Half-life Regression - An Investigation of the Forgetting Curve

Aimee Co and Haram Yoon

[1]MA179 - Mathematics of Big Data at Harvey Mudd College

acpy2022@mymail.pomona.edu, hyhn2022@mymail.pomona.edu

Abstract. *In this paper, we investigate the half-life regression (HLR), an algorithm to represent spaced repetition practice for second language acquisition. HLR was first introduced by Settles and Meeder in 2016 [1], making it a relatively new machine learning model. Data from 13 million Duolingo learning sessions was used to train and test the model. We successfully replicated the results of Settles and Meeder, finding that the HLR model with lexeme tags performed the best in accuracy of prediction.*

1. Introduction

The forgetting curve was first documented by Ebbinghaus in 1885, where he found that his memory of nonsense syllables decays exponentially following rehearsal [2]. This decay decreases with spaced repetition, as seen in Figure 1. This observation is coupled with the lag effect [3], which states that memory increases even more if the spacing between repetitions gradually increases. Spaced repetition is the technique of reviewing material at increasing intervals in order to embed it securely in long-term memory. This contrasts with strategies like mass repetition or cramming, which lead to short-term retention.

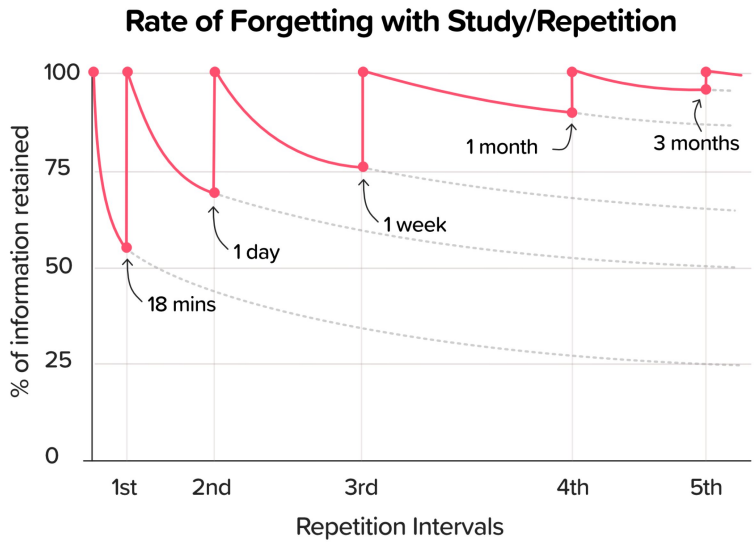


Figure 1. Graphical representation of the forgetting curve with spaced repetition

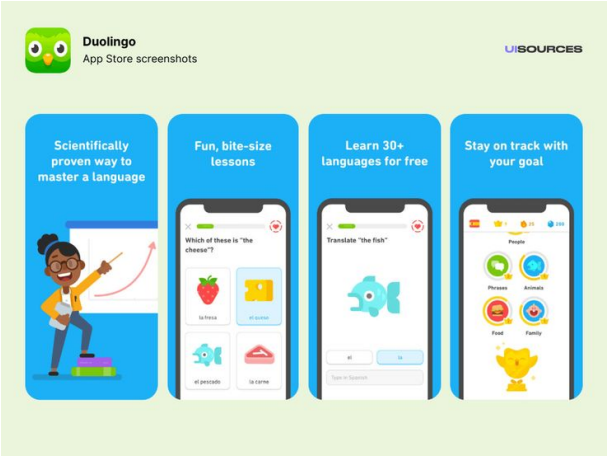
This is especially applicable to second language acquisition, where rehearsal of words and phrases is necessary to gain a fundamental understanding of the language.

Algorithms that intend to model spaced repetition were developed between the 1960's and 1980's, and thus generally involve few and simple parameters. The increasing popularity of online language learning software now allows for the collection of large-scale student data, which can be used to train machine learning models. We intend to harness such data to develop and train a machine learning algorithm based on the forgetting curve that can predict if a student will remember or not remember a word.

2. Duolingo

Duolingo is a online language learning platform, which garnered over 150 million students since 2012. Developed by engineers Luis von Ahn and Severin Hacker [5], Duolingo's mission is to provide free access to high-quality language education for everyone. The company's name is derived from the Spanish words "duo" meaning two, and "lingo" referring to a language or tongue.

At its core, Duolingo aims to break down barriers to language learning through its engaging, gamified approach. The platform currently offers courses in over 40 different languages taught through bite-sized lessons that feel more like playing a game than traditional rote study. The most popular language courses are English, Spanish, French, and German, and the platform provides many interface languages as well.



Duolingo integrates the Leitner Model, a form of spaced repetition to determine if the student requires practice on specific words or skills based on their history of recalling it correctly or incorrectly, and if they have practiced it recently. The algorithm adapts the spaced repetition intervals dynamically for each individual user based on their areas of strength and weakness. Progress is represented by a skill tree, where each skill has a progress bar that updates with practice according to the Leitner algorithm.

3. Spaced Repetition Models

3.1. Pimsleur Method

The Pimsleur method was developed in 1967 as a program for audio-based language learning [4]. Words would be rehearsed in gradually increasing intervals, with the introduction of new words interspersed between. However, this model does not take into account the student's recall ability, as words that the student recalls easily would be practiced at the same frequency as words that the student tends to not recall.

