POLITECNICO
MILANO 1863

SCUOLA DI INGEGNERIA INDUSTRIALE
E DELL'INFORMAZIONE

Memory Models for Spaced Repetition Systems

Tesi di Laurea Magistrale in
Mathematical Engineering - Ingegneria Matematica

Author: **Giacomo Randazzo**

Student ID: 944611
Advisor: Prof. Marco D. Santambrogio
Academic Year: 2020-21

Abstract

Spaced repetition systems (SRS) allow students to learn more in less time by reviewing cards. A card is a test, usually a question-answer pair, that is reviewed over multiple intervals of time. In each review, the card is marked as remembered or forgotten. SRS schedule reviews to minimize forgetting while being parsimonious with the student's time. In order to do that effectively, they rely on memory models. Memory models are computational models that predict the probability that the student will recall each card. 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 and introduce two novel models, DASH[RNN] and R-17. We compare several adaptive memory models from a predictive perspective on two real-world spaced repetition datasets, one of which has been collected as part of this work. DASH[RNN] outperforms the state of the art, R-17 performs comparably well. The latter result hints at the performance of the proprietary SuperMemo Algorithm SM-17 as an adaptive memory model, of which R-17 is a neural network approximation.

Keywords: spaced repetition, human learning, adaptive spacing, recurrent neural networks

Abstract in lingua italiana

Gli Spaced Repetition System (SRS) permettono agli studenti di imparare di più in meno tempo rivedendo delle carte. Una carta è un test, di solito una coppia domanda-risposta, che viene rivista a più intervalli di tempo. Ad ogni revisione la scheda è segnata come ricordata o dimenticata. Gli SRS programmano le revisioni in modo da minimizzare le dimenticanze, cercando allo stesso tempo di essere parsimoniosi con il tempo dello studente. Per fare questo in modo efficace, utilizzano dei memory model. I memory model sono modelli computazionali che prevedono la probabilità che lo studente si ricordi ciascuna carta. L'efficacia degli SRS è stata ripetutamente dimostrata negli ultimi tre decenni, ma migliorarla ulteriormente sarebbe utile per avvicinare un numero maggiore di studenti. Migliorare l'accuratezza delle previsioni dei memory model è certamente un passo importante in questa direzione. In questo lavoro presentiamo una struttura per lo sviluppo di memory model adattivi per SRS e introduciamo due nuovi modelli, DASH[RNN] e R-17. Confrontiamo diversi memory model adattivi da un punto di vista predittivo su due dataset di spaced repetition, uno dei quali è stato ottenuto in vista dell'eleborazione di questo lavoro. DASH[RNN] ottiene prestazioni migliori dello stato dell'arte. R-17 ottiene prestazioni comparabili allo stato dell'arte, quest'ultimo risultato suggerisce quali potrebbero essere le prestazioni dell'algoritmo proprietario SuperMemo Algorithm SM-17 come memory model adattivo, di cui R-17 è un'approssimazione sotto forma di rete neurale.

Parole chiave: spaced repetition, human learning, adaptive spacing, recurrent neural networks

Contents

Introduction

Spaced repetition systems allow students to effortlessly review the knowledge they care about over time. They allow students to learn more in less time. Their effectiveness has been repeatedly proven over the past three decades and many implementations have become popular, serving millions of students [46].

In spaced repetition systems, knowledge is encoded in cards: tests that elicit a binary recall-forgotten response, often in the form of question-answer pairs. Cards are actively reviewed, students try to answer each test before checking the corresponding answer and letting the system know whether they recalled it or not. Every day the system suggests to review some cards according to a schedule. Ideally, students would always remember the knowledge they care about, but forgetting happens. The tension at the hearth of a spaced repetition system lies in adjusting the schedule to minimize forgetting, while being parsimonious with the student's time.

To solve that tension, it is fundamental to estimate how likely the student is to remember or forget each card. This role is played by memory models. As formalized in this thesis, memory models are probabilistic binary classifiers that predict the student's knowledge state for each card. The focus of this thesis is on developing adaptive memory models that adjust their predictions for each student and each card by learning from data collected from spaced repetition systems deployed in the real world, and that therefore can scale with the systems in a positively reinforcing feedback loop.

The primary contribution of this thesis is twofold: we present a framework for developing adaptive memory models specifically for spaced repetition systems and introduce two novel adaptive memory models, DASH[RNN] and R-17. We compare the models to the state of the art from a predictive perspective. The DASH[RNN] model outperforms the state of the art, R-17 performs comparably well. The significance of the latter result is in filling the gap between industry and academia, R-17 is a neural network approximation of SuperMemo Algorithm SM-17 by the popular spaced repetition system SuperMemo, for which a description detailed enough to replicate it is not publicly available.

In Chapter 1 we introduce spaced repetition systems, detailing their fundamental components, and discussing important results from psychology on the nature of memory.

In Chapter 2 we formalize adaptive memory models for spaced repetition systems, discuss their inherent limitations, and review the state of the art.

In Chapter 3 we introduce two novel memory models: DASH[RNN] and R-17. The goal of the former is to obtain accurate predictions, outperforming the state of the art. The goal of the latter is to get a hint about the predictive performance of SuperMemo Algorithm SM-17.

In Chapter 4 we compare the newly introduced memory models to the state of the art from a predictive perspective. In doing so, we will introduce and justify sensible metrics for evaluating and comparing adaptive memory models. We employ two different datasets in the comparison; one of them has been collected as part of this thesis: we provided a spaced repetition system, along with pre-written cards to students of the 2020/2021 IEIM course at Politecnico di Milano by Prof. Santambrogio.

1 | Introduction to spaced repetition systems

The central problem is that of forgetting. As people, we forget what we want to remember.

It is common experience that memories naturally decay, we forget over time. Informally, if we learn about a topic made up of 10 concepts, the next day we might remember 7 of those and, after a week, just a couple of the most important ones. Imagine remembering 8-9 of the 10 concepts after a week. The first-order effect is the memorization itself: we can remember more. But there is also an important second-order effect that should not be underestimated: knowledge compounds, if we have more concepts readily available to us, we have a much easier time learning new, related, material. We pay less in terms of time to learn it and the pool of concepts we can learn expands. This is one way in which memorization is fundamental to learning.

As humans, we have been tackling the problem of forgetting for millennia. We came up with many solutions to this problem. A remarkable example is the *memory palace technique* described by Cicero in ancient Rome [11]. Only recently, psychology has identified key insights into the dynamics of memory: *testing effect* and *spacing effect*.

The testing effect suggests that actively testing our knowledge has a stronger effect on memory consolidation compared to restudying the topic of interest [28, 34].

The spacing effect suggests that cramming should be avoided, information is better retained if repeatedly reviewed over spaced intervals of time [6, 7].

Spaced repetition is the practice of reviewing one or more items over time according to a schedule. In particular, it can be understood as a combination of the testing effect (reviews are usually tests) and the spacing effect (reviews are spaced over time). In the present thesis, we only focus on time in a scale of days, weeks, months, and years. In particular, we focus on long-term memory. The effectiveness of spaced repetition has also been studied in shorter time spans [6].

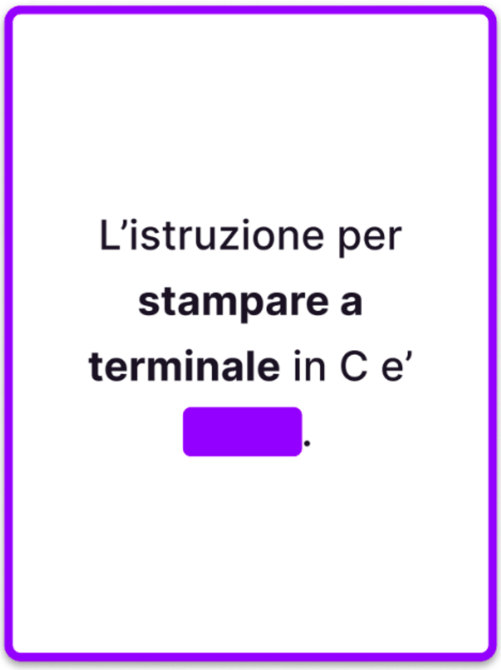
Spaced repetition systems (SRS) allow us to practice spaced repetition and are generally implemented as software applications. They allow students to aggregate material of interest and review it according to an optimized schedule. It would be exceedingly hard to keep an index of the large number of concepts and ideas that we want to remember, managing the reviews by ourselves would be daunting. We rely on an SRS.

The goal of this chapter is to introduce some concepts that we believe are important for understanding and developing spaced repetition systems.

1.1. Spaced repetition systems

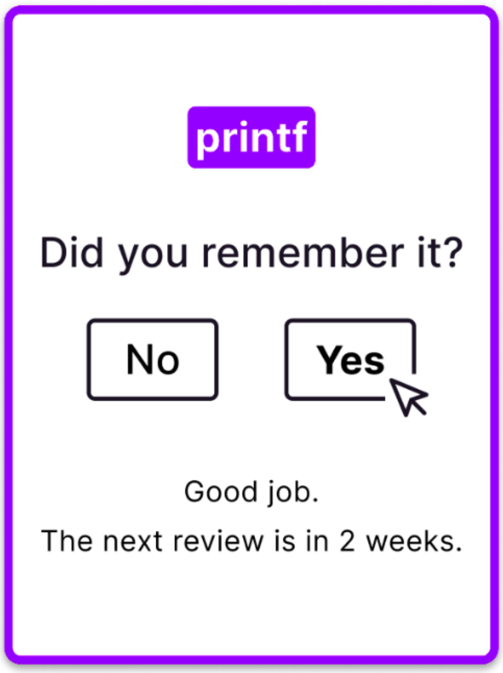
We now describe the components shared by many spaced repetition systems. In particular, we refer to the web-app employed in the IEIM experiment described in Section 4.4.

The basic unit is the *card* (we also use the more general term *item*), which represents a piece of knowledge. A card is a test, usually very precise and narrow. An instance of a card is depicted in Figure 1.1. The *front* of the card represents a prompt for the user and the *back* of the card represents the corresponding answer.



L'istruzione per **stampare a terminale** in C e' .

(a) The front of a card.



Did you remember it?

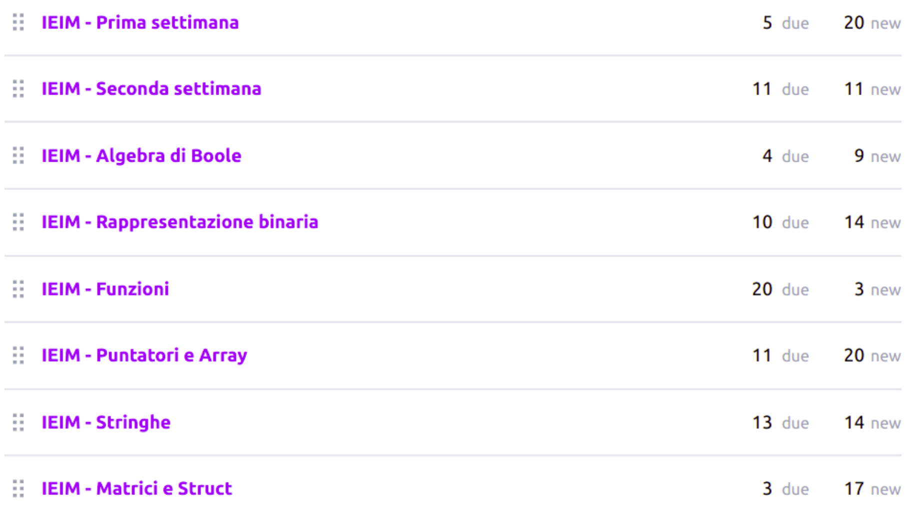
Good job.
The next review is in 2 weeks.

(b) The back of a card.

Figure 1.1: A simple card asking a question about the C programming language instruction *printf*.

Figure 1.2: A deck is a group of cards, usually about the same topic.

Cards are grouped in *decks*, Figure 1.2. The cards in the same deck should be about the same topic. For example, if a university course is divided into three modules, one could have three decks, one per module. An alternative is to have one deck for each week of the course. The former strategy is the one employed in Section 4.4.



IEIM - Prima settimana	5 due	20 new
IEIM - Seconda settimana	11 due	11 new
IEIM - Algebra di Boole	4 due	9 new
IEIM - Rappresentazione binaria	10 due	14 new
IEIM - Funzioni	20 due	3 new
IEIM - Puntatori e Array	11 due	20 new
IEIM - Stringhe	13 due	14 new
IEIM - Matrici e Struct	3 due	17 new

Figure 1.3: A list of decks in the web-app dashboard of the IEIM experiment described in Section 4.4.

When students log into their account, they are presented with a list of decks on a dashboard, as in Figure 1.3. For each deck, they review a sequence of cards.

The review of each card proceeds as follows. First, the review is presented as a test, usually a question or fill-in-the-blanks, Figure 1.1a. The student actively recalls the answer, which is then revealed, Figure 1.1b. The student indicates to the system whether the recall was successful or not through a binary rating, Figure 1.1b. The test is usually self-rated. The next card is then presented, if any is available.

On the dashboard, along each deck, two counters are shown: the number of *due cards* and the number of *new cards*. See Figure 1.3. A card is due if the student studied or reviewed it before and should review it again today. The new cards are those the student has never seen before; they are presented for the first time today. Every day the student is asked to review a set of cards, therefore the due and new counts change every day. It is the responsibility of a *reviews scheduler* to decide which cards are new and due on any day.

1.1.1. Review schedulers and memory models

In this section, we first introduce the idea of a review scheduler and then the idea of a memory model, highlighting the differences between the two.

Students review cards over time, in order to improve the overall retention of the underlying material. After the material has been imported, the system prompts the student to review some of the cards. It is the task of a *review scheduler* to decide which cards the student should review each and every day. It solves the tension between the introduction of new material and the review of old material. It decides which cards to study and how often to maximize both learning and retention. The review scheduler is a core component of any spaced repetition system.

Many students are deeply concerned with the management of their study time. In the amount allocated to spaced repetition, they completely trust the system in managing it, in scheduling the reviews of the material in an optimal way. That is why the review scheduler is of crucial importance. The goal of the review scheduler is to help the student learn more in less time [14]. In building a spaced repetition system, we don't want it to overwhelm the student with reviews. If you review something every day, you will remember it, of course, but you want to make space for free time and other material. The goal of SRS should be to get the problem memorization out of the way for the students.

Note that review schedulers should not only be reliable, but also be flexible. The problem of scheduling reviews requires considerations about the *UX (User Experience)* of the system. Here are some example scenarios for which a scheduler might want to accommodate.

- We can't expect students to study exactly at the scheduled times, as argued in [41]. We schedule reviews every day, but, for example, the student might access the system only once a week. Students might also take breaks from studying with the systems, for instance on vacation.
- Student might want to limit the amount of time they spend reviewing the cards on any given day.
- We don't want to overwhelm the student. It is common experience and complain of existing SRS like Anki (`https://apps.ankiweb.net/`) to come back to the system after a few weeks of inactivity to find a lot of due cards, all the cards that were due in that period. This might discourage students and, in the worst case, they could leave the system. Solutions include balancing the load of reviews, avoiding huge spikes in the number of reviews [41].
- We want to help the student be consistent in reviewing the material, and we might want to embed habit formation best practices in the scheduler. Assuming that we have some estimate of the difficulty for each card, we might also take that into account.
- If a student uses the SRS to study for an exam, we might have the option to prioritize cards about the exam material.

There are different possible review schedulers with varying degrees of effectiveness for different purposes. Fixed uniform, expanding and contracting schedules have been studied traditionally; more recently, the focus has shifted to adaptive schedulers [14]. In several of those, a fundamental role is played by the *memory model*. The memory model predicts how likely the student is to remember a card, given the review history and the elapsed time since the last review. Memory models are the central topic of this thesis, we discuss them in more detail in Chapter 2.

Given a memory model, every day the review scheduler can access the review history of the student and use a memory model to get an estimate of his current knowledge state. Based on the estimated knowledge state, the scheduler can decide which cards should be due or new. To do that, we can either resort to heuristics, for instance [9, 46], or we can design an optimization problem over all possible future review histories with some predefined target. A common target is to achieve the largest possible retention across all cards in a finite and fixed amount of time [19]. [31] optimizes the rate of study and of introduction of new material. [32, 37, 40, 52] apply Deep Reinforcement Learning to the problem. [39, 41] resort to the theory of stochastic optimal control of SDEs with jumps. The optimization problem is often based on predictions from a memory model, the accuracy of the latter is of fundamental importance. The degree to which the estimated schedules can be improved is constrained by the performance of the memory model.

Finally, we would like to spend a few more words on the distinction between review scheduler and memory model. A review scheduler employing a memory model queries the latter, asking for present or future probability of recalling each of the student's cards. This information is then used to schedule reviews. The memory model allows the scheduler, for example, to prioritize cards that are more likely to be forgotten. In thinking and designing spaced repetition systems, the two are often reasoned about as one and the same. We want to show a couple of examples where distinguishing between them can be very useful. Imagine the following scenario, a student studies along a university course with spaced repetition cards. A deck of cards is available for each week of the course, just as in our IEIM experiment in Section 4.4. The exam is coming up; it would be nice for the student to assess the state of his knowledge of the course. We can help by showing how likely the student is to remember any card about the course. We could even aggregate the data at the level of decks or even output a summary for the entire course. To compute these data, all we need is a memory model. The scheduler does not play any role in this task. What we have just shown is a plausible real-world situation in which a memory model is useful independently of the review scheduler. Likewise, we now show a similar situation where the opposite is true: the scheduler is useful independently of the memory model. Thanks to the aforementioned feature, the student realizes that his knowledge of the material from the second week of the course is weak. He would like to quickly go over the material to increase his confidence of passing the test. Many popular spaced repetition systems, including SuperMemo (`https://www.supermemo.com/`) and Anki, support a cramming feature: they allow students to review immediately all the cards in one or more decks, independently of when they are scheduled. Going back to our topic, the implementation of the cramming feature by the review scheduler does not need a memory model (although it might be employed to sort the cards).

1.2. Some results from the study of memory

In this section, after an introduction to the importance of memorization as enabled by a spaced repetition system, we turn our attention to some results of the psychological literature of the past two centuries. Results that are foundational for, and in some cases have developed along spaced repetition systems. In Section 1.2.1 we introduce a description of memory strength based on two components: retrievability and stability. In Section 1.2.2 we introduce the notion of forgetting curve.

As people, when we think of memorization, we might imagine repeating poetry or geography labels ad nauseam in order to recite them back when requested, without a meaningful personal connection to the material. This is commonly referred as *rote memorization*. It is an expensive and time-consuming task.

