A Tensor Framework for Learning in Structured Domains

Daniele Castellana
Postdoc at Università di Pisa
Department of Computer Science

Math4DS-AI-ML Seminar
Trento, 20 April 2022
Outline

- What are structured data
- Recursive processing of structured data
- Recursive tensor models
- Tensor decompositions & Model approximations
  - Some results on synthetic task
- Tensor models for unbounded structures
  - Some results on NLP tasks
- Conclusion & Future directions
Structured data are everywhere

A structured data comprises a set of **atomic entities** connected by a structure.

(a) Sentence

(b) HTML

(c) Chemical compound [11]

(d) Image [1]
Let’s consider a **sentiment prediction** task.

**Example:**

- **Very Negative**
  - An awkward and indigestible movie

- **Positive**
  - Admirable but not much fun to watch
Machine Learning (ML) models for structured data should:

- process different structures:
  - Effective but too-tepid
  - A very funny movie
  - Not a bad journey at all

- process the atomic information + contextual information:
  - The sky is blue and the grass is green ≠ The sky is green and the grass is blue

- capture complex interactions:
  - negative and negative = very negative
  - positive but negative = ?

Daniele Castellana | A Tensor Framework for Learning in Structured Domains
A **Directed Oriented Acyclic Graph (DOAG)** is a directed acyclic graph where it exists a **total order** on the edges leaving from each vertex.

- a label is attached to each node;
- there is a **root**;
- a node can have at most \( P \) **parents**;
- a node can have at most \( L \) **children**;
- a node has a **position** w.r.t. its siblings;

\( L \) is the structure maximum **out-degree**.
We tackle **supervised learning** tasks on structured data as recursive transductions [8] between an input structure $X$ and an output structure $Y$.

The **state-transition** function $f_{\theta_f}$ computes the **hidden states** recursively:

$$h_v = f_{\theta_f}(x_v, h_{v1}, \ldots, h_{vL});$$

The **output** function $g_{\theta_g}$ computes the **output labels** as:

$$y_v = g_{\theta_g}(h_v);$$

The parameters $\theta_f$ and $\theta_g$ are **learned** from data.
In probabilistic recursive model:
- the labels $x_v$ and $y_v$ are realisations of random variables;
- the hidden states $h_v \in [1, C]$ are realisations of discrete random variables;
- $g_{\theta_g}$ is a probability distribution:
  $$P(y_v \mid h_v, \theta_g);$$
- $f_{\theta_f}$ is a categorical probability distribution:
  $$P(h_v \mid x_v, h_{v1}, \ldots, h_{vL}, \mathbf{P});$$

The state-transition distribution is parametrised by a non-negative tensor $\mathbf{P}$ of size $M \times C^L \times C$, assuming that $x_v$ has $M$ states.

We denote this model as Hidden Recursive Tensor Model (HRTM).
We can represent HRTM as a **graphical model**.

\[ P(y_v \mid h_v) \text{ and } P(h_v \mid x_v, h_{v1}, \ldots, h_{vL}) \text{ are shared among all nodes.} \]
The expressiveness of HRTM is related to the **conditional independences** introduced by the model.

- **Causality**: the hidden state of a node does not depend on its siblings, e.g.:
  \[ H_{v1} \perp \perp H_{vI} | H_{v1} \]

- **Recursiveness**: the hidden state of a node depends only on its children (if their state is given), e.g.:
  \[ H_{v} \perp \perp H_{v1} | H_{v1} \]

Both conditional independences are **intrinsic** in the recursive processing of the structure.

**HRTM has high expressiveness**
Recursive Neural Models

If we move to the neural world:

- \( x_v \in \mathbb{R}^M \) is a real vector;
- \( h_v \in \mathbb{R}^C \) is a real vector;
- \( g_{\theta_g} \) and \( f_{\theta_f} \) are neural networks;

\( f_{\theta_f} \) and \( g_{\theta_g} \) are unrolled over the input structure.
RecNN in the Literature

Most of RecNNs employ first-order state-transition function:

\[ h_v = \sigma \left( W x_v + \sum_{j=1}^{L} U_j h_{vj} + b \right), \]

where \( \theta_f = \{ W, U_1, \ldots, U_L, b \} \) are the learnable parameters, and \( \sigma \) is a non-linear activation function.

