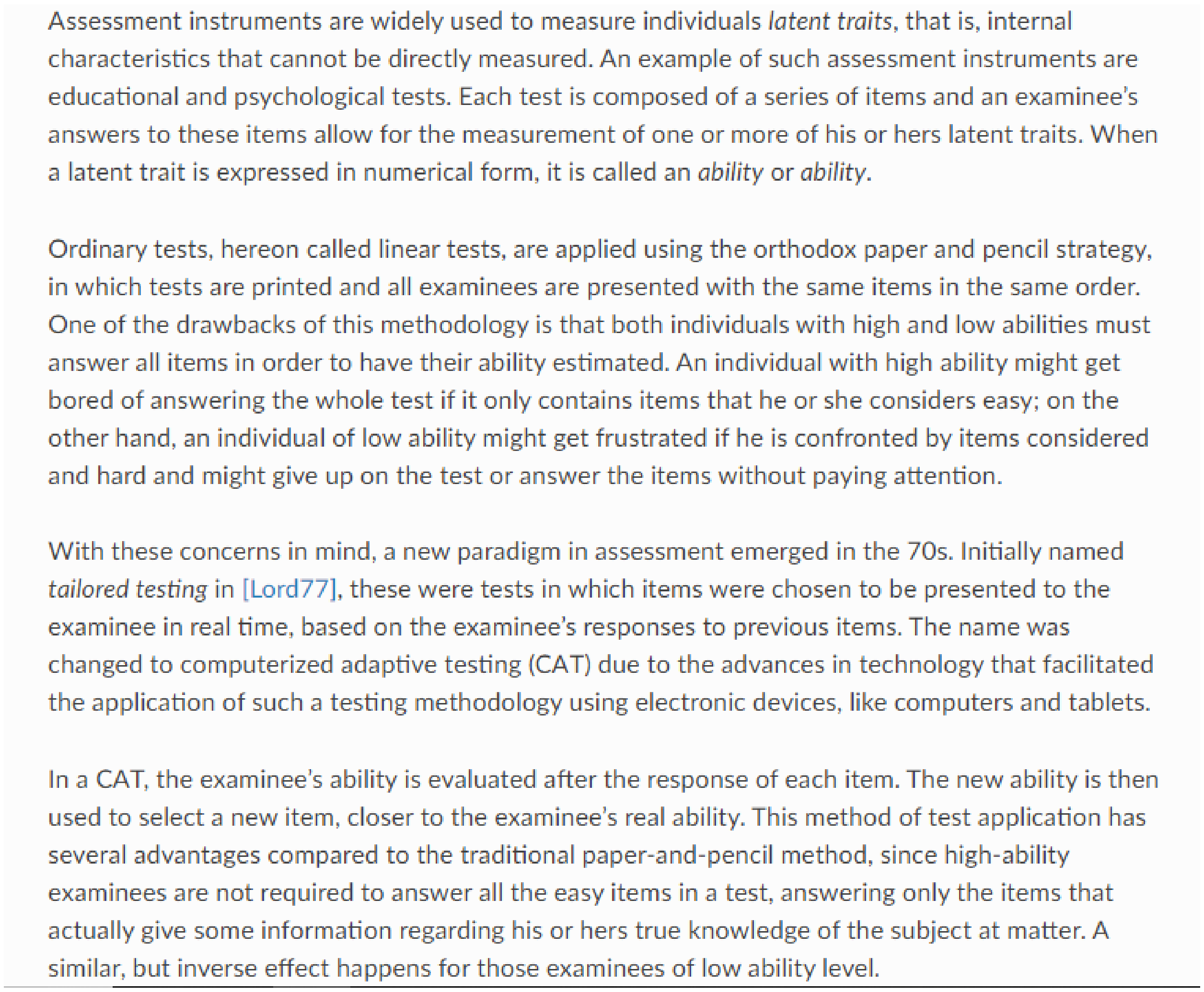
The level update project basically is comprised of four parts(not parts exactly but things one need to understand):

