Predictive Querying for Autoregressive Neural Sequence Models

Anonymous Author(s)
Affiliation
Address
email

Abstract

In reasoning about sequential events it is natural to pose probabilistic queries such as "when will event A occur next" or "what is the probability of A occurring before B," with applications in areas such as user modeling, medicine, and finance. However, with machine learning shifting towards neural autoregressive models such as RNNs and transformers, probabilistic querying has been largely restricted to simple cases such as next-event prediction. This is in part due to the fact that future querying involves marginalization over large path spaces, which is not straightforward to do efficiently in such models. In this paper we introduce a general typology for predictive queries in autoregressive sequence models and show that such queries can be systematically represented by sets of elementary building blocks. We leverage this typology to develop new query estimation methods based on beam search, importance sampling, and hybrids. Across four large-scale sequence datasets from different application domains, as well as for the GPT-2 language model, we demonstrate the ability to make query answering tractable for arbitrary queries in exponentially-large predictive path-spaces, and find clear differences in cost-accuracy tradeoffs between search and sampling methods.

1 Introduction

One of the major successes in machine learning in recent years has been the development of neural sequence models for categorical sequences, particularly in natural language applications but also in other areas such as automatic code generation and program synthesis [Shin et al., 2019, Chen et al., 2021], computer security [Brown et al., 2018], recommender systems [Wu et al., 2017], genomics [Shin et al., 2021, Amin et al., 2021], and survival analysis [Lee et al., 2019]. Many of the models (although not all) rely on autoregressive training and prediction, allowing for the sequential generation of sequence completions in a recursive manner conditioned on sequence history.

A natural question in this context is how to compute answers to predictive queries that go beyond traditional one-step-ahead predictions. Examples of such queries are "how likely is event A to occur before event B?" and "how likely is event C to occur (once or more) within the next K steps of the sequence?" These types of queries are very natural across a wide variety of application contexts, for example, the probability that an individual will finish speaking or writing a sentence within the next K words, or that a user will use one app before another. See Fig. 1 and the Appendix for examples.

In this paper we develop a general framework for answering such predictive queries in the context of autoregressive (AR) neural sequence models. This amounts to computing conditional probabilities of propositional statements about future events, conditioned on the history of the sequence as summarized by the current hidden state representation. We focus in particular on how to perform near real-time computation of such queries, motivated by use-cases such as answering human-generated queries and utilizing query estimates within the optimization loop of training a model. Somewhat

Submitted to 36th Conference on Neural Information Processing Systems (NeurIPS 2022). Do not distribute.

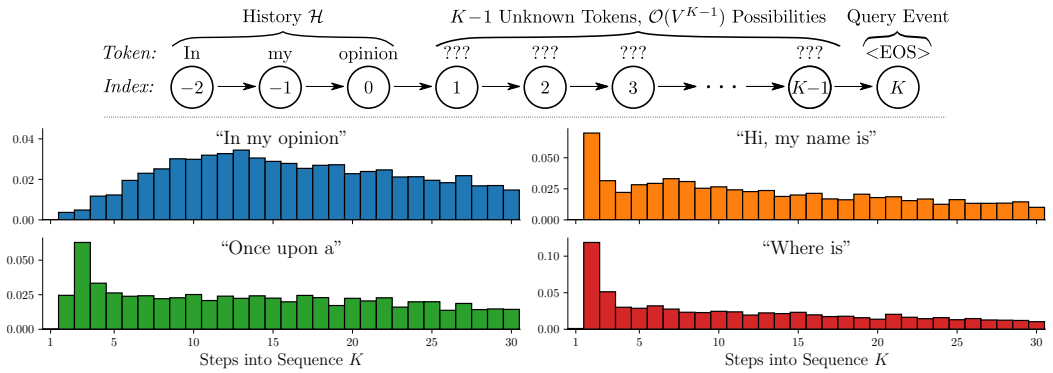

Figure 1: (top) Illustration of a query for the probability of a given sentence ending in K steps. (bottom) GPT-2 query estimates across 4 prefixes with $V = 50,257, K \leq 30$. Importance sampling query estimates maintain a 6x reduction in variance relative to naive model sampling for the same computation budget. (Additional details provided in Sections 4, 5, and the Appendix).

surprisingly, although there has been extensive prior work on multivariate probabilistic querying in areas such as graphical models and database querying, as well as for restricted types of queries for traditional sequence models such as Markov models, querying for neural sequence models appears to be unexplored. One possible reason is that the problem is computationally intractable in the general case (as we discuss later in Section 3), typically scaling as $O(V^{K-1})$ or worse for predictions K-steps ahead, given a sequence model with a V-ary alphabet (e.g. compared to $O(KV^2)$ for Markov chains).

Our contributions in this work are three-fold:

1. We introduce and develop the problem of **predictive querying** in neural sequence models.
2. We propose a novel query representation framework that yields a natural proposal distribution over the restricted query path space. This distribution is beneficial for estimating queries through importance sampling, novel beam search variants, or an innovative search/sampling hybrid.
3. We perform a systematic evaluation with a variety of different real world datasets on the different success and failure modes of these estimation methods, with an emphasis on query computation in real-time, producing new insights about failure modes.

Upon publication, all code and trained models will be available through a public GitHub repository.

2 Related Work

Research on efficient computation of probabilistic queries has a long history in machine learning and AI, going back to work on exact inference in multivariate graphical models [Pearl, 1988, Koller and Friedman, 2009]. Queries in this context are typically of two types. The first are *conditional probability queries*, the focus of our attention here: computing probabilities over a subset X of variables of interest, conditioned on a second subset $Y = y$ of observed variable values, and marginalizing over the set Z of all other variables. The second type of queries can broadly be referred to as *assignment queries*, seeking the most likely (highest conditional probability) assignment of values x for X, again conditioned on $Y = y$ and marginalizing over the set Z. Assignment queries are also referred to as most probable explanation (MPE) queries, or as maximum a posteriori (MAP) queries when Z is the empty set [Koller et al., 2007].

