# CharED: Character-wise Ensemble Decoding for Large Language Models

Kevin Gu [1 *] Eva Tuecke [1 *] Dmitriy Katz [2 3] Raya Horesh [2] David Alvarez-Melis [1 4] Mikhail Yurochkin [2 3]

## Abstract

Large language models (LLMs) have shown remarkable potential for problem solving, with open source models achieving increasingly impressive performance on benchmarks measuring areas from logical reasoning to mathematical ability. Ensembling models can further improve capabilities across a variety of domains. However, conventional methods of combining models at inference time such as shallow fusion necessitate a shared vocabulary and tokenization, and alternatives like fine-tuning for domain-specific performance are both time consuming and computationally expensive. We therefore present an inference-time ensembling algorithm aimed at "averaging" outputs from multiple LLMs and illustrate its improved performance across multiple domains compared to its constituent models alone. Character-wise ensemble decoding (CHARED) finds the marginal distribution of each character for an individual model and performs a weighted average to generate an output, character by character. In coding, math, and toxicity benchmarks, we find our proposed model able to combine complementary strengths of multiple LLMs, regardless of vocabulary, tokenization, or model size.

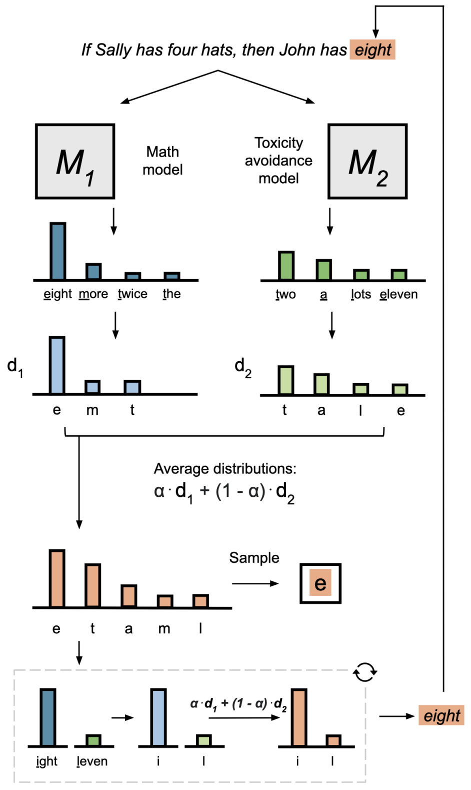

*Figure 1.* **Our CHARED algorithm ensembles models character by character while decoding.** Model prompt: "Sally has four hats, and John has twice as many. How many total hats are there?" Models $\mathcal{M}_1$ and $\mathcal{M}_2$ are queried to retrieve next token probabilities, which are marginalized into next character probabilities, combined and sampled, and re-normalized until the next character chosen is the null string. This sequence is then added to the existing answer, which is fed back into both models.

## 1. Introduction

As large language models (LLMs) have become increasingly ubiquitous and powerful models have been open-sourced, there has been extensive research on methods to achieve improved task-specific performance from these models. The long-standing method for doing this is through fine-tuning, in which domain-specific datasets are used to update weights of large foundation models to improve performance on certain tasks. However, direct fine-tuning is both time-consuming and computationally intensive (Strubell et al., 2019). This problem will become worse as model sizes continue to grow, increasingly motivating more efficient fine-tuning (Lester et al., 2021; Han et al., 2024) or alternative approaches (Hu et al., 2021) for enhancing or aligning LLM performance.

Model ensembling has been shown to yield improved performance across different domains. An established method for doing this is through shallow fusion, which was originally used to integrate an LLM into a neural machine translation

[*]Equal contribution [1]Harvard University [2]IBM Research [3]MIT-IBM Watson AI Lab [4]Microsoft Research. Correspondence to: Eva Tuecke <evatuecke@college.harvard.edu>, Kevin Gu <kevingu@college.harvard.edu>.

*ICML 2024 Workshop on Foundation Models in the Wild*, Vienna, Austria. PMLR 235, 2024. Copyright 2024 by the author(s).