The reputation of memorization seems to be declining. Awareness of its role, power, and goals seems to be obscured by the rote chanting of names and labels. Memory palaces are being filled with digits of π. The acceleration of access to information might be a reason for this cultural image of memorization: it might seem pointless to memorize information that can be easily found online, but the purpose of memorization is not just to quickly access information, it plays a fundamental role in learning and understanding. A firmer hold on our attention and the process of forgetting leads to more robust internal representations, to more efficient, wise, and enduring learning. Enhanced access to our internal library of ideas, notions, vocabulary, and experiences leads to more ideas, connections, and insight, to finer sensibility and wit. Cicero described the memory palace technique in his *De Oratore*, where it is described as a useful tool for orators. At the same time, they need to learn and access vast amounts of knowledge and be able to recite the arguments and counterarguments they developed [11].

The current thesis is motivated by a view of memory as underlying any form of learning. Not only targeted to theory but also to practice, not only for explicit and ordered knowledge but also for tacit and embodied knowledge. Memory is much more than rote memorization.

A quick note on the distinction between spaced repetition and memory palaces or other memory techniques. The latter are techniques for representing knowledge. These representations follow the dynamics of memory as any other kind of knowledge. Therefore, a memory palace could be used to encode information that can be reviewed according to a spaced repetition schedule to greatly increase the chance of recall of the underlying knowledge. The trade-off is the risk of learning useless representations, a mathematical formula in a memory palace is remembered by its looks and by how it is written. We might miss out on improving our understanding of the formula over time, which is more likely to happen if we write several cards asking, for instance, about the significance and importance of the formula, it's components, the reason the components are put together in a particular way or how it relates with other formulas and concepts.

1.2.1. The two components of long term memory

As people, memory allows us to carry information in the future. Human memory is a complex phenomenon, with implications and interrelations with other cognitive tasks, such as creativity [15]. It is fundamental and ubiquitous for our functioning.

Human memory storage capacity is pretty much unlimited, what is limited is our ability to access the stored information [3]. We perceive lost access as forgetting. Forgetting is closely related to adaptation to the environment. We forget so that we are able to quickly access what the environment requires from us, what is salient, without having to search a vast library of information. In addition, forgetting serves generalization, the reason we forget might be that it is beneficial to make better decisions [33].

Both Wozniak *et al.* [49–51] and Bjork [3] describe two different components of memory strength; in particular, they both argue against single measures of memory strength. Wozniak *et al.* [50] write:

> We may say that "after a review, memory is strong". On the other hand, after years of keeping a piece of information in memory, we say that "memory of the fact is strong" even though, at times, we may hesitate while trying to recall the fact. In the two said cases, we speak of two different phenomena that are both labeled as memory strength. Disentangling the two is vital if we are to avoid contradictory findings in memory research.

The first phenomenon is captured by *retrievability*: how easily one can access the item in memory at a certain point in time.
The second phenomenon is captured by *stability*: how long a memory trace can last in memory.

We have followed the nomenclature and descriptions of Wozniak *et al.* [49–51]. The respective concepts in Bjork [3] are *retrieval strength* and *storage strength*, there are nuanced differences between the definitions, some of which we review at the end of the section.

Ideally, for an item we care to memorize, we would like both retrievability and stability to be large: we want the item to be easily accessible for a long time. How can we achieve this goal? A possible answer is spaced repetition systems, and we describe the reason in the following paragraphs.

Both theories agree on the effect of retrieval or re-study of the item. Retrievability is brought back to a large level, but stability is also increased. What this means is that after we review an item we have a large change of being able to recall it right away, but that is not the only benefit, the item is accessible for longer. An explanation for this phenomenon is that since we have found the item in the environment again, there is a greater chance that it is a feature of the environment, that it is important. Therefore, we adapt and remember the item for a longer time.

The testing effect and the spacing effect can be expressed in terms of retrievability and stability. Retrieval has a larger effect on stability for active retrieval compared to restudy (testing effect), this is linked to the idea of *desirable difficulties* [2]. If retrievability is high, stability is less affected by repetition (spacing effect). Both effects can be understood as helping us better adapt to the environment.

However large the stability, if an item is not reviewed periodically, access to it will be lost. We will forget. *Spaced repetition* is a tool that leverages the spacing and testing effect that we can use to review and memorize the items we, as people, specially care about, in as little time as possible. We are able to somehow control stability to make the item accessible as long as we can; in particular, we can review the item according to a certain schedule. In an illustrative way, we wait some time as to let retrievability drop, we then review the item in order to gain in stability and reset retrievability to a large level. Then we can wait a longer amount of time before retrievability drops to the previous level, thanks to the increased stability. We can repeat this process as long as we need. We need a decrease in retrievability because of the spacing effect. This is the basic mechanism that underlies spaced repetition. Quoting Michael Nielsen in [25] on the popular Anki spaced repetition system: "Anki makes memory a choice". If we anticipate that we might forget something we care about, we can decide to increase the likelihood of remembering it by employing spaced repetition.

Another important variable to consider to describe the dynamics of forgetting is *item complexity*, it is important when we compare different items. An item can be simple (e.g. first three digits of π) or composed of many interlinked but not reducible items (e.g. Maxwell equations). It can be well connected or not. An idea that repeatedly comes up to surprise us in terms of explaining other unrelated ideas is well connected, a formula memorized for a university exam and then forgotten is not well connected. Emotional connection to the item likely also plays a role. Other factors could also be taken into account. The complexity of a particular item could also change over time for a student, for instance, a composed item might be understood in a novel way. Making it much easier to remember. In addition, different people have different predispositions to memory. One might have a harder time remembering a symphony compared to a painting, or vice versa. Age might also be taken into account

Memory behaves differently for items of different complexity. Whether the difference is in kind or just in degree, whether some items have completely different dynamics compared to simpler ones, is a question that, to our knowledge, has not been answered. Solving the problem is not straightforward. The literature on spaced repetition has often focused on simple items, such as vocabulary in language learning. It is not clear whether the above-stated results hold in the same way for items of different complexity. Moreover, given the subjective nature of item complexity, it is not obvious how to estimate it from data coming, for instance, from a spaced repetition system. It certainly is an interesting possible avenue for future studies.

Finally, we want to consider the ways in which the theories of Wozniak *et al.* [49–51] and Bjork [3] disagree, which are nevertheless outnumbered by those in which they agree. Retrievability and retrieval strength are defined in the same way. Storage strength is defined more broadly than stability as the degree to which an item has been learned. The view presented by Bjork [3] is broader and more nuanced, especially in terms of the retrievability dynamics. The latter not only decreases with time (with disuse) as in Wozniak [51], but also takes into account competition between different items. In particular, retrieval capacity is limited, unlike storage capacity; therefore, access to some items might be reduced by retrieval of others. An interesting direction for future work could certainly be to develop a memory model that accounts for not only time, but also other effects on retrievability, such as interference between items. It is worth noting that the two theories apparently arose independently from two different contexts. [3] was developed in academia while [49–51] mutually developed along with the SuperMemo spaced repetition system.

1.2.2. The forgetting curve

We now turn our attention to the *forgetting curve*, it describes how the probability of remembering an item changes in time, assuming the item is not retrieved or restudied. The definition has been chosen among many possible alternatives; we have partially followed Wozniak *et al.* [50]. We resume the discussion about forgetting curve varieties at the end of the section. Many models we study in Chapter 2,3 are based on this concept. Some memory models include forgetting curves; in particular, they output a forgetting curve for each different combination of card, student, and review history.

Given an item studied at the initial time $t = 0$, define retrievability $p(t)$ as the probability of recalling the item at time $t > 0$. We usually expect $p(t)$ to be regular, monotonically decreasing (i.e. the item is subject to forgetting) and starting at $p(0) = 1$, we cannot state those as strict assumptions because in Chapter 2 we will see instances of forgetting curves that do not satisfy them. Define stability s such that $s = \min\{t \geq 0 : p(t) \leq 0.9\}$, it is the interval of time such that retrievability drops to 90% [50]. We consider the day as the unit of time. Note that the two definitions given here are consistent with the descriptions of retrievability and stability of Section 1.2.1.

The function $t \mapsto p(t), t > 0$ is the forgetting curve. The name can be confusing, as $1 - p(t)$ is the probability of having forgotten the item at time $t > 0$.

We can say a little more about the forgetting curve, in particular about its shape. Wozniak *et al.* [50] assume a negatively exponential shape for the forgetting curve. We depart from their report to compare their *exponential forgetting curve* with the *Wickelgren power law* [45], both of which are shown in Figure 1.4.

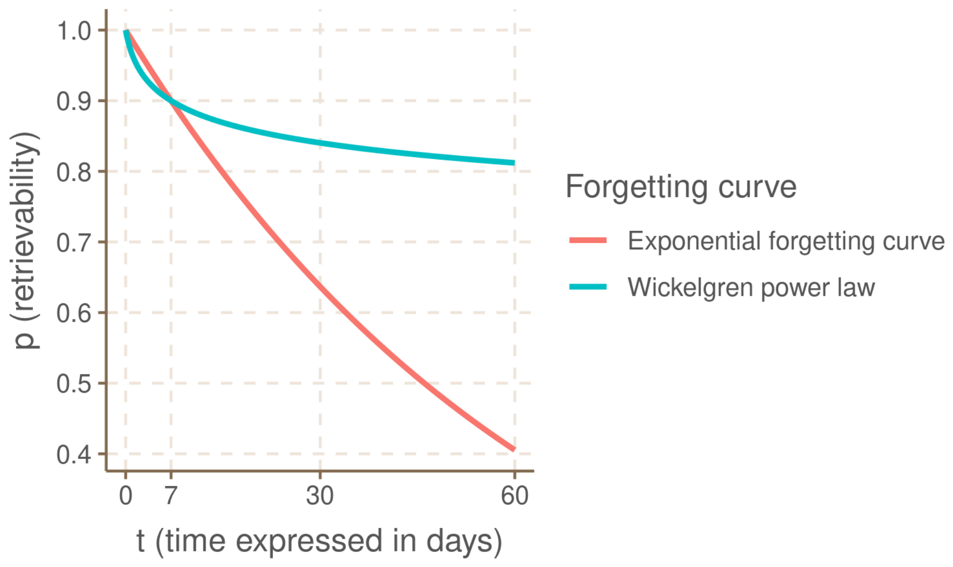


Figure 1.4: A plot of the exponential forgetting curve (in blue) and the Wickelgren power law forgetting curve (in red), both with stability $s = 7$ days. Notice that after 60 days, the Wickelgren power law predicts a value of retrievability that is approximately twice the one predicted by the exponential forgetting curve.

The exponential forgetting curve is at the foundation of the review schedulers developed at SuperMemo [46], of which the memory models are described in Section 2.3.4.

$$p_{\exp}(t; s) = e^{-k\frac{t}{s}} \tag{1.1}$$

the constant $k = -\ln(0.9)$, given our definition of stability s.

Here is the Wickelgren power law [45].

$$p_{\mathrm{pl}}(t; \lambda, \beta, \psi) = \lambda(1 + \beta t)^{-\psi} \tag{1.2}$$

where λ is the initial retrievability, β is a time scaling factor and ψ is the rate of forgetting. For simplicity, we assume that the retrievability starts at 100%, therefore $\lambda = 1$. We also assume $\beta = 1$, time is expressed in days. We also express ψ as a function of stability s. Finally:

$$p_{\mathrm{pl}}(t; s) = (1 + t)^{-\psi(s)}, \quad \psi(s) = -\frac{\ln(0.9)}{\ln(1 + s)} > 0 \tag{1.3}$$

As in Equation 1.1 the only variable other than time that appears in the expression is stability s. The main difference between the two equations is that in Equation 1.1 the rate of decrease is constant, whereas in Equation 1.3 it is not, it slows down with time. Therefore in the exponential forgetting curve we are assuming a constant forgetting rate. In the Wickelgren power law model the forgetting decreases over time. This might seem like a small difference, but it has far-reaching implications when building theories for the inner workings of memories; for example, see [44].

Note that the Wickelgren power law is presented in a slightly different context in [45]. A fixed homogeneous collection of items is studied at an initial time, and we record how many of them are remembered at different time lags. Equivalently we might divide the count of remembered items by the count of items initially studied, we refer to this quantity as *retention rate*. The retention rate is an empirical attempt to approximate retrievability, which is not observed directly.

Empirically, it appears that power laws better fit the data; see [44, 45]. Wozniak in [46] argues that power laws forgetting curves arise from the aggregation of multiple exponentially decaying memory traces. Wixted in [44] reviews this hypothesis along with several others favoring the exponential forgetting curve as the best model to capture the forgetting dynamics. He concludes that the Wickelgren power law is a better model of forgetting both from a predictive and explanatory perspective. Coarsely, for what concerns Wozniak's hypothesis, he reports that you need a large variability in forgetting rate of the different memory traces to produce a credible power law in the aggregate, a condition that is rarely satisfied.

The definition of the forgetting curve presented above has been chosen among many possible alternatives; the idea has a long history. It actually predates the concepts of retrievability and stability. To be as general as possible, we can say that the forgetting curve relates forgetting to time. The phenomenon has been studied many times in the literature, and forgetting has been measured in different ways, leading to different interpretations of the curve. A notable example is the Ebbinghaus forgetting curve, which sparked the empirical study of memory. We now attempt to highlight the differences between the Ebbinghaus forgetting curve and the definition of forgetting curve given above; the latter is closer to the needs of many recently developed memory models (see Chapter 2).

In 1885 Hermann Ebbinghaus reported [13] a years-long series of experiments over which he memorized lists of nonsense syllables to study the dynamics of memory. The setting in which the Ebbinghaus forgetting curve arises is different from the typical setting of a spaced repetition system. He learns a series of nonsense syllables, recording the duration of the initial study. Then he let some time pass. Finally, he restudies the same list and notes the saving in terms of duration to get back to proficiency. For example, assume that initially it takes 10 seconds to study a specific list of syllables until it can be recollected in its entirety at least once. After 24 hours, re-studying up to the same proficiency level takes 8 seconds, the saving is 2 seconds or 20%. The experiment can be opportunely repeated at different time lags, and finally the re-study duration can be plotted for each time lag. He fits a power law relationship between the restudy duration and the log of the time lag. This is typically called *Ebbinghaus forgetting curve*. By the time Ebbinghaus restudied the series of syllables he had mostly lost retrieval access to them, they were unfamiliar. Therefore, the focus of the experiment was on the effects of relearning items to which we have lost access. The purpose of spaced repetition is to maintain access to memorized knowledge, a related but different problem.
In another noteworthy experiment, he noticed the spacing effect in the memorization of nonsense syllables. Loosely, he observes that no amount of initial overlearning saves as much restudying time as a repetition after 24 hours. Ebbinghaus performed the experiments only on himself; therefore, as he also warns, the quantitative conclusions have only individual significance despite his meticulous attention. Subsequently, part of the experiments have been replicated, for example, in [24].

Retrievability concerns cued recall, conscious access to information, whereas the Ebbinghaus forgetting curve concerns time savings in relearning. Although the different meanings of the forgetting curve should not be confused, they all try to capture the dynamics of the same phenomenon: forgetting.

2 | Memory models and state of the art

In this chapter, we first introduce a framework for defining a developing memory models in Section 2.1, then we discuss some of their limitations in Section 2.2 and finally we explore the state of the art in Section 2.3.

2.1. Memory models

In the context of spaced repetition systems, a memory model predicts how likely the student is to remember a card given the review history and the time elapsed since the last review.

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 that represent, respectively, the card under review and the student reviewing it.
- **review history** We define as *review history* of length K an ordered sequence of reviews $R^{(1)}, \dots, R^{(K)}$ where $R^{(i)} = (\Delta^{(i)}, Y^{(i)}) \in \mathbb{R}^+ \times \{0, 1\}$ 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 review rating.
- **time elapsed** Let $\Delta \in \mathbb{R}^+$ be a random variable expressing in days the time elapsed since the last review of the history, or, if the history is empty, since the student was introduced the card. 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 $X = (C, S, (R^{(1)}, \dots, R^{(K)}), \Delta)$. We seek a *memory model*, a function $p_\theta(X)$ with parameters θ such that

$$\mathbb{P}(Y = 1|X = x) = p_\theta(x) \tag{2.1}$$

We call *retrievability* the output probability of recall $p_\theta(x)$. We can highlight the dependence of retrievability on the elapsed time $\delta > 0$. For fixed card c, student s and review history $(r^{(1)}, \ldots, r^{(k)})$, $p_\theta(\delta) = p_\theta(c, s, (r^{(1)}, \ldots, r^{(k)}), \delta)$ is a *forgetting curve*. The now presented framework is similar to the one in Section 1.2.2, the main addition is that the forgetting curve now accounts for the review history.

Our goal is to find an approximation $\hat{p} = p_{\hat{\theta}}$ given a previously collected review 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 $\ell : \{0, 1\} \times \{0, 1\} \to \mathbb{R}^+$ to penalize prediction errors, we compute $\hat{\theta}$ as

$$\hat{\theta} = \underset{\theta}{\operatorname{argmin}} \sum_{c,s} \sum_{k=1}^{k_{cs}} \ell\left(y_{cs}^{(k)}, p_\theta\left(c, s, (r_{cs}^{(1)}, \ldots, r_{cs}^{(k-1)}), \delta_{cs}^{(k)}\right)\right) \tag{2.2}$$

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. We are performing a retrievability regression. What we care about is modeling how retrievability changes over time, so that we can pick a specific date at which the student should review a certain card. The task would be much more difficult with just a set of binary predictions. As we will see in Section 4.1 this framing leads to sensible metrics for comparing memory models.

We close the section with a few final remarks. In this thesis, we consider point predictions about retrievability; future work could explore prediction intervals and their implications for review schedulers. Moreover, future work might account for card interference. We have implicitly assumed *local independence*: the cards are not related to each other. The dependency between review histories of two or more cards covering the same concepts might be explored since reviewing one of them might influence the retrievability of the others. In the literature, memory models are sometimes referred to as *student models*. We prefer the former nomenclature because it is more specific. Student models are employed, for instance, in knowledge tracing, where binary events for correctness of an answer are also studied, but memory and forgetting are not necessarily taken into account. For example, see [30].

2.2. Limitations in modeling student's knowledge

Memory models for spaced repetition systems suffer from three main limitations due to the nature of the data available to us as developers of a spaced repetition system: the data we collect is sparse, fragile, and biased.

The data is sparse. The memory model tries to capture the internal state of memory of the students. The state of memory is dynamic and very complex, each memory depends on other memories, new ideas and understanding are constantly generated, destroyed, and recreated in new forms. With the currently available tools, we cannot directly observe this state. What we can do is approximate a useful representation, limited to material covered by the cards that the student reviews. For each card and for each student, we observe a time-stamped binary sequence of review ratings: after reviewing a card, the student indicates whether the recall was successful or not. We know when a card was introduced to the student and whether he recalled it or not at certain points in time. This is a faint but useful signal into the complex and intertwined state of our memory. It is all we have. Mastery of a subject is dynamic, it is affected by learning and forgetting. We ask students questions and collect binary responses. We cannot expect to fully reconstruct the state of mastery from this little information.

