The Credibility Transformer

Mario V. Wüthrich
RiskLab, ETH Zurich

Joint work with Ronald Richman (Old Mutual Insure) and
Salvatore Scognamiglio (University of Naples Parthenope)

Waseda University, Tokyo

October 28, 2024

Table of Contents

- **Section 1: Feature Tokenizer and the CLS Token**

Introduction

- Attention layers and Transformers of Vaswani et al. (2017) celebrate huge success in large language models (LLMs) like ChatGPT.

- These network architectures process time-series data.

- There are only a few applications of Transformers to tabular data; Huang et al. (2020), Kuo–Richman (2021), Brauer (2024).

- These applications require to tokenize tabular data; Gorishniy et al. (2021).

- First attempts are not fully convincing:
  - ⋆ Training of such architectures seems difficult.
  - ⋆ Not all information is equally important and credible.

- The Credibility Transformer equips Transformers with a credibility mechanism.

- For this it uses a special token called classify (CLS) token; Devlin et al. (2018).

Construction of input tensor

We need to pre-process the input data to make it suitable for Transformers.

This includes the following steps:

(1) Feature tokenizer takes care of categorical and continuous covariates:

  (a) Entity embedding of categorical covariates.
  (b) Tokenization of continuous covariates.

(2) Positional encoding.

(3) Classify (CLS) token.

We discuss these steps in the following slides.

Pre-processing of categorical covariates

- In actuarial pricing there are many categorical covariates, and many of these categorical covariates are of nominal type. E.g., car brands like Toyota, Nissan, Honda, Mitsubishi, Mazda, Subaru, Suzuki, Daihatsu, ...

- To use categorical covariates in regression models, one needs to bring them into a numerical representation by embedding them into a Euclidean space.

- Assume the categorical covariate x has L levels $\mathcal{A} = \{a_1, \ldots, a_L\}$. One hot-encoding maps each level $a_l \in \mathcal{A}$ to a unit basis vector of $\mathbb{R}^L$

$$x \in \mathcal{A} \quad \mapsto \quad \left(\mathbb{1}_{\{x=a_1\}}, \ldots, \mathbb{1}_{\{x=a_L\}}\right)^\top \in \mathbb{R}^L.$$

- One-hot encoding implies that all different levels are orthogonal in $\mathbb{R}^L$ and there is no notion of similarity (or adjacency).

- **Question:** Can we learn a more dense representation?

One-hot encoding vs. more dense representation

	Actuary	Accountant	Quant	Statistician	Economist	Underwriter
Actuary	1	0	0	0	0	0
Accountant	0	1	0	0	0	0
Quant	0	0	1	0	0	0
Statistician	0	0	0	1	0	0
Economist	0	0	0	0	1	0
Underwriter	0	0	0	0	0	1

	Finance	Maths	Statistics	Liabilities
Actuary	0.5	0.25	0.5	0.5
Accountant	0.5	0	0	0
Quant	0.75	0.25	0.25	0
Statistician	0	0.5	0.85	0
Economist	0.5	0.25	0.5	0
Underwriter	0	0.1	0.05	0.75

The latter maps the $L = 6$ different levels to a smaller space $\mathbb{R}^b$, with $b = 4$, and more similar job profiles are closer in $\mathbb{R}^4$.

Entity embedding of categorical covariates

- Assume there are T_1 categorical covariates $(x_t)_{t=1}^{T_1}$ in feature vector $\boldsymbol{x} = (x_t)_{t=1}^{T}$.
- Assume the t-th categorical covariate x_t has L_t levels $\mathcal{A}_t = \{a_1, \ldots, a_{L_t}\}$.
- Select a (fixed) embedding dimension b (being independent of t).
- An entity embedding (EE) of covariate x_t is obtained by a mapping

$$\boldsymbol{e}_t^{\mathrm{EE}} : \mathcal{A}_t \to \mathbb{R}^b, \qquad x_t \mapsto \boldsymbol{e}^{\mathrm{EE}}(x_t).$$

- This entity embedding involves $L_t \cdot b$ parameters (to be learned).
- This gives us an input tensor of the categorical information

$$(x_t)_{t=1}^{T_1} \;\mapsto\; \left[\boldsymbol{e}_1^{\mathrm{EE}}(x_1), \ldots, \boldsymbol{e}_{T_1}^{\mathrm{EE}}(x_{T_1})\right] \;\in\; \mathbb{R}^{b \times T_1}.$$

- This input tensor has the same structure as a time-series.

Tokenization of continuous covariates

- Consider the continuous covariates $(x_t)_{t=T_1+1}^{T}$ in feature vector $\boldsymbol{x} = (x_t)_{t=1}^{T}$.

- These continuous covariates do not need any transformation for neural network modeling. However, we would like to bring them into the same tensor structure as the entity embedding of the categorical covariates.

- Select fully-connected feed-forward neural networks (FNNs)

$$x_t \mapsto \boldsymbol{z}_t^{(2:1)}(x_t) = \left(\boldsymbol{z}_t^{(2)} \circ \boldsymbol{z}_t^{(1)}\right)(x_t) \in \mathbb{R}^b,$$

e.g., of depth 2 being composed of two FNN layers

$$\boldsymbol{z}_t^{(1)} : \mathbb{R} \to \mathbb{R}^b \qquad \text{and} \qquad \boldsymbol{z}_t^{(2)} : \mathbb{R}^b \to \mathbb{R}^b.$$

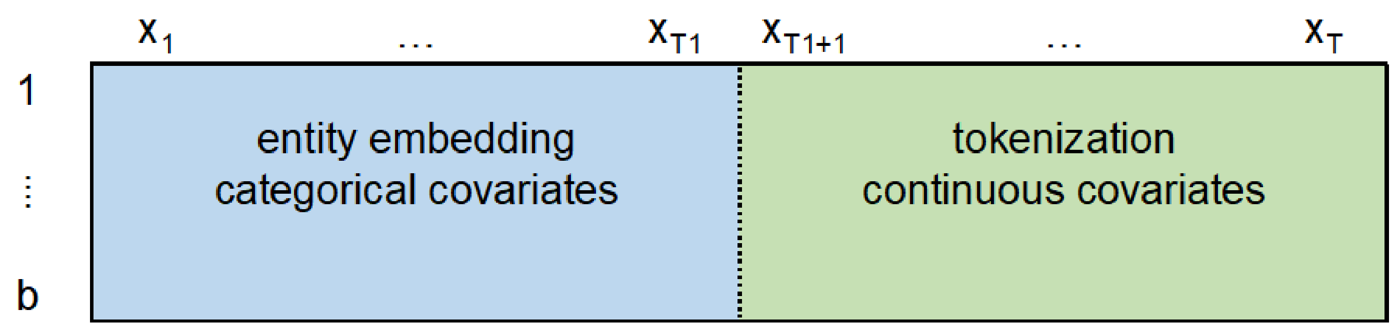
- This embeds each continuous (real-valued) covariate x_t into $\mathbb{R}^b$.

