Executive Summary of the Thesis

Memory Models for Spaced Repetition Systems

Laurea Magistrale in Mathematical Engineering - Ingegneria Matematica

Author: Giacomo Randazzo

Advisor: Prof. Marco D. Santambrogio

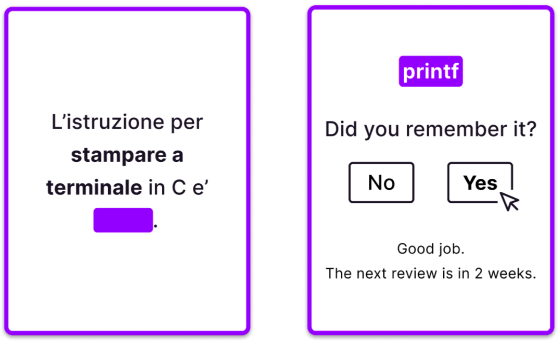
Academic year: 2020-21

1. Introduction

Spaced repetition systems (SRS) allow students to learn more in less time, by reviewing knowledge over multiple intervals of time. SRS schedule reviews to minimize forgetting while being parsimonious with the student's time. To do that effectively, they rely on memory models. Memory models are computational models that predict the probability that the student will recall a piece of knowledge. The effectiveness of SRS has been repeatedly proven over the past three decades, but further improvement would be beneficial for scaling SRS to a larger number of students. Improving the accuracy of memory model predictions is certainly an important step in this direction.
In this work, we present a framework for developing adaptive memory models for SRS (Section 4) and introduce two novel models, DASH[RNN] and R-17 (Sections 5, 6).

2. Overview of spaced repetition systems

Spaced repetition is the practice of reviewing one or more cards over time according to a schedule, in order to prevent forgetting. The practice is based on results from cognitive science, in particular the testing and spacing effects [2]. The *testing effect* suggests actively testing our knowledge rather than re-studying. The *spacing effect* suggests repeatedly reviewing our knowledge at time-spaced intervals. In the present work we only focus on time on a scale of days, weeks, months, and years. In particular, we focus on long-term memory.
Now we describe the fundamental components of SRS.
The basic unit is the *card* (or *item*). A card encodes a piece of knowledge. An example of a card is shown in Figure 1. The *front* of the card represents a test for the student and the *back* of the card represents the corresponding answer.



(a) The front of a card. (b) The back of a card.

Figure 1: A simple card asking a question about the C programming language instruction *printf*.

Each day, the SRS asks the student to review

a few cards according to a schedule. Each review proceeds as follows. The front of the card is presented, the student actively recalls the answer before revealing it, and finally the card is marked as either recalled or forgotten before the next card is presented.
An important component of many SRSs is the memory model. The *memory model* predicts how likely the student is to remember a card, given the review history and the time elapsed since the last review. Thanks to the memory model, every day the SRS can get an estimate of the student's current knowledge state. Based on this estimate, it decides which cards should be introduced or reviewed by the student.

3. Memory models

In this section we formalize the memory model. Let Y be a binary random variable representing a review rating, $Y = 1$ in the case of recall, and $Y = 0$ in the case of forgetting. We want to predict Y given additional information:

- **card and student** Let $\mathcal{C}$ be a set of cards and $\mathcal{S}$ a set of students. Let $C \in \mathcal{C}$ and $S \in \mathcal{S}$ be categorical random variables.
- **review history** We define as *review history* of length K an ordered sequence of reviews $R^{(1)}, \ldots, R^{(K)}$ where $R^{(i)} = (\Delta^{(i)}, Y^{(i)}) \in R^{+} \times \{0, 1\}$ for all i. Here $\Delta^{(i)}$ is a random variable representing the time elapsed between reviews i and $i-1$ for $i > 1$, or the time since the card was introduced to the student for $i = 1$ (time is expressed in days). $Y^{(i)}$ is a binary random variable for the rating of the review.
- **time elapsed** Let $\Delta \in R^{+}$ be a random variable that expresses in days the time elapsed since the last review in the history, or, if the history is empty, since the card was introduced to the student. We observe Δ before making a prediction for the target rating Y, therefore, we include it as a predictor.

