Speculative Sampling via Exponential Races

Szymon Kobus
Electrical and Electronic Engineering
Imperial College London
`szymon.kobus17@imperial.ac.uk`

Deniz Gündüz
Electrical and Electronic Engineering
Imperial College London

Abstract

Speculative decoding accelerates large language model inference using a smaller draft model. In this paper, we establish a surprising connection between speculative sampling and the concept of channel simulation from information theory, which aims at simulating a noisy channel using as few bits as possible. This connection allows us to provide an information-theoretic analysis of the speed up that can be achieved by speculative sampling. Leveraging this link, we derive an explicit relation between generation speed-up and the number of tokens k generated by the draft model for large k, which serves as an upper bound for all k. We also propose a novel speculative sampling method via exponential races called ERSS that matches state-of-the-art performance.

1 Introduction

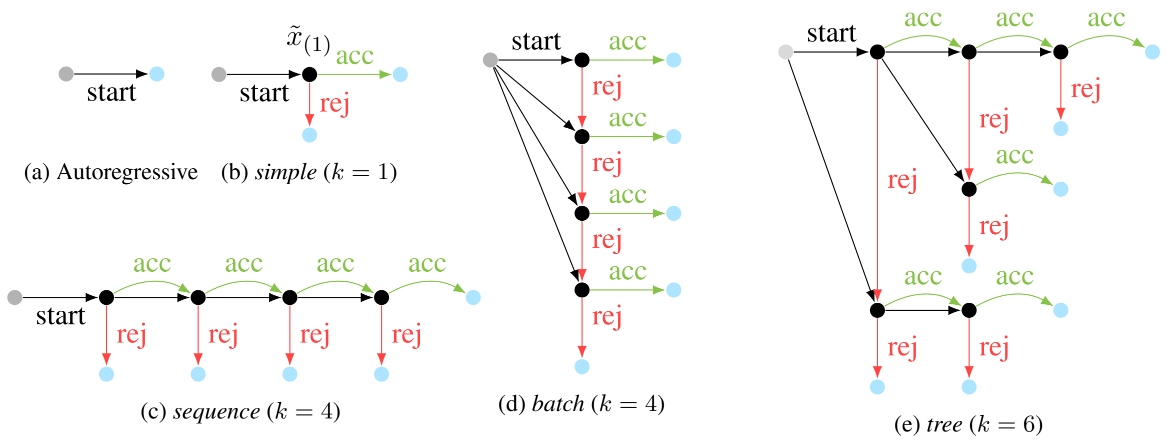
Transformer-based large language models (LLMs) are at the forefront of the AI revolution, driving rapid advancements across numerous applications. However, as their adoption accelerates, the speed of text generation emerges as a critical bottleneck, alongside compute and memory constraints. Standard LLM generation involves calculating the conditional probability distribution using the target model P given a partial output $x_{:n} = (x_1, x_2, ..., x_n)$, which is the concatenation of the initial context and all previously generated tokens. The target model P outputs the conditional probability distribution $P(\,\cdot \mid x_{:n})$, from which a single token x_{n+1} is sampled and concatenated to $x_{:n}$ to form the updated sequence $x_{:n+1}$. These steps are repeated iteratively until a stopping condition is met, such as reaching a maximum sequence length or generating an end-of-sequence token. This auto-regressive token-by-token approach inherently leads to slow generation speeds.

Speculative decoding accelerates this process by generating multiple draft tokens for each target model evaluation using a more efficient draft model. These draft tokens are then verified by the target model, and some of them are chosen and appended to the input token sentence, while guaranteeing **identical output quality**; that is, the distribution of generated text remains the same as standard auto-regressive decoding. In more detail, speculative sampling consists of the following three steps:

1. **Drafting**–a smaller draft model Q is used to generate one or more possible continuations of the token sequence $x_{:n}$; this may involve multiple sampling calls of the draft model.
2. **Evaluation**–target model P evaluates all drafted continuations in parallel.
3. **Verification**–one or more tokens are accepted, based on their probabilities under Q and P.

Speculative decoding achieves a **2-3**$\times$ speed-up in text generation (Leviathan et al., 2023; Chen et al., 2023), the speed-up increasing with the alignment of the draft model Q with the target model P. Following common computational models, we quantify speed-up as the expected number of generated tokens per evaluation of the target model P. For a fair comparison of different speculative decoding strategies, we fix the number of drafted tokens k and analyze the speed-up achieved. While 'speculative decoding' describes the overall acceleration strategy for LLMs, the specific methods implementing these steps via probabilistic acceptance mechanisms are known as *speculative sampling*.

Channel simulation is a compression problem focused on efficiently generating samples from a target probability distribution P at a decoder (Li, 2024). While only the encoder knows the target distribution P, both the encoder and decoder can access to samples from common reference distribution Q. Unlike standard source coding, the goal



(a) Autoregressive (b) *simple* ($k = 1$)

(c) *sequence* ($k = 4$) (d) *batch* ($k = 4$) (e) *tree* ($k = 6$)

Figure 1: Speculative sampling trees: a black draft tree overlaid with a green/red GSS decision tree. Black vertices and arrows represent the draft tree; each vertex is a drafted token, and paths from the gray root are potential text continuations. Green/red arrows show GSS acceptance/rejection decisions. Blue leaf vertices signify sampling from a distribution. (Note: ERSS does not follow the same decision tree.)

is not to communicate a specific sample, but rather to ensure that the decoder can generate any sample distributed according to P. Common methods rely on generating a shared codebook (list) of samples at the encoder and decoder. Based on the target distribution P, the encoder communicates to the decoder an index from this list, guaranteed to follow the target distribution P. A good introduction is provided by Theis and Yosri (2022), and Li (2024) compiles a comprehensive literature review.

Intuitively, both speculative sampling and channel simulation aim to generate samples from some target distribution P by generating samples from a draft/reference distribution Q. In channel simulation, the goal is to minimize the entropy of the chosen index. In speculative sampling, the goal is to maximize the probability of one of the drafted tokens being accepted. Although these objectives seem different, the most common speculative sampling method (Leviathan et al., 2023; Chen et al., 2023), which we refer to as greedy speculative sampling (GSS), (explained in the next section), and a channel simulation method called greedy rejection sampling (Harsha et al., 2010; Flamich and Theis, 2023) *are essentially the same procedures!* To explain this surprising connection, we develop a theory that explicitly links the entropy of the accepted token with the speed-up of speculative sampling by considering the generation of multiple tokens. Additionally, we propose a novel speculative sampling strategy, exponential race speculative sampling (ERSS), based on another channel simulation method called Poisson functional representation (Li and El Gamal, 2018). We demonstrate that its performance matches that of GSS.

Our contributions in this paper are:

- Establish a surprising connection between speculative sampling and channel simulation.
- Propose a new speculative sampling method using Poisson functional representation.
- Derive an explicit relation between the number of drafted tokens and the expected number of accepted tokens, i.e., the speed-up of response generation.

Throughout this paper, Ω denotes the finite set of tokens, where each token is represented by an integer corresponding to a textual element. Sets are enclosed in braces $\{\}$, sequences in parentheses $(\,)$. Concatenation of sequences a and b is written as $a||b$. When clear from context, we write $a||e$ as a shorthand for $a||(e)$, where (e) is a sequence with a single element e.

2 Speculative sampling

During inference, LLMs predict the distributions of tokens conditioned on the preceding sequence. We can consider an LLM as a black box that takes a sequence of tokens $x_{:n}$ as input, and outputs a set of distributions $\{P(\,\cdot \mid x_{:i})\}_{i\leq n}$. In standard auto-regressive decoding, we typically utilize only the last distribution $P(\,\cdot \mid x_{:n})$ to sample the next token. In contrast, speculative decoding leverages distributions conditioned on different subsequences to accelerate generation.

