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:

```
Input: ??? n?w?th???...
```

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 $\text{Fit}(a, C_B(a))$ 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 $\text{KL}(p^* \parallel \hat{p}_{\theta_B})$; we will make the assumption that $A$ and $\theta_A$ are chosen well enough that $\hat{p}_{\theta_A}$ is already close to $p^*$, and thus that $\text{KL}(\hat{f}_{\theta_A} \parallel \hat{f}_{\theta_B})$ is a good surrogate for $\text{KL}(p^* \parallel \hat{p}_{\theta_B})$. We will also ignore normalization constants and instead consider the divergence $\text{KL}(\hat{f}_{\theta_A} \parallel \hat{f}_{\theta_B})$ 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 $\text{Fit}(a, C_B(a))$ depends only on $a$, find the subset $B \subseteq A$ of vertices of size $k$ such that $\text{KL}(\hat{f}_{\theta_A} \parallel \hat{f}_{\theta_B})$ 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 (19), we have

\[
\text{KL}(\hat{f}_{\theta_A} \parallel \hat{f}_{\theta_B}) = \sum_{a \in A} \text{KL}_{\alpha^+}(\hat{f}_{\theta_A} \parallel \hat{f}_{\theta_B})
\]

where $\text{KL}_{\alpha}(a \parallel \alpha_B(a)) = \sum_{x \in C(a)} \hat{f}_{\theta_A}(x) \log \left( \frac{\hat{f}_{\theta_A}(x)}{\hat{f}_{\theta_B}(x)} \right)$. It is here that we make use of the assumption that $\text{Fit}(\alpha_B(a), C_B(\alpha_B(a)))$ depends only on $a$; otherwise, $\text{KL}_{\alpha}(a \parallel \alpha_B(a))$ would depend on the particular value of $C_B(\alpha_B(a))$. In the remainder of this appendix, we will write out a succession of recursive formulas for computing $\text{KL}(\hat{f}_{\theta_A} \parallel \hat{f}_{\theta_B})$, expanding the state space each time until we eventually have a recursion for optimizing $\text{KL}(\hat{f}_{\theta_A} \parallel \hat{f}_{\theta_B})$ over all subsets $B \subseteq A$ of size $k$.

Computing $\text{KL}(\hat{f}_{\theta_A} \parallel \hat{f}_{\theta_B})$ for fixed $B$. To make the expression 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 $\text{KL}(\hat{f}_{\theta_A} \parallel \hat{f}_{\theta_B})$ assuming that $\alpha_B(a) = p$. More formally, we define $D(a, p)$ recursively as

\[
D(a, p) \overset{\text{def}}{=} \left\{ \begin{array}{ll}
K_{\alpha^+}(a \parallel a) + \sum_{b \in C(a)} D(b, a) & : a \in B \\
K_{\alpha^+}(a \parallel p) + \sum_{b \in C(a)} D(b, p) & : a \notin B
\end{array} \right.
\]

(22)

(Note that $K_{\alpha^+}(a \parallel a)$ 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 $\text{KL}(\hat{f}_{\theta_A} \parallel \hat{f}_{\theta_B})$. (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 $\text{KL}(\hat{f}_{\theta_A} \parallel \hat{f}_{\theta_B})$ 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 ex-
tend 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

$$D(a, p, m) \overset{\text{def}}{=} \min \left\{ \begin{array}{l} K_{a^+}(a || a) + \min_{m_{b} = m-1} \left[ \sum_{b \in C_A(a)} D(b, a, m_b) \right], \\ K_{a^+}(a || p) + \min_{m_{b} = m} \left[ \sum_{b \in C_A(a)} D(b, p, m_b) \right]. \end{array} \right. \quad (23)$$

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

$$D(a, p, m, j) \overset{\text{def}}{=} \begin{cases} \min \{ D(a, a, m - 1, 0), D(a, p, m, 0) \} : j = -1 \\ \min_{0 \leq m' \leq m} \{ D(b_j, p, m', -1) + D(a, p, m - m', j + 1) \} : 0 \leq j < |C_A(a)| \\ K_{a^+}(a || p) : j = |C_A(a)|. \end{cases} \quad (24)$$

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 $b_j$
- $D(a, p, m, |C_A(a)|)$ computes the local contribution of $a$ to $\text{KL} \left( \hat{f}_{\theta_A} \parallel \hat{f}_{\theta_B} \right)$.

Overall, then, $D(X, X, k, -1)$ is equal to the minimum value of $\text{KL} \left( \hat{f}_{\theta_A} \parallel \hat{f}_{\theta_B} \right)$ 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 $O(|A|^{dk^2})$. 

---

**Filtering with Abstract Particles**