For convenience, we denote the random input vector by $X = (C, S, (R^{(1)}, \ldots, R^{(K)}), \Delta)$. We seek a *memory model*, a function $p_\theta(X)$ with parameters θ such that

$$P(Y = 1 | X = x) = p_\theta(x)$$

We call *retrievability* the output probability of recall $p_\theta(x)$.

Our goal is to find an approximation $\hat{p} = p_{\hat{\theta}}$ given a previously collected reviews dataset $\mathcal{D} = \{(r_{cs}^{(1)}, \ldots, r_{cs}^{(k_{cs})})\}_{c \in \mathcal{C}, s \in \mathcal{S}}$ with $r_{cs}^{(i)} = (\delta_{cs}^{(i)}, y_{cs}^{(i)})$. Each card-student pair identifies an independent review history of length k_{cs}: all reviews on the same card c by the student s. Given a loss function $\ell : \{0, 1\} \times \{0, 1\} \to R^{+}$ to penalize prediction errors, let $\ell_{c,s}^{(k)} = \ell(y_{cs}^{(k)}, p_\theta(c, s, (r_{cs}^{(1)}, \ldots, r_{cs}^{(k-1)}), \delta_{cs}^{(k)}))$ be the loss in the k-th review step, $k = 1, \ldots, k_{c,s}$. We compute $\hat{\theta}$ as

$$\hat{\theta} = \operatorname*{argmin}_{\theta} \sum_{c,s} \sum_{k=1}^{k_{cs}} \ell_{c,s}^{(k)} \tag{1}$$

The outer sum is over review histories. The inner sum is over review steps of a single review history; for each step, we consider only information available up to that point in time. In the inner sum, for $k = 1$ the review history is empty.
A memory model is a *probabilistic binary classifier*. We are not only interested in classifying the next review as success or failure, we are also concerned about predicting the probability of the outcome. It is a regression task. As we will see in Section 7 this framing leads to sensible metrics for comparing memory models.

4. State of the art

This section briefly discusses the state of the art. Our goal is to employ the memory model in an SRS that can scale to a large number of students, therefore, we only focus on adaptive memory models. We leave out of the discussion many important memory models that were not designed to account for interactions between students and cards.
Lindsey et al. [4] present the *DASH (Difficulty, Ability and Study History)* framework. Letting $p_\theta(\delta) = p_\theta(c, s, (r^{(1)}, \ldots, r^{(k)}), \delta)$ the framework can be summarized as

$$p_\theta(\delta) = \sigma\left(a_s - d_c + h_\theta^{(k)}(\delta)\right) \tag{2}$$

where a_s and d_c are parameters for, respectively, the ability of the student $s \in \mathcal{S}$ and the difficulty of the card $c \in \mathcal{C}$. $h_\theta^{(k)}(\delta) = h_\theta((r^{(1)}, \ldots, r^{(k)}), \delta)$ isolates the dependence of retrievability on the review history and the time elapsed since the last review. The DASH, DASH[MCM], DASH[ACT-

R] and 1PL-IRT memory models can all be understood as part of this framework.
HLR [5] takes a different approach and assumes that retrievability decreases exponentially as

$$p_\theta(\delta) = 2^{-\frac{\delta}{h_\theta}}$$

where $h_\theta = h_\theta(c, s, (r^{(1)}, \dots, r^{(k)}), \delta)$ here can also depend on $c \in \mathcal{C}$ and $s \in \mathcal{S}$.

5. The R-17 memory model

In this section we present *R-17*, a neural network approximation of *SuperMemo Algorithm SM-17 (SM-17)* by the popular SRS SuperMemo (`https://www.supermemo.com/en`).