Raw input tensor

- Concatenate the entity embeddings of the categorical covariates $(x_t)_{t=1}^{T_1}$ and the FNN tokenizations of the continuous covariates $(x_t)_{t=T_1+1}^{T}$.

- This gives us the **raw input tensor**

$$\boldsymbol{x}^{\circ}_{1:T} := \left[\boldsymbol{e}_1^{\mathrm{EE}}(x_1), \ldots, \boldsymbol{e}_{T_1}^{\mathrm{EE}}(x_{T_1}),\, \boldsymbol{z}_{T_1+1}^{(2:1)}(x_{T_1+1}), \ldots, \boldsymbol{z}_{T}^{(2:1)}(x_T)\right] \in \mathbb{R}^{b \times T}.$$



- This was used in Huang et al. (2020), Kuo–Richman (2021), Brauer (2024).

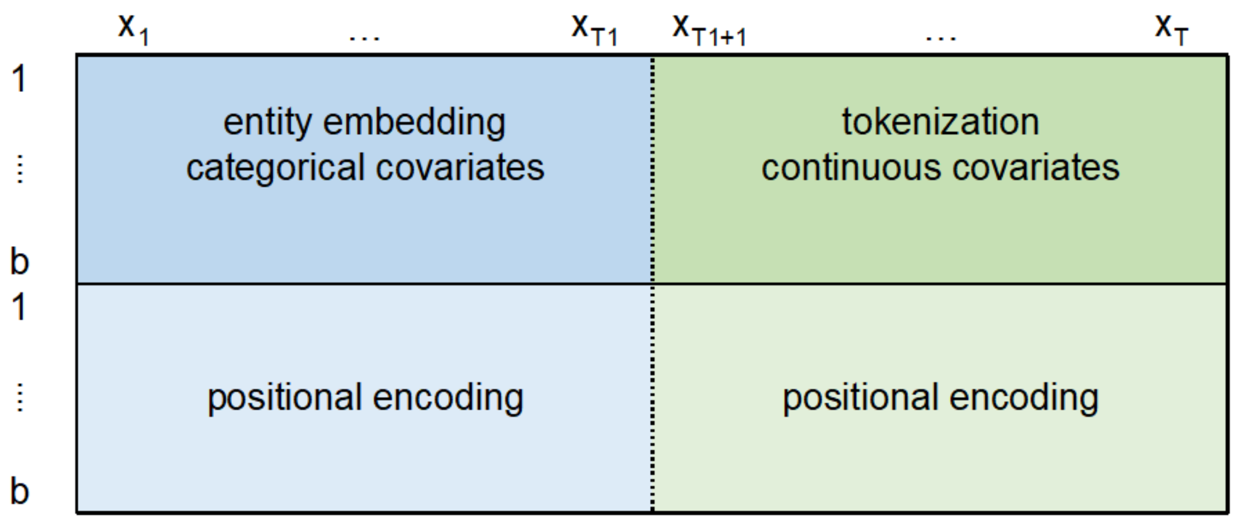
Positional encoding

- The raw input tensor $\boldsymbol{x}^{\circ}_{1:T}$ does not have any notion of time or position.
- Add positional encodings to the raw tensor

$$\boldsymbol{e}^{\mathrm{pos}}:\{1,\ldots,T\}\to\mathbb{R}^{b},\qquad t\mapsto \boldsymbol{e}^{\mathrm{pos}}(t).$$

- This gives us the **input tensor**

$$\boldsymbol{x}_{1:T}:=\begin{bmatrix}\boldsymbol{e}^{\mathrm{EE}}_{1}(x_1) & \cdots & \boldsymbol{e}^{\mathrm{EE}}_{T_1}(x_{T_1}) & \boldsymbol{z}^{(2:1)}_{T_1+1}(x_{T_1+1}) & \cdots & \boldsymbol{z}^{(2:1)}_{T}(x_T)\\ \boldsymbol{e}^{\mathrm{pos}}(1) & \cdots & \boldsymbol{e}^{\mathrm{pos}}(T_1) & \boldsymbol{e}^{\mathrm{pos}}(T_1+1) & \cdots & \boldsymbol{e}^{\mathrm{pos}}(T)\end{bmatrix}\in\mathbb{R}^{2b\times T}.$$



CLS token

- Each component of the input tensor $\boldsymbol{x}_{1:T}$ contains specific input information.

- **New feature:** We add one more token to the input tensor. This additional token does **not** carry any information.

- We call this additional token the **CLS token**. The CLS token has been introduced in BERT[1] by Devlin et al. (2018). It learns classification probabilities for the next part of the sentence in language tasks.

- Our CLS token will not learn probabilities but real numbers. Since technically it works similarly to the CLS token in BERT, we keep the term **CLS token**.

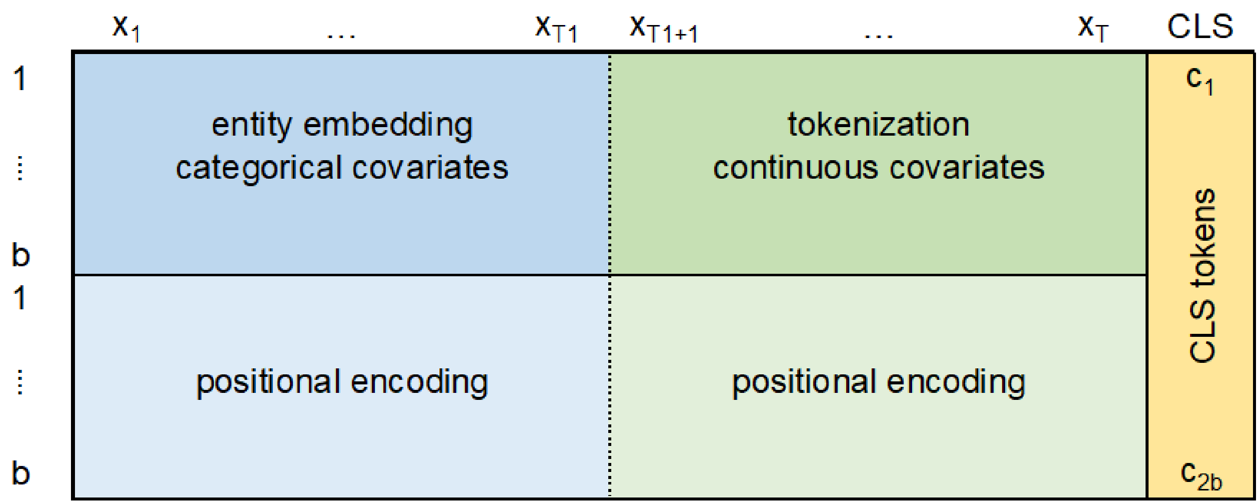
- We use the CLS token for a **credibility mechanism** (further explained below).

