Natural Language Processing - Problem Set #3

Emma Ziegellaub Eichler - edz2103@columbia.edu

November 9, 2012

Problem 1

a

$french = alignment = []$
$\forall 1 \leq n \leq m$
    $max = 0$
    $\forall 0 \leq k \leq l$
        $q = q(k|n, l, m)$
        $\forall f \in F$
            if $t(f|e_k) * q > max$
                $max = t(f|e_k) * q$
                $french[n-1] = f$
                $alignment[n-1] = k$
Return $french$, $alignment$

The key observation to writing an efficient algorithm for this problem is that each French word f_i and alignment variable a_i are independent of all other French words f_j and alignment variables a_j, $i \neq j$, so we need only the maximum likelihood French word and alignment for each position n, $1 \leq n \leq m$, in the French sentence. There are $|F|$ possible French words and $l+1$ possible alignments (since the English sentence is of length l, and there is also the *null* alignment). Thus, this algorithm is $O(m(l+1)|F|)$.

Out of interest:

We in fact may improve on this slightly if we assume the number of non-zero probability translations for each English word is considerably less than $|F|$. Then by hashing the possible translations for each word in the English sentence, we can avoid iterating over all of F more than once by initializing a translations hash for the English sentence:

$translations = defaultdict(set())$ // This is Python for a hash that provides a default value of an empty set () for each uninitialized key

$\forall f \in F$
    $\forall 1 \leq k \leq l$
        $ift(f|e_k) \neq 0$
            $translations[e_k].add(f)$

Then we modify the algorithm slightly:
$french = alignment = []$
$\forall 1 \leq n \leq m$
  $max = 0$
  $\forall 0 \leq k \leq l$
    $q = q(k|n, l, m)$
    $\forall f \in translations[e_k]$
      if $t(f|e_k) * q > max$
        $max = t(f|e_k) * q$
        $french[n-1] = f$
        $alignment[n-1] = k$
Return $french$, $alignment$
We now have an algorithm that runs in $O(l|F|+m(l+1)max_{1\leq k\leq l}|translations[e_k]|) < O(m(l+1)|F|)$ since we assume $max_{1\leq k\leq l}|translations[e_k]| << |F|$. $\square$

b

$french = []$
$\forall 1 \leq n \leq m$
  $max = 0$
  $\forall f \in F$
    $temp = 0$
    $\forall 0 \leq k \leq l$
      $temp += t(f|e_k) * q(k|n, l, m)$
    if $temp > max$
      $max = temp$
      $french[n-1] = f$
Return $french$
The key observation to writing an efficient algorithm for this problem is that each French word f_i is independent of all other French words f_j, $i \neq j$, so we need only the maximum likelihood French word for each position n, $1 \leq n \leq m$, in the French sentence. There are $|F|$ possible French words and $l+1$ possible alignments to sum over (since the English sentence is of length l, and there is also the *null* alignment). Thus, this algorithm is $O(m(l+1)|F|)$.
Out of interest:
We in fact may improve on this if we assume the number of non-zero probability translations for each English word is considerably less than $|F|$. Then by hashing the possible translations for each word in the English sentence, we can avoid iterating over all of F more than once by initializing a translations hash for the English sentence:
$translations = defaultdict(set())$ // This is Python for a hash that provides a default value of an empty set () for each uninitialized key
$\forall f \in F$
  $\forall 1 \leq k \leq l$

