

A Kick-ass Application

Phillip Potamites

May 31, 2007

A really kick-ass application, for me, as a linguistic researcher, would allow me to compare the predictive capabilities of different learning algorithms and different sets of features for different kinds of items. For instance, I am interested in the different entailments that embedded clauses allow, and I would like to collect and arrange observations about which predicates, in which environments, select clauses with which entailments.¹ My friend, on the other hand, has been working on some speech analysis software, and he would like to improve the accuracy of his [+/- voice] decisions by including more contextual observations. A sufficiently abstract implementation would allow both of us to determine which analysis of which features provides the best predictions about the phenomena that interest us. Such results are theoretically interesting in so far as they determine abstract causal dependencies, and they are also practically useful, exactly in so far as the selected phenomena is useful to predict.

Similarity clustering, decision trees, and log-linear models all allow the researcher to arbitrarily select any number of features as the basis of observation. However, these algorithms only achieve successful outcomes if those features are sufficiently relevant to the prediction being sought. There are techniques for evaluating feature quality, and also ways of auto-generating kinds of features to observe.

One of the researcher's major responsibilities, then, is to provide sufficient observational richness to allow the algorithms to do their work. Still, a really kick-ass application needs additional algorithms which can re-evaluate the decisions of the researcher, and determine which feature dependencies are most significant, how they affect each other, and how decisions can be made most efficiently.

This kick-ass application will integrate implementations of the following learning techniques, for comparative evaluation, and, perhaps, even integrated decision making.

- Similarity Clustering:

¹For instance, different predicates give us different messages about the actuality of embedded events.

- (1) a. I doubt that she won.
b. I love that she won.

Clustering by feature vector similarity is an unsupervised learning technique, which returns a selected number of groups from numerous observations. New observations can then be categorized by the same techniques. If we can determine the right number of significant kinds of groups (for different feature sets), our groups should be a valuable predictive device. For instance, if we can collect all the predicates (in specific environments) that always have implicative entailments ('managed to win'), we can definitely predict the actuality of the embedded event. Determining those predicates might require annotated examples, or we would hopefully uncover other properties that define that same grouping.

'Mutual information' for features is calculated according to their relative significance in identifying a given event (2). Mutual information values should also be adjusted (multiplied) by a discounting factor (3). Those mutual information values are used to compare the cosine similarity of feature vectors (4).

Patrick Pantel (lecture notes, 2007):

$$(2) \quad mi_{wf} = \log_2 \frac{\frac{c_{wf}}{N}}{\frac{c_{*f}}{N} * \frac{c_{w*}}{N}}$$

$$(3) \quad discount = \frac{c_{wf}}{c_{wf}+1} * \frac{\min(c_{w*}, c_{*f})}{\min(c_{w*}, c_{*f})+1}$$

$$(4) \quad \cos(w_i, w_j) = \frac{\sum_f mi_{w_i f} * mi_{w_j f}}{\sqrt{\sum_f mi_{w_i f}^2} * \sqrt{\sum_f mi_{w_j f}^2}}$$

The clustering algorithm groups together the two most similar events, and then the next, and so on. It stops when a specified number is reached. As noted, a really kick-ass application needs a way to evaluate the significance of these groups. We need methods to search for groupings, on the basis of some features, which make decent predictions about other features of interest.

- Decision Trees:

Decision trees are built by a supervised learning technique which ranks attributes to build the shortest paths to final decisions. New observations can be categorized on the basis of the determined attribute rankings. Because these trees are built on arbitrarily selected features, they also allow us to investigate how well different sets of features predict others.

The information contained in the knowledge of some outcome is a function of the probability of the potential values of that outcome (5). When evaluating a given attribute, the remaining information is the sum of the proportional information left

in each of the values of that attribute (6). The information gain is the original information minus the remainder (7).

Russell & Norvig (2006), p. 687-8:

$$(5) I(P(v_1), \dots, P(v_n)) = \sum_{i=1}^n -P(v_i) \log_2 P(v_i)$$

$$(6) \text{Remainder}(A) = \sum_{i=1}^v \frac{p_i + n_i}{p + n} I\left(\frac{p_i}{p_i + n_i}, \frac{n_i}{p_i + n_i}\right)$$

$$(7) \text{Gain}(A) = I\left(\frac{p}{p+n}, \frac{n}{p+n}\right) - \text{Remainder}(A)$$

This learning method again gives the researcher the ability to compare broad permutations of the predictive relevance of different features. If we determine which features mark implicative predicates, we can better infer entailed events. If we determine which features mark voicing, we can build more accurate voice recognition systems.

- Log-Linear Models:

Log-linear models calculate feature weights to give the maximum likelihood of the observed feature vector-label pairs. New observations can then be labeled according to their features and these weights. For different labels, we can determine which features have more or less weight.

The conditional probability of a label given some observation is given by (8). Training requires maximizing the probability of all labeled observations (9).

Noah Smith (2004): (conditional)

$$(8) \Pr(y|x) = \frac{e^{\vec{\theta} \cdot \vec{f}(x,y)}}{\sum_{y'} e^{\vec{\theta} \cdot \vec{f}(x,y')}}$$

$$(9) \vec{\theta}^* = \max_{\vec{\theta}} \sum_{j=1}^m \log \Pr_{\vec{\theta}}(y_j^* | x_j) = \left(\sum_{j=1}^m \vec{\theta} \cdot \vec{f}(x_j, y_j^*) \right) - \sum_{j=1}^m \log \sum_{y'} e^{\vec{\theta} \cdot \vec{f}(x_j, y')}$$

Given sufficient annotations, log-linear models should be able to solve both the entailment and voicing problems exactly by determining the most significant features. They further open the way for analysis of all the possible predictive permutations across different sets of features.

In summary, my really kick-ass application needs both manual and automatic methods of adjusting all of the following, necessary capabilities:

(10) Necessary capabilities:

- a. specify features to observe
- b. observe data and learn about relationships of those features
- c. annotate data on the basis of those feature relations
- d. evaluate the significance/accuracy of those annotations

Such a highly adaptable application should be an excellent tool for theoretical researchers, and provide superior performance for a wide-variety of practical tasks. In clearly assuming these as unified goals, this kind of implementation might also help the linguistic field desist from both theoretical inquiry with no practical benefit and statistical success without principled understanding. Proper learning involves both the understanding of abstract, principled dependencies and gaining the ability to make solid, practical predictions. In proper scientific inquiry, there is no divide between what is practically useful and what is theoretically interesting.