For models that can be characterized with sparse Markov dependence structure, there is a significant body of work on efficient inference algorithms that can leverage the Markov structure [Koller and Friedman, 2009], in particular for sequential models where recursive computation can be effectively leveraged [Bilmes, 2010]. However, autoregressive neural sequence models are inherently non-Markov since the real-valued current hidden state is a function of the entire history of the sequence. Each hidden state vector induces a tree containing V^k unique future trajectories with state-dependent probabilities for each path. Techniques such as dynamic programming (used effectively in

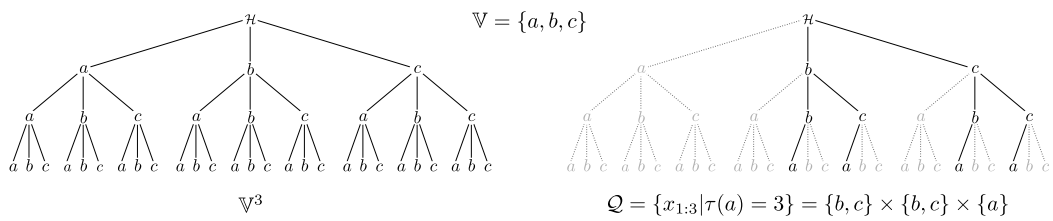

Figure 2: Tree diagram of the complete sequence space for a vocabulary $\mathbb{V} = \{a, b, c\}$ and the corresponding query space $\mathcal{Q}$ for $p^*_\theta(\tau(a) = 3)$, defined as the set product of restricted domains.

Markov-structured sequence models) are not applicable, and both assignment queries and conditional probability queries are NP-hard in general (e.g., Chen et al. [2017]).

For assignment-type queries there has been considerable work in natural language processing with neural sequence models, particularly for the MAP problem of generating high-quality/high-probability sequences conditioned on sequence history or other conditioning information. A variety of heuristic decoding methods have been developed and found to be useful in practice, including beam search [Sutskever et al., 2014], best-first search [Xu and Durrett, 2021], sampling methods [Holtzman et al., 2019], and hybrid variants [Shaham and Levy, 2022]. However, for conditional probability queries and neural sequence models (the focus of this paper), there has been no prior work in general on this problem to our knowledge. While decoding techniques such as beam search can also be useful in the context of conditional probability queries, as we will see later in Section 4, such techniques have significant limitations in this context, since by definition they produce lower-bounds on the probabilities of interest and, hence, are biased estimators.

3 Probabilistic Queries

Notation Let $X_{1:N} := [X_1, X_2, \ldots, X_N]$ be a sequence of random variables with arbitrary length N. Additionally, let $x_{1:N} := [x_1, x_2, \ldots, x_N]$ be their respective observed values where each x_i takes on values from a fixed vocabulary $\mathbb{V} := \{1, \ldots, V\}$. Examples of these sequences include sentences where each letter or word is a single value, or streams of discrete events generated by some process or user. We will refer to individual variable-value pairs in the sequence as events.

We consider an autoregressive model $p_\theta(X_{1:N}) = \prod_{i=1}^{N} p_\theta(X_i | X_{1:i-1})$, parameterized by θ and trained on a given dataset of M independent draws from a ground truth distribution $\mathbb{P}(X_{1:N})$. We assume that this model can be conditioned on a subsequence, termed the *history* $\mathcal{H}$. We will remap the indices of subsequent random variables to start at 1. [1] We abbreviate conditioning on a history by an asterisk *, i.e., $\mathbb{P}^*(\cdot) := \mathbb{P}(\cdot|\mathcal{H})$ and $p^*_\theta(\cdot) := p_\theta(\cdot|\mathcal{H})$.

Defining Probabilistic Queries Given a specific history of events $\mathcal{H}$, there are a variety of different questions one could ask about the future continuing where the history left off: (Q1) What event is likely to happen next? (Q2) Which events are likely to occur $k > 1$ steps from now? (Q3) What is the distribution of when the next instance of $a \in \mathbb{V}$ occurs? (Q4) How likely is it that we will see event $a \in \mathbb{V}$ occur before $b \in \mathbb{V}$? (Q5) How likely is it for $a \in \mathbb{V}$ occur n times in the next k steps?

We define a common framework for such queries by defining probabilistic queries to be of the form $p^*_\theta(X_{1:K} \in \mathcal{Q})$ with $\mathcal{Q} \subset \mathbb{V}^K$. This can be extended to the infinite setting (i.e., $p^*_\theta([X_k]_k \in \mathcal{Q})$ where $\mathcal{Q} \subset \mathbb{V}^\infty$). Exact computation of an arbitrary query is straightforward to represent:

$$p^*_\theta(X_{1:K} \in \mathcal{Q}) = \sum_{x_{1:K} \in \mathcal{Q}} p^*_\theta(X_{1:K} = x_{1:K}) = \sum_{x_{1:K} \in \mathcal{Q}} \prod_{k=1}^{K} p^*_\theta(X_k = x_k | X_{<k} = x_{<k}). \quad (1)$$

Depending on $|\mathcal{Q}|$, performing this calculation can quickly become intractable, motivating lower bounds or approximations (developed in more detail in Section 4). In this context it is helpful to

[1] For example, if $|\mathcal{H}| = 3$ then $\mathbb{P}(X_4|\mathcal{H})$ is distribution of the 7th value in a sequence after conditioning on the first 3 values in the sequence.

