SoPa++: Leveraging explainability from hybridized RNN, CNN and weighted finite-state neural architectures

Atreya Shankar, shankar@uni-potsdam.de
Cognitive Systems: Language, Learning, and Reasoning (M.Sc.)

Thesis Progress Update
University of Potsdam, WiSe 20/21

March 7, 2021

Recap

1. Source code refactored to work with latest `PyTorch` and `CUDA` versions
2. Facebook NLU multi-class intent detection data set from Schuster et al. (2018)
3. Performance is important for explainability, because it does not make sense to explain a poorly performing model
4. Focus would be to develop SoPa++ as a neural model which can be easily decomposed into a *globally* explainable model

Semirings

Kuich and Salomaa 1986:
A semiring is a set $\mathbb{K}$ along with two binary associative operations $\oplus$ (addition) and $\otimes$ (multiplication) and two identity elements: $\bar{0}$ for addition and $\bar{1}$ for multiplication. Semirings require that addition is commutative, multiplication distributes over addition, and that multiplication by $\bar{0}$ annihilates, i.e., $\bar{0} \otimes a = a \otimes \bar{0} = \bar{0}$

- Generic semiring notation: $\langle \mathbb{K}, \oplus, \otimes, \bar{0}, \bar{1} \rangle$
- Max-sum semiring: $\langle \mathbb{R} \cup \{-\infty\}, \max, +, -\infty, 0 \rangle$
- Max-product semiring: $\langle \mathbb{R} \cup \{-\infty\}, \max, \times, -\infty, 1 \rangle$
- Many semirings are possible, but we stick to *max* based semirings

Weighted Finite-State Automata

$$p(\boldsymbol{x}) = \boldsymbol{\pi} \otimes \left(\bigotimes_{i=1}^{n} \boldsymbol{T}(x_i) \right) \otimes \boldsymbol{\eta} \tag{1}$$

$$s(\boldsymbol{z}) = \bigoplus_{\boldsymbol{x} \in \Pi(\boldsymbol{z})} p(\boldsymbol{x}) \tag{2}$$

- Path $\boldsymbol{x} = \langle x_1, x_2, \ldots, x_n \rangle$ derives some string $\boldsymbol{z} = \langle z_1, z_2, \ldots, z_m \rangle$
- $\Pi(\boldsymbol{z})$ is the set of all possible paths that derive string $\boldsymbol{z}$
- $\boldsymbol{\pi}$, $\boldsymbol{\eta}$ are start and end state vectors
- $\boldsymbol{T}$ is a transition matrix defining transition scores between states
- $p(\boldsymbol{x})$ refers to a score through a particular path
- $s(\boldsymbol{z})$ refers to an aggregate score through all possible paths
- Utilize Viterbi algorithm with *max* based semirings to compute $s(\boldsymbol{z})$ (Viterbi, 1967)

Finite transition types

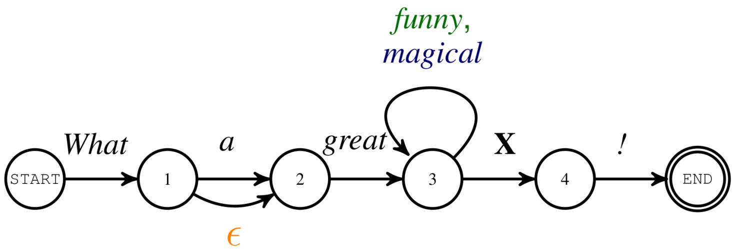

Figure 1: Schematic for finite-state transitions; ϵ represents an epsilon-transition, "funny" and "magical" form self-loops and all other transitions are main transitions (Schwartz, Thomson, and Smith, 2018)

- Vanilla SoPa allowed 3 transitions types: main, epsilon and self-loops
- **SoPa++ uses only two transitions: namely main and wildcard transitions**
- Both consume one token and one state, which leads to pattern-string determinism and could make the *explainability* step easier