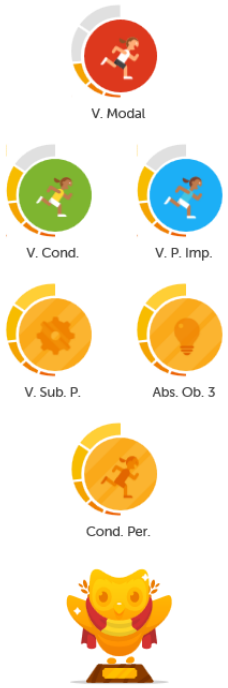


Figure 2. Duolingo skill tree with progress bars

3.2. Leitner System

The Leitner system is another spaced repetition framework [6]. Considering n queues $\{q_0, q_1, \ldots, q_{n-1}\}$ where all instances are placed in the first queue q_0 to begin with. Following correct recall by the learner, an instance originally in q_i is promoted to q_{i+1}. If the instance is incorrectly classified, then it is demoted to q_{i-1}. Thus, the Leitner model is a more sophisticated model compared to the Pimsleur model, which only represents positive engagement.

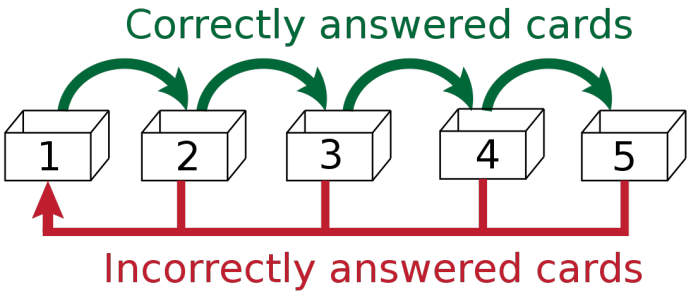


Figure 3. Leitner model showing promotion and demotion corresponding to correct and incorrect recall

3.3. Half-life Regression

The half-life regression combines the forgetting curve with machine learning techniques.

Half-life is denoted as:

$$\hat{h}_\theta = 2^{\theta \cdot x} \tag{1}$$

where x is a feature vector summarizing the student's past exposure to the word, and θ is a vector containing weights corresponding to x.
The probability of correctly recalling a word can be represented as

$$p = 2^{-\frac{\Delta}{h}} \tag{2}$$

Δ is the lag time since the word was last practiced in minutes. h is the half-life of the learner's long term memory.

$$\Delta = 0 \rightarrow p = 2^0 = 1$$
$$\Delta = h \rightarrow p = 2^{-1} = 0.5$$
$$\Delta \gg h \rightarrow p \approx 0$$

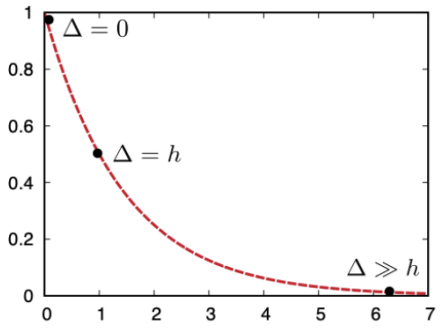


Figure 4. Ebbinghaus model with $h = 1$

The Pimsleur and Leitner models can be represented as special cases of (2) with fixed calculated weights. To calculate the weights for the Pimsleur model, we consider the original practice schedule used by Pimsleur: 5 sec, 25 sec, 2 min, 10 min, 1 hr, . . . , 4 mo, 2 yr. Interpreting the sequence as half-lives $\hat{h}_\theta$, we can rewrite (2) and solve for $\log_2(\hat{h}_\theta)$ as a linear equation. We get $\theta = \{x_n = 2.4, x_b = -16.5\}$ where x_n is the number of practices and x_b is the bias weight. Calculating the weights for the Leitner model simply involves the promotion/demotion system, yielding $\theta = \{x_\oplus = 1, x_\ominus = -1\}$ where $x_\oplus$ represents the number of past correct responses and $x_\ominus$ is the number of incorrect responses.

To train our model, we require the minimization of a loss function for some data set $\mathcal{D} = \{\langle p, \Delta, x\rangle_i\}_{i=1}^D$ containing p, the proportion of times a word was recalled correctly in a practice session, Δ, the lag time since the word was last seen, and the feature vector x for learner fitting. We want to find the optimal model weights θ^* so that:

$$\theta^* = \arg\min_\theta \sum_{i=1}^{D} \ell(\langle p, \Delta, x\rangle_i; \theta) \tag{3}$$

for the modified L_2 regularized squared loss function:

$$\ell(\langle p, \Delta, x\rangle; \theta) = (p - \hat{p}_\theta)^2 + \alpha(h - \hat{h}_\theta)^2 + \lambda\|\theta\|_2^2 \tag{4}$$

where λ is a parameter to control the regularization term and prevent overfitting and α controls the importance of the half-life term. The half-life h is approximated by solving $h = \frac{-\Delta}{\log_2(p)}$. By substituting (1) and (2) into the equation, we obtain:

$$\ell(\langle p, \Delta, x\rangle; \theta) = \left(p - 2^{\frac{-\Delta}{2^{\theta \cdot x}}}\right)^2 + \alpha \left(\frac{-\Delta}{\log_2(p)} - 2^{\theta \cdot x}\right) + \lambda \|\theta\|_2^2 \tag{5}$$