#	Question	Probabilistic Query	$K \cdot \lvert\mathcal{Q}\rvert$
Q1	Next event?	$p^*_\theta(X_1)$	$\mathcal{O}(1)$
Q2	Event K steps from now?	$p^*_\theta(X_K)$	$\mathcal{O}(V^{K-1})$
Q3	Next instance of a?	$p^*_\theta(\tau(a) = K)$	$\mathcal{O}((V-1)^{K-1})$
Q4	Will a happen before b?	$p^*_\theta(\tau(a) < \tau(b))$	$\mathcal{O}((V-2)^{K})^\dagger$
Q5	How many instances of a in K steps?	$p^*_\theta(N_a(K) = n)$	$\mathcal{O}(\binom{K}{n}(V-1)^{K-n})$

Table 1: List of example questions, corresponding probabilistic queries, and associated costs of exact computation computation with an autoregressive model. The cost of accommodating a history $\mathcal{H}$ is assumed to be an additive constant for all queries. While Q4 extends to infinite time, the cost reported is for computing a lower bound up to K steps.

impose structure on the query $\mathcal{Q}$ to make subsequent estimation easier, in particular by breaking $\mathcal{Q}$ into the following structured partition:

$$\mathcal{Q} = \cup_i \mathcal{Q}^{(i)} \text{ where } \mathcal{Q}^{(i)} \cap \mathcal{Q}^{(j)} = \emptyset \text{ for } i \neq j \tag{2}$$

$$\text{and } \mathcal{Q}^{(i)} = \mathcal{V}_1^{(i)} \times \mathcal{V}_2^{(i)} \times \cdots \times \mathcal{V}_K^{(i)} \text{ where } \mathcal{V}_k^{(i)} \subseteq \mathbb{V} \text{ for } k = 1, \ldots, K. \tag{3}$$

In words, this means a given query $\mathcal{Q}$ can be broken into a partition of simpler queries $\mathcal{Q}^{(i)}$ which take the form of a set cross product between restricted domains $\mathcal{V}_k^{(i)}$, one domain for each token X_k.[2] An illustration of an example query set can be seen in Fig. 2. A natural consequence of this is that:

$$p^*_\theta(X_{1:K} \in \mathcal{Q}) = \sum_i p^*_\theta(X_{1:K} \in \mathcal{Q}^{(i)}) = \sum_i p^*_\theta(\cap_{k=1}^K X_k \in \mathcal{V}_k^{(i)}), \tag{4}$$

which lends itself to more easily estimating each term in the sum. This will be discussed in Section 4.

Queries of Interest All of the queries posed earlier in this section can be represented under the framework detailed in Eq. (2) and Eq. (3), as illustrated in Table 1.

Q1 & Q2 The queries $p^*_\theta(X_1 = a)$ and $p^*_\theta(X_K = a)$ for some $a \in \mathbb{V}$ can be represented with $\mathcal{Q} = \{a\}$ and $\mathcal{Q} = \mathbb{V}^{K-1} \times \{a\}$ respectively.

Q3 The probability of the next instance of $a \in \mathbb{V}$ occurring at some point in time $K \geq 1$, $p^*_\theta(\tau(a) = K)$ where $\tau(\cdot)$ is the *hitting time*, can be represented as $\mathcal{Q} = (\mathbb{V} \setminus \{a\})^{K-1} \times \{a\}$. This can be adapted for a set $A \subset \mathbb{V}$ by replacing $\{a\}$ in $\mathcal{Q}$ with A.

Q4 The probability of $a \in \mathbb{V}$ occurring before $b \in \mathbb{V}$, $p^*_\theta(\tau(a) < \tau(b))$, is represented as $\mathcal{Q} = \cup_{i=1}^\infty \mathcal{Q}^{(i)}$ where $\mathcal{Q}^{(i)} = (\mathbb{V} \setminus \{a, b\})^{i-1} \times \{a\}$. Lower bounds to this can be computed by limiting $i < i'$. Like Q3, this can also be extended to disjoint sets $A, B \subset \mathbb{V}$.

Q5 The probability of $a \in \mathbb{V}$ occurring n times in the next K steps, $p^*_\theta(N_a(K) = n)$, is represented as $\mathcal{Q} = \cup_{i=1}^{C(K,n)} \mathcal{Q}^{(i)}$, where $N_a(K)$ is a random variable for the number of occurrences of events of type a from steps 1 to K and $\mathcal{Q}^{(i)}$'s are defined to cover all unique permutations of orders of products composed of: $\{a\}^n$ and $(\mathbb{V} \setminus \{a\})^{K-n}$. Like above, this can easily be extended for $A \subset \mathbb{V}$.

Query Complexity From Eq. (1), exact computation of a query involves computing $K \cdot |Q|$ conditional distributions (e.g., $p^*_\theta(X_k | X_{<k} = x_{<k})$) in an autoregressive manner. Under the structured representation, the number of conditional distributions needed is equivalently $\sum_i \prod_{k=1}^K |\mathcal{V}_k^{(i)}|$. Non-attention based neural sequence models often define $p^*_\theta(X_k | X_{<k} = x_{<k}) := f_\theta(h_k)$ where $h_k = \text{RNN}_\theta(h_{k-1}, x_{k-1})$. As such, the computation complexity for any individual conditional distribution remains constant with respect to sequence length. We will refer to the complexity of this atomic action as being $\mathcal{O}(1)$. Naturally, the actual complexity depends on the model architecture and has a multiplicative scaling on the cost of computing a query. The number of atomic operations needed to exactly compute Q1-Q5 for this class of models can be found in Table 1. Should p_θ be an attention based model (e.g., a transformer [Vaswani et al., 2017]) then the time complexity of computing a single one-step-ahead distribution becomes $\mathcal{O}(K)$, further exacerbating the **exponential**

[2] Ideally, the partitioning is chosen to have the smallest number of $\mathcal{Q}^{(i)}$'s needed.

growth of many queries. Note that with some particular parametric forms of p_θ, query computation can be more efficient, e.g., see the Appendix for details on query complexity for Markov models.

4 Query Estimation Methods

Since exact query computation can scales exponentially in K it is natural to consider approximation methods. In particular we focus on importance sampling, beam search, and a hybrid of both methods. All methods will be based on a novel proposal distribution, discussed next.