It can be shown that they are universal, i.e. they can represent any function on structure [9];

The number of parameters does not grow exponentially w.r.t. \( L \).

We fill this gap by introducing Recursive Neural Tensor Network (RecNTN).

RecNTN Definition

RecNTN defines $f_{\theta_t}$ as:

$$h_v = \sigma(T(\bar{x}_v, \bar{h}_{v1}, \ldots, \bar{h}_{vL})).$$

The above equation is the generalisation of a NN layer to the multivariate case:

- $\sigma$ is the non-linear activation function;
- $T$ is an augmented tensor $(M + 1) \times (C + 1)^L \times C$ which define a multi-affine map. ($\bar{a}$ are the homogeneous coordinates of $a$)

The number of parameters is $O(C^L)$.

---

Augmented Tensors
And Higher-Order Interactions

Let’s consider a node with only two children. The augmented tensors define the following multi-affine map:

$$T(\vec{x}_v, \vec{h}_{v1}, \vec{h}_{v2}) = \sum_{i=1}^{M+1} \sum_{j_1=1}^{C+1} \sum_{j_2=1}^{C+1} T[i, j_1, j_2, :] \cdot \vec{x}_v[i] \cdot \vec{h}_{v1}[j_1] \cdot \vec{h}_{v2}[j_2].$$

$$T(\vec{x}_v, \vec{h}_{v1}, \vec{h}_{v2}) =$$

$$= b +$$

$$+ Wx_v + U_1 h_{v1} + U_2 h_{v2} +$$

$$+ A(x_v, h_{v1}) + B(x_v, h_{v2}) + C(h_{v1}, h_{v2}) +$$

$$+ D(x_v, h_{v1}, h_{v2}),$$

Augmented tensors model interactions of any order.

If all entries of $A, B, C, D$ are zero, we obtain the first-order neural model.
The augmented tensor allows:

1. modelling higher-order interactions;
2. representing any multi-affine map.

The point 2) ensure us that RecNTN can simulate any neural recursive model which use a multi-affine map in the state-transition function (using the same hidden state size).

RecNTN has high expressiveness.

For example, First-Order Neural Models can be obtained by imposing most of the entries of $T$ to zero.

Note that also more complex aggregation functions exist (e.g. multivariate polynomials) and they cannot be represented by augmented tensors. Nevertheless, these functions have not been used on structured data.
Our **tensor-based** framework for recursive models is based on the following observations:

- more expressive models are parametrised by a tensor $T$ of order $O(L)$ (for the state-transition function);
- other recursive models can be defined by imposing specific constraints on $T$.
  - e.g. imposing zero constraint on $T$ we obtain First-Order NN.

We can interpret the **constraints** as a model **inductive bias**.

**Inductive bias** $\implies$ any **prior assumption** that the model can exploit to prefer a hypothesis over another [16].

This observation is valid for both **probabilistic** and **neural** models.

---

**D. Castellana and D. Bacciu.** “A tensor framework for learning in structured domains”. In: *Neurocomputing* 470 (2022), pp. 405–426. DOI: https://doi.org/10.1016/j.neucom.2021.05.110
While full tensor models have **high expressiveness**, they are **not feasible** in practice due to the exponential number of parameters $O(C^L)$.

Nevertheless, the tensor parameterisation paves the way to a principle method application to reduce the number of parameter.

We can leverage **tensor decompositions** (TD).

We focus on three tensor decompositions:
- **Candecomp/Parafrac (CP)** [12];
- **High-Order Singular Value Decomposition (HOSVD)** [7];
- **Tensor-Train (TT)** [19].
How to use TD to approximate models

We do not apply TDs to compress the tensor parameter $T$!

On the contrary, we build new recursive models where:

- Decomposition factors $\Rightarrow$ Model parameters
- Decomposition rank $\Rightarrow$ Model hyper-parameters

In these new models:

- The hidden state size $C$ reflects the structure complexity,
- the decomposition rank $R$ reflects the aggregations complexity.

