Learning Finite-State Assemblies for Efficient Language Modelling

Georgi Shopov Stoyan Mihov

Institute of Information and Communication Technologies
Bulgarian Academy of Sciences

International Conference on Mathematical Logic
September 2023

Section 1

Language Modelling Foundations

Language Models

Definition

A ***language model*** over Σ is a discrete probability distribution over Σ^*; that is, a function $\mathbb{P}\colon \Sigma^* \to [0,1]$ such that $\sum_{\alpha \in \Sigma^*} \mathbb{P}(\alpha) = 1$.

Automatic Speech Recognition

Given acoustic features $A \in \mathbb{R}^n$, $\mathbb{P}_A\colon \Sigma^* \to [0,1]$ is a probability distribution over the transcriptions Σ^*.

Machine Translation

Given a sentence $S \in \Gamma^*$ in the source language, $\mathbb{P}_S\colon \Sigma^* \to [0,1]$ is a probability distribution over the translations Σ^* in the target language.

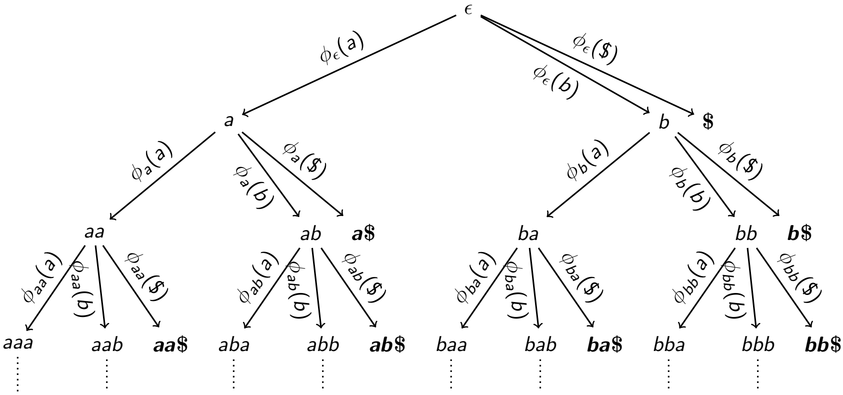
Large Language Models

Given a prompt $P \in \Gamma^*$, $\mathbb{P}_P\colon \Sigma^* \to [0,1]$ is a probability distribution over the responses Σ^*.

Sequential Factorisations

Definition

A ***sequential factorisation*** over Σ is a family $(\phi_\alpha)_{\alpha\in\Sigma^*}$ of discrete probability distributions over $\Sigma_\$:= \Sigma \sqcup \$$, where $\phi_\alpha(a)$ models the probability of the next letter being $a \in \Sigma_\$$ given the context $\alpha \in \Sigma^*$.



Sequential Models

Definition

The ***sequential model*** generated by a sequential factorisation $(\phi_\alpha)_{\alpha \in \Sigma^*}$ is the total function $S\colon \Sigma^* \to [0,1]$ defined as

$$S(\alpha) \coloneqq \phi^*_\epsilon(\alpha\$) \coloneqq \left(\prod_{i=1}^{|\alpha|} \phi_{\alpha_{<i}}(\alpha_i) \right) \phi_\alpha(\$).$$

A sequential model is called

1. ***tight*** if $\sum_{\alpha \in \Sigma^*} S(\alpha) = 1$ (i.e., it is a language model);
2. ***unambiguous*** if it is generated by a unique sequential factorisation.

Questions

1. Do sequential models coincide with language models?
2. Are all sequential models unambiguous?

Question 1: $\mathrm{SM}(\Sigma) \supseteq \mathrm{LM}(\Sigma)$

Proposition

Every language model $\mathbb{P}\colon \Sigma^* \to [0,1]$ *is a sequential model.*

Proof.

For $\alpha \in \Sigma^*$ *such that* $\mathbb{P}(\alpha\Sigma^*) > 0$*, define*

$$\phi_\alpha(a) \coloneqq \begin{cases} \mathbb{P}(\alpha a \Sigma^* \mid \alpha\Sigma^*) & \text{if } a \in \Sigma \\ \mathbb{P}(\alpha \mid \alpha\Sigma^*) & \text{if } a = \$ \end{cases}.$$