Since ℓ is smooth with respect to θ, we can use gradient descent to fit the weights to the learning traces. The partial gradient of ℓ with respect to θ_k is:

$$\frac{\partial \ell}{\partial \theta_k} = 2(\hat{p}_\theta - p) \ln^2(2) \hat{p}_\theta \left(\frac{\Delta}{\hat{h}_\theta}\right) x_k + 2\alpha \left(\hat{h}_\theta + \frac{\Delta}{\log_2(p)}\right) \ln(2) \hat{h}_\theta x_k + 2\lambda \theta_k \tag{6}$$

Figure 5 shows the 30 day student learning trace, where each $\times$ represents a data instance. The vertical position is the recall rate p of the word in the session, and the horizontal distance between points is the lag time Δ between practice sessions. The prediction $\hat{p}_\theta = 2^{-\frac{\Delta}{\hat{h}_\theta}}$ is plotted by the red dashed line, where the recall probability resets to 1 after each exposure.

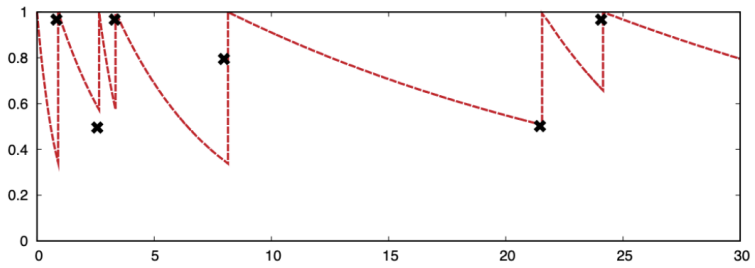


Figure 5. 30 day student learning trace with predicted forgetting curve

4. Data

4.1. Overview

To understand the effectiveness of the spaced repetition model employed by Duolingo, we use a dataset collected by the Duolingo Dataset team. Duolingo maintains a rich dataset tracking student performance across practice sessions. The dataset is called "Replication Data for: A Trainable Spaced Repetition Model for Language Learning." It contains 13 million Duolingo student learning traces that Duolingo personally used in their experiments.

This dataset contains detailed information on each exercise a user completes, with each trace containing the proportion of times a given word or lexeme was correctly recalled (p-recall), the timestamp of the session, the time elapsed since the lexeme was last seen (delta), anonymous user id, languages involved, the specific lexeme string and id, and counts of how many times the lexeme had been previously seen and answered correctly before this session (history seen, history correct) as well as during this session itself (session seen, session correct). In addition, we also added the column for what word is being practiced by extracting the lexeme string.

Each language has different frequencies of each word showing up. Here are the word clouds of the frequencies for the top words studied in English and Spanish below.

	p_recall	timestamp	delta	user_id	learning_language	ui_language	lexeme_string	history_seen	history_correct	session_seen	session_correct	word
0	1.0	1362351020	285	u:e9cF	en	es	phone/phone<n><sg>	1	1	1	1	phone
1	1.0	1362854949	24067	u:eMgB	en	es	rain/rain<vblex><inf><@future>	4	4	2	2	rain
2	0.0	1362255138	4062	u:ffPg	en	es	your/your<det><pos><sp>	46	42	1	0	your
3	0.0	1362490546	1209220	u:fb6u	de	en	eine/ein<det><ind><f><sg><acc>	50	42	1	0	ein
4	1.0	1362399006	321336	u:gOYJ	en	es	restaurant/restaurant<n><sg>	10	7	1	1	restaurant

Figure 6. Example Rows of Dataset (excluding lexeme id)

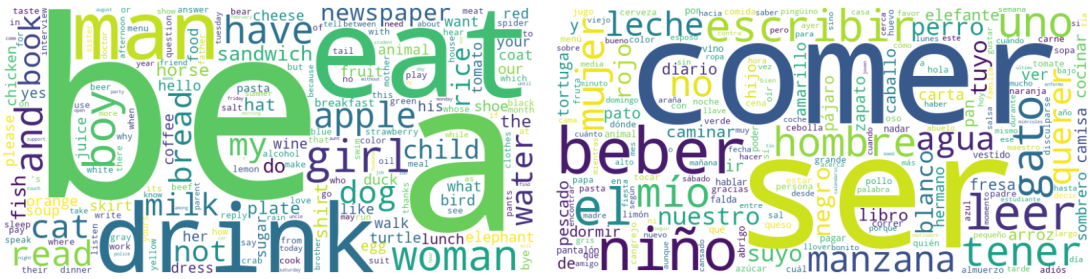


Figure 7. Word clouds of the most frequently studied words for English and Spanish

4.2. Interaction Features

The feature vector x summarizes the student's history with each word, including the number of times the word has been seen, x_n, the number of times it was correctly recalled, $x_\oplus$, and the number of times it was incorrectly recalled, $x_\ominus$.

recall rate	lag(days)	x_n	$x_\oplus$	$x_\ominus$
1.0	0.6	3	2	1
0.5	1.7	6	5	1
0.8	4.7	13	10	3

Table 1. Example training instances

4.3. Lexeme Tagging

Each word is tagged with indicator variables to capture the difficulty of the word. Figure 7 shows examples of lexeme tags for various English words, such as 'camera', which is tagged N for noun and SG for singular. θ_k is the resulting weight, $-1 \leq \theta_k \leq 1$, with negative weights signifying that the word is more difficult and positive weights attributed to easier words.