Algorithm 1 Simple GSS ($k = 1$)

```
1: Input: partial output $x_{:n}$, draft model $Q$, target model $P$
2: EVALUATE($Q, x_{:n}$)
3: $\tilde{x} \sim Q(\cdot \mid x_{:n})$
4: EVALUATE($P, x_{:n}||\tilde{x}$)
5: $U \sim$ Uniform$(0, 1)$
6: if $\frac{P(\tilde{x}|x_{:n})}{Q(\tilde{x}|x_{:n})} > U$ then
7:     $x_2^{next} \sim P(\cdot \mid x_{:n}||\tilde{x}))$
8:     Return $(\tilde{x}, x_2^{next})$
9: end if
10: $P_{residual} \leftarrow \max(P(\cdot \mid x_{:n}) - Q(\cdot \mid x_{:n}), 0)$
11: $P_{residual} \leftarrow P_{residual}/\text{SUM}(P_{residual})$
12: $x_{residual} \sim P_{residual}$
13: Return $(x_{residual})$
```

Algorithm 2 Simple ERSS ($k = 1$)

```
1: Input: partial output $x_{:n}$, draft model $Q$, target model $P$
2: EVALUATE($Q, x_{:n}$)
3: for all $i \in \Omega$ do
4:     $e_i \sim$ Exp$(1)$
5: end for
6: $\tilde{x} \leftarrow \arg\min_{i\in\Omega} \frac{e_i}{Q(i|x_{:n})}$
7: EVALUATE($P, x_{:n}||\tilde{x}$)
8: $x_1^{next} \leftarrow \arg\min_{i\in\Omega} \frac{e_i}{P(i|x_{:n})}$
9: if $\tilde{x} = x_1^{next}$ then
10:     $x_2^{next} \sim P(\cdot \mid x_{:n}||\tilde{x})$
11:     Return $(x_1^{next}, x_2^{next})$
12: end if
13: Return $(x_1^{next})$
```

The simplest case of GSS is outlined in Algorithm 1. In the **drafting** stage (lines 2-3), a single token $\tilde{x}$ is generated using the draft model Q, we dub this the *simple* drafting strategy. Subsequently, in the **evaluation** stage (line 4), the target model processes the sequence $x_{:n}||\tilde{x}$ and produces, among others, two distributions: $P(\cdot \mid x_{:n})$ and $P(\cdot \mid x_{:n}||\tilde{x})$. The **verification** stage then determines whether to accept $\tilde{x}$ based on probabilities of both $P(\cdot \mid x_{:n})$ and $Q(\cdot \mid x_{:n})$. If $\tilde{x}$ is accepted, the algorithm effectively generates two tokens per target model evaluation, as $P(\cdot \mid x_{:n}||\tilde{x})$ is already computed and can be sampled from. Otherwise, if $\tilde{x}$ is rejected, the algorithm performs a standard auto-regressive step, but samples from a *residual distribution* $P_{residual}$ derived from $P(\cdot \mid x_{:n})$ and $Q(\cdot \mid x_{:n})$. This residual distribution focuses on sampling from tokens to which the target model P assigns proportionally higher probability than the draft model Q.

A straightforward extension to the *simple* drafting strategy is to speculate a *sequence* of k tokens instead of just one. During the **drafting** stage, the draft model Q is run auto-regressively k times to generate a draft sequence of length k, $(\tilde{x}_{(1)}, \tilde{x}_{(1,1)}, \ldots, \tilde{x}_{(1)_{i=1}^k})$. The notation $\tilde{x}_{\mathbf{j}}$ describes that the token at position $\mathbf{j} = (j_1, \ldots, j_{m-1}, j_m)$ comes after token $\tilde{x}_{\mathbf{j}'}$ where $\mathbf{j}' = (j_1, \ldots, j_{m-1})$. This choice of notation is made to ensure consistency across all strategies introduced later. The **evaluation** step then processes the entire drafted sequence $x_{:n}||(\tilde{x}_{(1)}, \tilde{x}_{(1,1)}, \ldots, \tilde{x}_{(1)_{i=1}^k})$ in parallel using the target model P. The drafted tokens are **verified** and accepted sequentially, starting from the first, until a rejection occurs, or all k drafted tokens are accepted. Consequently, the *sequence* strategy can generate from 1 to $k+1$ tokens per evaluation of the target model P.

An alternative to *sequence* drafting is *batch* drafting (Sun et al., 2024b), where multiple candidate continuations are explored. Specifically, Q is evaluated once to sample multiple draft token alternatives, which are then evaluated in parallel by the target model P. While naively computationally intensive, Miao et al. (2024) showed that it can be done at the same computational cost as *sequence* drafting.[1] This yields distributions $(P(\cdot \mid x_{:n}||\tilde{x}_{(i)}))_{i\leq k}$. The verification stage then sequentially considers each drafted token $\tilde{x}_{(i)}$ until the first acceptance. This strategy generates two tokens if any drafted token is accepted, or one if all get rejected. To avoid redundancy, candidates should be sampled without replacement (Jeon et al., 2024). To recall the connection highlighted in Section 1, the acceptance criterion in greedy rejection sampling (Harsha et al., 2010; Flamich and Theis, 2023) and *batch* GSS when $k = |\Omega|$ is exactly the same.

The *tree* drafting strategy, which subsumes *simple*, *sequence*, and *batch* methods, operates on any

[1]A straightforward implementation of *batch* drafting would require evaluation of k sequences of length $(n+1)$, but this can be reduced to a single sequence of length $(n+k)$ by manipulating attention masks and token embeddings – resulting in the same cost as *sequence* drafting. Please refer to Miao et al. (2024) for details.

ordered tree topology. Each vertex represents a token drawn from Q. Each path from the root to a leaf constitutes a possible sequence drafted autoregressively. The general *tree* GSS algorithm, detailed in Appendix A and Algorithm 4, recursively applies *batch* selection at each vertex of the tree to generate the accepted sequence. All the strategies are illustrated in Figure 1.

The work most closely resembling ours is Chen et al. (2025), which investigates optimal tree topologies for GSS under specific assumptions. Our analysis is more general and encompasses their problem setting as a special case. Moreover, by establishing connections to information theory – specifically channel simulation (Li, 2024) and Tunstall coding (Tunstall, 1968) – we derive an upper bound on the expected speed-up, which is asymptotically exact as number of drafted tokens k increases. Our algorithm for generating sampling trees achieves $O(k \log k)$ complexity, outperforming the $O(k^2|\Omega|)$ complexity of Chen et al. (2025). Additionally, we consider token-by-token generation, unlike global generation (Hu and Huang, 2024), which operates on trees natively.

An alternative approach to speculative decoding leverages optimal transport (OT) principles (Sun et al., 2023; Ahn et al., 2023; Sun et al., 2024a). These methods frame the selection of accepted tokens as a linear programming problem. However, the dimensionality of this problem scales aa $|\Omega|^k$, rendering it computationally intractable even for small values of k. Consequently, approximations are employed to make OT-based methods practical. While these methods aim to maximize the immediate probability of accepting *any* single drafted token, this prioritization increases the entropy of accepted tokens. As we show in Section 4, entropy of the acceptance distribution is inversely proportional to long-term speed-up (as k grows). In contrast, through the established connection to channel simulation, we show that both GSS and our proposed ERSS implicitly aim at minimizing this acceptance entropy, thereby maximizing long-term generation speed-up.

3 Exponential Races

This section introduces the concept of exponential races. We first explain their use as a method for sampling from a distribution P. We then show their use for speculative sampling, leveraging a distribution Q to 'predict' a sample from P.

An exponential race (Maddison, 2017) is a Poisson process with time-ordered points, each corresponding to a sample from a distribution P. We are interested in the winner (first point) of this race. In the discrete case, we can simulate relevant points, i.e., potential winners, by associating each element i of the sample space Ω with an exponential random variable $e_i \sim \text{Exp}(1)$, $\mathbf{e} = \{e_i\}_{i\in\Omega}$. The winner of the race, $i^* = \arg\min_{i\in\Omega} \frac{e_i}{p_i}$, is distributed according to $i^* \sim P$. This is also known as the Gumbel-max trick (Jang et al., 2017), where arrival times are obtained via the monotonic transformation $-\log(\cdot)$. Thus, given a distribution P, exponential races allow us to sample from it using independent exponential random variables. For completeness, the proof of correctness of this result is provided in Appendix B.

Let P and Q be distributions with the same support Ω. Using the same realization of $\mathbf{e}$ for exponential races yields $i_P^* = \arg\min_{i\in\Omega} \frac{e_i}{p_i} \sim P$ and $i_Q^* = \arg\min_{i\in\Omega} \frac{e_i}{q_i} \sim Q$, both following their respective distributions. If P and Q are similar (i.e., the ratio $\frac{p_i}{q_i}$ is close to 1), then it is likely that $i_P^* = i_Q^*$. An example is illustrated in Figure 2, where Q and P are distributions generated by LLMs.