The data is fragile. The measurement itself messes with the memory state [3]. As a simple illustrative example, if the student successfully reviews a card, it is very likely that he also recalls it a few seconds later, independently of the initial uncertainty. Each review is fundamental to the memory model to predict future retrievability. If a single review data point went missing, we could be underestimating the retrievability for that item (or overestimating, depending on the time and rating of the review). This is what always happens though. Students do not live inside a crystal ball only interacting with the spaced repetition system. They live in a rich environment. Our hope is that the material they review plays an important role in their lives and that they interact with it outside the context of spaced repetition. Each and every time they do that it is an interaction we cannot capture with our system, but that possibly plays an important role in shaping the student's internal memory state. The real-world performance of a memory model is severely constrained by the information we can extract from the environment, which is limited just to the student's interactions with the spaced repetition system.

The data is biased. Another important limitation is that spaced repetition systems suffer from a *chicken or the egg* problem: the memory model is fitted to the data, but the data collected is biased by the memory model. We want to fit an accurate memory model to the data collected from the system and then use this model to schedule reviews. The schedule will bias the future collected data, possibly impairing further optimization of the memory model. Some of the implications have been explored in [27].

However, not everything is lost; we will see how we can make good use of the available information. As we will see in Chapter 4 the memory models presented in the following and in Chapter 3 are able to make retrievability predictions that are better than chance and are often accurate. Not only that, many of those memory models have fairly interpretable dynamics that might allow us to glimpse the inner workings of memory. We do not only seek accurate prediction, a good memory model should also be a source of good explanations.

2.3. State of the art

This section provides a detailed report on the state of the art in the development of memory models. Since we are exploring memory models with the goal of employing them in a spaced repetition system, 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.

2.3.1. 1PL-IRT

Our dataset contains reviews for different students and different cards. We can think of reviews as tests and employ the *Item Response Theory (IRT)* statistical framework to predict test responses for student-card pairs. The role of *items* in IRT is played by cards.

The drawback of this approach is that we are discarding time information, that is we are not accounting for forgetting. If the student does not review any card, the retrievability predicted from the model described below does not change over time.

We employ the simplest IRT model: *1PL-IRT*. We cast the model in the *Generalized Linear Mixed Models (GLMM)* framework, following [4]. We have a linear component:

$$\eta_\theta(s,c) = a_s - d_c \tag{2.3}$$

where $\theta = \{a_s\}_{s\in\mathcal{S}} \cup \{d_c\}_{c\in\mathcal{C}}$. $a_s \sim \mathcal{N}(0,\sigma_s^2)$ are random effect parameters that represent student ability and d_c are fixed effect parameters for card difficulty. Student ability does not necessarily capture the student's skills in the domain being studied. Student ability might capture other factors such as attitude towards the domain or towards spaced repetition in general.

The linear component η_θ and p_θ are related through the logit link function:

$$\eta_\theta = \ln\left(\frac{p_\theta}{1-p_\theta}\right) \tag{2.4}$$

Finally, with $\sigma(x) = (1 + \exp(-x))^{-1}$ indicating the logistic function, we have:

$$p_\theta(c, s) = \sigma(a_s - d_c) \tag{2.5}$$

By considering the student ability as a random effect, we assume that the students have been randomly and independently sampled from a larger population of students with probability distribution empirically approximated by our observed sample of students [43, Section 2.2.2]. In particular, we do not focus our attention on estimating the ability of every individual student in $\mathcal{S}$. The same cannot be said for the difficulty of the card, as it is regarded as a fixed effect.

The model assumes *local independence*: independence between reviews by the same student. We note that our data does not satisfy this assumption. In particular, dependencies are induced by both the hierarchical structure (decks are made of cards) and the repeated reviews on the same card over time. In the present thesis, we ignore this problem. The problem can be tackled in future work, for instance, by introducing additional random effects representing the structure.

We do not consider 2PL-IRT or 3PL-IRT, since [21] found a negligible gain in predictive performance compared to 1PL-IRT when they are employed to model student memory.

2.3.2. DASH and variants

Lindsey et al. [21, 23] set out to build a model for personalized review that tracks the state of knowledge of each student and adapts to them. They did that by integrating psychological theory with big-data methods in a spirit that is very precious for the present thesis.

They present the *DASH (Difficulty, Ability and Study History)* model that, as in the framework of Section 2.1, relates retrievability to three factors: card difficulty, student ability, and review history for a card-student pair. The elapsed times between reviews in the card history enter the model through several time windows W. We pick $W = \{1, 7, 30, \infty\}$ days; we follow the choice in [21] but drop the hour time window, since we use a day resolution. The model predicts retrievability as

$$p_\theta(c, s, (r^{(1)}, \dots, r^{(k)}), \delta) = \sigma\left(a_s - d_c + \sum_{w=1}^{|W|} \theta_{2w-1} \ln(1 + c_w) + \theta_{2w} \ln(1 + n_w)\right) \tag{2.6}$$

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 $c \in \mathcal{C}$, as for the 1PL-IRT model of Section 2.3.1. c_w is the number of times the student s correctly recalled card c in window W_w out of n_w attempts. $\{\theta_1, \dots, \theta_{2|W|}\}$ are window-specific parameters. The parameters are $\theta = \{a_s\}_{s \in \mathcal{S}} \cup \{d_c\}_{c \in \mathcal{C}} \cup \{\theta_1, \dots, \theta_{2|W|}\}$. Using the notation $\delta^{(i:j)} = \sum_{h=i}^{j} \delta^{(h)}$:

$$c_w\left((r^{(1)}, \dots, r^{(k)}), \delta\right) = \sum_{i=1}^{k} \mathbb{I}_{[0, W_w]}\left(\delta + \delta^{(i+1:k)}\right) y^{(i)} \tag{2.7}$$

$$n_w\left((r^{(1)}, \dots, r^{(k)}), \delta\right) = \sum_{i=1}^{k} \mathbb{I}_{[0, W_w]}\left(\delta + \delta^{(i+1:k)}\right) \tag{2.8}$$

$\delta + \delta^{(i+1:k)}$ represents the time that elapsed between now and the i-th review of the history (assuming δ represents the elapsed time between now and the last review of the history).

The forgetting curve after one or more reviews is a step-function. As time elapses, fewer and fewer reviews of the history are included in the time windows, retrievability eventually reaches a constant positive level that depends on a_s, d_c (and eventually window parameters if a time window of infinite length is included, as happens in [21] and as we replicate for the comparison of Chapter 4). Examples of DASH forgetting curves are reported in Figure 2.1.

DASH was originally presented as part of a more general framework:

$$p_\theta(c, s, (r^{(1)}, \dots, r^{(k)}), \delta) = \sigma\left(a_s - d_c + h_\theta\left((r^{(1)}, \dots, r^{(k)}), \delta\right)\right) \tag{2.9}$$

The dependence of retrievability on the review history is isolated by the function h_θ. Notice how 1PL-IRT is an example of this general framework with $h_\theta = 0$.

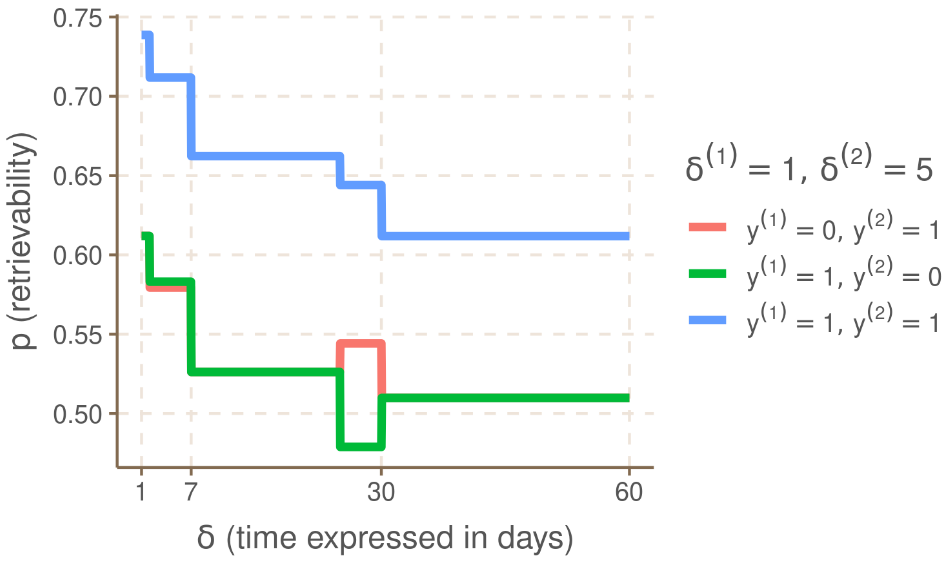


Figure 2.1: Examples of DASH forgetting curves. We sampled a student and a card from the first train-test sample of Section 4.3 and simulated three review histories of two reviews each, the details are reported in the figure's legend. For each review history, we fit a DASH memory model $\hat{p}_\theta$ and plot the predicted retrievability as a function of the time δ elapsed since the last of the two reviews.

They present two more instances of this framework, inspired by psychological theory: *DASH[MCM]* inspired by the Multiscale Context Model (MCM) [29] and *DASH[ACT-R]* inspired by the memory module in the ACT-R cognitive architecture.

In DASH[MCM] the counts c_w and n_w decay over time at a window-specific rate τ_w (fixed a priori). As in the choice of W, we fix the decay rates similarly to the original paper, as $\tau_{1:W} = \{0.2434,\ 1.9739,\ 16.0090,\ 129.8426\}$ [21]. In Equation 2.6 c_w and n_w are replaced with

$$c_w\left((r^{(1)},\dots,r^{(k)}),\delta\right) = \sum_{i=1}^{k} \mathbb{I}_{[0,W_w]}\left(\delta+\delta^{(i+1:k)}\right) e^{-\left(\delta+\delta^{(i+1:k)}\right)/\tau_w} y^{(i)} \tag{2.10}$$

$$n_w\left((r^{(1)},\dots,r^{(k)}),\delta\right) = \sum_{i=1}^{k} \mathbb{I}_{[0,W_w]}\left(\delta+\delta^{(i+1:k)}\right) e^{-\left(\delta+\delta^{(i+1:k)}\right)/\tau_w} \tag{2.11}$$

In DASH[ACT-R] we have:

$$h_\theta\left((r^{(1)},\dots,r^{(k)}),\delta\right) = \theta_1 \ln\left(1+\sum_{i=1}^{k}\theta_{3+y^{(i)}}\left(\delta+\delta^{(i+1:k)}\right)^{-\theta_2}\right) \tag{2.12}$$

Only in DASH[ACT-R] h_θ is not linear in θ. Information on how we fit the models for the experiment in Chapter 4 is provided in Section 4.2. In Figures 2.2, 2.4, we show examples of DASH[MCM] and DASH[ACT-R] forgetting curves.

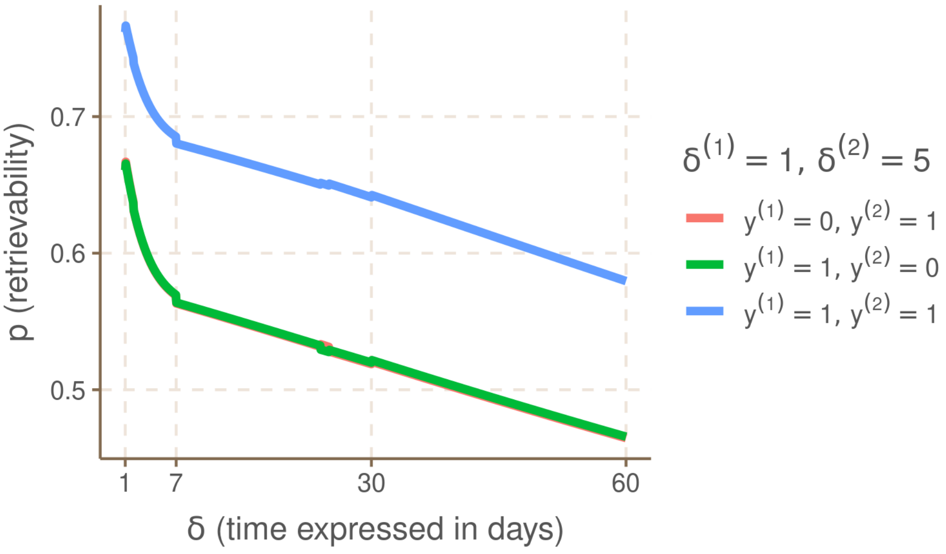


Figure 2.2: Examples of DASH[MCM] forgetting curves. We proceed as in Figure 2.1, with the same student and card.

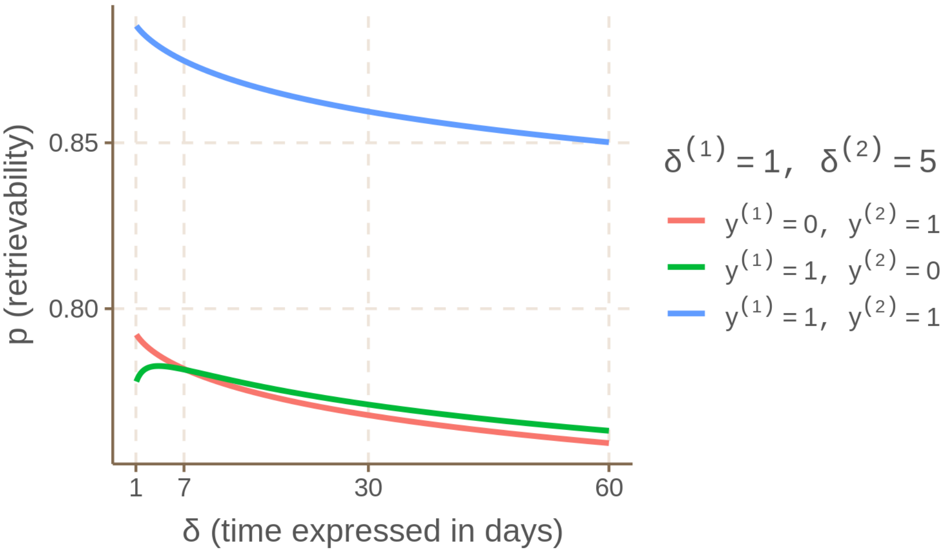


Figure 2.3: Examples of DASH[ACT-R] forgetting curves. We proceed as in Figure 2.1, with the same student and card.

Choffin et al. [8] introduce a new model: *DAS3H*. DAS3H extends DASH by introducing multiple-skills tagging. Each card is tagged with one or more skills required to answer correctly. The memory dynamics are then allowed to differ between skills. In our experience skills data is hard to come by in spaced repetition systems, in particular, in the datasets considered in Chapter 4 we do not have that kind of data and so we exclude DAS3H from the comparison.

2.3.3. Half-Life Regression

Half-life regression (HLR) is a memory model designed for learning language vocabulary with spaced repetition [36], we adapt it to a more general setting. In the original model each card regards a word. Words are tagged by lexeme, these *lexeme* tags are taken into account when predicting the retrievability of words.

HLR assumes an exponential forgetting curve of the form:

$$p_\theta(\delta) = 2^{-\frac{\delta}{h_\theta}} \tag{2.13}$$

where h_θ is called *half-life*. Compared to Equation 1.1 the base of the exponential changes and h_θ is stability up to a constant factor, so the discussion in Section 1.2.2 applies.

The half-life depends on the scalar product of the weights θ and a feature vector. Here, we report the shape of the feature vector employed in the comparison of Chapter 4 which tries to be as faithful as possible to the original paper, but is inevitably constrained by the lack of word-specific information:

$$h_\theta(c, s, (r^{(1)}, \dots, r^{(k)})) = 2^{\theta_1\sqrt{1+\sum_{i=1}^{k} y_i}+\theta_2\sqrt{1+\sum_{i=1}^{k}(1-y_i)}+\theta_c+\theta_s} \tag{2.14}$$

In the original model the predictors are enhanced with additional indicator variables, one for each lexeme tag considered. The set of weights θ is empirically fit to review data by minimizing the following loss function.

$$\ell(y, p_\theta(\delta)) = (y - p_\theta(\delta))^2 + \lambda\|\theta\|_2^2 \tag{2.15}$$

where λ is a hyperparameter. In the original paper the loss contains an additional term for the squared deviation of h_θ to the observed half-life $\frac{-t}{\log_2 p_\theta}$, the computation of the latter is possible because they consider review ratings in the interval $[0, 1]$, instead of binary review ratings.

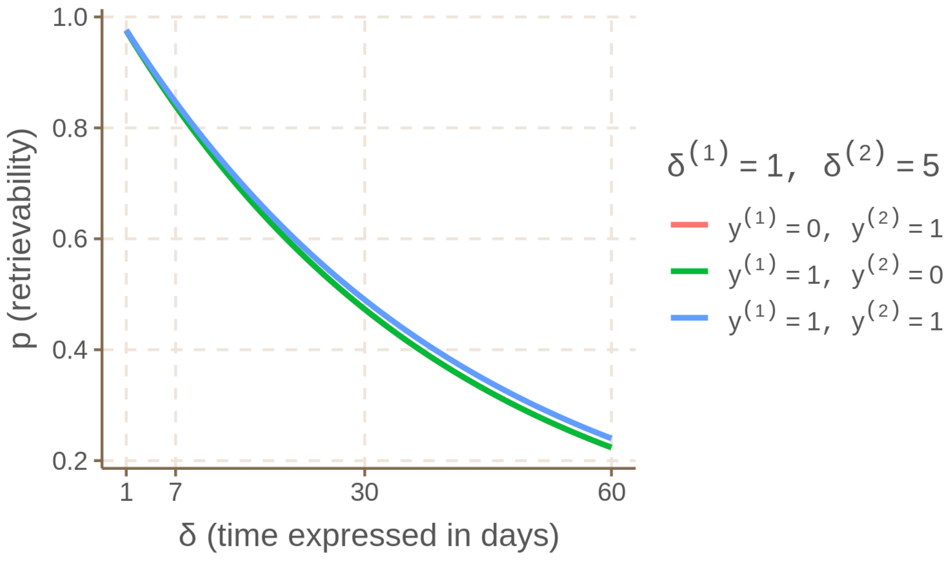


Figure 2.4: Examples of HLR forgetting curves. We proceed as in Figure 2.1, with the same student and card.

They fit the model on a large dataset (containing more than 12 million observations) consisting of two weeks of log data from the popular Duolingo language learning app. They employ gradient descend.

In this thesis, we are not assuming any specific learning domain; HLR can still be applied by dropping the word features. The fairness of the comparison of Chapter 4 might be compromised; we argue that it is not the case since other models could similarly be enhanced with lexeme tags if available.

2.3.4. SuperMemo Algorithm SM-17 and SM-18

Piotr Wozniak, along with the SuperMemo World company, has been developing the SuperMemo software (`https://www.supermemo.com`) for three decades. SuperMemo is the first spaced repetition system and one that still serves millions of students. SuperMemo and the literature on review schedulers developed in parallel, as far as we know the recent SuperMemo algorithms have not been considered in the literature. One of the goals of this thesis is to fill this gap; the SuperMemo algorithms are potential sources of invaluable insights for the development of memory models and on the inner workings of memory. An account of the history of the SuperMemo algorithms can be found in [46].