Lg.	Word	Lexeme Tag	θ_k
EN	camera	*camera*.N.SG	0.77
EN	ends	*end*.V.PRES.P3.SG	0.38
EN	circle	*circle*.N.SG	0.08
EN	rose	*rise*.V.PST	*-0.09*
EN	performed	*perform*.V.PP	*-0.48*
EN	writing	*write*.V.PRESP	*-0.81*

Figure 8. Example lexeme tags for various English words.

5. Methods and Code

5.1. Data Processing

When working with a large dataset like Duolingo's 13 million row practice session data, sampling is often necessary for computational feasibility and to reduce processing time. Our main strategy was to extract a 10 percent random sample of the full dataset for analysis. There are a few key reasons why this sampling approach was appropriate:

1. **Sample Size:** Even at 10 percent, a sample of 1.3 million rows provides a very large number of data points to work with. This sample size ensures we can derive statistically robust insights while making the data more manageable.
2. **Representativeness:** By taking a simple random sample, each row had an equal probability of being selected. This helps ensure the 10 percent sample is representative of the same underlying data distribution as the full dataset.
3. **Computational Power:** Working with the entire 13 million row dataset would likely be extremely computationally intensive and slow, especially for iterative analysis. The 10 percent sample strikes a balance between preserving ample data and making processing feasible.

Initially, we considered whether to stratify the sampling to ensure an even distribution of languages being learned in the sample. However, we found that the language traces in the 10 percent random sample were not drastically skewed and had similar summary statistics compared to the full data.

Since the stratified sampling did not appear necessary to achieve language representation, we decided to proceed with the simple 10 percent random sample. This allowed us to leverage the representativeness advantages of random sampling while avoiding potential imbalances that could be introduced through stratification.

By working with this 10 percent sample comprising over 1 million rows, we were able to perform in-depth analysis in a timely manner without losing the ability to derive robust, generalizable insights from Duolingo's large-scale practice data. Sampling is an effective technique that we employed to make large datasets more tractable while preserving validity.

5.2. P-Recall Distribution

Our exploration of the dataset starts with the probability recall data which reveals a concerning distribution. The data appears heavily skewed towards 1.0, with even the 25th percentile (Q1) reaching this value. This suggests a the majority of the data points represent instances where users perfectly recalled the information. Interestingly, the second largest data point is 0, indicating a sharp drop-off after perfect recall. The overall mean of 0.89 further emphasizes this skewed distribution.

The skewed distribution of Duolingo's recall data, heavily concentrated around perfect recall (1.0), could be attributed to several factors. Selection bias might be at play, as users who consistently struggle are more likely to abandon the platform altogether. Additionally, the dataset might be limited to European languages, potentially excluding languages with different learning difficulties. Furthermore, the ease of cheating within Duolingo, where users might exploit external resources or peek at answers, could inflate

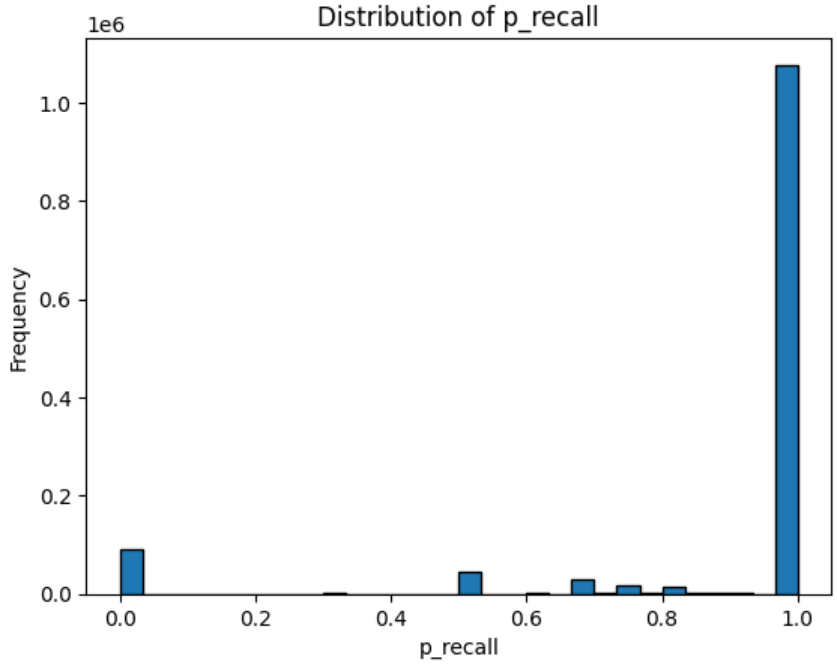


Figure 9. Histogram of Recall Rates

perfect recall scores. A ceiling effect could also be present, as the data might not capture near-perfect recall with minor mistakes, forcing all successful attempts into the 1.0 category. Finally, users strategically focusing on easier lessons to maintain their streaks could contribute to the overrepresentation of perfect recall data, neglecting the learning curve for more challenging concepts.

