technical appendix

We shall continue to employ, with no more ado, the terminology and notation we have already used to elaborate upon and examine mathematical matters.

1 For the supervised procedure we looked at above the raw training data was prepared by grouping extracted lingual features into clusters distinguished from each other by discrete degrees of polarity, recall, with a feature being assigned to its cluster via the polarity of the documents which exhibit that feature: we shall set out here that process of clustering the extracted features.

Suppose there are n labelled documents $D_1, D_2, \ldots, D_n$ in the training data, out of which n^+ are positive, say, and n^- are negative. Let $\phi_1, \phi_2, \ldots, \phi_m$ be the extracted features. For each $1 \leq j \leq m$ let c_j^+ be the count of positive documents exhibiting ϕ_j and c_j^- the count of negative documents doing the same. Assume that neither count is zero: how the ratio c_j^+/c_j^- diverges from the general ratio n^+/n^- seems a plausible indicator, now, of the polarity of ϕ_j itself. The inventors of the procedure do not worry that either c_j^+ or c_j^- might be zero, we should note, and they take

$$\omega_j \equiv \log\left(\frac{c_j^+/c_j^-}{n^+/n^-}\right) = \log c_j^+ - \log c_j^- - \log n^+ + \log n^-$$

as their measure of polarity. The simplest way to deal with zeroes is to add one to the counts $c_j^\pm$ and, declaring $\epsilon_j^\pm = n^\pm/c_j^\pm$ if $c_j^\pm \neq 0$ and 0 otherwise, set

$$\omega_j \equiv \log(c_j^+ + 1) - \log(c_j^- + 1) - \log(n^+ + \epsilon_j^+) + \log(n^- + \epsilon_j^-)$$

instead: which will not perturb the measure at all when $c_j^\pm > 0$, of course, since $a/b = (a+1)/(b+b/a)$ when $a, b \neq 0$. But however zeroes are circumvented these measures will lie toward zero when $c_j^+/c_j^+ \approx n^+/n^-$, and — because $\log(u/v) = -\log(v/u)$ for $u, v > 0$, and as each ω_j is $\log(c_j^+/c_j^-)$ translated by $\log(n^-/n^+)$ uniformly — the numbers so obtained will increase or decrease symmetrically according to whether c_j^+ exceeds c_j^- or falls short of it: except when either count is zero, of course, when the shifting will be marginally more or less. But zero counts would be rare presumably. Neither n^+ nor n^- will be miniscule compared to the other, one expects — which should keep the uniform translation small compared to the spread of the measures — so with $\lambda = min_j\{\omega_j\}$ and $\rho = max_j\{\omega_j\}$ our quantities will all lie in the interval $[\lambda, \rho]$ with $\lambda < 0 < \rho$ one may expect, and, assuming some parity in number and degree between features with opposed polarities, the extent to which ω_j is positive or negative may be taken to gauge the polarity of the feature ϕ_j now. The features are clustered now by clustering the corresponding measures: which, ideally, would group themselves through varying proximities. The inventors of the procedure choose to divide and enclose $[\lambda, \rho]$ within some $q < m$ intervals of equal length, rather, taking equally spaced points $\xi_0 \leq \lambda < \xi_1 < \ldots < \xi_{q-1} < \rho < \xi_q$ to determine their clusters, and they declare two features ϕ_i and ϕ_j equivalent in polarity if the corresponding measures ω_i and ω_j both lie in the same interval $[\xi_{k-1}, \xi_k)$ for some $0 < k \leq q$. How they decide on an optimal number of clusters they do not say, and, besides, proceeding so is proper only if the $\{\omega_j\}$ are distributed along $[\lambda, \rho]$ in a more or less uniform way: which is one reason more, besides the reasons already given, to question the propriety of proceeding so.

2 The general scheme of 'local explication' we looked at above, set out in [11], attempted linear approximations to particular machine deliverances. The requirements there were that the random vector X of which the inputs x are instances should contain *interpretable* real-valued components comprising a random vector X^ι, recall, whose instances are interpretable counterparts x^ι of the inputs x, and, furthermore, that the input vectors should admit a natural distance between them. Let u be some fixed input for which deliverance is to be explicated, as before, and $\sigma_u(x)$ a real-valued index of proximity, between the fixed u and any other input x, which decreases according to distance from u to x, and becomes practically zero when x is far from u. For any input x the algorithm proceeds on the basis of a computed scalar value $f(x)$ of an objective function f, recall, and in [11] the interpretable linear approximation to f near u is a vector of real weights ω which *minimizes,* over all inputs x in a sufficiently large sample S randomly drawn from the general vicinity of u, the scaled sum of squared differences

$$L_u^{\iota;f}(\omega) \equiv \sum_{x\in S} \sigma_u(x)[f(x) - \langle \omega, x^\iota\rangle]^2$$

where $\langle \omega, x^\iota\rangle$ is the standard inner-product of ω with the interpretable vector x^ι. The notation means to convey that f and u are fixed while ω varies. Let ω_u^ι be the optimal weighting that minimizes the scaled sum $L_u^{\iota;f}$ for the interpretable redaction u^ι of the given u : one cannot take ω_u^ι to *explain* how the algorithm arrived at $f(u)$, we had recorded, because a different algorithm and objective function might have given us a similar vector of weights. Why f itself scored u just as it did would not be disclosed by ω_u^ι either, we had noted, unless we could *assess* how u^ι itself contributes to $f(u)$: which could seldom be done for an algorithm whose success depends on learning some *non-linear* objective function. In the main text we had adverted to a particular complication, moreover, with the general scheme of [11]: here that is. Replace f with $g(x) = f(x) + c$ for some real constant c : as the objective function is changed by a fixed quantity, only, one should be able to recover the original deliverance of the algorithm from g as well. We could do all this with the naive bayesian procedure we had detailed, for instance, by setting $f(x) = \mathbf{s}_+(x)\pi[+] - \mathbf{s}_-(x)\pi[-]$ and *sign* $f(x)$ as the orginal *effective* output of the algorithm. Now

$$\omega_u^{\iota;g} \equiv \mathit{argmin}_\omega \left\{\sum_{x\in S} \sigma_u(x)[g(x) - \langle \omega, x^\iota\rangle]^2\right\}$$

would be the locally linear approximation to g around u and, as the original deliverance of the algorithm can be got from g as well as f, very likely, this vector is a candidate for explicating its result at u as well: but

$$\omega_u^{\iota;g} \neq \mathit{argmin}_\omega \left\{\sum_{x\in S} \sigma_u(x)[f(x) - \langle \omega, x^\iota\rangle]^2\right\} \equiv \omega_\iota^{u,f}$$

generally, and the resulting problem with the scheme in [11], as we noted, is that it ignores how the values of the objective function are converted into effective output.