Community-based scoring of metadictionary terms

Christopher Patton

July 18, 2013

Abstract

The *SeaIce* metadictionary classifies terms in three categories. *Vernacular* terms are those for which there is no community consensus. The debate may be centered around the term's definition or the usefulness of the term itself. The owner has the oppurtunity to modify the defintion or term itself; therefore, vernacular terms are subject to change as the debate proceeds. *Canonical* terms are those whose definitions stabilize over time and upon which the community largely agrees. Finally, *deprecated* terms are stable terms that the community has largely deemed to not be useful, or for which there is a better alternative. The goal of *SeaIce* is to evolve a set of stable, canonical terms within the context of a reputation-based social ecosystem by promoting vernacular terms to the canon (shorthand meaning the canonical class) and deprecating those deemed not useful. I propose a scoring and classification system that automates this process for the online metadictionary. In this document I refer to community as the body of *SeaIce* users.

1 Scoring

The first step is to quantify consensus. The most obvious definition is the percentage of the community that finds the term and its definition useful. We'll call this score. Let u be the number of up-votes and d be the number of down-votes. If every user casts a vote, then a term's score is simply

$$S = \frac{u}{u+d}$$

We could declare a term canonical if the term stabilizes (the owner makes no modifications for some period of time) and the score remains above some threshold (for the same time period). Similarly, when a stable term's score falls below some threshold (deprecation). However, and this is the central observation, we can't assume that every user will vote on every term as the dictionary increases in size. We therfore need to introduce a heuristic for community consensus based on user repuation. This makes it possible for domain-experts to promote useful terms that don't need to be verified by the entire community. At the sametime, it is critical that substantial dissent can prevent a term from entering the canon.

A user's reputation is a positive integer value that is initially seeded in the database. A user gains reputation by participating in the community (voting and commenting on terms) and contributing terms that enter the canon.[1] Let R_i be the reputation acquired by user i and let R be the total reputation of the users who have voted on a particular term. Let $r_i = R_i/R$ for each user i who voted on the term. Let t be the total number of users in the community and v be the number of votes cast such that $d = v - u$. The weight w_i of a user's vote is based on his or her reputation in the following relationship:

$$w_i = 1 + r_i \cdot (t - v)$$

The influence of the user's reputation decreases linearily as the number of voters for the term approaches the total number of users. However, everyone is guarenteed one vote. Now let $U = \{w_1, w_2, \dots w_u\}$ be the

[1] Note that the actual rules for gaining reputation within the community have yet to be defined.

set of weighted up-votes and $D = \{v_1, v_2, \ldots v_d\}$ the set of weighted down-votes. Substituting these values into the equation for S, we have

$$S = \frac{\sum_U w_i}{\sum_U w_i + \sum_D v_i}$$

Pulling out some terms from the sums and substituting $w_i = 1 + U_i/R \cdot (t - v)$ for each up voter i and $v_i = 1 + D_i/R \cdot (t - v)$ for each down voter i:

$$\begin{aligned} &= \frac{u + (t - v) \cdot \sum_U U_i/R}{u + d + (t - v) \cdot \left(\sum_U U_i/R + \sum_D D_i/R\right)} \\ &= \frac{u + (t - v) \cdot \sum_U U_i/R}{u + d + (t - v) \cdot 1} \\ S &= \frac{u + \sum U_i/R \cdot (t - v)}{t} \end{aligned}$$

This equation has the property that as v approaches t, votes become increasingly fair; i.e., S approaches $u/(u + d)$ Notice as well that we don't use a user's community-wide reputation percentage; this gives every term an equal chance of reaching a stable, canonical state despite the proposer's reputation. At the same time, it allows users with high reputation to debunk bad ones immediately with a single down-vote.

An advantage of this equation is that it's easy to compute. To collect the reputations of users voting on a particular term would normally require a join on the *Users* and *Tracking* tables. However, we need only keep track of u, v, t, R and $\sum U_i$ for each term. Then when a user casts or changes a vote, we update these values and calculate the new score in constant time. The other instance when it's necessary to update a term score is when a user's reputation changes; this can be done in linear time in terms of the number of votes cast.

We can refine this relationship a bit more if a lineear function with respect to v isn't adequate in practice. For instance, it may be useful to scale down reputation exponentially:

$$w_i = 1 + \frac{r_i \cdot t}{v}$$

Howver, we'll have a better idea of how to do this once we get users and a sense of the distribution of reputation values.

2 Classification

Once a vernacular term stabilizes, we decide whether to include it in the canon. I suggest two conditions for this property: (1) the owner hasn't modifed the term or its defnition for some predefined time interval and (2) the rate at which the term's score changes has dropped below some threshold close to zero for the same time interval.

2.1 Stability

Term stability is recalculated each time the consensus score is recalculated. To calculate the rate of change, we store the time at which the consensus score S was computed. Given previous score (S_0, t_0) and current score (S, t), the rate of change for this interval is:

$$\frac{dS}{dt} = \frac{S - S_0}{t - t_0}$$

Let t_S be the time point when the term became stable, or `nil` if the term is unstable. Let ϵ be stability error, or the amount that S can fluctuate and still be considered stable. If $|\frac{dS}{dt}| < \epsilon$ and $t_S = \texttt{nil}$, then set $t_S \leftarrow t$; if $|\frac{dS}{dt}| > \epsilon$, then set $t_S \leftarrow \texttt{nil}$; otherwise, S is stable and t_S remains unchanged.

Now, let T be the interval of time required for stability and t be the time at which we wish to classify a term. The following conditions are perscribed:

1. $t - t_M > T$, where t_M is the time at which the term was modified, and
2. $t_S \neq \texttt{nil}$ and $t - T - S > T$ **or** $t - t_0 > T$.

2.2 Promotion and demotion

If a term has stabilized, we use its consensus score S to classify it. We need only to decide on reasonable threshold values. To start, I suggest a stabile term should require 75% consensus to be promoted to the canon. If consnesus drops below 25% after it stabilizes, it should be deprecated. Otherwise, the term should remain in the vernacular.

If a term is canonical or deprecated and becomes unstable, it does not immediately reenter the vernacular. Instead, we wait until it restabilizes and classify it based on the resulting score.

We expect that terms will be deprecated by the community when there is a viable alternative. In this case, the standard practice should be to include a reference to the new term in the old's definition.

3 Software components

In Chicago, we were brainstorming about a consistency issue related to voting and the term being modified. The solution we discussed was to require a call-for-votes and a score reset when the owner modifies the term. It was remarked that this is a bit clunky. An alternative approach would be to notify users with a term they're interested in get's updated, promoted, or demoted.

3.1 Notifications

First, we will allow users to "star" and "unstar" terms that they're interested in. On the homepage of the SeaIce website, we could have a notification page, much like facebook, which provides updates on terms you own and terms your're tracking:

1. User X has commented on your term A
2. User X has updated term B
3. Term B has been promoted to the canon
4. Term C has been deprecated
5. Term C has reentered the vernacular

Once you see the notifcation, you can respond by commenting, changing your vote, unstarring the term, etc. Because term stabilization depends not only on the owner not editing the term but on the score's rate of change, there is no need to reset the score on modification.

3.2 Browsing and search

We have a number of distinct goals in term discovery: most stable (canonical, lots of consensus), most recently contributed, and most frequently discussed to name a few. In the manner of stackoverflow, we'll

provide a listing of terms per criterion. Making these available will be very useful for contribution. The default search query ranks pages first by relevance to the query, second by stability and classificaition, and third by consensus score.