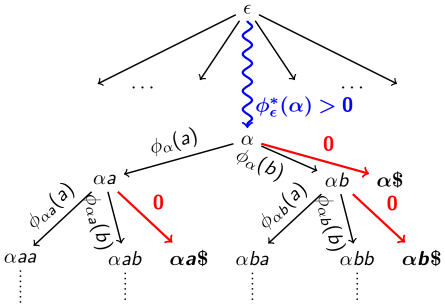
Consequently,

$$\begin{aligned} \mathbb{P}(\alpha) &= \mathbb{P}(\alpha_{\leq 1}\Sigma^*)\mathbb{P}(\alpha_{\leq 2}\Sigma^* \mid \alpha_{<2}\Sigma^*) \cdots \mathbb{P}(\alpha\Sigma^* \mid \alpha_{<|\alpha|}\Sigma^*)\mathbb{P}(\alpha \mid \alpha\Sigma^*) \\ &= \phi_\epsilon(\alpha_1)\phi_{\alpha_{<2}}(\alpha_2)\cdots\phi_{\alpha_{<|\alpha|}}(\alpha_{|\alpha|})\phi_\alpha(\$) \\ &= \phi_\epsilon^*(\alpha\$). \end{aligned}$$

□

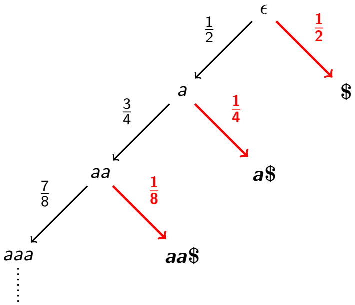
Question 1: $\mathrm{SM}(\Sigma) \not\subseteq \mathrm{LM}(\Sigma)$

Example 1



$$\phi^*_\alpha(\{a,b\}^*\$) = 0,$$
$$\phi^*_\epsilon(\{a,b\}^*\$) \le 1 - \phi^*_\epsilon(\alpha) < 1$$

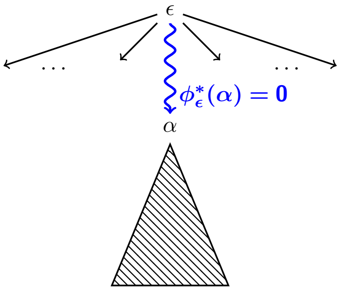
Example 2



$$\phi^*_\epsilon(a^*\$) = \sum_{n=0}^{\infty} \phi^*_\epsilon(a^n)\phi_{a^n}(\$)$$
$$< \sum_{n=0}^{\infty} \frac{1}{2^{n+1}} = 1$$

Question 2: $\mathrm{SM}(\Sigma) \neq \mathrm{USM}(\Sigma)$

Example 1

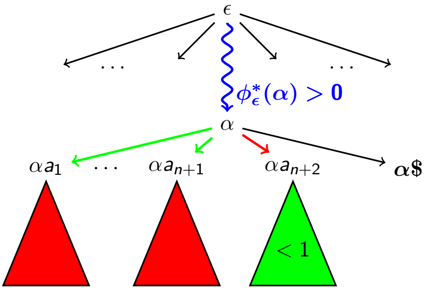


For $|\Sigma| \geq 1$, every $(\psi_\alpha)_{\alpha \in \Sigma^*}$ s.t.

$$\alpha \not\leq \beta \implies \phi_\beta = \psi_\beta$$

generates the same sequential model.

Example 2



For $|\Sigma| \geq 2$, weights can be redistributed

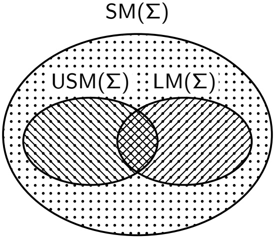
$$\phi_\alpha(a_{n+2}) \longrightarrow \phi^*_{\alpha a_{n+2}}(\Sigma^*\$),$$
$$\phi^*_{\alpha a_i}(\Sigma^*\$) \longrightarrow \phi_\alpha(a_i).$$

Tightness and Ambiguity of Sequential Models

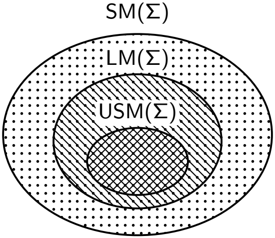
Proposition

A sequential model generated by a sequential factorisation Φ is unambiguous if and only if

1. *($|\Sigma| = 1$) Φ is accessible;*
2. *($|\Sigma| > 1$) Φ is tight and accessible.*



(a) $|\Sigma| = 1$



(b) $|\Sigma| > 1$

Section 2

Finite-State Language Modelling

Finite-State Language Models

Definition

$(\phi_\alpha)_{\alpha\in\Sigma^*}$ is ***finite-state*** if it is ***compatible*** with a right invariant equivalence relation $\equiv\subseteq\Sigma^*\times\Sigma^*$ with a finite index; that is,

$$\alpha\equiv\beta \implies \phi_\alpha=\phi_\beta.$$

Then $(\phi_\alpha)_{\alpha\in\Sigma^*}$ is determined by the finite family $(\phi_\alpha)_{[\alpha]_\equiv\in\Sigma^*/\equiv}$.

Proposition

A sequential factorisation $(\phi_\alpha)_{\alpha\in\Sigma^*}$ is finite-state if and only if it can be represented by a stochastic sequential finite-state transducer.

Proof.

