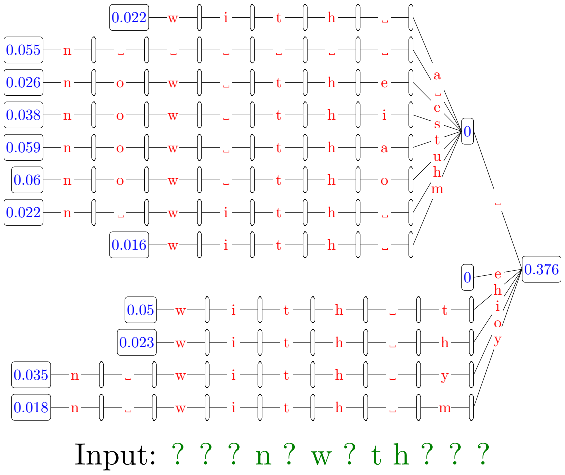
A. Pruning a hierarchical decomposition

To provide further intuition for how our method behaves, we have included the hierarchical decomposition for one of the test examples from our experiments:



This is the hierarchical decomposition used to infer the missing characters for the phrase *...???n?w?th???...*. The decomposition doesn't waste resources representing the first 3 unknown characters, and maintains plausible hypotheses for the hidden characters such as *...with a...*, *...now thee...*, and *...now this...*. Each blue decimal number indicates a region in the decomposition together with the local probability mass assigned to that region.

B. Pruning a hierarchical decomposition

In our inference algorithm for choosing a good hierarchical decomposition B, we had two major steps: refining the decomposition, and pruning it back down to a given size k. In this appendix, we will provide a dynamic programming algorithm for computing an optimal pruning B of A, assuming that $\mathtt{Fit}(a, \mathcal{C}_B(a))$ depends only on a. Let $\hat{p}_{\theta_A}$ be the approximating distribution corresponding to A, and $\hat{p}_{\theta_B}$ be the approximating distribution corresponding to B. Our goal is to minimize $\mathrm{KL}\left(p^* \parallel \hat{p}_{\theta_B}\right)$; we will make the assumption that A and θ_A are chosen well enough that $\hat{p}_{\theta_A}$ is already close to p^*, and thus that $\mathrm{KL}\left(\hat{f}_{\theta_A} \,\middle\|\, \hat{f}_{\theta_B}\right)$ is a good surrogate for $\mathrm{KL}\left(p^* \parallel \hat{p}_{\theta_B}\right)$. We will also ignore normalization constants and instead consider the divergence $\mathrm{KL}\left(\hat{f}_{\theta_A} \,\middle\|\, \hat{f}_{\theta_B}\right)$ between the unnormalized distributions $\hat{f}_{\theta_A}$ and $\hat{f}_{\theta_B}$. Formally, we will solve the following problem:

Given a hierarchical decomposition A, and assuming that $\mathtt{Fit}(a, \mathcal{C}_B(a))$ depends only on a, find the subset $B \subseteq A$ of vertices of size k such that $\mathrm{KL}\left(\hat{f}_{\theta_A} \,\middle\|\, \hat{f}_{\theta_B}\right)$ is minimized.

For a hierarchical decomposition A and a subset B of A, let $\alpha_B(a)$ denote the smallest $b \in B$ such that $a \subseteq b$. By equation (4), we have

$$\begin{aligned}
\mathrm{KL}\left(\hat{f}_{\theta_A} \,\middle\|\, \hat{f}_{\theta_B}\right) &= \sum_{a \in A} \mathrm{KL}_{a^\circ}\left(\hat{f}_{\theta_A} \,\middle\|\, \hat{f}_{\theta_B}\right) && (19)\\
&= \sum_{a \in A} \mathrm{KL}_{a^\circ}\left(\hat{f}_{\theta_a} \,\middle\|\, \hat{f}_{\theta_{\alpha_B(a)}}\right) && (20)\\
&= \sum_{a \in A} K_{a^\circ}(a \| \alpha_B(a)), && (21)
\end{aligned}$$

where $K_c(a\|b) \overset{\text{def}}{=} \sum_{x \in c} \hat{f}_{\theta_a}(x) \log\left(\frac{\hat{f}_{\theta_a}(x)}{\hat{f}_{\theta_b}(x)}\right)$. It is here that we make use of the assumption that $\mathtt{Fit}(\alpha_B(a), \mathcal{C}_B(\alpha_B(a)))$ depends only on a; otherwise, $K_{a^\circ}(a\|\alpha_B(a))$ would depend on the particular value of $\mathcal{C}_B(\alpha_B(a))$.

In the remainder of this appendix, we will write out a succession of recursive formulas for computing $\mathrm{KL}\left(\hat{f}_{\theta_A} \,\middle\|\, \hat{f}_{\theta_B}\right)$, expanding the state space each time until we eventually have a recursion for optimizing $\mathrm{KL}\left(\hat{f}_{\theta_A} \,\middle\|\, \hat{f}_{\theta_B}\right)$ over all subsets $B \subseteq A$ of size k.

