rnns_for_spice

Results of Experiments

■ The tables show the top-5 scores for each problem in the SPiCe challenge.

Table 1: comparison of scores with the dim. 400

	simple(400)	sp2(400)	bigram(400)
1	0.906(0.002)	0.915 (0.000)	0.836(0.009)
2	0.920(0.000)	0.919(0.000)	0.878(0.003)
3	0.884(0.001)	0.884(0.001)	0.848(0.001)
4	0.615(0.001)	0.614(0.002)	0.633 (0.002)
6	0.848(0.001)	0.855 (0.001)	0.836(0.001)
7	0.717(0.001)	0.718(0.000)	0.730 (0.001)
8	0.646(0.001)	0.646(0.001)	0.630(0.001)
9	0.959(0.001)	0.960(0.000)	0.958(0.000)
10	0.575(0.001)	0.577(0.001)	0.569(0.001)
11	0.529(0.001)	0.527(0.001)	—
12	0.782(0.002)	0.796 (0.000)	0.727(0.003)
13	0.588(0.001)	0.588(0.001)	0.578(0.001)
14	0.344(0.001)	0.346(0.001)	0.332(0.001)
15	0.264(0.001)	0.265(0.001)	0.261(0.000)

Table 2: comparison of scores with the dim. 600

	simple(600)	sp2(600)	bigram(600)
1	0.909(0.002)	0.915 (0.000)	0.769(0.003)
2	0.920(0.000)	0.920(0.000)	0.838(0.004)
3	0.888(0.001)	0.886(0.001)	0.831(0.001)
4	0.619(0.002)	0.616(0.002)	0.634 (0.001)
6	0.863(0.001)	0.867 (0.001)	0.828(0.002)
7	0.736(0.000)	0.736(0.001)	0.747 (0.001)
8	0.645(0.001)	0.644(0.001)	0.614(0.001)
9	0.962(0.000)	0.962(0.000)	0.959(0.000)
10	0.574(0.001)	0.573(0.001)	0.570(0.002)
11	0.520(0.001)	0.519(0.001)	—
12	0.799(0.002)	0.807 (0.001)	0.713(0.001)
13	0.592(0.001)	0.590(0.001)	0.581(0.000)
14	0.350(0.002)	0.351(0.002)	0.333(0.002)
15	0.263(0.001)	0.263(0.001)	0.258(0.001)

■ Comparison proposed models to basic models:

- SP-2 model : for some problems, is significantly better,
and for other problems, has no significant difference.
- Bigram model : for some problems, is significantly better as well,
but for other problems, is significantly worse.

Conclusion and Prospect

🎬 For the SPiCe competition, we ran three different network structures with two sizes of vectors.

🎬 Overall, the experiments shows that the SP-2 hybrid model was the best.

🎬 We believe the narrow advantage of the SP-2 hybrid model is due to its explicit representation of long-term dependencies in terms of subsequences.

🎬 Different types of formal languages may shed light on the different kinds of long-term dependencies that different types of RNNs can and cannot learn.