$(\Longrightarrow)$ *Consider* $(\Sigma^*\times[0,1],\Sigma^*/\equiv,([\epsilon]_\equiv,1),\mathbb{F},\delta,\lambda)$, *where*

$$\begin{aligned}
\mathbb{F} &:= \{([\alpha]_\equiv,\phi_\alpha(\$)) &&\mid \alpha\in\Sigma^*\},\\
\delta &:= \{(([\alpha]_\equiv,a),[\alpha a]_\equiv) &&\mid \alpha\in\Sigma^*\wedge a\in\Sigma\},\\
\lambda &:= \{(([\alpha]_\equiv,a),\phi_\alpha(a)) &&\mid \alpha\in\Sigma^*\wedge a\in\Sigma\}.
\end{aligned}$$

□

N-gram Language Models

Definition

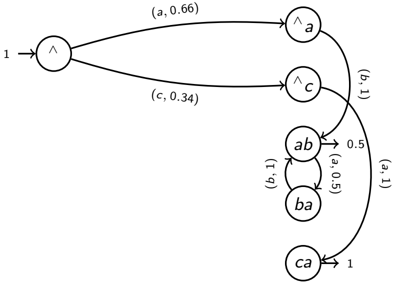
$(\phi_\alpha)_{\alpha \in \Sigma^*}$ is ***n*-gram** if it is compatible with $\equiv_n \subseteq \Sigma^* \times \Sigma^*$, where

$$\alpha \equiv_n \beta \iff \alpha_{>|\alpha|-n+1} = \beta_{>|\beta|-n+1}.$$

Proposition

$\equiv_n$ is a right invariant equivalence relation with a finite index. Thus, every n-gram sequential factorisation is finite-state.

Stochastic sequential finite-state transducer



Smoothed n-gram Language Models

Definition (Simplified)

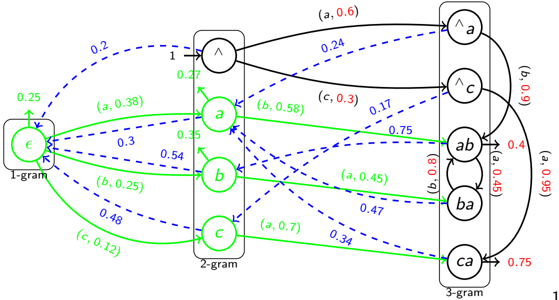
$(\psi^{(n)}_\alpha)_{\alpha\in\Sigma^*}$ is a ***smoothing*** of the n-gram $(\phi^{(n)}_\alpha)_{\alpha\in\Sigma^*}$ if

$$\psi^{(n)}_\alpha(a) := \begin{cases} \lambda_{\alpha a}\phi^{(n)}_\alpha(a) & \text{if } \phi^{(n)}_\alpha(a) > 0 \\ \mu_\alpha \psi^{(n-1)}_{\alpha_{>1}}(a) & \text{otherwise} \end{cases}.$$

Proposition

Every smoothed n-gram sequential factorisation is trim.

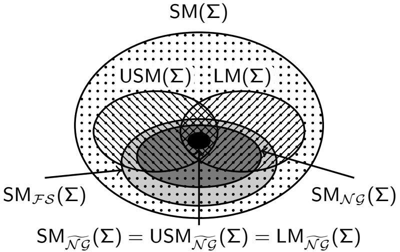
Stochastic sequential finite-state f-transducer



Tightness and Ambiguity of Finite-State Sequential Models

Proposition

A sequential model generated by a finite-state sequential factorisation Φ is tight if and only if every accessible factor of Φ is co-accessible.



(a) $|\Sigma| = 1$ (b) $|\Sigma| > 1$

Section 3

Neural Language Modelling

Recurrent Neural Networks (RNNs)

Definition (Automata-based view)

An ***RNN*** is a letter-to-letter pure sequential transducer

$$\mathcal{R} \coloneqq ((\mathbb{R}^{d_i})^* \times (\mathbb{R}^{d_o})^*, H \subseteq \mathbb{R}^{d_h}, (h_0, x_0), \delta, \lambda).$$

The ***behaviour*** of $\mathcal{R}$ is the function $[\![\mathcal{R}]\!] \colon (\mathbb{R}^{d_i})^* \to (\mathbb{R}^{d_o})^*$, where $[\![\mathcal{R}]\!](\alpha) \coloneqq x_0 \odot \lambda^*(h_0, \alpha)$. $\mathcal{R}$ is ***bounded*** if $\{\|x\|_p \mid x \in \mathrm{Im}(\lambda)\}$ is bounded.

