GitHub Gist

oampo / **spaced-repetition.md** Secret
Last active 3 months ago

Spaced repetition project brief

spaced-repetition.md

Brief

You have been hired by language learning startup FrenchX to create an app which uses spaced repetition to help people memorize a foreign language. The app will display words in one language, and ask you to recall the corresponding word in a second language. You will need to develop an algorithm which determines which words to test the user on, create a frontend which displays the words and allows the user to enter answers, and a backend to store the user's progress so they can pick up and continue learning at any point.

Specification

Spaced repetition algorithm

Spaced repetition is a method for efficient learning that has you practice concepts or skills over increasing periods of time. It's based on the notion of a "forgetting curve," or the idea that over time, if we don't actively use or reflect on something we know, our knowledge decays. With spaced repetition, we stay ahead of that moment of forgetting, but we do it in a smart way: if we know something, we don't need to practice it for some period of time. If we don't know something, we do need to practice it.

For example, let's say that you wanted to learn four new words, A, B, C and D. Using spaced repetition you might test the words in this order:

ABABCACBDCADB...

Notice how the spacing between the questions gets longer as you go on. So subsequent tests on question A are separated by one question (B), then two questions (BC), then four questions (CBDC). And the same thing happens with question B and question C. If you got one of the questions wrong then you would reduce the spacing for that question to make sure that the correct answer is.

To complete this part of the project you need to write an algorithm which generates the sequence of words to ask, taking into account whether the user got the question correct or incorrect. There is no single correct solution to this problem and we are not looking for the perfect algorithm; the goal is for your algorithm to have the basic spaced repetition characteristics.

For the first version, aim for the simplest possible implementation, rather than the most efficient or the most effective. You can start with certain assumptions that over time you may want to re-factor; for example, feel free to assume there are only 10 questions and that the user can go through all of those questions in a session, rather than needing to prioritize which questions the user will tackle in a day. Similarly, feel free to assume that the user will do one session every day. We'll later re-factor these.

Requirements

- Has a list of pairs of words as an input
- Outputs the next pair to test the user on
- Word pairs earlier in the list should first appear earlier than pairs later in the list

- Repetitions should start out close together
- Repetitions should get further apart if you answer correctly
- Repetitions should get closer together if you answer incorrectly

Frontend

The frontend for the app will be built using React and Redux, and allow users to login, answer the questions, and see how many questions they have successfully answered. To answer a question the user will be shown a word in the language they are trying to learn on the left-hand side of the screen, and asked to type the corresponding word in their native language on the right-hand side. When they submit the they will be given feedback on whether they were correct, and taken to the next question.

The frontend will need to submit information stating whether the question was answer correctly or not to the backend so that the spaced repetition algorithm can take that into account. This will also allow the user's score (how many questions they have answered correctly) to be updated.

To authenticate, you are going to be using Google's implementation of OAuth. This will allow anyone with a Google account to simply and easily register or login to your app. On the frontend this makes your requirements very simple: you simply need a login button which links to the appropriate backend endpoint, and a combination of Google and your backend will take care of the rest.

Requirements

- Technologies: React, Redux
- Two pages: Landing page and spaced repetition page
- Landing page:
  - Advertise the app
  - Register/login with Google button
- Spaced repetition page:
  - Displays current word
  - Text input for answer
  - Notifies the user whether they were correct or incorrect
  - Submits correct/incorrect to backend
  - Displays a count of how many questions were answered correctly

Backend

The backend of the app plays three key roles. The first is authentication. To allow users to authenticate, the backend should use the Google OAuth 2.0 strategy for Passport.

The second role is to integrate the spaced repetition algorithm into your app. It should have an endpoint for the frontend to fetch the next question from, and an endpoint for the frontend to record what the user's response was.

The third role is to store the users' progress in a MongoDB database. This should include both the number of questions which they have answered correctly, plus any information about their answer history that your spaced repetition algorithm needs in order to generate a new sequence of words to test the user.

Requirements

- Technologies: Node.js, Express, MongoDB, Passport, OAuth
- Allow users to register/login using Google OAuth
- Use the spaced repetition algorithm to generate the next word pair
- Pairs of words should be stored in a Mongo database
  - This should be a fixed array of questions for an MVP
- Store the number of questions which users have answered correctly in the database

- Store whatever information is needed for the algorithm about the user's answer history in the database

Mockups

- Landing page - This allows the user to login or register to use the app.
- Spaced repetition page - This allows the user to answer questions by typing answers in the text input.