

# 050/520/600.666 Information Extraction from Speech and Text

## Homework # 5

Due March 17, 2006.

### The Multiple-stack Algorithm as A\* Search

Show that the multiple-stack algorithm of Section 6.6.1 is an instance of the generic A\* algorithm.

Specifically, the A\* search in Section 6.3 is concerned with finding the best hypothesis  $\hat{\mathbf{w}}_1^n$  among all hypotheses  $\mathbf{w}_1^n$  of a given length  $n$ . The parameter  $k$  is associated with the length of a partial hypothesis  $\mathbf{w}_1^k$ , and a goodness function  $F(\mathbf{w}_1^k) = g(\mathbf{w}_1^k) + d(\mathbf{w}_1^k)$  is defined for each partial hypothesis. Starting with the empty hypothesis in a *stack*, the current best partial hypothesis  $\hat{\mathbf{w}}_1^k$  is extended at each iteration to  $\hat{\mathbf{w}}_1^{k+1} = \hat{\mathbf{w}}_1^k || \hat{w}_{k+1}$ , until the best partial hypothesis attains a length  $n$ .

However, given the acoustics  $\mathbf{a}_1^m = a_1, \dots, a_m$ , the goal in Section 6.6.1 is to find

$$\hat{\mathbf{w}}_1^n = \arg \max_{\mathbf{w}} P(\mathbf{a}_1^m, \mathbf{w}) = \arg \max_{\mathbf{w}} \log P(\mathbf{a}_1^m, \mathbf{w}),$$

among all word sequences  $\mathbf{w}$  regardless of their length, and  $n$  is not provided a priori. Let us contemplate how to extend the notation of Section 6.3 to this problem.

Define a partial hypothesis to be a *pair*  $\langle l, \mathbf{w}_1^k \rangle$ , where  $0 \leq l \leq m$ , and let

$$g(\langle l, \mathbf{w}_1^k \rangle) = \log P(\mathbf{a}_1^l, \mathbf{w}_1^k) = \max_{l_1, \dots, l_{k-1}} \sum_{i=1}^k \log P(a_{l_{i-1}+1}^{l_i}, w_i), \quad 0 = l_0 \leq l_1 \leq \dots \leq l_k = l.$$

Define the length of a partial hypothesis  $\langle l, \mathbf{w}_1^k \rangle$  to be  $l$ , not  $k$ , so that the length of a complete hypothesis is  $m$ . Extending an  $l$  length hypothesis to a  $l+1$  length hypothesis therefore does not necessarily entail extending  $\mathbf{w}_1^k$  to  $\mathbf{w}_1^{k+1}$ : we could have  $\langle l, \mathbf{w}_1^k \rangle \rightarrow \langle l+1, \mathbf{w}_1^k \rangle$  just as easily as  $\langle l, \mathbf{w}_1^k \rangle \rightarrow \langle l+1, \mathbf{w}_1^k || w_{k+1} \rangle$ . To see that the multiple-stack algorithm is an A\* algorithm, let

$$d(\langle l, \mathbf{w}_1^k \rangle) = \max_{\mathbf{z}} \max_{l_1, \dots, l_{|\mathbf{z}|-1}} \sum_{i=1}^{|\mathbf{z}|} \log P(a_{l_{i-1}+1}^{l_i}, z_i), \quad l = l_0 \leq l_1 \leq \dots \leq l_{|\mathbf{z}|} = m,$$

where  $|\mathbf{z}|$  is the length of the word sequence  $\mathbf{z}$ . Note that  $d(\langle l, \mathbf{w}_1^k \rangle)$  does not depend on  $\mathbf{w}_1^k$ !

1. Show that if an A\* search is conducted using a *single* stack, with  $g(\langle l, \mathbf{w}_1^k \rangle) + d(\langle l, \mathbf{w}_1^k \rangle)$  as the goodness of a partial hypothesis  $\langle l, \mathbf{w}_1^k \rangle$ , and is stopped the first time an  $m$  length hypothesis  $\langle m, \hat{\mathbf{w}}_1^n \rangle$  percolates to the top, then  $\hat{\mathbf{w}}_1^n$  is indeed the most likely word sequence.

2. Derive a relationship between  $d(\langle l, \mathbf{w}_1^k \rangle)$  and  $g^*(l)$  as defined in Equation (11) on p100.
3. Show that sorting the stack entries during this A\* search according to

$$F^*(\langle l, \mathbf{w}_1^k \rangle) = g(\langle l, \mathbf{w}_1^k \rangle) - g^*(l),$$

will also lead to the discovery of the same  $\hat{\mathbf{w}}_1^n$ , i.e.  $\langle m, \hat{\mathbf{w}}_1^n \rangle$  will be the first  $m$  length hypothesis to percolate to the top of the (single) stack.

Argue based on these results that when the multiple-stack algorithm on p101 stops, the top entry in the  $m$ -th stack is the most likely word sequence  $\hat{\mathbf{w}}_1^n$ .

## N-best Paths Using the Multiple-stack Algorithm

Modify the multiple-stack algorithm of Section 6.6 to obtain the  $N$  best hypotheses instead of only the best. In particular, assume that the conditions (a), (b) and (c) of Section 6.6.1 on p100 are satisfied.