- Computerized adaptive testing(CAT)
- Item Parameter estimation(using girth)
- Question selection(using catsim)
- Level update(using Mathematical model)

Computerized adaptive testing(CAT)



Assessment instruments are widely used to measure individuals *latent traits*, that is, internal characteristics that cannot be directly measured. An example of such assessment instruments are educational and psychological tests. Each test is composed of a series of items and an examinee's answers to these items allow for the measurement of one or more of his or hers latent traits. When a latent trait is expressed in numerical form, it is called an *ability* or *ability*.

Ordinary tests, hereon called linear tests, are applied using the orthodox paper and pencil strategy, in which tests are printed and all examinees are presented with the same items in the same order. One of the drawbacks of this methodology is that both individuals with high and low abilities must answer all items in order to have their ability estimated. An individual with high ability might get bored of answering the whole test if it only contains items that he or she considers easy; on the other hand, an individual of low ability might get frustrated if he is confronted by items considered and hard and might give up on the test or answer the items without paying attention.

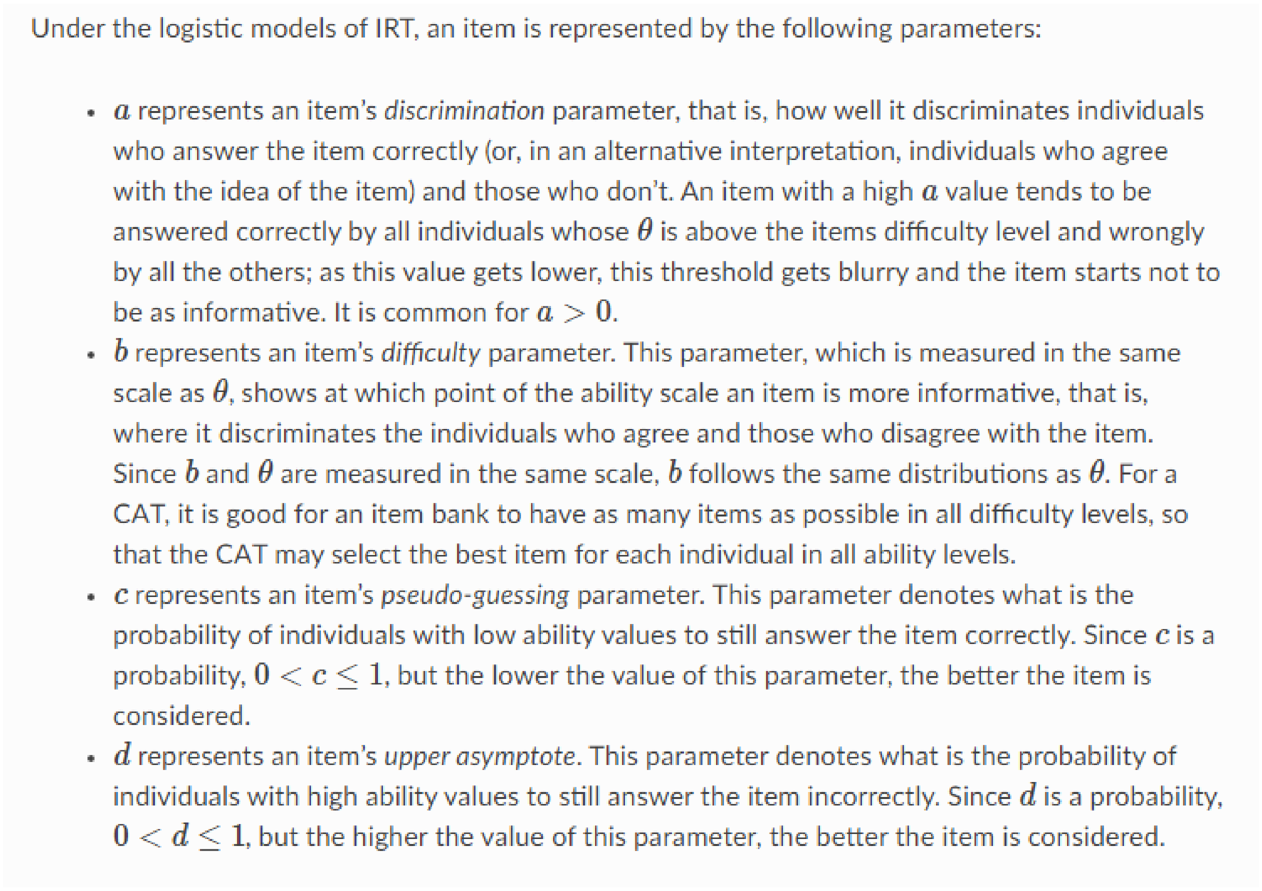
With these concerns in mind, a new paradigm in assessment emerged in the 70s. Initially named *tailored testing* in [Lord77], these were tests in which items were chosen to be presented to the examinee in real time, based on the examinee's responses to previous items. The name was changed to computerized adaptive testing (CAT) due to the advances in technology that facilitated the application of such a testing methodology using electronic devices, like computers and tablets.