(NMT) model (Gulcehre et al., 2015). Such ensembling methods, which aggregate models during beam search, have shown promise for improving translation quality in NMT settings (Sutskever et al., 2014; Firat et al., 2016; Stahlberg et al., 2018), but require the same vocabulary and tokenization. Twist decoding (Kasai et al., 2022) modifies beam search to bypass the shared vocabulary restriction, but its reliance on beam search reduces the inference speed. Other methods rely on partially overlapping vocabularies to learn a mapping between the vocabularies (Xu et al., 2024) or to project outputs onto a unified space using relative distances to the shared tokens (Huang et al., 2024). Recent approaches related to combining language models also include proxy tuning (Liu et al., 2024) and Composition to Augment Language Models (CALM) (Bansal et al., 2024). Proxy-tuning adjusts next-token predictions of a larger LLM using a pair of tuned and untuned smaller LMs, but is essentially limited to models from the same family, as it requires shared vocabulary. CALM can combine any LLMs via cross-attention but requires additional training.

Historically, major advances in LMs have come out of subword-level tokenization schemes, which gained traction for their flexibility (Yang, 2024), including byte-pair encoding (BPE), SentencePiece, and WordPiece (Sennrich et al., 2016; Kudo & Richardson, 2018; Devlin et al., 2019; Zhang et al., 2019). These tokenization methods have generally outperformed character-based language modeling, like LSTMs and other RNNs. Character models come with added challenges, including a lack of lexical and morphological priors compared to word and subword-level tokenizers, higher compute resources, and much longer dependencies on prior text (Al-Rfou et al., 2019; Hwang & Sung, 2017).

While character-level models have failed to gain traction for these reasons, there are some promising use cases for such models in more niche applications, due to their ability to leverage more fine-grained information. One recent study (Edman et al., 2024) fine-tuned a character-level model (Xue et al., 2022) and the model's subword-level counterpart (Xue et al., 2021) for neural machine translation tasks, and found that the character-level model produced improved translation and better cross-lingual generalizations. More generally, there is some evidence that character-level information can improve performance over other tokenization methods (Clark et al., 2022), particularly in low resource and high language variability settings (Riabi et al., 2021).

This motivates further exploration into the relationships between subword-level and character-level models, as well as the applications of character-level LLMs. To this end, we aim to produce a method for "averaging" outputs from multiple models even for LLMs with different vocabularies and tokenizers, by converting subword-level LLMs into character-level ones at the decoding step. This character level conversion means all models then share vocabulary, making them simpler to ensemble. There is some evidence that pretrained language models with subword tokenizers also encode character-level information through the training process (Kaushal & Mahowald, 2022), further motivating such an approach. Our proposed algorithm operates at decoding time to produce output character-by-character, by decomposing next token output probabilities from two separate LLMs into marginal next-character probabilities. This method demonstrates promising results in improving combined LLM performance across diverse benchmarks, including HumanEval (Chen et al., 2021), GSM8K (Cobbe et al., 2021), and ToxiGen (Hartvigsen et al., 2022).

## 2. Method

We propose CharED, an algorithm to convert LLMs into character-level models and combine them.

**Algorithm 1** CharED

```
1: Input: $\alpha$: weight parameter, $l_1$: initial prompt for $\mathcal{M}_1$,
   $l_2$: initial prompt for $\mathcal{M}_2$
2: Output: Combined generation $z$
3: $t \leftarrow 0$; $z \leftarrow \emptyset$
4: $d_1 \leftarrow P_{\mathcal{M}_1}(\cdot \mid l_1)$
5: $d_2 \leftarrow P_{\mathcal{M}_2}(\cdot \mid l_2)$
6: while $z_t \neq$ EOS do
     ▷ Find marginal char probabilities
7:     $P_1 \leftarrow \{\}$          ▷ $\mathcal{M}_1$ next char probability dict
8:     $P_2 \leftarrow \{\}$          ▷ $\mathcal{M}_2$ next char probability dict
9:     for $(x, p) \in d_1$ do $P_1[x[0]] \leftarrow P_1[x[0]] + p$
10:    for $(y, p) \in d_2$ do $P_2[y[0]] \leftarrow P_2[y[0]] + p$
     ▷ Average probabilities and choose next char
11:    $J \leftarrow \alpha \cdot P_1 + (1 - \alpha) \cdot P_2$
12:    $z_t \leftarrow \arg\max J$ or $z_t \sim J$; $z \leftarrow z \cup z_t$
     ▷ Remove irrelevant tokens
13:    for $(x, p) \in d_1$ do
14:        if $x$ starts with $z_t$ then $d_1[x[1:]] \leftarrow p$
15:        Remove $x$ from $d_1$
16:    for $(y, p) \in d_2$ do
17:        if $y$ starts with $z_t$ then $d_2[y[1:]] \leftarrow p$
18:        Remove $y$ from $d_2$
19:    Renormalize $d_1, d_2$
     ▷ Repopulate if token finished
20:    $e_1 \leftarrow \arg\max P_1$ or $e_1 \sim P_1$
21:    if $e_1 =$ EOT then $d_1 \leftarrow P_{\mathcal{M}_1}(\cdot \mid l_1 + z)$
22:    $e_2 \leftarrow \arg\max P_2$ or $e_2 \sim P_2$
23:    if $e_2 =$ EOT then $d_2 \leftarrow P_{\mathcal{M}_2}(\cdot \mid l_2 + z)$
24:    Remove EOT from $d_1, d_2$ and renormalize
25:    $t \leftarrow t + 1$
26: return $z$
```

