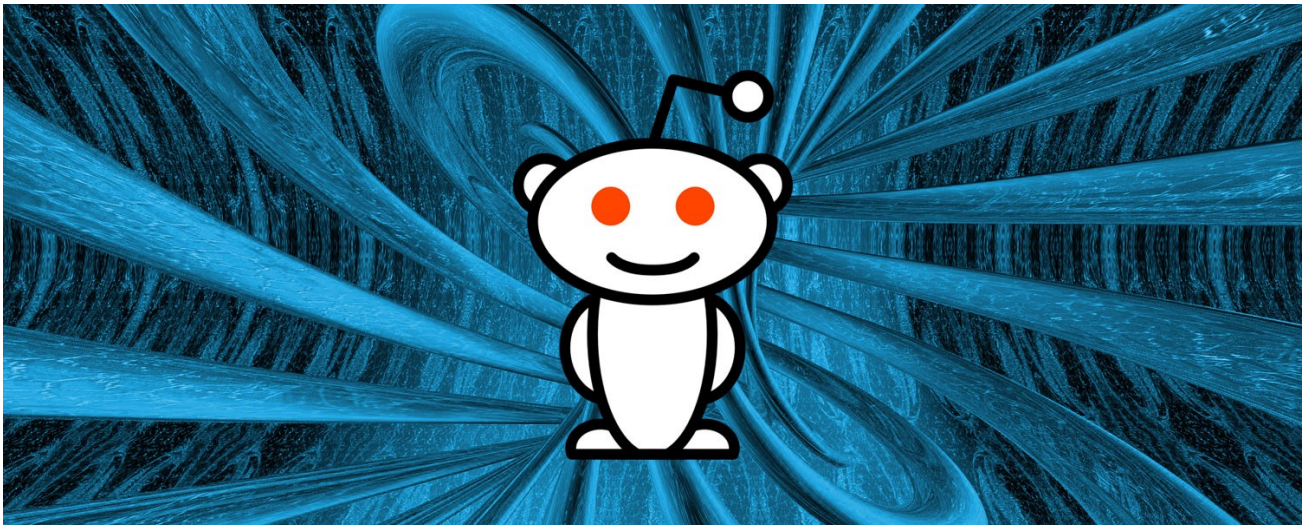


How Reddit ranking algorithms work

This is a follow up post to How Hacker News ranking algorithm works. This time around I will examine how Reddit's story and comment rankings work.

The first part of this post will focus on how are Reddit stories ranked? The second part of this post will focus on comment ranking, which does not use the same ranking as stories (unlike Hacker News). Reddit's comment ranking algorithm is quite interesting and the idea guy behind it is Randall Munroe (the author of xkcd!)

Digging into the story ranking code

Reddit is open sourced and the code is freely available. Reddit is implemented in Python and their code is located here. Their sorting algorithms are implemented in Pyrex, which is a language to write Python C extensions. They have used Pyrex for speed reasons. I have rewritten their Pyrex implementation into pure Python since it's easier to read.

The default story algorithm called the hot ranking is implemented like this:

```
# Rewritten code from /r2/r2/lib/db/_sorts.pyx

from datetime import datetime, timedelta
from math import log

epoch = datetime(1970, 1, 1)

def epoch_seconds(date):
    td = date - epoch
    return td.days * 86400 + td.seconds + (float(td.microseconds)
/ 1000000)

def score(ups, downs):
    return ups - downs

def hot(ups, downs, date):
    s = score(ups, downs)
    order = log(max(abs(s), 1), 10)
    sign = 1 if s > 0 else -1 if s < 0 else 0
    seconds = epoch_seconds(date) - 1134028003
    return round(sign * order + seconds / 45000, 7)
```

In mathematical notation the hot algorithm looks like this:

Given the time the entry was posted A and the time of 7:46:43 a.m. December 8, 2005 B, we have t_s as their difference in seconds

$$t_s = A - B$$

and x as the difference between the number of up votes U and the number of down votes D

$$x = U - D$$

where $y \in \{-1, 0, 1\}$

$$y = \begin{cases} 1 & \text{if } x > 0 \\ 0 & \text{if } x = 0 \\ -1 & \text{if } x < 0 \end{cases}$$

and z as the maximal value, of the absolute value of x and 1

$$z = \begin{cases} |x| & \text{if } |x| \geq 1 \\ 1 & \text{if } |x| < 1 \end{cases}$$

we have the rating as a function $f(t_s, y, z)$

$$f(t_s, y, z) = \log_{10} z + \frac{y t_s}{45000}$$

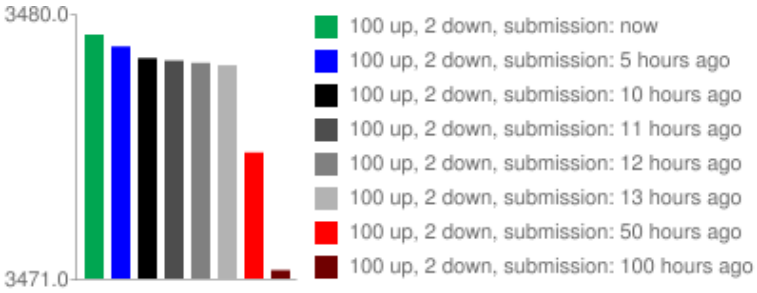
Effects of submission time

Following things can be said about submission time related to story ranking:

> Submission time has a big impact on the ranking and the algorithm will rank newer stories higher than older

> The score won't decrease as time goes by, but newer stories will get a higher score than older. This is a different approach than the Hacker News's algorithm which decreases the score as time goes by

Here is a visualization of the score for a story that has same amount of up and downvotes, but different submission time:

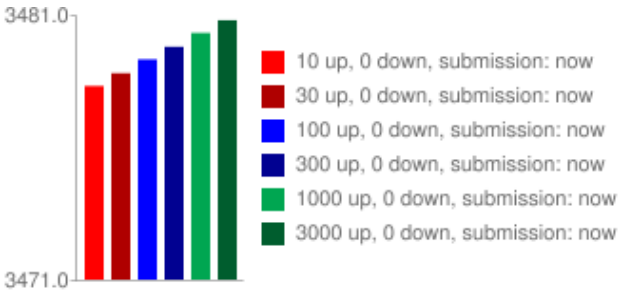


The logarithm scale

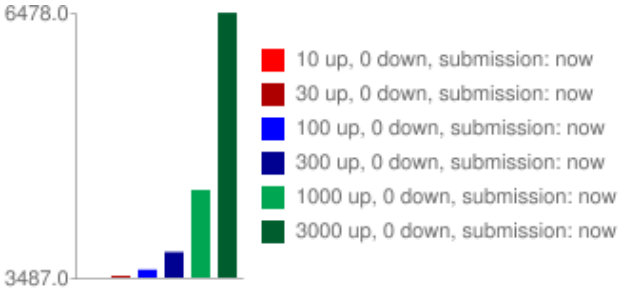
Reddit's hot ranking uses the logarithm function to weight the first votes higher than the rest. Generally this applies:

> The first 10 upvotes have the same weight as the next 100 upvotes which have the same weight as the next 1000 etc…

Here is a visualization:



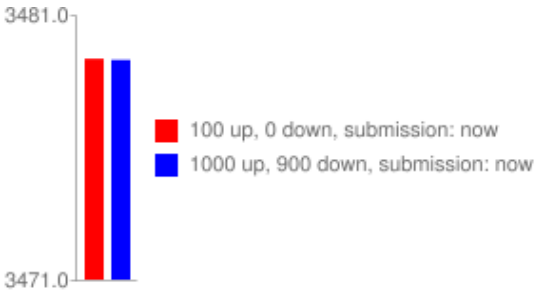
Without using the logarithm scale the score would look like this:



Effects of downvotes

Reddit is one of the few sites that has downvotes. As you can read in the code a story's "score" is defined to be:

The meaning of this can be visualized like this:



This has a big impact for stories that get a lot of upvotes and downvotes (e.g. controversial stories) as they will get a lower ranking than stories that just get upvotes. This could explain why kittens (and other non-controversial stories) rank so high :)

Conclusion of Reddit's story ranking

> Submission time is a very important parameter, generally newer stories will rank higher than older
>
> The first 10 upvotes count as high as the next 100. E.g. a story that has 10 upvotes and a story that has 50 upvotes will have a similar ranking
>
> Controversial stories that get similar amounts of upvotes and downvotes will get a low ranking compared to stories that mainly get upvotes