The simplest version of ERSS is presented in Algorithm 2, where only a single token is generated from the draft model in the **drafting** stage: a winner of an exponential race under $Q(\cdot \mid x_{:n})$. As in GSS, the target model P is then **evaluated** on the sequence $x_{:n}||\tilde{x}$. During **verification**, if the winner of the exponential race under $P(\cdot \mid x_{:n})$ is also $\tilde{x}$, one more token is drafted. It is straightforward to extend this approach to a *sequence* strategy, by generating an exponential race for every node in the draft sequence. For the *batch* strategy with k alternatives, the drafted tokens are selected as the first k arrivals of the exponential race under the draft model Q, as those are the most likely winners of the race under law P. Like the traditional GSS, the *batch* case can be generalized to the full *tree* by continuing different trajectories. The general procedure for this *tree* approach is presented in Appendix A as Algorithm 5.

We now highlight key distinctions between ERSS and GSS. First, during verification, the fundamental operation of ERSS is finding the minimum of a vector, which is efficiently parallelizable. This contrasts with GSS, which involves an entirely sequential chain of accept/reject steps (see Figure 1e). Second, ERSS verification can

Context: **The cat sat on the**

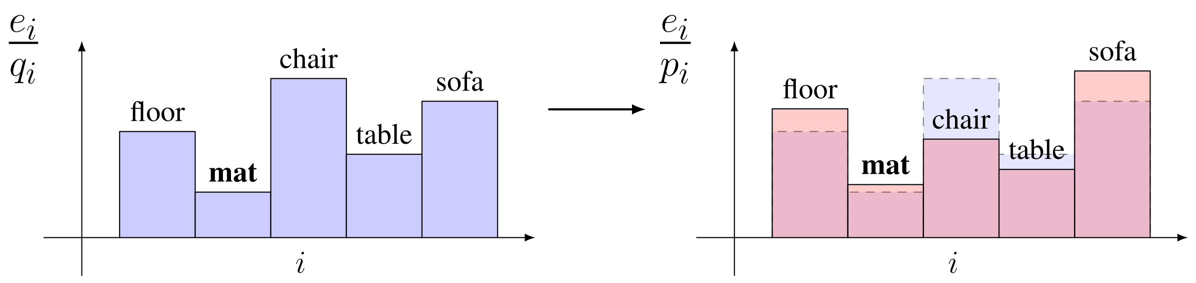


Figure 2: Illustration of exponential races for speculative sampling. Each bar represents a potential next token, with height corresponding to arrival time. The first arrival under the draft model distribution Q (left) predicts the first arrival under the target model distribution P (right).

be implemented entirely in the unnormalized logit space. GSS, however, requires the calculation of residual probability distributions and associated normalization constants at each step along its verification path. Simplified computations due to its logit-only nature make ERSS applicable to continuous data, including images, video or audio, for use in variational autoencoders (Kingma and Welling, 2014) and diffusion models (Sohl-Dickstein et al., 2015). Third, the parallel minimum-finding mechanism lends ERSS greater conceptual simplicity compared to the stateful sequential logic of GSS, potentially easing implementation and debugging. Finally, regarding theoretical guarantees, Section 4 presents analyses showing how ERSS's speed-up bounds (Equation 8) compare favorably to those derived for GSS (Equation 7), implying tighter worst-case performance bounds for ERSS.

4 Markov Chain Simulation

This section focuses on determining the optimal drafting strategy for speculative decoding. We derive this strategy by modeling the underlying distributions as Markov chains. We then establish a connection to Tunstall codes (Tunstall, 1968), which facilitates rigorous performance upper bounds for all speculative sampling methods and exact asymptotics for optimal speculative sampling.

The central task is to identify the optimal drafting strategy, represented by a drafting tree τ. In such a tree, each vertex, except the root, is associated with a drafted token. Specifically, for a vertex indexed by $\mathbf{j} = (j_1, \ldots, j_l)$, the associated token $\tilde{x}_{\mathbf{j}}$ is sampled from $Q(\,\cdot \mid x_{:n} \| (\tilde{x}_{\mathbf{j}_{:1}}, \ldots, \tilde{x}_{\mathbf{j}_{:l-1}}))$, where $\mathbf{j}_{:m} = (j_1, \ldots, j_m)$. Let $R(\mathbf{j} \mid x_{:n})$ denote the probability that vertex $\mathbf{j}$, and thus the drafted token $\tilde{x}_{\mathbf{j}}$, is accepted during verification, given context $x_{:n}$ and a chosen speculative sampling algorithm. $R(\mathbf{j} \mid x_{:n})$ is defined as the probability of acceptance, marginalized over all possible draft token sequences from Q:

$$R(\mathbf{j}\,|x_{:n}) = \mathop{\mathbf{E}}_{\tilde{x} \sim Q(\cdot|x_{:n})} \mathbf{Pr}\left\{\tilde{x}_{\mathbf{j}} \text{ accepted} \mid x_{:n}, \tilde{x}\right\} \tag{1}$$

For the rest of the section, we assume the target and draft model distributions, P and Q, be m-th order Markov sources. Then, the acceptance probability $R(\mathbf{j} \mid x_{:n})$ is an m-th order Markov chain. The expected number of accepted tokens is the sum of acceptance probabilities for each vertex in the drafting tree. Therefore, the optimal drafting tree τ^* with $k+1$ vertices (for speculating k tokens) is the solution to:

$$\tau^* = \mathop{\arg\max}_{\text{tree } \tau, |\tau| = k+1} \sum_{\mathbf{j} \in \tau} R(\mathbf{j} \mid x_{:n}). \tag{2}$$

Crucially, for any vertex $\mathbf{j}$ and its descendant $\mathbf{j}'$ in a drafting tree, the acceptance probability is decreasing: $R(\mathbf{j} \mid x_{:n}) \geq R(\mathbf{j}' \mid x_{:n})$. This is because $\mathbf{j}'$ can only be accepted if its ancestor $\mathbf{j}$ is accepted. Consequently, the top k vertices with the highest acceptance probabilities form a valid tree. Any node $\mathbf{j}$ in this top-k list will have all its ancestors $\mathbf{j}_{:m}$ (where $m < |\mathbf{j}|$) also present in the list due to their higher acceptance probabilities. This observation leads to a greedy algorithm (Algorithm 3) for constructing an optimal tree with $k+1$ vertices. Algorithm 3 iteratively builds the tree by greedily adding the next most likely token to be accepted. This is achieved efficiently using a priority queue to maintain candidate vertices, ordered by their acceptance probabilities. The algorithm's computational complexity is $O(k \log k)$ because the loop iterates k times, and each iteration involves priority queue operations (push and pop) with a maximum queue size of $2k$, each taking $O(\log k)$ time.

Algorithm 3 Optimal tree construction

```
1: Input: # drafted tokens $k$, partial output $x_{:n}$
2: $tree \leftarrow \{()\}$    ▷ initialize tree with root only
3: $C \leftarrow$ PRIORITYQUEUE( )
4: ADD($C, (R((0) \mid x_{:n}), (0))$)
5: for all $i \in (1, \dots, k)$ do
6:     $\_, \mathbf{j} \leftarrow$ POPMAX($C$)
7:     $tree \leftarrow tree \cup \{\mathbf{j}\}$
8:     $\mathbf{j}_{child} \leftarrow \mathbf{j} || (0)$
9:     $C$.ADD($(R(\mathbf{j}_{child} \mid x_{:n}), \mathbf{j}_{child})$)
10:    if $\mathbf{j}_{|\mathbf{j}|} < |\Omega|$ then
11:        $\mathbf{j}_{sibling} \leftarrow \mathbf{j}_{:(|\mathbf{j}|-1)} || (\mathbf{j}_{|\mathbf{j}|} + 1)$
12:        $C$.ADD($(R(\mathbf{j}_{sibling} \mid x_{:n}), \mathbf{j}_{sibling})$)
13:    end if
14: end for
15: Return tree
```

Our analysis relies on the acceptance probability function R, which is typically unknown in practice. If we approximate R using an empirical acceptance distribution – effectively treating it as a 0-th order Markov source – the resulting algorithm becomes equivalent to that proposed by Chen et al. (2025). We employ this empirical approximation in our experiments. However, as Section 5 will demonstrate, this 0-th order approximation proves to be inaccurate, leading to an underestimation of the speed-up we can achieve with speculative decoding. Developing more refined approximations of R presents a promising avenue for future investigation.

