SpecTr++: Improved transport plans for speculative decoding of large language models

Kwangjun Ahn[*]
MIT EECS
kjahn@mit.edu

Ahmad Beirami
Google Research
beirami@google.com

Ziteng Sun
Google Research
zitengsun@google.com

Ananda Theertha Suresh
Google Research
theertha@google.com

Abstract

We revisit the question of accelerating decoding of language models based on speculative draft samples, inspired by Leviathan et al. [2023], Chen et al. [2023]. Following Sun et al. [2023] which makes connections between speculative decoding and optimal transport theory, we design improved transport plans for this problem with no sacrifice in computational complexity in terms of the alphabet size.

1 Introduction

Autoregressive large language models (LLMs) such as GPT-3 [Brown et al., 2020] and PALM [Chowdhery et al., 2022] have shown impressive performance in several natural language processing (NLP) tasks [Thoppilan et al., 2022, Touvron et al., 2023]. Given a context $x^t := x(1), x(2), \dots, x(t)$, an autoregressive language model $\mathcal{M}$ generates the next token $x(t+1)$ via sampling from the conditional distribution $\mathcal{M}(\cdot|x^t)$. One of the most popular method is to sample from a temperature-scaled version of $\mathcal{M}(\cdot|x^t)$ usually called *temperature sampling* [Ackley et al., 1985, Ficler and Goldberg, 2017]. If the temperature is chosen to be zero (referred to as *greedy decoding*), the next token is determined by $x(t+1) = \arg\max_{x\in\Omega}\mathcal{M}(x|x^t)$, where Ω is the entire set of tokens (usually called *vocabulary set*).

However, there is one inherent limitation to autoregressive language models: during the inference time, the language model $\mathcal{M}$ has to generate or decode each token one-by-one *autoregressively*. In other words, the total decoding time scales with the total number of tokens need to be generated, which could be prohibitive in many applications [Stern et al., 2018]. Consequently, several techniques have been developed to speed up transformer-based language model decoding. Before we discuss those, we first present a simplified computational model, following Sun et al. [2023].

Computational model. Given an autoregressive model $\mathcal{M}$ and a context x^t, with $O(t^2)$ computation and $O(1)$ time, one can compute the conditional distribution $\mathcal{M}(\cdot|x^t)$. Specifically, we allow a ***parallel computation along time and batch:*** given a several contexts, $x_1^t, x_2^t, \dots, x_k^t$, with $O(k \cdot t^2)$ computation and $O(1)$ time, one can compute $\mathcal{M}(\cdot|x_j^i)$ for all $i = 1, 2, \dots, t$ and $j = 1, \dots, k$.

The main premise of the above computation model is that the computation along time and batch axes does not increase the computation time, with an appropriate parallel computing. It is a simplified characterization of the typical hardware used in practice, such as TPUs and GPUs.

In this work, following [Leviathan et al., 2023, Chen et al., 2023, Sun et al., 2023], we propose a generic algorithm for accelerating language model decoding/inference based on *speculative decoding*.

[*]This work was done while Kwangjun Ahn was an intern at Google Research.

Accepted to the Optimal Transport and Machine Learning Workshop at NeurIPS 2023.

Similar to prior work, the approach is **generic and modular**, which could be combined with other methods that improve the latency of the model, such as efficient transformer architectures. We first discuss some background on speculative sampling to set the stage for our main results.

2 Preliminaries: speculative sampling and optimal transport

2.1 Speculative sampling

The main idea of speculative sampling is: *First predict multiple tokens with smaller models* $\mathcal{M}_{\mathsf{s}}$ *and validate these tokens with bigger models* $\mathcal{M}_{\mathsf{b}}$. This general idea was extensively employed for greedy decoding (i.e., zero temperature) [Stern et al., 2018, Ge et al., 2022, Yang et al., 2023a, Kim et al., 2023, Yang et al., 2023b], and recently extended to a more practical setting of sampling in two concurrent works [Leviathan et al., 2023, Chen et al., 2023] under the name of *speculative sampling/decoding*. While deferring a more mathematical exposition to the next section, we first qualitatively describe the main ideas.

The main premise is that we have access to a smaller, i.e., computationally cheaper, model $\mathcal{M}_{\mathsf{s}}$ for draft selection, with a property that it predicts the conditional distribution $\mathcal{M}_{\mathsf{s}}(\cdot|x^t)$ which is "similar" to that of a larger model $\mathcal{M}_{\mathsf{b}}(\cdot|x^t)$ for any given prefix x^t. Given this access, speculative sampling/decoding works as follows:

1. *Draft construction.* We first use the smaller model $\mathcal{M}_{\mathsf{s}}$ to efficiently and "speculatively" construct ℓ draft tokens, and extend the current prefix $x^t = x(1), x(2), \ldots, x(t)$ with $\tilde{x}(t+1), \tilde{x}(t+2), \ldots, \tilde{x}(t+\ell)$. We then keep the conditional distributions $\mathcal{M}_{\mathsf{s}}(\cdot|x^t, \tilde{x}^{t+1:t+i})$ for each $i < \ell$.
2. *Parallel computation of actual conditional distributions.* Given the observed draft samples, we compute the conditional distribution of the larger model, $\mathcal{M}_{\mathsf{b}}(\cdot|x^t, \tilde{x}^{t+1:t+i})$ for each $i \le \ell$.
3. *Draft validation.* "***Carefully validate***" and select ℓ' of the ℓ tokens and set $x(t+i) = \tilde{x}(t+i)$ for $i \le \ell'$. And with the new context $x^{t+\ell'}$, we repeat this whole process from start.

The most important part of speculative sampling is the "***careful validation***" step. In particular, we need to carefully design the validation step such that the output samples have the same distribution as that of $\mathcal{M}_{\mathsf{b}}$. Fortunately for us, this question of generating samples from a distribution given access to samples from another distribution is well-studied in probability theory, and generally referred to as *coupling* (see, e.g., [Den Hollander, 2012]), which we formally define here.

Definition 1 (Coupling)**.** For two distributions p over Ω_p and q over Ω_q, we say a joint distribution π over $\Omega_p \times \Omega_q$ is called a coupling between p and q if its marginal on Ω_p and Ω_q are p and q, respectively. In other words, $\sum_{y \in \Omega_q} \pi(x, y) = p(x)$ and $\sum_{x \in \Omega_p} \pi(x, y) = q(y)$. Let $\Pi(p, q)$ be the set of all possible couplings between p and q.

With the notion of coupling, the design of validation boils down to a design of an appropriate coupling. In particular, we want to design a coupling carefully such that the number of *accepted* tokens is maximized. This draft validation problem is formalized as an optimal transport problem [Villani et al., 2009, Villani, 2021] in [Sun et al., 2023]. For a principled understanding, we follow the view of Sun et al. [2023] and summarize the previous approaches from the perspective of designing a good coupling.

2.2 Optimal transport view of speculative sampling

For simplicity, we describe previous approaches at the token-level draft selection stage, i.e., there is a single-token draft and the validation step decides whether to accept this draft. Throughout the section, we fix a context x^t and let

$$p \coloneqq \mathcal{M}_{\mathsf{s}}(\cdot|x^t) \text{ and } q \coloneqq \mathcal{M}_{\mathsf{b}}(\cdot|x^t).$$

Then we want to design a coupling that maximizes the chance of getting a draft sample accepted. This can be nicely formulated in the framework of *optimal transport*.

Definition 2 (Optimal transport [Villani et al., 2009, Villani, 2021])**.** For a cost function $c : \Omega_p \times \Omega_q \to \mathbb{R}_+$, we define the cost associated to a coupling $\pi \in \Pi(p, q)$ as

$$C(\pi) = \mathbb{E}_{(X,Y)\sim\pi} c(X, Y)\,.$$

The *optimal transport plan* is referred to a coupling π that minimizes the transport cost. More precisely, the transport map $T_\pi : \Omega_p \to \mathcal{P}(\Omega_q)$ is defined as $T_\pi(y \mid x) = \pi(x,y)/p(x)$.

