Modelo competencias

A la hora de realizar una plataforma educativa, no hay que tener en cuenta únicamente los contenidos, si no que también existen las competencias.
En una situación normal, las competencias no están sujetas a un solo contenido, sino que una unidad puede contener varias competencias y estas pueden participar en varios. El problema que se da con la librería a utilizar, Ebisu, es que no entiende ninguna relación entre los contenidos. Esta librería crea modelos a partir de una unidad y no se relacionan entre ellos.
Es por esto que propongo dos modelos:

- Asumir que es una competencia por cuestionario/temario/contenido. Cada vez que el usuario se conecte a la plataforma, esta realizará un cálculo del porcentaje de recuerdo y procederá a mostrar los contenidos menos adquiridos por el usuario. Este usuario guardará un modelo por cada unidad realizada.
- Asumir que cada pregunta pertenece a una competencia. Con esto podemos calcular la puntuación de competencias teniendo en cuenta la unidad total o los contenidos realizados por el alumno.
    - Se pierde un punto por cada pregunta fallida.
    - Se gana un punto a la hora de realizar un repaso o cuestionario.
    - A la hora de hacer repasos, primero calculará el porcentaje de recuerdo como en el anterior modelo. Después se mirará la puntuación de competencias teniendo en cuenta la situación actual del alumno.
    - A su vez, se puede recomendar que haga determinados ejercicios teniendo en cuenta la puntuación del usuario respecto a la totalidad de las competencias, comprobando qué competencia tiene un valor menor respecto al total.

Ebisu

Esta librería tiene 4 funciones principales:

- ***defaultModel(half-life, alpha, beta)***: crea el modelo utilizado por esta librería (distribución Beta(alpha, beta)). En half-life le enviamos el tiempo que transcurre hasta que el porcentaje de recuerdo sea aproximadamente 50%. Este modelo se deberá guardar en la base de datos del usuario, y se actualizará cada vez que usemos la función *updateRecall*.
- ***predictRecall(model, tnow, exact)***: devuelve el porcentaje de recuerdo que tiene un contenido.
    - model: modelo almacenado del estudiante
    - tnow: tiempo que ha pasado desde la última vez que se realizó el último cuestionario.
    - exact: para ser más exactos, si este es true devuelve el porcentaje teniendo en cuenta la estadística. Si su valor es falso, devuelve una aproximación para ahorrar a la hora de realizar el cálculo.

- ***updateRecall(model, successes, total, tnow)***: devuelve un nuevo modelo teniendo en cuenta el resultado del cuestionario.
    - model: modelo almacenado del estudiante
    - successes: resultados acertados
    - total: total de preguntas
    - tnow: tiempo que ha pasado desde la última vez que se realizó el cuestionario
- ***modelToPercentileDecay(model, percentile, exact)***: devuelve el número de horas que faltan para que el porcentaje llegue al valor deseado.
    - model: modelo almacenado del estudiante
    - percentile: porcentaje de recuerdo deseado
    - exact: para ser más exactos, tiene en cuenta la estadística. Si es falso, devuelve una aproximación para ahorrar en rendimiento.

Para entender cómo funciona, explicaremos un ejemplo:
Ana es una estudiante que realiza un cuestionario de 3 preguntas. Este cuestionario tiene un half-life de 24 horas, es decir, al cabo de 24 horas tendrá un porcentaje de recuerdo aproximado del 50%.

El 5-4-2021, Ana realiza el cuestionario a las 9:00h de la mañana. Como es el primer cuestionario que realiza, llamamos a defaultModel con alpha y beta con valores iguales(4). Así se vería nuestro modelo cuando acaba de realizar el cuestionario. Si llamamos a *predictRecall*, nos devuelve un porcentaje de recuerdo del 100%.