How Reddit's comment ranking works

Randall Munroe of xkcd is the idea guy behind Reddit's best ranking. He has

written a great blog post about it:

> reddit's new comment sorting system

You should read his blog post as it explains the algorithm in a very understandable way. The outline of his blog post is following:

> Using the hot algorithm for comments isn't that smart since it seems to be heavily biased toward comments posted early
>
> In a comment system you want to rank the best comments highest regardless of their submission time
>
> A solution for this has been found in 1927 by Edwin B. Wilson and it's called "Wilson score interval", Wilson's score interval can be made into "the confidence sort"
>
> The confidence sort treats the vote count as a statistical sampling of a hypothetical full vote by everyone—like in an opinion poll
>
> How Not To Sort By Average Rating outlines the confidence ranking in higher detail, definitely recommended reading!

Digging into the comment ranking code

The confidence sort algorithm is implemented in _sorts.pyx, I have rewritten their Pyrex implementation into pure Python (do also note that I have removed their caching optimization):

```
# Rewritten code from /r2/r2/lib/db/_sorts.pyx

from math import sqrt

def _confidence(ups, downs):
    n = ups + downs

    if n == 0:
        return 0

    z = 1.0 #1.0 = 85%, 1.6 = 95%
    phat = float(ups) / n
    return sqrt(phat+z*z/(2*n)-z*((phat*(1-phat)+z*z/(4*n))
/n))/(1+z*z/n)

def confidence(ups, downs):
    if ups + downs == 0:
        return 0
    else:
        return _confidence(ups, downs)
```

The confidence sort uses Wilson score interval and the mathematical notation looks like this:

$$\frac{\hat{p} + \frac{1}{2n}z^2_{1-\alpha/2} \pm z_{1-\alpha/2}\sqrt{\frac{\hat{p}(1-\hat{p})}{n} + \frac{z^2_{1-\alpha/2}}{4n^2}}}{1 + \frac{1}{n}z^2_{1-\alpha/2}}$$

In the above formula the parameters are defined in a following way:

p is the observed fraction of positive ratings

n is the total number of ratings

zα/2 is the (1-α/2) quantile of the standard normal distribution

Let's summarize the above in a following manner:

The confidence sort treats the vote count as a statistical sampling of a hypothetical full vote by everyone

The confidence sort gives a comment a provisional ranking that it is 85% sure it will get to

The more votes, the closer the 85% confidence score gets to the actual score

Wilson's interval has good properties for a small number of trials and/or an extreme probability

Randall has a great example of how the confidence sort ranks comments in his blog post:

> If a comment has one upvote and zero downvotes, it has a 100% upvote rate, but since there's not very much data, the system will keep it near the bottom. But if it has 10 upvotes and only 1 downvote, the system might have enough confidence to place it above something with 40 upvotes and 20 downvotes—figuring that by the time it's also gotten 40 upvotes, it's almost certain it will have fewer than 20 downvotes. And the best part is that if it's wrong (which it is 15% of the time), it will quickly get more data, since the comment with less data is near the top.

Effects of submission time: there are none!

The great thing about the confidence sort is that submission time is

irrelevant (much unlike the hot sort or Hacker News's ranking algorithm). Comments are ranked by confidence and by data sampling—— i.e. the more votes a comment gets the more accurate its score will become.

Visualization

Let's visualize the confidence sort and see how it ranks comments. We can use Randall's example:

As you can see the confidence sort does not care about how many votes a comment have received, but about how many upvotes it has compared to the total number of votes and to the sampling size!

Application outside of ranking

Like Evan Miller notes Wilson's score interval has applications outside of ranking. He lists 3 examples:

> Detect spam/abuse: What percentage of people who see this item will mark it as spam?
>
> Create a "best of" list: What percentage of people who see this item will mark it as "best of"?
>
> Create a "Most emailed" list: What percentage of people who see this page will click "Email"?

To use it you only need two things:

> the total number of ratings/samplings
>
> the positive number of ratings/samplings

Given how powerful and simple this is, it's amazing that most sites today use the naive ways to rank their content. This includes billion dollar companies like Amazon.com, which define Average rating = (Positive ratings) / (Total

ratings).

Conclusion

I hope you have found this useful and leave comments if you have any questions or remarks.

Happy hacking as always!

. . .

Originally published at amix.dk.