4.1 Proposal Distribution

For various estimation methods which will be discussed later, it is beneficial to have a proposal distribution $q(X_{1:K} = x_{1:K})$ whose domain matches that of the query $\mathcal{Q}$. For importance sampling, we will need this distribution as a proposal distribution, while we use it as our base model for selecting high-probability sequences in beam search. We would like the proposal distribution to resemble our original model while also respecting the query. One thought is to have $q(X_{1:K} = x_{1:K}) = p^*_\theta(X_{1:K} = x_{1:K} | X_{1:K} \in \mathcal{Q})$, but note that computing this probability involves normalizing over $p^*_\theta(X_{1:K} \in \mathcal{Q})$ which is exactly what we are trying to estimate in the first place. Instead of restricting the *joint* distribution to the query, we can instead restrict every *conditional* distribution to the query's restricted domain. To see this, we first partition $\mathcal{Q} = \cup_i \mathcal{Q}^{(i)}$ and define an autoregressive proposal distribution for each $\mathcal{Q}^{(i)} = \prod_{k=1}^{K} \mathcal{V}_k^{(i)}$ as follows:

$$q^{(i)}(X_{1:K} = x_{1:K}) = \prod_{k=1}^{K} p^*_\theta(X_k = x_k | X_{<k} = x_{<k}, X_k \in \mathcal{V}_k^{(i)}) \tag{5}$$

$$= \prod_{k=1}^{K} \frac{p^*_\theta(X_k = x_k | X_{<k} = x_{<k}) \mathbb{1}(x_k \in \mathcal{V}_k^{(i)})}{\sum_{v \in \mathcal{V}_k^{(i)}} p^*_\theta(X_k = v | X_{<k} = x_{<k})} \tag{6}$$

where $\mathbb{1}(\cdot)$ is the indicator function. That is, we constrain the outcomes of each conditional probability to the restricted domains $\mathcal{V}_k^{(i)}$ and renormalize them accordingly. To evaluate the proposal distribution's probability, we multiply all conditional probabilities according to the chain rule. Since the entire distribution is computed for a single model call $p^*_\theta(X_k | X_{<k} = x_{<k})$, it is possible to both sample a K-length sequence and compute its likelihood under $q^{(i)}$ with only K model calls. Thus, we can efficiently sample sequences from a distribution that is both informed by the underlying model p_θ and that respects the given domain $\mathcal{Q}$.

4.2 Estimation Techniques

Sampling One can naively sample any arbitrary probability value using Monte Carlo samples to estimate $p^*_\theta(X_{1:K} \in \mathcal{Q}) = \mathbb{E}_{p^*_\theta}[\mathbb{1}(X_{1:K} \in \mathcal{Q})]$; however, this typically will have high variance. This can be substantially improved upon by estimating the correct expectation. Arbitrary queries can be written as a sum of probabilities of individual sequences, as seen in Eq. (1). This summation can be equivalently written as an expected value,

$$p^*_\theta(X_{1:K} \in \mathcal{Q}) = \sum_{x_{1:K} \in \mathcal{Q}} p^*_\theta(X_{1:K} = x_{1:K}) = |\mathcal{Q}| \, \mathbb{E}_{x_{1:K} \sim \mathcal{U}(\mathcal{Q})} \left[p^*_\theta(X_{1:K} = x_{1:K}) \right], \tag{7}$$

where $\mathcal{U}$ is a uniform distribution. It is common for p^*_θ to concentrate most of the available probability mass on a small subset of the total possible space $\mathbb{V}^K$. Should $|\mathcal{Q}|$ be large, then $|\mathcal{Q}| \, p^*_\theta(X_{1:K} = x_{1:K})$ will have very large variance for $x_{1:K} \sim \mathcal{U}(\mathcal{Q})$. One way to mitigate this is to use importance sampling with our proposal distribution q (Section 4.1):

$$p^*_\theta(X_{1:K} \in \mathcal{Q}) = |\mathcal{Q}| \, \mathbb{E}_{x_{1:K} \sim \mathcal{U}(\mathcal{Q})} \left[p^*_\theta(X_{1:K} = x_{1:K}) \right] = \mathbb{E}_{x_{1:K} \sim q} \left[\frac{p^*_\theta(X_{1:K} = x_{1:K})}{q(X_{1:K} = x_{1:K})} \right] \tag{8}$$

$$\approx \frac{1}{M} \sum_{m=1}^{M} \frac{p^*_\theta(X_{1:K} = x_{1:K}^{(m)})}{q(X_{1:K} = x_{1:K}^{(m)})} \qquad \text{for } x_{1:K}^{(1)}, \ldots, x_{1:K}^{(M)} \overset{iid}{\sim} q. \tag{9}$$

It is worth noting that this estimator could be further improved by augmenting the sampling process to produce samples without replacement from q (e.g., [Meister et al., 2021, Kool et al., 2019, Shi et al., 2020]); for the purposes of the scope of this paper we focus on sampling with replacement.

Search An alternative to estimating a query by sampling is to instead produce a lower bound,

$$p^*_\theta(X_{1:K} \in \mathcal{Q}) = \sum_{x_{1:K} \in \mathcal{Q}} p^*_\theta(X_{1:K} = x_{1:K}) \geq \sum_{x_{1:K} \in \mathcal{B}} p^*_\theta(X_{1:K} = x_{1:K}), \tag{10}$$

where $\mathcal{B} \subset \mathcal{Q}$. In many situations, only a small subset of sequences $x_{1:K}$ in $\mathcal{Q}$ have a non-negligible probability of occurring due to the vastness of the total path space $\mathbb{V}^K$. As such, it is possible for $|\mathcal{B}| \ll |\mathcal{Q}|$ while still having a minimal gap between the lower bound and the actual query value.

One way to produce a set $\mathcal{B} \subset \mathcal{Q}$ is through beam search. To ensure that beam search only explores the query space, instead of searching with p_θ we utilize q for ranking beams. Since beam search is a greedy algorithm and for a given conditional $q(X_k = a|X_{<k}) \propto p^*_\theta(X_k = a|X_{<k})$ for $a \in \mathcal{V}_k$, the rankings will both respect the domain and be otherwise identical to using p_θ to rank. Typically, the goal of beam search is to find the most likely completion of a sequence without having to explore the entire space of possible sequences. This is accomplished by greedily selecting the top-B most likely next step continuations, or *beams*, at each step into the future. Rather than finding a few high-likelihood beams, we are more interested in accumulating a significant amount of probability and less so in the specific quantity of beams collected.