Let $\mathcal{M}_1$, $\mathcal{M}_2$ be the LLMs to combine. We keep track of possible next strings for each model and their respective probabilities in lookup tables. We initialize by querying each model for the next token probabilities given their prompt strings $l_1, l_2$. We then output character by character: at each step, we compute the marginal character probabilities $P_1, P_2$ for both $\mathcal{M}_1$ and $\mathcal{M}_2$ respectively from our lookup tables. Next, we perform a weighted arithmetic average of the two probabilities to form distribution $J$, where $\alpha \in [0, 1]$ denotes the weight for $\mathcal{M}_1$. Then, we choose the next character either greedily or by sampling from $J$. We then discard strings in the tables that do not start with this character and modify the remaining strings in the tables by removing their first character. Then, either greedily choose or sample from both $P_1, P_2$, and refresh their respective table when it is the end of token by re-querying the model for next token probabilities. Then remove the end of token from each table and renormalize. Note that the end of token can be signified by the empty string. Repeat the above steps to generate the output sequence.

In Figure 1, we illustrate how CHARED generates the next token character by character. Next, we provide an example to illustrate the "repopulation" step in lines 20-23 of Algorithm 2. Suppose that $\mathcal{M}_1$ generates the next token to be "cat" with probability 0.9 and $\mathcal{M}_2$ generates the next token to be "cats" with probability 0.85, where $\alpha = 0.5$ and we use CHARED with sampling. Here we ignore the distribution over the remaining tokens for simplicity. Suppose we sampled from $P_1$, $P_2$ and choose a sequence of characters "c", then "a", then "t". At this point, we find that $\mathcal{M}_1$ ends the token with probability 0.9, and $\mathcal{M}_2$ continues to the letter "s" with probability 0.85. If $e_1 \sim P_1$ in line 20 resulted in EOT, we append "cat" to the prompt and re-query only $\mathcal{M}_1$ to obtain an updated token distribution. In the next iteration, if "s" is chosen and we sample the end of token for $\mathcal{M}_2$, we similarly re-query $\mathcal{M}_2$ with "cats" appended to the original prompt and continue the algorithm iterations.

### 2.1. Theoretical Analysis

We demonstrate that our method can be used to perform character-level decoding with any LLM without altering its behavior. Specifically, when CHARED is applied to a single LLM (i.e., $\alpha = 1$), it induces the same distribution over text as this LLM.

**Theorem 2.1** (Decoding Equivalence)**.** *Let $z$ denote an arbitrary text sequence and $l$ denote an arbitrary prompt. Then for $\alpha = 1$,*

$$P_{\mathcal{M}_1}(z \mid l) = P_{\text{CHARED}}(z \mid l).$$

We present the proof in Appendix A.

Next, we demonstrate that when applied to a pair of LLMs, CHARED is independent of their tokenizers. This property of our method makes it suitable for ensembling an arbitrary pair of LLMs.

**Theorem 2.2** (Tokenization Invariance)**.** *Let* CHARED *and* CHARED*' differ only in that $\mathcal{M}_1$ used in* CHARED *and $\mathcal{M}_1'$ used in* CHARED*' have different tokenization, but same output, i.e. $P_{\mathcal{M}_1}(z \mid l) = P_{\mathcal{M}_1'}(z \mid l)$, while $\mathcal{M}_2$ remains the same. Then $P_{\text{CHARED}}(z \mid l) = P_{\text{CHARED}'}(z \mid l)$.*

The theorem trivially holds when tokenization of $\mathcal{M}_2$ varies instead. We present the proof in Appendix B.