Example

An ***Elman RNN*** is a bounded RNN such that

$$\begin{aligned}\delta(h, x) &\coloneqq f(Uh + Vx + b_\delta),\\ \lambda(h, x) &\coloneqq g(W\delta(h, x) + b_\lambda),\end{aligned}$$

where f and g are sigmoid functions, $U \in \mathbb{R}^{d_h \times d_h}$, $V \in \mathbb{R}^{d_h \times d_i}$, $b_\delta \in \mathbb{R}^{d_h}$, $W \in \mathbb{R}^{d_o \times d_h}$ and $b_\lambda \in \mathbb{R}^{d_o}$.

Sequential Neural Networks (SNNs)

Definition

An ***SNN*** is a tuple $(\Sigma, e, \mathcal{R}, f, g)$, where

- $e\colon \Sigma \to \mathbb{R}^{d_i}$ (***embedding***);
- $\mathcal{R}$ is an RNN (***encoder***);
- $f\colon \mathbb{R}^{d_o} \to \mathbb{R}^{\Sigma_\$}$ (***decoder***);
- $g\colon \mathbb{R}^{\Sigma_\$} \to \mathbf{\Delta}^{\Sigma_\$}$ (***projection***).

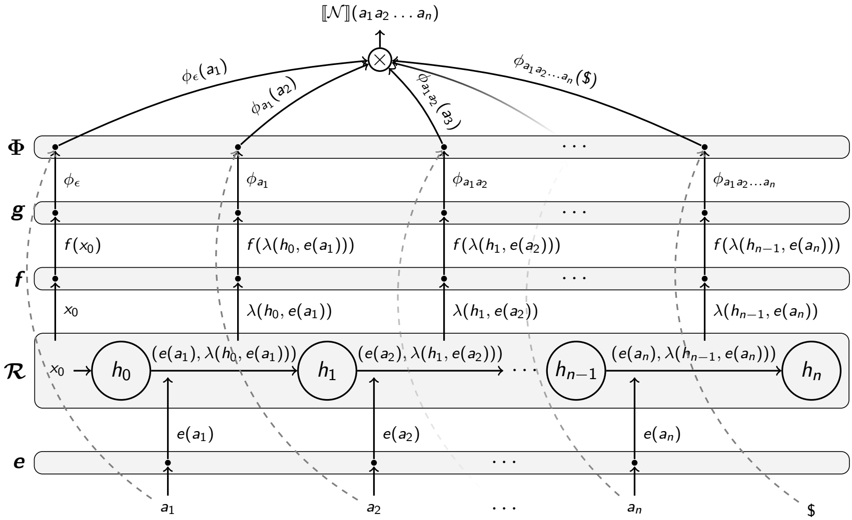
Definition

The ***sequential factorisation associated with*** $\mathcal{N} \coloneqq (\Sigma, e, \mathcal{R}, f, g)$ is $\Phi \coloneqq (\phi_\alpha)_{\alpha \in \Sigma^*}$ such that for $a_1 a_2 \ldots a_n \in \Sigma^*$,

$$(\phi_\epsilon, \phi_{a_1}, \phi_{a_1 a_2} \ldots, \phi_{a_1 a_2 \ldots a_n}) \coloneqq (e^* \circ [\![\mathcal{R}]\!] \circ f^* \circ g^*)(a_1 a_2 \ldots a_n),$$

where $e^*\colon \Sigma^* \to (\mathbb{R}^{d_i})^*$, $f^*\colon (\mathbb{R}^{d_o})^* \to (\mathbb{R}^{\Sigma_\$})^*$, $g^*\colon (\mathbb{R}^{\Sigma_\$})^* \to (\mathbf{\Delta}^{\Sigma_\$})^*$ are the homomorphic extensions of e, f and g. The ***behaviour*** of $\mathcal{N}$, denoted $[\![\mathcal{N}]\!]$, is the sequential model generated by Φ.

Behaviour of an SNN



$[\![\mathcal{N}]\!]$ can be represented by a stochastic sequential transducer.

Common Decoders and Projections

- Typically, $f(x) := Wx + b$, where $W \in \mathbb{R}^{\Sigma_\$ \times d_o}$ and $b \in \mathbb{R}^{\Sigma_\$}$.
  - W_a is the ***encoding of*** $\boldsymbol{a \in \Sigma_\$}$.
  - $x := (e^* \circ [\![\mathcal{R}]\!])(\alpha)_{|\alpha|+1}$ is the ***encoding of*** $\boldsymbol{\alpha \in \Sigma^*}$.
  - $f(x) \in \mathbb{R}^{\Sigma_\$}$ represents the ***similarities*** between α and $\Sigma_\$$.
- The role of g is to project the similarity scores onto the probability simplex such that $\phi_\alpha(a)$ is proportional to the similarity between α and a.

