ÉCOLE POLYTECHNIQUE FÉDÉRALE DE LAUSANNE
School of Computer and Communication Sciences

Handout 10 Information Theory and Coding
Notes on Tunstall Coding Oct. 12, 2015

Variable to Fixed Length Source Coding — Tunstall Coding

These notes are based largely on notes by P. A. Humblet and R. G. Gallager.

We first prove an auxiliary result about computing the expectation of non-negative integer valued random variables:

Theorem 1. *If N is a non-negative integer valued random variable then*

$$E[N] = \sum_{n=0}^{\infty} \Pr(N > n).$$

Proof.

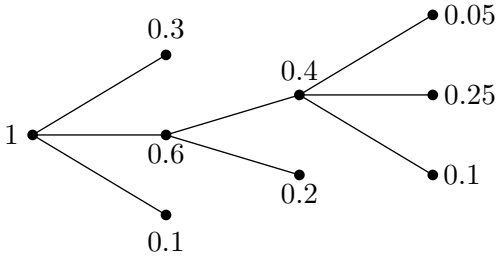
$$\begin{aligned} E[N] &= \sum_{i=1}^{\infty} i \Pr(N = i) \\ &= \Pr(N = 1) \\ &\quad + \Pr(N = 2) + \Pr(N = 2) \\ &\quad + \Pr(N = 3) + \Pr(N = 3) + \Pr(N = 3) \\ &\quad + \Pr(N = 4) + \Pr(N = 4) + \Pr(N = 4) + \Pr(N = 4) \\ &\quad + \quad \vdots \quad + \quad \vdots \quad + \quad \vdots \quad + \quad \vdots \\ &= \Pr(N > 0) + \Pr(N > 1) + \Pr(N > 2) + \Pr(N > 3) + \cdots \end{aligned}$$

□

Corollary. *Given a tree with probabilities assigned to the leaves, let the probability of an intermediate node be defined as the sum of the probabilities of all leaves that descend from that node. Then, the expected depth of a leaf is given by the sum of the probabilities of all intermediate nodes including the root.*

Proof. The sum of the probabilities of the intermediate nodes at depth i is equal to the probability that a leaf has depth strictly greater than i. □

Example 1.



$$3 \times (0.05) + 3 \times (0.25) + 3 \times (0.1) + 2 \times (0.2) + 1 \times (0.3) + 1 \times (0.1) = 1 + 0.6 + 0.4$$

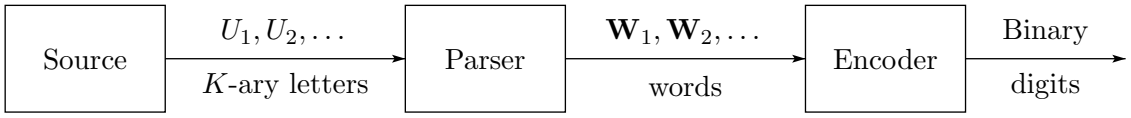
Parsers and Dictionaries

A *parser* segments the sequence of letters from a source into a concatenation of words. For example, the source string

```
a b a a c b a a b a c a a a b c ...
```

may be parsed as the concatenation of `ab`, `aac`, `b`, `aab`, `ac`, `aaa`, `b`, `c`, ...

The conceptual situation looks like this:



The parser does its job with the help of a dictionary of size M. Such a dictionary is a collection of words $\mathbf{w}_1, \ldots, \mathbf{w}_M$, where each $\mathbf{w}_m$ is a sequence of source letters. We will map each dictionary entry into a binary b-tuple; we will choose b so that $M \leq 2^b$ which ensures that a distinct mapping is possible.

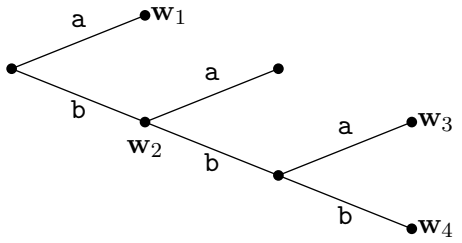
Given a dictionary of strings $\mathbf{w}_1, \ldots, \mathbf{w}_M$, the parser picks the *longest* prefix of the source sequence $U_1, U_2, \ldots$ that is in the dictionary (say, $U_1, \ldots, U_L$) and then continues with $U_{L+1}, U_{L+2}, \ldots$.

Example 2. For a binary source with letters `a` and `b`, and a dictionary with 4 words $\mathbf{w}_1 =$ `a`, $\mathbf{w}_2 =$ `b`, $\mathbf{w}_3 =$ `bba`, $\mathbf{w}_4 =$ `bbb`, the source sequence `abbababba...` would be parsed as `a`, `bba`, `b`, `a`, `bba`, ...