With the notion of optimal transport, Sun et al. [2023] formulate the design of a coupling as the following optimal transport problem with $\Omega_p = \Omega_q = \Omega$ and

$$\forall x, y \in \Omega, \quad c(x,y) = \mathbb{1}\{x \neq y\} \ . \tag{1}$$

Hence the coupling cost is equal to $C(\pi) = \mathbb{E}_{X,Y\sim\pi}\mathbb{1}\{X \neq Y\} = \mathbb{P}(X \neq Y)$. In fact, the validation step designed in the previous works [Leviathan et al., 2023, Chen et al., 2023] is a transport map for the optimal transport problem (1), as pointed out by Sun et al. [2023]. The transport map for (1) is also known as *maximal coupling* in the literature [Den Hollander, 2012].

2.3 Sequential speculative sampling of Sun et al. [2023]

Noticing the connection between the scheme in [Leviathan et al., 2023, Chen et al., 2023] and optimal tranport, Sun et al. [2023] proposed an improvement over the scheme by further exploiting ***parallelization over batch***. The main idea is to introduce ***multiple draft tokens*** to increase the chance of acceptance. More formally, instead of sampling a single draft sample x from $\mathcal{M}_{\mathsf{s}}(\cdot|x^t)$, we draw k samples $x_1, x_2, \ldots, x_k$. Then the corresponding optimal transport problem becomes: $\Omega_{p^{\otimes k}} = \Omega^{\otimes k}$, $\Omega_q = \Omega$ and

$$\forall (x_1,\ldots,x_k) \in \Omega^{\otimes k}, y \in \Omega, \quad c((x_1,\ldots,x_k), y) = \mathbb{1}\{y \notin \{x_1,\ldots,x_k\}\} \ . \tag{OTM_k}$$

We call the optimal transport problem with the cost given as (OTM_k) the ***optimal transport with membership cost*** and denote its optimal cost by OTM_k. Note that OTM_k recovers (1) when $k = 1$.

Although OTM_k improves upon the scheme in [Leviathan et al., 2023, Chen et al., 2023], unfortunately, to the best of our knowledge, we are currently unaware of an efficient method to solve OTM_k. More precisely, although the optimal transport problem can be written as a linear program [Villani et al., 2009, Pele and Werman, 2009], given that the support of $p^{\otimes k}$ has $|\Omega|^k$ elements, the computational complexity of solving the program scales polynomially with $|\Omega|^k$, which could be too costly. We provide a detailed discussion with the comparison between our method and the existing optimal transport solvers in Subsection 4.3.

To provide an efficient speedup algorithm, an approximate solution to the OTM_k was proposed in [Sun et al., 2023], which they called k-sequential selection algorithm (kSEQ). We will revisit their method and give a new interpretation for it, which will form a basis for our new algorithm development. This will be the focus of the next section.

3 A family of sequential speculative sampling schemes

The basis of our method is the following canonical "sequential" selection process:

Algorithm 1 k-sequential selection algorithm (kSEQ)

```
input: $k$ draft samples, $x_1,\ldots,x_k$ sampled as per $p$, and carefully chosen constants
    $\alpha_1,\alpha_2,\ldots,\alpha_k > 0$ constants and subsets $\Omega_1,\ \Omega_2,\ \cdots,\ \Omega_k$ of the vocabulary set $\Omega$.
for $i = 1$ to $k$ do
    Accept $y = x_i$ with probability $\begin{cases} \alpha_i \cdot \frac{q(x_i)}{p(x_i)} & \text{if } x_i \in \Omega_i \\ 1 & \text{otherwise} \end{cases}$ and continue if rejected.
end for
output: Accepted sample $y$. (If all are rejected, sample $y$ from the "residual" distribution.)
```

For $k = 1$, the optimal transport plan for OTM_k [Leviathan et al., 2023, Chen et al., 2023] precisely corresponds to Algorithm 1 where $\alpha_1 = 1$ and $\Omega_1 = \{x \in \Omega : q(x) \geq p(x)\}$. We extend their scheme to multiple draft samples, and give it an additional flexibility to vary α_i's and Ω_i's. In fact, the improved algorithm of Sun et al. [2023] is also a special case of Algorithm 1 where $\alpha_i \equiv \alpha^\star$ and $\Omega_i \equiv \Omega^\star$ for carefully chosen $\alpha^\star$ and $\Omega^\star$ (we will shortly discuss this more precisely). However, below we discuss that choosing identical α_i's and Ω_i's leads to a suboptimal performance even for a very benign example of Bernoulli random variables.

Example 1 (Bernoulli random variables)**.** It turns out choosing identical α_i's and Ω_i's as in Sun et al. [2023] leads to a suboptimal performance even for the simplest example of Bernoulli random variables. Consider the case where $p = \mathsf{Bern}(0.5)$ and $q = \mathsf{Bern}(0.75)$ and $\Omega = \{0,1\}$. In this case, we have $\mathsf{OTM}_1 = 0.25$ and $\mathsf{OTM}_k = 0$ for $k \geq 2$. However, when we choose identical α_i's and Ω_i's as in Sun et al. [2023], the optimal cost for $k = 2$ is ≈ 0.095, which is strictly greater than OTM_2. We will revisit this example in Subsection 4.4 and show that our proposed algorithms can close this gap and achieve the optimal cost.

Given the above example, it is natural to ask how to achieve optimal performance for the above Bernoulli example. To that end, let us inspect and compare the transport plan for two schemes. It turns out that for the optimal kSEQ plan, the optimal choice is given as $\alpha_i \equiv 0.764$, while for the OTM_2 plan, $\alpha_1 = 0$ and $\alpha_2 = 2$. Hence, even for this basic example, the lesson is that: ***one needs to carefully choose*** α_i***'s for higher draft acceptance rate.*** We discuss this next.

3.1 How to choose α_i's?

For the moment, let us assume that Ω_i's are already chosen and the main focus is to find an optimal α_i's; we will discuss how to select Ω_i's in Section 4. For simple exposition, we assume throughout the section that $q(x) > 0$ for all $x \in \Omega$ $\max_{x\in\Omega} \frac{q(x)}{p(x)} < \infty$.[2]

First, carefully inspecting Algorithm 1, note that the transport (or the probability that all draft samples are rejected) is given as:

$$\textbf{Transport cost:} \quad \prod_{i=1}^{k}(1-\beta_{i;\alpha_i}), \quad \text{where} \quad \boxed{\beta_{i;\alpha_i} := p(\Omega_i^c) + \alpha_i \cdot q(\Omega_i)}\,. \tag{2}$$

This is because we accept x_i with probability 1 if $x_i \in \Omega_i^c$ and with probability $\alpha_i \cdot \frac{q(x_i)}{p(x_i)}$ if $x_i \in \Omega_i$. Hence our goal is to minimize the quantity (2). On the other hand, in order for Algorithm 1 to be a valid coupling, there are several constraints we need to consider, resulting in:

$$\boxed{\begin{cases} \underline{\textbf{Optimization variables: } \alpha_i \ \text{ for } i = 1,2,\dots,k \text{ s.t. } \Omega_i \neq \emptyset.} \\ \text{(for } i \text{ s.t. } \Omega_i = \emptyset\text{, we set } \alpha_i = 0 \text{ and exclude it from optimization variables)} \\ \min_{\alpha_i} \prod_{i=1}^{k}(1-\beta_{i;\alpha_i}) \qquad (\triangleright \text{ transport cost.}) \\ \text{s.t. } \sum_{i=1}^{k}\left[r_i(x)\cdot\prod_{j=1}^{i-1}(1-\beta_{j;\alpha_j})\right] \leq q(x), \quad \forall x \in \Omega \ (\triangleright \text{ accept prob. } \leq q(x)) \\ \qquad \text{where } \ r_i(x) := \begin{cases} q(x)\alpha_i & \text{if } x \in \Omega_i \\ p(x) & \text{otherwise.} \end{cases} \\ \quad 0 \leq \alpha_i \leq \min_{x\in\Omega_i}\frac{p(x)}{q(x)}, \quad \forall i \text{ s.t. } \Omega_i \neq \emptyset. \qquad (\triangleright \text{ accept prob. } \leq p(x)) \end{cases}} \tag{OPT_k}$$

The next result shows that the validity of the coupling together with the optimality.