SuperMemo Algorithm SM-18 (SM-18) [48] is the review scheduler used in SuperMemo since 2019. In this section, we focus mainly on its predecessor *SuperMemo Algorithm*

SM-17 (SM-17) [47], which has been a great improvement over previous versions of the algorithm, and is of significant importance for the developments of Chapter 3. The improvement of SM-18 over SM-17 is not as large; we briefly discuss the differences between the two at the end of the section. Previous versions of the SuperMemo Algorithm were largely heuristic in nature, the significance of SM-17 is that, in contrast, the algorithm is based on psychology results and can now learn and adapt to any review history, much in the spirit of [23].

SM-17 is described on the web page [47], but many important details are missing that prevent a faithful reproduction of the algorithm. That is why we do not include SM-17 in the comparison of Chapter 4. SM-17 is a review scheduler, but we can isolate the memory model component. Here we try to summarize some of its aspects that are important for the development of Chapter 3.

SM-17 is based on the two components description of memory of Section 1.2.1, besides retrievability SM-17 explicitly models stability and its dynamics as the student reviews a card. Moreover, card difficulty is taken into account. The core idea is to model memory with forgetting curves, which we described in Section 1.2.2. They employ an exponential forgetting curve (Equation 1.1) to describe the decline in retrievability over time at a rate determined by stability.

We need a few definitions. For the remainder of the section we focus on a single review history, therefore fix a card $c \in \mathcal{C}$ and a student $s \in \mathcal{S}$. Let $r^{(1)}, \ldots, r^{(K)}$ be the review history of length K of c by s. Denote by $p^{(i)}$ the retrievability estimate before the i-th review, $p^{(i)}(\delta) = p_\theta(c, s, (r^{(1)}, \ldots, r^{(i-1)}), \delta)$ is the retrievability δ days after the $i-1$-th review. Note that the review history can be empty. Below we recursively define $s^{(i)}$, the stability before the i-th review. It represents the interval of time after which a theoretical retrievability estimate obtained with the exponential forgetting curve falls below 90%. In addition to retrievability and stability, a third variable is introduced: difficulty $d_c \in [0, 1]$ for the card $c \in \mathcal{C}$. It is defined as the maximum possible increase in stability for the card c linearly mapped to the interval $[0, 1]$. We omit the computation of card difficulty from the discussion, since it is not relevant to the developments of Chapter 3. Since earlier we fixed the card $c \in \mathcal{C}$, we dropped the related index, $d = d_c$. Finally, let $l^{(i)} = \sum_{j=1}^{i}(1-y^{(j)})$ be the number of lapses up to and including review i, *lapse* is just another name for an incorrect review. Notice that before the instant of time in which the student rates the i-th review $\delta^{(i)}$, $s^{(i)}$, and $p^{(i)}$ are known (as well as all previous reviews in the history); $y^{(i)}$ and $l^{(i)}$ are not known, they will be available only after the student rates the card; our goal is to compute $p^{(i)}$ before observing $y^{(i)}$, before the student rates the card.

To model the dynamics of memory, SM-17 uses three functions: *stability increase function*, *first post-lapse stability function*, and *recall function*. We discuss each of them below:

- The stability increase function $S_{Inc}(p, s, d)$ depends on retrievability p, stability s, and difficulty d, it determines how stability changes after a successful review. If the review $i-1$ is successful ($y^{(i-1)} = 1$), then $s^{(i)} = s^{(i-1)} S_{Inc}\left(p^{(i-1)}(\delta^{(i-1)}), s^{(i-1)}, d\right)$. $p^{(i-1)}(\delta^{(i-1)})$ is the retrievability estimate right before the review $i-1$ is rated.
- The first post-lapse stability function $PLS(l, p)$ depends on the number of lapses l and retrievability p, it determines the stability after a lapse. If the review $i-1$ is unsuccessful ($y^{(i-1)} = 0$), then $s^{(i)} = PLS(l^{(i-1)}, p^{(i-1)}(\delta^{(i-1)}))$.
- The recall function $Recall(p, s, d)$ depends on an estimate of retrievability p, stability s, and difficulty d, and is used to correct a theoretical estimate of retrievability. Given the stability $s^{(i)}$, a theoretical estimate of retrievability is computed based on the exponential forgetting curve formula of Equation 1.1: $p_{fc}^{(i)}(\delta) = \exp(\ln(0.9)\delta/s^{(i)})$. Retrievability δ days after review $i-1$ is finally computed as $p^{(i)}(\delta) = Recall(p_{fc}^{(i)}(\delta), s^{(i)}, d)$.

The memory model p_θ is built iteratively, at each review step we compute stability using the functions S_{Inc} and PLS, then the exponential forgetting curve gives us an estimate of retrievability that we correct with the function $Recall$. Information about the shape of the three functions or about how they are fitted is not provided by SuperMemo. We remark that the retrievability and stability estimates in SM-17 are more refined compared to the present description; what we have reported is a summary of the information we consider important for the developments of Chapter 3, more information is available in [47].

To model retrievability at any point in time, we miss a final ingredient: the startup stability $s^{(1)}$, which determines the stability for newly introduced cards, before they are reviewed by the student. Finally, we can put everything together. Let us start from the first review step. As a card of difficulty d is introduced to the student, we assign stability $s^{(1)}$. The student reviews the card at time $\delta^{(1)}$. Before observing the rating $y^{(1)}$ we can compute the retrievability estimate $p^{(1)}(\delta^{(1)}) = Recall(p_{fc}^{(1)}(\delta^{(1)}), s^{(1)}, d)$. We then observe $y^{(1)}$ and compute the stability $s^{(2)} = y^{(1)} s^{(1)} S_{Inc}(p^{(1)}(\delta^{(1)}), s^{(1)}, d) + (1 - y^{(1)}) PLS(l^{(1)}, p^{(1)}(\delta^{(1)}))$. Now, again, thanks to the $Recall$ function we obtain $p^{(2)}(\delta)$ and we can iterate the whole process for any future review.

We left out the discussion on how difficulty is computed; we only remark that the main difference between SM-18 and SM-17 lies here. The card difficulty is estimated from the data; in the older version, it is assumed constant, while in the newer version, it is allowed to change in the course of learning.

3 | The R-17 and DASH[RNN] memory models

In this chapter, we develop two novel memory models. Both models are developed within the framework of *recurrent neural networks (RNNs)* [16], the R-17 model is described in Section 3.1 and the DASH[RNN] model in Section 3.2. We use the notation of Chapter 2.

3.1. R-17

We now introduce *R-17*. It is a neural network approximation of SuperMemo Algorithm SM-17, which has been described in Section 2.3.4. We use the notation of the latter section. Our aim with R-17 is to get a hint about the performance of SM-17 as a memory model.

Here is the general idea. Since we do not know the shapes of the functions S_{Inc}, PLS, and $Recall$, we approximate them with neural networks. Together, the neural networks form a single RNN that is trained end-to-end.

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 iteratively as in SM-17. First, we calculate an estimate of the ease (or equivalent difficulty) of the card c for the student s. Then, for each review step $i = 1$ to $i = k$, iteratively we obtain a stability estimate for the forgetting curve and incorporate time information to obtain a retrievability estimate. Let us unpack the modules that make up R-17:

- We compute the *ease* $\sigma_e(c, s) \in [0, 1]$ of the card c for student s as in Equation 3.1. $\sigma_e(c, s)$ can be viewed as the output of a single dense layer with a logistic (sigmoid) activation function. This choice was inspired by the 1PL-IRT model described in Section 2.3.1. With respect to Section 2.3.4 the ease is to be interpreted as one minus difficulty d_c.
- 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 ReLU activation function, the third of 5 units with ReLU activation function, therefore $h^{(i)} \in \mathbb{R}^5$ for all i. We need to pay special attention to the first hidden state $h^{(1)}$, we cannot compute it with H since we lack the required inputs. We set $h^{(1)} = H_0 \in \mathbb{R}^5$ as a vector with trainable components.

- At each review step $i \geq 1$, we calculate $\psi^{(i)} = \Psi(h^{(i)}, \sigma_e(c, s)) \in \mathbb{R}$ 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. Together, Ψ and H replace the functions S_{Inc} and PLS in SM-17.

- Given stability $\psi^{(i)}$ we can compute $p_{fc}(\delta; \psi^{(i)}) = (1+\delta)^{-\psi^{(i)}}$ the theoretical retrievability estimate δ days after review $i-1$. Notice that we replaced the exponential forgetting curve of SM-17 with a Wickelgren power law, the choice is motivated by the discussion of Section 1.2.2. Moreover, notice that for numerical stability reasons, we changed the interpretation of stability (compare p_{fc} with Equation 1.2) the result is unchanged given that there exists a bijection between ψ and s in the latter equation. 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 correct the estimate with the neural network P which replaces the *Recall* function of SM-17. We compute the retrievability prediction before the i-th review step $p^{(i)} = P(p_{fc}^{(i)}, \psi^{(i)}, \sigma_e(c, s)) \in \mathbb{R}$. Where P is a neural network with 3 dense layers, the first and second layers of 8 units each, and the ReLU activation function; the last layer with univariate output and the logistic (sigmoid) activation function.

The modules are composed in Figure 3.1, which shows the computation of a couple of review steps for R-17. For what concerns the first review step $i = 1$, note that $p^{(i)}$ depends on the information of previous review steps only through $h^{(i)}$, $h^{(1)}$ as defined above resolves the issue of the empty review history.

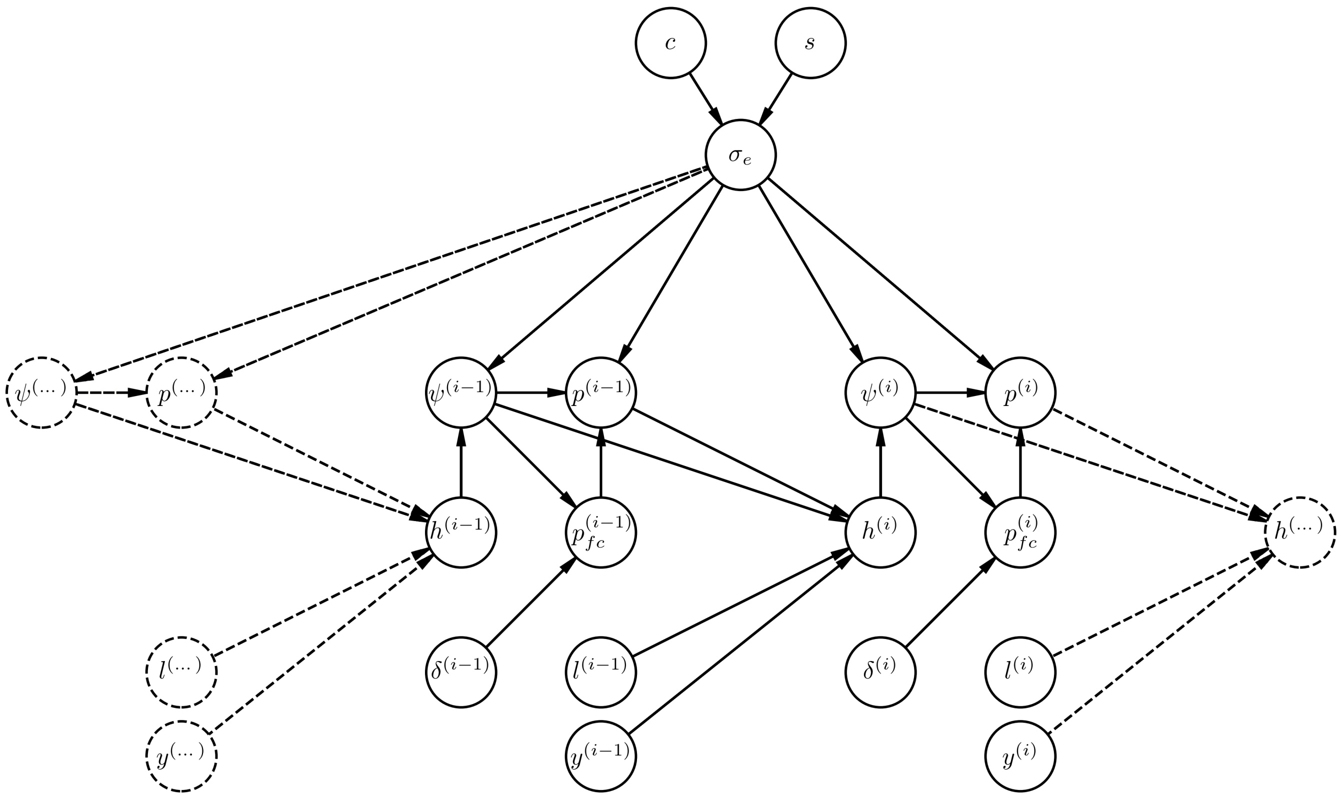


Figure 3.1: The R-17 memory model computing retrievability $p^{(i)} = p_\theta(c, s, (r^{(1)}, \dots, r^{(i-1)}), \delta^{(i)})$ for review $i \geq 2$. At each review step the inputs are $\delta^{(i)}$, $y^{(i-1)}$ and $l^{(i-1)}$. First we compute a new hidden state $h^{(i)}$ from previous stability 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 ease $\sigma_e(c, s)$. 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 ease $\sigma_e(c, s)$. The calculation is reported in detail in Equations 3.1 to 3.10.

In Figure 3.2 we show examples of forgetting curves obtained with R-17 and in Figure 3.3 the approximations of the SM-17 functions S_{Inc}, PLS, and $Recall$. The latter figure also displays the interpretability of R-17. The neural networks composing it all have a small number of interpretable inputs, therefore we can inspect the behavior of R-17 by plotting them.

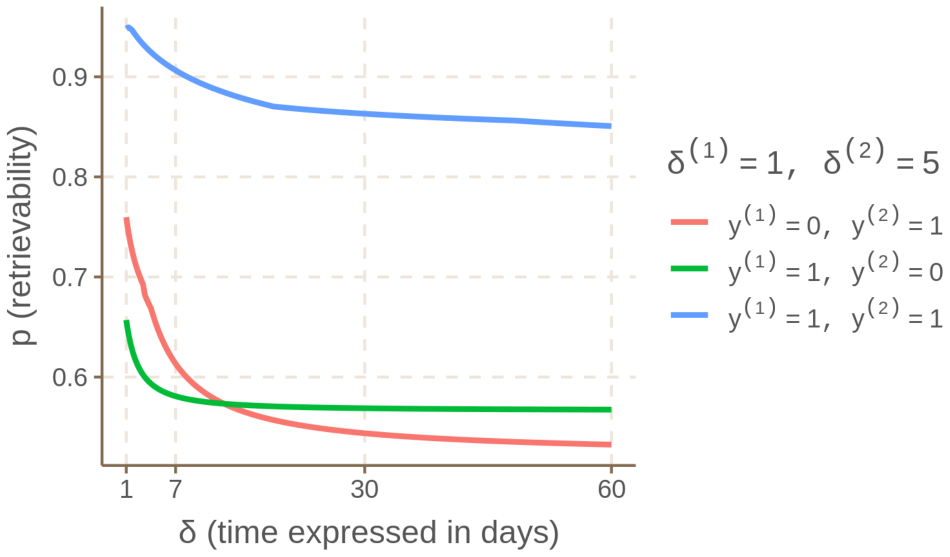


Figure 3.2: Examples of R-17 forgetting curves. We proceed as in Figure 2.1, with the same student and card.

The equations to obtain $p^{(k)} = p_\theta(c, s, (r^{(1)}, \dots, r^{(k-1)}), \delta^{(k)})$ are reported below. First, we compute $\sigma_e(c, s)$ as

$$\sigma_e(c, s) = \sigma(b_e + a_s - d_c) \tag{3.1}$$

Equation 3.1 is the calculation of a dense layer with a single output and a logistic (sigmoid) activation function $\sigma(x) = (1 + \exp(-x))^{-1}$. $b_e \in \mathbb{R}$ is a trainable bias and $a_s, d_c \in \mathbb{R}$ are trainable weights, one for each card plus one for each student, which are one-hot encoded to be fed into the dense layer.

After computing $\sigma_e(c, s)$, for each time step from $i = 1$ to $i = k$, we computed

$$\begin{aligned}
h^{0,(i)} &= H(\psi^{(i-1)}, p^{(i-1)}(\delta^{(i-1)}), y^{(i-1)}, l^{(i-1)}) = && (3.2)\\
&= \max(0, b_{h^0} + \mathbf{W}_{h^0}[\psi^{(i-1)}, p^{(i-1)}(\delta^{(i-1)}), y^{(i-1)}, l^{(i-1)}]^T) \quad \text{if } i \geq 2 && (3.3)\\
h^{1,(i)} &= \max(0, b_{h^1} + \mathbf{W}_{h^1} h^{0,(i)}) \quad \text{if } i \geq 2 && (3.4)\\
h^{(i)} &= \begin{cases} H_0 & \text{if } i = 1 \\ \max(0, b_h + \mathbf{W}_h h^{1,(i)}) & \text{if } i \geq 2 \end{cases} && (3.5)\\
\psi^{(i)} &= \Psi(h^{(i)}, \sigma_e(c, s)) = \max(0, b_\psi + \mathbf{W}_\psi [h_1^{(i)}, \dots, h_5^{(i)}, \sigma_e(c, s)]^T) && (3.6)
\end{aligned}$$

$$p_{fc}^{(i)} = p_{fc}(\delta^{(i)}; \psi^{(i)}) = (1 + \delta^{(i)})^{-\psi^{(i)}} \tag{3.7}$$

$$p^{0,(i)} = \max(0, b_{p^0} + \mathbf{W}_{p^0}[p_{fc}^{(i)}, \psi^{(i)}, \sigma_e(c,s)]^T) \tag{3.8}$$

$$p^{1,(i)} = \max(0, b_{p^1} + \mathbf{W}_{p^1} p^{0,(i)}) \tag{3.9}$$

$$p^{(i)} = \sigma(b_p + \mathbf{W}_p p^{1,(i)}) \tag{3.10}$$

The $282 + |\mathcal{C}| + |\mathcal{S}|$ trainable parameters are

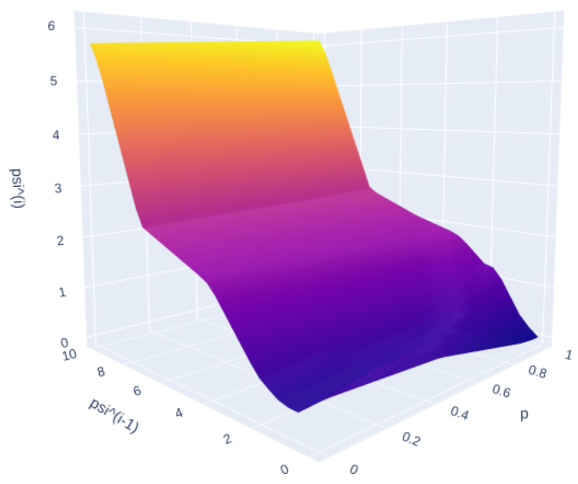
- $b_e \in \mathbb{R}$, $a_s, d_c \in \mathbb{R}$ for all $c \in \mathcal{C}, s \in \mathcal{S}$ for a total of $1 + |\mathcal{C}| + |\mathcal{S}|$ parameters;
- $H_0 \in \mathbb{R}^5$, $b_{h^0}, b_{h^1} \in \mathbb{R}^8$, $b_h \in \mathbb{R}^5$, $\mathbf{W}_{h^0} \in \mathbb{R}^{8\times 4}$, $\mathbf{W}_{h^1} \in \mathbb{R}^{8\times 8}$, $\mathbf{W}_h \in \mathbb{R}^{5\times 8}$ for a total of 162 parameters;
- $b_\psi \in \mathbb{R}$, $\mathbf{W}_\psi \in \mathbb{R}^{1\times 6}$ for a total of 7 parameters;
- $b_{p^0}, b_{p^1} \in \mathbb{R}^8$, $b_p \in \mathbb{R}$, $\mathbf{W}_{p^0} \in \mathbb{R}^{8\times 3}$, $\mathbf{W}_{p^1} \in \mathbb{R}^{8\times 8}$, $\mathbf{W}_p \in \mathbb{R}^{1\times 8}$ for a total of 113 parameters.