In another connection to information theory, the process of generating optimal drafting trees (Algorithm 3) closely resembles the construction of Tunstall codes (Tunstall, 1968). Due to their similarity, quantities in speculative sampling, such as the number of drafted tokens, speed-up, and entropy of the acceptance distribution, have direct counterparts in Tunstall codes: the number of expanded nodes in the Tunstall tree, the expected length of the consumed source symbols, and the source entropy, respectively. This connection allows us to express the expected speed-up of speculative sampling (both GSS and ERSS) in terms of these quantities.

Tunstall code is a type of variable-to-fixed length source code used in the compression of discrete sources. In source coding, the goal is to represent sequences of symbols from a source alphabet (like tokens in our case) using the fewest number of bits (in the case of a binary alphabet). Variable-to-fixed length codes, such as Tunstall codes, achieve this by mapping variable-length sequences of source symbols to fixed-length codewords from the code alphabet. Specifically, a Tunstall code takes a variable-length prefix of the source symbol sequence and encodes it into a fixed-length output sequence. This encoding step is repeated until the entire source sequence is processed. Tunstall codes are known to be optimal in the sense that for sufficiently long codewords, the average number of output alphabet symbols per source symbol approaches the entropy of the source – this also holds true for sources with memory (Savari and Gallager, 1997).

Let the entropy of the acceptance probability distribution be defined as:

$$\mathbf{H}[R] = \mathop{\mathbf{E}}_{x_{:n} \sim P} \mathbf{H}[R(\cdot \mid x_{:n})] \,. \tag{3}$$

Definition 4.1. *Let $\mathcal{S} : \mathcal{P} \times \mathcal{Q} \times \Omega^* \times \mathcal{T} \to \Omega^*$ be a speculative sampling procedure that takes a target model P, draft model Q, context $x_{:n}$, and drafting strategy τ, and outputs a sequence of generated tokens. The expected number of tokens generated by $\mathcal{S}$ under strategy τ is defined as*

$$G_\tau = \mathop{\mathbf{E}}_{x_{:n} \sim P} \left[|\mathcal{S}(P, Q, x_{:n}, \tau)| \right] \tag{4}$$

where the expectation is taken over contexts $x_{:n}$ distributed according to the target model P.

Theorem 4.2. *(Upper bound Tunstall) For speculative sampling employing the optimal drafting strategy τ^*, the expected number of generated tokens is bounded as follows:*

$$G_{\tau^*} \leq \frac{\log |\Omega| + \log(k+1)}{\mathbf{H}[R]} \,. \tag{5}$$

The proof of Theorem 4.2 is provided in Appendix D. Since this bound applies to the optimal strategy τ^*, it holds for any strategy τ. As the theorem suggests, for a large number of drafted tokens k, the generation speed-up (G_{τ^*}) is fundamentally governed by the entropy of the acceptance distribution, $\mathbf{H}[R]$. Channel simulation algorithms, such as greedy rejection sampling and Poisson functional representation, are explicitly designed to minimize this entropy. Consequently, these methods, which underpin both GSS and ERSS, inherently aim to maximize generation speed-up by reducing $\mathbf{H}[R]$. From a channel simulation perspective, the KL divergence $\mathbf{D}_{\text{KL}}[P \| Q]$ between the target distribution P and the draft distribution Q provides a

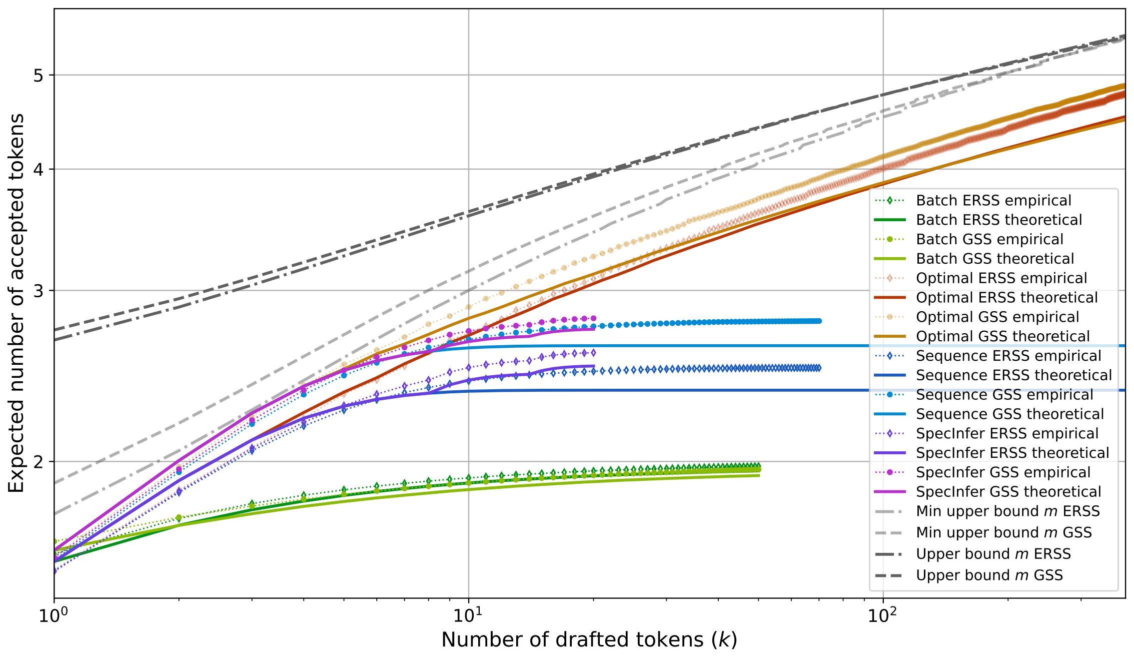


Figure 3: Expected number of accepted tokens as a function of the number of drafted tokens for *sequence*, *batch*, τ^* *tree* (optimal), and SpecInfer *tree* drafting strategies, for GSS and ERSS. Results shown for draft model Q Llama-3.2-1B, and target model P Llama-3.1-70B-Instruct. For optimal drafting strategy both GSS and ERSS converge in performance as k increases.

lower bound on this crucial entropy (Li, 2024):

$$\mathbf{D}_{\text{KL}}[P\|Q] \leq \mathbf{H}[R]. \tag{6}$$

Furthermore, channel simulation theory provides upper bounds on $\mathbf{H}[R]$ in terms of $\mathbf{D}_{\text{KL}}[P\|Q]$ for greedy rejection coding (Harsha et al., 2010) (relevant to GSS):

$$\mathbf{H}[R] \leq \mathbf{D}_{\text{KL}}[P\|Q] + (1+\epsilon)\log(\mathbf{D}_{\text{KL}}[P\|Q]+1) + O(1), \tag{7}$$

for any $\epsilon > 0$, and for Poisson functional representation (Li and El Gamal, 2018) (relevant to ERSS):

$$\mathbf{H}[R] \leq \mathbf{D}_{\text{KL}}[P\|Q] + \log(\mathbf{D}_{\text{KL}}[P\|Q]+1) + 4. \tag{8}$$

These bounds show the connection of speed-up in speculative sampling to KL divergence $\mathbf{D}_{\text{KL}}[P\|Q]$ between the models.

While asymptotically accurate for large k, the bound from Theorem 4.2 is dominated by the full token alphabet size $|\Omega|$ when the number of drafted tokens k is small. For such values of k, we observe that the optimal drafting tree typically explores only a limited number of drafting positions. Indeed, multiple acceptance distributions can yield identical optimal drafting trees and the same acceptance probabilities for all nodes within those trees. Thus, evaluating the bound from Theorem 4.2 for any such distribution provides a valid upper bound for all of them. To simplify the notation and exposition, we focus on the 0-th order acceptance distribution R, noting that our findings can be generalized to higher-order Markov sources.