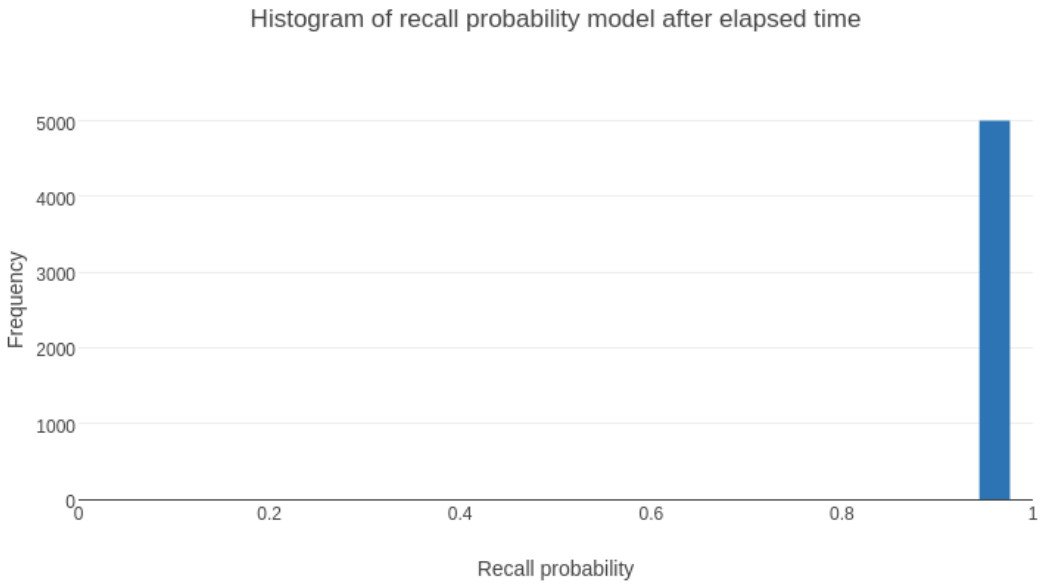

Figura 1: alfa=beta=4, T=24, Telapsed=0

Al cabo de 12 horas si llamamos a *predictRecall* nos devuelve un porcentaje de recuerdo de 69.6%:

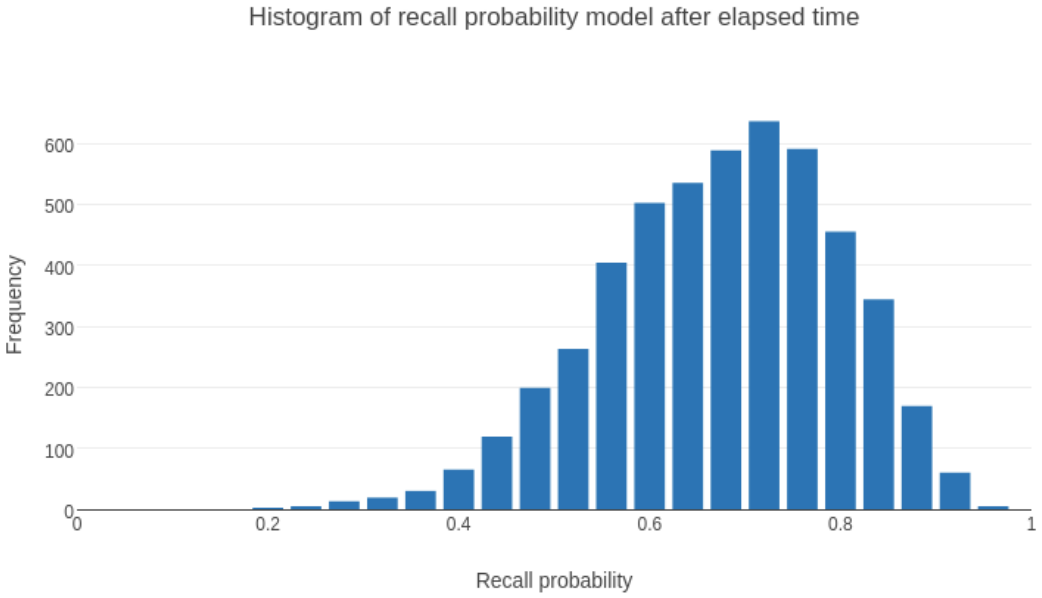

Figura 2: alfa=beta=4, T=24, Telapsed=12

Al cabo de 24 horas el resultado sería distinto, y podemos observar cómo se desplaza el modelo obtenido. A la hora de llamar a *predictRecall* nos devuelve un valor exacto de 50%.