[1] BERT = Bidirectional Encoder Representations from Transformers

CLS token augmented input tensor

- The CLS token **augmented input tensor** is defined by

$$\boldsymbol{x}^+_{1:T+1} := \begin{bmatrix} \boldsymbol{e}^{\mathrm{EE}}_1(x_1) & \cdots & \boldsymbol{e}^{\mathrm{EE}}_{T_1}(x_{T_1}) & \boldsymbol{z}^{(2:1)}_{T_1+1}(x_{T_1+1}) & \cdots & \boldsymbol{z}^{(2:1)}_{T}(x_T) & \boldsymbol{c}_1 \\ \boldsymbol{e}^{\mathrm{pos}}(1) & \cdots & \boldsymbol{e}^{\mathrm{pos}}(T_1) & \boldsymbol{e}^{\mathrm{pos}}(T_1+1) & \cdots & \boldsymbol{e}^{\mathrm{pos}}(T) & \boldsymbol{c}_2 \end{bmatrix} \in \mathbb{R}^{2b\times(T+1)}.$$



- We emphasize, before interacting with the other columns of the augmented input tensor $\boldsymbol{x}^+_{1:T+1}$, the CLS token $\boldsymbol{c} = (\boldsymbol{c}_1, \boldsymbol{c}_2) \in \mathbb{R}^{2b}$ does **not** contain any information.

- The input data is now in tensor structure which allows us to apply Transformers.

- **Section 2: Transformer**

Transformers and attention layers

- Transformers of Vaswani et el. (2017) essentially rely on attention layers.

- An attention layer can be seen as a network architecture that allows features to interact with each other.

- There are three different objects involved: queries, keys and values.

- These three objects are processed from time-distributed FNNs applied individually to all components of the augmented input tensor $\boldsymbol{x}^+_{1:T+1} = (x^+_t)_{t=1}^{T+1}$

$$\begin{aligned} \boldsymbol{q}_t &= \boldsymbol{z}_Q\left(x^+_t\right) \in \mathbb{R}^{2b}, \\ \boldsymbol{k}_t &= \boldsymbol{z}_K\left(x^+_t\right) \in \mathbb{R}^{2b}, \\ \boldsymbol{v}_t &= \boldsymbol{z}_V\left(x^+_t\right) \in \mathbb{R}^{2b}, \end{aligned}$$

with query, key and value FNNs $\boldsymbol{z}_Q$, $\boldsymbol{z}_K$ and $\boldsymbol{z}_V$, respectively.

Attention mechanism (1/2)

- Collecting all components $1 \leq t \leq T+1$ gives query, key and value tensors

$$
\begin{aligned}
Q &= [\boldsymbol{q}_1, \ldots, \boldsymbol{q}_{T+1}]^\top \in \mathbb{R}^{(T+1)\times 2b}, \\
K &= [\boldsymbol{k}_1, \ldots, \boldsymbol{k}_{T+1}]^\top \in \mathbb{R}^{(T+1)\times 2b}, \\
V &= [\boldsymbol{v}_1, \ldots, \boldsymbol{v}_{T+1}]^\top \in \mathbb{R}^{(T+1)\times 2b}.
\end{aligned}
$$

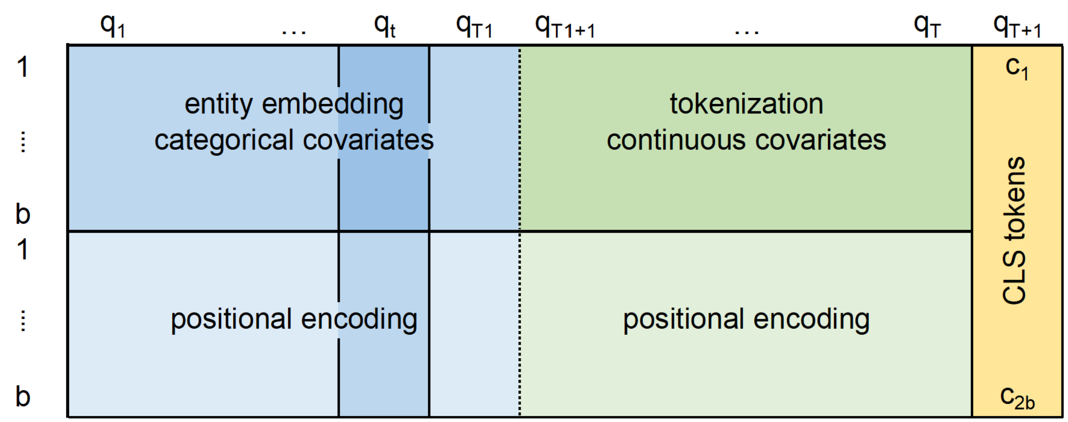
- Note that $\boldsymbol{q}_t = \boldsymbol{q}_t(x_t^+)$, $\boldsymbol{k}_t = \boldsymbol{k}_t(x_t^+)$, $\boldsymbol{v}_t = \boldsymbol{v}_t(x_t^+)$, i.e., each time-component has so far only seen the information of that specific time point t.

- Now we let these components interact (across time t):

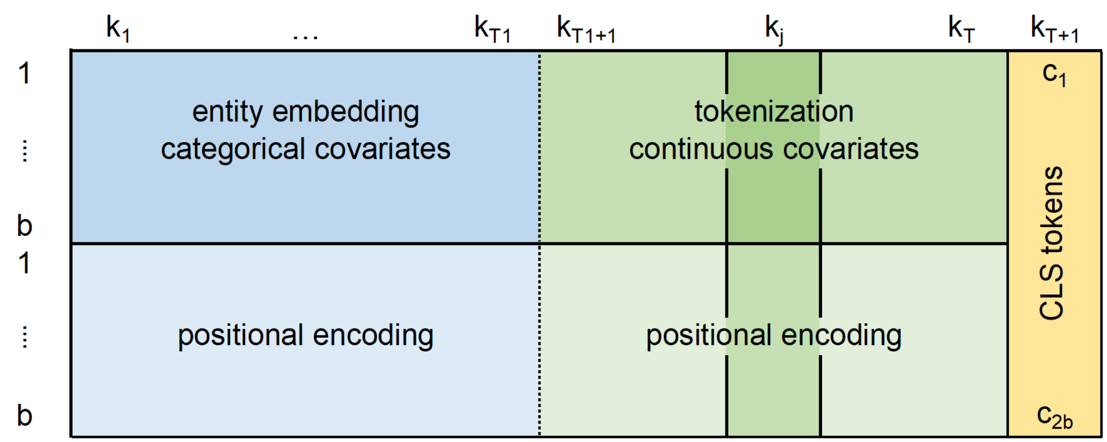
Each query $\boldsymbol{q}_t = \boldsymbol{q}_t(x_t^+)$ searches for keys $\boldsymbol{k}_j = \boldsymbol{k}_j(x_j^+)$ that give a "match".

