Relation Aware Self-attention Knowledge Tracing

Description of how correlation between problems embeddings and performance is defined

Exercises are represented using a function f: M -> R^d.
Each word of the exercise is first translated into unique embidding, and then exercize is represented as weighted combination of embeddings of all the words present using Smooth Inverse Frequence:

"a" is a trainable parameter

$$\mathbf{E}_i = \frac{1}{|s_i|} \sum_{w \in s_i} \frac{a}{a + p(w)} f(w),$$

After being described as weighted combination of 50 ("d" in general) most important words (embedded), we need to define correlation between problems.
This correlation is defined as SUM of two components: a collaborative filter, calculating correlation between performance of users for same two problems, and a content filter, calcualting the similarity between two problems with cosine similarity of the embedded words.

		exercise i incorrect	exercise i correct	total
exercise j	incorrect	n_{00}	n_{01}	n_{0*}
	correct	n_{10}	n_{11}	n_{1*}
	total	n_{*0}	n_{*1}	n

$$\phi_{i,j} = \frac{n_{11}n_{00} - n_{01}n_{10}}{\sqrt{n_{1*}n_{0*}n_{*1}n_{*0}}}.$$

$$\mathrm{sim}_{i,j} = \frac{\mathbf{E}_i \mathbf{E}_j}{\|\mathbf{E}_i\|_2 \|\mathbf{E}_j\|_2}$$

$$\mathbf{A}_{i,j} = \begin{cases} \phi_{i,j} + \mathrm{sim}_{i,j}, & \text{if } \mathrm{sim}_{i,j} + \phi_{i,j} > \theta \\ 0, & \text{otherwise.} \end{cases}$$

IDEA: we can try to change this suj with a WEIGHTED SUM or mantain the two matrices separated.
Or we could try using a more complex model for this system, which is similar to recommender systems..

Description of how forgetting factor is modeled

If we consider the current exercise E, for any previous exercise Ej at time Tj, we define deltaJ = Ti -Tj and we modeled forgetting factor as:

$$\mathbf{R}^T = [\exp(-\Delta_1/S_u), \exp(-\Delta_2/S_u), \ldots, \exp(-\Delta_{n-1}/S_u)],$$

where Su is the forgetting factor which can be learnt for every user and represent the tendence to forget topics.

matrix of embeddings

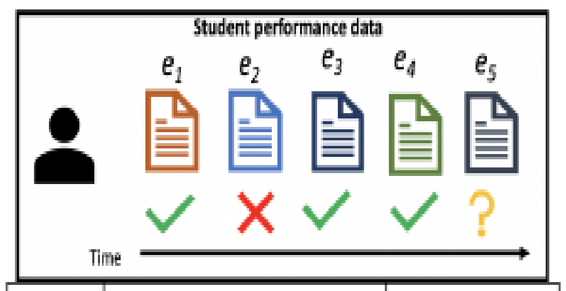
matrix of forgetting factors

$$\hat{\mathbf{x}}_j = [\mathbf{E}_{e_j} \oplus \mathbf{r}_j] + \mathbf{P}_j$$

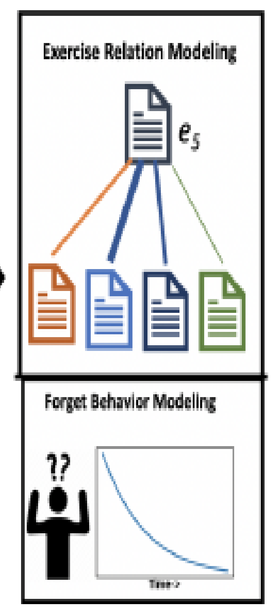
Inputs are represented as using triples of (embedding, correctnes value, time). We need to include a positional encoding too to allow discrimination and this positional encoding is done using P, a positional encoding matrix of dimension l x 2d, where l is maximum allowed sequence lenght and d is number of embedding words used to represent an exercise, so the lenght of Eej.