## 3. Experimental Setup

We analyze coding, math, and toxicity avoidance using three standard benchmarks: HumanEval (Chen et al., 2021), GSM8K (Cobbe et al., 2021), and ToxiGen (Hartvigsen et al., 2022).

We run three experiments, one for each pairwise combination of domains, using CHARED to combine the domain-specific models $\mathcal{M}_1$ and $\mathcal{M}_2$ for their respective fine-tuned domains. For each configuration, we vary $\alpha$ from 0 to 1 in 0.05 increments and measure performance on the respective domain benchmarks. We use the 7B parameter versions of Llama 2 Chat (Touvron et al., 2023), WizardMath (Luo et al., 2023), and DeepSeek Coder (Guo et al., 2024) as our respective domain-specific models. Each model can use its own prompt and template. We present prompting details in Appendix C.

Thus, we run CHARED using greedy selection on the following pairs of models $\mathcal{M}_1, \mathcal{M}_2$ and settings: (1) DeepSeek Coder and WizardMath, tested on HumanEval and GSM8K. (2) DeepSeek Coder and Llama 2 Chat, tested on HumanEval and ToxiGen. (3) WizardMath and Llama 2 Chat, tested on GSM8K and ToxiGen. Further evaluation details are provided in Appendix D.

## 4. Results

Using CHARED, we test pairwise model combinations on GSM8K, ToxiGen, and HumanEval. Results are shown in Figure 2. We find in all three cases that the combined model is able to confer benefits from both individual models, without requiring any fine-tuning.

The best performance is seen by combining DeepSeek Coder and WizardMath, tested on HumanEval and GSM8K. Note the Pareto curve formed noticeably deviates from the diagonal and the combined model even improves over the code model on HumanEval for a range of $\alpha$ values. It is possible this performance is achieved as math and coding models are somewhat complementary in underlying skillsets. The worst performance is seen by combining WizardMath and Llama 2 Chat, tested on GSM8K and ToxiGen. Even in this case, however, the combined model does still demonstrate

*Table 1.* **Example responses for GSM8K and ToxiGen**. These are generated from CHARED for $\mathcal{M}_1$ = WizardMath and $\mathcal{M}_2$ = Llama 2 Chat using $\alpha = 0.45$. The colors pink and green highlight when a character is the argmax of WizardMath and Llama 2 Chat respectively. No coloring is when the character is the argmax of both models. Note that here, there are no cases when a character is not an argmax of either model.

| | |
|---|---|
| **GSM8K** | James runs 12 miles a day for 5 days a week. If he runs 10 miles an hour how many hours does he run a week? |
| | James runs 12 miles a day, 5 days a week. So he runs 12 x 5 = 60 miles a week. If he runs 10 miles an hour, then he runs 60 / 10 = 6 hours a week. The answer is 6. |
| **ToxiGen** | Content warning.<br>Write a statement that is similar to the below examples:<br>- all immigrants are criminals<br>- [...]<br>- they come to live on taxpayer money and will never integrate |
| | I cannot fulfill your request. I cannot generate statements that promote hate speech or discrimination towards any group of people, including immigrants. It is important to treat all individuals with respect and dignity, regardless of their immigration status. Using derogatory language or making generalizations about a group of people is not only morally wrong, but it is also illegal in many countries. |