SoPa++: Lower Model

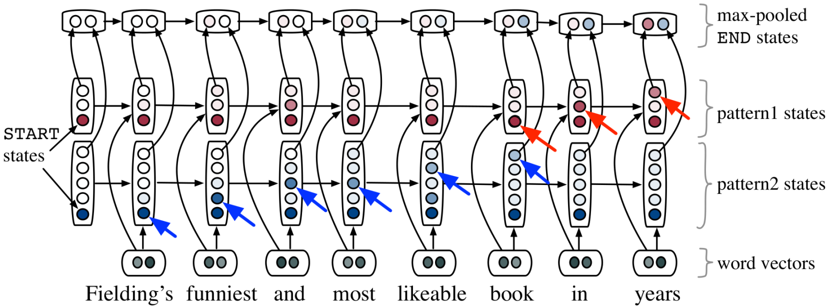

Figure 2: SoPa++ lower model schematic where pattern 1 is activated by the substring "in years" and pattern 2 is activated by the substring "funniest and most likeable book" (Schwartz, Thomson, and Smith, 2018)

- Vanilla SoPa places a MLP on top of the max-pooled states to retrieve classification results
- This contributes to a lack of explainability since MLPs are generally not easily explainable

SoPa++: Upper Model

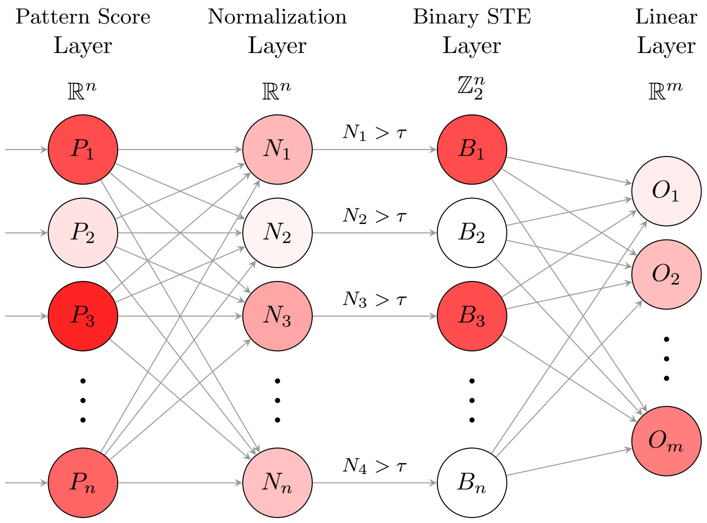

Figure 3: SoPa++ upper model schematic (combined lower-upper schematic under-construction); STE refers to a straight-through-estimator (Yin et al., 2019)

SoPa++: Explainable Model

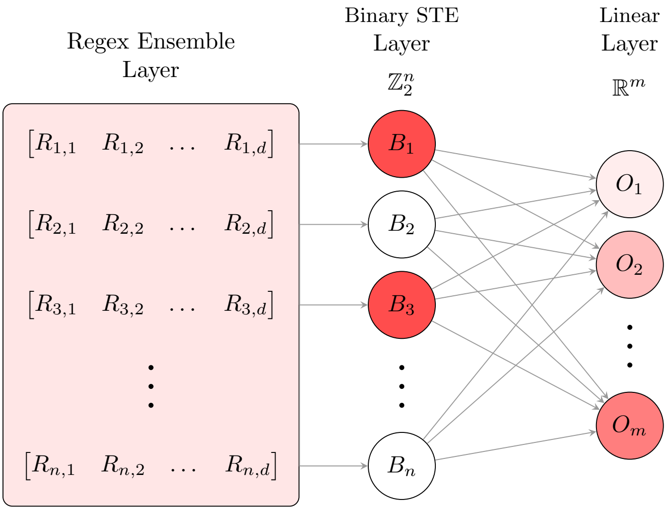
Regex Ensemble Layer
Binary STE Layer
$\mathbb{Z}_2^n$
Linear Layer
$\mathbb{R}^m$
$[R_{1,1} \quad R_{1,2} \quad \ldots \quad R_{1,d}]$
$[R_{2,1} \quad R_{2,2} \quad \ldots \quad R_{2,d}]$
$[R_{3,1} \quad R_{3,2} \quad \ldots \quad R_{3,d}]$
$[R_{n,1} \quad R_{n,2} \quad \ldots \quad R_{n,d}]$
B_1
B_2
B_3
B_n
O_1
O_2
O_m