$\quad\quad ift(f|e_k) \neq 0$
$\quad\quad\quad translations[f].add(k)$
Then we modify the algorithm slightly:
$french = []$
$\forall 1 \leq n \leq m$
$\quad max = 0$
$\quad \forall f \in translations$
$\quad\quad temp = 0$
$\quad\quad \forall k \in translations[f] \bigcap \{0\}$
$\quad\quad\quad temp+ = t(f|e_k) * q(k|n, l, m)$
$\quad\quad$ if $temp > max$
$\quad\quad\quad max = temp$
$\quad\quad\quad french[n-1] = f$
Return $french$
We now have an algorithm that runs in $O(l|F|+m|translations|max_{f \in translations}|translations[f] \bigcap \{0\}|)$ $O(m(l+1)|F|)$ since we assume $|translations| << |F|$ (and $|max_{f \in translations}|translations[f] \bigcap \{0\}| \leq l + 1 \ \forall f$). $\square$

c

By searching for $argmax_e p(f|e)PLM(e)$ instead of $argmax_e p(e|f)$, we may reuse our t and q parameters $t(f|e)$ and $q(a_j|j, l, m)$. Were we to search for $argmax_e p(e|f)$, we would have to calculate a new set of t and q parameters, $t(e|f)$ and $q(a_j|j, m, l)$. Thus, using only one set of translation parameters, we may translate in both directions, assuming we have a language model for one of the languages (which is a reasonable assumption for the native language).

2

Initialize $temp(u, 1) = q(u|0, 1, l, m) * t(f_1|e_u)$ $\forall 0 \leq u \leq l$.
$\forall 1 \leq k \leq m$
  $\forall 0 \leq v \leq l$
    $temp(v, k) = max_{0 \leq u \leq l} q(v|u, k, l, m) * t(f_k|e_v) * temp(u, k-1)$
    $bp(v, k) = argmax_{0 \leq u \leq l} q(v|u, k, l, m) * t(f_k|e_v) * temp(u, k-1)$
$a_m = argmax_{0 \leq v \leq l} temp(v, m)$
$\forall 2 \leq k \leq m$
  $a_{k-1} = bp(a_k, k)$
Return $a_1, ..., a_m$
Given every possible alignment at every position k, $1 \leq k \leq m$ in the French sentence, we find the maximum likelihood preceding alignment (that is, the maximum likelihood alignment for position $k-1$). We then find the maximum likelihood alignment for the last position, m, and use backpointers to recover the maximum likelihood sequence of alignments. Since we use a first-order Markov assumption (each alignment is dependent only on the alignment directly preceding it), the algorithm runs in $O(m(l+1)^2) = O(ml^2)$ time. (The initialization and backpointer recovery run respectively in $O(l+1) = O(l) < O(ml^2)$ and $O(l+m) < O(ml^2)$ time.) $\square$

3

Given input $x = x_1...x_l$, initialize $temp(0,0) = 0$

$\forall 1 \leq e_0 \leq l$

  $\forall 1 \leq s_0 \leq e_0$

    if $(s_0, e_0, x_{s_0}...x_{e_0}) \in P$

      $temp(s_0, e_0) = g(s_0, e_0, x_{s_0}...x_{e_0}) + max_{1 \leq s \leq s_0 - 1} temp(s, s_0 - 1)$

      $bp(s_0, e_0) = argmax_{1 \leq s \leq s_0 - 1} temp(s, s_0 - 1)$

$p_0 = (argmax_{1 \leq s_0 \leq l} temp(s_0, l), l, x_{s_0}...x_l)$

$s = s_0$, $i = 0$, $e = l$

while $e > 0$ $p_{i+1} = (bp(s, e), s - 1, x_{bp(s,e)}...x_{s-1})$

  $i++$, $s = bp(s, e)$, $e = s - 1$

Return $p_{i-1}, ..., p_0$

Given every possible phrase p ending at every position e_0, $1 \leq e_0 \leq l$ in the sentence, we find the maximum likelihood preceding phrase (that is, the maximum likelihood phrase ending at position $s(p) - 1$). We then find the maximum likelihood phrase ending in the last position, l, and use backpointers to recover the maximum likelihood sequence of phrases. The algorithm runs in $O(l^2)$ time. (The backpointer recovery is linear in the number of phrases in the maximum likelihood derivation, which is a maximum of l.) $\square$