Definition

softmax: $\mathbb{R}^n \to \boldsymbol{\Delta}^n$ is defined for $x \in \mathbb{R}^n$ and $1 \leq i \leq n$ as

$$\text{softmax}(x)_i := \frac{\exp(x_i)}{\sum_{j=1}^{|x|} \exp(x_j)}.$$

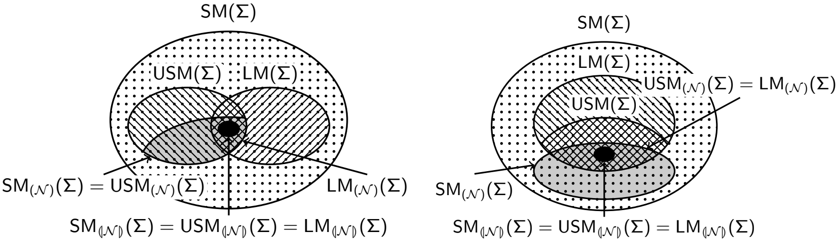
Tightness and Ambiguity of Neural Sequential Models

Proposition

Every softmax-*based neural sequential factorisation is trim.*

Proposition

Every bounded softmax-*based neural sequential model is tight.*



(a) $|\Sigma| = 1$ (b) $|\Sigma| > 1$

Section 4

Learning Finite-State Assemblies

Comparison of Finite-State and Neural Language Models

Finite-state language models

+ computationally optimal (all computations are stored in the transition table);
+ easier to interpret (using efficient constructions and closure properties);
− discrete nature, which significantly hinders learning;
− less performant in practice.

Neural language models

− computationally expensive (the transition table is computed on demand);
− opaque and hard to interpret (consist of millions of floating point parameters);
+ continuous nature, which facilitates learning;
+ very performant in practice.

Quantised SNNs

Definition

A ***quantisation function*** on X is any $q\colon X \to Q$, where Q is finite.

Definition

The ***q-quantisation*** of an RNN $((\mathbb{R}^{d_i})^* \times (\mathbb{R}^{d_o})^*, H, (h_0, x_0), \delta, \lambda)$ is $((\mathbb{R}^{d_i})^* \times (\mathbb{R}^{d_o})^*, \widehat{H}, (\widehat{h}_0, x_0), \widehat{\delta}, \lambda)$, where

- $q\colon H \to \widehat{H}$ is a quantisation function;
- $\widehat{h}_0 := q(h_0)$;
- $\widehat{\delta} := \delta \circ q$.

Proposition

Quantised SNNs are stochastic sequential finite-state transducers and vice versa.

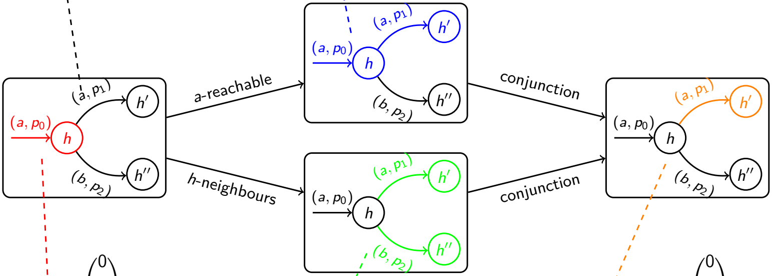
Proof.

($\Longrightarrow$) Straightforward.
($\Longleftarrow$) Minsky's construction. $\square$

Minsky's Construction

$$\mathbb{B}^{\Sigma} \ni e(a) := \begin{array}{cl} \begin{pmatrix} 0 \\ \vdots \\ 0 \\ 1 \\ 0 \\ \vdots \\ 0 \end{pmatrix} & \begin{array}{l} \\ \\ \\ \leftarrow a \\ \\ \\ \\ \end{array} \end{array}$$

$$\mathbb{B}^{Q \times \Sigma} \ni Ve(a) := \begin{array}{cl} \begin{pmatrix} \vdots \\ 1 \\ \vdots \\ 1 \\ \vdots \end{pmatrix} & \begin{array}{l} \\ \leftarrow (h, a) \\ \\ \leftarrow (h', a) \\ \\ \end{array} \end{array}$$



$$\mathbb{B}^{Q \times \Sigma} \ni h := \begin{array}{cl} \begin{pmatrix} 0 \\ \vdots \\ 0 \\ 1 \\ 0 \\ \vdots \\ 0 \end{pmatrix} & \begin{array}{l} \\ \\ \\ \leftarrow (h, a) \\ \\ \\ \\ \end{array} \end{array}$$

$$\mathbb{B}^{Q \times \Sigma} \ni Uh := \begin{array}{cl} \begin{pmatrix} \vdots \\ 1 \\ \vdots \\ 1 \\ \vdots \end{pmatrix} & \begin{array}{l} \\ \leftarrow (h', a) \\ \\ \leftarrow (h'', b) \\ \\ \end{array} \end{array}$$

$$\sigma(Uh + Ve(a) - \mathbf{1}) = \begin{array}{cl} \begin{pmatrix} 0 \\ \vdots \\ 0 \\ 1 \\ 0 \\ \vdots \\ 0 \end{pmatrix} & \begin{array}{l} \\ \\ \\ \leftarrow (h', a) \\ \\ \\ \\ \end{array} \end{array}$$

Product Quantisation

Quantisation in high dimensional vector spaces such as $\mathbb{R}^n$ is difficult.