Low ranks $\Rightarrow$ More compression
High ranks $\Rightarrow$ Better approximation
The CP decomposition of a tensor $T$ is defined as:

$$T[j_1, \ldots, j_L, k] \approx \sum_{r=1}^{R} U_1[j_1, r] \cdots U_L[j_L, r] Q[k, r],$$

Let $P$ be the tensor which parametrises the HRTM state-transition distribution. Applying the CP decomposition, we obtain:

$$P(h_v | h_v1, \ldots, h_vL, P) \approx \sum_{r_v} P(h_v | r_v, Q) \prod_{l=1}^{L} P(r_v | h_vl, U_l),$$

$r_v \in [1, R]$ are the states of a new discrete random variable $R_v$. 

Daniele Castellana | A Tensor Framework for Learning in Structured Domains
Let $T$ be the augmented tensor which parametrises the multi-affine map of RecNTN state-transition function. Applying the CP decomposition, we obtain:

$$T(\bar{h}_v^1, \ldots, \bar{h}_v^L) = \sum_{j_1=1}^{C+1} \cdots \sum_{j_L=1}^{C+1} T[j_1, \ldots, j_L, :] \times \bar{h}_v^1[j_1] \cdots \bar{h}_v^L[j_L]$$

$$\approx \sum_{j_1=1}^{C+1} \cdots \sum_{j_L=1}^{C+1} \sum_{r=1}^{R} U_1[j_1, r] \cdots U_L[j_L, r] Q[:, r] \bar{h}_v^1[j_1] \cdots \bar{h}_v^L[j_L]$$

$$= \sum_{r=1}^{R} \left( \left( \sum_{j_1=1}^{C+1} U_1[j_1, r] \bar{h}_v^1[j_1] \right) \cdots \left( \sum_{j_L=1}^{C+1} U_L[j_L, r] \bar{h}_v^L[j_L] \right) Q[r, :] \right)$$

$$= \sum_{r=1}^{R} ((U_1^T \bar{h}_v^1)[r] \cdots (U_L^T \bar{h}_v^L)[r]) Q[:, r]$$

$$= Q \left( U_1^T \bar{h}_v^1 \odot \cdots \odot U_L^T \bar{h}_v^L \right).$$
CP approximation
Graphical Representation

CP-HRTM

\[ H_{v_l} \perp H_{v_l'} \mid R_v \]

Child contributions are **independent**!

CP-RecNTN

The number of parameters required is \( O(LCR) \).
The **HOSVD** approximate $\mathbf{T}[j_1, \ldots, j_L, k]$ as:

$$
\sum_{r=1}^{R} \sum_{r_1=1}^{R} \cdots \sum_{r_L=1}^{R} \mathbf{G}[r_1, \ldots, r_L, r] \mathbf{U}_1[j_1, r_1] \cdots \mathbf{U}_L[j_L, r_L] \mathbf{Q}[k, r],
$$

Thus, the **HOSVD-HRTM** defines the following state-transition distr.: 

$$
P(h_v) = \sum_{r_v} P(h_v \mid r_v, \mathbf{Q}) \sum_{r_{v1}} \cdots \sum_{r_{vL}} P(r_v \mid r_{v1}, \ldots, r_{vL}, \mathbf{G}) \prod_{l=1}^{L} P(r_{vl} \mid h_{vl}, \mathbf{U}_l),
$$

**HOSVD-RecNTN** defines the following state-transition func.: 

$$
h_v = \mathbf{Q} \left( \mathbf{G} \left( \mathbf{U}_1^T \mathbf{h}_{v1}, \ldots, \mathbf{U}_L^T \mathbf{h}_{vL} \right) \right).
$$
The number of parameters required is $O(R^L)$.
TT Approximation

Definition

The TT approximate $T[j_1, \ldots, j_L, k]$ as:

$$
\sum_{r_1=1}^{R} \cdots \sum_{r_L=1}^{R} U_1[j_1, r_1] \cdots U_L[r_{L-1}, j_L, r_L] Q[r_L, k],
$$

Thus, the TT-HRTM defines the following state-transition distr.:

$$
P(h_v) = \sum_{r_{v1}} \cdots \sum_{r_{vL}} P(r_{v1} \mid h_{v1}, U_1) \cdots P(r_{vL} \mid r_{vL-1}, h_{vL}, U_L) P(h_v \mid r_{vL} Q),
$$