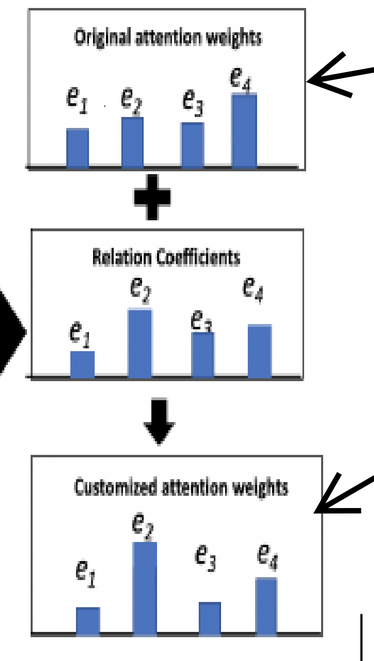
$$\mathbf{R} = \mathrm{softmax}(\mathbf{R}^E + \mathbf{R}^T),$$

We could change softmax with a sort of KNN



Exercise	Exercise Text	KC
e_1	Solve x: $2\|x+6\|+7 = 4x+6+4$	Linear Equations
e_2	Solve for x , $x^2 + 3x - 28 = 0$	Quadratic Equations
e_3	Evaluate $\sqrt{28}$	Square Root
e_4	What are factors of 5?	Division
e_5	Where will $f(x)$ intersect the x axis? $f(x) = (x+9)^2$	Quadratic Equations





Description of how attention weights are managed

Attention mechanism is the core and main difference introduced by RKT.
Coeficients learnt from attentio mechanism are equal to Alpha:

$$\alpha_j = \frac{\exp(e_j)}{\sum_{k=1}^{n-1} \exp(e_k)}, e_j = \frac{\mathbf{E}_{e_n} \mathbf{W}^Q (\hat{\mathbf{x}}_j \mathbf{W}^K)^T}{\sqrt{d}},$$

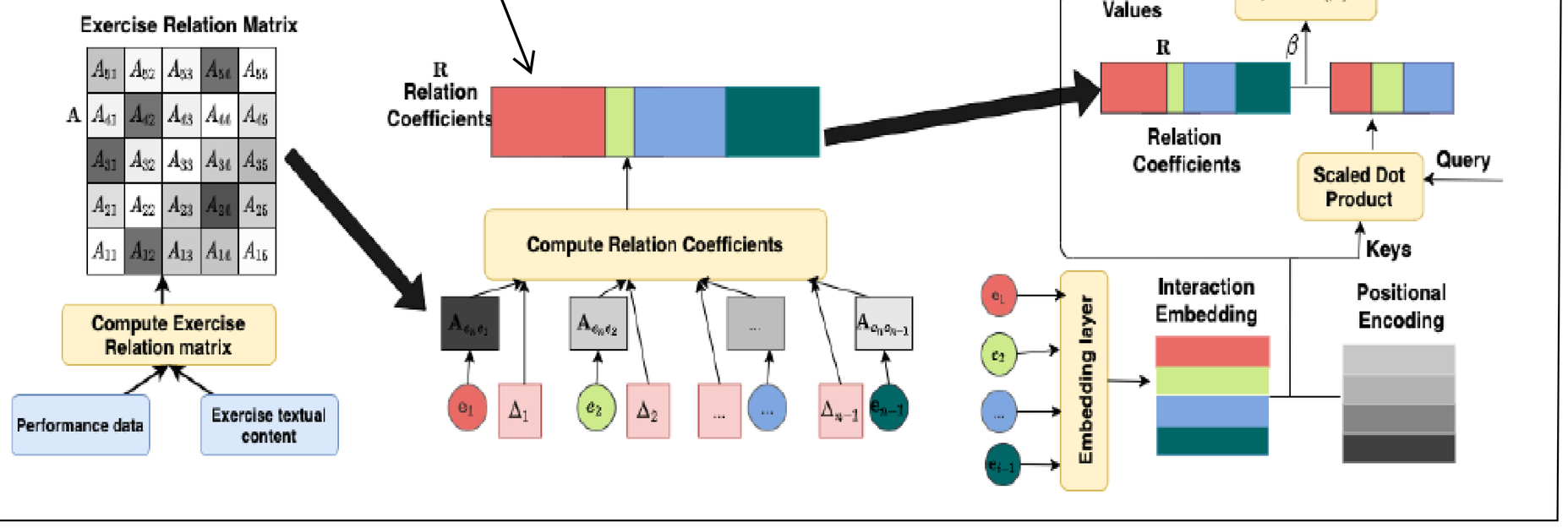
where WQ and WK are respectively query and key vectors used to represent learn attentions on exercises.

