

SpecDec

Avi Arya & Shreya Singh & Theo Urban
10-423/623 Generative AI Course Project

November 24, 2025

1 Introduction

As large language models have continued to grow, inference latency has become a bigger issue. Improvements in inference latency can significantly affect workflows, and having lower latency may result in faster systems and feedback loops. A recent proposal to address the issue of inference latency is Speculative Decoding, a method of using smaller models to generate a response, then using a larger language model to attempt to either accept or disregard the smaller language model's result. If the larger language model does not accept the smaller language model's result, then the larger language model generates its own result fully. However, the hope is that, oftentimes, the smaller language model and the larger language model will result, resulting in improved latency.

In this project, we hope to apply finetuning to Speculative Decoding to create a faster model for code generation. Specifically, we will use The Stack, a dataset of code files, to finetune minGPT, then use this as the small model in our speculative decoding model, eventually comparing this model to speculative decoding without this additional finetuning.

2 Dataset / Task

For this project, we will use The Stack, a large dataset of source code developed by the BigCode project. The dataset is one of the most extensive collections of code, containing 6.4 terabytes spanning 358 programming languages.

Our use of this dataset will be to finetune our draft model, minGPT 2. We intend to create a training split with code from higher resource languages (Python, C++, Javascript, etc) and perhaps some lower resource languages as well. The lower resource languages can be used to evaluate the finetuning process for its impact on cross lingual performance disparities.

The test set will comprise data from The Stack not used during finetuning. The set will have several code completion prompts where we will use a function signature, comment block, or partial line of code as input.

Using this will allow us to compare the base model and finetuned model fairly on their code generation ability.

The primary task at hand is domain-specific accelerated code completion. In other words, given an input context s (a string of source code), we want to generate a plausible and syntactically correct continuation of the code using a speculative decoding framework to potentially accelerate the rate of output.

We define our two primary models as follows:

1. **Verifier Model** (P_θ) represents the probability distribution $p(x | s)$ over the next token x given some context s . For this project, we will use the standard GPT 2 model as P_θ .
2. **Draft Model** (Q_ϕ) represents the probability distribution $q(x | s)$. For this project, we will use minGPT 2 which is smaller and faster than the verifier model.

Our project will compare two distinct draft models:

1. **Baseline Draft Model** (Q_ϕ) which is the unmodified minGPT 2 model.
2. **Finetuned Draft Model** (Q'_ϕ) which is the minGPT 2 model after it has been finetuned on the training split from The Stack.

The task involves providing the system with a context string s and sampling a fixed length sequence of tokens from the draft model. These tokens are then validated by the verifier in a single pass.

To quantitatively evaluate our model's success, we will look at its acceptance rate, speedup over the baseline model, and crosslingual disparity. These metrics are detailed more in a later section.

3 Related Work

Reducing inference times in LLMs is a significant area of past and current research. Modern LLMs have high associated computation and memory bandwidth costs due to standard decoding, which in turn provide a bottleneck for many real world applications. Our project

78 aims to build on the core technique of speculative de-
79 coding, first presented in ?. The paper demonstrated
80 that using a smaller draft model to generate tokens and
81 then passing them to a larger verifier model could pro-
82 vide speedups in generations without sacrificing gen-
83 eration quality. Then ? introduced a similar method
84 called speculative sampling that further solidified the
85 technique’s potential at reducing overhead. Our base-
86 line model will be a direct implementation of the algo-
87 rithms presented in these two papers.

88 The reason for our finetuned model, however, is that
89 the benefits of speculative decoding are not uniformly
90 distributed. This issue is raised and explored by ?, who
91 point out performance disparities in speculative decod-
92 ing. They show that it accelerates generation primar-
93 ily for high resource languages and predictable text se-
94 quences. To this extent, we aim to evaluate our model
95 on crosslingual performance as well. We will adopt the
96 disparity metric $U(T)$ that they introduced to measure
97 our approach’s success.

98 Our central hypothesis is that specializing the draft
99 model to a particular task can improve its performance,
100 which is well supported in recent literature. An exam-
101 ple is using knowledge distillation, which is done by
102 ? via DistillSpec. Here, the draft model is explicitly
103 trained on the verifier’s output distribution. However,
104 our approach diverges slightly to finetune on a special-
105 ized corpus which can be seen as domain-specific dis-
106 tillation.

107 Finally, another consideration that we have to make
108 is alignment. As shown by ?, it is imperative that the
109 verifier and draft model are aligned, especially when
110 the larger model is finetuned in some way (i.e. for chat)
111 because a generic draft model will perform quite poorly
112 in such cases. The solution they use is to finetune the
113 draft model using the verifier model as a ”teacher”.
114 While our project is not specifically aligning the draft
115 model with the verifier, it does aim to align the draft
116 model to the domain we are testing on.

