

# Project 3

520.476

(Extraction of Information from Speech and Text)

Issued April 4, 2003      Due: May 2, 2003

In this project you will derive language models for letter prediction by two different decision tree growing methods. You will compute, on test data, the respective perplexities and compare them. Make sure you keep test and training data separate.

## 1 Hierarchical clustering of letters

1. Let the letter alphabet  $\mathcal{L}$  consist of lower case letters plus space. Extract bigram statistics  $f(l_1, l_2)$ ,  $l_1, l_2 \in \mathcal{L}$ , from the training set of Project 1 and use them to derive a complete clustering tree for the letter set  $\mathcal{L}$ . Your algorithm should aim to maximize mutual information between letter clusters (classes).

That is, if at any stage clusters  $\mathcal{C}_1, \mathcal{C}_2, \dots, \mathcal{C}_n$  exist, you will be looking for indices  $i$  and  $j$  which will maximize

$$J = \sum_{\substack{k \notin \{i,j\} \\ m \notin \{i,j\}}} f(\mathcal{C}_k, \mathcal{C}_m) \log \frac{f(\mathcal{C}_k, \mathcal{C}_m)}{f(\mathcal{C}_k)f(\mathcal{C}_m)} + \sum_{k \notin \{i,j\}} f(\mathcal{C}_k, \mathcal{C}_i \cup \mathcal{C}_j) \log \frac{f(\mathcal{C}_k, \mathcal{C}_i \cup \mathcal{C}_j)}{f(\mathcal{C}_k)f(\mathcal{C}_i \cup \mathcal{C}_j)} \\ + \sum_{m \notin \{i,j\}} f(\mathcal{C}_i \cup \mathcal{C}_j, \mathcal{C}_m) \log \frac{f(\mathcal{C}_i \cup \mathcal{C}_j, \mathcal{C}_m)}{f(\mathcal{C}_i \cup \mathcal{C}_j)f(\mathcal{C}_m)} + f(\mathcal{C}_i \cup \mathcal{C}_j, \mathcal{C}_i \cup \mathcal{C}_j) \log \frac{f(\mathcal{C}_i \cup \mathcal{C}_j, \mathcal{C}_i \cup \mathcal{C}_j)}{f(\mathcal{C}_i \cup \mathcal{C}_j)f(\mathcal{C}_i \cup \mathcal{C}_j)}$$

In above,  $f(l_1, l_2)$  denotes the frequency of the letters  $l_1$  and  $l_2$  following each other in the training text, and

$$f(\mathcal{C}_k, \mathcal{C}_m) = \sum_{l_1 \in \mathcal{C}_k} \sum_{l_2 \in \mathcal{C}_m} f(l_1, l_2)$$

etc.

2. Draw the clustering tree you have obtained.
3. List the letters belonging to the best 2 clusters  $\mathcal{C}_1$  and  $\mathcal{C}_2$  (of course,  $\mathcal{C}_1 \cup \mathcal{C}_2 = \mathcal{L}$  and  $\mathcal{C}_1 \cap \mathcal{C}_2 = \phi$ ). How do  $\mathcal{C}_1$  and  $\mathcal{C}_2$  compare to the output sets corresponding to the states of the two state HMMs obtained in Project 1?
4. List the letters belonging to the best 4 clusters  $\mathcal{C}_1, \dots, \mathcal{C}_4$  (of course,  $\mathcal{C}_1 \cup \dots \cup \mathcal{C}_4 = \mathcal{L}$ ). How do they compare to the output sets corresponding to the states of the four state HMMs obtained in Project 1?

## 2 Clustering as the basis of a class language model

1. Use the hierarchical clustering of letters you derived for Problem 1 above to assign to each letter (space included - all letters are lower case - there is no punctuation) its unique bit string identification of length  $K$ , where  $K$  is the longest distance in branches from the root of your clustering tree (derived in Problem 1) to any of its leaves (i.e., letters). For letters that are less distant from the root than  $K$ , pad the bit identification string (letter codeword) with 0s. Make sure your padding follows the least significant bits, that is, the bits that correspond to the leaf!<sup>1</sup> What was the value of  $K$  you found?