Attention mechanism (2/2)

- Each query $\boldsymbol{q}_t$ ...



- ... searches for keys $\boldsymbol{k}_j$ ...



- ... that give a “match”. In that case, value $\boldsymbol{v}_j$ gets a high attention for index t.
- Mathematically this is done by the dot/scalar product between queries and keys.

Attention head

- The attention weight matrix A is defined by applying the softmax function to all rows in the following matrix[2]

$$A = \text{softmax}\left(QK^\top\right) \;\in\; \mathbb{R}^{(T+1)\times(T+1)},$$

where the softmax operation is applied row-wise (for fixed query $\boldsymbol{q}_t$)

$$a_{t,j} = \frac{\exp(\boldsymbol{q}_t^\top \boldsymbol{k}_j)}{\sum_{s=1}^{T+1}\exp(\boldsymbol{q}_t^\top \boldsymbol{k}_s)} \;\in\; (0,1), \quad \text{for } j = 1, \ldots, T+1.$$

- The attention head of the Transformer is received by the matrix multiplication

$$H(\boldsymbol{x}_{1:T+1}^+) = (A\,V)^\top \;\in\; \mathbb{R}^{2b\times(T+1)}.$$

- This is a new representation of the augmented input tensor $\boldsymbol{x}_{1:T+1}^+$, paying more attention to more relevant information.

[2] For simplicity we omit the dimension scaling here.

Transformer

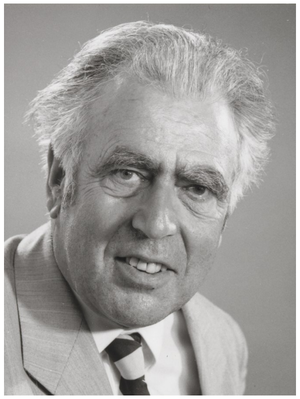
- $\boldsymbol{x}^+_{1:T+1} \in \mathbb{R}^{2b\times(T+1)}$ is the augmented input tensor and ...
- ... $H(\boldsymbol{x}^+_{1:T+1}) \in \mathbb{R}^{2b\times(T+1)}$ is the attention head transformed version thereof.
- They have the same tensor structure, thus, we can use a skip connection.
- In a simplified version, the Transformer essentially considers the new information

$$\boldsymbol{z}^{\text{trans}}(\boldsymbol{x}^+_{1:T+1}) = \boldsymbol{x}^+_{1:T+1} + H(\boldsymbol{x}^+_{1:T+1}) \ \in \ \mathbb{R}^{2b\times(T+1)}.$$

- The full Transformer applies additional FNN transformations, but in essence it is the same; see Vaswani et al. (2017).

- **Section 3: The Credibility Transformer**

Bühlmann credibility



- This step is the essential difference to the classical Transformer.
- For this we take advantage of the integrated **CLS token** $\boldsymbol{c} \in \mathbb{R}^b$:
  - Before running through the attention mechanism, the CLS token has not seen any input information; in the sense of Bühlmann (1967) credibility it is a prior parameter.
  - After the attention mechanism has been exploited, the CLS token has extracted the relevant information of the input $\boldsymbol{x}_{1:T+1}^{+}$, and it can be seen as data guided.
- The main idea is to combine the prior parameter and the data guided information to a Bühlmann credibility formula for optimally training the model.
- Furthermore, in contrast to the classical Transformer, we do not further process the entire Transformer information $\boldsymbol{z}^{\mathrm{trans}}(\boldsymbol{x}_{1:T+1}^{+})$, but only the information encoded in the CLS token, thus, the CLS token acts as an input encoder.

Credibility Transformer

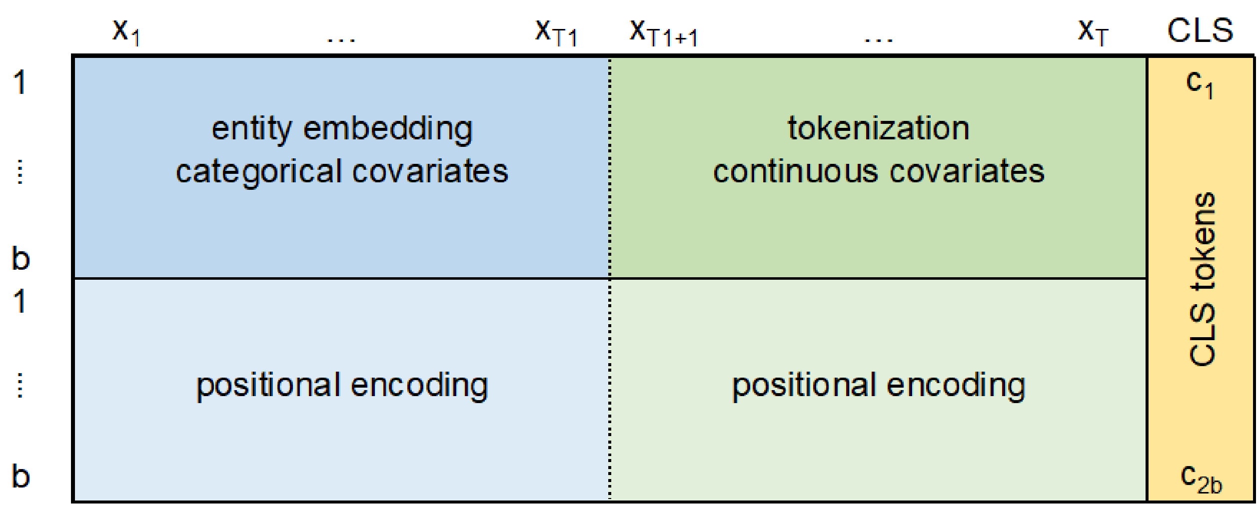
- Prior information $\boldsymbol{c}^{\text{prior}} := \boldsymbol{v}_{T+1} = \boldsymbol{v}_{T+1}(x^{+}_{T+1}) \in \mathbb{R}^{2b}$.

