$(520|600).666$
Information Extraction from Speech and Text

Project # 3

Due 9:00 AM on May 11, 2017.

The goal of this project is to build a decision-tree language model for English letters.

1. You will first create an encoding of the letters into bit-strings using the mutual information criterion described in Section 10.12, based on the provided *Text A*.

2. You will then build a decision-tree language model by asking questions from a restricted set based on this encoding;

   you will divide *Text A* into a development and held out portion for cross-validation for tree growing, and test the tree performance on *Text B*.

3. Finally, you will build a decision-tree with unrestricted questions using Chou's algorithm.

You will use two different goodness criteria for tree growing: *entropy* of the predicted letter for the first tree, and the *Gini index* for the second.

1   Agglomerative Clustering of the Vocabulary

Let the letter alphabet $\mathcal{L}$ consist of the twenty-six lower case letters and space. Extract bigram statistics $f(l_1, l_2)$, $l_1, l_2 \in \mathcal{L}$, from the training set (Text A), and use them to derive a bottom-up *clustering-tree* of the symbols of $\mathcal{L}$ as described in Chapter 10.

1. Specifically, if $\mathcal{C}_1, \ldots, \mathcal{C}_n$, are the clusters at any stage of the clustering process, with $\bigcup_{i=1}^{n} \mathcal{C}_i = \mathcal{L}$ and $\mathcal{C}_i \cap \mathcal{C}_j = \phi$ for all $i \neq j$, find the indices i^* and j^* that maximize

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

$$+ \sum_{m \notin \{i,j\}} f([\mathcal{C}_i \cup \mathcal{C}_j], \mathcal{C}_m) \log \frac{f([\mathcal{C}_i \cup \mathcal{C}_j], \mathcal{C}_m)}{f(\mathcal{C}_i \cup \mathcal{C}_j)f(\mathcal{C}_m)} + f([\mathcal{C}_i \cup \mathcal{C}_j], [\mathcal{C}_i \cup \mathcal{C}_j]) \log \frac{f([\mathcal{C}_i \cup \mathcal{C}_j], [\mathcal{C}_i \cup \mathcal{C}_j])}{f(\mathcal{C}_i \cup \mathcal{C}_j)f(\mathcal{C}_i \cup \mathcal{C}_j)},$$

over all $\frac{n(n-1)}{2}$ pairs $i, j \in \{1, \ldots, n\}$, and merge the clusters $\mathcal{C}_{i^*}$ and $\mathcal{C}_{j^*}$, where

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

Start with $n = |\mathcal{L}|$ clusters, each containing one letter, and iterate this process until a single cluster is obtained, i.e. $n = 1$.

2. Draw a clustering-tree to represent the clustering you obtained in Step 1 above.

3. List the elements of the best 2-way clustering of the letters, i.e. the letters belonging to the two children of the root of the clustering-tree.

   How do these two sets compare to the outputs to which the states of the 2-state HMM of Project 1 assigned large probabilities?

   How do these sets compare to the partition obtained in Problem 2 in Homework #9?

4. List the elements of the best 4-way clustering of the letters, i.e. the letters belonging to the last four clusters ($n = 4$) that were eventually merged.

   How do these four sets compare to the outputs to which the states of the 4-state HMM of Project 1 assigned large probabilities?

This clustering of the letters will be used to create a bit-encoding of each letter.

2 A Bit-Encoding Based Decision-Tree Language Model

We will use the clustering above to create letter equivalence-classes, which will form the basis of permissible questions in a decision-tree language model.

1. If K is the maximum depth of the clustering-tree, assign to each letter $l \in \mathcal{L}$, a K-bit codeword corresponding to its path in the clustering-tree, with 0-padding to the right. In other words, arbitrarily assign a 0 or 1 to each pair of siblings in the clustering-tree, and enumerate the nodes along the path from the root to the leaf containing l to obtain the encoding $c(l)$ of l, padding the encoding with zeros if the path is shorter than K. If you think of $c(\cdot)$ as a binary integer, the 0-padding is to be done in the least significant bits of the integer.

2. Divide the training text (A) into 80% for decision-tree development and hold out 20% for cross-validation. From both the development and cross-validation sets, extract letter 4-grams $\langle l_1, l_2, l_3, l_4 \rangle$, and encode the "history" $\langle l_1, l_2, l_3 \rangle$ into a bit vector of length $3K$ by concatenating $c(l_3)$, $c(l_2)$ and $c(l_1)$. Let $b_{i,j}$ denote the j-th bit of the i-th letter in the history, $1 \leq j \leq K$ and $1 \leq i \leq 3$. Treat the bit-vector

$$c(l_3) \cdot c(l_2) \cdot c(l_1) = [b_{3,1} \ldots b_{3,K} \; b_{2,1} \ldots b_{2,K} \; b_{1,1} \ldots b_{1,K}]$$

as the observed variable and the *letter l_4* as predicted variable in a decision-tree.

3. Begin by placing all the *development* data at the root of the decision-tree. Determine the question about $\langle l_1, l_2, l_3 \rangle$ that divides this data into two parts, so as to yield the maximum possible reduction in the entropy of the predicted letter l_4. i.e. predict the entire letter l_4, not its individual bits $c(l_4)$.

   (a) Restrict the questions to simply be about the value of a particular bit: i.e. there are *only* $3K$ permitted questions, in total, anywhere in the tree.