This skewed distribution presents several challenges for training the spaced repetition model effectively. Firstly, the model might prioritize optimizing recall for users who consistently achieve perfect scores, neglecting the needs of a potentially large user base struggling with lower recall rates. Secondly, with a significant portion of the data concentrated at 1.0, the model might have difficulty learning the nuances of spaced repetition for users with varying recall abilities. Finally, evaluating the model's performance becomes problematic. Standard metrics like mean squared error might be skewed towards optimizing for perfect recall, even if the model performs poorly for users with lower recall rates. For example, when we tried to implement a random forest regression as an additional regression algorithm, we found that it would end up predicting 1.0 every time, making seem it like it had essentially perfect recall.

Consequently, the study has opted into using specific regression models as well as alternative evaluation metrics that are less sensitive to outliers, such as median absolute error, to provide a more accurate assessment of the model's performance for the broader user base. Another solution would be to simply oversample the data points with lower recall rates and undersample those at 1.0 to artificially balance the distribution. We could also try different data transformation techniques such as log transformations to try and make the data more normalized. However, in order to replicate results similar to those in the study, we have not looked too deeply into this strategy.

5.3. Delta (Lag time since the word was last seen)

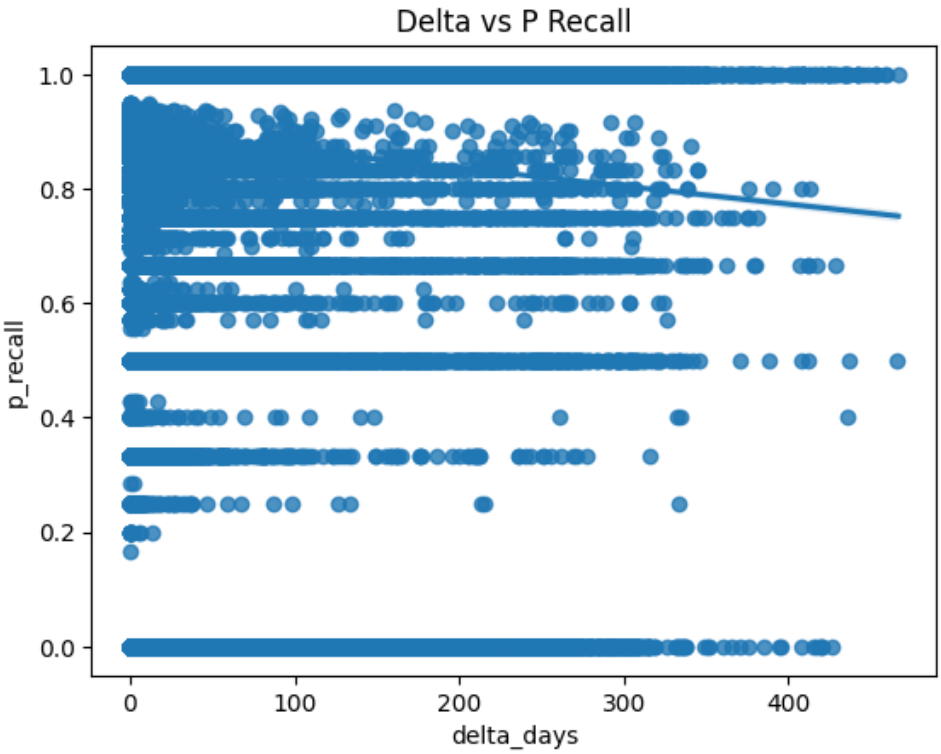


Figure 10. Delta (Days) vs P-recall Scatterplot

The clustering of data points at 1, 0, and around 0.6-0.8 for p-recall makes it difficult to definitively determine a relationship between delta and p-recall. While there seems to be a trend of more concentrated higher p-recall values with increasing delta, it's not conclusive. This suggests incorporating other variables like learning history, session information, lexeme difficulty, and session length into the model is necessary for a more comprehensive understanding of factors affecting recall probability within our spaced repetition model.

5.4. Language Insight

A closer look at the recall probability for individual languages within the dataset reveals some interesting details. While all six European languages (Italian, Portuguese, English, Spanish, German, and French) exhibit a high average recall probability above 0.88, there is a minimal variation between them. This could suggest a general consistency in user performance across these languages, but it's important to consider the possibility that the skewed data distribution towards perfect recall might be masking underlying differences. French, with the lowest average recall (0.883975), might be inherently more challenging for Duolingo users compared to the others. Later on, we shall see if this difference is substantial enough to make a difference when predicting recall probability rates by country.