In a CAT, the examinee's ability is evaluated after the response of each item. The new ability is then used to select a new item, closer to the examinee's real ability. This method of test application has several advantages compared to the traditional paper-and-pencil method, since high-ability examinees are not required to answer all the easy items in a test, answering only the items that actually give some information regarding his or hers true knowledge of the subject at matter. A similar, but inverse effect happens for those examinees of low ability level.

source: https://douglasrizzo.com.br/catsim/introduction.html

Item Parameter estimation

In CAT each question is treated as an item and each item under logistic models of Item Response Theory has 4 parameters namely discrimination parameter(a), difficulty parameter(b), guessing parameter(c), upper asymptote(d).



Under the logistic models of IRT, an item is represented by the following parameters:

- a represents an item's *discrimination* parameter, that is, how well it discriminates individuals who answer the item correctly (or, in an alternative interpretation, individuals who agree with the idea of the item) and those who don't. An item with a high a value tends to be answered correctly by all individuals whose θ is above the items difficulty level and wrongly by all the others; as this value gets lower, this threshold gets blurry and the item starts not to be as informative. It is common for $a > 0$.
- b represents an item's *difficulty* parameter. This parameter, which is measured in the same scale as θ, shows at which point of the ability scale an item is more informative, that is, where it discriminates the individuals who agree and those who disagree with the item. Since b and θ are measured in the same scale, b follows the same distributions as θ. For a CAT, it is good for an item bank to have as many items as possible in all difficulty levels, so that the CAT may select the best item for each individual in all ability levels.
- c represents an item's *pseudo-guessing* parameter. This parameter denotes what is the probability of individuals with low ability values to still answer the item correctly. Since c is a probability, $0 < c \leq 1$, but the lower the value of this parameter, the better the item is considered.
- d represents an item's *upper asymptote*. This parameter denotes what is the probability of individuals with high ability values to still answer the item incorrectly. Since d is a probability, $0 < d \leq 1$, but the higher the value of this parameter, the better the item is considered.

source: https://douglasrizzo.com.br/catsim/introduction.html

Models are classified based on how many parameters they estimate:

1. 1PL model estimates solely the difficulty parameter
2. 2PL model estimates discrimination and difficulty parameter
3. 3PL model estimates the guessing parameter along with the other two.
4. 4PL models estimate all the four parameters.

Based upon the correctness parameter there are different models available:

1. Dichotomous models: Where the user's response is evaluated if the response is correct/incorrect i.e 0 or 1 marks given.
2. Polytomous models: When partial marking is given if response is partially correct.