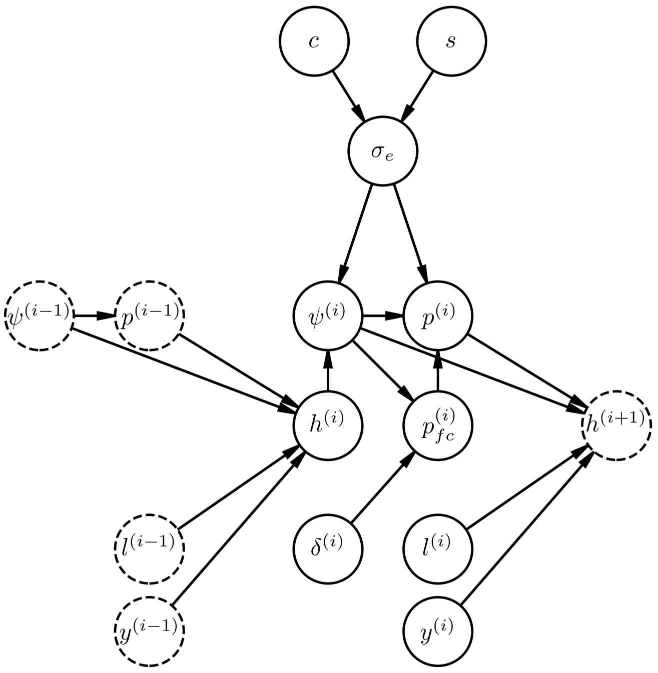


Figure 2: The R-17 memory model computing retrievability $p^{(i)} = p_\theta(c, s, (r^{(1)}, \dots, r^{(i-1)}), \delta^{(i)})$ for review $i \geq 2$. At each review step the inputs are $\delta^{(i)}$, $y^{(i-1)}$ and $l^{(i-1)}$. First we compute a new hidden state $h^{(i)}$ from previous stability and retrievability estimates $\psi^{(i-1)}$ and $p^{(i-1)}$ along with the inputs $y^{(i-1)}$ and $l^{(i-1)}$. We update the stability estimate $\psi^{(i)}$ from $h^{(i)}$ and easiness $\sigma_e(c, s)$. We then use $\psi^{(i)}$ and the remaining input $\delta^{(i)}$ to obtain a first theoretical retrievability estimate $p_{fc}^{(i)}$ through the Wickelgren power law forgetting curve. Finally we correct the estimate to obtain $p^{(i)}$, in the correction we also account for stability $\psi^{(i)}$ and easiness $\sigma_e(c, s)$.

Our aim with R-17 is to get a hint of the performance of SM-17 as a memory model. R-17 is a *recurrent neural networks (RNNs)* [3].
In order to compute the retrievability estimate $p^{(k)} = p_\theta(c, s, (r^{(1)}, \dots, r^{(k-1)}), \delta^{(k)})$ in the k-th review step, we proceed as follows:

- We compute the *ease* $\sigma_e(c, s) = \sigma(b_e + a_s - d_c) \in [0, 1]$ which can be viewed as the output of a single dense layer with logistic (sigmoid) activation function. This choice was inspired by the 1PL-IRT model.
- A hidden state $h^{(i)}$ is updated at each review step i of the review history, we need it to compute stability $\psi^{(i)}$ (we will see below how) and to propagate information between review steps in the RNN. In particular, for each review step $i \geq 2$, $h^{(i)}$ is the output of a neural network H: $h^{(i)} = H(\psi^{(i-1)}, p^{(i-1)}(\delta^{(i-1)}), y^{(i-1)}, l^{(i-1)})$, where $\psi^{(i-1)}$ is the previous estimate of stability, $p^{(i-1)}(\delta^{(i-1)})$ is the previous estimate of retrievability, $y^{(i-1)}$ is the previous rating, and $l^{(i-1)}$ the previous number of lapses (a lapse is a review with rating 0). H is composed of 3 dense layers, the first and second of 8 units each and with the ReLU activation function, the third of 5 units and with the ReLU activation function, therefore $h^{(i)} \in R^5$ for all i. We need to pay special attention to the first hidden state $h^{(1)}$, we cannot compute it with H because we lack the required inputs. We set $h^{(1)} = H_0 \in R^5$ as a vector with trainable components.
- At each review step $i \geq 1$, we calculate $\psi^{(i)} = \Psi(h^{(i)}, \sigma_e(c, s)) \in R$ where Ψ is a dense layer with univariate output and ReLU activation. Notice how, in the review steps $i \geq 2$, a new stability estimate is obtained from the composition of Ψ and H given estimates and data available before the i-th review is rated.
- Given stability $\psi^{(i)}$, we can compute $p_{fc}(\delta; \psi^{(i)}) = (1 + \delta)^{-\psi^{(i)}}$ the theoretical retrievability estimate δ days after review $i - 1$. The function is a Wickelgren power law [4]. For each review step $i \geq 1$, we plug $\delta^{(i)}$ to obtain $p_{fc}^{(i)} = p_{fc}(\delta^{(i)}; \psi^{(i)})$.
- We don't use $p_{fc}^{(i)}$ directly as a prediction for retrievability, but we correct the estimate with the neural network P. We compute