Theorem 1. *Let p, q be the probability distributions over the vocabulary set Ω. For given subsets $\Omega_1, \Omega_2, \dots, \Omega_k$ of Ω, any feasible points $\alpha_1, \alpha_2, \dots, \alpha_k$ of (OPT_k), when used in Algorithm 1 results in a valid coupling between $p^{\otimes k}$ and q. Moreover, let $\alpha_1^\star, \alpha_2^\star, \dots, \alpha_k^\star$ be the solution to (OPT_k). Then Algorithm 1 with Ω_i's and $\alpha_i^\star$'s is a valid coupling that is optimal in terms of the membership cost (OTM_k) among all the couplings of the form of Algorithm 1 with Ω_i's.*

Proof. See Subsection A.1 for the proof. □

Given Theorem 1, we are now interested in how to solve this nonlinear program (OPT_k).

How to solve (OPT_k)? At first glance, the optimization problem (OPT_k) is a nonconvex polynomial optimization problem, which could be hard to solve in general. However, fortunately to us, it turns out one can reparametrize the problem so that it becomes a linear program in new variables, as summarized in the following theorem.

[2] Our discussion applies to the general case by considering the "effective" alphabets defined as $\Omega_{\mathsf{eff}} := \{x \in \Omega \ : \ q(x) > 0\}$, and assuming that $(\Omega \setminus \Omega_{\mathsf{eff}}) \subseteq \Omega_i$ for all $i = 1, 2, \dots, k$.

Theorem 2 (OPT_k **can be solved via a linear program!).** *Under the setting of Theorem 1, consider the following substitution of* (OPT_k)*:* $u_i = \prod_{j=1}^{i}(1-\beta_{j;\alpha_j})$ *for* $i = 1, 2, \ldots, k$*, and* $u_0 := 1$*. Then, the optimization problem* (OPT_k) *is equivalent to the* ***linear program*** (LP_k) *which has* k *variables and* $|\Omega| + k$ *constraints.*

Moreover, let $u_1^\star, u_2^\star, \ldots, u_k^\star$ *be the solution to the linear program* (LP_k)*. Then, the following conversion rule turns the optimal solution* $u_1^\star, u_2^\star, \ldots, u_k^\star$ *of* (LP_k) *into an optimal solution of* (OPT_k)*: for* $i = 1, 2, \ldots, k$*, (i) if* $\Omega_i \neq \emptyset$*, set* $\alpha_i^\star := {}^{p(\Omega_i)}/_{q(\Omega_i)} - {}^{u_i}/_{(u_{i-1}\cdot q(\Omega_i))}$*; if* $\Omega_i = \emptyset$*, set* $\alpha_i^\star := 0$*. In other words,* $\alpha_i^\star$*'s defined as above are the optimal solution to* (OPT_k)*.*

Proof. See Subsection A.2 for the proof. □

Thanks to Theorem 2, one can efficiently compute the optimal α_i's given the choice of Ω_i's. We next build on the results from this section and discuss how to iteratively improve the solution.

3.2 Iterative scheme

Let $\Omega_i^{(0)}$, $i = 1, 2, \ldots, k$, be the initial choices of subsets. The main idea is that for a given collection of initial subsets, we would like to iteratively update the optimal $\alpha_i^{(j)}$'s and $\Omega_i^{(j)}$'s, for $j = 1, 2, \ldots$. Given Theorem 2, this iterative scheme can be implemented as Algorithm 2.

Algorithm 2 Iterative optimal ratios selection ($\mathsf{iOPT}_k^{(T)}$)

```
input: Initial choice of subsets $\Omega_1^{(0)}, \Omega_2^{(0)}, \cdots, \Omega_k^{(0)} \subset \Omega$. Number of iterations $T$.
for $j = 1, 2, \ldots, T$ do
  Let $\alpha_i^{(j-1)}$'s be the solution to ($\mathsf{OPT}_k$) with $p$ and $q$, and the subsets $\Omega_i^{(j-1)}$'s.
  ▷ Note: thanks to Theorem 2, this step amounts to solving a linear program ($\mathsf{LP}_k$).
  Set $\Omega_i^{(j)} := \{x \in \Omega_i^{(j-1)} : \alpha_i^{(j-1)} q(x) < p(x)\}$ for $j = 1, 2 \ldots, k$.
  if $\Omega_i^{(j)} = \Omega_i^{(j-1)}\ \forall i$ then
    break
  end if
end for
output: $\Omega_1^{(j)}, \Omega_2^{(j)}, \cdots, \Omega_k^{(j)}$ and $\alpha_1^{(j)}, \ldots, \alpha_k^{(j)}$.
```

Given this iterative scheme, the natural question is whether Algorithm 2 terminates within a finite number of iterations. The following result precisely addresses this.

Theorem 3. *Each iteration of the iterative scheme* $\mathsf{iOPT}_k^{(\infty)}$ *(Algorithm 2) monotonically decreases the transport cost. Moreover, it terminates in at most* $k|\Omega|$ *iterations.*

Proof. Let us consider a single iteration of the iterative scheme. For a given collection of subsets Ω_i's, let α_i's be the solution to (OPT_k) with Ω_i's. Let us denote the subsets for the next iteration by Ω_i', i.e., $\Omega_i' := \{x \in \Omega_i : \alpha_i \cdot q(x) < p(x)\}$. Then, it holds that $\Omega_i' \subseteq \Omega_i$ for each $i = 1, 2, \ldots, k$. Then by definition, it holds that $\Omega_i' \subseteq \Omega_i$. In order to show that each iteration monotonically decreases the transport cost, for previous solution α_i's to (OPT_k), it holds that for all $x \in \Omega_i$, $\alpha_i \cdot q(x) \leq p(x)$. This is precisely due to the last constraint in (OPT_k). This shows that the previous α_i's are still feasible for the next Ω_i''s. Hence, the next iteration will only decrease the transport cost further. □

4 How to initialize the algorithm?

In our main algorithm, we initialize the subsets Ω_i's based on the scheme of Sun et al. [2023].

4.1 The algorithm of Sun et al. [2023]

Although, Sun et al. [2023] had a different motivation for their algorithm design, one can in fact present their algorithm from a general perspective, given our discussion in Subsection 3.1.

Interpretation of Sun et al. [2023]. It corresponds to the optimization problem (OPT_k) where

- we choose subsets Ω_i's to be same, i.e., $\Omega_i \equiv \widehat{\Omega}$ for some $\widehat{\Omega}$, and
- more importantly, we add **an additional constraint** that **all the α_i's have to be the same**, i.e., $\alpha_i \equiv \widehat{\alpha}$ for all $i = 1, 2, \dots, k$.