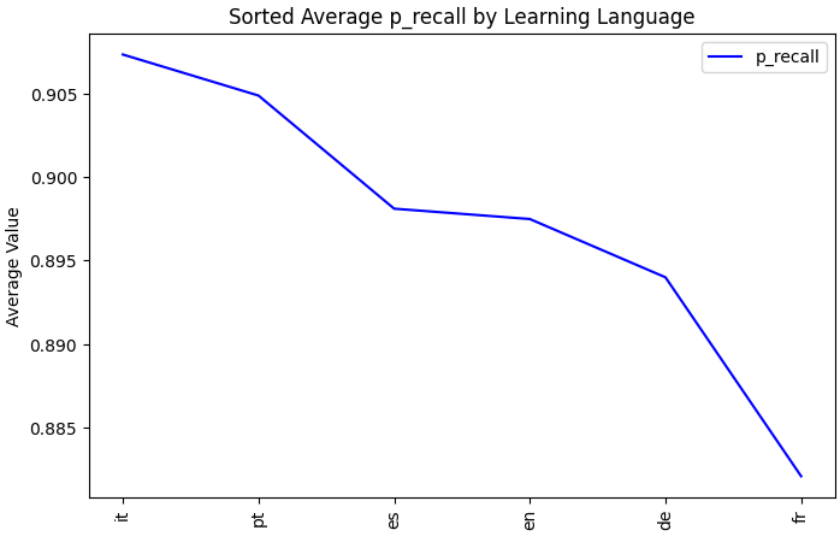


Figure 11. Highest to Lowest Recall Rate By Language

5.5. Session Accuracy and Learning History

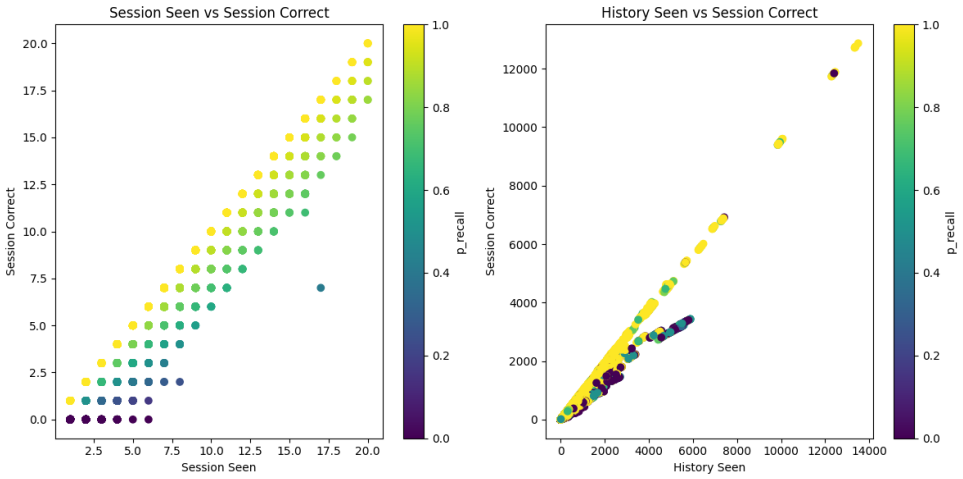


Figure 12. Session and Learning History Data and their effect on P-recall

Our scatterplots suggests positive correlations between session interactions and p-recall. We can see that frequent exposure and correct answers within a session (session seen/correct up) and across learning history (history seen/correct up) both lead generally to a higher probability of recall (p-recall up). Therefore, integrating these variables can enhance the spaced repetition model to be more insightful and meaningful.

5.6. Main model

In the context of language learning, spaced repetition models aim to predict the optimal intervals for reviewing vocabulary words and grammatical concepts. These models leverage features such as the time elapsed since the last review, the learner's past performance history, and the lexeme characteristics of the content itself. By accurately forecasting the probability of successful recall, spaced repetition models can dynamically schedule review sessions in a way that maximizes retention while minimizing redundant practice.

Duolingo, a leading language learning platform, employs spaced repetition as a core component of its pedagogy. To continuously refine and optimize its spaced repetition algorithms, we are replicated Duolingo's modelling framework to allow for the comparison and evaluation of various regression techniques. This framework encompasses four main stages: reading input data, training regression models, evaluating performance, and storing model outputs.

This is what our function looks like:

```
def model_eval(method, input_file, max_lines=None, omit_bias = True, omit_lexemes= True, omit_h_term= True):
    trainset, testset = read_data(input_file, method, omit_bias, omit_lexemes, max_lines)
    model = SpacedRepetitionModel(method=method, omit_h_term=omit_h_term)
    model.train(trainset)
    model.eval(testset, 'test')
    if not os.path.exists('results/'+method+'/'):
        os.makedirs('results/'+method+'/')
    model.dump_weights(f'results/{method}/h-weights')
    model.dump_predictions(f'results/{method}/h-preds', testset)
```

Figure 13. Model Steps

5.6.1. Read Data

The read data function preprocesses Duolingo's practice session dataset into training and testing sets for input to the regression models. It reads the raw CSV data line by line, extracting relevant features like recall proportion, time since last review, language details, lexeme string, and prior performance history. Depending on the specified regression method (e.g. half-life, logistic, Leitner, Pimsleur), it computes derived features customized for that approach.

The function constructs feature vectors comprising tuples of feature names and values. It allows omitting the bias term or lexeme-specific features via configuration flags. After processing all instances, it splits the data into a training set and a testing set.

The read data function returns the training and testing sets as lists of instances containing the target recall proportion, feature vector, language pair, and session-level performance counters. This modular preprocessing step enables consistent data formatting and feature extraction across the regression experiments.

5.6.2. Train

After preprocessing the data with the read data function, we proceeded to train the regression models on the extracted training set. Our modeling framework includes implementations of different regression techniques tailored for the spaced repetition problem, such as half-life regression (HLR), logistic regression (LR), as well as methods based on the Leitner system and Pimsleur's model.