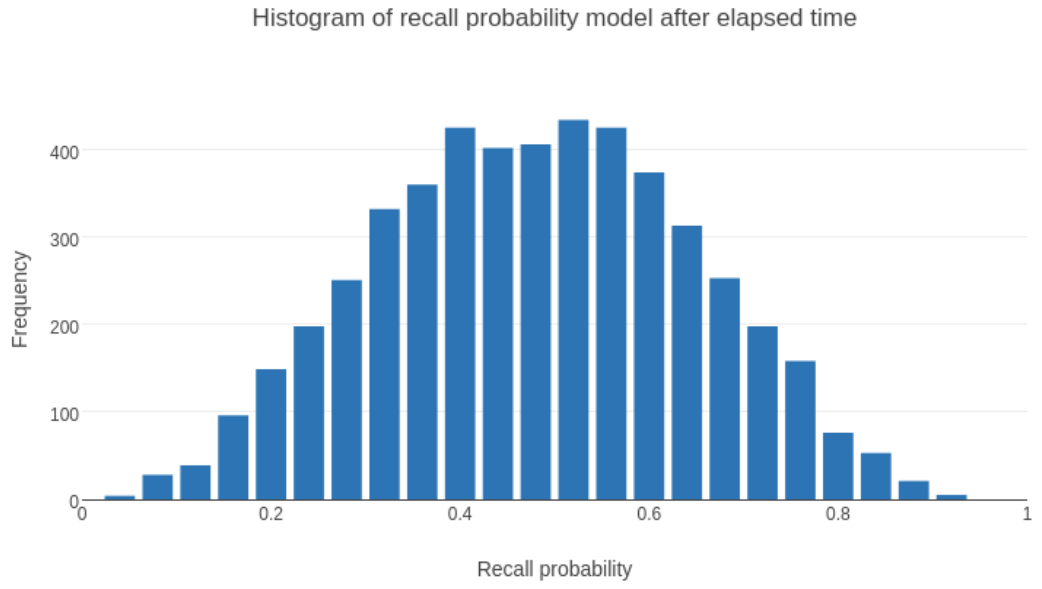

Figura 3: alfa=beta=4, T=24, Telapsed=24

Por último, al cabo de 36 horas *predictRecall* nos devuelve un porcentaje de recuerdo del 36.9%:

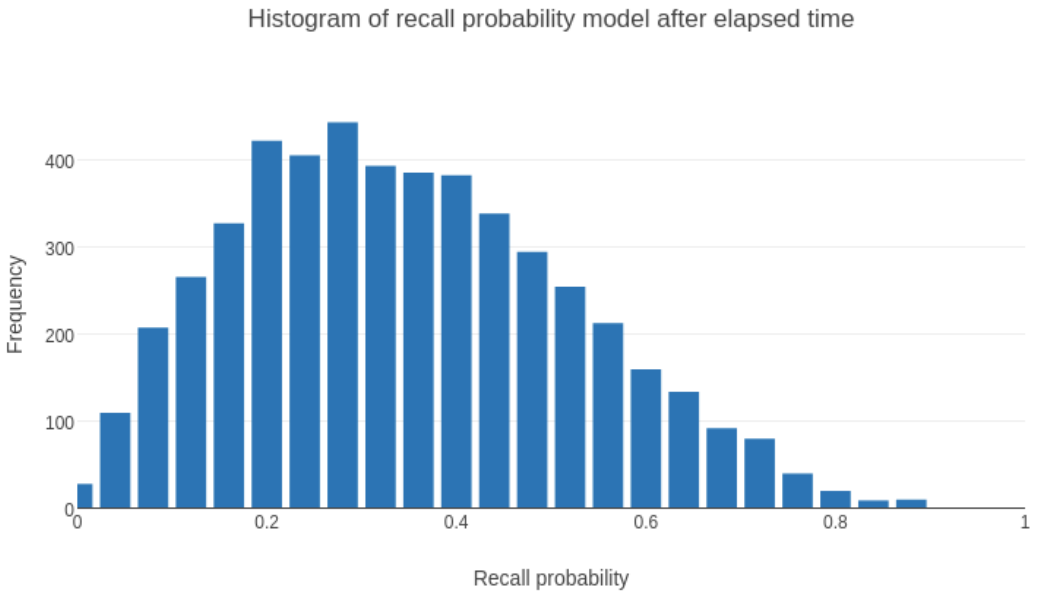

Figura 4: alfa=beta=4, T=24, Telapsed=36

Teniendo en cuenta esto, Ana decide repasar el cuestionario al día siguiente, el 6-4-2021 a las 11:00h de la mañana. Es decir, han pasado 26 horas. Su porcentaje de recuerdo actual es del 47.4%. Si acierta las 3 preguntas, *updateRecall* nos devuelve un nuevo modelo:

alfa	**beta**
7.312	3.999

Con este nuevo modelo, a las 24 horas de haber realizado el cuestionario, tendríamos un porcentaje de recuerdo del 64%. Como podemos observar, hay una diferencia respecto al 50% que teníamos antes. Esto es gracias al repaso realizado, y a medida que realice más repasos los conocimientos se olvidarán con más dificultad.