117 4 Approach

118 In this project, our overall goal is to determine the im-
119 pact of finetuning on speculative decoding models. Our
120 baseline model is a speculative decoding model that
121 uses minGPT 2 as the smaller model and GPT 2 as
122 our larger, verifier model. We first must implement
123 speculative decoding using these two models, and run
124 experiments to determine the speed at which this gen-
125 erates its output on a few sample prompts. Next, we
126 will finetune minGPT 2 on The Stack, a large dataset
127 from Hugging Face containing code files in 300 differ-
128 ent languages. By using this dataset, we hope to better
129 inform our smaller model on this particular task. We

will then reimplement speculative decoding using this
130 finetuned model alongside GPT2, and will evaluate the
131 inference latency and results on our given prompts.
132

133 5 Experiments

134 5.1 Experimental Setup

We will conduct all of our experiences on an A100 ob-
135 tained through Google Colab Pro. We use 12B Pythia
136 by ElutherAI as our Verifier model and 70M Pythia as
137 our Draft model, following from the initial experiments
138 below. All experiments are conducted with a fixed seed
139 of 42 to ensure consistency across runs.
140

For our main set of experiemnts, we fix $\gamma = ???$ to
141 tokens to standardize across runs with a reasonable bal-
142 ance between draft and verification speed.
143

144 5.2 Baseline Performance

Our first experiment establishes the baseline perfor-
145 mance of a few different models and on different tasks
146 - natural language completions and a few coding tasks
147 in different languages outlined in the appendix.
148

Model	Completion	Acce. Rate	Speedup
Qwen	NL	38.3%	0.60x
	Code (c)	67.9%	0.50x
	Code (go)	67.0%	0.58x
	Code (python)	69.4%	0.63x
	Code (rust)	69.3%	0.58x
GPT-2	NL	36.5%	1.15x
	Code (c)	63.6%	1.62x
	Code (go)	55.4%	1.51x
	Code (python)	65.0%	1.64x
	Code (rust)	57.2%	1.50x
Pythia	NL	34.7%	1.26x
	Code (c)	54.9%	1.17x
	Code (go)	54.1%	1.18x
	Code (python)	55.0%	1.23x
	Code (rust)	56.2%	1.26x

Baseline speculative decoding performance across
150 different model pairs and tasks. Qwen: (Qwen2.5-
151 7B+Qwen2.5-0.5B), GPT-2: (gpt2-large+distilgpt2),
152 Pythia: (pythia-12b+pythia-70m).
153

154 6 Proposed Experiments

In this section, we outline the experiments designed
155 to evaluate how domain-specific finetuning of the draft
156 model influences speculative decoding effectiveness.
157

158 **6.1 RQ1: Does Domain-Specific Finetun-
159 ing Improve Speculative Decoding Perfor-
160 mance?**

161 We compare two systems:

- 162 1. **Baseline SpecDec**: GPT2 (verifier) + unfinetuned
163 minGPT2 (draft)
- 164 2. **Finetuned SpecDec**: GPT2 (verifier) + minGPT2
165 finetuned on The Stack

166 Both models will be evaluated on:

- 167 • **In-domain**: Code completions across Python,
168 C++, Go, Rust, and others
- 169 • **Out-of-domain**: Natural language tasks (news,
170 stories, instructions)

171 Each prompt set will generate 128-token continua-
172 tions with fixed draft length γ .

Model Pair	Domain	Accept Rate	Speedup
Baseline	Code (Py)	???	???
Finetuned	Code (Py)	???	???
Baseline	NL	???	???
Finetuned	NL	???	???

Table 1: Effect of draft model finetuning on acceptance rate and decoding speed.

173 **Expected Outcome.** Finetuning should increase ac-
174 ceptance rate and speedup on code tasks, while main-
175 taining comparable performance on natural language
176 prompts.

177 **6.2 RQ2: Does Finetuning Improve Perfor-
178 mance for Low-Resource Languages?**

179 **Goal.** Determine whether finetuning disproportionately helps high-resource languages or narrows cross-
180 lingual disparities.

182 **Experimental Setup.** We construct a language-
183 stratified test set from The Stack with three buckets:**High-resource**: Python, C++, Java, **Medium**: Lua,
184 Haskell, Ruby, and **Low-resource**: Zig, Elixir, OCaml

186 For each language bucket, we measure (1) accept-
187 ance rate, (2) end-to-end speedup, and (3) disparity
188 metric $U(T)$ from ? before and after finetuning.

Group	Base $U(T)$	Tuned $U(T)$	Δ Disparity
High	???	???	???
Medium	???	???	???
Low	???	???	???

Table 2: Change in cross-lingual performance disparity after finetuning.

189 **Expected Outcome.** Finetuning should reduce per-
190 formance disparity across languages by improving
191 draft alignment with code structure.

192 **6.3 RQ3: How Does Draft Length γ Affect
193 Speedup Under Fixed Compute?**

194 **Goal.** Characterize how varying γ impacts acceptance
195 rate, rejection bursts, and overall speedup.

196 **Experimental Setup.** Using the finetuned draft
197 model, we vary $\gamma \in \{1, 2, 4, 8, 16, 32\}$. For each γ , we
198 record (1) Acceptance rate, (2) rejection burst statistics,
199 and (3) real-time decoding speedup.

γ	Avg. Burst Len.	$P(\text{Burst} > 10)$	Speedup	Accept Rate
1	???	???	???	???
4	???	???	???	???
16	???	???	???	???

Table 3: Effect of draft length on rejection patterns and speedup.

200 **Expected Outcome.** We hypothesize a unimodal
201 trend where moderate values of γ maximize speedup
202 by avoiding excessive rejections while still leveraging
203 draft efficiency.

204 **7 Compute**

205 **8 Plan**

206 We can break this project in to three sequential steps:

- 207 1. Set up speculative decoding using minGPT and
208 GPT2 as paired models
- 209 2. Create metrics
- 210 3. Finetune minGPT and GPT2 on The Stack
- 211 4. Evaluate metrics pre and post finetune

212 We intend to complete through the metrics creation
213 and initial testing by the executive summary deadline.
214 The work is highly sequential so it will involve a lot
215 of pair programming and back-and-forth. However, we
216 will assign directly responsible members for items 1, 2,
217 and 3, to Avi, Shreya, and Theo in that order.