Before detailing the choice of $\Omega_\star$, we first write down the optimization problem (OPT_k) for the case when all Ω_i's are chosen to be same, i.e., $\Omega_i \equiv \widehat{\Omega}$, and α_i's are chosen to be the same, i.e., $\alpha_i \equiv \widehat{\alpha}$ for all $i = 1, 2, \dots, k$. Hence, the token-wise acceptance probability is now defined as $\beta_{\widehat{\alpha}} := p(\widehat{\Omega}^c) + \widehat{\alpha} \cdot q(\widehat{\Omega})$. In that case, the optimization problem (OPT_k) gets simplified to:

$$\begin{cases} \min_{\widehat{\alpha}} & (1-\beta_{\widehat{\alpha}})^k \\ \text{s.t.} & 1-(1-\beta_{\widehat{\alpha}})^k \leq \widehat{\alpha}^{-1} \cdot \beta_{\widehat{\alpha}} \,, \\ & 1-(1-\beta_{\widehat{\alpha}})^k \leq \min_{x \in \widehat{\Omega}^c} \frac{q(x)}{p(x)} \cdot \beta_{\widehat{\alpha}} \,, \\ & 0 \leq \widehat{\alpha} \leq \min_{x \in \widehat{\Omega}} \frac{p(x)}{q(x)} \,. \end{cases} \tag{3}$$

See Subsection A.3 for details. In fact, now that there is only a single optimization variable, we can make another modification that enables us to optimize over $\widehat{\Omega}$. Since we only have a single optimization variable $\widehat{\alpha}$, one can actually include the choice of $\widehat{\Omega}$ in the optimization. One can do this by setting $\widehat{\Omega} = \{x \in \Omega \ : \ p(x) \geq \widehat{\alpha} \cdot q(x)\}$. With this choice, the token-wise acceptance probability can be written as:

$$\widehat{\beta}_{\widehat{\alpha}} := \sum_{x \in \Omega} \min\{p(x),\ \widehat{\alpha} \cdot q(x)\} \,,$$

which is not a linear function but a piece-wise linear function of $\widehat{\alpha}$. With this choice, (3) gets further simplified (see Subsection A.4):

$$\begin{cases} \min_{\widehat{\alpha} \geq 0} & (1-\widehat{\beta}_{\widehat{\alpha}})^k \\ \text{s.t.} & 1-(1-\beta_{\widehat{\alpha}})^k \leq \widehat{\alpha}^{-1} \cdot \widehat{\beta}_{\widehat{\alpha}} \,. \end{cases} \tag{4}$$

Inspecting the constraint of (4), notice that the LHS is increasing in $\widehat{\alpha}$ and the RHS is decreasing in $\widehat{\alpha}$. Hence, the unique solution $\alpha_\star^{(k)}$ of the above optimization problem is the solution of the equation

$$\alpha_\star^{(k)} \geq 0 \quad \text{s.t.} \quad 1-(1-\beta_{\alpha_\star^{(k)}})^k = (\alpha_\star^{(k)})^{-1} \beta_{\alpha_\star^{(k)}} \,. \tag{5}$$

We now discuss how we initialize Algorithm 2 with this scheme.

4.2 SpecTr++: two improvements over [Sun et al., 2023]

We now use this solution to initialize the subsets Ω_i's for Algorithm 2. As we discussed in Subsection 4.1, the optimization problem (4) lets us also optimize over the subset $\widehat{\Omega}$. Hence, the subset induced by the optimization problem (4) is a good candidate for the initialization.

Definition 3 (Saturated subsets). For a given pair of distributions p and q, let $\alpha_\star^{(k)}$ be the optimal solution of (5). We define the saturated subset as $\Omega_\star^{(k)} := \{x \in \Omega \ : \ p(x) \geq \alpha_\star^{(k)} \cdot q(x)\}$.

For Algorithm 2, let us initialize the subsets Ω_i's to be $\Omega_\star^{(k)}$. Since each iteration of $\mathsf{iOPT}_k^{(T)}$ monotonically decreases the transport cost, we have the two improvements over Sun et al. [2023].

Definition 4 (SpecTr++). For a given pair of distributions p and q, consider running $\mathsf{iOPT}_k^{(T)}$ (Algorithm 2) with the initialization of subsets given as $\Omega_i \equiv \Omega_\star^{(k)}$. Let $\mathsf{STr}_k(p, q)$ be the algorithm of [Sun et al., 2023], i.e., $\mathsf{iOPT}_k^{(0)} = \mathsf{STr}_k(p, q)$. Then, we define the following two **improvements**:

- $\mathsf{STr}_k^{+}(p, q)$: the result of $\mathsf{iOPT}_k^{(1)}$.
- $\mathsf{STr}_k^{++}(p, q)$: the result of $\mathsf{iOPT}_k^{(\infty)}$.

Computational complexity of proposed methods. Here we discuss the computational complexity of the methods in Definition 4. As noted by Sun et al. [2023], in practical applications, we typically have $k \ll |\Omega|$, so we mainly focus on the complexity dependence on $|\Omega|$.

- As discussed in [Sun et al., 2023], the runtime of STr_k is equal to $O(|\Omega| \log k)$.
- During the first iteration of Algorithm 2, since Ω_i's are all equal, the first $|\Omega|$ constraints of (LP_k) becomes identical. Hence, (LP_k) has k variables and $k+1$ constraints. Hence, the total complexity of STr_k^+ is still $O(|\Omega| \log k)$, where the polynomial dependence on $|\Omega|$ is disregarded.
- Recall that the complexity of linear program in the fixed dimension with m constraints is $O(m)$ (see, e.g., [Megiddo et al., 1986]). Hence, for T iterations of Algorithm 2, the total complexity is $O(|\Omega| \log k + T \cdot |\Omega|)$. In light of Theorem 3, it holds that the worst-case complexity of STr_k^{++} is $O(k \cdot |\Omega|^2)$. However, in practice, we observe that the number of iterations Algorithm 2 takes to converge is much smaller than $k \cdot |\Omega|$ and the bound is rather conservative.

We now compare the computation complexity of our schemes with the existing(approximate) optimal transport solvers.

4.3 Comparison with optimal transport solvers

Let $V = |\Omega|$. We first note that the optimal transport problem (OTM_k) is a linear program with V^{k+1} variables (in order to express the joint distribution) and $O(V^k + V)$ constraints to enforce the constraints for correct marginal distributions. Hence, the computational complexity of solving this linear program using the interior point method requires time $O(V^{3.5(k+1)})$ [Renegar, 1988], and a method based on Lee-Sidford barrier requires time $\widetilde{O}(V^{2.5(k+1)})$ [Lee and Sidford, 2014]. Moreover, other practical algorithm like Orlin's algorithm based on minimul cost flow has complexity of $\widetilde{O}(V^{3(k+1)})$. Note that all these methods have complexity of $O(V^{\Omega(k)})$, which could be highly costly for applications where both V and k become larger.

Next, we make a comparison with another popular optimal transport solvers based-on Sinkhorn projections [Cuturi, 2013, Altschuler et al., 2017]. For a detailed comparison, let $\mathsf{C} \in (\mathbb{R}^V)^{\otimes(k+1)}$ be the cost matrix of (OTM_k), i.e.,

$$\mathsf{C}_{i_1,i_2,\ldots,i_k,j} = \mathbb{1}\left\{j \notin \{i_1,\ldots,i_k\}\right\} \quad \text{for } i_1,i_2,\ldots,i_k,j \in \Omega.$$

Then, note that the number of nonzero elements in C is $\Omega(V^k)$. The main component of Sinknorn iteration is the Sinkhorn projection of the matrix $A = \exp(-\eta \mathsf{C})$ for a chosen step size $\eta > 0$. More specifically, the for scaling constants given an iteration, say $x \in (\mathbb{R}^V)^{\otimes k}$ and $y \in \mathbb{R}^V$, the Sinkhorn projection requires the computation of $\mathrm{diag}(x) A \mathrm{diag}(y)$, which requires time $\Omega(V^k)$, since A has at least $\Omega(V^k)$ nonzero entries. Hence the overall computation complexity would be still $O(V^{\Omega(k)})$.

We conclude this section by revisiting the example of Bernoulli distributions (Example 1) and discuss how our proposed methods overcome the limitations of [Sun et al., 2023].