- Data guided information $\boldsymbol{c}^{\text{trans}} := \boldsymbol{z}^{\text{trans}}_{T+1}(\boldsymbol{x}^{+}_{1:T+1}) \in \mathbb{R}^{2b}$.

- Credibilitized information

$$\boldsymbol{c}^{\text{cred}} = Z\,\boldsymbol{c}^{\text{trans}} + (1-Z)\,\boldsymbol{c}^{\text{prior}} \;\in\; \mathbb{R}^{2b},$$

with $Z \sim \text{Bernoulli}(\alpha)$.



Remarks on the Credibility Transformer

- Credibilitized information

$$\boldsymbol{c}^{\text{cred}} = Z\,\boldsymbol{c}^{\text{trans}} + (1-Z)\,\boldsymbol{c}^{\text{prior}} \;\in\; \mathbb{R}^{2b},$$

  with $Z \sim \text{Bernoulli}(\alpha)$.

- We apply this credibility mechanism only during SGD training (similar to drop-out), and for prediction in a trained model we set $Z \equiv 1$.

- This credibilitized version takes for $(1-\alpha) \cdot 100\%$ of the instances in each SGD steps the prior information $\boldsymbol{c}^{\text{prior}}$. This has a smoothing effect during SGD training.

- One can verify that in a properly trained model, $\boldsymbol{c}^{\text{prior}}$ corresponds to the global mean not considering any covariates.

- $\alpha \in [0,1]$ is a hyper-parameter that needs to be optimized by out-of-sample validation (grid search). In our example, we have found $\alpha = 90\%$.

Hidden credibility mechanism

- There is a 2nd (hidden) credibility mechanism involved.

- We come back to the attention weight matrix A. Recall, this matrix is obtained by applying the softmax operation to the rows of matrix $QK^\top$. Thus, the rows of A add up to the total weight of 1.

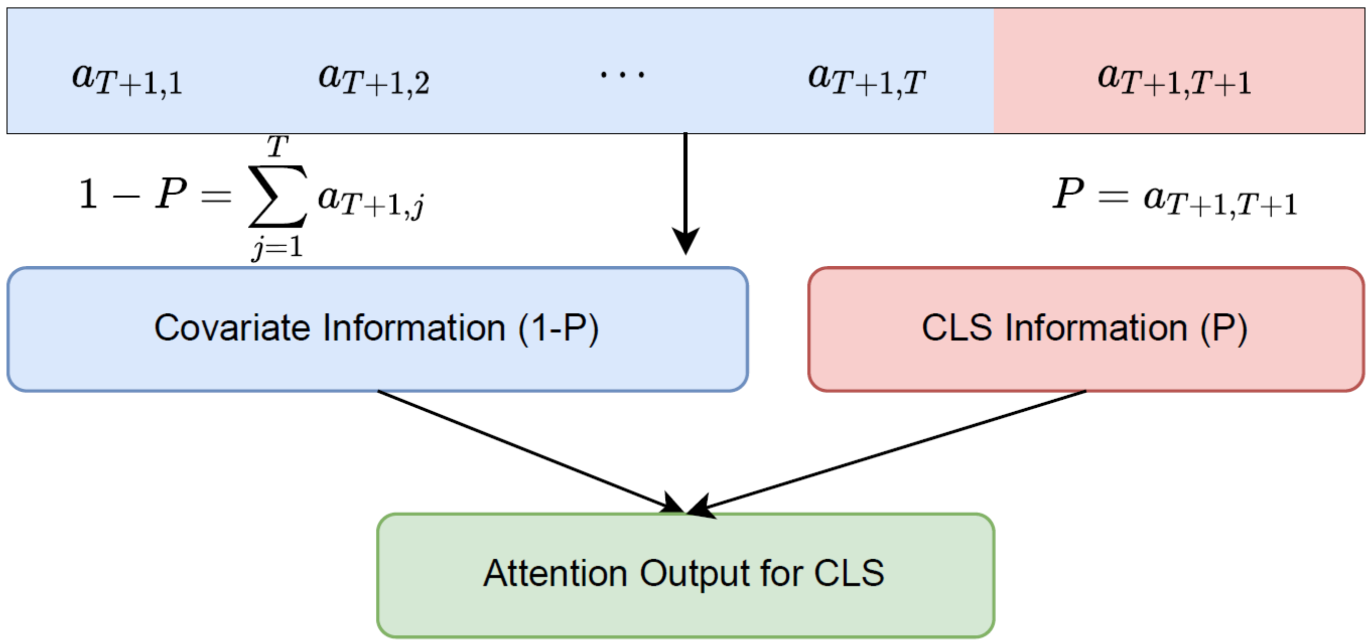
- The last row of A corresponds to the CLS token with attention weights

$$a_{T+1,j} = \frac{\exp(\boldsymbol{q}_{T+1}^\top \boldsymbol{k}_j)}{\sum_{s=1}^{T+1} \exp(\boldsymbol{q}_{T+1}^\top \boldsymbol{k}_s)} \in (0,1), \quad \text{for } j = 1, \ldots, T+1.$$

- This implies that the attention weights on the CLS token $\boldsymbol{c}^{\text{prior}}$ and the values of the input tensor $(\boldsymbol{v}_j(x_j^+))_{j=1}^T$, respectively, are

$$P := a_{T+1,T+1} \in (0,1) \qquad \text{and} \qquad 1 - P = \sum_{j=1}^{T} a_{T+1,j} \in (0,1).$$

Attention Weights for CLS Token



$$\boldsymbol{v}^{\text{trans}} = P \cdot \boldsymbol{v}_{T+1} + (1 - P) \cdot \boldsymbol{v}^{\text{covariate}}$$

with prior information $\boldsymbol{c}^{\text{prior}} = \boldsymbol{v}_{T+1}$ and covariate information (values)

$$\boldsymbol{v}^{\text{covariate}} = \sum_{t=1}^{T} \frac{a_{T+1,t}}{1 - P} \boldsymbol{v}_t(x_t^+).$$

- **Section 4: Decoder**

Decoder for the readout

- The covariate information $\boldsymbol{x}$ is now encoded in the credibilitized information

$$\boldsymbol{c}^{\rm cred} = \boldsymbol{c}^{\rm cred}(\boldsymbol{x}) = Z\,\boldsymbol{c}^{\rm trans} + (1-Z)\,\boldsymbol{c}^{\rm prior} \;\in\; \mathbb{R}^{2b}.$$

- The final step is the decoder that builds predictions from $\boldsymbol{c}^{\rm cred}(\boldsymbol{x})$.