Traditional beam search has a fixed beam size B that is considered for its search; however, this is not ideal for collecting probability mass. As an alternative we develop *coverage-based* beam search where at each step in a sequence we restrict the set of beams being considered not to the top-B but rather to the smallest set of beams that collectively exceed a predetermined probability mass α, referred to as the "coverage".[3] More specifically, let $\mathcal{B}_k \subset \{x_{1:k}|x_{1:K} \in \mathcal{Q}\}$ be a set containing $|\mathcal{B}_k|$ beams for subsequences of length k. For brevity, we will assume that $\mathcal{Q} = \mathcal{V}_1 \times \cdots \times \mathcal{V}_K$.[4] $\mathcal{B}_{k+1}$ is a subset of $\mathcal{B}_k \times \mathcal{V}_{k+1}$ and is selected specifically to minimize $|\mathcal{B}_{k+1}|$ such that $q(X_{1:k+1} \in \mathcal{B}_{k+1}) \geq \alpha$. It can be shown that $p^*_\theta(X_{1:K} \in \mathcal{Q}) - p^*_\theta(X_{1:K} \in \mathcal{B}_K) \leq 1 - q(X_{1:K} \in \mathcal{B}_K)$ (and is often quite less). See Appendix for a proof.

There is one slight problem with having α be constant throughout the search. Since we are pruning based on the joint probability of the entire sequence, any further continuations of $\mathcal{B}_k$ will reduce the probability $q(X_{1:k+1} \in B_{k+1})$. This may lead to situations in which every possible candidate sequence is kept in order to maintain minimal joint probability coverage. This can be avoided by filtering by α_k where $\alpha_1 > \cdots > \alpha_K = \alpha$, e.g., the geometric series $\alpha_k = \alpha^{k/K}$.

A Hybrid Approach Importance sampling produces an unbiased estimate, but can still experience large variance in spite of a good proposal distribution q when p_θ is a heavy tailed distribution. Conversely, the beam search lower bound can be seen as a biased estimate with zero variance. We can remedy the limitations of both methods by recognizing that since $p^*_\theta(X_{1:K} \in \mathcal{Q}) = \sum_{x_{1:K} \in \mathcal{B}_K} p^*_\theta(X_{1:K} = x_{1:K}) + \sum_{x_{1:K} \in \mathcal{Q} \setminus \mathcal{B}_K} p^*_\theta(X_{1:K} = x_{1:K})$ where $\mathcal{B}_K$ is the set of sequences resulting from beam search, we can use importance sampling on the latter summation. The only caveat to this is that the proposal distribution must match the same domain of the summation: $\mathcal{Q} \setminus \mathcal{B}_K$.

The proposal distribution we use to accomplish this is $q(X_{1:K} = x_{1:K}|X_{1:K} \notin \mathcal{B}_K)$ (see Eq. 5). This is implemented by storing all intermediate distributions (both original p_θ and proposal q) found during beam search and arranging them into a tree structure. All leaf nodes that are associated with $x_{1:K} \in \mathcal{B}_K$ have their transition probability zeroed out and then the effect of restriction and normalization is propagated up the tree to their ancestor nodes (more details provided in the Appendix). Sampling from this new proposal distribution involves sampling from this normalized tree until we reach either an existing K-depth leaf node, in which case we are done, or a $< K$-depth leaf node in which case we complete the rest of the sequence by sampling from q from Eq. (5) as usual.

Lastly, since our ultimate goal is to sample from the long tail of p_θ, targeting a specific coverage α during beam search is no longer effective. Instead, we propose *tail-splitting* beam search to better match our goals. Let $w_k^{(i)} = p^*_\theta(X_{1:k+1} = x^{(i)}_{1:k+1})$ for $x^{(i)}_{1:k+1} \in \mathcal{B}_k \times \mathcal{V}_{k+1}$ such that $w_k^{(i)} \geq w_k^{(j)}$ if $i < j$. In this regime, $\mathcal{B}_{k+1} = \{x^{(i)}_{1:k+1}\}_{i=1}^B$ where $B = \arg\min_b \sigma(w_k^{(1:b)}) + \sigma(w_k^{(b+1:|\mathcal{B}_k \times \mathcal{V}_{k+1}|)})$. $\sigma(w_k^{(u:v)})$ is the empirical variance of $w_k^{(i)}$ for $i = u, \ldots, v$. This can be seen as performing 2-means clustering on w_k's and taking the cluster with higher cumulative probability.

[3] This is similar to the distinction between top-K and top-p / nucleus sampling commonly used for natural language generation [Holtzman et al., 2019].

[4] If $\mathcal{Q}$ requires partitioning into multiple $\mathcal{Q}^{(i)}$'s, we apply beam search to each sub query $p^*_\theta(X_{1:K} \in \mathcal{Q}^{(i)})$.

4.3 Saving Computation on Multiple Queries

Should multiple queries need to be performed, such as $p^*_\theta(\tau(a) = k)$ for multiple values of k, then there is potential to be more efficient in computing estimates for all of them. The feasibility of reusing intermediate computations is dependent on the set of queries considered. For simplicity, we will consider two base queries $\mathcal{Q} = \mathcal{V}_1 \times \cdots \times \mathcal{V}_K$ and $\mathcal{Q}' = \mathcal{V}'_1 \times \cdots \times \mathcal{V}'_{K'}$ where $K < K'$. Due to the autoregressive nature of p_θ, if $\mathcal{V}_i = \mathcal{V}'_i$ for $i = 1, \ldots, K-1$ then all of the distributions and sequences needed for estimating $p^*_\theta(X_{1:K} \in \mathcal{Q})$ are guaranteed to be intermediate results found when estimating $p^*_\theta(X_{1:K'} \in \mathcal{Q}')$. To be explicit, when estimating the latter query with beam search the intermediate $\mathcal{B}_K$ is the same as what would be directly computed for the former query. Likewise, for importance sampling if $x_{1:K'} \sim q(X_{1:K'})$ using Eq. (5) then the subsequence $x_{1:K} \sim q(X_{1:K})$. This does not apply when the sample path domain is further restricted, such as with the hybrid approach, in which case we cannot directly use intermediate results to compute other queries for "free."

