Datathon

The GC

February 2026

1 Model

1.1 Assumptions

- Single global half-life parameter for all user and all words (can be adjusted to improve power)
- independent factorization of difficulty via language, type of speech, and word length
- Approximate memory decay outside of Beta distribution

1.2 Derivation

The model we use to answer the question:

What is the probability $\mathbb{P}$ that the user U still remembers the word W in language L, given W is in type T with length l after time t, where

$$T \in \{noun, verb, number, adjective, pronoun, preposition, adverb\},$$

and

$$L \in \{English, Spanish, German, French, Italian, Portuguese\}.$$

uses uses a Bayesian Beta–Binomial model and the posterior mean $\mathbb{P} = \frac{\alpha}{\alpha+\beta}$ as the recall probability.,

$$\mathbb{P} \to f(\mathbb{P} \mid x) = \frac{f(x \mid \mathbb{P})g(\mathbb{P})}{\int_{\mathbb{P}} f(x \mid \mathbb{P})g(\mathbb{P})d\mathbb{P}}$$

where $\mathbb{P} \sim Beta(\alpha, \beta)$.

Let α_p and β_p denote prior beliefs of α and β, then we can derive them as:

$$p_{prior} = \sigma(b_0 + b_L f(L) + b_T g(T) + b_l h(l))$$

where

$$\sigma(x) = \frac{1}{1 + e^{-x}}$$

and

$$\alpha_p = z \cdot p_{prior}$$
$$\beta_p = z \cdot (1 - p_{prior})$$

where f, g, h return difficulty related features (higher=easier), which are combined in a logistic regression to get the prior recall probability p_{prior}

$f(L)$ is computed based on studies FSI language difficulty,

$$f(L) = \epsilon + (1 - 2\epsilon)\Big[1 - \frac{H(L) - H_{min}}{H_{max} - H_{min}}\Big]$$

where ϵ is a tiny number that ensures $f(L) \in (0, 1)$, $H(L)$ returns the hours needed to learn language L.

$g(T)$ is retrieved from the study of Suranto [1], which describes the percentage of identified words in different word types for students.

$h(l)$ is estimated using a linear regression with rows with history_seen ≤ 3:

$$h(l) = \beta_0 + \beta_1 l$$

Once we have α_p and β_p, we obtain α and β by:

$$\alpha = \text{History correct} + \alpha_p$$
$$\beta = \text{History seen} - \text{History correct} + \beta_p$$

and $\mathbb{P}$ is calculated by:

$$\mathbb{P} = \frac{\alpha}{\alpha + \beta}$$

we can further apply a memory decay function

$$d(t) = 2^{-t/h}$$
$$t = \frac{delta}{3600}(\text{unit in hours})$$

with the half life model. Then the probability $\mathbb{P}_t$ of the user U remembering the word W after time t is calculated by:

$$\mathbb{P}_t = d(t) \cdot \mathbb{P}$$

2 Fitting

Once we have the model working, we fit our model, then test the correctness and evaluate.

2.1 Fit&Correctness

To fit our model, we use the following procedure:

1. randomly split users into train and test (80%:20%)
2. for each user, sort by timestamp
3. use the train group to train the model, and test it with the test group

Then for each row i,

- n_i = session_seen
- m_i = session_correct

compute Binomial log-likelihood:

$$logY_i = m_i log\mathbb{P}_{ti} + (n_i - m_i)log(1 - \mathbb{P}_{ti})$$

then calculate the average negative log-likelihood (NLL):

$$NLL = -\frac{1}{N}\sum_i logY_i$$

Where N is the total number of trails. The smaller the NLL, the better the model describes the data.

Parameters we can tune to improve the model:

- z: adjust the weight of prior beliefs
- h: half life parameter
- $\vec{b} = \{b_0, b_L, b_T, b_l\}$

We want:

$$\theta^* = (z^*, h^*, \vec{b^*}) = argmin_\theta NLL(\theta),$$

where $\theta = (z, h, \vec{b})$. We can use scipy.optimize.minimize.

2.2 Evaluation

We can compare our model to other common models. Namely:

- Half-Life Regression (HLR)
- ACT-R Memory Model
- Logistic Regression / Neural Classifiers for Recall
- Simple Exponential Forgetting Curve

3 Advantages and limitations

3.1 Limitations

- Cannot fully utilize time data
- The linear regression used in $h(l)$ may not predict well for words with length exceeding the longest word in the dataset.

3.2 Advantages

- More accurate than HLR
- Computationally cheap
- Does not require a lot of data
- Since it is structurally simply, modifications can be made easily.

4 Test results

4.1 Bayesian Beta-binomial model

The result is obtained by running model_stream_fixed.py with parameters:

- CHUNK_SIZE = 200_000 # The size of each chunk of data
- RANDOM_SEED = 42
- EPS = 1e-9 # to avoid log(0)
- EPS_F = 1e-3 # epsilon in f(L)
- USE_SUBSAMPLE_FOR_TRAINING = True
- TRAIN_SUBSAMPLE_ROWS = 1_000_000 # rows used per NLL eval while fitting
- MAXITER = 50 # optimizer iterations
- PRIOR_SAMPLE_MAX = 1_000_000 # sample size for estimating the prior
- half-life upper bound = 50000.0

Note that for performance improvements, $\vec{b}$ is optimized first, then (z, h) follows.

```
Pass 1: collecting unique users...
Total unique users: 115222
Train users: 92177, Test users: 23045
Pass 2: estimating h(l) = beta0 + beta1*l via streaming regression...
```

```
h(l): beta0 = 0.8878, beta1 = 0.0012  (from 2509589 rows)
Pass 3: sampling low-history rows to fit prior coefficients (b0, bL, bT, bl)...
Prior logistic regression sample size (before cleaning): 1008354
Prior logistic regression sample size (after cleaning): 931422
Prior coefficients (from grouped-binomial logistic regression):
b0 = 1.1238, bL = 0.1456, bT = 0.1393, bl = 1.0098
Pass 4: precomputing numeric feature file (this is a one-time cost)...
Numeric feature file written to precomputed_features.csv
Fitting z and half-life h on training users (using subsample for speed)...

Estimated z  = 30.4329
Estimated h  = 23023.5617 hours (half-life)

Computing final NLL on full train and test sets...

Final NLL (train): 0.310319
Final NLL (test) : 0.312115
```

4.2 HLR

The test result is obtained by using default parameters, but added a function to compute NLL.

```
Reading data...
1000000 rows read...
2000000 rows read...
3000000 rows read...
4000000 rows read...
5000000 rows read...
6000000 rows read...
7000000 rows read...
8000000 rows read...
9000000 rows read...
10000000 rows read...
11000000 rows read...
12000000 rows read...
Done reading.
Total instances: 12854226
Total unique users: 115222
|train| = 10354172 instances
|test|  = 2500054 instances
Training HLR model...
Eval on train set:
train 283.1 (p=0.1, h=149.0, l2=134.0) mae(p)=0.125
cor(p)=0.038 mae(h)=117.387 cor(h)=0.208
Eval on test set:
```

```
test 283.9 (p=0.1, h=149.8, l2=134.0) mae(p)=0.125
cor(p)=0.034 mae(h)=117.855 cor(h)=0.195
Computing binomial NLL (per row)...
NLL (train): 0.586105
NLL (test) : 0.578390
```

References

[1] Suranto and Yuspik. “The Analysis of Student’s Ability to Identify Parts of Speech”. In: *English Teaching and Applied Linguistics Journal* 1.1 (2024), pp. 18–26.