4.4 The Bernoulli example (Example 1) revisited

Consider a pair of Bernoulli distributions, i.e., $p = \mathsf{Bern}(0.5)$ and $q = \mathsf{Bern}(0.75)$ and $\Omega = \{0,1\}$. For $k = 2$, the optimal transport plan is equal to:

- OTM_2 **plan:** $\begin{cases} \text{w.p. } 1\,, & \text{if } X_1 = 1 \\ \text{w.p. } 0\,, & \text{if } X_1 = 0 \end{cases}$ and $\begin{cases} \text{w.p. } 1\,, & \text{if } X_2 = 1 \\ \text{w.p. } 1\,, & \text{if } X_2 = 0 \end{cases}$.

On the other hand, $\alpha_\star^{(k)}$ defined in (5) is 0.764, which leads to the following transport plan:

- **Sun et al. [2023] plan:** $\begin{cases} \text{w.p. } 1\,, & \text{if } X_i = 1 \\ \text{w.p. } 0.764 \cdot 0.5 = 0.382\,, & \text{if } X_i = 0 \end{cases}$ for $i = 1, 2$.

This transport plan does not achieve the OTM transport cost of 0. On the other hand, a single iteration of Algorithm 2 (namely $\mathsf{STr}_k^+(p,q)$) would lead to the solution of $\alpha_1 = 0$ and $\alpha_2 = 2$. Note that this choice of α_1, α_2 corresponds to the optimal transport plan.

5 Experiments

In order to test the performance of our proposed methods from Definition 4, we conduct an experiment where the distributions p and q are generated randomly over a fixed vocabulary set Ω. In NumPy, we used the code `p = np.random.rand(|Ω|)` and `p = p/np.sum(p)`. We measure the performance of each algorithm according to the *token acceptance probability*: for an algorithm $\mathcal{A}$ and a pair of distributions p, q, we define the acceptance probability $\mathsf{acc}^{\mathcal{A}}(p, q)$ as $(1 -$ transport cost of $\mathcal{A}(p, q))$. We repeat the experiments over 100 different random pairs of distributions. We choose the support size $|\Omega| \in \{5, 10\}$ to make the OTM solution feasible. We compare the following five algorithms:

- Three algorithms $\mathsf{STr}_k(p, q)$, $\mathsf{STr}_k^+(p, q)$, and $\mathsf{STr}_k^{++}(p, q)$ from Definition 4.
- The optimal transport plan (OTM_k).
- The optimal kSEQ algorithm (of the form Algorithm 1) where we search over all possible sequences of subsets $\Omega_1, \Omega_2, \ldots, \Omega_k$ and pick the best choice.

In Figure 1 and Figure 2, we report the experimental results for the support sizes 5 and 10, respectively. For each algorithm $\mathcal{A}$, we report two different metrics: (i) acceptance probability $\mathsf{acc}^{\mathcal{A}}$; (ii) the worst-case approximation ratio for $\mathcal{A}$ formally defined as $\min_{p,q} \frac{\mathsf{acc}^{\mathcal{A}}(p,q)}{\mathsf{acc}^{\mathrm{OTM}}(p,q)}$, where the minimum is taken over 100 different random pairs of distributions.

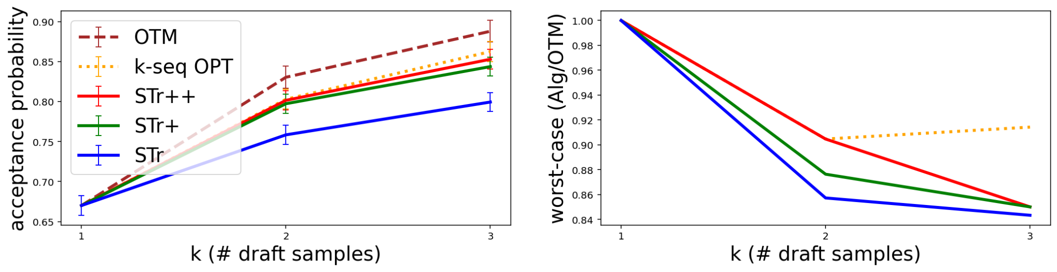

Figure 1: Results for $|\Omega| = 5$.

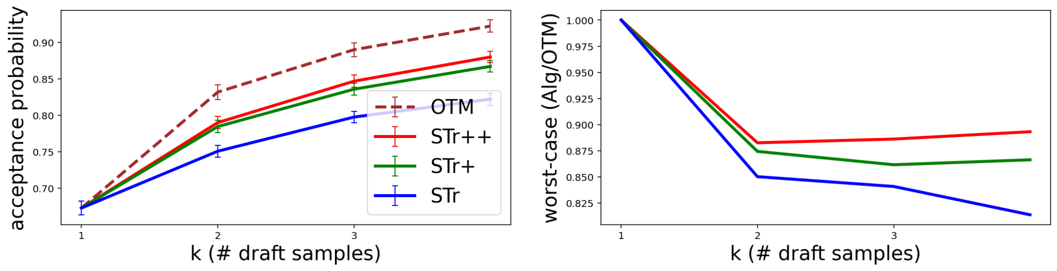

Figure 2: Results for $|\Omega| = 10$.

From Figure 1, one can see that the performance of our proposed methods are quite close to that of the optimal kSEQ algorithm. Compared to the optimal transport plan, we observe that among the examples, our proposed methods achieve an approximation ratio of at least ≈ 0.85, which is an improvement over STr [Sun et al., 2023]. Similar observations holds for a bigger support size of 10 as shown in Figure 2. We leave rigorously proving that the propose methods achieve better worst-case approximation ratios than STr as an important future direction.

References

David H Ackley, Geoffrey E Hinton, and Terrence J Sejnowski. A learning algorithm for boltzmann machines. *Cognitive science*, 9(1):147–169, 1985.

Jason Altschuler, Jonathan Niles-Weed, and Philippe Rigollet. Near-linear time approximation algorithms for optimal transport via sinkhorn iteration. *Advances in neural information processing systems*, 30, 2017.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020.

Charlie Chen, Sebastian Borgeaud, Geoffrey Irving, Jean-Baptiste Lespiau, Laurent Sifre, and John Jumper. Accelerating large language model decoding with speculative sampling. *arXiv preprint arXiv:2302.01318*, 2023.

Aakanksha Chowdhery, Sharan Narang, Jacob Devlin, Maarten Bosma, Gaurav Mishra, Adam Roberts, Paul Barham, Hyung Won Chung, Charles Sutton, Sebastian Gehrmann, et al. Palm: Scaling language modeling with pathways. *arXiv preprint arXiv:2204.02311*, 2022.

Marco Cuturi. Sinkhorn distances: Lightspeed computation of optimal transport. *Advances in neural information processing systems*, 26, 2013.

Frank Den Hollander. Probability theory: The coupling method. *Lecture notes available online (http://websites. math. leidenuniv. nl/probability/lecturenotes/CouplingLectures. pdf)*, 2012.

Jessica Ficler and Yoav Goldberg. Controlling linguistic style aspects in neural language generation. *arXiv preprint arXiv:1707.02633*, 2017.

Tao Ge, Heming Xia, Xin Sun, Si-Qing Chen, and Furu Wei. Lossless acceleration for seq2seq generation with aggressive decoding. *arXiv preprint arXiv:2205.10350*, 2022.

Sehoon Kim, Karttikeya Mangalam, Jitendra Malik, Michael W Mahoney, Amir Gholami, and Kurt Keutzer. Big little transformer decoder. *arXiv preprint arXiv:2302.07863*, 2023.

Yin Tat Lee and Aaron Sidford. Path finding methods for linear programming: Solving linear programs in o (vrank) iterations and faster algorithms for maximum flow. In *2014 IEEE 55th Annual Symposium on Foundations of Computer Science*, pages 424–433. IEEE, 2014.