Definition. A dictionary $\{\mathbf{w}_1, \ldots, \mathbf{w}_M\}$ is *valid* if every infinite sequence of source letters has a prefix in the dictionary.

If the source alphabet $\mathcal{U}$ contains K letters, we can associate a unique, labelled, K-ary tree with a dictionary $\mathbf{w}_1, \ldots, \mathbf{w}_M$:

Example 3. The tree associated with the dictionary in the previous example is given by

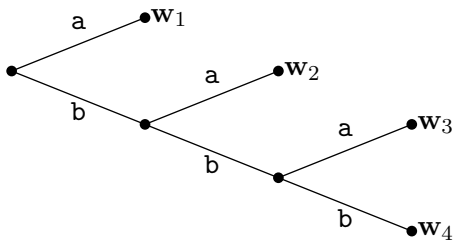


From the definition, we see that a dictionary is valid, if and only if there is at least one dictionary word on every path that connects the root to a leaf.

Definition. A dictionary is *prefix-free*, if no dictionary word is a prefix of another. Such a dictionary is also called *instantaneously encodable*, since the parser can emit the dictionary word as soon as the source sequence is seen to form a dictionary word.

From the definition we see that a words of a valid and prefix-free dictionary are all the leaves (and only the leaves) of a K-ary tree.

Example 4. The dictionary in Example 2 is not prefix-free. On the other hand, the dictionary consisting of the words $\mathbf{w}_1 = \mathtt{a}$, $\mathbf{w}_2 = \mathtt{ba}$, $\mathbf{w}_3 = \mathtt{bba}$, and $\mathbf{w}_4 = \mathtt{bbb}$ is valid and prefix-free. The corresponding tree is



Consider building such a tree starting from the root: at the first stage, we have K leaves and 1 intermediate node (the root). Each time we convert a leaf to an intermediate node, we replace it with K leaves that are its children. Thus, a set of $K-1$ leaves are added each time a new intermediate node is made. This gives us the relation between the number of leaves M and the number of intermediate nodes α of the tree representation of a valid, prefix-free dictionary:

$$M = 1 + (K-1)\alpha.$$

To use a valid, prefix-free dictionary, and represent dictionary words with binary strings of b bits, we choose α such that $M \leq 2^b < M + K - 1$, i.e., we make the dictionary size M as large as possible without exceeding 2^b.

The expected number of source letters $E[L]$ encoded by a dictionary word is simply the expected length of a dictionary word. This can be found from the tree representation of the dictionary. Label each leaf by the probability of the corresponding word and label each intermediate node by the sum of the probabilities of all leaves growing from that node. Using the corollary we proved in the preliminaries, $E[L]$ is given by the sum of the probabilities of the intermediate nodes including the root.

Entropy of a Dictionary

Theorem 2. *If the dictionary is valid and prefix-free and the source is memoryless (i.e., $U_1, U_2, \ldots$ is an i.i.d. sequence) then the parsed sequence $\mathbf{W}_1, \mathbf{W}_2, \ldots$ of words is also memoryless. Moreover $H(\mathbf{W})$ equals $E[L]H(U)$.*

Proof. Because the dictionary is prefix-free, the parser can emit $\mathbf{W}_1$ before it sees any letters of $\mathbf{W}_2, \mathbf{W}_3, \ldots$. Thus, knowing that $\mathbf{W}_1 = u_1 u_2 \ldots u_l$ only tells us that the first l letters of the source were $U_1 = u_1, \ldots, U_l = u_l$. Since the source emits independent letters, it thus follows that knowing $\mathbf{W}_1$ does not give us any information about $U_{l+1}, U_{l+2}, \ldots$ which will form $\mathbf{W}_2, \mathbf{W}_3, \ldots$, and so we conclude that $\mathbf{W}_1, \mathbf{W}_2, \ldots$ are independent. Furthermore, since the source letters are not only independent but identically distributed, $U_{l+1}, U_{l+2}, \ldots$ have the same statistics as $U_1, U_2, \ldots$, it then follows that the random variables $\mathbf{W}_1, \mathbf{W}_2, \ldots$, are not only independent independent but identically distributed as well. Let P_W denote the distribution of the $\mathbf{W}_1$. It now remains to show that $H(\mathbf{W}_1) = E[L]H(U)$.

