NER features exercises

September 13, 2016

1 Sequence Prediction: Bigram Features, Weights and Perceptron

Recall the factored linear models for sequence prediction, and think of a named entity task. A bigram-factored sequence model computes:

\[
\begin{align*}
\hat{f}(x_{1:n}) &= \arg\max_{y_{1:n} \in \mathcal{Y}^n} w \cdot \hat{f}(x, y_{1:n}) \\
&= \arg\max_{y_{1:n} \in \mathcal{Y}^n} \sum_{i=1}^{n} w \cdot f(x, i, y_{i-1}, y_{i})
\end{align*}
\]

where \(x_{1:n}\) is an input sentence of \(n\) tokens (\(x_i\) is the \(i\)-th token), \(y_{1:n}\) is an output sequence of \(n\) tags (\(\mathcal{Y}\) is the set of valid tags). \(f(x, i, y_{i-1}, y_{i})\) is a function returning a feature vector of the bigram \(y_{i-1}, y_i\) at position \(i\) of the sentence (assume that \(y_0\) is a special tag \texttt{START} that indicates the start of the sequence). \(w\) is a vector of parameters of the same dimensionality of the feature vectors. As usual, we will specify features using templates, and we will work with the following templates:

- **Type 1:** the current tag is \(a\) and the current word is capitalized
  \[
  f_{1,a}(x_{1:n}, i, j, y) = \begin{cases}
  1 & \text{if \texttt{is\_capitalized}(x_i) and } y_i = a \\
  0 & \text{otherwise}
  \end{cases}
  \]

- **Type 2:** the current tag is \(a\) and the current word is \textit{not} capitalized
  \[
  f_{2,a}(x_{1:n}, i, j, y) = \begin{cases}
  1 & \text{if \texttt{not\_is\_capitalized}(x_i) and } y_i = a \\
  0 & \text{otherwise}
  \end{cases}
  \]

- **Type 3:** the previous tag is \(a\), the current tag is \(b\) and the current word is capitalized
  \[
  f_{3,a,b}(x_{1:n}, i, j, y) = \begin{cases}
  1 & \text{if \texttt{is\_capitalized}(x_i) and } y_{i-1} = a \text{ and } y_i = b \\
  0 & \text{otherwise}
  \end{cases}
  \]

- **Type 4:** the current tag is \(a\) and the current word is \(w\)
  \[
  f_{4,a,w}(x_{1:n}, i, j, y) = \begin{cases}
  1 & \text{if } x_i = w \text{ and } y_i = a \\
  0 & \text{otherwise}
  \end{cases}
  \]

- **Type 5:** the current tag is \(a\) and the next word is \(w\)
  \[
  f_{5,a,w}(x_{1:n}, i, j, y) = \begin{cases}
  1 & \text{if } x_{i+1} = w \text{ and } y_i = a \\
  0 & \text{otherwise}
  \end{cases}
  \]
• Type 6: the current tag is \( a \) and the previous word is \( w \)

\[
    f_{b,a,w}(x_{1:n}, i, j, y) = \begin{cases} 
        1 & \text{if } x_{i+1} = w \text{ and } y_i = a \\
        0 & \text{otherwise}
    \end{cases}
\]

• Type 7: the previous tag is \( a \) and the current tag is \( b \)

\[
    f_{r,a,b}(x_{1:n}, i, j, y) = \begin{cases} 
        1 & \text{if } y_{i-1} = a \text{ and } y_i = b \\
        0 & \text{otherwise}
    \end{cases}
\]

**Question 1: Perceptron Updates**

Consider the following training example:

\[
    y \text{ PER PER - - LOC LOC} \\
    x \text{ Jack London went to South Paris}
\]

Say we are running Perceptron, and under our current \( w \) the prediction given by Equation 1 is the following sequence \( z \):

\[
    z \text{ PER LOC - - LOC} \\
    x \text{ Jack London went to South Paris}
\]

Write the perceptron update. That is, write a weight vector \( g \) corresponding to \( g = f(x, y) - f(x, z) \). Write the non-zero values only.

**Question 2: Setting Weights**

Using the feature definitions above, write a weight vector that correctly classifies all the examples below, i.e. the weight vector should predict the correct tag sequences for all the examples. Try to set as few non-zero weights as possible. Justify your answer.

\[
    \text{PER - - Maria is beautiful} \\
    \text{LOC - - Barcelona is beautiful} \\
    \text{PER - - LOC Jack went to London} \\
    \text{LOC - - Paris is nice} \\
    \text{PER PER - - LOC LOC Jack London went to South Paris} \\
    \text{ORG - - ORG Barcelona played against Paris}
\]