- For this, we select a plain-vanilla FNN $\boldsymbol{z}^{\rm FNN}$ to receive predictions

$$\boldsymbol{x} \;\mapsto\; \mu(\boldsymbol{x}) = \exp\left\{\boldsymbol{z}^{\rm FNN}(\boldsymbol{c}^{\rm cred}(\boldsymbol{x}))\right\} \;>\; 0,$$

the log-link is chosen because we want strictly positive predictions.

- The full architecture is summarized on the next slide.

Summary of the Credibility Transformer architecture

- Our example below has 9 covariates, 5 are continuous (driver's age, power of car, etc.) and 4 are categorical (car brand, province of living, etc.).

- As embedding dimension we selected $b = 5$.

Module	Variable/layer	# Weights
Feature tokenizer (raw input tensor)	$\boldsymbol{x}_{1:9}^{\circ}$	405
Positional encoding	$\boldsymbol{e}_{1:9}^{\mathrm{pos}}$	45
CLS tokens	$\boldsymbol{c}$	10
Time-distributed normalization layer	$\boldsymbol{z}^{\mathrm{norm}}$	20
Credibility Transformer	$\boldsymbol{c}^{\mathrm{cred}}$	1,073
Plain-vanilla FNN decoder	$\boldsymbol{z}^{\mathrm{FNN}}$	193
Total architecture		1,746

- **Section 5: Real Data Example**

French Motor Insurance Claims Frequency

- We use the standard French Motor Third Party Liability (MTPL) claims frequency data set of Dutang et al. (2024). This data set has been used in many studies.

- Data has $n = 678,007$ instances with 5 continuous and 4 categorical covariates.

- We use the same training-validation split as in Wüthrich–Merz (2023) and Brauer (2024) $\implies$ the results are directly comparable.

- We show the results of the ensemble predictors over 20 different SGD runs to reduce randomness and improve SGD training; see Richman–Wüthrich (2020).

- We use the Poisson deviance loss for training and validation.

- For further implementation details, see Richman et al. (2024).

Results of the Credibility Transformer

Architecture	# Param.	Out-of-sample Poisson loss
Poisson null (no covariates)	1	25.445
Poisson GLM3	50	24.102
Ensemble plain-vanilla FNN	792	23.783
Ensemble Feature Tokenizer Transformer (FTT)	27,133	23.759
Ensemble Credibility Transformer: `nadam`	1,746	23.717
Ensemble Credibility Transformer: `NormFormer`	1,746	23.711

- 1st block is taken from Wüthrich–Merz (2023), 2nd one from Brauer (2024).
- `nadam`: Nesterov accelerated version of SGD variant `adam`.
- `NormFormer`: SGD adapted to Transformers; see Shleifer et al. (2021).
- Credibility parameter selected $\alpha = 90\%$.

Features to improve the Credibility Transformer

State-of-the-art LLMs use some of the following features:

- Multi-head attention: The above Credibility Transformer only uses one attention head, multi-head attention uses multiple heads in parallel.

- Deep Credibility Transformer: serial multiple Credibility Transformers.

- Gated Linear Unit (GLU) by Dauphin et al. (2017): down-weighting of less important input components by considering a Hadamard product

$$\boldsymbol{z}^{\mathrm{GLU}}(\boldsymbol{x}) = \boldsymbol{z}^{\mathrm{FNN}_{\mathrm{sigmoid}}}(\boldsymbol{x}) \odot \boldsymbol{z}^{\mathrm{FNN}_{\mathrm{linear}}}(\boldsymbol{x}).$$

- Piecewise linear encoding (PLE) of continuous covariates by Gorishniy et al. (2021): this provides a more informative embedding of continuous covariates compared to our plain-vanilla FNN, as it partly preserves the topology.

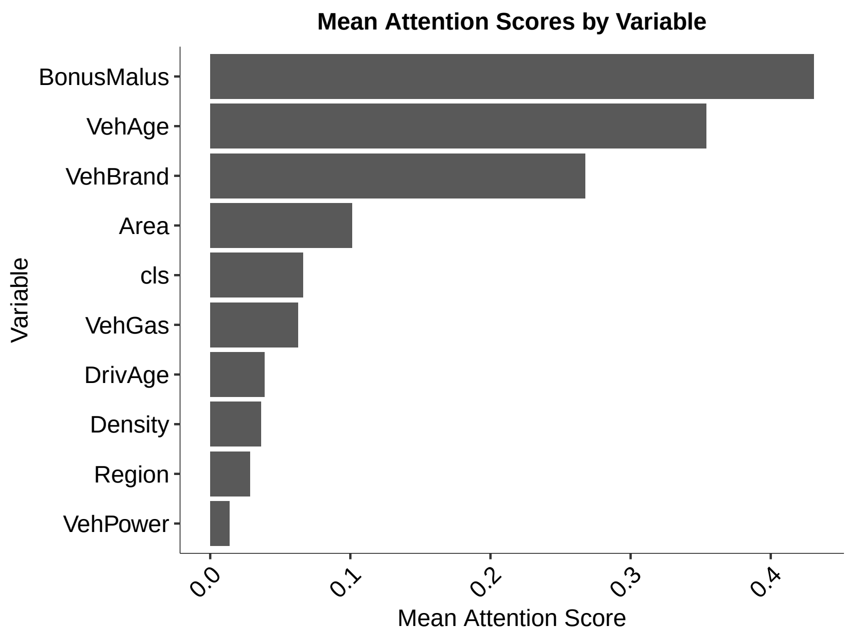
Improved results of the Credibility Transformer