Yaniv Leviathan, Matan Kalman, and Yossi Matias. Fast inference from transformers via speculative decoding. *International Conference on Machine Learning*, 2023.

Nimrod Megiddo et al. *On the complexity of linear programming*. Citeseer, 1986.

Ofir Pele and Michael Werman. Fast and robust earth mover's distances. In *2009 IEEE 12th international conference on computer vision*, pages 460–467. IEEE, 2009.

James Renegar. A polynomial-time algorithm, based on newton's method, for linear programming. *Mathematical programming*, 40(1-3):59–93, 1988.

Mitchell Stern, Noam Shazeer, and Jakob Uszkoreit. Blockwise parallel decoding for deep autoregressive models. *Advances in Neural Information Processing Systems*, 31, 2018.

Ziteng Sun, Ananda Theertha Suresh, Jae Hun Ro, Ahmad Beirami, Himanshu Jain, Felix Yu, Michael Riley, and Sanjiv Kumar. Spectr: Fast speculative decoding via optimal transport. In *Workshop on Efficient Systems for Foundation Models @ ICML2023*, 2023. URL `https://openreview.net/forum?id=d0mGsaheuT`.

Romal Thoppilan, Daniel De Freitas, Jamie Hall, Noam Shazeer, Apoorv Kulshreshtha, Heng-Tze Cheng, Alicia Jin, Taylor Bos, Leslie Baker, Yu Du, et al. Lamda: Language models for dialog applications. *arXiv preprint arXiv:2201.08239*, 2022.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. LLaMA: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*, 2023.

Cédric Villani. *Topics in optimal transportation*, volume 58. American Mathematical Soc., 2021.

Cédric Villani et al. *Optimal transport: old and new*, volume 338. Springer, 2009.

Nan Yang, Tao Ge, Liang Wang, Binxing Jiao, Daxin Jiang, Linjun Yang, Rangan Majumder, and Furu Wei. Inference with reference: Lossless acceleration of large language models. *arXiv preprint arXiv:2304.04487*, 2023a.

Seongjun Yang, Gibbeum Lee, Jaewoong Cho, Dimitris Papailiopoulos, and Kangwook Lee. Predictive pipelined decoding: A compute-latency trade-off for exact llm decoding. *arXiv preprint arXiv:2307.05908*, 2023b.

A Deferred derivations from the main text

A.1 Derivation of the optimization problem (OPT$_k$)

In this section, we provide the precise derivation of the optimization problem (OPT$_k$). We first compute the probability that the algorithm accepts a draft sample, namely, $\mathbb{P}\left(\text{accept},\ y = x\right) = \sum_{i=1}^{k} \mathbb{P}\left(\text{accept at } i\text{th sample},\ y = x\right)$. Let's compute for each $i = 1, 2, \ldots, k$:

$$\mathbb{P}\left(\text{accept at } i\text{th sample},\ y = x\right) = \begin{cases} \alpha_i \cdot q(x) \cdot \prod_{j=1}^{i-1}(1 - \beta_{j;\alpha_j}), & \text{if} \quad x \in \Omega_i\,, \\ p(x) \cdot \prod_{j=1}^{i-1}(1 - \beta_{j;\alpha_j}), & \text{if} \quad x \in \Omega_i^c\,. \end{cases}$$

Using this calculation, we compute the probability that Algorithm 1 accepts the draft sample $x \in \Omega$:

$$\mathbb{P}\left(\text{accept},\ y = x\right) = \sum_{i=1}^{k} \left[\left(\mathbb{1}\left\{x \in \Omega_i^c\right\} \cdot p(x) + \mathbb{1}\left\{x \in \Omega_i\right\} \cdot q(x)\alpha_i\right) \cdot \prod_{j=1}^{i-1}(1 - \beta_{j;\alpha_j}) \right]$$

In order to ensure that this probability is smaller than $q(x)$, we hence need the constraint:

$$\sum_{i=1}^{k} \left[\left(\mathbb{1}\left\{x \in \Omega_i^c\right\} \cdot p(x) + \mathbb{1}\left\{x \in \Omega_i\right\} \cdot q(x)\alpha_i\right) \cdot \prod_{j=1}^{i-1}(1 - \beta_{j;\alpha_j}) \right] \le q(x), \quad \forall x \in \Omega$$

Next, we also need a constraint that for each alphabet x in Ω_i, α_i can't be scaled up too much; the probability with which we accept the token cannot go beyond 1. For this constraint, we focus on the case $\Omega_i \neq \emptyset$.

- For each alphabet $x \in \Omega_i$, the token acceptance probability is given as $\alpha_i \cdot \frac{q(x)}{p(x)}$. This has to be less than equal to 1, which introduces the following constraint:

$$\forall i = 1, 2 \ldots, k, \quad 0 \le \alpha_i \le \frac{p(x)}{q(x)} \quad \forall x \in \Omega_i\,,$$

  or equivalently, we have

$$\forall i = 1, 2 \ldots, k, \quad 0 \le \alpha_i \le \min_{x \in \Omega_i} \frac{p(x)}{q(x)}\,.$$

Note that when $\Omega_i = \emptyset$, then α_i is not relevant anymore, so we set α_i to be 0 and exclude α_i from the optimization variables. This completes the derivation of (OPT$_k$).

A.2 Proof of Theorem 2

Before we get into the proof, recall the definitions. First, we have $\beta_{i;\alpha} = p(\Omega_i^c) + \alpha \cdot q(\Omega_i)$ and

$$\boxed{u_i = \prod_{j=1}^{i} \Big(p(\Omega_i) - \alpha_i \cdot q(\Omega_i)\Big) \qquad \text{for} \quad i = 1, 2, \ldots, k\,,} \tag{6}$$

and $u_0 := 1$. With this substitution, we will show that (OPT$_k$) becomes the following linear program:

$$\boxed{\begin{cases} \underline{\textbf{Optimiazation variables: } u_i \ \text{ for } i = 1, 2, \ldots, k \text{ s.t. } \Omega_i \neq \emptyset.} \\ \text{(for } i \text{ s.t. } \Omega_i = \emptyset\text{, we set } u_i = u_{i-1} \text{ and exclude it from optimization variables)} \\ \min_{u_1, \ldots, u_k} u_k \\ \text{s.t. } \sum_{i=1}^{k} f_i(x) \le q(x), \quad \forall x \in \Omega \\ \qquad \text{where } f_i(x) := \begin{cases} q(x)\left(\frac{p(\Omega_i)}{q(\Omega_i)} u_{i-1} - \frac{1}{q(\Omega_i)} u_i\right) & \text{if } x \in \Omega_i \\ p(x) u_{i-1} & \text{otherwise.} \end{cases} \\ (p(\Omega_i) - \gamma_i^{\min} \cdot q(\Omega_i)) \cdot u_{i-1} \le u_i \le p(\Omega_i) \cdot u_{i-1}, \quad \forall i \text{ s.t. } \Omega_i \neq \emptyset. \end{cases}} \tag{LP$_k$}$$

Here, $\gamma_i^{\min} := \min_{x \in \Omega_i} \frac{p(x)}{q(x)}$ for $i = 1, 2, \ldots, k$.