5 Experiments and Results

Experimental Setting We investigate the hitting time query on various datasets, comparing beam search, importance sampling, and the hybrid method. We find that hybrid outperforms both pure search and sampling for a comparable computation budget on various data. We also investigate the dependence of all three methods on the model entropy.

Datasets We evaluate our query estimation methods on three user behavior and two language sequence datasets. **Reviews** [Ni et al., 2019] contains 60 million sequences of Amazon customer reviews for products belonging to one of $V = 29$ categories; **Mobile Apps** [Aliannejadi et al., 2021] consists of app usage records over $V = 8$ unique applications; **MOOCs** consists of student interaction with online course resources over $V = 98$ actions. We also use the works of William **Shakespeare** [sha] by modeling the occurrence of $V = 67$ unique ASCII characters. Lastly, we examine **WikiText** [Merity et al., 2016] to explore word-level sequence modeling applications with GPT-2, a large-scale language model with a vocabulary of $V = 50,257$ word-pieces [Radford et al., 2019, Wu et al., 2016]. After preprocessing, none of the datasets contain any personal identifiable information.

Base Models While our proposed methods are amenable to autoregressive models of arbitrary structure, we focus our analysis on recurrent neural networks to model sequences. For all datasets except WikiText, we train Long-short Term Memory (LSTM) networks until convergence. To explore modern applications of sequence modeling with WikiText, we utilize GPT-2 with pre-trained weights from HuggingFace [Wolf et al., 2020]. Model training and experimentation utilized roughly 200 NVIDIA GeForce 2080ti GPU hours. Please refer to the Appendix for additional details.

Experimental Methods We investigate computation-accuracy trade-offs between 3 estimation methods (beam search, importance sampling, and the hybrid) across all datasets. Query histories $\mathcal{H}$ are defined by randomly sampling $N = 1000$ sequences from the test split for all datasets except WikiText, from which we sample only $N = 100$ sequences due to computational limitations. For each query history and method, we compute the hitting time query estimate $p^*_\theta(\tau(a) = K)$ over $K = 3, \ldots, 11$, with a determined by the Kth symbol of the ground truth sequence. We focus exclusively on hitting time queries in our experiments as often more advanced queries decompose into operations over individual hitting times.

To ensure an even comparison of query estimators, we fix the computation budget per query in terms of model calls $f_\theta(h_k)$ to be equal across all 3 methods, repeating experiments for different budget magnitudes roughly corresponding to $O(10), O(10^2), O(10^3)$ model calls (see Appendix for full details). We intentionally select relatively small computation budgets per query to support systematic large-scale experiments over multiple queries up to relatively large values of K. Results for queries with GPT-2 are further restricted because of computational limits and are reported separately below.

To evaluate the accuracy of the estimates for each query and method, we compute the true probability of K using exact computation for small values of $K = 3, 4$. For larger values of K, we run importance sampling with a large number of samples S, where S is adapted per query to ensure the resulting unbiased estimate has an empirical variance less than $\epsilon \ll 1$ (see Appendix). This computationally-expensive estimate is then used as a surrogate for ground truth in error calculations.

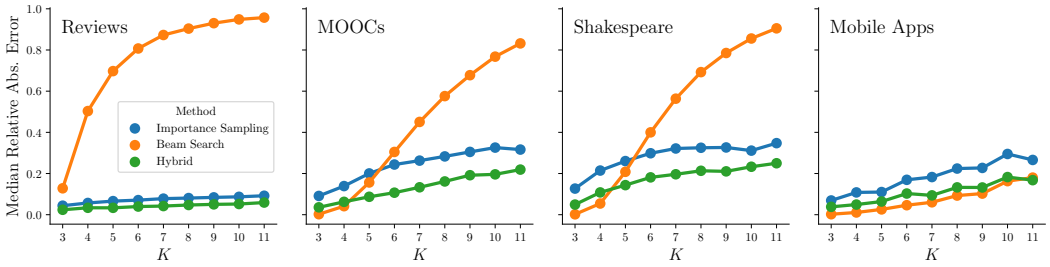

Figure 3: Median relative absolute error (RAE) between estimated probability and (surrogate) ground truth for $p^*_\theta(\tau(\cdot) = K)$ for importance sampling, beam search, and the hybrid method. As query path space grows with K, beam search quickly fails to bound ground truth while sampling remains robust, with the hybrid consistently outperforming all other methods, especially for large values of K.

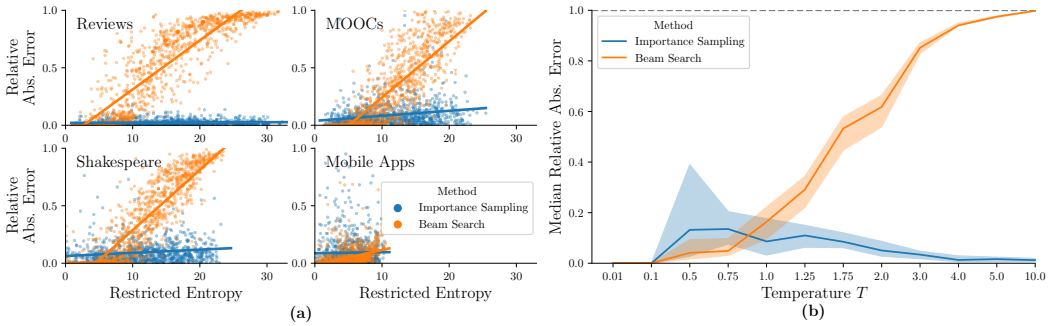

Figure 4: (left) RAE vs restricted entropy per query (with best linear fits), (right) Median RAE versus model temperature T. All errors computed using the same queries as in Fig. 3. Beam search errors correlate highly with model entropy, and increasing temperature T directly induces this failure mode.

Coverage-based beam search is not included in our results: we found that it experiences exponential growth with respect to K and does not scale efficiently due to its probability coverage guarantees. Similarly, we found that the naive sampling procedure (see Section 4) to be far inferior to beam or sampling-based methods and did not include it. Additional details are provided in the Appendix.

