Kanji Tester: Error Modelling

Lars Yencken

April 9, 2019

Contents

1 Prior distributions

This section documents the prior error distributions use for each error type.

1.1 ($\mathbf{kanji}'|\mathbf{kanji}$)

Given a frequency distribution $\Pr(K)$ and a similarity measure $s : K \times K \to [0, 1]$, we can calculate the error distribution:

$$\Pr(K'|K) = \frac{\Pr(K')s(K', K)}{\alpha}$$

where α is a normalisation constant.

1.2 ($\mathbf{reading}'|\mathbf{reading}, \mathbf{kanji}$)

Describe FOKS reading model.

2 Update rule

This section describes how we update our error models from user responses. When the error model is based on an individual item, we use a simple updating method, and when it is based on a sequence of items from our error distribution, we need to use a different sequence model.

2.1 Simple updating

First we start with some basic definitions:

$$\begin{aligned} O &= \text{the finite set of options available for a question} \\ D &= \text{the options displayed to the user, } D = d_1 \dots d_n,\ D \subset O \\ c &= \text{the option the user picks} \end{aligned}$$

We displayed a random subset D of possible options O to the user, and they chose option c as their answer. $\Pr(c|D)$ is known from our prior distribution. We wish to determine the posterior value for $\Pr'(c)$. We base our update rule on the constraint:

$$\forall_{\{i:d_i \neq c\}} \Pr(c|D) \geq \Pr(d_i|D) + \epsilon$$

That is, the user chose c because it was better than any other option by a margin of ϵ. Our update rule simply enforces this margin of ϵ in the posterior distribution $(C|D)$ used in the next iteration.

1. Let $m = \max_{\{i:d_i \neq c\}} \Pr(d_i|D) + \epsilon$
2. Define the posterior distribution $(C|D)$ as follows:
   - $\Pr'(d_i|D) = \Pr(d_i|D)$ if $d_i \neq c$
   - $\Pr'(c|D) = \max\{\Pr(c|D), m\}$
   - Normalise $(C|D)$ such that $\sum_i \Pr'(d_i|D) = 1$
3. Retain $\Pr'(d) = \Pr(d)$ for all $d \in O \setminus D$[1]
4. Set $\Pr'(d_i) = \Pr'(d_i|D) \Pr'(D)$

2.2 Updating sequence models

2.2.1 Draft model

In our sequence model, each distractor d is a sequence of elementary items $d_1 \ldots d_n$ in the distractor space $D = D_1 \times \cdots \times D_n$.[2] We use the same steps as in the simple update, up to the point where we have calculated $\forall d : \Pr'(d|D)$. At this point, we know the new sequence probability, but wish to determine the new primitive probability $\Pr'(d_i|D_i)$. We define:

$$\Delta = \frac{\Pr'(d|D)}{\Pr(d|D)}$$

which has known value at this stage. We make some simplifying assumptions to further reduce the right-hand side:

$$\begin{aligned}
\Pr(d|D) &= \Pr(d_1 \ldots d_n|D) \\
&\approx \prod_{j=1}^{n} \Pr(d_j|D) \\
&\quad \text{assuming indepencence of each } (d_j|D) \\
&\approx \prod_{j=1}^{n} \Pr(d_j|D_j) \\
&\quad \text{assuming further independence of } d_j \text{ from all } D_i,\ i \neq j
\end{aligned}$$

Whilst these assumptions clearly *do not* hold in real life, whether our sequences models are of kanji readings or of highly similar characters, since real-life cooccurrence frequencies differ significantly for different combinations of sequence items. We nonetheless make these assumptions to retain a simple and tractable model. Plugging this result back into our initial recipe for Δ:

$$\Delta \approx \prod_{j=1}^{n} \frac{\Pr'(d_j|D_j)}{\Pr(d_j|D_j)}$$

[1] The evidences gives us no reason to increase or decrease the overall likelihood of D.

[2] Note the differing meaning of this subscript from the previous section. d_i previously represented the ith option displayed to the user; it now represents the ith sequence element of the single option d.

We choose to distribute the probability mass evenly, which adds the constraint:

$$\frac{\Pr'(d_j|D_j)}{\Pr(d_j|D_j)} = \frac{\Pr'(d_k|D_k)}{\Pr(d_k|D_k)} \quad \forall j,k \in \{1,\ldots,n\}$$

Then our final update rule emerges:

$$\Pr'(d_j|D_j) = \Delta^{1/n} \Pr(d_j|D_j)$$

Note that this rule requires us to change our earlier constraint that no unseen options should have their likelihoods changed. At the primitive level, any unseen symbol will retain its original likelihood. However, at the sequence level, unseen combinations of seen primitives *will* have their likelihoods changed as the likelihoods of their constitutents are updated. This is not problematic, merely a consequence of our unigram model.

A Alternative error models

A.1 Linear interpolation

A.1.1 Description

An alternative error model which could be considered is one where fixed priors are used to model different sources of error, and linear interpolation these alternating error models are used. We use an example to illustrate, that where we want to know the probability of the distractor にっき for word 日本. Let:

$$\begin{aligned}
R &= R_1 R_2 \quad \text{the sequence reading} \\
&= \text{にっき} \\
R_1 &= \text{にっ} \quad \text{the segment reading} \\
R_1^* &= \text{にち} \quad \text{the canonical segment reading} \\
R_2 = R_2^* &= \text{き} \\
K &= K_1 K_2 \\
&= \text{日本} \\
K_2' &= \text{木} \quad \text{a similarity variant of } K_2
\end{aligned}$$

We construct our probability of seeing this sequence reading as follows:

$$\begin{aligned}
\Pr(R|K) = {} & \alpha_{\text{rc}} \prod_{i=1}^{n} \Pr_{\text{rc}}(R_i^*|K_i) + \\
& \beta_{\text{vl}} \prod_{i=1}^{n} \Pr_{\text{vl}}(R_i|R_i^*) + \\
& \gamma_{\text{on}} \prod_{i=1}^{n-1} \Pr_{\text{on}}(R_i|R_i^*) + \\
& \delta_{\text{sv}} \prod_{i=2}^{n} \Pr_{\text{sv}}(R_i|R_i^*) + \\
& \epsilon_{\text{gs}} \prod_{i=1}^{n} \Pr_{\text{gs}}(R_i^*|K_i') \Pr_{\text{gs}}(K_i'|K_i)
\end{aligned}$$

This model uses many fixed probability tables for reading confusion (rc), vowel length (vl), sound euphony (on), sequential voicing (sv) and graphical similarity (gs). The final probability is determined by using the user-specific vector

$$\langle \alpha_{\text{rc}}, \beta_{\text{vl}}, \gamma_{\text{on}}, \delta_{\text{sv}}, \epsilon_{\text{gs}} \rangle$$

where each value is the number of times a user has made a mistake generated using that model. After the user makes a number of responses, the vector is updated with the new error counts. A normalised version of the updated vector is then used to generate the next set of test questions.

A.1.2 Comparison

Comparing this model to our chosen model yields advantages, disadvantages, and differences which could be either.

Advantages	Disadvantages	Differences
• Simple per-user model • Simple update function • Easy interpretation as error type distribution	• Monolithic error model more complex • Explicit error distributions must be reconstructed	• Adapts to the type of error a learner is making, not the exact error itself

The two error models consider learner error at a different level of abstraction, but ultimately neither limits the forms of post-hoc evaluation which can be performed. Ideally, both would be implemented and compared on actual user data, to see which more accurately predicts user responses.