---

<sup>1</sup>To be absolutely clear. Suppose a leaf is  $L$  branches distant from the root. Then its codewords will be

$$\underbrace{00\dots 0}_{K-L \text{ zeros}} b_L b_{L-1} \dots b_1$$

where  $b_1$  specifies the direction of the branch leaving the root node, on the path to the leaf.

2. Your language model should predict letters (i.e., not code bits of letters!). It will therefore consist of one tree only. Divide your training data into a *development (kept)* (80%) and a *check (heldout)*(20%) set.
  - (a) As history, use the sequence of 3 blocks of  $K$  bits identifying the three previous letters. Let  $b_{i,j}$  denote the  $j^{th}$  bit ( $1 \leq j \leq K$ ) of the  $i^{th}$  preceding letter ( $1 \leq i \leq 3$ ) <sup>2</sup>.
  - (b) The questions defining the decision tree should concern the identity of particular bits making up the history.
  - (c) A question about  $b_{i,j}$  will be legal only if either  $j = 1$  or if the identity of  $b_{i,j-1}$  was ascertained by a question associated with some node on the path from the root node to the current node (the node where the query about  $b_{i,j}$  is contemplated).
  - (d) Use the value of the entropy of letter prediction as the criterion for selecting a question at any given node. The probabilities inducing the entropy are the relative frequencies of letters in the development or check set (as the case may be) associated with the leaves of the tree.
  - (e) A node is split if the *best* question (found by evaluation over development data) at the leaf results in a reduction of entropy on the *check* data of more than 0.005 bits. Note that it is the entropy induced over the complete decision tree that must be so reduced, **not** just the partial entropy corresponding only to the leaf being split.
  - (f) Decision tree construction ends when there are no more leaves to be split, i.e., when attempts at splitting all leaves resulted in insufficient entropy reductions of check data.
3. Print out the decision tree you constructed, indicating which bit was the cause of each node split. Were any questions asked about  $b_{3,1}$  before the letter corresponding to  $b_{1,j}$  was identified?
4. Compute the per letter *logprob* of the test data corresponding to the tree language model you developed. Can you use the decision tree to

---

<sup>2</sup>For the  $i^{th}$  letter the most significant bit is denoted by  $b_{i,1}$

compute it? Describe how to. If smoothing should prove absolutely necessary, carry out some simple variety of it.

### 3 A class letter language model based on the Chou algorithm

Use the Chou algorithm to derive a decision tree language model predicting letters. Restrict the history to 3 letters, i.e., it will be a kind of four-gram language model. Use the same stopping rule based on the check set as in the preceding problem. Divide your training data into a *development* (80%) and a *check* (20%) set.

1. The questions defining the decision tree should concern the identity of letters in a particular position in the three letter history.
2. Use the value of the Gini index of letter prediction as the criterion for selecting a question at any given node. Split that node which gives the largest Gini index gain. The probabilities inducing the entropy are the relative frequencies of letters in the development or check set (as the case may be) associated with the leaves of the tree.
3. A node is split if the *best* question (found by evaluation over *development* data) at the leaf results in a reduction of entropy (Gini index) on the *check* data of more than 0.005 bits. Note that it is the entropy induced over the complete decision tree that must be so reduced, **not** just the conditional entropy corresponding to the leaf being split.
4. Decision tree construction ends when there are no more leaves to be split, i.e., when attempts at splitting all leaves results in insufficient entropy reduction on check data.

You should pay attention to the unseen data problem discussed in Section 10.13.3. You may (but need not) approach it as suggested there. You can, for instance, distribute the unseen atoms at random among sets  $\mathcal{A}$  and  $\overline{\mathcal{A}}$ .

Compute the per letter *logprob* of the test data corresponding to the tree language model you developed. If smoothing should prove necessary, carry out some simple variety of it. Compare you results to those obtained in Problem 2.