In our case it's Dichotomous models we use since correctness is 0 or 1.

For estimating the parameters, a(discrimination), b(difficulty) and considering c = 0.25 since we have 4 options and assuming all those options are equally correct we have used two approaches-

By using Girth-

To estimate the parameters of the questions having enough number of responses. The IRT models just consider the responses(True/False) to evaluate parameters. Girth is a python package for estimating item response theory (IRT) parameters. There are different types of IRT models to estimate parameters and these can be classified into:

1. Basic IRT Models implemented using Girth
   a. Joint Maximum Likelihood
   b. Marginal Maximum Likelihood
2. Bayesian IRT Models implemented in Girth_mcmc
3. And there are also some other deep IRT models developed recently.

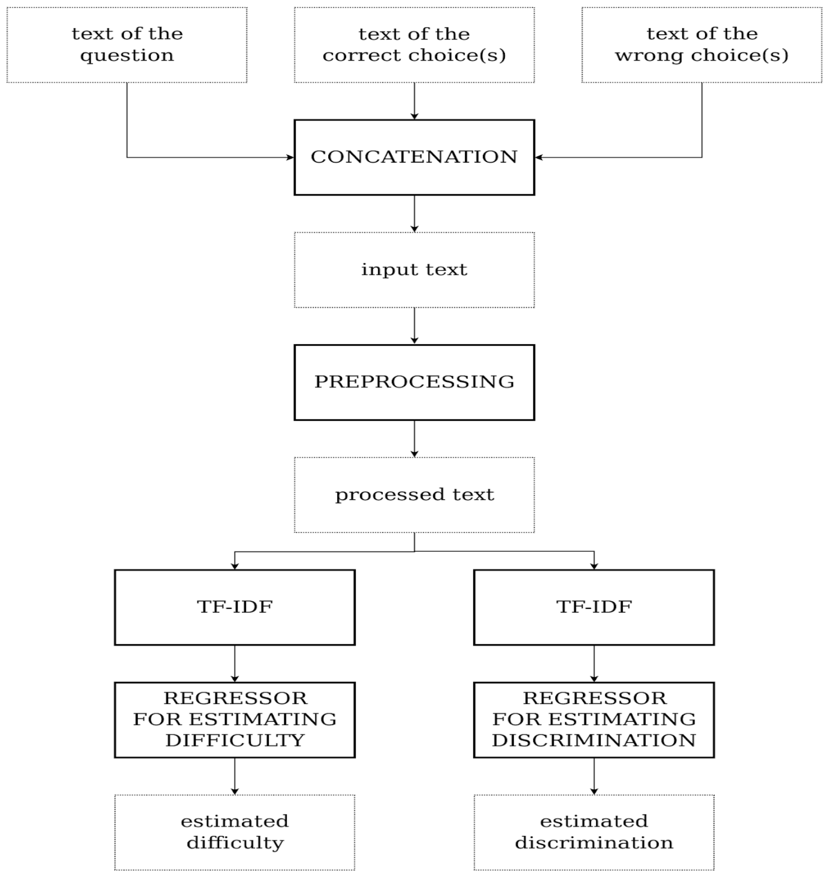
Each of the four parameter logistic models(1 to 4PL) will be using one of the above functions/models(jml, mml, bayesian …..) to estimate the parameters. In our project we confined ourselves to using 2PL with Marginal Maximum likelihood which is implemented as 2ppl_mml in the girth package.

To estimate stable parameters we need good data. Since we are using 2PL model stability might be achieved if we have sufficient number of questions with number of responses >= 500. Every now and then we can just look how the graph looks after parameters are scaled and if it looks meaningful then we can use the estimated parameters to calculate the information such that the questions recommended to the user will be optimal to his level i.e neither too easy nor too tough.