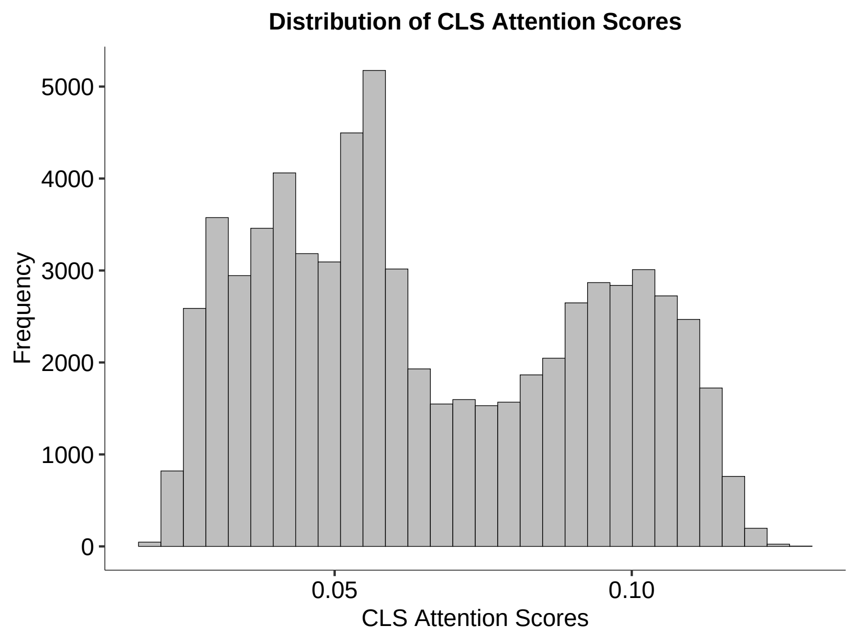
Architecture	# Param.	Out-of-sample Poisson loss
Poisson null (no covariates)	1	25.445
Poisson GLM3	50	24.102
Ensemble plain-vanilla FNN	792	23.783
Ensemble Credibility Transformer (best-performing)	1,746	23.711
Ensemble Credibility Transformer $\alpha = 98\%$	320,000	23.577

- 2 attention heads (multi-head), and 3 transformer layers (deep).
- Embedding dimension $b = 40$.
- This architecture has roughly $320,000$ parameters.
- This architecture is trained on GPUs.

- **Section 6: Explainability**

Attention scores by variables (1/3)

