Lomb

Predicting the Memory Curve in Text-based Language Learning

Overview

Language learning has benefitted significantly from the Information Age. SRS (Spaced Repetition System) learning apps, such as Anki, Memrise or Duolingo, have become very popular and have been shown to improve the memorization rate of vocabulary in several studies. In the SRS paradigm, a word or concept is prompted (in the form of a question, flashcard or exercise) at progressively longer intervals; an approach which mirrors Ebbinghaus' well-known Forgetting Curve.

However this approach is not without problems. The main problems are: 1) Since the algorithms work by scheduling review times, students must review when the algorithm requires them to and for the amount of time it requires them to. This lack of control is generally seen as undesirable: often people don't study full time but must work or study and therefore have varying amounts of time through out the week to devote to doing reviews, or conversely they might wish to devote more review time before an upcoming exam. To illustrate, failing to do reviews on a weekend will result in having to devote three or four times more time to doing reviews next time. 3) Typically students, particularly intermediate and advanced students, will attend classes, read, and generally have much exposure in the target language. The SRS model has no way of accounting for this additional exposure, even when user activity data is available, and in this situation leads to mostly redundant and unnecessary reviews.

For the latter reason, intermediate learners often get bored with SRS apps and devote more time to reading, watching films or videos and generally increasing their exposure in the target language. They sometimes use assisted reading apps (like LWT or LingQ), dual language subtitles or popup dictionary extensions for their browsers. Due to the lack of intermediate-level material in many languages, or simply lack of material that is appealing to learners, this is often done using native-level media.

This approach is also problematic. Although intermediate learners might be able to understand 60% of the words in a given sentence, much of the meaning will be contained in new words which must be learnt. But which words to learn? Which words to revise?

For this research, we developed an app which can be used to both read books and review vocabulary. The data collected from user reading activity is used to predict how likely a user is to forget a word. Our model differs from SRS in several key ways: - Accounts for additional vocabulary exposure outside review sessions. - Adapts to learner's desired review time.

Data collected

On reading, the following data is logged for each word that is read : `EXPOSURE_AND_NO_LOOKUP`, `EXPOSURE_AND_SENTENCE_LOOKUP`.

On review we collect `REVIEW_CLICKED` and `REVIEW_NOT_CLICKED`.

Activity logged for each word can therefore be summarised into a sequence, which can be of arbitrary length. Each element of the sequence will contain the following variables:

REVIEW_NUMBER, SECONDS_SINCE_LAST_REVIEW, CLICKED_LAST_REVIEW, EXPOSURES_SINCE_LAST_REVIEW_WITH_LOOKUP, EXPOSURES_SINCE_LAST_REVIEW_WITHOUT_LOOKUP, SECONDS_SINCE_LAST_EXPOSURE_WITH_LOOKUP, SECONDS_SINCE_LAST_EXPOSURE_WITHOUT_LOOKUP, TIME_SPENT_LOOKING_AT_DEFINITIONS_LAST_REVIEW, CLICKED_LAST_REVIEW

The program keeps a database with these sequences. There is also a `next_review` entry which is created whenever a `REVIEW_CLICKED` or `REVIEW_NOT_CLICKED` event is received, and which is updated whenever the reading data of that word is received.

A LSTM is trained using this data. Hopefully, it can predict something!!

Keywords

- Personalized learning.
- Computer testing
- CSC Computer Supported Cooperative work
- CHI Computer Human Interaction
- Psychology journals.
- Transaction on learning technology
- Education technology
- Adaptive learning (that seems to be more of a ML term)
-

2020/05/08

I wrote the first service of the Lomb application. It is the tracking, which is the most important part. Initially it had terrible performance, because every time it received a request it would convert between the MongoDB and a dataclass. I rewrote the domain and db modules so that they are now together (yes, coupled). This allows me to execute the MongoDB queries, particularly the updates, directly. CPU usage went from ~30% to ~10% by doing this.

Stuff to do:

- Implement word lookups in the reader by dividing the window into three parts (similar to the words view).
- Look into why the word cumulative percentage counts are wrong (they clearly are).
- Implement a users service to handle registrations, logins and tokens.
- Implement a library. This should just keep texts as arrays of chunks (locations). It could also do all of the heavy NLP lifting and store that for each location. This would make a lot of sense for stuff like filtering by tag.
- Implement a word service. Among other things, this keeps the known words for users, so that the frontend can filter words sent to tracking.

- Implement a reader which grabs known words, text (the array of locations and tokens, etc.). I guess the most performant way to make this work is to shift a lot of logic to the frontend. Otherwise Python will have to do a lot of work...
- Implement a simple RNN or LSTM with the data so that I go through the process.
- Implement other algorithms to see how they work.