We begin by deriving the conversion formula between α_i's and u_i's. From (6), it follows that

$$\frac{u_i}{u_{i-1}} = 1 - p(\Omega_i^c) - \alpha_i \cdot q(\Omega_i) = p(\Omega_i) - \alpha_i \cdot q(\Omega_i)\,.$$

Hence, whenever $\Omega_i \neq \emptyset$, we have the following conversion formula between α_i's and u_i's.

$$\boxed{\alpha_i = \frac{p(\Omega_i)}{q(\Omega_i)} - \frac{1}{q(\Omega_i)}\frac{u_i}{u_{i-1}}, \qquad \text{for } i = 1, 2, \dots, k \text{ s.t. } \Omega_i \neq \emptyset.}$$

This in particular implies that for i s.t. $\Omega_i \neq \emptyset$,

$$\begin{aligned} r_i(x) &= \mathbb{1}\{x \in \Omega_i^c\} \cdot p(x) + \mathbb{1}\{x \in \Omega_i\} \cdot q(x)\alpha_i \\ &= \mathbb{1}\{x \in \Omega_i^c\} \cdot p(x) + \mathbb{1}\{x \in \Omega_i\} \cdot q(x)\left(\frac{p(\Omega_i)}{q(\Omega_i)} - \frac{1}{q(\Omega_i)}\frac{u_i}{u_{i-1}}\right), \end{aligned}$$

and hence the "left hand side" of the first constraint in (OPT$_k$) becomes:

$$\begin{aligned} \sum_{i=1}^{k}\left[r_i(x) \cdot \prod_{j=1}^{i-1}(1-\beta_{j;\alpha_j})\right] &= \sum_{i=1}^{k}\left[r_i(x) \cdot u_{i-1}\right] \\ &= \sum_{i=1}^{k}\left[\mathbb{1}\{x \in \Omega_i^c\} \cdot p(x)u_{i-1} + \mathbb{1}\{x \in \Omega_i\} \cdot q(x)\left(\frac{p(\Omega_i)}{q(\Omega_i)}u_{i-1} - \frac{1}{q(\Omega_i)}u_i\right)\right]. \end{aligned}$$

Now using the notation $\gamma_i^{\min} := \min_{x\in\Omega_i}\frac{p(x)}{q(x)}$ for $i = 1, 2, \dots, k$, the last constraint of (OPT$_k$) becomes:

$$0 \leq \alpha_i \leq \gamma_i^{\min}, \quad \forall i \text{ s.t. } \Omega_i \neq \emptyset.$$

Hence, for i s.t. $\Omega_i \neq \emptyset$, using the relation $\alpha_i = \frac{p(\Omega_i)}{q(\Omega_i)} - \frac{1}{q(\Omega_i)}\frac{u_i}{u_{i-1}}$, it follows that

$$0 \leq \alpha_i \leq \gamma_i^{\min} \quad \iff \quad (p(\Omega_i) - \gamma_i^{\min}q(\Omega_i)) \cdot u_{i-1} \leq u_i \leq p(\Omega_i) \cdot u_{i-1}\,.$$

This completes the derivation of (LP$_k$).

A.3 Derivation of the optimization problem (3)

Recall that we have the following two additional constraints:

- we choose subsets Ω_i's to be same, i.e., $\Omega_i \equiv \widehat{\Omega}$ for some $\widehat{\Omega}$, and
- $\alpha_i \equiv \widehat{\alpha}$ for all $i = 1, 2, \dots, k$.

With these additional constraints, we have the following simplifications:

- The cost simplifies to $(1-\beta_{\widehat{\alpha}})^k$.
- As for the first constraint, we only need to maintain two constraints depending on whether $x \in \widehat{\Omega}$ or not:
  - When $x \in \widehat{\Omega}$, $r_i(x) = \alpha q(x)$, and hence the LHS of the constraint becomes

    $$\sum_{i=1}^{k}\left[r_i(x) \cdot \prod_{j=1}^{i-1}(1-\beta_{j;\alpha_j})\right] = \alpha q(x) \cdot \sum_{i=1}^{k}(1-\beta_\alpha)^{i-1} = \alpha q(x) \cdot \frac{1-(1-\beta_\alpha)^k}{\beta_\alpha}\,.$$

    Thus, the first constraint in this case becomes

    $$\alpha q(x) \cdot \frac{1-(1-\beta_\alpha)^k}{\beta_\alpha} \leq q(x) \quad \iff \quad 1-(1-\beta_\alpha)^k \leq \alpha^{-1} \cdot \beta_\alpha$$

  - When $x \notin \widehat{\Omega}$, $r_i(x) = p(x)$, and hence the LHS of the constraint becomes

    $$p(x) \cdot \frac{1-(1-\beta_\alpha)^k}{\beta_\alpha} \leq q(x) \quad \iff \quad 1-(1-\beta_\alpha)^k \leq \min_{x\in\Omega^c}\frac{q(x)}{p(x)} \cdot \beta_\alpha\,.$$

- Lastly, the second constraint becomes $0 \leq \alpha \leq \min_{x\in\widehat{\Omega}}\frac{p(x)}{q(x)}$.

Combining all these simplifications, we obtain (3).

A.4 Derivation of the optimization problem (4)

Recall that relative to the optimization problem (4), we set $\widehat{\Omega} = \{x \in \Omega \ : \ p(x) \geq \widehat{\alpha} \cdot q(x)\}$, and hence, the token-wise acceptance probability is

$$\widehat{\beta}_{\widehat{\alpha}} := \sum_{x \in \Omega} \min\{p(x), \ \widehat{\alpha} \cdot q(x)\} \,.$$

In particular, some constraints in (3) become redundant:

- By the choice of $\widehat{\Omega}$, we have $\widehat{\alpha}^{-1} < \frac{q(x)}{p(x)}$ for all $x \in \widehat{\Omega}^c$, which implies that $\widehat{\alpha}^{-1} < \min_{x \in \widehat{\Omega}^c} \frac{q(x)}{p(x)}$. Hence, in (3), the first constraint implies the second constraint, and we can omit the second constraint.
- Moreover, we have $p(x) \geq \widehat{\alpha} \cdot q(x)$ for all $x \in \widehat{\Omega}$, which implies that $\widehat{\alpha} \leq \min_{x \in \widehat{\Omega}} \frac{p(x)}{q(x)}$ holds.

Hence, with this simplification, the optimization problem (3) reduces to (4).

B Algorithm derivations for general case

B.1 Derivation of the optimization problem (general)

Questions:

- How to initialize Ω_i's (when p_i's are different)?

$$\textbf{Transport cost:} \quad \prod_{i=1}^{k} (1 - \beta_{i;\alpha_i}) \,, \quad \text{where} \quad \boxed{\beta_{i;\alpha_i} := p_i(\Omega_i^c) + \alpha_i \cdot q(\Omega_i)} \,.$$

This is because we accept x_i with probability 1 if $x_i \in \Omega_i^c$ and with probability $\alpha_i \cdot \frac{q(x_i)}{p(x_i)}$ if $x_i \in \Omega_i$.

$$\boxed{\begin{cases} \underline{\textbf{Optimization variables: } \alpha_i \ \text{ for } i = 1, 2, \ldots, k \text{ s.t. } \Omega_i \neq \emptyset.} \\ \text{(for } i \text{ s.t. } \Omega_i = \emptyset \text{, we set } \alpha_i = 0 \text{ and exclude it from optimization variables)} \\ \min_{\alpha_i} \prod_{i=1}^{k} (1 - \beta_{i;\alpha_i}) \qquad (\triangleright \text{ transport cost.}) \\ \text{s.t.} \sum_{i=1}^{k} \left[r_i(x) \cdot \prod_{j=1}^{i-1} (1 - \beta_{j;\alpha_j}) \right] \leq q(x), \quad \forall x \in \Omega \quad (\triangleright \text{ accept probability} \leq q(x)) \\ \qquad \text{where } \ r_i(x) := \begin{cases} q(x)\alpha_i & \text{if } x \in \Omega_i \\ p_i(x) & \text{otherwise.} \end{cases} \\ \quad 0 \leq \alpha_i \leq \min_{x \in \Omega_i} \frac{p_i(x)}{q(x)}, \quad \forall i \text{ s.t. } \Omega_i \neq \emptyset. \qquad (\triangleright \text{ accept} \leq p(x)) \end{cases}}$$