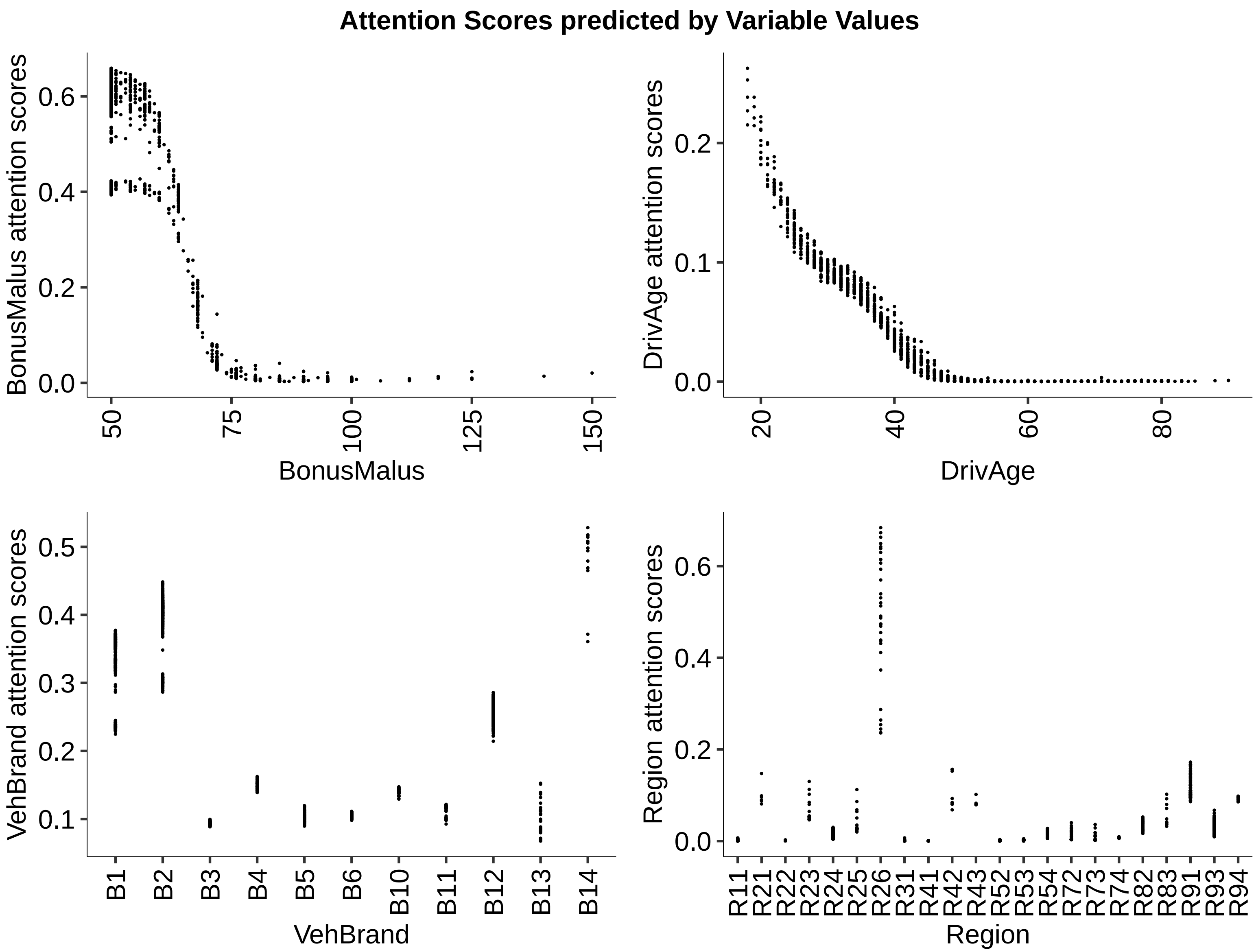




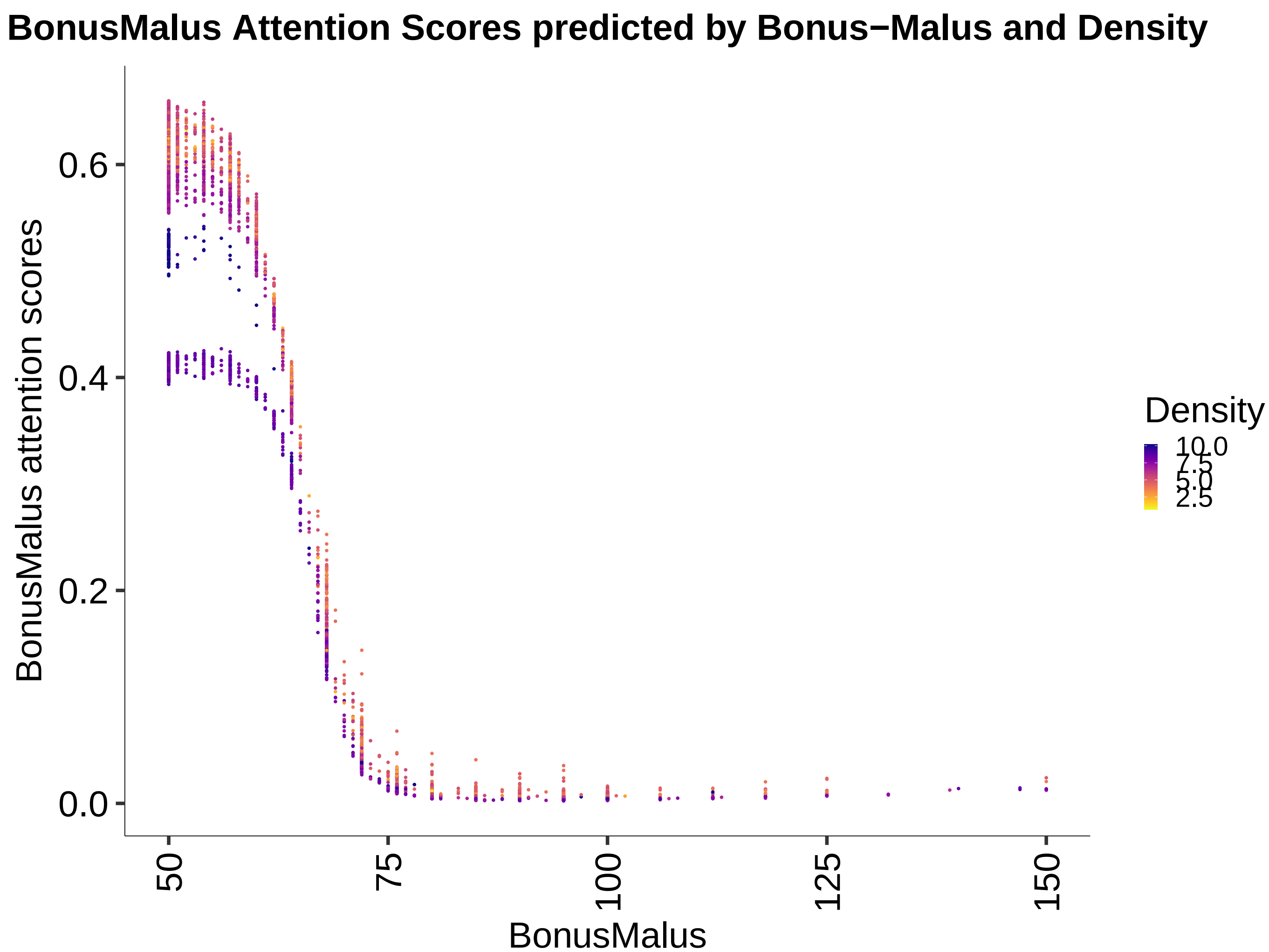
- Recall attention weights/scores in the CLS token

$$a_{T+1,j} = \frac{\exp(\boldsymbol{q}_{T+1}^\top \boldsymbol{k}_j)}{\sum_{s=1}^{T+1} \exp(\boldsymbol{q}_{T+1}^\top \boldsymbol{k}_s)} \in (0,1), \quad \text{for } j = 1, \ldots, T+1.$$

Attention scores by variables (2/3)



Attention scores by variables (3/3)



Summary

- Starting point: Transformers are very successful on time-series data.
- We discussed adaptation of Transformers to tabular data by tokenizing continuous and categorical input information.
- We added a non-informative CLS token to the input. This CLS token is used for a credibility mechanism. This is beneficial in network training (similar to drop-out).
- Besides the obvious Bühlmann credibility interpretation, the Credibility Transformer has a second (more subtle) Bühlmann credibility mechanism.
- Since the decoder only processes the trained credibilitized CLS token (similar to a bottleneck network), we receive some explainability about the inner working of the network architecture.
- The trained Credibility Transformer provides the best out-of-sample results.

References

[1] Brauer, A. (2024). Enhancing actuarial non-life pricing models via Transformers. *European Actuarial Journal*, published online.

[2] Brébisson, de A., Simon, É., Auvolat, A., Vincent, P., Bengio, Y. (2015). Artificial neural networks applied to taxi destination prediction. *arXiv*:1508.00021.

[3] Bühlmann, H. (1967). Experience rating and credibility. *ASTIN Bulletin - The Journal of the IAA* **4/3**, 199-207.

[4] Dauphin, Y.N., Fan, A., Auli, M., Grangier, D. (2017). Language modeling with gated convolutional networks. *Proceedings of the 34th International Conference on Machine Learning* **70**, 933-941.

[5] Devlin, J., Chang, M.-W., Lee, K., Toutanova, K. (2018). BERT: Pre-training of deep bidirectional Transformers for language understanding. *arXiv:*1810.04805.

[6] Dutang, C., Charpentier, A., Gallic, E. (2024). Insurance dataset. *Recherche Data Gouv*. https://doi.org/10.57745/P0KHAG

[7] Gorishniy, Y., Rubachev, I., Babenko, A. (2022). On embeddings for numerical features in tabular deep learning. *Advances in Neural Information Processing Systems* **35**, 24991-25004.

[8] Gorishniy, Y., Rubachev, I., Khrulkov, V., Babenko, A. (2021). Revisiting deep learning models for tabular data. In: Beygelzimer, A., Dauphin, Y., Liang, P., Wortman Vaughan, J. (eds). *Advances in Neural Information Processing Systems*, **34**. Curran Associates, Inc., New York, 18932-18943.

[9] Guo, C., Berkhahn, F. (2016). Entity embeddings of categorical variables. *arXiv*:1604.06737.

[10] Huang, X., Khetan, A., Cvitkovic, M., Karnin, Z. (2020). TabTransformer: Tabular data modeling using contextual embeddings. *arXiv*:2012.06678.

[11] Kuo, K., Richman, R. (2021). Embeddings and attention in predictive modeling. *arXiv*:2104.03545.

[12] Richman, R. (2020). AI in Actuarial Science - A review of recent advances - parts 1 and 2. *Annals of Actuarial Science* **15(2)**, 207-258.

[13] Richman, R., Scognamiglio, S., Wüthrich, M.V. (2024). The credibility transformer. *arXiv*:2409.16653.

[14] Richman, R., Wüthrich, M.V. (2020). Nagging predictors. *Risks* **8(3)**, article 83.

[15] Shleifer, S., Weston, J., Ott, M. (2021). NormFormer: Improved Transformer pretraining with extra normalization. *arXiv*:2110.09456.

[16] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, Ł., Polosukhin, I. (2017). Attention is all you need. *arXiv*:1706.03762v5.

[17] Wüthrich, M.V., Merz, M. (2023). *Statistical Foundations of Actuarial Learning and its Applications*. Springer Actuarial. https://link.springer.com/book/10.1007/978-3-031-12409-9