WikiRater – About

I've started a project to create an algorithm to rate articles on Wikipedia, and you can help! My hope is to take user supplied rating and content and create an algorithm that can reliably rate how interesting an article is.

I realize "interesting" is a very ambiguous word, and that's mostly on purpose. I want to find the articles on Wikipedia that you are most likely to enjoy reading, if presented randomly to you. If you're curious I suggest you read the "history" section to learn how I came up with this idea.

The WikiRater comes in two parts. The Algorithm and User generated content. If you're interested in the algorithm, please click the above link. My Algorithm can automatically create a best guess rating on each Wikipedia article, it may be close, or it may be way off, it all depends on how I ultimately tune and tweak it. I'm not trying to use any fancy Artificial Intelligence here, just trying to find correlations between certain pieces of data that I can easily collect and how interesting the article is to most people.

This is where you come in.

If I have enough ratings from real people I can start to see how the different pieces of data correlate to real ratings. Does it matter more if an article is very long? What about if it is edited often? What about the total number of links to or from the article?

Once I have a reasonable amount of data I'll use your votes to tune The Algorithm to be able to get in the same ballpark as your average votes, this way you can see how interesting an article might be before you read it!

History

For a long time I had my browser's homepage set to the random Wikipedia article: <http://en.wikipedia.org/wiki/Special:Random>. After a while I got pretty disappointed by the quality of the article that came up. (For example, I just clicked it now and got: <http://en.wikipedia.org/wiki/List_of_Newark_Bears_(AFL)_players> currently rates a 2). I was complaining about this to a friend and he said "why don't you fix it?" After a bit of thinking "why don't I fix it" I set out to create an algorithm.

I quickly discovered collecting the data was the easy part, but I had seriously no idea what to do with it or how it correlated to what was actually interesting I decided to enlist my the "crowd" for feedback. I appreciate you spending time with this, you're helping to make the world a better place.

The Algorithm

For all you nerds out there, here's the current state of the algorithm:

Sorry, it's late, so I don't have much time to explain what this means if you don't read source, but suffice it to say, it's rough, and is in need of some serious tuning.

///rating = (linksInWeight \* linksIn) +

/// (linksOutWeight \* linksOut) +

/// (minutesSinceLastEditWeight \* minutesSinceLastEdit \* -1) +

/// (totalEditsWeight \* totalEdits) +

/// (totalMinorEditsWeight \* totalMinorEdits) +

/// (isFeaturedWeight \* isFeatured) +

/// (totalLengthWeight \* totalLength) +

/// (viewsInLast30DaysWeight \* viewsInLast30Days)

int linksToWeight = 10;

int linksFromWeight = 5;

int minutesSinceLastEditWeight = 0;

int totalEditsWeight = 5;

int totalMinorEditsWeight = 2;

int isFeaturedWeight = 300;

int totalLengthWeight = 1;

int viewsInLast30DaysWeight = 2;

int CurLinksTo = GetLinksTo();

int CurLinksFrom = GetLinksFrom();

int CurMinutesSinceLastEdit = GetMinSinceLastEdit();

int CurTotalEdits = GetTotalEdits();

int CurTotalMinorEdits = GetTotalMinEdits();

int CurIsFeatured = GetIsFeatured();

int CurTotalLength = GetTotalLength();

int CurViewsInLast30Days = GetViewsInLast30Days();

int internalRating = (linksToWeight \* CurLinksTo) +

(linksFromWeight \* CurLinksFrom) +

(minutesSinceLastEditWeight \* CurMinutesSinceLastEdit) +

(totalEditsWeight \* CurTotalEdits) +

(totalMinorEditsWeight \* CurTotalMinorEdits) +

(isFeaturedWeight \* CurIsFeatured) +

(totalLengthWeight \* CurTotalLength) +

(viewsInLast30DaysWeight \* CurViewsInLast30Days);

//Thresholds

//1: 0 - 300

//2: 301 - 1000

//3: 1001 - 2500

//4: 2501 - 4500

//5: 4501 - 7000

//6: 7001 - 10000

//7: 10001 - 12000

//8: 12001 - 15000

//9: 15001 - 20000

//10: 20001 -

if (internalRating <= 300)

rating = 1;

else if (internalRating > 300 && internalRating <= 1000)

rating = 2;

else if (internalRating > 1000 && internalRating <= 2500)

rating = 3;

else if (internalRating > 2500 && internalRating <= 4500)

rating = 4;

else if (internalRating > 4500 && internalRating <= 7000)

rating = 5;

else if (internalRating > 7000 && internalRating <= 10000)

rating = 6;

else if (internalRating > 10000 && internalRating <= 12000)

rating = 7;

else if (internalRating > 12000 && internalRating <= 15000)

rating = 8;

else if (internalRating > 15000 && internalRating <= 20000)

rating = 9;

else if (internalRating <= 20000)

rating = 10;