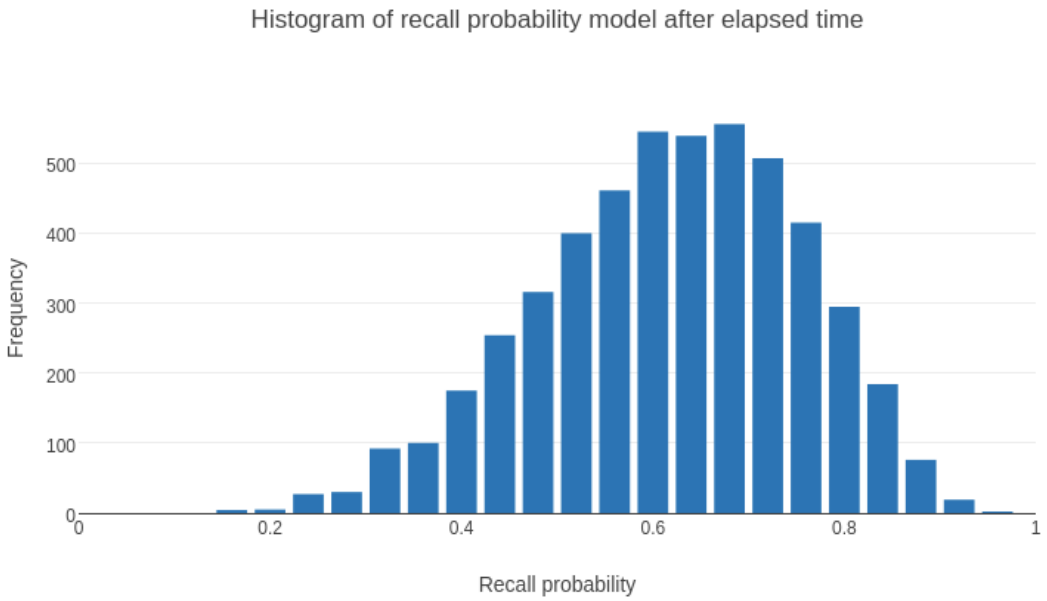

Figura 5: alfa=7.312, beta=3.999, T=24, Telapsed=24

En el caso contrario, donde Ana fallaría las 3 preguntas, a la hora de llamar a updateRecall, esta nos devuelve el modelo siguiente:

alfa	beta
4.066	6.948

Al cabo de 24 horas, si llamamos a la función predictRecall nos devuelve un porcentaje de recuerdo del 36.9%:

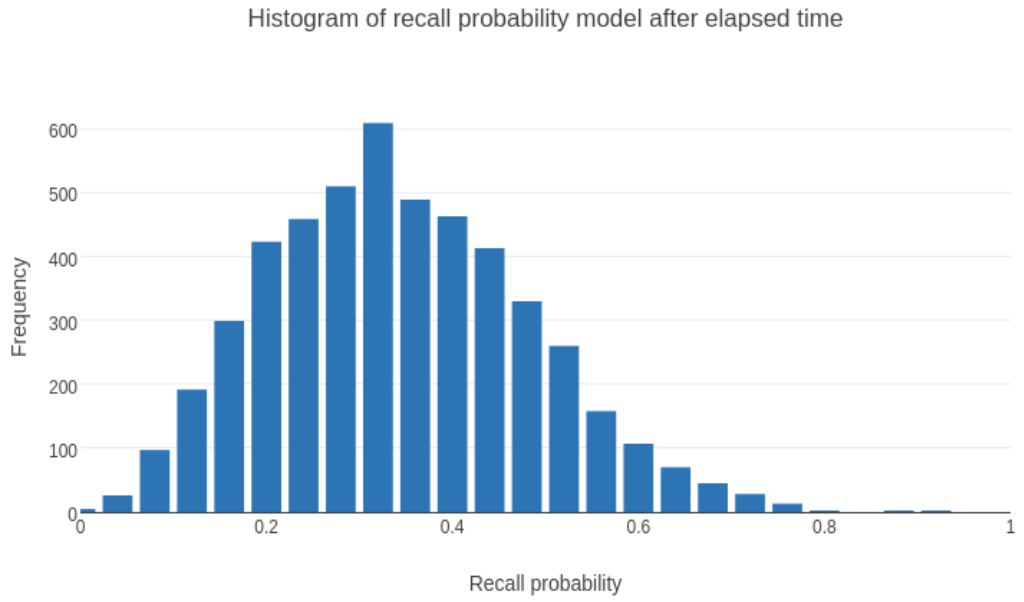

Figura 5: alfa=4.066, beta=6.948, T=24, Telapsed=24