1. Minimally modify the algorithm of Section 6.6.1 to obtain the *two* best paths.
2. Check if the extension of Section 6.6.2 will hold for your new algorithm.
3. Generalize your modification for any  $N \geq 2$  best paths.

Discuss whether the extension to the actual multiple-stack algorithm of Section 6.6.3, when the assumptions (a), (b) and (c) do not hold, will be possible for your new algorithm. What are the pitfalls for  $N \geq 2$  that were not there for  $N = 1$ ?

## Coarse-to-Fine Viterbi Search

Replace the Viterbi search (pp 22-23) for finding the most likely state sequence  $s_1, \dots, s_k$ , given the observations  $y_1, \dots, y_k$ , and initial state  $s_0$ , by a succession of Viterbi searches on *increasingly complex* HMMs as follows.

- (a) Let  $\mathcal{S} = \{1, \dots, c\}$  denote the set of HMM states, and  $\mathcal{Y}$  the output alphabet. Assume that output probabilities  $q(y|s' \rightarrow s)$  are associated with the arcs  $s' \rightarrow s$  of the HMM.

Hierarchically cluster the HMM states into an approximately balanced binary tree, the  $\approx \frac{c}{2}$  first-level clusters comprising of 2 states each, which are then clustered into  $\approx \frac{c}{4}$  second-level clusters of 4 states each, and so on. This clustering can be arbitrary.

Let  $\mathcal{S}_0$  and  $\mathcal{S}_1$  denote the top split of the tree, containing complementary subset of  $\mathcal{S}$ . Let  $\mathcal{S}_{00}$  and  $\mathcal{S}_{01}$  denote the split of  $\mathcal{S}_0$ ,  $\mathcal{S}_{10}$  and  $\mathcal{S}_{11}$  denote the split of  $\mathcal{S}_1$ , and so on.

- (b) Construct a 2-state HMM with *super*-states  $\mathcal{S}^{(2)} = \{0, 1\}$ . Designate the initial state of this HMM to be 0 if, in the original HMM,  $s_0 \in \mathcal{S}_0$ , and designate it to be 1 otherwise.

Construct arcs  $i \rightarrow j$  with output “probabilities”

$$q^{(2)}(y|i \rightarrow j) = \max_{s' \in \mathcal{S}_i, s \in \mathcal{S}_j} q(y|s' \rightarrow s) \quad i, j \in \{0, 1\}, y \in \mathcal{Y},$$

omitting arcs  $i \rightarrow j$  whenever there are *no arcs* from any state  $s' \in \mathcal{S}_i$  to a state  $s \in \mathcal{S}_j$  in the original HMM.

Construct the  $k$ -stage trellis for this HMM, with exactly 2 states per trellis stage (time step).

- (c) Use the Viterbi algorithm to find the most likely path  $s_1^{(2)}, s_2^{(2)}, \dots, s_k^{(2)}$  through this trellis.
- (d) Along (only) this winning path in the trellis, “shatter” or *refine* each winning *super*-state into two smaller *super*-states based on the hierarchical clustering of  $\mathcal{S}$ .

Note that in some trellis stages, state  $0 \in \mathcal{S}^{(2)}$  will be shattered to obtain states 00 and 01 based on the division of  $\mathcal{S}_0$  into  $\mathcal{S}_{00}$  and  $\mathcal{S}_{01}$ , while for other stages, state  $1 \in \mathcal{S}^{(2)}$  will be shattered to obtain states 10 and 11 based on the division of  $\mathcal{S}_1$  into  $\mathcal{S}_{10}$  and  $\mathcal{S}_{11}$ . Each trellis stage will have exactly 3 states drawn from the state-space  $\mathcal{S}^{(3)} = \{00, 01, 1\} \cup \{0, 10, 11\}$ .

- (e) Construct arcs  $i \rightarrow j$  for states in the refined trellis, with output “probabilities”

$$q^{(3)}(y|i \rightarrow j) = \max_{s' \in \mathcal{S}_i, s \in \mathcal{S}_j} q(y|s' \rightarrow s) \quad i, j \in \{0, 1, 00, 01, 10, 11\}, y \in \mathcal{Y},$$

unless there are *no arcs* from any state  $s' \in \mathcal{S}_i$  to any state  $s \in \mathcal{S}_j$  in the original HMM.

Use the Viterbi algorithm to find the most likely path  $s_1^{(3)}, s_2^{(3)}, \dots, s_k^{(3)}$  in the refined trellis.

- (f) Iterate the trellis refinement of Step (d) and Viterbi decoding of Step (e) until the winning path *cannot be refined further*, i.e. it consists only of individual states in the original HMM.

Answer the following questions for this iterated Viterbi search procedure.

1. After Step (c), if, say,  $s_t^{(2)} = 0$  for some  $t$ , is there any guarantee that the Viterbi path  $s_1, \dots, s_k$  in the original HMM passes through some state  $s_t \in \mathcal{S}_0$ ? Why or why not?
2. Argue why the procedure described above finds the correct Viterbi path in the original HMM.
3. What is the best-case and worst-case computational complexity of this iterated Viterbi procedure? Compare it with the regular Viterbi search, which has complexity  $O(c^2k)$ .
4. Discuss how the hierarchical clustering of the states ought to be done, so that performance closer to the best-case than the worst-case is likely to be obtained.