For a given number of drafted tokens k, let d be the maximum index explored in the optimal drafting tree τ_k^* with $k+1$ vertices, i.e., $d = \max_{\mathbf{j}\in\tau_k^*} \max_i\{i \mid i \in \mathbf{j}\}$. We define an equivalent acceptance distribution $\hat{R}$ such that for indices $i \leq d$, $\hat{R}(i) = R(i)$, and for $i > d$, $\hat{R}(i) \leq \min_{i'\leq d} R(i')$. In short, $\hat{R}$ matches R on tree nodes for the first d indices and is upper-bounded by $\min_{i'\leq d} R(i')$ thereafter.

Our objective is to minimize the upper bound on the expected number of accepted tokens. Upon inspection of Theorem 4.2, we see that a tighter bound is achieved by maximizing the entropy of $\hat{R}$ and minimizing the alphabet size. Minimizing the bound requires strategically distributing the remaining probability mass, $p_{res} = 1 - \sum_{i=1}^{d} R(i)$, associated with indices beyond d. For any choice to incorporate m additional indices, entropy maximization is achieved by distributing the residual

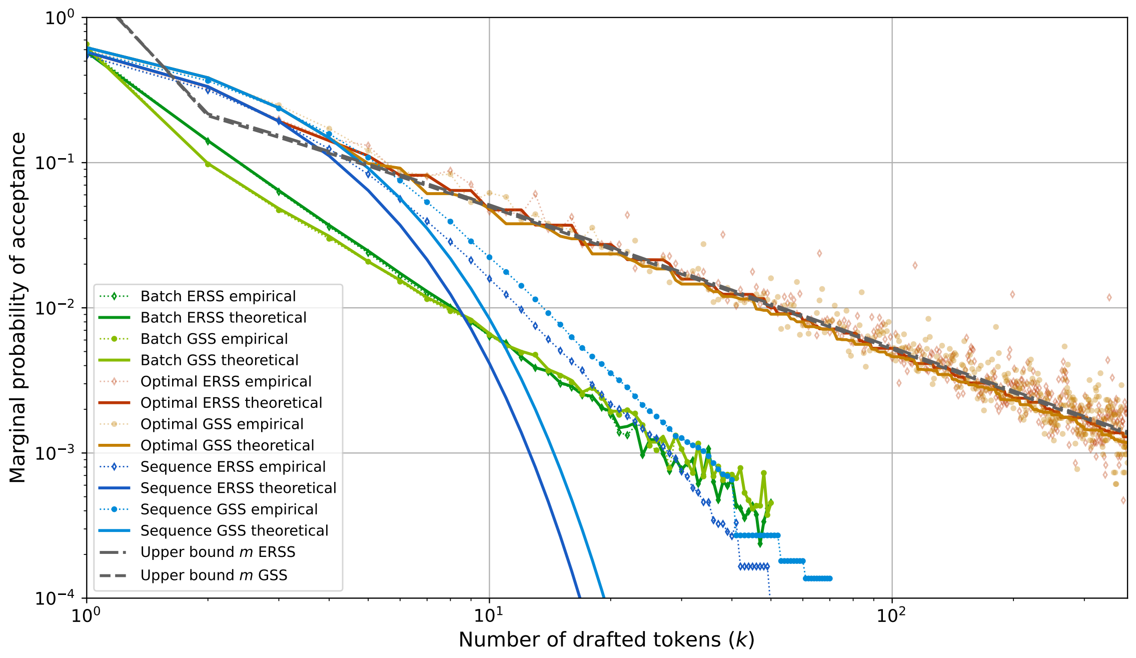


Figure 4: Marginal probability of acceptance as a function of the number of drafted tokens for *sequence* , *batch* , τ^* *tree* (optimal) drafting strategies, for GSS and ERSS. Results shown for draft model Q Llama-3.2-1B, and target model P Llama-3.1-70B-Instruct.

mass equally across them.

Lemma 4.3. *(Upper bound m) For a fixed number of drafted tokens k and initial index range d, consider a 0-th order acceptance probability distribution R. Let $p_{res} = 1 - \sum_{i=1}^{d} R(i)$ be the residual probability mass. Define R^{res} such that $R^{res}(i) = R(i)$ for $i \leq d$, and $R^{res}(d+1) = p_{res}$. Then, for any integer $m \geq \frac{p_{res}}{\min_{i \leq d} R(i)}$, the expected number of generated tokens is upper-bounded by:*

$$G_{\tau^*} \leq \frac{\log(d+m) + \log(k+1)}{\mathbf{H}\left[R^{res}\right] + p_{res} \log m}.$$

Note: R^{res} defined in above lemma is not $\hat{R}$.

5 Experiments

To validate our theoretical analysis, we conducted numerical experiments comparing GSS and ERSS across different drafting strategies. We performed open-ended text generation up to 200 tokens, accumulating $100k$ generated tokens per strategy. The experiments were performed on 8 Nvidia RTX A6000 GPUs, with experiments running for 140 hours of wall-clock time. We estimated the acceptance probability function R using empirical acceptance probabilities for different indices (from *sequence* and *batch*), approximating it as a 0-th order Markov chain. Based on this estimated R, we computed the optimal drafting tree τ^* for both GSS and ERSS using Algorithm 3.

Figure 3 illustrates the expected number of accepted tokens as a function of drafted tokens for the target model Llama-3.1 70B and the draft model Llama-3.2 1B (Grattafiori et al., 2024), used under the Meta Llama 3.1 and 3.2 Licenses. Figure 4 shows the marginal change in acceptance probability with each additional drafted token. We compare four drafting strategies: *batch*, *sequence*, optimal *tree* (τ^*), and the SpecInfer *tree* (Miao et al., 2024), which drafts 3-sequences with the first two drafted tokens being common. For each strategy and speculative sampling method (GSS, ERSS), we present both theoretical and empirical performance. The theoretical plots show the expected number of generated tokens based on the estimated R, while the empirical plots display the actual value observed in our experiments.

Figure 3 reveals that for *batch* drafting with $k = 1$, GSS outperforms ERSS–it has a higher chance of accepting the first proposed sample. This can be explained by the 'greediness' of GSS, which maximizes the acceptance probability of each sample sequentially, a strategy that is suboptimal when considering multiple samples. This also explains the performance gap between strategies for small values of drafted samples k. Further discussion regarding the probability of acceptance of the first sample for GSS and ERSS is presented in Appendix C. However, with $k = 2$, both methods exhibit comparable performance. For $k \geq 3$, ERSS achieves a higher expected number of accepted tokens, and hence speed-up. The acceptance probabilities for each index $((0), (1), (2), \dots)$ are shown in Figure 4. For *sequence* drafting, GSS demonstrates superior performance compared to ERSS, since this strategy comprises a series of *batch* drafting steps with $k = 1$, considering only a single possible continuation at each step. Intriguingly, our empirical results also challenge the 0-th order acceptance assumption (Chen et al., 2025). The empirical performance plateaus at a higher level than predicted by the measured R, indicating higher-order Markov dependencies. This can be explained by regions of language where the target model P and draft model Q exhibit greater alignment, leading to extended sequences of accepted tokens. The SpecInfer *tree* achieves slightly improved performance over *sequence* drafting but still plateaus.

As predicted, the optimal τ^* *tree* strategy yields the best performance for both GSS and ERSS, exhibiting a logarithmic relationship between drafted tokens k and expected generated tokens (Figure 3). At low k, the optimal *tree* τ^* and *sequence* strategies show similar performance, with GSS slightly outperforming ERSS. However, as k increases, the performance gap diminishes, and they converge.

Figure 3 presents the minimal upper bound derived from Lemma 4.3 by optimizing the parameter m for each number of drafted tokens k. We also depict the upper bound from Lemma 4.3, calculated with a fixed m for $k = 384$ to show the behavior of the bound. While the marginal changes in the minimal upper bound are omitted for visual clarity– due to their step-like transitions–, we observe that the marginal changes of the fixed-m upper bound closely follow the trends of the optimal τ^* *tree* .