Definition

A ***product quantisation function*** on $\mathbb{R}^n$ is a quantisation function $\overline{q}\colon \mathbb{R}^n \to Q_1 \times Q_2 \times \cdots \times Q_m$ defined as

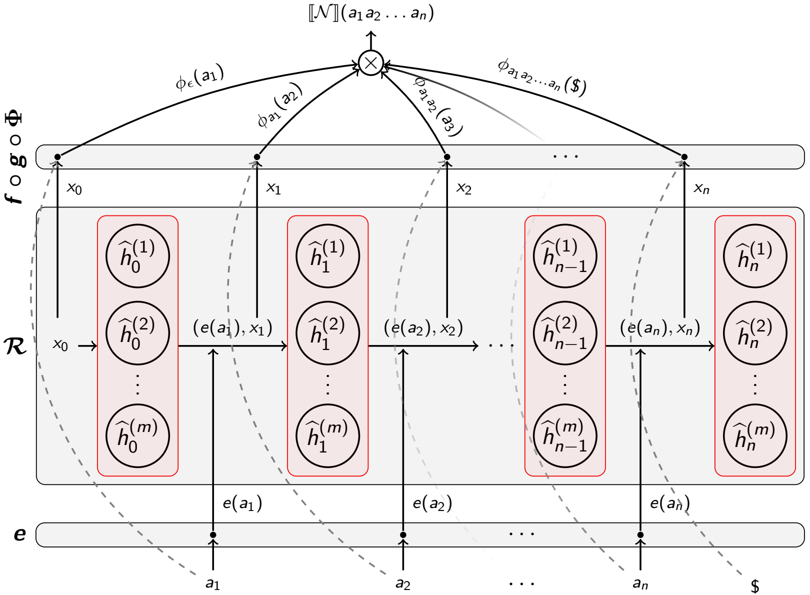
$$\overline{q}(x) \coloneqq (q_1(x^{(1)}), q_2(x^{(2)}), \ldots, q_m(x^{(m)})),$$

where $x = x^{(1)} \odot x^{(2)} \odot \cdots \odot x^{(m)}$ and $q_i\colon \mathbb{R}^{n_i} \to Q_i$ is a quantisation function for $1 \leq i \leq m$.

Remark

If $|Q_1| = |Q_2| = \cdots = |Q_m| = k$, then $\overline{q}$ produces k^m states. In practice $n \sim 1024$; thus, if $m = n$, a $\overline{q}$-quantisation of an RNN would have in the order of k^{1024} states.

Monolithic QSNNs



Cartesian Decomposition of RNNs

Consider, for $1 \leq j \leq m$, the RNNs

$$\mathcal{R}^{(j)} \coloneqq ((\mathbb{R}^{d_i})^* \times (\mathbb{R}^{d_o^{(j)}})^*, H^{(j)} \subseteq \mathbb{R}^{d_h^{(j)}}, (h_0^{(j)}, x_0^{(j)}), \delta^{(j)}, \lambda^{(j)}).$$

If $\mathcal{R} \coloneqq ((\mathbb{R}^{d_i})^* \times (\mathbb{R}^{d_o})^*, H \subseteq \mathbb{R}^{d_h}, (h_0, x_0), \delta, \lambda)$ is an RNN such that

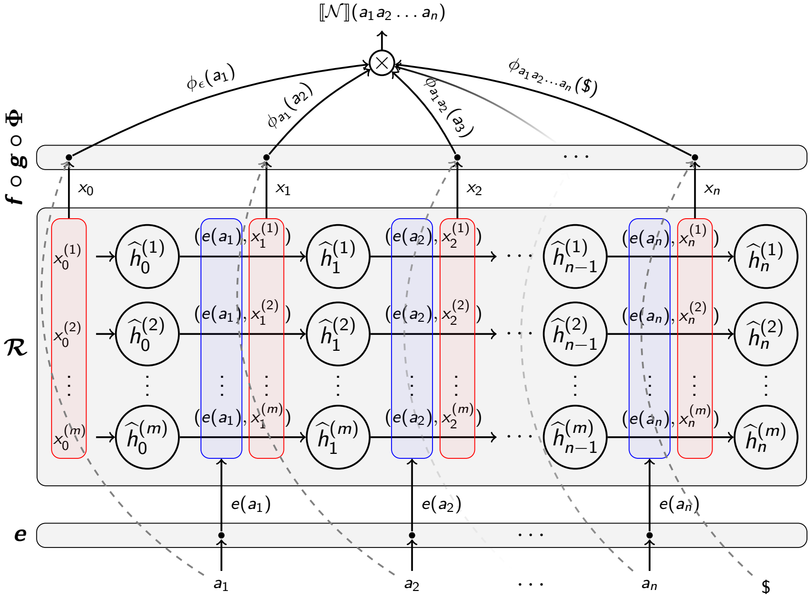
- $d_h = \sum_{j=1}^{m} d_h^{(j)}$, $d_o = \sum_{j=1}^{m} d_o^{(j)}$,
- $h_0 = \bigodot_{j=1}^{m} h_0^{(j)}$, $x_0 = \bigodot_{j=1}^{m} x_0^{(j)}$,
- $\delta(h, x) = \bigodot_{j=1}^{m} \delta^{(j)}(h^{(j)}, x)$, $\lambda(h, x) = \bigodot_{j=1}^{m} \lambda^{(j)}(h^{(j)}, x)$, where $h = \bigodot_{j=1}^{m} h^{(j)}$,

