A Max-Margin Best-Previous-Match Coreference Resolution Algorithm

Jason D. M. Rennie
jrennie@gmail.com

July 25, 2005

1 Introduction

A basic coreference resolution algorithm, best-previous-match, calls two mentions coreferent if the first is the smallest distance mention from the second (and below a distance threshold), or if there is a chain of coreferent mentions linking the two. The inference algorithm is simple: for each mention, find the previous mention with smallest distance. If the distance is below a threshold, then the two mentions are coreferent. Two mentions are also coreferent if there is a chain of coreferent mentions connecting them.

2 Classification and Objective

The core task herein is classification. For each mention, we must identify which of the previous mentions (including a special "null" mention, accounting for the "threshold" discussed above) is most likely to be coreferent. Instead of calculating a real distance between mentions, we calculate a feature vector between each pair of mentions and try to find a linear projection so that the correct mention will always have the largest projected value. To improve our chances of achieving generalization, we require not simply that the correct mention has the largest value, but that it has the largest value by a margin of 1. However, in practice, there may not exist a linear projection that meets this criterion. One option is to use a non-linear projection, which we will not discuss here (however, such projections are widely discussed in conjunction with kernels and the Support Vector Machine). Instead, as in done in "soft-margin" formulations of the SVM, we will impose a penalty for each example less than a distance 1 away from the next-best candidate. The penalty will be equal to the violation distance (the amount by which the value for the correct mention would need to be increased to satisfy the margin constraint), summed over all mentions that come previously in the text. By summing over all mentions, rather than just the next-best, we more severly penalize projections that allow multiple incorrect mentions within the margin or to be projected higher than the correct mention.

3 Math

Let $\{x_1, \ldots, x_n\}$ be a set of noun phrase mentions in a document. Define x_0 to be the special null mention. Let $\{y_1, \ldots, y_n\}$, $y_i \in \{0, \ldots, i-1\}$, identify (by index number) the prior mention to which each mention refers. We use 0 to denote the null reference. Let $h(z) = (1-z)_+$ define the "hinge" penalty function. Let $\vec{f}(x_i, x_j) \in \mathbb{R}^d$ be the feature function, which produces a d-dimensional feature vector for each pair of mentions. Then, our optimal projection vector is

$$\vec{w}^* = \min_{\vec{w}} \sum_{i=1}^{n} \sum_{j=0}^{i-1} h\left(\vec{w}^T \left[\vec{f}(x_i, x_{y_i}) - \vec{f}(x_i, x_j)\right]\right) + \frac{c}{2} \sum_k w_k^2. \tag{1}$$

Note that we have not included the constant $-nh(0)$ term since it is immaterial and will not impact the optimal weight vector. To solve for the optimal projection vector, $\vec{w}^*$, we use gradient descent. The gradient of the objective with respect to the projection vector is

$$\frac{\partial J}{\partial \vec{w}} = \sum_{i=1}^{n} \sum_{j=0}^{i-1} h'\left(\vec{w}^T \left[\vec{f}(x_i, x_{y_i}) - \vec{f}(x_i, x_j)\right]\right) \left[\vec{f}(x_i, x_{y_i}) - \vec{f}(x_i, x_j)\right] + c\vec{w}, \tag{2}$$

where $h'(\cdot)$ represents the gradient of the penalty function. Since the hinge does not have a smooth gradient, we substitute what we call the "smooth hinge,"

$$h(z) = \begin{cases} \frac{1}{2} - z & \text{for } z \leq 0 \\ \frac{1}{2}(1-z)^2 & \text{for } 0 < z \leq 1 \\ 0 & \text{for } 1 < z \end{cases}. \tag{3}$$

4 Implementation

For training, we assume that we are given a set of mentions, $\{x_1, \ldots, x_n\}$ and a set of antecedents, $\{y_1, \ldots, y_n\}$, one for each mention. The antecentent for a mention must either be (1) a mention that occurs earlier in the text ($y_i \in \{1, \ldots, i{-}1\}$), or (2) the null mention, $y_i = 0$. In order to find the optimal weight vector, we calculate feature vector differences, $f(x_i, x_{y_i}) - \vec{f}(x_i, x_j)$, for all (i, j) mention pairs. These values are used to find the optimal feature vector. Once $\vec{w}^*$ is found, we do inference by finding, for each mention, the prior mention that yields the largest score (calculated via dot-product between weight and feature vectors).