6 Future Directions

Our channel simulation perspective on speculative sampling suggests several avenues for future research. First, beyond exact sampling which guarantees output identical to the target model P, one could explore approximate speculative sampling. Drawing from work on approximate channel simulation (Havasi et al., 2019; Theis and Yosri, 2022), which relates approximation quality to communication cost (analogous to draft sample count), this could illuminate the trade-offs between speed-up and distribution shifts, potentially leading to new algorithms. Second, further application of the Poisson Matching Lemma (Li and Anantharam, 2021), utilized for our Lemma C.1 upper bound, could yield lower bounds on acceptance probability or index entropy for ERSS, complementing our analysis with worst-case performance guarantees for speed-up. Third, the channel simulation framework links speed-up to the accepted index entropy. This motivates comparing different draft model Q optimization strategies: directly minimizing this index entropy versus minimizing $D_{KL}(P||Q)$, or using reinforcement learning objectives targeting empirical speed-up. Finally, while prior work has connected speculative decoding to OT, our channel simulation framework suggests investigating minimal entropy couplings as a more direct theoretical foundation for maximizing long-term speed-up, distinct from standard OT objectives.

7 Conclusion

This work establishes a connection between speculative decoding – a technique for accelerating autoregressive LLM generation – and channel simulation. This connection enabled us to propose ERSS, a novel speculative sampling method. By linking the optimal drafting strategy to Tunstall codes, we derived a theoretical upper bound and the asymptotic relationship between the number of speculated tokens and the expected speed-up, for GSS and ERSS. These findings offer a deeper understanding of, and potential improvements to, the efficiency of speculative decoding.

8 Limitations

Speculative decoding's speed gains are most significant when drafting only a few tokens, k. Therefore, for small values k, GSS is often the most practical choice. While token-by-token generation was the focus of this work, joint sequence generation in speculative sampling or channel simulation could lead to further improvements, but its computational practicality is uncertain. Furthermore, our current stateless approximation of the acceptance probability function R oversimplifies the contextual nature of language. Developing more context-aware approximations could yield improvements in future work.

References

Kwangjun Ahn, Ahmad Beirami, Ziteng Sun, and Ananda Theertha Suresh. 2023. Spectr++: Improved transport plans for speculative decoding of large language models. In *NeurIPS 2023 Workshop Optimal Transport and Machine Learning*.

Charlie Chen, Sebastian Borgeaud, Geoffrey Irving, Jean-Baptiste Lespiau, Laurent Sifre, and John Jumper. 2023. Accelerating Large Language Model Decoding with Speculative Sampling. *arXiv preprint*. ArXiv:2302.01318 [cs].

Zhuoming Chen, Avner May, Ruslan Svirschevski, Yu-Hsun Huang, Max Ryabinin, Zhihao Jia, and Beidi Chen. 2025. Sequoia: Scalable and Robust Speculative Decoding. *Advances in Neural Information Processing Systems*, 37:129531–129563.

Imre Csiszár and János Körner. 2011. *Information Theory: Coding Theorems for Discrete Memoryless Systems*, 2 edition. Cambridge University Press.

Gergely Flamich and Lucas Theis. 2023. Adaptive Greedy Rejection Sampling. In *2023 IEEE International Symposium on Information Theory (ISIT)*, pages 454–459. ISSN: 2157-8117.

Aaron Grattafiori et al. 2024. The Llama 3 Herd of Models. *arXiv preprint*. ArXiv:2407.21783 [cs].

Prahladh Harsha, Rahul Jain, David McAllester, and Jaikumar Radhakrishnan. 2010. The Communication Complexity of Correlation. *IEEE Transactions on Information Theory*, 56(1):438–449.

Marton Havasi, Robert Peharz, and José Miguel Hernández-Lobato. 2019. Minimal Random Code Learning: Getting Bits Back from Compressed Model Parameters. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*.

Zhengmian Hu and Heng Huang. 2024. Accelerated speculative sampling based on tree monte carlo. In *Forty-first International Conference on Machine Learning*.

Eric Jang, Shixiang Gu, and Ben Poole. 2017. Categorical reparameterization with gumbel-softmax. In *International Conference on Learning Representations*.

Wonseok Jeon, Mukul Gagrani, Raghavv Goel, Junyoung Park, Mingu Lee, and Christopher Lott. 2024. Recursive speculative decoding: Accelerating LLM inference via sampling without replacement. In *ICLR 2024 Workshop on Large Language Model (LLM) Agents*.

Diederik P. Kingma and Max Welling. 2014. Auto-Encoding Variational Bayes. In *2nd International Conference on Learning Representations, ICLR 2014, Banff, AB, Canada, April 14-16, 2014, Conference Track Proceedings*.

Yaniv Leviathan, Matan Kalman, and Yossi Matias. 2023. Fast Inference from Transformers via Speculative Decoding. In *Proceedings of the 40th International Conference on Machine Learning*, pages 19274–19286. PMLR. ISSN: 2640-3498.

Cheuk Ting Li. 2024. Channel Simulation: Theory and Applications to Lossy Compression and Differential Privacy. *Foundations and Trends® in Communications and Information Theory*, 21(6):847–1106. Publisher: Now Publishers, Inc.

Cheuk Ting Li and Venkat Anantharam. 2021. A Unified Framework for One-Shot Achievability via the Poisson Matching Lemma. *IEEE Transactions on Information Theory*, 67(5):2624–2651.

Cheuk Ting Li and Abbas El Gamal. 2018. Strong Functional Representation Lemma and Applications to Coding Theorems. *IEEE Transactions on Information Theory*, 64(11):6967–6978.

Chris J. Maddison. 2017. A Poisson Process Model for Monte Carlo. In *Perturbations, Optimization, and Statistics*, pages 193–231. MIT Press. Conference Name: Perturbations, Optimization, and Statistics.

Xupeng Miao, Gabriele Oliaro, Zhihao Zhang, Xinhao Cheng, Zeyu Wang, Zhengxin Zhang, Rae Ying Yee Wong, Alan Zhu, Lijie Yang, Xiaoxiang Shi, Chunan Shi, Zhuoming Chen, Daiyaan Arfeen, Reyna Abhyankar, and Zhihao Jia. 2024. SpecInfer: Accelerating Large Language Model Serving with Tree-based Speculative Inference and Verification. In *Proceedings of the 29th ACM International Conference on Architectural Support for Programming Languages and Operating Systems, Volume 3*, volume 3 of *ASPLOS '24*, pages 932–949, New York, NY, USA. Association for Computing Machinery.

S.A. Savari and R.G. Gallager. 1997. Generalized Tunstall codes for sources with memory. *IEEE Transactions on Information Theory*, 43(2):658–668.

Jascha Sohl-Dickstein, Eric Weiss, Niru Maheswaranathan, and Surya Ganguli. 2015. Deep Unsupervised Learning using Nonequilibrium Thermodynamics. In *Proceedings of the 32nd International Conference on Machine Learning*, pages 2256–2265. PMLR.

Ryan Sun, Tianyi Zhou, Xun Chen, and Lichao Sun. 2024a. SpecHub: Provable Acceleration to Multi-Draft Speculative Decoding. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 20620–20641, Miami, Florida, USA. Association for Computational Linguistics.

Ziteng Sun, Uri Mendlovic, Yaniv Leviathan, Asaf Aharoni, Ahmad Beirami, Jae Hun Ro, and Ananda Theertha Suresh. 2024b. Block verification accelerates speculative decoding. In *Workshop on Efficient Systems for Foundation Models II @ ICML2024*.

Ziteng Sun, Ananda Theertha Suresh, Jae Hun Ro, Ahmad Beirami, Himanshu Jain, and Felix Yu. 2023. SpecTr: Fast Speculative Decoding via Optimal Transport. *Advances in Neural Information Processing Systems*, 36:30222–30242.

Lucas Theis and Noureldin Yosri. 2022. Algorithms for the Communication of Samples. In *Proceedings of the 39th International Conference on Machine Learning*, pages 21308–21328. PMLR.

Brian Tunstall. 1968. *Synthesis of Noiseless Compression Codes*. Ph.d. thesis, Georgia Institute of Technology.