the retrievability prediction before the i-th review step $p^{(i)} = P(p_{fc}^{(i)}, \psi^{(i)}, \sigma_e(c, s)) \in R$. P is a neural network with 3 dense layers, the first and second layers of 8 units each and with the ReLU activation function; the last layer with univariate output and logistic (sigmoid) activation function.

The modules are composed in Figure 2.

There are a total of $282 + |\mathcal{C}| + |\mathcal{S}|$ trainable parameters. We train the RNN end-to-end with gradient descent. The total loss is minimized as in Equation 1, with single review step loss

$$\ell(y, p) = (y - p)^2 + \lambda \|\theta_{\sigma_e}\|_2^2$$

for the target rating y and the corresponding retrievability prediction p. We penalize large ease weights $\theta_{\sigma_e} = \{a_s : s \in \mathcal{S}\} \cup \{d_c : c \in \mathcal{D}\}$, λ is a hyperparameter that controls the penalization. The gradients are computed with the *back-propagation through time (BPTT)* algorithm, for the optimization, we employ the Adam optimizer [3].

6. The DASH[RNN] memory model

The *DASH[RNN]* memory model is an instance of the DASH framework, where h_θ is the output of a RNN (see Equation 2). Our aim with DASH[RNN] is to obtain accurate predictions that outperform the state-of-the-art.

DASH[RNN] is developed as a *recurrent neural networks (RNNs)* [3], which is presented in Figure 3.

If $h^{(k)}$ is the hidden state in the k-th review step, setting $h_\theta = b + \mathbf{W}h^{(k)}$ in Equation 2 we obtain the retrievability prediction

$$p_\theta = \sigma\left(a_s - d_c + b + \mathbf{W}h^{(k)}\right) \qquad (3)$$

where $b \in R^5$ and $\mathbf{W} \in R^{1\times 5}$ are, respectively, trainable bias and weights for the computation p_θ, which is cast as a dense layer with logistic (sigmoid) activation function and single output. That way we can train the model end-to-end as an RNN.

For training the model parameters, we proceed as in Section 5, with single review step loss $\ell(y, p) = (y - p)^2$.