TT-RecNTN defines the following state-transition func.:

$$
h_v = Q^T \left( U_L \left( \cdots U_2 \left( U_1^T \tilde{h}_{v1}, \tilde{h}_{v2} \right) \cdots, \tilde{h}_{vL} \right) \right).
$$
The number of parameters required is $O(LCR^2)$.
We evaluate the following models:

<table>
<thead>
<tr>
<th>Approx.</th>
<th>Recursive Models</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Approx.</td>
</tr>
<tr>
<td>None</td>
<td>Full-HRTM</td>
</tr>
<tr>
<td>CP</td>
<td>CP-HRTM</td>
</tr>
<tr>
<td>HOSVD</td>
<td>HOSVD-HRTM</td>
</tr>
<tr>
<td>TT</td>
<td>TT-HRTM</td>
</tr>
</tbody>
</table>

On two tasks:
- the **BoolSent task**, $L \in 2, 3, 4, 5$ (synthetic);
- the **ListOps task** [18], $L = 5$.

¹ *Long Short-Term Memory* [10] is an architecture commonly used on sequence that combines multiple NNs.
Experimental Analysis
Tasks Description & Experimental Settings

**BoolSent**

```
  OR
 /    \
IMPLY 0   AND
   /    \
  0    1
```

**ListOps**

```
  MIN
 /    \
2     MED
   /    \
  1     4
   /    \
  MAX   5
     /  \
  4    5
```

Experimental settings:

- all the **probabilistic models** are trained using **EM algorithm**;
- all the **neural models** are trained using **back-propagation**;
- we **validate C and R** on all models;
- the **input labels** are used to select **different parametrisations**.
BoolSent Task
Results

### Probabilistic

<table>
<thead>
<tr>
<th>Method</th>
<th>Test Results</th>
<th>Validation Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>SP-HRTM</td>
<td><img src="image1.png" alt="Graph" /></td>
<td><img src="image2.png" alt="Graph" /></td>
</tr>
<tr>
<td>Full-HRTM</td>
<td><img src="image3.png" alt="Graph" /></td>
<td><img src="image4.png" alt="Graph" /></td>
</tr>
<tr>
<td>CP-HRTM</td>
<td><img src="image5.png" alt="Graph" /></td>
<td><img src="image6.png" alt="Graph" /></td>
</tr>
<tr>
<td>HOSVD-HRTM</td>
<td><img src="image7.png" alt="Graph" /></td>
<td><img src="image8.png" alt="Graph" /></td>
</tr>
<tr>
<td>TT-HRTM</td>
<td><img src="image9.png" alt="Graph" /></td>
<td><img src="image10.png" alt="Graph" /></td>
</tr>
</tbody>
</table>

### One-layer NN

<table>
<thead>
<tr>
<th>Method</th>
<th>Test Results</th>
<th>Validation Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sum-RecNTN</td>
<td><img src="image11.png" alt="Graph" /></td>
<td><img src="image12.png" alt="Graph" /></td>
</tr>
<tr>
<td>Full-RecNTN</td>
<td><img src="image13.png" alt="Graph" /></td>
<td><img src="image14.png" alt="Graph" /></td>
</tr>
<tr>
<td>CP-RecNTN</td>
<td><img src="image15.png" alt="Graph" /></td>
<td><img src="image16.png" alt="Graph" /></td>
</tr>
<tr>
<td>HOSVD-RecNTN</td>
<td><img src="image17.png" alt="Graph" /></td>
<td><img src="image18.png" alt="Graph" /></td>
</tr>
<tr>
<td>TT-RecNTN</td>
<td><img src="image19.png" alt="Graph" /></td>
<td><img src="image20.png" alt="Graph" /></td>
</tr>
</tbody>
</table>

### LSTM-based

<table>
<thead>
<tr>
<th>Method</th>
<th>Test Results</th>
<th>Validation Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sum-LSTM</td>
<td><img src="image21.png" alt="Graph" /></td>
<td><img src="image22.png" alt="Graph" /></td>
</tr>
<tr>
<td>Full-LSTM</td>
<td><img src="image23.png" alt="Graph" /></td>
<td><img src="image24.png" alt="Graph" /></td>
</tr>
<tr>
<td>CP-LSTM</td>
<td><img src="image25.png" alt="Graph" /></td>
<td><img src="image26.png" alt="Graph" /></td>
</tr>
<tr>
<td>HOSVD-LSTM</td>
<td><img src="image27.png" alt="Graph" /></td>
<td><img src="image28.png" alt="Graph" /></td>
</tr>
<tr>
<td>TT-LSTM</td>
<td><img src="image29.png" alt="Graph" /></td>
<td><img src="image30.png" alt="Graph" /></td>
</tr>
</tbody>
</table>