We train the RNN end-to-end with gradient descend. The total loss is minimized as in Equation 2.2, with single review step loss

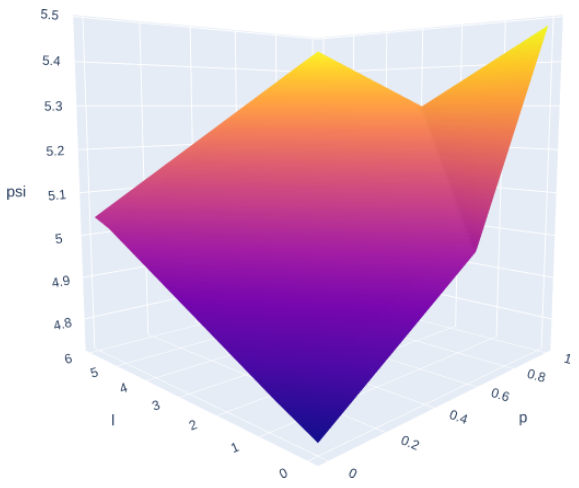
$$\ell(y, p) = (y - p)^2 + \lambda \|\theta_{\sigma_e}\|_2^2 \tag{3.11}$$

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 [16]. We need a retrievability prediction for each review step of the histories in our dataset. Here we presented R-17 as making a retrievability prediction after k steps, but notice that we do not have to recompute the whole sequence of hidden states for each review step, it is clear how we can obtain a retrievability prediction after each subsequent step.

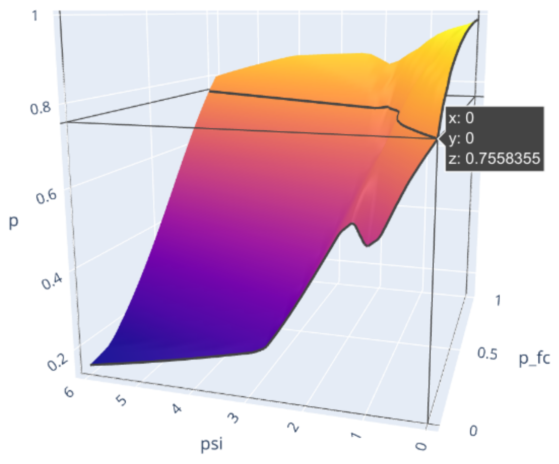
Note that memory model R-17 adapts to a student or to a card only through the ease σ_e, which induces a dependency structure between cards reviewed by the same student and students of the same card. This is a limitation of R-17 compared to SuperMemo Algorithm SM-17. The latter is able to tune the functions S_{Inc}, PLS, and $Recall$ after each review step to adapt to the student. Another limitation of R-17, this time with respect to SuperMemo Algorithm SM-18, is that the ease of a card for a student cannot change as new reviews arrive in the system. An possible advantage of R-17 over SM-17 is that information is shared between review steps through a hidden state vector of 5 real components. In SM-17 the same happens only through the univariate stability estimate.



(a) $\Psi(H(\psi, p, 1, 0), \sigma_e))$



(b) $\Psi(H(5, p, 0, l), \sigma_e))$



(c) $P(p_{fc}, \psi, \sigma_e))$

Figure 3.3: We show the approximations of the SM-17 functions S_{Inc}, PLS, and $Recall$. The S_{Inc} function (multiplied by previous stability) is approximated by the composition $\Psi \circ H$ for the inputs ψ, p and σ_e while fixing $y = 1$. In the plot we fix σ_e to the value computed for the same student and card as in Figure 3.2, and the additional input $l = 0$. The PLS function is approximated by the same composition $\Psi \circ H$ but with inputs p and l while fixing $y = 1$. In the plot we fix the additional inputs $\psi = 5$ and σ_e, the latter as before. Note that in both approximations we need to fix additional inputs that are not part of the corresponding SM-17 functions. Finally, the $Recall$ function is approximated by P for the inputs ψ, p, and σ_e; in the plot the latter is fixed as before.

3.2. DASH[RNN]

The *DASH[RNN]* memory model is an instance of the DASH framework where h_θ is the output of a RNN (see Equation 2.9 in Section 2.3.2). The idea is hinted at in [35]. Our aim with DASH[RNN] is to obtain accurate retrievability predictions.

The model is presented in Figure 3.5 and the equations are reported in detail below.

Let $r^{(1)}, \dots, r^{(k-1)}$ be a review history of length $k-1$, where $r^{(i)} = (\delta^{(i)}, y^{(i)})$. $\delta^{(1)}$ is the time elapsed between the first review and the introduction of the card to the student, since we need a previous review rating but we don't have one, we set the dummy rating $y^{(0)} = 1$. Moreover, to simplify the notation, we set $\delta^{(k)} = \delta$ (our goal is to predict the retrievability for the k-th review). Let $h^{(0)} = 0 \in \mathbb{R}^5$ be the initial hidden state. For each time step from $i = 1$ to $i = k$, we compute

$$d^{0,(i)} = \max(0, b_{d^0} + \mathbf{W}_{d^0}[\delta^{(i)}, y^{(i-1)}]^T) \tag{3.12}$$

$$d^{(i)} = \max(0, b_d + \mathbf{W}_d d^{0,(i)}) \tag{3.13}$$

$$h^{(i)} = \max(0, b_h + \mathbf{W}_{h,d} d^{(i)} + \mathbf{W}_{h,h} h^{(i-1)}) \tag{3.14}$$

Equation 3.12 is the computation of a dense layer of 12 units and with the ReLU activation function; with trainable bias $b_{d^0} \in \mathbb{R}^{12}$ and weights $\mathbf{W}_{d^0} \in \mathbb{R}^{12\times 2}$, the output is $d^{0,(i)} \in \mathbb{R}^{12}$. Equation 3.13 is the computation of a dense layer of 12 units and with the ReLU activation function; with trainable bias $b_d \in \mathbb{R}^{12}$ and weights $\mathbf{W}_d \in \mathbb{R}^{12\times 12}$, the output is $d^{(i)} \in \mathbb{R}^{12}$. Equation 3.14 is the computation of a simple RNN layer of 5 hidden units and with the ReLU activation function; with trainable bias $b_h \in \mathbb{R}^5$, weights for dense-to-hidden connections $\mathbf{W}_{h,d} \in \mathbb{R}^{5\times 12}$ and weights for hidden-to-hidden connections $\mathbf{W}_{h,h} \in \mathbb{R}^{5\times 5}$, the output is $h^{(i)} \in \mathbb{R}^5$. Finally, setting $h_\theta = b_\sigma + \mathbf{W}_\sigma h^{(k)}$ in Equation 2.9 we obtain the retrievability prediction

$$p_\theta(c, s, (r^{(1)}, \dots, r^{(k-1)}), \delta) = \sigma\left(a_s - d_c + b_\sigma + \mathbf{W}_\sigma h^{(k)}\right) \tag{3.15}$$

Equation 3.15 can be seen as the computation of a dense layer with a single output and logistic (sigmoid) activation function with trainable bias $b_\sigma \in \mathbb{R}$ and weights $\mathbf{W}_\sigma \in \mathbb{R}^{1\times 5}$, where $c \in \mathcal{C}$ and $s \in \mathcal{S}$ are one-hot encoded. Casting the DASH model as a neural network allows us to train DASH[RNN] end-to-end as a RNN.

The total number of parameters is $N_{d^0} + N_d + N_h + N_\sigma + |\mathcal{C}| + |\mathcal{S}| = 288 + |\mathcal{C}| + |\mathcal{S}|$ with $N_{d^0} = 36$ parameters for the first dense layer, $N_d = 156$ parameters for the second dense layer, $N_h = 90$ for the simple RNN layer, and $N_\sigma = 6$ plus $|\mathcal{C}| + |\mathcal{S}|$ for the final layer.

We train the RNN end-to-end with gradient descend. The total loss is minimized as in Equation 2.2, with single review step loss

$$\ell(y, p) = (y - p)^2 \tag{3.16}$$

for the target rating y and the corresponding retrievability prediction p. The gradients are computed with the *back-propagation through time (BPTT)* algorithm, for the optimization we employ the Adam optimizer [16].

In Figure 4.1 we show examples of forgetting curves obtained with DASH[RNN].

Compared to R-17, the DASH[RNN] memory model has a less convoluted structure, but a similar number of parameters. Our aim with this model is to obtain accurate retrievability predictions; in order to do that, we sacrifice the model interpretability of R-17. In the model comparison of Chapter 4, DASH[RNN] often outperforms other models, including R-17, in some carefully chosen metrics for the evaluation of the predictive performance of memory models.

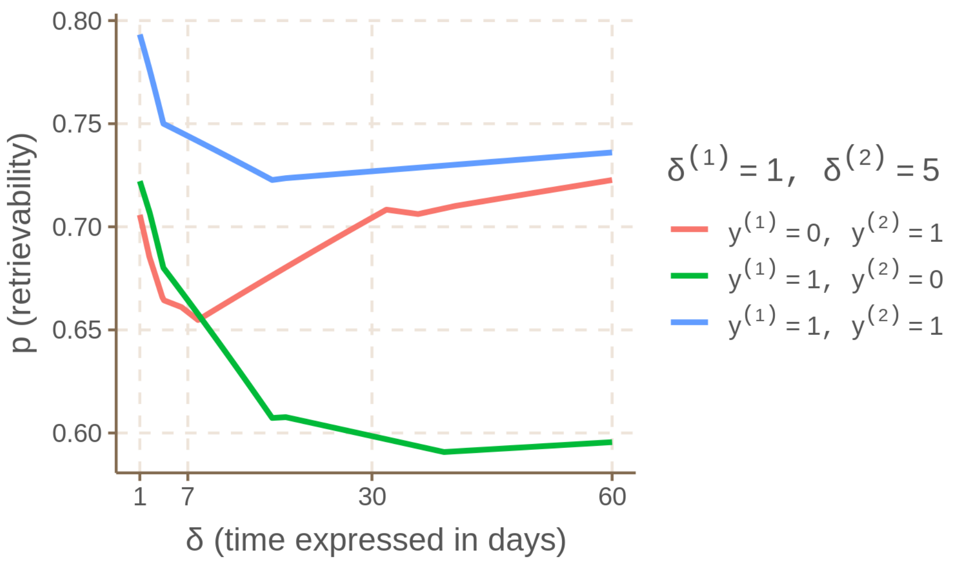


Figure 3.4: Examples of DASH[RNN] forgetting curves. We proceed as in Figure 2.1, with the same student and card.

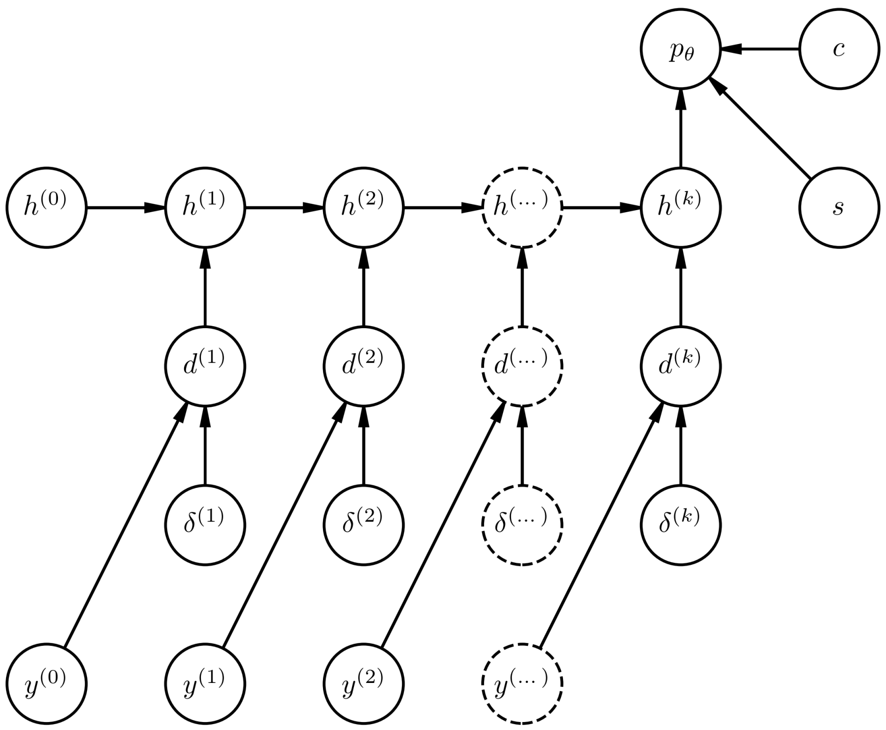


Figure 3.5: The time-unfolded computational graph of the DASH[RNN] memory model that computes retrievability $p_\theta(c, s, (r^{(1)}, \dots, r^{(k-1)}), \delta)$ for the k-th review in the history. At each review step i the inputs are $\delta^{(i)}$ and $y^{(i-1)}$, $\delta^{(i)}$ is the time elapsed 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 3.15. The calculation is reported in detail in Equations 3.12 to 3.15.

4 | Comparison of memory models

In this section we compare the memory models introduced in Chapter 3 of this thesis with the state of the art reported in Chapter 2.

The models we compare are DASH[RNN], R-17, DASH, DASH[MCM], DASH[ACT-R], HLR and IRT. We exclude DAS3H since it requires cards to be tagged by skill, we don't have that information in the datasets we consider. We exclude SuperMemo Algorithm SM-17 since a description accurate enough to replicate it is not publicly available. In Figure 4.1 we show a comparison of the forgetting curves obtained from the memory models.

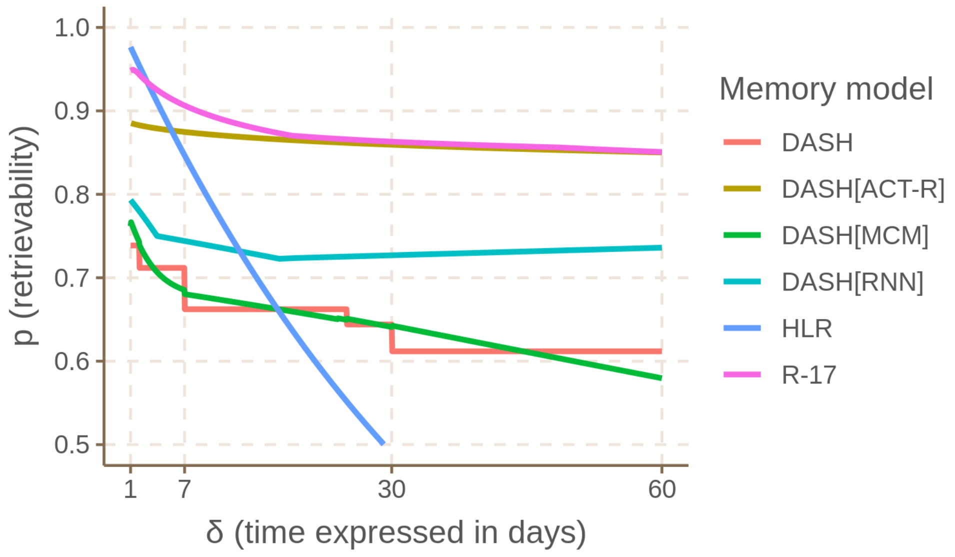


Figure 4.1: Examples of forgetting curves for the memory models compared in this section. We set the same card and student as in Figure 2.1, then consider a review history $r^{(1)}, r^{(2)}$ with $\delta^{(1)} = 1$, $\delta^{(2)} = 5$, and $y^{(1)} = y^{(2)} = 1$. Finally we plot retrievability prediction $\hat{p}_\theta$ from each of the models as a function of time δ elapsed since the last of the two reviews in the history.

In Sections 4.3, 4.4 we perform the comparison on two different datasets. They provide valuable variety: in the nature of the knowledge tested by the cards and in the schedules the students followed. The comparison is predictive in nature. The results depend on the data we use for the evaluation of the models. The Swift dataset from Section 4.3 was collected in the context of learning how to drive a car and the IEIM experiment from Section 4.4 in the context of an undergraduate computer science classroom. The cards of the Swift dataset are not publicly available, but we can posit that they mostly test descriptive knowledge. The knowledge taught in the IEIM course is mostly procedural. Moreover, as mentioned in Section 2.2, spaced repetition systems suffer from a *chicken or the egg* problem: the memory model is fitted to the data, but the collected data is biased by the memory model. The schedule used in the IEIM experiment follows a variation of the SuperMemo Algorithm SM-2 [46], Swift data was collected from an experiment in which different schedules were compared. The heterogeneity of the considered schedules alleviates the bias to some extent. Finally, we want to remark how both dataset were collected in real-world practical deployments of spaced repetition systems. The results can therefore be referenced in the development of future systems.

We start by introducing the metrics for the comparison in Section 4.1, we describe how each of the models was fitted in Section 4.2, we report the results on the Swift dataset in Section 4.3, we present the IEIM dataset and report the corresponding results in Section 4.4 and finally we briefly discuss the results in Section 4.5.

4.1. Metrics for evaluating and comparing memory models

We compare predictions of the memory models based on three metrics that are described below: *area under the ROC (AUC)*, *integrated calibration index (ICI)* and E_{max}. The state of the art for memory models lacks consensus in the choice of metrics [30], after considering possible options we settled on a choice of metrics that we now justify.

We are interested in making predictions. As discussed in Section 2.1, for the practical purposes of a spaced repetition system, 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. We need to predict the outcome of a review that is not part of our historical set of data. For this reason, we decided to compare the models in terms of their predictive power, based on how accurately they can predict the retrievability of new reviews.