A Algorithms

Algorithm 4 Greedy speculative sampling

```
 1: Input: partial output $x_{:n}$, draft model $Q$, target model $P$, # draft tokens $k$, draft strategy $\tau$
 2: for all $\mathbf{j} \in \tau$ do                                  ▷ Assumes lexicographical ordering of indexes $\mathbf{j}$
 3:     if $\mathbf{j}_{|\mathbf{j}|} = 1$ then                              ▷ If $\mathbf{j}$ is the first child evaluate $Q$
 4:         $\text{EVALUATE}(Q, x_{:n}||(\tilde{x}_{\mathbf{j}_{:i}})_{i=1}^{|\mathbf{j}|-1})$
 5:     end if
 6:     $Q_{draft} \leftarrow Q\left(\cdot \mid x_{:n}||(\tilde{x}_{\mathbf{j}_{:i}})_{i=1}^{|\mathbf{j}|-1}\right)$
 7:     $\mathbf{j}_{anc} \leftarrow \mathbf{j}_{:|\mathbf{j}|-1}$
 8:     for all $i \in \mathbb{N}^+, i < \mathbf{j}_{|\mathbf{j}|}$ do                ▷ Ensure sampling without replacement
 9:         $Q_{draft}(\tilde{x}_{\mathbf{j}_{anc}||(i)}) \leftarrow 0$
10:     end for
11:     $Q_{draft} \leftarrow Q_{draft}/\text{SUM}(Q_{draft})$
12:     $\tilde{x}_{\mathbf{j}} \sim Q_{draft}$
13: end for
14: $\text{EVALUATE}(P, x_{:n}, \{\tilde{x}_{\mathbf{j}}\}_{\mathbf{j}\in\tau})$                  ▷ Evaluate all draft tokens in parallel
15: $acc \leftarrow$ True
16: $y \leftarrow (\ )$
17: $\mathbf{j} \leftarrow (\ )$
18: while $acc$ do
19:     $acc \leftarrow$ False
20:     $P_{target} \leftarrow P(\cdot \mid x_{:n}||y)$
21:     $Q_{draft} \leftarrow Q(\cdot \mid x_{:n}||y)$
22:     $i \leftarrow 1$                                            ▷ Current considered child
23:     while $\mathbf{j}||(i) \in \tau$ do
24:         if $\frac{P_{target}(\tilde{x}_{\mathbf{j}})}{Q_{draft}(\tilde{x}_{\mathbf{j}})} >$ Uniform$(0, 1)$ then             ▷ Accept token and continue
25:             $y \leftarrow y \cup \{\tilde{x}_{\mathbf{j}}\}$
26:             $acc \leftarrow$ True
27:             break
28:         end if
29:         $P_{target} \leftarrow \max(P_{target} - Q_{draft}, 0)$
30:         $P_{target} \leftarrow P_{target}/\text{SUM}(P_{target})$          ▷ On rejection calculate residual distribution
31:         $Q_{draft}(token) \leftarrow 0$
32:         $Q_{draft} \leftarrow Q_{draft}/\text{SUM}(Q_{draft})$
33: e                                  ▷ Update target distribution for sampling w.o. replacement
34:         $i \leftarrow i + 1$
35:     end while
36: end while
37: $token \leftarrow \text{SAMPLE}(P_{target})$                        ▷ Accept token from residual distribution
38: $y \leftarrow y \cup \{token\}$
39: return $y$
```

The draft token selection step–sampling without replacement–in the general GSS Algorithm 4 can be implemented using exponential races just as in Algorithm 5, or equivalently the Gumbel-max trick. Furthermore, the verification step in ERSS Algorithm 5 is a significanly simpler compared to GSS.

Algorithm 5 Exponential Race Speculative Sampling

```
1: Input: partial output $x_{:n}$, draft model $Q$, target model $P$, # draft tokens $k$, draft strategy $\tau$
2: for all $\mathbf{j} \in \tau$ do                    ▷ Assumes lexicographical ordering of indexes $\mathbf{j}$
3:     $\mathbf{j}_{anc} = \mathbf{j}_{:|\mathbf{j}|-1}$
4:     if $\mathbf{j}_{|\mathbf{j}|} = 1$ then          ▷ If $\mathbf{j}$ is the first child evaluate $Q$ and generate the race
5:         for all $i \in \Omega$ do
6:             $e_{\mathbf{j}_{anc}||(i)} \leftarrow \mathrm{Exp}(1)$
7:         end for
8:         EVALUATE$(Q, x_{:n}||(\tilde{x}_{\mathbf{j}_{:i}})_{i=1}^{|\mathbf{j}|-1})$
9:     end if
10:    $\tilde{x}_{\mathbf{j}} \leftarrow \mathbf{j}_{|\mathbf{j}|}\text{-th} \arg\min_{i\in\Omega} \frac{e_{\mathbf{j}_{anc}||(i)}}{Q\left(i|x_{:n}||(\tilde{x}_{\mathbf{j}_{:i}})_{i=1}^{|\mathbf{j}|-1}\right)}$          ▷ Find $\mathbf{j}_{|\mathbf{j}|}$-th race arrival under $Q$
11: end for
12: EVALUATE$(P, x_{:n}, \{\tilde{x}_{\mathbf{j}}\}_{\mathbf{j}\in\tau})$          ▷ Evaluate all draft tokens in parallel
13: $y \leftarrow (\,)$
14: $\mathbf{j} \leftarrow (\,)$
15: while $true$ do
16:     $x^{next} \leftarrow \arg\min_{i\in\Omega} \frac{e_{\mathbf{j}_{anc}||(i)}}{Q\left(i|x_{:n}||(\tilde{x}_{\mathbf{j}_{:i}})_{i=1}^{|\mathbf{j}|-1}\right)}$          ▷ Winner of race under $P$
17:     $y \leftarrow y||(x^{next})$
18:     if $x^{next} \notin \{\tilde{x}_{\mathbf{j}||(i)} \mid i \in \Omega, \mathbf{j}||(i) \in \tau\}$ then
19:         break
20:     end if
21: end while
22: return $y$
```

B Proof of Correctness for Exponential Races

This appendix provides a proof for the statement in Section 3 that the winner $i^* = \arg\min_{i\in\Omega} \frac{e_i}{p_i}$ of an exponential race, where $e_i \sim \mathrm{Exp}(1)$ are independent and p_i are probabilities from a distribution P over Ω, follows the distribution P. That is, we aim to show that $\Pr(i^* = y) = p_y$ for any $y \in \Omega$.

The probability density function (PDF) of each e_i is $f(t) = e^{-t}$ for $t \geq 0$. The cumulative density function is $\Pr(e_i < t) = 1 - e^{-t}$. We compute the probability that a specific element $y \in \Omega$ is the winner of the race. The element y wins if its associated 'arrival time' e_y/p_y is smaller than all other arrival times e_x/p_x for $x \neq y$.

$$
\begin{aligned}
\Pr(i^* = y) &= \Pr\left(\frac{e_y}{p_y} < \frac{e_x}{p_x}, \forall x \in \Omega \setminus \{y\}\right) \\
&= \mathbb{E}_{e_y}\left[\Pr\left(e_x > \frac{p_x}{p_y} e_y, \forall x \in \Omega \setminus \{y\} \mid e_y\right)\right] \\
&= \mathbb{E}_{e_y}\left[\prod_{x\in\Omega\setminus\{y\}} \Pr\left(e_x > \frac{p_x}{p_y} e_y \mid e_y\right)\right] \\
&= \mathbb{E}_{e_y}\left[\prod_{x\in\Omega\setminus\{y\}} \exp\left(-\frac{p_x}{p_y} e_y\right)\right] \\
&= \mathbb{E}_{e_y}\left[\exp\left(-e_y \frac{\sum_{x\in\Omega\setminus\{y\}} p_x}{p_y}\right)\right] \\
&= \mathbb{E}_{e_y}\left[\exp\left(-e_y \frac{1-p_y}{p_y}\right)\right] \\
&= \int_0^\infty \exp\left(-t\frac{1-p_y}{p_y}\right) e^{-t} dt \\
&= \int_0^\infty \exp\left(-\frac{t}{p_y}\right) dt \\
&= p_y
\end{aligned}
$$

This confirms that the probability of selecting element y is exactly p_y, and thus the winner i^* is distributed according to P, i.e., $i^* \sim P$.

C First token acceptance rate

Standard GSS prioritizes maximizing the probability of accepting the initially proposed draft token. Indeed, for the *simple* (and thus, *sequence*) drafting strategy, greedy sampling achieves optimality within token-based speculation (Sun et al., 2023), with the acceptance probability given by $1 - D_{TV}[P(\cdot \mid x_{:n}), Q(\cdot \mid x_{:n})]$, where $D_{TV}[P, Q] = \frac{1}{2}\sum_{i\in\Omega} |P(i) - Q(i)|$ is the total variation distance (Csiszár and Körner, 2011). However, this method is not optimal when considering multiple possible tokens. We can establish a corresponding bound for the acceptance probability of the first drafted token in ERSS using the Poisson matching lemma (Li and Anantharam, 2021).