Since $\mathbf{W}_1$ is a function of the sequence of source letters, so is the length of $\mathbf{W}_1$, and so is $\log P_W(\mathbf{W}_1)$. (Note that $\mathbf{W}_1$ is a random variable, thus length$(\mathbf{W}_1)$, $\log P_W(\mathbf{W}_1)$ are also random variables). As the length of $\mathbf{W}_1$ is a function of $U_1, U_2, \ldots$, denote this length by $l(U_1, , U_2, \ldots)$. Since the source letters are independent, we see that

$$\Pr(\mathbf{W}_1) = \prod_{j=1}^{l(U_1, U_2, \ldots)} P_U(U_j)$$

and by taking logarithms

$$\log P(\mathbf{W}_1) = \sum_{j=1}^{l(U_1,U_2,\dots)} \log P_U(U_j).$$

Let $1\{a\}$ equal to 1 if a is true and 0 if a is false. Noticing that an index $j \geq 1$ participates in the sum above if and only if $l(U_1, U_2, \dots) \geq j$, we can write

$$\log P(\mathbf{W}_1) = \sum_{j=1}^{l(U_1,U_2,\dots)} \log P_U(U_j) = \sum_{j=1}^{\infty} 1\{l(U_1, U_2, \dots) \geq j\} \log P_U(U_j),$$

and taking expectations

$$H(\mathbf{W}_1) = -\sum_{j=1}^{\infty} E\big[\log(P_U(U_j))1\{l(U_1, U_2, \dots)\} \geq j\}\big].$$

Since the dictionary is prefix free, the event $\{l(U_1, U_2, \dots) \leq j-1\}$ is determined by $U_1, \dots, U_{j-1}$ and thus is independent of U_j. Since $\{l(U_1, U_2, \dots) \geq j\}$ is the complement of this event it is also independent of U_j. Thus

$$\begin{aligned} E\big[\log(P_U(U_j))1\{l(U_1, U_2, \dots) \geq j\}\big] &= E\big[\log(P_U(U_j))\big] E\big[1\{l(U_1, U_, \dots) \geq j\}\big] \\ &= -H(U) \Pr(l(U_1, U_2, \dots) \geq j) \end{aligned}$$

Thus,

$$\begin{aligned} H(\mathbf{W}_1) &= H(U) \sum_{j=1}^{\infty} \Pr(l(U_1, U_2, \dots) \geq j) \\ &= H(U) \sum_{j=0}^{\infty} \Pr(l(U_1, U_2, \dots) > j) \\ &= H(U) E[L] \end{aligned}$$

concluding the proof. □

An alternate proof of this theorem is given in the exercises.

Tunstall Construction

In general, we would like to construct the dictionary for a given M so as to maximize $E[L]$. This will minimize $b/E[L]$, the number of binary digits per average number of source letters encoded. In light of the corollary proved in the preliminaries, we want to keep the most likely α nodes as intermediate nodes. This is easy; pick out the intermediate nodes one by one in decreasing order of probability, starting at the root. Each node picked will have all its ancestors already picked since an ancestor of a node has higher probability than the node. Because of this, the intermediate nodes picked as such will form the intermediate nodes of a tree. We thus have:

Tunstall algorithm

1. Start with the root as intermediate node and all level 1 nodes as leaves.
2. If number of leaves is equal to the desired dictionary size stop.
3. Otherwise, pick the highest probability leaf, make it an intermediate node and grow K leaves on it. Goto step 2.

Analysis of the Tunstall algorithm

Let Q be the probability of the last intermediate node picked by the Tunstall algorithm. Since at the time it was picked this intermediate node was the largest probability leaf, we see that Q is an upper bound to the probability of any leaf, i.e., $P(\text{leaf}) \leq Q$.

Furthermore, since intermediate nodes are picked in decreasing order of probability, every intermediate node has probability at least Q. Let $P_{\min}$ be the probability of the least probable source letter. Since the probability of a leaf is given by the probability of its immediate ancestor times the probability of a source letter, $P(\text{leaf}) \geq QP_{\min}$. We thus see that

$$QP_{\min} \leq P(\text{leaf}) \leq Q.$$

Summing over all leaves in the left inequality we conclude that $MQP_{\min} \leq 1$ and thus $1/Q \geq MP_{\min}$. From the right hand inequality we see that

$$-\log P(\text{leaf}) \geq \log(1/Q) \geq \log(MP_{\min}).$$

Taking expectations we see that $H(\text{leaves}) \geq \log(MP_{\min})$. Now, $H(\text{leaves}) = H(U)E[L]$. Thus

$$\begin{aligned} b &< \log(M+K-1) \\ &= \log(M) + \log(1+(K-1)/M) \\ &= \log(MP_{\min}) + \log(1/P_{\min}) + \log(1+(K-1)/M) \\ &\leq H(U)E[L] + \log(1/P_{\min}) + \log(1+(K-1)/M) \end{aligned}$$

and thus

$$\frac{b}{E[L]} < H(U) + \frac{1}{E[L]}[\log(1/P_{\min}) + \log(1+(K-1)/M)].$$

By choosing b large, we are choosing M large and thus $E[L]$ large. This makes the term excess of $H(U)$ approach zero. Thus, by taking a large dictionary, the number of bits per source letter the scheme uses can be made as close to $H(U)$ as desired.