We first compute the probability that the algorithm accepts a draft sample, namely, $\mathbb{P}\left(\text{accept}, \ y = x\right) = \sum_{i=1}^{k} \mathbb{P}\left(\text{accept at } i\text{th sample}, \ y = x\right)$. Let's compute for each $i = 1, 2, \ldots, k$:

$$\mathbb{P}\left(\text{accept at } i\text{th sample}, \ y = x\right) = \begin{cases} \alpha_i \cdot q(x) \cdot \prod_{j=1}^{i-1} (1 - \beta_{j;\alpha_j}) \,, & \text{if} \quad x \in \Omega_i \,, \\ p_i(x) \cdot \prod_{j=1}^{i-1} (1 - \beta_{j;\alpha_j}) \,, & \text{if} \quad x \in \Omega_i^c \,. \end{cases}$$

Using this calculation, we compute the probability that Algorithm 1 accepts the draft sample $x \in \Omega$:

$$\mathbb{P}\left(\text{accept}, \ y = x\right) = \sum_{i=1}^{k} \left[\left(\mathbb{1}\left\{x \in \Omega_i^c\right\} \cdot p_i(x) + \mathbb{1}\left\{x \in \Omega_i\right\} \cdot q(x)\alpha_i \right) \cdot \prod_{j=1}^{i-1} (1 - \beta_{j;\alpha_j}) \right]$$

In order to ensure that this probability is smaller than $q(x)$, we hence need the constraint:

$$\sum_{i=1}^{k} \left[\left(\mathbb{1}\left\{x \in \Omega_i^c\right\} \cdot p_i(x) + \mathbb{1}\left\{x \in \Omega_i\right\} \cdot q(x)\alpha_i \right) \cdot \prod_{j=1}^{i-1} (1 - \beta_{j;\alpha_j}) \right] \leq q(x), \quad \forall x \in \Omega$$

Next, we also need a constraint that for each alphabet x in Ω_i, α_i can't be scaled up too much; the probability with which we accept the token cannot go beyond 1. For this constraint, we focus on the case $\Omega_i \neq \emptyset$.

- For each alphabet $x \in \Omega_i$, the token acceptance probability is given as $\alpha_i \cdot \frac{q(x)}{p_i(x)}$. This has to be less than equal to 1, which introduces the following constraint:
$$\forall i = 1, 2 \ldots, k, \quad 0 \le \alpha_i \le \frac{p_i(x)}{q(x)} \quad \forall x \in \Omega_i \,,$$
or equivalently, we have
$$\forall i = 1, 2 \ldots, k, \quad 0 \le \alpha_i \le \min_{x \in \Omega_i} \frac{p_i(x)}{q(x)} \,.$$

Note that when $\Omega_i = \emptyset$, then α_i is not relevant anymore, so we set α_i to be 0 and exclude α_i from the optimization variables. This completes the derivation of (OPT$_k$).

B.2 Proof of Theorem 2

Before we get into the proof, recall the definitions. First, we have $\beta_{i;\alpha} = p_i(\Omega_i^c) + \alpha \cdot q(\Omega_i)$ and
$$\boxed{u_i = \prod_{j=1}^{i} \Big(p_i(\Omega_i) - \alpha_i \cdot q(\Omega_i) \Big) \qquad \text{for} \quad i = 1, 2, \ldots, k \,,}$$
and $u_0 := 1$. With this substitution, we will show that (OPT$_k$) becomes the following linear program:

$$\boxed{\begin{cases} \underline{\textbf{Optimiazation variables: } u_i \ \text{ for } i = 1, 2, \ldots, k \text{ s.t. } \Omega_i \neq \emptyset}. \\ \text{(for } i \text{ s.t. } \Omega_i = \emptyset \text{, we set } u_i = u_{i-1} \text{ and exclude it from optimization variables)} \\ \min_{u_1, \ldots, u_k} u_k \\ \text{s.t. } \sum_{i=1}^{k} f_i(x) \le q(x), \quad \forall x \in \Omega \\ \qquad \text{where } f_i(x) := \begin{cases} q(x) \left(\frac{p_i(\Omega_i)}{q(\Omega_i)} u_{i-1} - \frac{1}{q(\Omega_i)} u_i \right) & \text{if } x \in \Omega_i \\ p_i(x) u_{i-1} & \text{otherwise.} \end{cases} \\ (p_i(\Omega_i) - \gamma_i^{\min} \cdot q(\Omega_i)) \cdot u_{i-1} \le u_i \le p_i(\Omega_i) \cdot u_{i-1}, \quad \forall i \text{ s.t. } \Omega_i \neq \emptyset. \end{cases}}$$

Here, $\gamma_i^{\min} := \min_{x \in \Omega_i} \frac{p_i(x)}{q(x)}$ for $i = 1, 2, \ldots, k$.

We begin by deriving the conversion formula between α_i's and u_i's. From (6), it follows that
$$\frac{u_i}{u_{i-1}} = 1 - p_i(\Omega_i^c) - \alpha_i \cdot q(\Omega_i) = p_i(\Omega_i) - \alpha_i \cdot q(\Omega_i) \,.$$
Hence, whenever $\Omega_i \neq \emptyset$, we have the following conversion formula between α_i's and u_i's.
$$\boxed{\alpha_i = \frac{p_i(\Omega_i)}{q(\Omega_i)} - \frac{1}{q(\Omega_i)} \frac{u_i}{u_{i-1}}, \quad \text{for } i = 1, 2, \ldots, k \text{ s.t. } \Omega_i \neq \emptyset.}$$
This in particular implies that for i s.t. $\Omega_i \neq \emptyset$,
$$\begin{aligned} r_i(x) &= \mathbb{1}\{x \in \Omega_i^c\} \cdot p_i(x) + \mathbb{1}\{x \in \Omega_i\} \cdot q(x) \alpha_i \\ &= \mathbb{1}\{x \in \Omega_i^c\} \cdot p_i(x) + \mathbb{1}\{x \in \Omega_i\} \cdot q(x) \left(\frac{p_i(\Omega_i)}{q(\Omega_i)} - \frac{1}{q(\Omega_i)} \frac{u_i}{u_{i-1}} \right) , \end{aligned}$$
and hence the "left hand side" of the first constraint in (OPT$_k$) becomes:
$$\begin{aligned} \sum_{i=1}^{k} \left[r_i(x) \cdot \prod_{j=1}^{i-1} (1 - \beta_{j;\alpha_j}) \right] &= \sum_{i=1}^{k} [r_i(x) \cdot u_{i-1}] \\ &= \sum_{i=1}^{k} \left[\mathbb{1}\{x \in \Omega_i^c\} \cdot p_i(x) u_{i-1} + \mathbb{1}\{x \in \Omega_i\} \cdot q(x) \left(\frac{p_i(\Omega_i)}{q(\Omega_i)} u_{i-1} - \frac{1}{q(\Omega_i)} u_i \right) \right] . \end{aligned}$$
Now using the notation $\gamma_i^{\min} := \min_{x \in \Omega_i} \frac{p_i(x)}{q(x)}$ for $i = 1, 2, \ldots, k$, the last constraint of (OPT$_k$) becomes:
$$0 \le \alpha_i \le \gamma_i^{\min}, \quad \forall i \text{ s.t. } \Omega_i \neq \emptyset.$$
Hence, for i s.t. $\Omega_i \neq \emptyset$, using the relation $\alpha_i = \frac{p_i(\Omega_i)}{q(\Omega_i)} - \frac{1}{q(\Omega_i)} \frac{u_i}{u_{i-1}}$, it follows that
$$0 \le \alpha_i \le \gamma_i^{\min} \quad \Longleftrightarrow \quad (p_i(\Omega_i) - \gamma_i^{\min} q(\Omega_i)) \cdot u_{i-1} \le u_i \le p_i(\Omega_i) \cdot u_{i-1} \,.$$
This completes the derivation of (LP$_k$).