   (b) At each node in the decision-tree, place a *further restriction* on the permissible questions as follows: a question about a bit $b_{i,j}$ is permitted at a node in the decision-tree only if

      (i) $j = 1$ or

      (ii) a question has already been asked about $b_{i,j-1}$ somewhere along the path from the root to the node in question.

   Said differently, only 3 out of the $3K$ questions are actually permissible at a given node, though which 3 questions they are differs from node to node.

   (Sanity check: Which three questions are permitted at the root?)

4. For each *frontier* node of the decision-tree, compute the entropy of the predicted letter for each *permissible* question, based on the development data that reaches that node, and choose the question that results in the greatest reduction in entropy. But don't consider this split permanent yet.

5. From among all the frontier nodes, choose the node that has the greatest reduction in entropy, if split using its best question, and make it a *candidate* for a permanent split.

6. Compute the entropy reduction on the *held out* data by splitting this candidate.

   If the reduction is greater than, say, 0.005 bits, make the split permanent and let the frontier grow — by replacing that node with its two children.

   If the reduction is less than that, declare the candidate node as "terminal." (NB: The threshold is on the entropy of the predicted letter over the entire tree — *without* versus *with* the selected node being split as proposed.)

7. Recursively grow the tree at the frontier nodes that are not yet declared "terminal," until all the frontier nodes become terminal. i.e. they result in almost no entropy reduction on the held out data.

8. Draw the decision-tree you generated, marking each node with the bit $b_{i,j}$ used to split the data at that node.

   Was a question asked about $b_{1,1}$ before sufficient questions had been asked about the bits $b_{3,j}$ so as to uniquely identify l_3? In other words, are even small details about the immediately preceding letter l_3 more informative than the coarsest information about the letter l_1 two positions away?

   Discuss your observations.

9. Compute the perplexity of the test data (B) using the decision tree developed above. Should it become necessary, use some smoothing technique (cf Section 10.14.1).

3   A Decision Tree Based on Chou's Algorithm

For the same 80%-20% division of Text A into development and cross-validation, we will develop a decision tree for predicting l_4 from $\langle l_1, l_2, l_3 \rangle$ using Chou's algorithm. Thus the decision tree will again be akin to a 4-gram language model, except that the histories will be grouped into equivalence classes using Chou's algorithm. We will first use *Gini index* as the goodness criterion.

Begin with all the development data at the root of the tree, carry out the following steps recursively on each internal node of the tree.

1. Compute the Gini index of the predicted letter l_4 based on all the 4-grams $\langle l_1, l_2, l_3, l_4 \rangle$ at the node.

2. Based on the *identity* of the letter l_3 that immediately precedes l_4, make 27 equivalence classes β_i of the histories at the node, and use Chou's algorithm to determine the 2-way partition of histories that results in the greatest improvement in the Gini index of l_4.

   Record this optimal partition in terms of the subsets $l_3 \in \mathcal{A}_3$ and $l_3 \in \overline{\mathcal{A}}_3$, where $\overline{\mathcal{A}}_3$ denotes not the complement of $\mathcal{A}_3$ in the 27-letter alphabet, but the letters from the set-complement of $\mathcal{A}_3$ that are *actually observed* at this node.

3. Repeat this exercise with the equivalence classes β_i derived from the identity of l_2 instead of l_3, and once more with β_i derived from the identity of l_1. Record the improvement in Gini index in each case, as well as the optimal partitions $l_2 \in \mathcal{A}_2$ vs $l_2 \in \overline{\mathcal{A}}_2$ and $l_1 \in \mathcal{A}_1$ vs $l_1 \in \overline{\mathcal{A}}_1$.

   Again, $\overline{\mathcal{A}}_2$ and $\overline{\mathcal{A}}_1$ need to be explicitly recorded, since they aren't merely the set-complements of $\mathcal{A}_2$ and $\mathcal{A}_1$ respectively, but the actually observed subsets thereof.

4. Of the three alternative optimal partitions, namely $\mathcal{A}_1 \cup \overline{\mathcal{A}}_1$ or $\mathcal{A}_2 \cup \overline{\mathcal{A}}_2$ or $\mathcal{A}_3 \cup \overline{\mathcal{A}}_3$, choose the one that yields the highest improvement in the Gini index on the development data, and make it the candidate split for this node.

5. Compute the improvement in the Gini index of the *held out* data for this candidate split. If the improvement is above a threshold, make this split permanent. If not, declare this node as terminal.

   Remark: Note that some samples (histories) in the held out data may have neither $l_i \in \mathcal{A}_i$ nor $l_i \in \overline{\mathcal{A}}_i$, where i denotes the letter position upon which the candidate split is based. How will you treat such histories? Discuss your chosen tie-breaking procedure.

   This "unseen" history problem, or more appropriately *unassigned* history problem may also occur on the test data. Does your tie-breaking procedure extend to such cases?

6. Carry out the recursion from Step 2 until all nodes in the tree are marked as terminal.

Since the Chou algorithm only guarantees finding a local optimum, you may want to try several random starts for the determination of $\mathcal{A}_i$ vs $\overline{\mathcal{A}}_i$ in the steps above.

Compute the perplexity of the test data (B) using the decision tree developed above. Should it become necessary, use some smoothing technique.

4   Optional: Comparison of Gini Index and Entropy

Repeat the construction of the letter-class based decision tree language model of the previous section using Entropy instead of the Gini index in Chou's algorithm, and compare the two resulting tree. Also compare the tree built using entropy and Chou's algorithm with the tree built using entropy and the bit encoding.

Discuss the relative merits and weaknesses of the various models and methods.