Finally we sum weights from correlation and forgetting factor with wwights from attention mechanism with a WEIGHTED sum:

$$\beta_j = \lambda \alpha_j + (1 - \lambda)\mathbf{R}_j,$$

More in deep

Each exercise is seen as combination of other exercise, this coeficients are leart from performance data and textual content



In the end we obtain outputs from weights as:

$$\mathbf{o} = \sum_{j=1}^{n-1} \beta_j \hat{\mathbf{x}}_j \mathbf{W}^V,$$

We add Always residual network and apply Normalization and Dropout

And this outputs are then given as input to a Feed Forward Neural Network, to insert possibility of relations between latent dimensiones, introducing non-linearities:

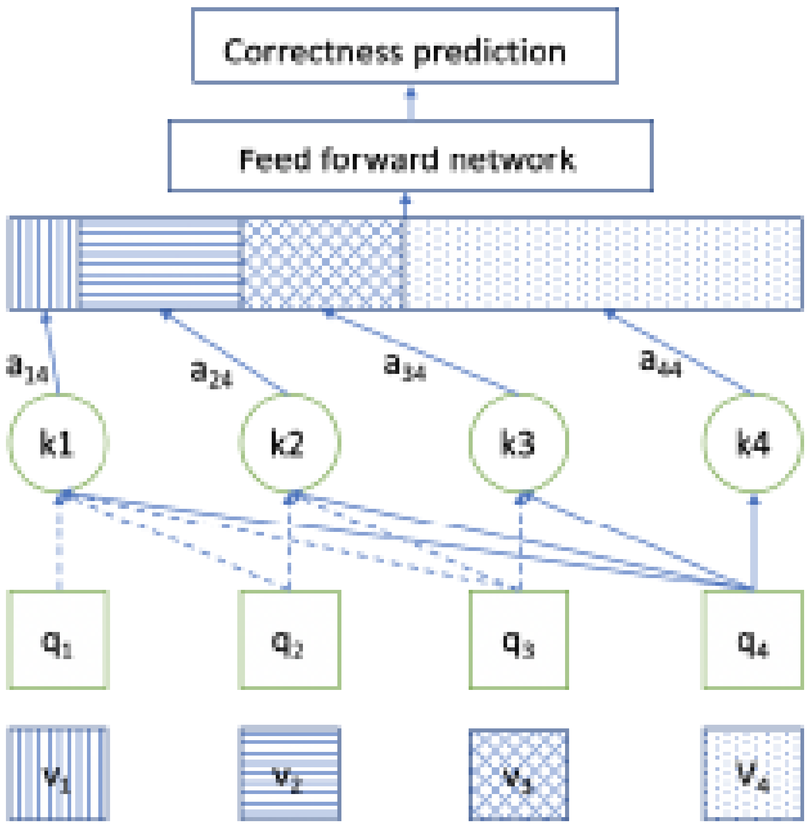
$$\mathbf{F} = \mathrm{ReLU}(\mathbf{o}\mathbf{W}^{(1)} + \mathbf{b}^{(1)})\mathbf{W}^{(2)} + \mathbf{b}^{(2)},$$

where W are weight matrices and b1 and b2 are Bias vectors.
We add Always residual network and apply Normalization and Dropout

In the end we apply a sigmoid function to a linear layer of F matrix obtained to generate probability of answering correctly:

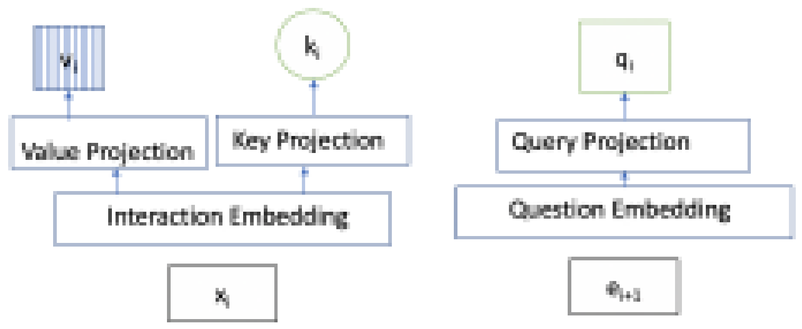
$$p = \sigma(\mathbf{F}\mathbf{W} + \mathbf{b}),$$

