Community-based scoring of metadictionary terms

Christopher Patton

July 2, 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. 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 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 is above some threshold (for the same time period). This applies in the converse case, when we deprecate a term. 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. Let R_i be the reputation acquired by user i and 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 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 linearly as the number of voters for the term approaches the total number of users. However, everyone is guarenteed one vote. Now let $U = \{w_1, w_2, \ldots w_u\}$ be the 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

$$\begin{aligned} S &= \frac{\sum_U w_i}{\sum_U w_i + \sum_D v_i} \\ &= \frac{u + \sum_U r_i^w \cdot (t - v)}{u + d + \left(\sum_U r_i^w + \sum_D r_i^v \right) \cdot (t - v)} \\ &= \frac{1}{1 + \dfrac{\sum_D r_i^v \cdot (t-v) + d}{\sum_U r_i^w \cdot (t-v) + u}} \end{aligned}$$

This equation has the property that as v approaches t, votes become increasingly fair (equal). 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.

I think this is a pretty good place to start, but we may find that a linear function (in terms of v) isn't adequate in practice. For instance, it may be useful to scale down reputation faster:

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

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 Stabilization and classificaiton

Once a term stabilizes, we decide whether to deprecate it or 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. The time interval could be two days to start.

Now, to decide whether to promote or demote a term, we need only to decide on reasonable threshold values. To start, I suggest a stabile term should require 75% consenus 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.

Another set of rules we need to define are when a canonical or deprecated term reenters the vernacular. We could base this on the rate of chang eof the score.

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; I'd like to suggest an alternative.

3.1 Notifications

First, we will allow uesrs 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 rentered 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.