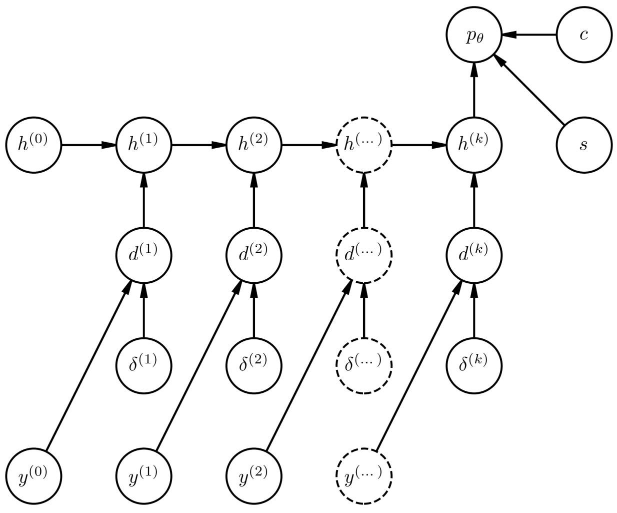


Figure 3: The time-unfolded computational graph of the DASH[RNN] memory model that computes retrievability $p_\theta(c, s, (r^{(1)}, \dots, r^{(k-1)}), \delta)$ for the k-th review in the history. At each review step i the inputs are $\delta^{(i)}$ and $y^{(i-1)}$, $\delta^{(i)}$ is the time elapsed between the i-th review and the $i-1$-th review, the latter with rating $y^{(i-1)}$. The inputs are processed through two dense layers of 12 units each, obtaining the output result $d^{(i)}$. $d^{(i)}$ is fed to a simple RNN layer of 5 hidden units to obtain the current hidden state $h^{(i)}$. Here is where the dependence on the previous review step $i-1$ is accounted for. Finally, at review step k, from $h^{(k)}$ and the additional inputs card c and student s we obtain p_θ as in Equation 3. We set $y^{(0)} = 1$ and $h^{(0)} = 0 \in R^5$.

7. Comparison of memory models

In this section we compare several memory models: DASH[RNN], R-17, DASH, DASH[MCM], DASH[ACT-R], IRT [4] and HLR [5].

The goal of the memory model is to estimate how likely the student is to successfully remember a card at a present or future point in time, and the prediction is then used to schedule reviews in an SRS. For this reason, we decided to compare the models in terms of their predictive power. We employ three metrics: AUC, ICI, and E_{max} [1]. AUC is a measure of discrimination, ICI and E_{max} of calibration of the probabilistic binary classifier. Both ICI and E_{max} are based on the concept of calibration curve. A calibration curve is a regression of the observed binary outcome y_i on the probability p_i predicted by

a probabilistic binary classifier, in our case the memory model. The plot of the calibration curve allows us to graphically examine the calibration of a memory model.
We run the comparison on two different datasets: the Swift dataset and the IEIM dataset.
The Swift dataset was collected from a popular German smart driving-learning app by Swift (`swift.ch`) [6]. After preprocessing, we decided to sample 5 different train-test samples of 10,000 training review histories and 5,000 testing review histories. This translates to approximately 80,000 training reviews and 40,000 testing reviews for each sample. The first train-test sample was used to tune the models. The models were then fitted to each of the 4 remaining training sets and evaluated on the corresponding 4 held-out test sets. We obtained the results reported in Table 1 and Figure 4.

Table 1: Sample mean of AUC, ICI, and E_{max} for different memory models, aggregated across 4 different train-test samples from the Swift dataset. Arrows indicate whether lower (↓) or higher (↑) scores are better, for each metric the model that achieved the best sample mean score is shown in bold.

	AUC↑	ICI↓	E_{max}↓
DASH[RNN]	**0.858**	**0.007**	0.044
R-17	0.841	0.014	**0.037**
DASH	0.829	0.020	0.133
DASH[MCM]	0.829	0.021	0.136
DASH[ACT-R]	0.833	0.012	0.202
IRT	0.768	0.034	0.168
HLR	0.610	0.145	0.509