SAKT- Self-Attentive model for Knowledge Tracing



In the end we take outputs of Feed Forward network to calcuate prediciton as:

$$p_i = \text{Sigmoid}(\mathbf{F}_i\mathbf{w} + \mathbf{b}),$$



Input Xi is composed of a couple (exercise, result) and we have a sequence of inputs Xi which constitutes the rel input of the system and the new exercise we want to predict the result Ei+1.

The past interactions are embedded using a trained interaction embedding matrix M of dimensions 2E*d , to express each exercise as a weighted combination of some terms (embedded concepts). Exercise to predict instead will be embedded with a trained exercise matrix of dimensions E*d. This will lead to a matrix M and a vector E, we can use them as inputs to a SELF ATTENTIVE layer.

$$\mathbf{Q} = \hat{\mathbf{E}}\mathbf{W}^Q, \mathbf{K} = \hat{\mathbf{M}}\mathbf{W}^K, \mathbf{V} = \hat{\mathbf{M}}\mathbf{W}^V,$$

Where Wq, Wk and Wv are respectively attention matrices for query, key and value and are learnt during training.

In the end we use Q, K and V as input to calculate attention:

$$\text{Attention}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \text{softmax}\left(\frac{\mathbf{Q}\mathbf{K}^T}{\sqrt{d}}\right)\mathbf{V}.$$

We use Multi Head Attention too, meaning that we calculate different Wq, Wk, Wv to allow detection of different templates.

In the end we take the outputs from different attention heads and use as inputs for a Feed Forward Neural Network to allow non linearities and relations between different heads.

$$\mathbf{F} = \text{FFN}(\mathbf{S}) = \text{ReLU}(\mathbf{S}\mathbf{W}^{(1)} + \mathbf{b}^{(1)})\mathbf{W}^{(2)} + \mathbf{b}^{(2)},$$

We use Normalization and add Residual Connections too.

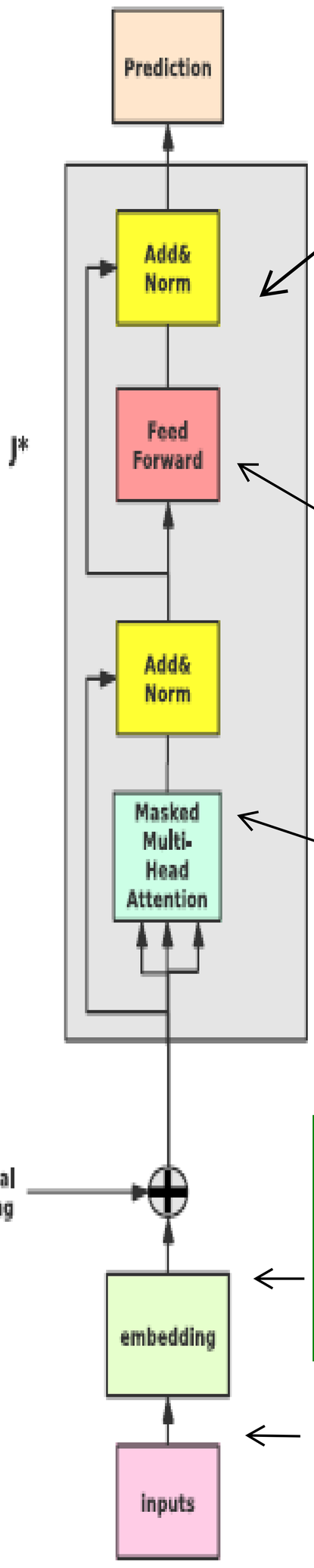
KT-XL: A Knowledge Tracing Model for Predicting Learning Performance Based on Transformer-XL

Prediction Layer

$$\mathbf{T}_i = \frac{\sum_0^{i-1} \mathbf{F}_x}{i-1}$$ ← Normalized

$$y_i = \text{Sigmoid}((\mathbf{T}_i, \mathbf{F}_i)\mathbf{W}_3 + b_3)$$ ← Function for objective result prediction (0 or 1)

$$y_i' = (\mathbf{T}_i, \mathbf{F}_i)\mathbf{W}_3 + b_3$$ ← Function for subjective result prediction



