**Score and Threshold Manipulation**

As discussed in our introduction, our model outputs the GIFs that match the message written by the user and cross a certain threshold T (predefined). This assures that the user is not thrown each and every GIF that is even remotely related to its message. To assure this we need to assign a score to each GIF that reflects the user likeliness for it.   
  
The above description seems to be a very simple and full proof idea to output accurate and limited results, but also raises many important questions on how to assign multiple scores and a universal threshold for the program. The overall problem can be broken down to a set of possible questions that need to be answered.

* What is our initial threshold and score for a new user?
* Do we need to change the score and threshold over time looking at user’s usage?
* Does the score change equally for all scenarios?
* What if an unrealistic number of GIFs (say 200) cross the threshold? Do we output all?
* Is no output better than no output?
* Is threshold/change in threshold same for a new and an old user?
* What if new GIFs are added over time?

There are multiple questions that need to be answered while designing the mechanism. This we have shown below all the possible ideas and why/why not we selected them for our project.  
  
**Rules to assign threshold T**

* The threshold should be lower than the minimum score assigned.
* The threshold can be selected keeping in mind the tradeoff between accuracy and variety of results.
  + A higher threshold will ensure accuracy.
  + A lower threshold will ensure more and more GIFs clear the threshold to be showcased as an option to user, this ensures a huge variety.

**Solution 1: Predefined score and constant threshold**  
  
Probably the most simple and old school approach. Here we assign scores (s) to each GIF (si) depending on how popular they are and keep escalating them every time the user selects them.   
  
Thus before selecting a GIF i, we need to make sure and on every selection of GIF i, , where alpha is the predefined increment.   
  
**Possible drawbacks to solution 1**  
Is our assignment of scores accurate? For something to be popular and liked be same for all users?   
  
GIFs are awarded for being selected, but what about the GIFs with a very high score (say 0.95) and not liked by the user? Shouldn’t we penalize GIFs that are constantly ignored by the user?   
  
**Solution 2: Score manipulation with constant threshold**   
  
Very similar to the above approach but the now even scores of GIFs not selected will be changed to develop a more accurate model.   
  
   
  
**Possible drawbacks to solution 2**  
  
Even though are approach of awarding and penalizing selected and not selected GIFs logically correct, does that mean we should treat both scenarios equally?   
  
Not selecting a GIF doesn’t mean user dislikes it as much as he/she likes a GIF that has been selected.   
  
**Solution 3: Unequal score manipulation with constant threshold**  
It may happen that the user may like GIF but didn’t select it just because the other option was better. To overcome the above problem we need to award and penalize scores at different rates. Rather than assigning different coefficients, we try to achieve the solution with just alpha, (the solution was inspired by AdaBoost).   
  
**Possible drawback for solution 3**  
As discussed in the beginning, there will be a scenario where an unrealistic number of GIFs satisfy the threshold or say no GIFs do.   
  
What if the threshold initialized is so big (to develop an accurate model) that a few or no GIFs qualify?   
  
What if the threshold initialized is small (to develop a model that produces a huge variety) that a lot of GIFs qualify?   
  
With so many problems revolving around the term ‘threshold’, do we need it at all?   
  
**Solution 4: Score manipulation without threshold**  
This solution makes no changes to the score manipulation as discussed above but elimates the entire need of Threshold T. Instead we propose a new term ‘n’ which is a user option of how many options he/she desires.   
  
Thus the solution proposes to output the first n GIFs with highest scores, output .  
  
**Possible Drawbacks to solution 4**  
  
The above solution seems complete. But it considers that there will always be n relevant solution.   
  
What if the dataset has only n-x relevant solutions? The algorithm will be forced to output options with very low score just to justify n options.   
  
Well to solve the above problem, we had threshold. So does that mean we need threshold? But what about the problems discussed in Solution 3?   
  
**Solution 5: Constant threshold with n first solutions**.   
  
To solve the problems of Solution 3 and 4 we need to combine them both. To do so we need to output the options that are above assigned threshold T and also satisfies the users request of first n outputs.   
  
**Possible Drawbacks to solution 5**  
  
Now that we have seemed to have answered almost all questions stated in the beginning and the outputs are also under check by two limits, T and n.   
  
But we are still left with a few. After all the above mentioned rules, can we say that our model is actually developing? Do we have a measurement to tell how much our model has learned? How can we deliver limited, yet accurate results?   
  
**Solution 6: Dynamic threshold**  
Do we need the threshold to be constant always? Can’t we use threshold T as a measurement of how much our model has progressed?   
  
We can, by increasing out threshold over time with the changes in the score. As explained in the beginning, a threshold T is initialized as a tradeoff between accuracy and variety of outputs. We also had one more constraint, T should be less than the scores originally initialized.   
  
But can’t we say with confidence that T doesn’t have to be less than 0.5 anymore?   
  
In an attempt to achieve higher accuracy and user satisfaction we can change the threshold after every iteration.   
  
, where   
  
Here Beta is a predefined constant that dictates the rate of change in threshold with respect to how much our model has been used by the user.  
  
After this alteration we can successfully say that the model is getting better with every passing hit from the user.   
  
**Possible drawbacks from Solution 6**  
With all these changes we can assume that we have corrected majority of the possible drawbacks.   
  
But the last two questions still remain unanswered. Does the rate of change in scores remain same for a new and an existing user?

**Solution 7: Dynamic Alpha**  
Just how we created a dynamic T that changed according to how much the data is manipulated, shouldn’t the same be achieved with Alpha?   
  
The above proposition makes sense because we want to award and penalize more when the user is new to understand the learn quickly. But once the model is ready (or almost ready), do we need to keep changing the score with equal rate?   
  
But like in threshold we can’t be changing Alpha value linearly. We need to change the rate in a way that slows down with time, i.e. change Alpha quickly in the first 100 rounds but slowly after 1000 iterations. To do so we need apply negative sigmoid function, this will be  
  
**Possible drawbacks from Solution 7**  
With all the above regulations we are narrowing our model to a very perfect stage. This may push our model into a scenario where a certain ignored GIFs will be downscaled to such a degree that it will never show up again.  
  
We need to ensure that the model is not trained to such an extent that it shows options that repeatedly used by the user. This is as good as selecting GIFs from a local library. We also need to give a fair chance to newly added GIFs even to an old model.   
  
**Solution 8: Randomness**With all the above solutions provided above we simply have to add an element of randomness, which outputs relevant GIFs with lower scores. But we need to ensure that the randomness is not increased at a level that the user feels that the model is failing. To do so we need to add one more element ‘r’ which is a predefined element that decides the number of random GIFs provided to the user.   
  
**Possible Drawbacks**  
We have designed the entire model, but not yet established a base case. The scenario when multiple GIFs with same score are encountered. This is an obvious situation for the first iteration when all scores are equally initialized to 0.5.   
  
**Solution 9: Final Result**  
Keeping everything similar to Solution 8, we now only need to resolve conflicts with equal scores. In such a scenario we sort all the GIFs with equal scores in decreasing order, such that the first GIF will have the maximum similar/equal tokens in the metadata w.r.t the input. After which the algorithm will randomly select among the GIFs with highest similar tokens to justify the above discussed constraints.