Results: Accuracy and Query Horizon Using the methodology described above, for each query, we compute the relative absolute error (RAE) $|p - \hat{p}|/p$, where $\hat{p}$ is the estimated query probability generated by a particular method and p is the ground truth probability or the surrogate estimate using importance sampling. For each dataset, for each of the 3 levels of computation budget, for each value of K, this yields $N = 1000$ errors for the N queries for each method.

Fig. 3 shows the median RAE of the N queries, per method, as a function of K, for each dataset, using the medium computation budget. Across the 4 datasets the error increases systematically as K increases. However, beam search is significantly less robust than the other methods for 3 of the 4 datasets: the error rate increases more rapidly in a non-linear fashion compared to importance sampling and hybrid. Beam search is also quite sensitive to characteristics of individual datasets, versus the relatively predictable error characteristics of the other two methods. The hybrid method systematically outperforms importance sampling along across all 4 datasets and for all values of$K$. In the Appendix we provide additional results; for the lowest and highest levels of computational budget, for mean (instead of median) RAEs, and scatter plots for specific values of K with more detailed error information, with consistent qualitative conclusions across all plots.

Results: The Effect of Model Entropy We conjecture that the entropy of the proposal distribution q conditioned on a given history $H^*_q(X_{1:K}) = -\mathbb{E}_{x_{1:K}\sim q}[\log q(X_{1:K} = x_{1:K})]$, which we refer to as *restricted entropy*, is a major factor in the performance of the estimation methods. Fig. 4(a) shows the RAE per query (with a linear fit) as a function of estimated restricted entropy for importance sampling and beam search. The results clearly show that entropy is driving the query error in general

and that the performance of beam search is much more sensitive to entropy than sampling. The difference in entropy characteristics across datasets explains the differences in errors we see in Fig. 3. In particular, the Mobile Apps dataset is in a much lower entropy regime than the other 3 datasets.

To further investigate the effect of entropy, we alter each model by applying a temperature $T > 0$ to every conditional factor: $p_{\theta,T}(X_k|X_{<k}) \propto p_\theta(X_k|X_{<k})^{1/T}$, effectively changing the entropy ranges for the models. Fig. 4(b) shows the median RAE, for query $p^*_{\theta,T}(\tau(\cdot) = 4)$, as a function of model temperature for the Mobile Apps data. As predicted from Fig. 4(a), the increase in T, and corresponding increase in entropy, causes beam search's error to converge to 1, while the sampling error goes to 0. As T increases, each individual sequence will approach having $1/|\mathcal{Q}|$ mass, thus needing many more beams to have adequate coverage. Results for other queries and the other 3 datasets (in the Appendix) further confirm the fundamental bifurcation of error between search and sampling (that we see in Fig. 4(b)) as a function of entropy.

Results: Queries with a Large Language Model We further explore entropy's effect on query estimation with GPT-2 and the WikiText dataset for $N = 100$, $K = 3, 4$, across 3 computation budgets. With a vocabulary 500x larger than the other models, GPT-2 allows us to examine queries suitable for modern applications. The resulting high entropy causes beam search to fail to match surrogate ground truth given the computation budgets (consistent with earlier experiments), with a median RAE of 82% (for $K = 4$ and a budget of $O(10^3)$). By contrast, importance sampling's median RAE under the same setting is 13%, **a 6x reduction**. Additional results are in the Appendix.

6 Discussion

Future Directions: This work provides a starting point for multiple different research directions beyond the scope of this paper. One such direction is exploring more powerful search and sampling methods for query estimation. This includes methods for sampling without replacement, such as the Gumbel-max method [Huijben et al., 2022], as well as new heuristics for automatically trading off search and sampling on a per-query basis. Another direction for exploration is amortized methods, such as learning models before queries are issued, that are specifically designed to help answer queries. Learning models that include queries as part of regularization terms in objective functions can also build on this work, e.g., learning models that don't only rely on one-step-ahead losses Meister and Cotterell [2021]. This work could also be broadened to continuous-time models for marked temporal point processes (requiring marginalization over time as well as event types).

Limitations: Our results rely on only four datasets with a single autoregressive neural sequence model trained for each, naturally limiting the breadth of conclusions that we can draw. However, given that the results are consistent and validated by an algorithm-independent entropy analysis, we believe these findings have general validity and provide a useful starting point for others interested in the problem of querying neural sequence models. Another potential limitation is that our four datasets have relatively small vocabularies ($V = O(10^2)$, small by NLP standards); this choice was largely driven by computational limitations in terms of being able to conduct conclusive experiments (e.g. averaging over many event histories). Our (limited) results with GPT-2 provide clues on what may be achievable with much larger vocabularies: systematic analysis of querying for such models is a natural target for future work. Our work also does not address the issue of model inaccuracy: our results are entirely focused on computing query estimates in terms of a model's distribution instead of the data-generating process. Exploring the effect of miscalibration errors in autoregressive models on k-step-ahead query estimates is a promising avenue for future work.

Potential Negative Societal Impacts: Since the focus of this work is making predictions with data that is typically generated by individuals (language, online activity), there is naturally a potential for abuse of such methods. For example, if the underlying model in a system is miscalibrated and users are unaware, decisions could be made that negatively impact individuals. Examples include recommending a student be dropped from an online course if the model incorrectly predicts they will not participate in course modules. Even if the underlying model is well-calibrated, predictive queries could potentially be used in a proactive manner to bias decisions against individuals whose event sequences are atypical, e.g., in a chatbot context, inaccurately predicting future language patterns for certain individuals, leading to interruption and generation of an inappropriate response.

References

The complete works of William Shakespeare: Project Gutenberg. https://ocw.mit.edu/ans7870/6/6.006/s08/lecturenotes/files/t8.shakespeare.txt. Accessed: 2020-04-11.

Mohammad Aliannejadi, Hamed Zamani, Fabio Crestani, and W. Bruce Croft. Context-aware target apps selection and recommendation for enhancing personal mobile assistants. *ACM Trans. Inf. Syst.*, 39(3), may 2021.