### Num. of params

- **Probabilistic**
  - SP-HRTM: $10^3$ to $10^5$
  - Full-HRTM: $10^4$ to $10^6$
  - CP-HRTM: $10^5$ to $10^7$
  - HOSVD-HRTM: $10^5$ to $10^7$
  - TT-HRTM: $10^6$ to $10^7$

- **One-layer NN**
  - Sum-RecNTN: $10^4$ to $10^6$
  - Full-RecNTN: $10^5$ to $10^7$
  - CP-RecNTN: $10^6$ to $10^7$
  - HOSVD-RecNTN: $10^6$ to $10^7$
  - TT-RecNTN: $10^7$

- **LSTM-based**
  - Sum-LSTM: $10^5$ to $10^7$
  - Full-LSTM: $10^6$ to $10^7$
  - CP-LSTM: $10^6$ to $10^7$
  - HOSVD-LSTM: $10^6$ to $10^7$
  - TT-LSTM: $10^7$
ListOps Task

Results

Test Results

<table>
<thead>
<tr>
<th></th>
<th>HRTM</th>
<th>RecNTN</th>
<th>LSTM</th>
</tr>
</thead>
<tbody>
<tr>
<td>Full</td>
<td>OOR</td>
<td>60.1 (1.0)</td>
<td>75.5 (1.2)</td>
</tr>
<tr>
<td>CP</td>
<td>22.7 (0.1)</td>
<td>94.3 (0.8)</td>
<td>94.2 (0.2)</td>
</tr>
<tr>
<td>HOSVD</td>
<td>33.9 (0.8)</td>
<td>96.2 (0.1)</td>
<td>97.8 (0.6)</td>
</tr>
<tr>
<td>TT</td>
<td>75.6 (2.7)</td>
<td>93.0 (0.4)</td>
<td>97.2 (0.2)</td>
</tr>
<tr>
<td>Existing</td>
<td>27.2 (2.5)</td>
<td>76.4 (0.1)</td>
<td>79.9 (1.0)</td>
</tr>
</tbody>
</table>

Validation Results

**Probabilistic**

- SP-HRTM
- CP-HRTM
- HOSVD-HRTM
- TT-HRTM

**One-layer NN**

- Sum-RecNTN
- Full-RecNTN
- CP-RecNTN
- HOSVD-RecNTN
- TT-RecNTN

**LSTM-based**

- Sum-LSTM
- Full-LSTM
- CP-LSTM
- HOSVD-LSTM
- TT-LSTM

Daniele Castellana | A Tensor Framework for Learning in Structured Domains
ListOps Task
Computational Complexity Analysis

<table>
<thead>
<tr>
<th>Num. of params</th>
<th>Avg. Time Tr. Epoch (s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>OOR</td>
</tr>
<tr>
<td>5000</td>
<td>SP-HRTM</td>
</tr>
<tr>
<td>10000</td>
<td>Full-HRTM</td>
</tr>
<tr>
<td>15000</td>
<td>CP-HRTM</td>
</tr>
<tr>
<td>20000</td>
<td>HOSVD-HRTM</td>
</tr>
<tr>
<td>400</td>
<td>Sum-RecNTN</td>
</tr>
<tr>
<td>600</td>
<td>Full-RecNTN</td>
</tr>
<tr>
<td>800</td>
<td>CP-RecNTN</td>
</tr>
<tr>
<td>1000</td>
<td>HOSVD-RecNTN</td>
</tr>
<tr>
<td>1200</td>
<td>TT-RecNTN</td>
</tr>
<tr>
<td>2000</td>
<td>Sum-LSTM</td>
</tr>
<tr>
<td>4000</td>
<td>Full-LSTM</td>
</tr>
<tr>
<td>6000</td>
<td>CP-LSTM</td>
</tr>
<tr>
<td>8000</td>
<td>HOSVD-LSTM</td>
</tr>
<tr>
<td>10000</td>
<td>TT-LSTM</td>
</tr>
</tbody>
</table>

In the neural case:

- **sum-based** models are the **least demanding** ones;
- **full-tensorial models** are **more efficient** than models which leverage tensor decompositions (with the same number of parameters);
- the **TT approximation are the most demanding** one due to the sequential processing of hidden child nodes imposed.
There are domains where the value of $L$ (i.e. the maximum out-degree) is unknown:

- abstract syntax trees for programs
- constituency trees for sentences

Recursive models defined so far cannot be applied!