Lemma C.1. *For ERSS with target distribution $P = P(\cdot \mid x_{:n})$ and draft distribution $Q = Q(\cdot \mid x_{:n})$, the probability of accepting the first drafted token, $P^{(1)}_{accept}$, satisfies:*

$$
1 - D_{TV}[P, Q] \geq P^{(1)}_{accept} \geq D_{HM}[P, Q], \tag{9}
$$

where D_{HM} denotes the harmonic mean distance, defined as:

$$
D_{HM}[P, Q] \overset{def}{=} \sum_{i\in\Omega} \frac{P(i)Q(i)}{P(i) + Q(i)}. \tag{10}
$$

Despite a lower first-token acceptance probability, exponential races are not inferior to GSS. As shown in section 4, the overall performance for both methods hinges on the entropy of the acceptance distribution. From a channel simulation perspective, this entropy is linked to the Kullback-Leibler (KL) divergence between P and Q.

To show the lemma, we begin by recalling the Poisson matching lemma (Li and Anantharam, 2021), adapted to our notation for discrete alphabets:

Lemma C.2. *(Poisson matching lemma) Let P and Q be two probability distributions over the alphabet Ω. Let $E_i \sim Exp(1)$ for each symbol $i \in \Omega$ be independent exponential random variables. Define $I_P^* = \arg\min_{i\in\Omega} \frac{E_i}{p_i}$ and $I_Q^* = \arg\min_{i\in\Omega} \frac{E_i}{q_i}$. The probability that I_P^* is different than I_Q^*, given I_Q^*, is bounded by:*

$$\mathbf{Pr}\left\{I_P^* \neq I_Q^* \mid I_Q^*\right\} \leq 1 - \left(1 + \frac{q_{I_Q^*}}{p_{I_Q^*}}\right)^{-1}. \tag{11}$$

In essence, this lemma bounds the probability that two races, driven by the same underlying exponential random variables but with different distributions P and Q, will have different winners.

Proof. (Lemma C.1) To obtain the average probability of differing first arrivals, we marginalize the Poisson matching lemma over all possible values of I_Q^*. Let $P_{accept}^{(1)}$ be the probability that the first drafted token in ERSS is accepted. This occurs when the winner of the race under P is the same as the winner under Q, i.e., $I_P^* = I_Q^*$. Thus:

$$\begin{aligned}
P_{accept}^{(1)} &= \mathbf{Pr}\left\{I_P^* = I_Q^*\right\} && (12)\\
&= 1 - \mathbf{Pr}\left\{I_P^* \neq I_Q^*\right\} && (13)\\
&= 1 - \mathop{\mathbf{E}}_{I_Q^*}\left[\mathbf{Pr}\left\{I_P^* \neq I_Q^* | I_Q^*\right\}\right] && (14)\\
&\geq 1 - \mathop{\mathbf{E}}_{I_Q^*}\left[1 - \left(1 + \frac{q_{I_Q^*}}{p_{I_Q^*}}\right)^{-1}\right] \quad \text{(by Lemma C.2)} && (15)\\
&= \mathop{\mathbf{E}}_{I_Q^*}\left[\left(1 + \frac{q_{I_Q^*}}{p_{I_Q^*}}\right)^{-1}\right] && (16)\\
&= \sum_{i\in\Omega} q_i \frac{1}{1 + \frac{q_i}{p_i}} \quad (\text{since } I_Q^* \sim Q) && (17)\\
&= \sum_{i\in\Omega} \frac{p_i q_i}{p_i + q_i} && (18)\\
&= D_{HM}[P, Q], && (19)
\end{aligned}$$

where the last step follows from the definition of the harmonic mean distance $D_{HM}[P, Q]$.

Furthermore, it is known that $P_{accept}^{(1)}$ is upper-bounded by $1 - D_{TV}[P, Q]$ (Sun et al., 2023), and in general those bounds do not coincide as:

$$\begin{aligned}
1 - D_{TV}[P, Q] &= 1 - \frac{1}{2}\sum_{i\in\Omega} |p_i - q_i| && (20)\\
&= \frac{1}{2}\sum_{i\in\Omega} p_i + \frac{1}{2}\sum_{i\in\Omega} q_i - \frac{1}{2}\sum_{i\in\Omega} |p_i - q_i| && (21)\\
&= \frac{1}{2}\sum_{i\in\Omega} \frac{(p_i + q_i)^2 - |p_i - q_i|(p_i + q_i)}{p_i + q_i} && (22)\\
& && (23)
\end{aligned}$$

$$= \frac{1}{2} \sum_{i \in \Omega} \frac{p_i^2 + 2p_i q_i + q_i^2 - |p_i^2 - q_i^2|}{p_i + q_i} \tag{24}$$

$$\geq \frac{1}{2} \sum_{i \in \Omega} \frac{p_i^2 + 2p_i q_i + q_i^2 - (p_i^2 + q_i^2)}{p_i + q_i} \tag{25}$$

$$= \sum_{i \in \Omega} \frac{p_i q_i}{p_i + q_i} \tag{26}$$

$$= D_{HM}[P, Q]. \tag{27}$$

D Tunstall Codes

This section provides a concise overview of Tunstall coding. Source coding is a fundamental technique for representing sequences of source symbols, like text or tokens, as sequences of bits (or symbols from another alphabet). The goal is efficient representation for storage or transmission. Tunstall coding is a variable-to-fixed length source coding method. This means it parses the source sequence into variable-length subsequences, and maps each of these subsequences to a fixed-length codeword. Let V denote the expected length of the encoded source subsequences.

Tunstall codes are constructed using trees. The construction process begins with a root node. Assuming a source alphabet Ω, the root is expanded to have $|\Omega|$ children. Subsequently, in each step, the leaf node representing the most probable source sequence is expanded by adding $|\Omega|$ children to it. This expansion process is repeated until a desired number of codewords is reached. The structure and construction of this Tunstall tree, specifically its inner nodes, are identical to the optimal draft tree employed in speculative sampling as described in Algorithm 3; conversely, the leafs of the Tunstall tree correspond to sampling of an additional token once no more drafted tokens are considered (i.e., sampling from residual in GSS or last token in ERSS) . If the construction process expands k nodes, the resulting Tunstall tree will have $|\Omega| + k(|\Omega| - 1)$ leaves. Each path from the root to a leaf represents a variable-length sequence of source symbols. Tunstall code assigns a unique codeword of fixed length $\lceil \log (k(|\Omega| - 1) + |\Omega|) \rceil$ bits to each leaf node.

Considering the fundamental limit of compression given by the source entropy, we can establish an inequality for compressing a source sequence of length L with a Tunstall code:

$$L\mathbf{H}[R] \leq \frac{L}{V} \log (k(|\Omega| - 1) + |\Omega|) \tag{28}$$

Here, the left-hand side of Equation (28) represents the lower bound on the average number of bits needed to encode a sequence of length L, and the right-hand side represents the expected number of bits used by the Tunstall code. The term $\frac{L}{V}$ represents the expected number of codewords needed to encode a source sequence of length L, and $\log (k(|\Omega| - 1) + |\Omega|)$ is the fixed length of each codeword. As both the Tunstall tree construction and the optimal drafting tree in Algorithm 3 describe the same mathematical structure in different contexts, the 'length of the encoded source sequence' in Tunstall coding corresponds precisely to the 'number of generated tokens' in speculative sampling. Rearranging the equation, we obtain:

$$G_{\tau*} = V \leq \frac{\log (k(|\Omega| - 1) + |\Omega|)}{\mathbf{H}[R]} \leq \frac{\log |\Omega| + \log(k + 1)}{\mathbf{H}[R]}. \tag{29}$$

Tunstall codes are known to be asymptotically optimal, even for sources with memory (Savari and Gallager, 1997), when a different code for each state, which would translate to different draft trees depending on the context. This asymptotic optimality means that the gap to the theoretical compression limit becomes constant as k increases. Therefore, this upper bound also characterizes the asymptotic behavior of speculative sampling.