We employ three metrics: AUC, ICI and $\mathrm{E}_{\mathrm{max}}$ [1]. AUC is a measure of discrimination, ICI and $\mathrm{E}_{\mathrm{max}}$ of calibration of the probabilistic binary classifier. Good discrimination means that successful reviews have higher predicted probability compared to unsuccessful reviews. Good calibration means that the predicted probabilities match the observed frequency [1, 5, 18, 20, 26, 42], e.g. if the classifier predicts a 40% chance of recall for 10 reviews, about 4 out of 10 should be successful. AUC alone does not suffice. If you all predicted probabilities are halved the AUC value does not change [30].

Both ICI and $\mathrm{E}_{\max}$ are based on the concept of a 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. Following [18], we employ *loess* regression; in particular, we resort to the R function `loess` with the number of iterations set to 1 to disable outlier detection. See Figure 4.3 for examples of calibration curves. The plot of the calibration curve allows us to graphically examine the calibration of a memory model. In a perfectly calibrated classifier, the curve would match the identity line, deviations from it indicate lack of calibration. ICI and $\mathrm{E}_{\max}$ are point estimates that summarize the calibration curve. Let $p \in (0,1)$ be a predicted probability, $c(p)$ the value of the calibration curve at p and $\phi(p)$ the density function of the distribution of predicted probabilities. ICI and $\mathrm{E}_{\max}$ are defined as [1]:

$$\mathrm{ICI} = \int_0^1 |c(p) - p|\phi(p)dp \tag{4.1}$$

$$\mathrm{E}_{\max} = \max_{p\in(0,1)} |c(p) - p|, \tag{4.2}$$

ICI is the mean distance of the calibration curve from the identity line weighted by the distribution of the predicted probabilities; denser areas of the unit interval are given greater weight. $\mathrm{E}_{\max}$ is the largest deviation of the calibration curve from the identity line.

In the memory models comparison we report, for each model, the score in the three metrics and, if applicable, a plot of the calibration curve (accompanied by the smoothed estimated density of the predicted probabilities, see Figure 4.3, obtained with the `geom_density` function of the `ggplot2` R package).

Other metrics could be considered, for instance, the Brier score or the log-likelihood. We decided to focus on just three to keep things simple and because together they capture different but important aspects of memory models as probabilistic binary classifiers [5, 30].

4.2. Model fitting

The compared memory models are fitted either using the Keras framework [10] in the Python programming language or using the R programming language.

We train the HLR model (Section 2.3.3) with gradient descent, as in the original paper [36]. In order to do that we cast the model as a custom layer in the Keras framework [10]. We employ the Adam [16] optimizer with a learning rate of 10^{-3}, we train for 100 epochs with batch size 128. In Equation 2.15 we set the weights penalization hyperparameter $\lambda = 0.1$.

We follow [4] to fit the 1PL-IRT model (Section 2.3.1) using the R programming language. For the comparison of Section 4.3, given the large amount of data and following [12], we use the function `bam` of the package `mgcv`, replacing `lmer` in [4]. Predictions account for random effects. Validation or test data might include cards and students that are not present in the training data. In the case of a new card, we predict with the overall retention rate of the training data; in the case of a new student, we set the random effect to 0.

The same considerations are valid for the DASH and DASH[MCM] models (Section 2.3.2), in case of a card not present in the training data, we set the corresponding fixed effect to 0. We cannot employ the same strategy for DASH[ACT-R], given the non-linearity in h_θ. We cast h_θ as a custom layer in the Keras framework and the DASH Equation 2.9 as a dense layer with logistic (sigmoid) activation and univariate output (as for DASH[RNN] in Section 3.2). We obtain an implementation of the model that we can train end-to-end with gradient descent, computing the gradient with the back-propagation algorithm [16]. In particular we estimate the parameters by maximizing the log-likelihood, to the loss we add a L2 regularization for the parameters of the final dense layer with hyperparameter $\lambda = 10^{-4}$. We employ the Adam [16] optimizer with a learning rate of 10^{-3}, we train for 50 epochs with batch size 128.

We implement the R-17 model (Section 3.1) in the Keras framework as a custom RNN cell, along with a dense layer for σ_e and a custom layer for the Wickelgren power law. We train for 1000 epochs in the comparison of Section 4.3, 100 epochs in the comparison of Section 4.4.1. In Adam optimizer, we clip the gradient norm at 1 and use a cyclical learning rate [38] with an initial learning rate $3 \cdot 10^{-3}$; a maximum learning rate 10^{-2}; a scheduling function $1/2^{x-1}$, where x is the cycle index; and a cycle step size of 8 times the number of batches in each epoch. We initialize the output bias b_p of the neural network P to $\sigma^{-1}(p^*)$ where p^* is the proportion of successful reviews in the training set.

We similarly train the DASH[RNN] model (Section 3.2), the differences are in the number of epochs, 50 epochs in the comparison of Section 4.3 and 100 epochs in the comparison of Section 4.4.1; in the gradient norm clip of 0.5 and in the initial learning rate of $3 \cdot 10^{-4}$ for the cyclical learning rate of the Adam optimizer.

4.3. Results on Swift data

We run the comparison on a large open source dataset [41] collected from a popular German smart driving-learning app by Swift (`swift.ch`). We refer to such dataset as *Swift dataset*. The data consists in review histories for learners studying for the written portion of the driver's permit in Germany, collected as part of an experiment to compare different spaced repetition review schedulers [41]. After preprocessing, we retrained and evaluated the models on 4 different sub-samples of the data and obtained the results reported in Table 4.3 and Figure 4.3. We now describe the process.

First, we preprocessed the data:

1. Following [41] we removed students who where assigned to different spaced repetition schedules upon re-installation of the app. We also removed incomplete review histories.
2. We discretized the elapsed time between reviews in days. The boundary between a day and the next was set at 4 a.m. as an attempt to capture the sleeping patterns of students, to prevent late night reviews to be incorrectly placed in the next day. The setting is inspired by Anki, a popular spaced repetition system (`https://docs.ankiweb.net/preferences.html`).
3. Focusing a single review history (a student-card pair), for each day we kept a single review. We kept the first review of the day but we changed its rating: if any of the reviews of that day was unsuccessful we placed a rating of 0, otherwise of 1. That's because we are focusing on long-term memory.
4. The first review in a review history is the point at which the student is introduced the card, memory does not play a role, and so we get rid of it.

Finally each review history (identified by a student-card pair) contains at most one review per day, moreover it starts at the first true review, when the card has been already been introduced to the student. An example of the data might look like Table 4.1.

After preprocessing we end up with $1,403,022$ review histories containing a total of $2,423,803$ unique reviews, with a proportion of successful reviews of 78.9%. Most review histories contain a single review, we plot the review history length distribution in Figure 4.2.

Table 4.1: Example of what the data looks like after preprocessing. As discussed in Section 2.1, each student-card pair identifies a review history, y is the rating and δ is the time interval discretized time interval elapsed since the last review or since the student was introduced the card.

y	δ	*student ID*	*card ID*
1	1	student1	card1
0	5	student1	card1
1	1	student1	card1
1	1	student1	card2
1	4	student1	card2

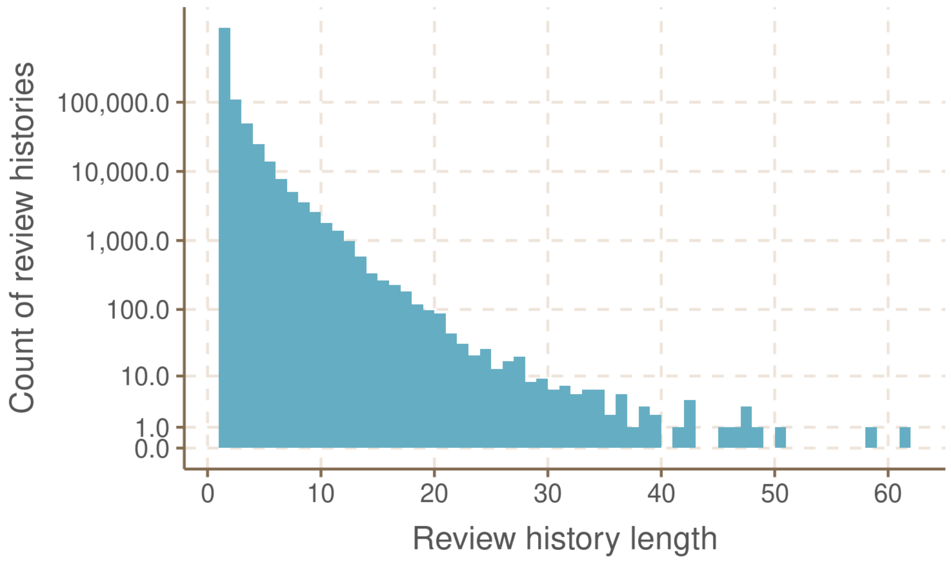


Figure 4.2: Distribution of review histories length in the full Swift dataset.

For time and computational constraints we could not work with such a large number of observations. We decided to sample 5 different train-test splits, created in the following way:

- Randomly sample $15,000$ review histories from the complete dataset, with probabilities proportional to the cube of the length of the history. Therefore, we favor longer review histories, which are generally harder to collect than shorter ones;
- Out of the $15,000$ review histories, randomly sample a test set of $5,000$ review histories, following a uniform distribution. The remaining $10,000$ review histories

are the training set.

Note that the splitting is performed by review history instead of by unique review [31], since for each retrievability prediction the memory models need access to the entire review history of the card. The first train-test sample was used for tuning the models. At a later point in time, the models were fitted to each of the 4 remaining training sets, and evaluated on the corresponding 4 held out test sets. The samples contain an heterogeneous number of reviews since the review histories might be of different length. We report the figures in Table 4.2.

Table 4.2: Review count for each of the 4 train-test samples employed in the comparison.

	Sample 1	Sample 2	Sample 3	Sample 4
Review count in train set	80,416	80,072	80,646	80,606
Review count in test set	40,510	40,433	39,820	40,052

For each of the test review histories, we proceed chronologically: for each review step k with the fitted model we generate a prediction $\hat{p}^{(k)}$ for the retrievability of the card right before review k, then compare it to the observed rating $y^{(k)}$. The end result is a set of (p, y) pairs, where p is a retrievability prediction and y the corresponding observed binary rating. From it we can compute all the metrics discussed in Section 4.1. In Table 4.3 we report the AUC, ICI and $\mathrm{E}_{\mathrm{max}}$ for each memory model, along with the calibration curves in Figure 4.3.

Table 4.3: AUC, ICI and E_{max} for different memory models, scored on 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 score is shown in bold. DASH[RNN] and R-17 are the models that have been introduced in Chapter 3 of this thesis.

(a) Sample 1

	AUC↑	ICI↓	E_{max}↓
DASH[RNN]	**0.853**	0.013	0.029
R-17	0.844	**0.008**	**0.020**
DASH	0.835	0.018	0.067
DASH[MCM]	0.836	0.018	0.068
DASH[ACT-R]	0.849	0.012	0.203
HLR	0.608	0.146	0.565
IRT	0.785	0.028	0.083

(b) Sample 2

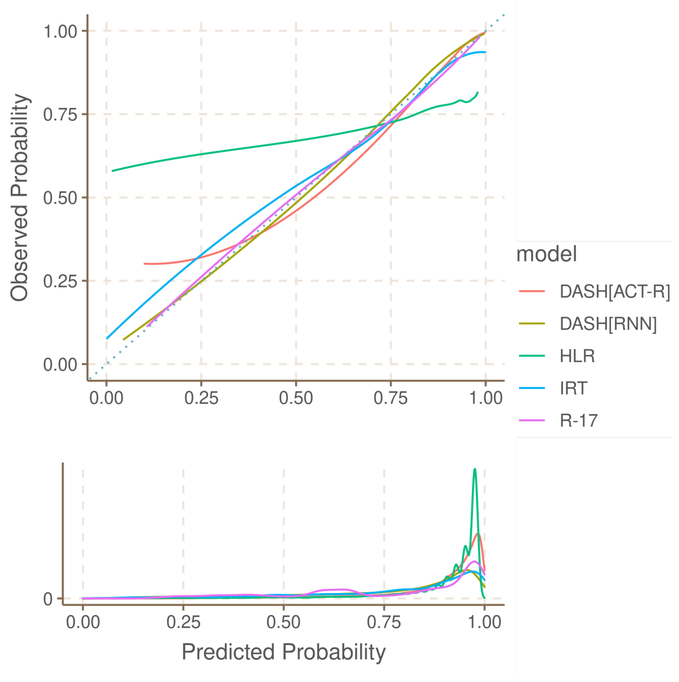
	AUC↑	ICI↓	E_{max}↓
DASH[RNN]	**0.857**	**0.004**	**0.050**
R-17	0.840	0.021	0.061
DASH	0.826	0.023	0.217
DASH[MCM]	0.826	0.023	0.223
DASH[ACT-R]	0.833	0.009	0.211
HLR	0.610	0.146	0.499
IRT	0.776	0.038	0.283

(c) Sample 3

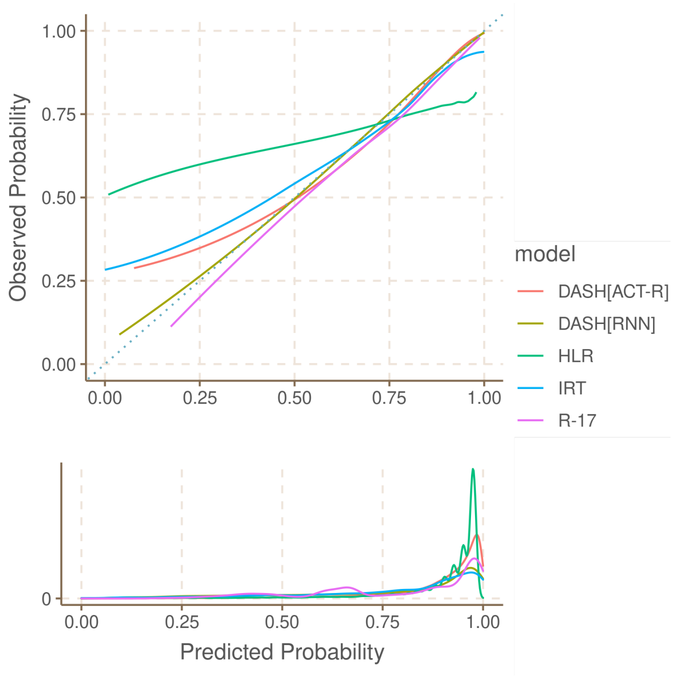
	AUC↑	ICI↓	E_{max}↓
DASH[RNN]	**0.861**	**0.004**	0.027
R-17	0.842	0.005	**0.012**
DASH	0.831	0.016	0.074
DASH[MCM]	0.831	0.017	0.076
DASH[ACT-R]	0.817	0.011	0.240
HLR	0.610	0.144	0.476
IRT	0.777	0.033	0.096

(d) Sample 4

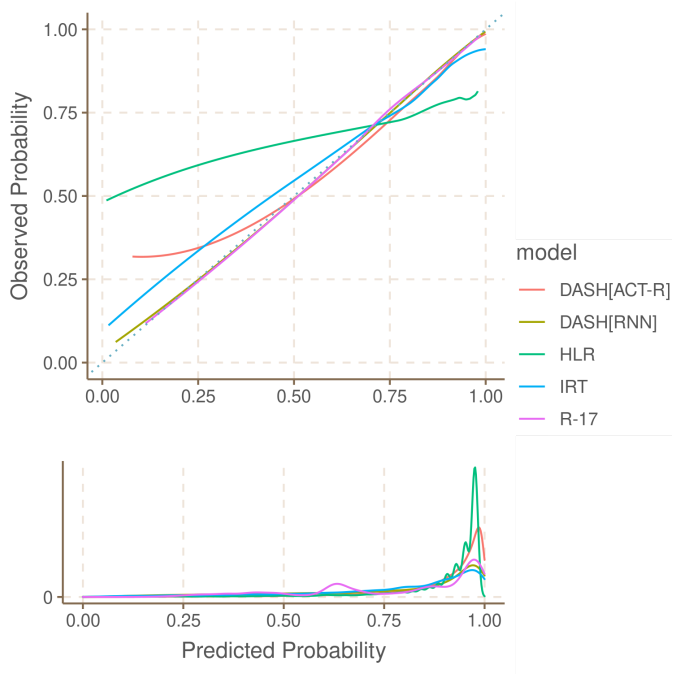
	AUC↑	ICI↓	E_{max}↓
DASH[RNN]	**0.861**	**0.008**	0.071
R-17	0.839	0.022	**0.055**
DASH	0.822	0.026	0.175
DASH[MCM]	0.822	0.026	0.176
DASH[ACT-R]	0.834	0.014	0.152
HLR	0.612	0.144	0.495
IRT	0.772	0.036	0.210



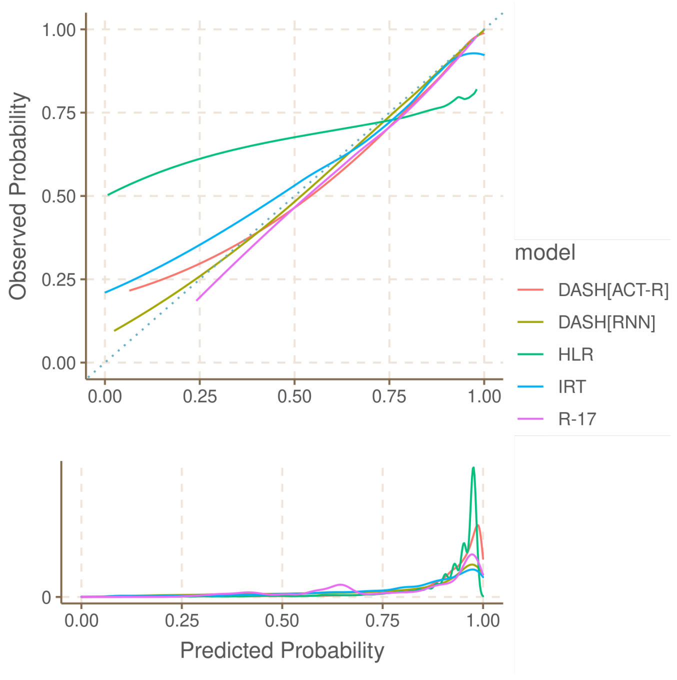
(a) Sample 1



(b) Sample 2



(c) Sample 3



(d) Sample 4

Figure 4.3: Calibration curves (above) and density of the distribution of the predicted probabilities (below) for different memory models, scored on 4 different train-test samples from the Swift dataset. In a perfectly calibrated memory model the curve would match the identity line. DASH[RNN] and R-17 are the models that have been introduced in Chapter 3 of this thesis. 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 4.3.

4.4. IEIM experiment and results on the collected data

In 2021 we conducted an experiment at the *informatica e elementi di informatica medica (IEIM)* course by Prof. Santambrogio at Politecnico di Milano. After each week of the course, we created cards on the topics covered, and the students could opt in to review those cards following a spaced repetition practice in a web application described in Section 1.1. The data collected has then been used to compare several memory models; the results are reported in Section 4.4.1.