then $\mathcal{R}$ is the Cartesian product $\mathcal{R}^{(1)} \times \mathcal{R}^{(2)} \times \ldots \times \mathcal{R}^{(m)}$.

	Transitions
Monolithic $\mathcal{R}$	$k^m\|\Sigma\|$
Decomposed $\mathcal{R}$	$mk\|\Sigma\|$

Table: Each $\mathcal{R}^{(j)}$ is quantised into k values.

Decomposed QSNNs



Sparse Multilayer RNNs

Consider the composition $\mathcal{R} \coloneqq \mathcal{R}_1 \circ \mathcal{R}_2 \circ \cdots \circ \mathcal{R}_\ell$, where

$$\mathcal{R}_i \coloneqq \bigtimes_{j=1}^{m} \mathcal{R}_i^{(j)}, \quad \text{for } 1 \leq i \leq \ell,$$

and let $s \in [0, m]$ be such that for $1 < i \leq \ell$ and $h = \bigodot_{j=1}^{m} h^{(j)}$,

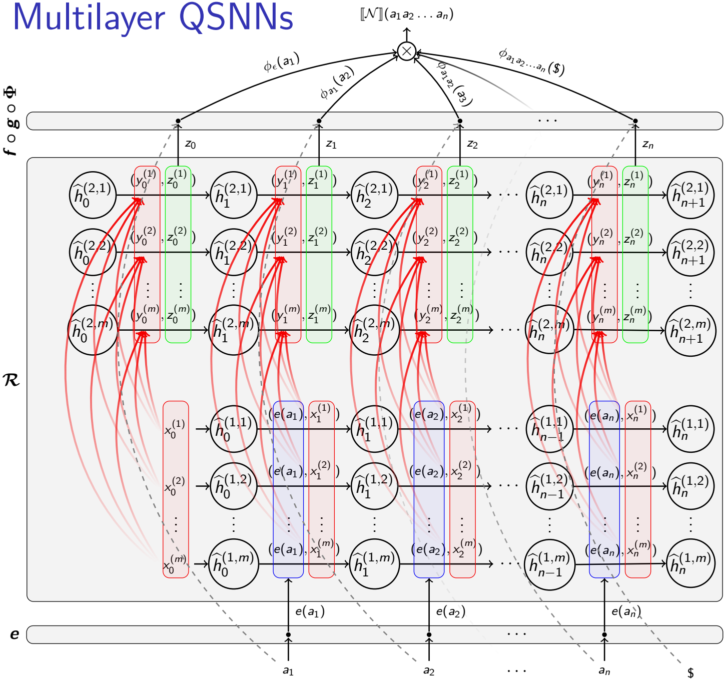
$$\delta_i(h, x) = \bigodot_{j=1}^{m} \delta_i^{(j)}(h^{(j)}, M_i^{(j)} x),$$

where $M_i^{(j)} \in \mathbb{R}^m$ and $\|M_i^{(j)}\|_0 = s$.

	Transitions
Monolithic $\mathcal{R}$	$k^{m\ell}\lvert\Sigma\rvert$
Sparse Multilayer $\mathcal{R}$	$mk\lvert\Sigma\rvert + (\ell - 1)mk^{1+s}$

Table: Every $\mathcal{R}_i^{(j)}$ is quantised into k values.

Sparse Multilayer QSNNs



Finite-State Assemblies

Definition

A ***finite-state assembly*** over Σ is a tuple $\mathcal{A} := (T, I, F, E)$, where

- T is a finite set of sequential finite-state transducers over Σ;
- $I \subseteq T$ is a set of ***initial transducers***;
- $F \subseteq T$ is an ordered set of ***final transducers***;
- (T, E) is an ordered DAG of ***compositional interdependencies***.