A common approach to fix $L$ is to binarize the input structure.

We propose to apply a weight sharing constraint to the model parameters used to process child hidden states.

We obtain infinite tensor approximations where factors are shared.

HOSVD cannot be extended in this fashion since the core tensor order depends on $L$.

D. Castellana and D. Bacciu. “Learning from Non-Binary Constituency Trees via Tensor Decomposition”. In: 28th International Conference on Computational Linguistic. 2020
Infinite CP approximation

The **infinite CP approximation** shares the first $L$ mode matrices, i.e. the mode matrices which process the child hidden states.

\[
T[j_1, \ldots, j_L, k] \approx \sum_{r=1}^{R} U[j_1, r] \ldots U[j_L, r] Q[k, r],
\]

Infinite-CP-HRTM

\[
\begin{align*}
H_v \\
R_v \\
H_{v1} & \quad \cdots \quad H_{vL}
\end{align*}
\]

Infinite-CP-RecNTN

\[
\begin{align*}
h_v \in \mathbb{R}^C \\
Q \\
U & \quad \cdots \quad U \\
h_1 \in \mathbb{R}^C & \quad \cdots \quad h_L \in \mathbb{R}^C
\end{align*}
\]
The **CP approximation** with **shared mode matrices** can represent any **symmetric tensor** [6].

Infinite-CP is a **permutational invariant** aggregation.
The infinite TT approximation shares the first \( L \) TT-tensors, i.e. the tensors which process the child hidden states.

\[
T[j_1, \ldots, j_L, k] \approx \sum_{r_1=1}^{R} \cdots \sum_{r_L=1}^{R} U[j_1, r_1] \cdots U[r_{L-1}, j_L, r_L] Q[r_L, k],
\]

The infinite-TT approximation depends on the children order.
Experimental Analysis on NLP tasks

We evaluate the following models:

<table>
<thead>
<tr>
<th>Approx.</th>
<th>Infinite Models</th>
<th>Binary Models</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Infinite Models</td>
<td>LSTM-based</td>
</tr>
<tr>
<td></td>
<td>Probabilistic</td>
<td></td>
</tr>
<tr>
<td>CP</td>
<td>Infinite-CP-HRTM</td>
<td>Infinite-CP-LSTM</td>
</tr>
<tr>
<td>TT</td>
<td>Infinite-TT-HRTM</td>
<td>Infinite-TT-LSTM</td>
</tr>
<tr>
<td>Existing</td>
<td>Infinite-SP-HRTM</td>
<td>Binary Sum-LSTM [21]</td>
</tr>
</tbody>
</table>

On NLP tasks:

- **SST-5** [20]: sentiment classification with 5 labels;
- **SST-2** [20]: sentiment classification with 2 labels;
- **TREC** [13]: question classification with 6 labels;
- **SICK-R** [14]: sentence pairs relatedness score;
- **SICK-E** [14]: sentence pairs entailment.

The root hidden state are the sentence encodings. The root hidden states of two sentences are combined together.
## Results