In the experiment we created 8 decks, one for each week of the course, for a total of 146 cards (average of 18 cards per deck). 74 students signed up, 58 reviewed at least one card, 24 reviewed at least one card in each of the 8 decks. In Figure 4.4 we display the count of students per number of decks studied.

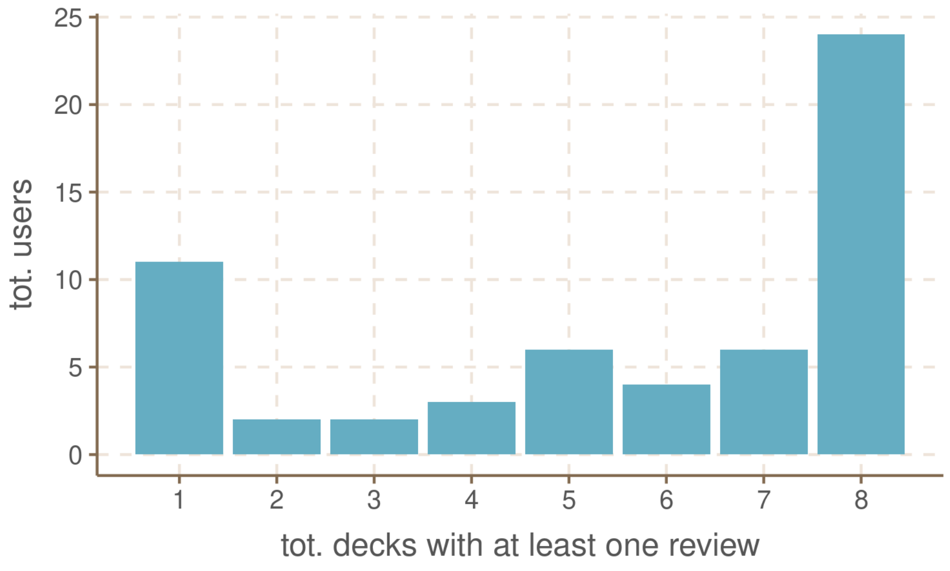


Figure 4.4: On the y-axis we find the total number of students that reviewed at least one card in x decks.

The review scheduler we employed is a variation of SuperMemo Algorithm SM-2 [46] inspired by Anki's implementation (`https://docs.ankiweb.net`). We registered a total of 16,189 unique reviews by the date of the last exam of the 2021 summer session. A couple of students reviewed their cards more than 800 times. In Figure 4.5 we display the review count distribution between students.

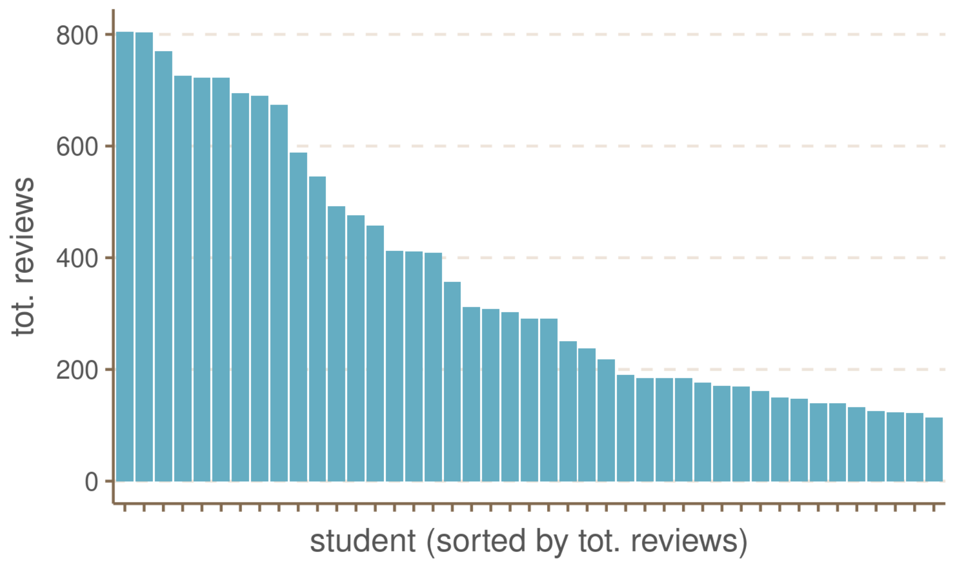


Figure 4.5: Review count per student, each bar corresponds to a student, students have been sorted in descending order by their review count.

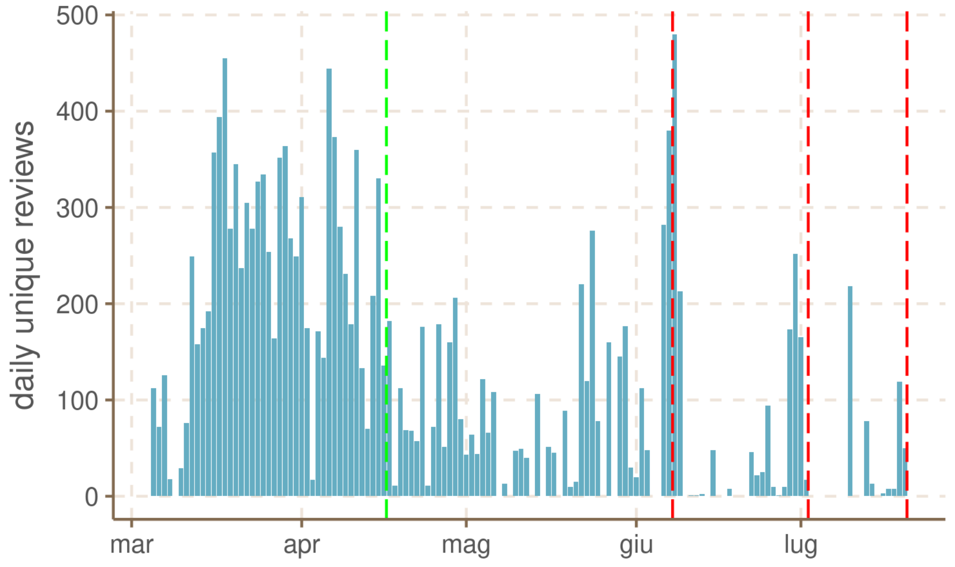


Figure 4.6: Total number of unique reviews for each day between the start of the experiment and the last exam of the summer session. The green vertical line denotes the date of the last lecture of the part of the course covered by the cards. The red vertical lines denote the dates of the exams.

In Figure 4.6 we report the amount of reviews over time, highlighting the dates of the exams. The cards cover material presented from February 22nd to April 16th 2021. There have been exams in June 7th, July 2nd, and July 20th 2021. The students had to choose one of the three dates in which to be tested. Upon failure, they could retry once again at a future date. We notice spikes in activity right before each exam. It suggests that students used the tool at their disposal also to check their knowledge of the course material, not only as a review tool. The spikes decrease in volume; we posit that the effect is due to the decreasing number of students left to be tested in subsequent exams.

In the remaining part of the section we analyze the retention rate. In particular we try to answer the following questions: are there students with a better retention rate than others? Were there decks/notes that were harder than others? In other words, can we identify the easier/harder concepts? We restrict the analysis to students with more than 100 unique reviews. In Figure 4.7 we plot the average retention rate for each student, notice that the values lie between 0.7 and 1. Identifying students that display better retention can be beneficial for their experience with the system. In particular, in designing a review scheduler we might increase the length of the intervals between reviews for students with a large average retention rate and shorten it for students that fail to recall more cards on average.

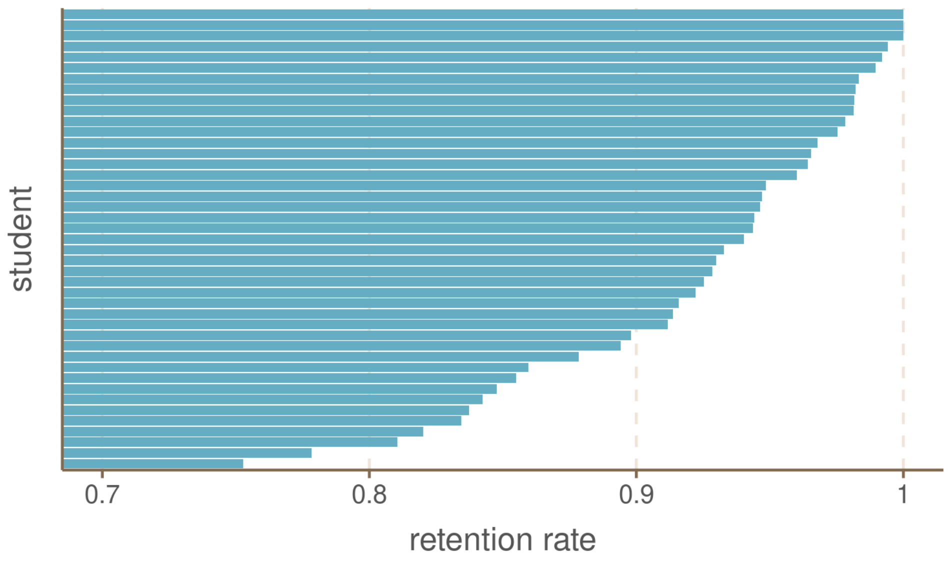


Figure 4.7: Retention rate per student. Each bar corresponds to a unique student that performed more than 100 reviews. The retention rate is computed as the average rating across all cards.

We now focus on how the retention rate varies between decks. The average retention rate for a deck is calculated by averaging the rating of the reviews of the cards within the deck across all students. We performed a statistical test to determine the significance of the difference in retention rate between decks. Similarly to the IRT model presented in

Section 2.3.1 we employ a Linear Mixed Model with the deck indicator variable as fixed effect and the student indicator variable as random effect. The reason the student has been considered a random effect is that we care about the decks under study, but not about the specific students that participated in the experiment. We consider the students as a random sample from a wider population of students. This allows us to account for the dependency between decks induced by students studying different decks. After fitting the complete model, we performed a statistical test for the significance of the covariate of the fixed effect using a parametric bootstrap approach, as illustrated in [17]. We fit the model on 10,000 bootstrap samples and obtain a p-value of 0.0001. This suggests that we can reject the null hypothesis, suggesting a significant difference in retention rate between decks.

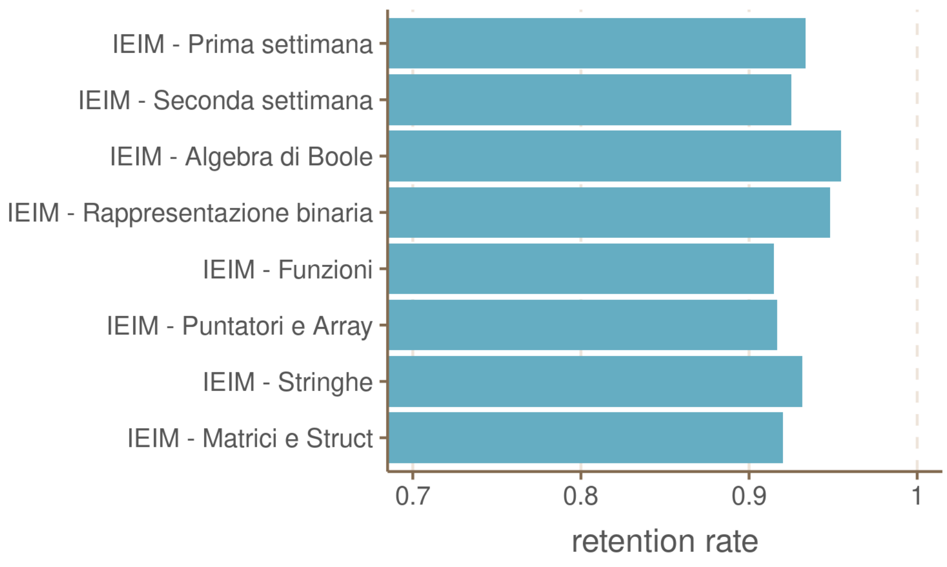


Figure 4.8: Retention rate per deck.

We repeat for cards a retention rate analysis similar to the one for decks. More precisely we consider two fixed effects: decks and cards; students are still considered random effects. Basically we are checking the significance of the card effect, while keeping into account the dependency induced by decks and students. We test the significance of the card fixed effect by the same parametric bootstrap discussed above. We are comparing the full model to the model without the card random effect. We run the test with 10,000 bootstrap samples and obtain a p-value of 0.0001. We sort the cards by retention rate. The result we obtain might be interesting for the professor designing a course, we can interpret the ranking of cards as a ranking by difficulty of the concepts presented in the course. The interpretation should be taken with a grain of salt: if students have a hard time recalling a concept it doesn't necessarily mean that it's a difficult one, moreover quality in the way the card was written is an important confounding factor. The way a card is written can make it easier or harder to recall, in the experiment we tried to follow the guidelines of [22, 25]. The top-5 easiest and top-5 hardest cards are displayed in Table 4.4.

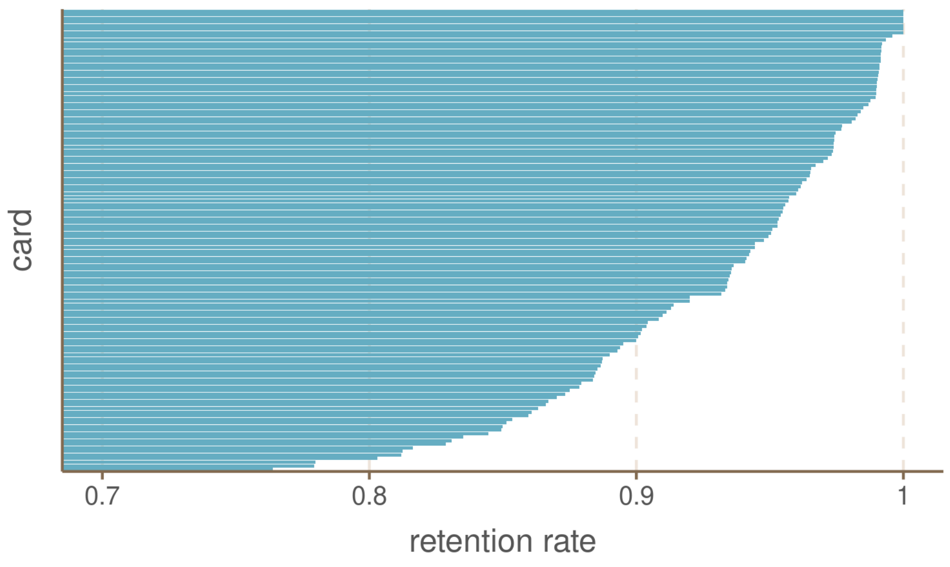


Figure 4.9: Retention rate per card.

Table 4.4: Easiest and hardest cards created in the IEIM experiment, ranked by retention rate. Instead of displaying the full question and answer for each of the cards, we instead opted for displaying a brief label for the tested concept of the course under the *tested concept* column.

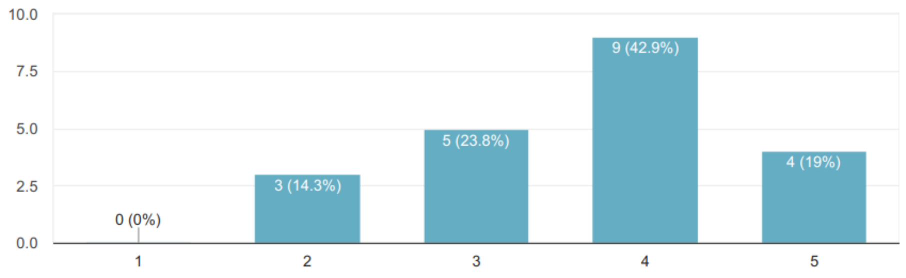
(a) Top-5 hardest cards.

card id	tested concept	retention rate	review count
VUOFvO6zZ2a3790HFovZ	General notions on algorithms and programming languages	0.764	161
mTBm7W5rHU3VCOijOVHq	Kilo, Mega, Giga as powers of 2	0.779	145
IMBTylkxsFVm7FLX7Inv	*scanf* and *printf*	0.780	159
xt0gyigZ30NbXZ2N2mVW	Declaration and definition of functions	0.803	137
jCj7VyrajcdCCt40xMz8	*scanf* and *printf*	0.812	149

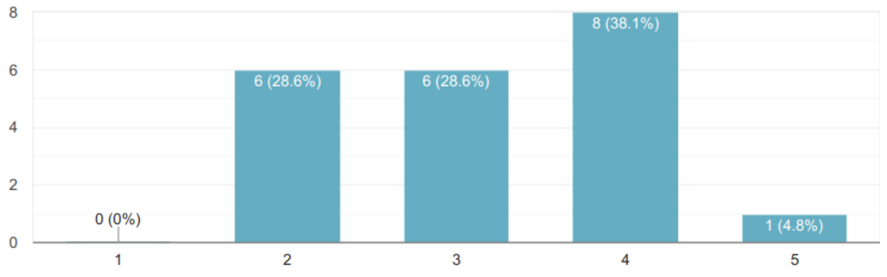
(b) Top-5 easiest cards.

card id	tested concept	retention rate	review count
brhF4rpIQ1Eo4gHMMQIA	Definition of pointer in C	1.000	80
J7To4cbfF1HdPpbu8hWE	Matrices in C	1.000	64
KJ4mt9zu5R2c7EDV4KRi	Iterative constructs	1.000	108
oP2AIioY4aN94RPQpv0S	Definition of array in C	1.000	79
SAjrvWoolvijcWNkW8nX	Definition of string in C	1.000	148

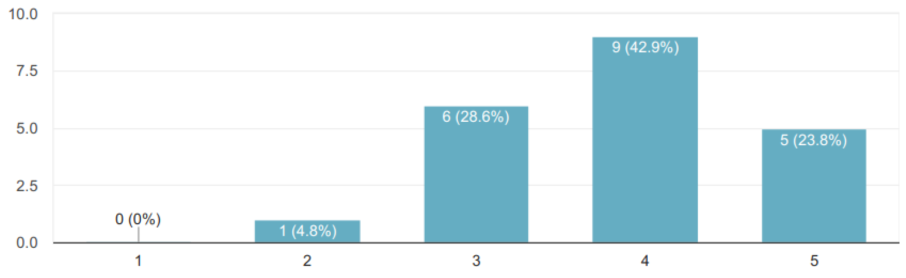
At the end of the semester, we sent a survey to the students who signed up for the experiment. Students were asked some questions to answer with a grade on the Likert scale. In Figure 4.10 we report the results.



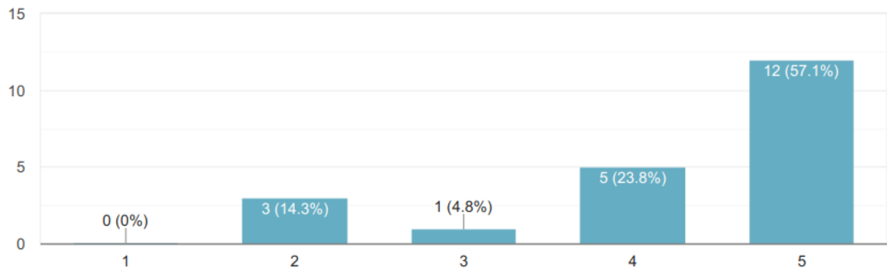
(a) Do you think Rember helped you study in less time?