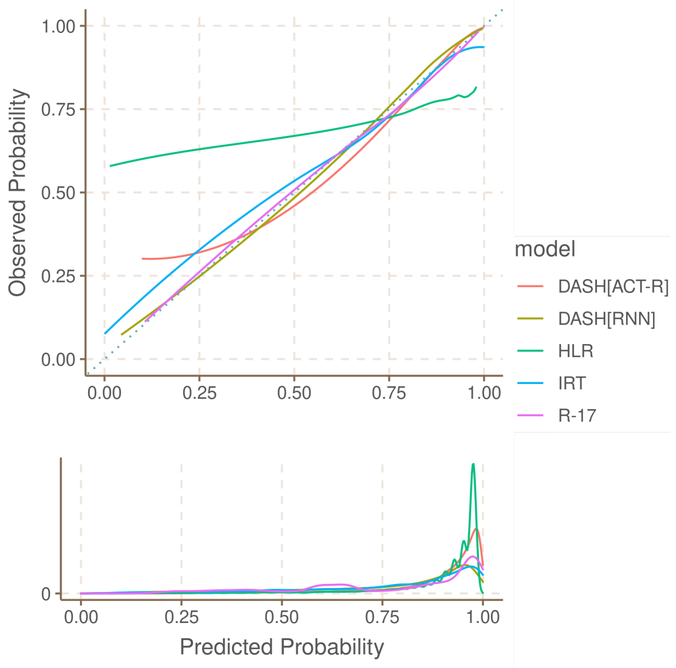


Figure 4: Calibration curves (above) and density of the distribution of predicted probabilities (below) for different memory models, scored on the first of the 4 different train-test samples from the Swift dataset. In a perfectly calibrated memory model the curve would match the identity line. Out of DASH, DASH[MCM], and DASH[ACT-R], we included only the latter variant in the plots, in order not to clutter them with too many lines. The choice is motivated by the better performance compared to the other two variants in Table 1.

The IEIM dataset was collected as part of this work. We provided an SRS web application to students of the 2020/2021 *informatica e elementi di informatica medica (IEIM)* course by Prof. Santambrogio at Politecnico di Milano. After each week of the course we created cards about the covered topics, the students could opt in to review those cards following a spaced repetition practice. We created a total of 146 cards and collected 16,189 reviews from 58 students. After preprocessing, we performed a 10-fold cross-validation repeated 50 times and obtained the results reported in Table 2.

Table 2: Sample mean of AUC, ICI, and E_{max} for different memory models, aggregated across a 10-fold cross-validation repeated 50 times. Arrows indicate whether lower (↓) or higher (↑) scores are better, for each metric the model that achieved the best sample mean score is shown in bold.

	AUC↑	ICI↓	E_{max}↓
DASH[RNN]	**0.842**	0.012	0.267
R-17	0.665	0.011	0.082
DASH	0.770	0.018	0.338
DASH[MCM]	0.771	0.019	0.357
DASH[ACT-R]	0.768	**0.010**	**0.070**
HLR	0.548	0.125	0.600
IRT	0.765	0.018	0.273

Overall DASH[RNN] and R-17, the two novel models introduced in this thesis (Sections 5, 6) fare well against the state of the art. DASH[RNN] outperforms the state of the art on the large Swift dataset, a significant result for developing spaced repetition systems at scale. The results are not as clear in the very small IEIM dataset. The small sample size, combined with the large proportion of correct responses (97% vs 78% for the Swift data set), makes it more difficult to train larger models. R-17 performs comparatively well to the state of the art, this result hints that SM-17 deserves to be studied with more care.

8. Conclusions

The primary contribution of this work is twofold. We introduce two novel adaptive memory models, DASH[RNN] and R-17; in building and comparing the models, we construct a framework for developing memory models specifically for SRS, that can scale and adapt as the system grows. DASH[RNN] outperforms the state of the art from a predictive perspective. R-17 performs comparably well, but the significance of the result lies in hinting at the performance of the proprietary SuperMemo Algorithm SM-17, as an adaptive memory model.

References

[1] Peter C. Austin and Ewout W. Steyerberg. The Integrated Calibration Index (ICI) and related metrics for quantifying the calibration of logistic regression models. *Statistics in Medicine*, 38(21):4051–4065, September 2019.

[2] Nicholas J. Cepeda, Harold Pashler, Edward Vul, John T. Wixted, and Doug Rohrer. Distributed practice in verbal recall tasks: A review and quantitative synthesis. *Psychological Bulletin*, 132(3):354–380, May 2006.

[3] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning.* MIT Press, 2016.

[4] Michael C. Mozer and Robert V. Lindsey. Predicting and improving memory retention: Psychological theory matters in the big data era. In *Big Data in Cognitive Science*, Frontiers of Cognitive Psychology, pages 34–64. Routledge/Taylor & Francis Group, New York, NY, US, 2017.

[5] Burr Settles and Brendan Meeder. A trainable spaced repetition model for language learning. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1848–1858, 2016.

[6] Utkarsh Upadhyay, Graham Lancashire, Christoph Moser, and Manuel Gomez-Rodriguez. Large-scale randomized experiments reveals that machine learning-based instruction helps people memorize more effectively. *npj Science of Learning*, 6(1):1–3, September 2021.