The core training logic is encapsulated in the train and train update functions. The train function iterates through each instance in the provided training set and invokes the train update method, which performs the actual model weight updates based on the specified regression approach.

For the half-life regression method, the train update function computes the predicted recall probability and half-life value for the current instance using the current model weights. It then calculates the gradients of the loss functions for the weights for both the recall probability and half-life terms. These gradients are used to update the model weights via stochastic gradient descent, with the learning rate decaying inversely with the square root of the feature counts to ensure convergence.

The function also includes options to omit the half-life term from the regression, apply L2 regularization to prevent overfitting and scale the learning rate by the inverse of the predicted recall probability to account for the characteristics of the logistic loss function.

For logistic regression, the train update function follows a similar approach, computing the predicted recall probability, calculating the error with respect to the true value, and updating the weights based on this error and the feature values. It also includes L2 regularization and decaying learning rates.

In contrast, for the Leitner system and Pimsleur's model, which do not rely on continuous optimization, the train function simply shuffles the order of instances in the training set to facilitate sampling during the evaluation phase.

By implementing these training procedures within a unified framework, our approach enables a systematic comparison of the different regression techniques on the same data distribution, while ensuring consistent handling of features, regularization, and optimization settings across methods.

5.6.3. Prediction

Now it was finally time to predict the recall probabilities described in their methodology within our codebase.

To compute the predicted recall probability $\hat{p}$ and half-life $\hat{h}$ for a given instance, we followed the approach outlined in their methodology, which varies based on the chosen regression method.

For the **half-life regression (HLR)** method, we first computed the predicted half-life $\hat{h}$ using the following equation:

$$\hat{h} = \text{hclip}(b^{\sum_k w_k x_k}) \tag{7}$$

Where b is the base (typically 2), w_k are the learned weights, x_k are the feature values, and $\text{hclip}(\cdot)$ is a clipping function to bound the half-life within a predefined range. The predicted recall probability $\hat{p}$ was then calculated as:

$$\hat{p} = \text{pclip}(2^{-t/\hat{h}}) \tag{8}$$

Here, t is the time elapsed since the last review, and $\text{pclip}(\cdot)$ clips the probability to a valid range (e.g., [0.0001, 0.9999]).

For the **Leitner system** method, the half-life $\hat{h}$ was computed as:

$$\hat{h} = \text{hclip}(2^{x_{\text{diff}}}) \tag{9}$$

Where x_{diff} is the feature representing the difference between the number of correct and incorrect responses. The recall probability $\hat{p}$ was then calculated using the same equation as in HLR.

For the **Pimsleur's** model, the half-life $\hat{h}$ was obtained as:

$$\hat{h} = \text{hclip}(2^{2.35x_{\text{total}} - 16.46}) \tag{10}$$

Where x_{total} is the feature representing the total number of exposures. Again, the recall probability $\hat{p}$ was computed using the same equation as in HLR.

For **logistic regression**, there is no explicit half-life term. Instead, the predicted recall probability $\hat{p}$ was computed directly using the logistic function:

$$\hat{p} = \text{pclip}\left(\frac{1}{1 + e^{-\sum_k w_k x_k}}\right) \tag{11}$$

By implementing these prediction functions within our codebase, we aimed to faithfully replicate the methodology described in the original experiments, enabling us to conduct comparative analyses and evaluations across the different regression techniques for spaced repetition modeling.

5.6.4. Evaluating

After obtaining the predicted recall probability and half-life values from the predict function, we create a losses function to compute the squared loss terms for evaluating the accuracy of the recall probability and half-life predictions made by the regression models. This losses function calculates the squared loss for recall probability and half-life predictions by taking the squared difference between the true and predicted values. It returns a tuple containing these squared loss terms, as well as the predicted recall probability and half-life values, which we use to evaluate our model's predictions.

The eval function plays a crucial role in evaluating the performance of the trained regression models on the held-out test set. It iterates through each instance in the test set, computing the squared loss terms for recall probability and half-life predictions using the losses function. These squared loss values, along with the ground truth and predicted values, are stored in a dictionary. After processing all instances, the function calculates evaluation metrics such as mean absolute error (MAE) and Spearman's rank correlation coefficient for both recall probability and half-life. It also computes the total loss, which is a combination of the squared loss terms for recall probability, weighted squared loss for half-life, and L2 regularization term. These evaluation results are then printed or written to a file.

In addition to the eval function, the code includes two helper functions: dump weights and dump predictions. The dump weights function saves the learned model

weights to a file, allowing for further analysis or deployment. The dump predictions function writes the ground truth and predicted values for recall probability and half-life, along with additional instance information, to a file. This file can be used for further inspection or analysis of the model's predictions.