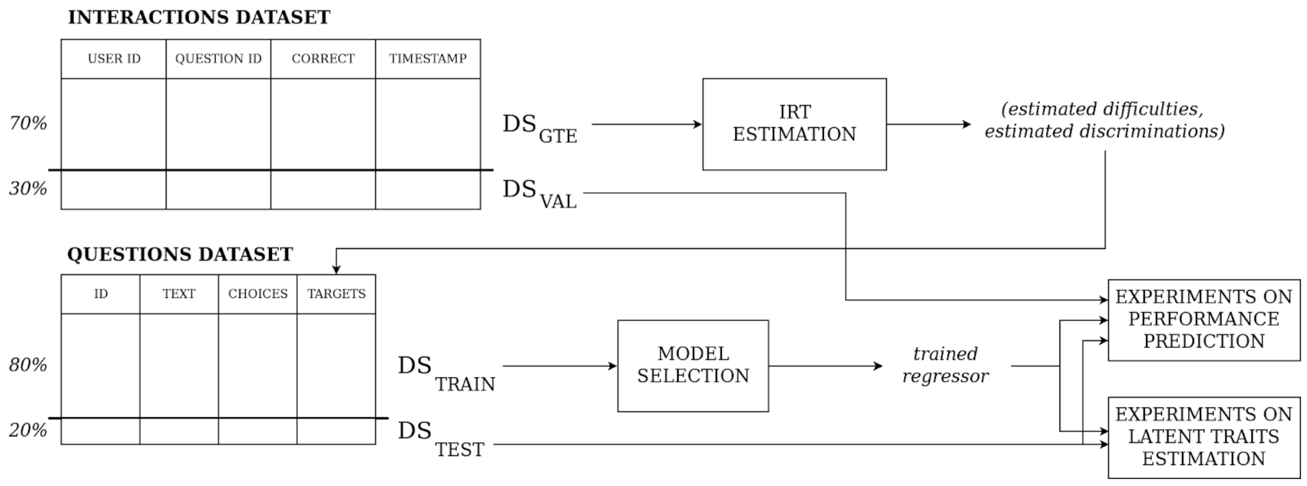
By using NLP-

To estimate the parameters of questions where there aren't enough responses such as newly designed questions which are not recommended to the user yet. The basic idea is to use the parameters estimated using girth for the questions with enough responses and use those as ground truth labels(in training set) and estimate the parameters of the questions where there aren't enough responses. This idea is basically from a research paper. In the project we haven't directly used the code of theirs but just the basic idea that parameters can be estimated using NLP from text.

Before jumping into modeling, the text used here is the question text or question + correct option text or question + options text. This is a hyperparameter left to decide after trying out all the three and looking at the results we can decide which works better. Then comes preprocessing of text and after preprocessing we use the TF-IDF model to extract features from text. We can also use wordnet or some other high level models to improve results. Separate TF-IDF models will be used for discrimination estimation and difficulty estimation with max_features parameter left as a hyperparameter to decide for both of these. We have tried using different values for the max_features parameter and the range of 100-300 looks okay for both dicrimination and difficulty estimation. Now the modeling part includes a RandomForest regressor model one for discrimination estimation and other for difficulty estimation. Since the results are not so good we haven't tried to tune the hyperparameters of the model as the training set consists of just around 200 questions if we filter out questions with at least 50 responses. This won't be enough for the model to learn and predict our parameters accurately.


text of the question
text of the correct choice(s)
text of the wrong choice(s)
CONCATENATION
input text
PREPROCESSING
processed text
TF-IDF
TF-IDF
REGRESSOR FOR ESTIMATING DIFFICULTY
REGRESSOR FOR ESTIMATING DISCRIMINATION
estimated difficulty
estimated discrimination

This is how the basic idea looks: first we estimate the parameters using girth and then use that data to estimate parameters of questions without enough responses from text.



So whenever we have enough data to get good results the hyperparameters of a model must be improved and even better models can be used such as boosting ones and also improving those hyperparameters to get accurate results.