Alan Amin, Eli N Weinstein, and Debora Marks. A generative nonparametric Bayesian model for whole genomes. *Advances in Neural Information Processing Systems*, 34:27798–27812, 2021.

Jeff Bilmes. Dynamic graphical models. *IEEE Signal Processing Magazine*, 27(6):29–42, 2010.

Andy Brown, Aaron Tuor, Brian Hutchinson, and Nicole Nichols. Recurrent neural network attention mechanisms for interpretable system log anomaly detection. In *Proceedings of the First Workshop on Machine Learning for Computing Systems*, pages 1–8, 2018.

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde de Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, et al. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*, 2021.

Yining Chen, Sorcha Gilroy, Andreas Maletti, Jonathan May, and Kevin Knight. Recurrent neural networks as weighted language recognizers. *arXiv preprint arXiv:1711.05408*, 2017.

Ari Holtzman, Jan Buys, Li Du, Maxwell Forbes, and Yejin Choi. The curious case of neural text degeneration. In *International Conference on Learning Representations*, 2019.

Iris AM Huijben, Wouter Kool, Max Benedikt Paulus, and Ruud JG Van Sloun. A review of the Gumbel-max trick and its extensions for discrete stochasticity in machine learning. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2022.

Daphne Koller and Nir Friedman. *Probabilistic Graphical Models: Principles and Techniques*. MIT Press, 2009.

Daphne Koller, Nir Friedman, Lise Getoor, and Ben Taskar. *Introduction to Statistical Relational Learning*, volume 43, chapter Graphical models in a nutshell. MIT press Cambridge, MA, 2007.

Wouter Kool, Herke Van Hoof, and Max Welling. Stochastic beams and where to find them: The Gumbel-top-k trick for sampling sequences without replacement. In Kamalika Chaudhuri and Ruslan Salakhutdinov, editors, *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 3499–3508. PMLR, 09–15 Jun 2019. URL https://proceedings.mlr.press/v97/kool19a.html.

Changhee Lee, Jinsung Yoon, and Mihaela Van Der Schaar. Dynamic-deephit: A deep learning approach for dynamic survival analysis with competing risks based on longitudinal data. *IEEE Transactions on Biomedical Engineering*, 67(1):122–133, 2019.

Clara Meister and Ryan Cotterell. Language model evaluation beyond perplexity. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 5328–5339, 2021.

Clara Isabel Meister, Afra Amini, Tim Vieira, and Ryan Cotterell. Conditional poisson stochastic beams. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 664–681. Association for Computational Linguistics, 2021.

Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. Pointer sentinel mixture models, 2016.

Jianmo Ni, Jiacheng Li, and Julian McAuley. Justifying recommendations using distantly-labeled reviews and fine-grained aspects. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 188–197, Hong Kong, China, November 2019. Association for Computational Linguistics.

Judea Pearl. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann, 1988.

Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language models are unsupervised multitask learners. 2019.

Uri Shaham and Omer Levy. What do you get when you cross beam search with nucleus sampling? *arXiv preprint arXiv:2107.09729*, 2022.

Kensen Shi, David Bieber, and Charles Sutton. Incremental sampling without replacement for sequence models. In *International Conference on Machine Learning*, pages 8785–8795. PMLR, 2020.

Eui Chul Shin, Miltiadis Allamanis, Marc Brockschmidt, and Alex Polozov. Program synthesis and semantic parsing with learned code idioms. *Advances in Neural Information Processing Systems*, 32, 2019.

Jung-Eun Shin, Adam J Riesselman, Aaron W Kollasch, Conor McMahon, Elana Simon, Chris Sander, Aashish Manglik, Andrew C Kruse, and Debora S Marks. Protein design and variant prediction using autoregressive generative models. *Nature Communications*, 12(1):1–11, 2021.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. Sequence to sequence learning with neural networks. *Advances in Neural Information Processing Systems*, 27, 2014.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander M. Rush. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45, Online, October 2020. Association for Computational Linguistics. URL `https://www.aclweb.org/anthology/2020.emnlp-demos.6`.

Chao-Yuan Wu, Amr Ahmed, Alex Beutel, Alexander J Smola, and How Jing. Recurrent recommender networks. In *Proceedings of the tenth ACM international conference on web search and data mining*, pages 495–503, 2017.

Yonghui Wu, Mike Schuster, Z. Chen, Quoc V. Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, Jeff Klingner, Apurva Shah, Melvin Johnson, Xiaobing Liu, Lukasz Kaiser, Stephan Gouws, Yoshikiyo Kato, Taku Kudo, Hideto Kazawa, Keith Stevens, George Kurian, Nishant Patil, Wei Wang, Cliff Young, Jason R. Smith, Jason Riesa, Alex Rudnick, Oriol Vinyals, Gregory S. Corrado, Macduff Hughes, and Jeffrey Dean. Google's neural machine translation system: Bridging the gap between human and machine translation. *ArXiv*, abs/1609.08144, 2016.

Jiacheng Xu and Greg Durrett. Massive-scale decoding for text generation using lattices. *arXiv preprint arXiv:2112.07660*, 2021.

Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes]
   (b) Did you describe the limitations of your work? [Yes] See Section 6.
   (c) Did you discuss any potential negative societal impacts of your work? [Yes] See Section 6.
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes] We mention computational resources, PII considerations, and other ethical factors in our paper directly.
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [Yes] See Section 3, 4
   (b) Did you include complete proofs of all theoretical results? [Yes] See Appendix.
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [Yes] Included in supplement as a .zip file
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes] See Appendix.
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [No] We gather data for multiple query histories and report median results for a given experimental setting.
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes] See Section 5.
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [Yes] We cite all public datasets utilized.
   (b) Did you mention the license of the assets? [Yes] Datasets used are all publicly available and the specific license is noted in the Appendix for each.
   (c) Did you include any new assets either in the supplemental material or as a URL? [Yes]
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [N/A] Datasets are publicly available.
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [Yes] See Section 5.
5. If you used crowdsourcing or conducted research with human subjects...
   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]
   (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]
   (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]