Computing $\mathrm{KL}\left(\hat{f}_{\theta_A} \,\middle\|\, \hat{f}_{\theta_B}\right)$ for fixed B. To make the expresion in (21) more amenable to dynamic programming, we will write it out recursively. For $a \subseteq p$, define $D(a,p)$ to be the contribution of the descendants of a (including a) to $\mathrm{KL}\left(\hat{f}_{\theta_A} \,\middle\|\, \hat{f}_{\theta_B}\right)$ assuming that $\alpha_B(a) = p$. More formally, we define $D(a,p)$ recursively as

$$D(a,p) \overset{\text{def}}{=} \begin{cases} K_{a^\circ}(a\|a) + \sum_{b \in \mathcal{C}_A(a)} D(b,a) & : \quad a \in B \\ K_{a^\circ}(a\|p) + \sum_{b \in \mathcal{C}_A(a)} D(b,p) & : \quad a \notin B \end{cases} . \qquad (22)$$

(Note that $K_{a^\circ}(a\|a)$ is equal to 0; we have left it in the recursion to expose the symmetry in the two cases.)

With this definition, one can verify that $D(X,X)$ expands out to (21) and hence is equal to $\mathrm{KL}\left(\hat{f}_{\theta_A} \,\middle\|\, \hat{f}_{\theta_B}\right)$. (Recall that X is the entire state space and is always an element of A.)

Optimizing over B. Equation (22) gives us a recursive formula for $\mathrm{KL}\left(\hat{f}_{\theta_A} \,\middle\|\, \hat{f}_{\theta_B}\right)$ when B is fixed. However, the only dependence on B is in deciding between the two cases in the recursion, so it is easy to extend the recursion to simultaneously choose B. In particular, define the three-variable function $D(a,p,m)$ to be the minimum value of $D(a,p)$ if there are m elements in B that are contained in a (including, possibly, a itself). We then have the recursion

$$D(a,p,m) \tag{23}$$
$$\stackrel{\text{def}}{=} \min \left\{ \begin{array}{l} K_{a^\circ}(a\|a) + \min_{\sum m_b = m-1} \left[\sum_{b \in \mathcal{C}_A(a)} D(b,a,m_b)\right], \\ K_{a^\circ}(a\|p) + \min_{\sum m_b = m} \left[\sum_{b \in \mathcal{C}_A(a)} D(b,p,m_b)\right]. \end{array} \right.$$

The first case corresponds to including a in B, in which case we have $m-1$ remaining elements of B to distribute among the descendants of a. The second case corresponds to excluding a from B, in which case we have m elements of B to distribute. Now $D(X,X,k)$ is the minimum value of $D(X,X)$ across all subsets B of size k, which is the quantity we are after.

Computing the minimum tractably. We are almost done, but we need an efficient way to compute the minimum over all m_j that sum to m. To do this, number the children of a as $b_1, b_2, \ldots$, and define the *four-variable* function $D(a,p,m,j)$, which, intuitively, tracks the minimum value of $D(a,p)$ if there are m elements in B left to be distributed among children $b_j, b_{j+1}, \ldots$ and their subtrees. More formally, define $D(a,p,m,j)$ via

$$D(a,p,m,j) \tag{24}$$
$$\stackrel{\text{def}}{=} \begin{cases} \min\{D(a,a,m-1,0), D(a,p,m,0)\} & : \ j=-1 \\ \min_{0 \le m' \le m} \{D(b_j,p,m',-1) \\ \qquad + D(a,p,m-m',j+1)\} & : \ 0 \le j < |\mathcal{C}_A(a)| \\ K_{a^\circ}(a\|p) & : \ j = |\mathcal{C}_A(a)|. \end{cases}$$

The three cases can be thought of as follows:

- $D(a,p,m,-1)$ decides whether or not to include a in B
- $D(a,p,m,j)$ decides how many elements of B to include among the descendants of b_j
- $D(a,p,m,|\mathcal{C}_A(a)|)$ computes the local contribution of a to $\mathrm{KL}\left(\hat{f}_{\theta_A} \,\middle\|\, \hat{f}_{\theta_B}\right)$.

Overall, then, $D(X,X,k,-1)$ is equal to the minimum value of $\mathrm{KL}\left(\hat{f}_{\theta_A} \,\middle\|\, \hat{f}_{\theta_B}\right)$ over all $B \subseteq A$ with $|B| = k$.

Runtime. Suppose that the decomposition A has depth d. Then there are $O(|A|d)$ triples (p,a,j), so the size of the state space is $O(|A|dk)$. Furthermore, the first case of the recursion can be computed in $O(1)$ time, the second case in $O(k)$ time, and the final case in $O(1)$ time (on average across all a). Therefore, the runtime is $O(|A|dk^2)$.