5.6.5. Performance Metrics

```python
def sr_evaluate(preds_file):
    print(f'%%%%%%%%%%%%%%%%%%%% {preds_file} %%%%%%%%%%%%%%%%%%%%')
    data = pd.read_csv(preds_file, sep='\t')

    print('==== mean absolute error ====')
    p_minus_pp = abs(data['p'] - data['pp'])
    p_minus_mean_p = abs(data['p'] - data['p'].mean())
    print(f'MAE: {p_minus_pp.mean()}')
    print(f'MAE (baseline): {p_minus_mean_p.mean()}')
    _, p_value = ttest_ind(p_minus_pp, p_minus_mean_p)
    print(f'p-value: {p_value}')

    print('==== area under the ROC curve ====')
    roc_auc = roc_auc_score(data['p'].round(), data['pp'])
    print(f'ROC AUC: {roc_auc}')

    _, p_value = wilcoxon(data['p'].round(), data['pp'], alternative='greater')
    print(f'p-value: {p_value}')

    print('==== half-life correlation ====')
    corr_coef, p_value = spearmanr(data['h'], data['hh'])
    print(f'Spearman correlation coefficient: {corr_coef}')
    print(f'p-value: {p_value}')
```

Figure 14. Performance Metrics

To evaluate the success of each model we use, we employ several statistical metrics. Mean Absolute Error (MAE) is calculated to measure the average magnitude of errors in the predicted recall probabilities, compared to both the true values and a baseline mean. A two-sample t-test assesses the statistical significance of the difference between the model's prediction errors and the baseline errors. The Area Under the ROC Curve (ROC AUC) is computed by treating the recall probabilities as a binary classification problem, providing a measure of the model's ability to discriminate between positive and negative instances. The Wilcoxon signed-rank test, a non-parametric statistical test, is applied to the true and predicted recall probabilities to determine if their distributions differ significantly. Additionally, Spearman's rank correlation coefficient quantifies the monotonic relationship between the true and predicted half-life values, evaluating the agreement between the two variables.

We will now display our results and findings to compare to the literature as well as some additional conclusions we were able to make in the following section.

6. Results and Conclusion

Model	MAE	AUC	COR
HLR	**0.129***	0.532*	**0.201***
HLR -lex	**0.129***	0.532*	0.163*
HLR -h	0.41	0.531*	-0.139*
HLR -lex -h	0.4	0.53*	-0.139*
Leitner	0.223	0.**543***	-0.09
Pimsleur	0.431	0.514*	-0.12*
LR	0.225	0.488*	0.002
LR -lex	0.225	0.489*	0.001

Table 2. Results of statistical tests for all algorithms. Bolded numbers represent the best results for each test, and asterisks represent statistically significant values.

Our investigation into half-life regression (HLR) as a model for spaced repetition in language learning has provided substantial insights into optimizing memory retention. By replicating and extending the study of Settles and Meeder (2016) using Duolingo's extensive dataset of 13 million learning sessions, we confirmed that the HLR model with lexeme tags performs the best in predicting memory retention over traditional models like the Leitner system and Pimsleur method. Table 2 shows the statistical test results for all algorithms. The Leitner model, which is Duolingo's previously used model, outperformed HLR in the AUC test by a small amount.

Lexeme tagging plays a crucial role in refining the HLR model's predictions, indicating that the complexity of words and their grammatical characteristics significantly impact memory retention. This insight underscores the importance of considering linguistic nuances in the design of spaced repetition algorithms. However, in the study conducted by Settles and Meeder, they implemented HLR with millions of real Duolingo users, who provided feedback. They found that words with highly negative lexeme tags (i.e. very "difficult" words) would tend to decay too rapidly, forcing students to constantly practice them.

After implementing the evaluation metrics and running the regression models on data from individual languages, the results revealed no significant differences in the ability to predict recall probability and half-life values across different languages. Despite the inherent linguistic variations between languages, the models exhibited comparable performance metrics such as mean absolute error, area under the ROC curve, correlation coefficients, and statistical test results, regardless of the specific language being learned. These findings suggest that the core principles of spaced repetition and the underlying factors influencing memory retention may be consistent across different language contexts. The consistency indicates our regression models developed in this study can be applied broadly across various language learning scenarios without the need for extensive language-specific customization or tuning.

Despite the strengths of the HLR model, the skewed distribution of recall data toward perfect recall presents challenges, such as potential overfitting to high-performing learners. Addressing these challenges through alternative evaluation metrics, data transformation techniques, or balanced sampling methods could further improve the model's

applicability and accuracy.

Future work could include the application of neural networks. Recurrent neural networks (RNNs) or variants like Long Short-Term Memory (LSTM) networks can analyze temporal patterns in learning data, enabling the prediction of when a learner is likely to forget a word or concept. These networks are adept at handling sequential information, mirroring the way spaced repetition schedules are designed to optimize memory retention over time. Through continuous training on user interaction data, the model would dynamically adjust the intervals of spaced repetition.

References

[1] Settles, B., & Meeder, B. (2016). A trainable spaced repetition model for language learning. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics* (pp. 1848-1858).

[2] Ebbinghaus, H. (1885). *Über das Gedächtnis: Untersuchungen zur experimentellen Psychologie*. Duncker & Humblot.

[3] Melton, A. W. (1970). The situation with respect to the spacing of repetitions and memory. *Journal of Verbal Learning and Verbal Behavior*, 9(5), 596-606.

[4] Pimsleur, P. (1967). A memory schedule. *The Modern Language Journal*, 51(2), 73-75.

[5] von Ahn, L., & Hacker, S. (2012). Duolingo: Learn a language for free while helping to translate the web. In *Proceedings of the Innovation and Technology in Computer Science Education Conference*.

[6] Leitner, S. (1972). *So lernt man lernen*. Herder.