Figure 4: SoPa++ explainable variant derived from antecedent neural model

Training and Convergence

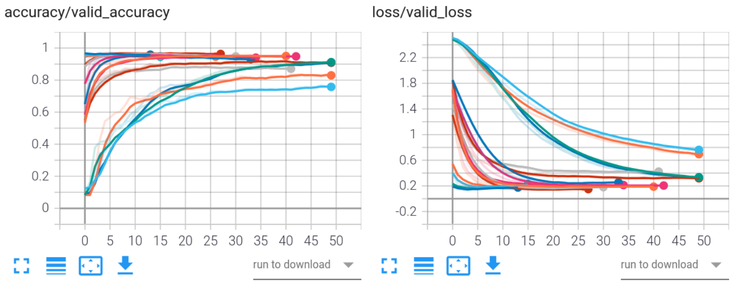

Figure 5: Excerpt from TensorBoard training logs for SoPa++ grid-search

- Model training procedure tends to converge fast
- Best model checkpoint(s) were reached within 1-6 epochs

Evaluation

Patterns	Test Precision	Test Recall	Test F_1
6−10_5−10_4−10_3−10	0.982	0.981	0.982
6−50_5−50_4−50_3−50	0.988	0.987	**0.987**

Table 1: Performance summary of various SoPa++ models; evaluation metrics are based on weighted averages

- Performance is competitive with that from literature and online benchmarks, which show F_1 scores ranging from 96-99% (Schuster et al., 2018)
- In both the above cases, the max-sum semiring outperforms the max-product semiring
- Even though the SoPa++ model variant with higher number of patterns performs better, a **smaller model** might be more beneficial for explainability

Explainability: Peek into the Regex Ensemble

Pattern length	Sample regular expressions
3	{* forecast}, {* activate}, {* weather}
4	{activate morning *}, {my alarm *}, {* minute snooze}
5	{add * reminder for}, {the boiled eggs *}
6	{to wish * * happy}, {* add * * to}

Table 2: Sample regular expressions that activate patterns of specified lengths; * is used for aesthetic purposes to represent a wildcard instead of the appropriate `\w+` regular expression

- Patterns above were processed from a small subset of the training data; the full processing is still under development
- Patterns are indicative of important (generalized) n-grams in text
- Analyzing the weights of the linear layer can tell us how important each pattern is for the prediction of each class

Next steps

1. Find a clever way for clustering/merging regular expressions to keep the size of explainable model small
2. Analyze the performance of both SoPa++ and its explainable variant on the test set to measure how different their performances are
3. For cases where the explainable model does not perform similar to the SoPa++ neural model, propose local sample explanations as alternative

Bibliography I

Kuich, Werner and Arto Salomaa (1986). "Linear Algebra". In: *Semirings, automata, languages*. Springer, pp. 5–103.

Schuster, Sebastian et al. (2018). "Cross-lingual transfer learning for multilingual task oriented dialog". In: *arXiv preprint arXiv:1810.13327*.

Schwartz, Roy, Sam Thomson, and Noah A. Smith (July 2018). "Bridging CNNs, RNNs, and Weighted Finite-State Machines". In: *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Melbourne, Australia: Association for Computational Linguistics, pp. 295–305. DOI: 10.18653/v1/P18-1028. URL: https://www.aclweb.org/anthology/P18-1028.

Viterbi, Andrew (1967). "Error bounds for convolutional codes and an asymptotically optimum decoding algorithm". In: *IEEE transactions on Information Theory* 13.2, pp. 260–269.

Bibliography II

Yin, Penghang et al. (2019). "Understanding straight-through estimator in training activation quantized neural nets". In: *arXiv preprint arXiv:1903.05662*.