The behaviour of $\mathcal{A}$ is $[\![\mathcal{A}]\!] \colon \Sigma^* \to \Sigma^*$ defined as

$$[\![\mathcal{A}]\!](\alpha) := \bigodot_{\mathcal{F} \in F} (\!|\mathcal{F}|\!)(\alpha),$$

where $(\!|\mathcal{T}|\!)$ is defined for $\mathcal{T} \in T$ inductively

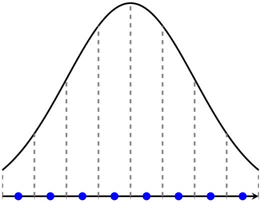
- $(\!|\mathcal{T}|\!)(\alpha) := [\![\mathcal{T}]\!](\alpha)$ if $\mathcal{T} \in I$;
- $(\!|\mathcal{T}|\!)(\alpha) := [\![\mathcal{T}]\!] \left(\bigodot_{\mathcal{T}' \in \mathcal{P}(\mathcal{T})} (\!|\mathcal{T}'|\!)(\alpha) \right)$ if $\mathcal{T} \in T \setminus I$.

Quantised sparse multilayer RNNs are a type of finite-state assemblies.

Differentiable Quantisation

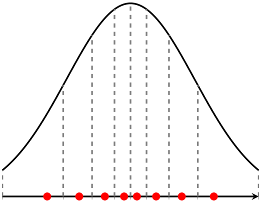
- Uniform quantisation of $[a, b]$

$$q(x) := \left\lfloor x \frac{2^b}{b-a} \right\rceil \frac{b-a}{2^b}, \quad \frac{\partial q}{\partial x} := 1.$$



- Non-uniform learnable quantisation

$$q(x, \mathcal{C}) := \arg\min_{c \in \mathcal{C}} \|x - c\|_2, \quad \frac{\partial q}{\partial x} := \frac{\partial q}{\partial c} := 1.$$

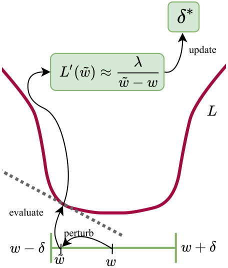


- Mixed-precision quantisation of $[-a, a]$

$$q(x) := x + \delta\epsilon, \quad \epsilon \sim U\left[-\frac{a}{2}, \frac{a}{2}\right], \delta \in [0, 1].$$

A precision term is added to the loss function

$$L(w + \delta\epsilon) + \lambda \log \frac{1}{\delta}.$$



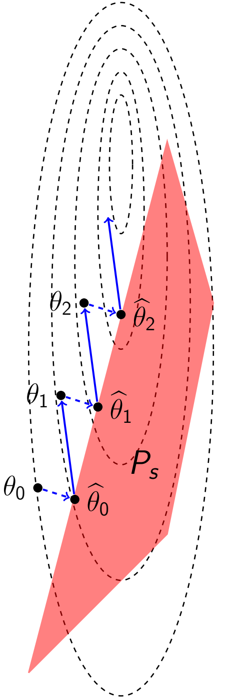
Differentiable Sparsification

- ***Projected gradient descent:*** After each gradient descent step we project $M_i^{(j)}$ to the closest vector in the subspace $P_s \subseteq \mathbb{R}^m$ of vectors with s non-zero coordinates

$$\widehat{M}_i^{(j)} := \arg\min_{x \in P_s} \left\| M_i^{(j)} - x \right\|_2,$$
$$P_s := \{x \in \mathbb{R}^m \mid \|x\|_0 = s\}.$$

- ***L_0 regularisation:*** The masks are probabilistic $M_i^{(j)} \sim Bern(p_i^{(j)})$. To make sampling differentiable $Bern(p_i^{(j)})$ is approximated with $Concrete(p_i^{(j)}, \lambda)$. The expected sparsity is optimised via

$$\sum_{i=1}^{\ell} \sum_{j=1}^{m} \mathbb{E}\left[\| M_i^{(j)} \|_0 \right] = \sum_{i=1}^{\ell} \sum_{j=1}^{m} p_i^{(j)}.$$



Experimental Results

- The hidden dimension is 1024.
- Each machine uses a separate non-uniform quantisation function with 16 learnable codes.
- The sparsification method is projected gradient descent.

Architecture	Layers	Machines per Layer	Density	Perplexity		Complexity
Gated Recurrent Unit	1	1	100%	124.6	7.08	MFLOPS
Finite-State Assembly	4	1024	5%	123.8	4096	lookups
Finite-State Assembly	10	1024	2.5%	122.9	10240	lookups

Conclusion

- Finite-state language models can be learned with gradient-based methods by relaxing the discreteness of the optimisation constraints.
- Assemblies of finite-state sequential transducers demonstrate promising results for language modelling by combining
  - the computational efficiency and interpretability of sequential finite-state transducers,
  - the performance and learnability of neural networks.
- Further work is required to
  - reduce the size of the transitions tables,
  - develop more effective quantisation and sparsification methods,
  - make learning of more general assembly architectures feasible.