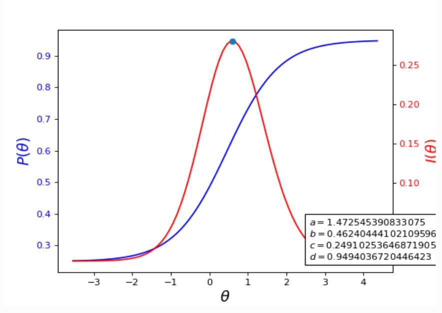
Final code:
https://colab.research.google.com/drive/1iyGtQ30KH7DmHXr0fRvOkPVHr8KWZcvt
After running all the cells the est_params.csv file will be saved with all the necessary columns in it.

Question selection

The only part where we use catsim is during question selection as it provides various algorithms.
CAT has inbuilt selectors which makes us able to select an item that is most appropriate for that estimate for the user. Technically, this is done by selecting the item with the greatest information at that level of user.
Information provided by each questions based on question's parameters are calculated as -

$$P(X_i = 1|\theta) = c_i + \frac{d_i - c_i}{1 + e^{-a_i(\theta - b_i)}}.$$

$$I_i(\theta) = \frac{a^2[(P(\theta) - c)]^2[d - P(\theta)]^2}{(d - c)^2(1 - P(\theta))P(\theta)}.$$



Where P is basically the probability of a user with a given level to answer question item i correctly, given the item parameters.
And I is Information provided by the Question item.

Selectors that we can use-

RandomesqueSelector - At every step, this selector will randomly choose the questions from the n most informative questions in the item bank.
Parameters - Question bank,current user level,already recommended item, user id, n - number of items in set containing most informative items.
return - Selected Question id or the list of question ids if it recommends 3/4/5 questions together.
For applying this selector online, after each step we have to compute the information provided by all the questions of the particular topic and arrange them in decreasing order. And recommends any random question(or set of questions) from top 10/15 questions after sorting .If number of question are too high for any particular topic , we can just compute the information of the question having difficulty level in the range (current user level-0.5,current user level+0.5)
Input data format- should Include questions ID,difficulty parameter(given and estimated both), descrimination parameter
https://colab.research.google.com/drive/1Upsu-noTI54AWRKGnOE8y9Xc13NSTZB3?usp=sharing

Urryselector -
Selector that returns will return the Question ID whose difficulty parameter is closest to the examinee's ability
Parameters - Question bank(referred as item bank in CATsim) ,current user level,already recommended item and user id
Return - Selected Question id

Level update

We do have some level estimation techniques when a user answers a question in catsim but we prefer using mathematical model instead as this is flexible and convenient to tweak however we want.

Summary -
Correctness is whether the user got the question correct/incorrect/skipped i.e 1,-1,0;

X1=(Correctness +1)/2
X2=(Correctness -1)/2

Correctness Factor:

Update¹ = (X1*(5-init level)/5 * 0.7*(alpha) + X2*(init level)/5 * 0.7*(alpha))
Where init level = Initial Level of the user

$$\boxed{\text{Alpha} = \text{x}^* \frac{(\sqrt{e^{(x^2)/37} - 1})}{5}}$$

X = value obtained by the user after answering a question i.e if level 4 question is correct then 4x1=4, if level 5 question is incorrect then (5-6)=-1

To add factor based on past responses-
Wrong_responses = number of wrong responses in last 5 questions
Correct_responses = number of correct responses in last 5 questions
If update¹>0, **update = update¹*(30-wrong_responses)/30**
Else, **update = update¹*(30-correct_responses)/30**

Time Factor:
Update = (X1*(5-init level)/5 *0.3*beta + X2*(init level/5) * 0.3*(beta1))
Where beta 1 is:

$$\text{Beta1} = (e^{-\frac{average\ time}{time\ taken\ *|x|}})$$

Where |x| is the difficulty of the question.