Recurrence Mechanism

Extended state — CONCATENATION

$$\tilde{\mathbf{h}}_{t+1}^{j-1} = \left[\text{SG}(\mathbf{h}_t^{j-1}) \circ \mathbf{h}_{t+1}^{j-1}\right],$$
$$\mathbf{q}'^{j}_{t+1}, \mathbf{k}'^{j}_{t+1}, \mathbf{v}'^{j}_{t+1} = \mathbf{h}_{t+1}^{j-1}\mathbf{W}_q^T, \tilde{\mathbf{h}}_{t+1}^{j-1}\mathbf{W}_k^T, \tilde{\mathbf{h}}_{t+1}^{j-1}\mathbf{W}_v^T,$$
$$\mathbf{h}_{t+1}^{j} = \text{Layer}(\mathbf{q}'^{j}_{t+1}, \mathbf{k}'^{j}_{t+1}, \mathbf{v}'^{j}_{t+1})$$

Hj at time t means the j-th layer of the hidden state at time t, while SG mean Stop-Gradient

Layer is Self attention layer and Feed Forward Layer and we start from initial hidden state equal to the row of B corresponding to exercise at time t

Attention Mechanism applied with input the state for Quey (exercise to predict result) and extended hidden state for Keys (previous exercises).

$$FFN(x) = \max(0, xW_1 + b_1)W_2 + b_2$$

We implement multiple head attention, by training different matrices Wq, Wk, Wv and then we use them to obtain different Qi, Ki, Vi from B and E.

We get $\mathbf{q}_i, \mathbf{k}_i, \mathbf{v}_i$ by the following equations:

$$\mathbf{q}_i = \hat{\mathbf{E}}_{i+1}\mathbf{W}_i^{\mathrm{Q}},\ \mathbf{k}_i = \hat{\mathbf{B}}_i\mathbf{W}_i^{\mathrm{K}},\ \mathbf{v}_i = \hat{\mathbf{B}}_i\mathbf{W}_i^{\mathrm{V}} \quad (2)$$

where $\mathbf{W}^{\mathrm{Q}}, \mathbf{W}^{\mathrm{K}}, \mathbf{W}^{\mathrm{V}} \in \mathbb{R}^{d\times d}$ are the query, key and value projection matrices.

In the end we compute Attention functions as:

$$\text{Attention}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \text{softmax}\left(\frac{\mathbf{Q}\mathbf{K}^\top}{\sqrt{d}}\right)\mathbf{V}$$

We need to MASK keys Kj for queries Qi such that: j > i
This avoids considering future interactions in the prediction.

$$\hat{\mathbf{B}} = \begin{bmatrix} \mathbf{B}_{x_1} + \mathbf{P}_1 \\ \mathbf{B}_{x_2} + \mathbf{P}_2 \\ \dots \\ \mathbf{B}_{x_n} + \mathbf{P}_l \end{bmatrix}, \hat{\mathbf{E}} = \begin{bmatrix} \mathbf{E}_{x_1} \\ \mathbf{E}_{x_2} \\ \dots \\ \mathbf{E}_{x_l} \end{bmatrix}$$

B has dimension Lu * d
E has dimension L * d

1. We split each sequence in smaller sequences of fixed lenght L
2. We train interaction embedding matrix B, where d is Hidden dimension.
3. We train Exercise embedding matrix E to embed current exercise we want to predict score. Bias is not needed, because can be inferred later in attention layer.
4. We create a positional encoding matrix P where i-th row represents relative distance between two positions and we sum it to B.

Inputs are sequences of already done exercises X1, X2,.... Xi, each composed by a couple (exercise ID, result). We conveniently modify classical input Xi in a single value:

$$Ui = Ei * (Amax + 1) - Ai$$

Where Ei is exercise ID and Ai result, while Amax is the maximum score of the exercise
So it can say that Ui can have a number of different values equal to:

$$Lu = (Amax+1) * L$$

where L is number of values of Ei, meaning the number of exercises

Figure 1: The architecture of KT-XL