(b) Do you think Rember helped you get a better grade?



(c) Without studying every time, it can become difficult to follow the new lessons. Do you think that Rember has helped you to keep up, allowing to follow the new lessons more easily?



(d) If it were available, would you use Rember for other classes?

Figure 4.10: Results of student questionnaire at the end of the semester. Each plot reports a summary of the Likert scale responses to the question in the caption under it.

4.4.1. Results on IEIM data

We compare different memory models on the data collected in the IEIM experiment. After preprocessing, we performed a 10-fold cross-validation repeated 50 times and obtained the results reported in Table 4.5. Here is the process we followed.

Preprocessing proceeds as described in Section 4.3, we skip the first step, as it is specific to the Swift dataset. Table 4.1 shows an example of what the data looks like after preprocessing. After preprocessing we end up with $3,700$ review histories containing a total of $9,283$ unique reviews, with a proportion of successful reviews of 97.0%. Given that we are interested in binary ratings, the latter figure means that the dataset is effectively smaller than it appears. A dummy memory model always predicting a large retrievability would do quite a good job; we need to focus on the few (274, out of 16,189) unsuccessful reviews in order to make better and sensible predictions.

To compensate for the small data problem, we employ a 10-fold cross-validation scheme repeated 50 times, for a total of 500 folds, following the recommendations in [18]. Each fold consists of the full dataset divided into a training set and a test set, containing, respectively, 90% and 10% of the review histories. The evaluation proceeds as described in Section 4.3.

In Table 4.5 for each memory model we are comparing we report aggregated scores for AUC, ICI and $\mathrm{E}_{\mathrm{max}}$ across the 500 folds, in particular we report the sample mean and sample standard deviation across all folds.

Table 4.5: Sample mean and sample standard deviations (in parentheses) of AUC, ICI and $\mathrm{E}_{\mathrm{max}}$ for different memory models, 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. DASH[RNN] and R-17 are the models that have been introduced in Chapter 3 of this thesis.

	AUC↑	ICI↓	$\mathrm{E}_{\mathrm{max}}$↓
DASH[RNN]	**0.842** (0.040)	0.012 (0.004)	0.267 (0.168)
R-17	0.665 (0.064)	0.011 (0.004)	0.082 (0.092)
DASH	0.770 (0.067)	0.018 (0.006)	0.338 (0.186)
DASH[MCM]	0.771 (0.067)	0.019 (0.006)	0.357 (0.195)
DASH[ACT-R]	0.768 (0.136)	**0.010** (0.013)	**0.070** (0.067)
HLR	0.548 (0.063)	0.125 (0.006)	0.600 (0.042)
IRT	0.765 (0.068)	0.018 (0.005)	0.273 (0.148)

4.5. Discussion of the results

Overall DASH[RNN] and R-17, the two novel models introduced in this thesis in Chapter 3 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 the development of spaced repetition systems at scale. R-17 performs comparatively well, this result hints that SM-17 deserves to be studied with more care. The goals we had in mind for the two memory models are achieved. The results are not as clear in the very small IEIM dataset.

On the larger Swift dataset, the DASH[RNN] model outperforms the state of the art in all three metrics (Table 4.3). The only exception is the ICI score in Sample 1, where DASH[ACT-R] obtains a slightly lower score. In all other samples, the smallest ICI score of the state of the art is reduced by almost 50% by DASH[RNN]. In all samples, the smallest E_{max} score of the state of the art is reduced by more than 50% by DASH[RNN]. In the IEIM data set, it is less clear which model performs best (Table 4.5). The DASH[RNN] model obtains by far the largest AUC score, but the one with better scores on the calibration metrics is DASH[ACT-R]. This is probably due to the smaller number of parameters in DASH[ACT-R] compared to the other models. The small sample size, combined with the large (97%) proportion of correct responses, makes it difficult to train larger models on the IEIM data. Each training sample from the Swift dataset contains about an order of magnitude more reviews compared to one of the IEIM dataset. We argue that, for what concerns the development of spaced repetition systems at scale, the results on the larger Swift dataset are more significant, unless one is bootstrapping a new spaced repetition system from a small number of students and cards in a short time. In that case, there are valid alternatives to training a memory model on a small dataset, such as using tried-and-tested heuristics as a review scheduler. The SuperMemo Algorithm SM-2 variant used by Anki schedules reviews for millions of students every day, and a complete description is available in Anki's documentation (`https://docs.ankiweb.net/`), a similar review scheduler was used for the IEIM experiment described in Section 4.4.

R-17 does not always outperform the state of the art in the Swift dataset. It obtains a larger AUC score in samples 2 to 4 compared to the state of the art (it comes very close in sample 1). Reduces ICI by less than 2/3 in samples 1 and 3, but increases it in samples 2 and 4. It reduced E_{max} almost by less than 1/3 in all samples. The calibration curves (Figure 4.3) of DASH[RNN] and R-17 are visually more convincing than those of the state of the art, for small retrievability values DASH[ACT-R] starkly deviates from the identity line, this is reflected in the large E_{max} scores. On the IEIM dataset, R-17 scores are comparable to those of DASH[ACT-R] but it suffers in the AUC metric.

5 | Conclusions

The primary contribution of this work is two-fold. We introduce two novel adaptive memory models, DASH[RNN] and R-17, the former outperforms the state of the art from a predictive perspective, the latter performs comparably well; in building and comparing the models we construct a framework for developing memory models specifically for spaced repetition systems, that can scale and adapt as the system grows.

R-17 is an attempt to approximate with neural networks the memory model component of SuperMemo Algorithm SM-17. SM-17 is the review scheduler employed by the popular spaced repetition system SuperMemo (`https://www.supermemo.com/`). There exists a public description for SM-17 but it lacks sufficient details to faithfully reimplement the algorithm. That might be the reason why the literature has never (to the best of our knowledge) considered SM-17. We attempt to fill this gap. The significance of the predictive result of R-17, in addition to standing on their own, is that they hint at SM-17 performance.

The thesis has the ambitious goal of serving as a reference framework for thinking about and developing adaptive memory models for spaced repetition systems. In Chapter 1 we introduced spaced repetition systems, breaking them down into fundamental components. We described the distinction between review schedulers and memory models and how both are employed by the latter systems. We introduced some results from psychology that memory models are often based on. In Chapter 2 we introduced a general formalization of adaptive memory models and discussed their inherent and unavoidable limitations. Finally, in Chapter 4 we introduced and justified sensible metrics to evaluate and compare adaptive memory models from a predictive perspective. The framework is far from being complete, our hope is that future work will build on it to enable the development of better adaptive memory models that will expand the horizon of possibilities of spaced repetition systems. Future work could introduce a way to reason about interference when the student reviews several related cards. It could introduce better tools for inspecting and interpreting the behavior of memory models, which might be useful for improving their performance but potentially also for generating insights about the inner workings of human memory. Future work could deepen our understanding of how memory models should behave for knowledge of different nature, complexity, or domain.

Spaced repetition systems have been around for three decades and their effectiveness has been proved over and over. Modern education has not caught up with them yet; our hope is that with better tools and systems this gap can be filled in order to let students learn more effectively and efficiently than what is currently possible. This thesis aims to be a step in this direction.

Bibliography

[1] P. C. Austin and E. 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, Sept. 2019. ISSN 0277-6715, 1097-0258. doi: 10.1002/sim.8281.

[2] R. A. Bjork. Memory and metamemory considerations in the training of human beings. In *Metacognition: Knowing about Knowing*, pages 185–205. The MIT Press, Cambridge, MA, US, 1994. ISBN 978-0-262-13298-5. doi: 10.7551/mitpress/4561.001.0001.

[3] R. A. Bjork, E. L. Bjork, et al. A new theory of disuse and an old theory of stimulus fluctuation. *From learning processes to cognitive processes: Essays in honor of William K. Estes*, 2:35–67, 1992.

[4] P. D. Boeck, M. Bakker, R. Zwitser, M. Nivard, A. Hofman, F. Tuerlinckx, and I. Partchev. The Estimation of Item Response Models with the lmer Function from the **lme4** Package in *R*. *Journal of Statistical Software*, 39(12), 2011. ISSN 1548-7660. doi: 10.18637/jss.v039.i12.

[5] R. Caruana and A. Niculescu-Mizil. Data mining in metric space: An empirical analysis of supervised learning performance criteria. In *Proceedings of the Tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '04, pages 69–78, New York, NY, USA, Aug. 2004. Association for Computing Machinery. ISBN 978-1-58113-888-7. doi: 10.1145/1014052.1014063.

[6] N. J. Cepeda, H. Pashler, E. Vul, J. T. Wixted, and D. Rohrer. Distributed practice in verbal recall tasks: A review and quantitative synthesis. *Psychological Bulletin*, 132(3):354–380, May 2006. ISSN 0033-2909. doi: 10.1037/0033-2909.132.3.354.

[7] N. J. Cepeda, E. Vul, D. Rohrer, J. T. Wixted, and H. Pashler. Spacing effects in learning: A temporal ridgeline of optimal retention. *Psychological Science*, 19(11):1095–1102, Nov. 2008. ISSN 1467-9280. doi: 10.1111/j.1467-9280.2008.02209.x.

[8] B. Choffin, F. Popineau, Y. Bourda, and J.-J. Vie. DAS3H: Modeling Student

Learning and Forgetting for Optimally Scheduling Distributed Practice of Skills. *arXiv:1905.06873 [cs, stat]*, May 2019.

[9] B. Choffin, F. Popineau, and Y. Bourda. Extending Adaptive Spacing Heuristics to Multi-Skill Items. *Journal of Educational Data Mining*, 13(3):69–102, Oct. 2021. ISSN 2157-2100. doi: 10.5281/zenodo.5634220.

[10] F. Chollet et al. Keras, 2015.

[11] M. T. Cicero, E. W. Sutton, and H. Rackham. *De Oratore.* Loeb classical library. Heinemann ; Harvard University Press, London (England) : Canbrudge (Mass.), 1942.

[12] M. Clark. Michael clark: Mixed models for big data, 2019.

[13] H. Ebbinghaus. *Memory; a Contribution to Experimental Psychology.* New York city, Teachers college, Columbia university, 1913.

[14] L. G. Eglington and P. I. Pavlik Jr. Optimizing practice scheduling requires quantitative tracking of individual item performance. *npj Science of Learning*, 5(1):1–10, Oct. 2020. ISSN 2056-7936. doi: 10.1038/s41539-020-00074-4.

[15] L. Gabora. The Making of a Creative Worldview. *arXiv:1811.11236 [q-bio]*, Nov. 2018.

[16] I. Goodfellow, Y. Bengio, and A. Courville. *Deep Learning.* MIT Press, 2016.

[17] U. Halekoh and S. Højsgaard. A Kenward-Roger Approximation and Parametric Bootstrap Methods for Tests in Linear Mixed Models – The R Package pbkrtest. *Journal of Statistical Software*, 59:1–32, Sept. 2014. ISSN 1548-7660. doi: 10.18637/jss.v059.i09.

[18] J. Harrell. *Regression Modeling Strategies: With Applications to Linear Models, Logistic and Ordinal Regression, and Survival Analysis.* Springer Series in Statistics. Springer International Publishing : Imprint: Springer, Cham, 2nd ed. 2015 edition, 2015. ISBN 978-3-319-19425-7. doi: 10.1007/978-3-319-19425-7.

[19] A. Hunziker, Y. Chen, O. Mac Aodha, M. Gomez Rodriguez, A. Krause, P. Perona, Y. Yue, and A. Singla. Teaching Multiple Concepts to a Forgetful Learner. In *Advances in Neural Information Processing Systems 32*, volume 6, pages 4025–4036. Curran, 2020. ISBN 978-1-71380-793-3.

[20] A. Kumar, P. S. Liang, and T. Ma. Verified Uncertainty Calibration. In *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019.

[21] R. Lindsey. *Probabilistic Models of Student Learning and Forgetting.* PhD thesis, University of Colorado at Boulder, 2014.

[22] A. Matuschak. How to write good prompts: Using spaced repetition to create understanding, San Francisco (2020).

[23] M. C. Mozer and R. 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. ISBN 978-1-138-79192-3 978-1-138-79193-0 978-1-315-41357-0.

[24] J. M. J. Murre and J. Dros. Replication and Analysis of Ebbinghaus' Forgetting Curve. *PLoS ONE*, 10(7):e0120644, July 2015. ISSN 1932-6203. doi: 10.1371/journal.pone.0120644.

[25] M. A. Nielsen. Augmenting Long-term Memory. 2018.

[26] J. Nixon, M. W. Dusenberry, L. Zhang, G. Jerfel, and D. Tran. Measuring Calibration in Deep Learning. In *CVPR Workshops*, volume 2, 2019.

[27] J. Niznan, J. Papousek, and R. Pelánek. Exploring the Role of Small Differences in Predictive Accuracy using Simulated Data. In *AIED Workshops*, 2015.

[28] S. C. Pan and T. C. Rickard. Transfer of test-enhanced learning: Meta-analytic review and synthesis. *Psychological Bulletin*, 144(7):710–756, July 2018. ISSN 1939-1455. doi: 10.1037/bul0000151.

[29] H. Pashler, N. Cepeda, R. V. Lindsey, E. Vul, and M. C. Mozer. Predicting the Optimal Spacing of Study: A Multiscale Context Model of Memory. In *Advances in Neural Information Processing Systems*, volume 22. Curran Associates, Inc., 2009.

[30] R. Pelanek. Metrics for Evaluation of Student Models. June 2015. doi: 10.5281/ZENODO.3554665.

[31] S. Reddy, I. Labutov, S. Banerjee, and T. Joachims. Unbounded Human Learning: Optimal Scheduling for Spaced Repetition. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 1815–1824, San Francisco California USA, Aug. 2016. ACM. ISBN 978-1-4503-4232-2. doi: 10.1145/2939672.2939850.

[32] S. Reddy, S. Levine, and A. Dragan. Accelerating human learning with deep reinforcement learning. In *NIPS'17 Workshop: Teaching Machines, Robots, and Humans*, pages 5–9, 2017.

[33] B. A. Richards and P. W. Frankland. The Persistence and Transience of Memory. *Neuron*, 94(6):1071–1084, June 2017. ISSN 1097-4199. doi: 10.1016/j.neuron.2017.04.037.

[34] C. A. Rowland. The effect of testing versus restudy on retention: A meta-analytic review of the testing effect. *Psychological Bulletin*, 140(6):1432–1463, Nov. 2014. ISSN 1939-1455. doi: 10.1037/a0037559.

[35] F. Sense, T. S. Jastrzembski, M. M. Mozer, M. Krusmark, and v. Rijn, Hedderik. Perspectives on Computational Models of Learning and Forgetting: 17th Annual Meeting of the International Conference on Cognitive Modelling. *Proceedings of ICCM 2019*, pages 216–221, 2019. ISSN 978-0-9985082-3-8.

[36] B. Settles and B. 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.

[37] S. Sinha. *Using Deep Reinforcement Learning for Personalizing Review Sessions on E-Learning Platforms with Spaced Repetition.* PhD thesis, 2019.

[38] L. N. Smith. Cyclical learning rates for training neural networks. In *2017 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pages 464–472. IEEE, 2017.

[39] B. Tabibian, U. Upadhyay, A. De, A. Zarezade, B. Schölkopf, and M. Gomez-Rodriguez. Enhancing human learning via spaced repetition optimization. *Proceedings of the National Academy of Sciences*, 116(10):3988–3993, Mar. 2019. ISSN 0027-8424, 1091-6490. doi: 10.1073/pnas.1815156116.

[40] U. Upadhyay, A. De, and M. Gomez-Rodrizuez. Deep reinforcement learning of marked temporal point processes. In *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, NIPS'18, pages 3172–3182, Red Hook, NY, USA, Dec. 2018. Curran Associates Inc.

[41] U. Upadhyay, G. Lancashire, C. Moser, and M. 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, Sept. 2021. ISSN 2056-7936. doi: 10.1038/s41539-021-00105-8.

[42] J. Vaicenavicius, D. Widmann, C. Andersson, F. Lindsten, J. Roll, and T. Schön. Evaluating model calibration in classification. In *Proceedings of the Twenty-Second*

International Conference on Artificial Intelligence and Statistics, pages 3459–3467. PMLR, Apr. 2019.

[43] W. J. van der Linden. *Handbook of Item Response Theory.* 2019. ISBN 978-0-367-22120-1.

[44] J. T. Wixted. On Common Ground: Jost's (1897) law of forgetting and Ribot's (1881) law of retrograde amnesia. *Psychological Review*, 111(4):864–879, Oct. 2004. ISSN 0033-295X. doi: 10.1037/0033-295X.111.4.864.

[45] J. T. Wixted and S. K. Carpenter. The Wickelgren Power Law and the Ebbinghaus Savings Function. *Psychological Science*, 18(2):133–134, Feb. 2007. ISSN 0956-7976. doi: 10.1111/j.1467-9280.2007.01862.x.

[46] P. Wozniak. The true history of spaced repetition. https://www.supermemo.com/en/articles/history, June 2018.

[47] P. A. Wozniak. Algorithm SM-17 - supermemo.guru. https://supermemo.guru/wiki/Algorithm_SM-17, .

[48] P. A. Wozniak. Algorithm SM-18 - supermemo.guru. https://supermemo.guru/wiki/Algorithm_SM-18, .

[49] P. A. Wozniak. *Optimization of Learning.* PhD thesis, Technical University of Poznan, Computer Science Center, 1990.

[50] P. A. Wozniak, E. J. Gorzelańczyk, and J. A. Murakowski. Building memory stability through rehearsal. http://www.super-memory.com/articles/stability.htm.

[51] P. A. Wozniak, E. J. Gorzelańczyk, and J. A. Murakowski. Two components of long-term memory. *Acta Neurobiologiae Experimentalis*, 55(4):301–305, 1995. ISSN 0065-1400.

[52] Z. Yang, J. Shen, Y. Liu, Y. Yang, W. Zhang, and Y. Yu. TADS: Learning Time-Aware Scheduling Policy with Dyna-Style Planning for Spaced Repetition. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 1917–1920, Virtual Event China, July 2020. ACM. ISBN 978-1-4503-8016-4. doi: 10.1145/3397271.3401316.

List of Figures

List of Tables

Acknowledgements

Ringrazio Marco D. Santambrogio per il suo aiuto e le importanti opportunità che mi ha offerto in ambito accademico e lavorativo e nella crescita personale. Ringrazio Andrea Vaccari per la pazienza e i numerosi insegnamenti. Ringrazio gli amici che hanno allietato questi anni universitari. Infine, un ringraziamento va a i miei genitori Brunella e Filippo, a mia sorella Maria e il resto della mia famiglia, che mi hanno supportato in tutti questi anni. Dedico questa tesi a nonna Margaret.