<table>
<thead>
<tr>
<th>Tasks</th>
<th>SST-5</th>
<th>SST-2</th>
<th>SICK-E</th>
<th>SICK-R</th>
<th>TREC</th>
</tr>
</thead>
<tbody>
<tr>
<td>Unbounded</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Infinite-SP-HRTM</td>
<td>17.3 (0.0)</td>
<td>49.9 (0.2)</td>
<td>-</td>
<td>-</td>
<td>31.7 (6.3)</td>
</tr>
<tr>
<td>Infinite-CP-HRTM</td>
<td>17.3 (0.0)</td>
<td>49.7 (0.0)</td>
<td>-</td>
<td>-</td>
<td>18.1 (1.0)</td>
</tr>
<tr>
<td>Infinite-TT-HRTM</td>
<td>17.3 (0.0)</td>
<td>49.7 (0.0)</td>
<td>-</td>
<td>-</td>
<td>18.8 (0.0)</td>
</tr>
<tr>
<td>Infinite-Sum-LSTM</td>
<td><strong>49.4</strong> (0.6)</td>
<td><strong>85.5</strong> (0.8)</td>
<td><strong>82.6</strong> (0.4)</td>
<td><strong>84.9</strong> (0.2)</td>
<td><strong>91.9</strong> (1.0)</td>
</tr>
<tr>
<td>Infinite-CP-LSTM</td>
<td><strong>48.3</strong> (0.8)</td>
<td><strong>85.3</strong> (0.3)</td>
<td><strong>84.2</strong> (0.4)</td>
<td><strong>86.4</strong> (0.1)</td>
<td>90.0 (0.7)</td>
</tr>
<tr>
<td>Infinite-TT-LSTM</td>
<td><strong>48.2</strong> (0.5)</td>
<td><strong>86.8</strong> (0.1)</td>
<td><strong>83.9</strong> (0.1)</td>
<td><strong>85.6</strong> (0.2)</td>
<td>90.7 (0.6)</td>
</tr>
<tr>
<td>Bin.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Binary Sum-LSTM</td>
<td><strong>51.5</strong> (0.7)</td>
<td><strong>87.9</strong> (0.2)</td>
<td><strong>82.3</strong> (0.5)</td>
<td><strong>84.3</strong> (0.7)</td>
<td><strong>92.3</strong> (0.8)</td>
</tr>
<tr>
<td>TreeNet</td>
<td><strong>48.4</strong> (1.5)</td>
<td><strong>87.0</strong> (0.5)</td>
<td><strong>81.2</strong> (0.3)</td>
<td><strong>84.5</strong> (0.3)</td>
<td><strong>91.3</strong> (1.1)</td>
</tr>
</tbody>
</table>

- **Probabilistic models** fail to build sentence encodings:
  - **root hidden states** do not encode useful information;

In the neural case:

- **binary models** outperform infinite models on the **SST** datasets:
  - data are originally binary. Unbounded dataset contains **less** trees;
- **tensor-based models** perform well on **SICK** dataset;
- **sum-based models** are suitable for the **TREC** dataset.
An example on the SICK dataset

A: The girl has red hair and eyebrows, several piercings in a ear and a tattoo on the back.
B: The girl has red hair and eyebrows, several piercings in a ear and no tattoo on the back.
To recap:

- **Tensors** play a key role to define **aggregation functions**;
- **full-tensor models** act as a mould to define **new** recursive model;
- **tensor decomposition** reduce **model complexity** by introducing specific **inductive bias**;
  - decomposition ranks control the trade-off between **expressiveness** and **compression**;
- **tensor decomposition + weight sharing** to handle **unbounded structures**;
  - different decomposition $\implies$ different behaviour
  - Infinite CP $\implies$ symmetric tensor
Future Directions

- Deepen the theoretical aspect of the tensor framework:
  - can we relate the task properties with the state-transition properties?

- Apply tensors to aggregate information in other contexts:
  - on sets (i.e. learn a function $f(x_1, \ldots, x_N)$, where $\{x_1, \ldots, x_N\}$ is the input set);
    - function must be permutational invariant;
    - NN + sum is universal [22];
  - on graphs to define neighbourhood aggregation functions;
    - number of neighbours is usually unbounded;
    - neighbourhood is considered a multi-set;

- Tensors beyond aggregation:
  - tensors arise on graphs when high-order Weisfeiler-Lehman (WL) test is considered;
    - WL is a test for the graph isomorphism problem (NP);
    - k-WL associate a colour for each $k$-tuple of graph nodes [15, 17];
    - can we use TD to derive a principled approximation of k-WL test?
Thank you for your attention!

Questions?

Daniele Castellana
daniele.castellana@di.unipi.it
http://pages.di.unipi.it/castellana
https://ciml.di.unipi.it/
"Most. freq." is a dummy model which simply outputs the most frequent output for each operator. Thus, it ignores the structure.
References I


References VI