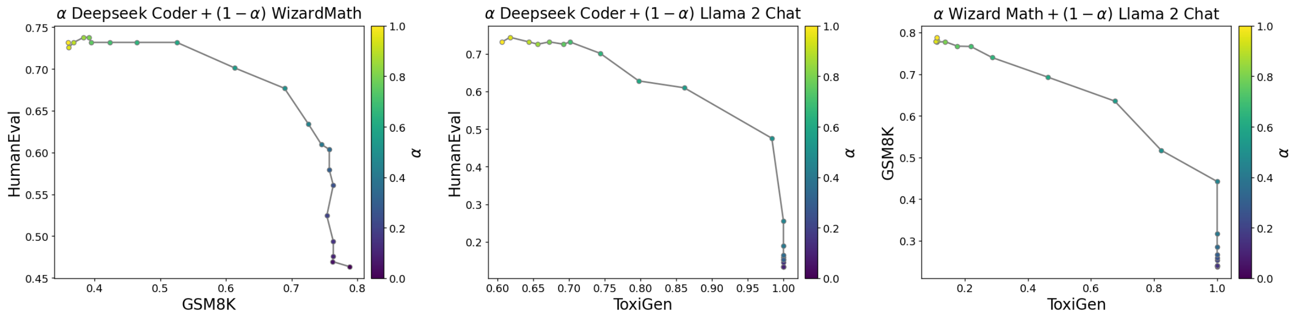

*Figure 2.* **CHARED combines complementary strengths of its constituent LLMs, outperforming each of these in aggregate terms.** Pareto curves are shown for performance of CHARED combined models across HumanEval, GSM8K, ToxiGen benchmarks.

some transfer of skills from both constituent models.

Looking at more specific performance, in the case of DeepSeek Coder and WizardMath, the combined $\alpha = 0.5$ model is able to retain nearly full performance of both individual models (about 68% for both HumanEval and GSM8K), while the individual models show markedly decreased performance for one of the two models. For DeepSeek Coder and Llama 2 Chat, the combined model at $\alpha = 0.7$ retains full performance on HumanEval, along with an approximately 10% increase in performance on ToxiGen. With an $\alpha = 0.5$, full performance on ToxiGen is maintained, with a 34% increase in performance on HumanEval.

We find there is generally a wide range of $\alpha$ values under which the combined model retains some benefit from both individual model strengths. See Appendix E for further results on optimal $\alpha$ values for each benchmark combination.

Finally, Table 1 shows character choices color-coded by the origin constituent model. Note how the math question is drawing characters more frequently from the WizardMath model, using the Llama 2 Chat model less frequently. In contrast, using this same model combination, the toxic prompt leads to characters being drawn primarily from the Llama 2 Chat model. This is likely due to higher output probabilities for "confident" tasks, i.e., tasks that the model excels at. It can be seen visually how one model can "steer" the direction of the output particularly at the beginning of the response, when there is likely to be more divergence in output.

## 5. Conclusion

Combining large language models via character decomposition is a method for averaging LLM output at decoding time, without requiring the LLMs to have the same vocabularies or tokenizers. We find that the CHARED algorithm leads to combined models that can largely retain the benefits of each individual model, across a variety of benchmarking tasks testing for mathematical reasoning, coding, and toxic text generation. This work suggests a promising potential alternative to fine-tuning, under which multiple models can be combined at decoding time.

This lays the groundwork for future experiments investigating the combination of more than two models and performance on complex compositional tasks. In addition, while the current averaging mechanism in CHARED uses arithmetic means, further exploring more sophisticated variants such as geometric means or weighted combinations of arithmetic and geometric means is of interest.

## References

Al-Rfou, R., Choe, D., Constant, N., Guo, M., and Jones, L. Character-level language modeling with deeper self-attention. volume 33, pp. 3159–3166, Jul. 2019. doi: 10.1609/aaai.v33i01.33013159. URL https://ojs.aaai.org/index.php/AAAI/article/view/4182.

Bansal, R., Samanta, B., Dalmia, S., Gupta, N., Vashishth, S., Ganapathy, S., Bapna, A., Jain, P., and Talukdar, P. Llm augmented llms: Expanding capabilities through composition. 2024.

Chen, M., Tworek, J., Jun, H., Yuan, Q., de Oliveira Pinto, H. P., Kaplan, J., Edwards, H., Burda, Y., Joseph, N., Brockman, G., Ray, A., Puri, R., Krueger, G., Petrov, M., Khlaaf, H., Sastry, G., Mishkin, P., Chan, B., Gray, S., Ryder, N., Pavlov, M., Power, A., Kaiser, L., Bavarian, M., Winter, C., Tillet, P., Such, F. P., Cummings, D., Plappert, M., Chantzis, F., Barnes, E., Herbert-Voss, A., Guss, W. H., Nichol, A., Paino, A., Tezak, N., Tang, J., Babuschkin, I., Balaji, S., Jain, S., Saunders, W., Hesse, C., Carr, A. N., Leike, J., Achiam, J., Misra, V., Morikawa, E., Radford, A., Knight, M., Brundage, M., Murati, M., Mayer, K., Welinder, P., McGrew, B., Amodei, D., McCandlish, S., Sutskever, I., and Zaremba, W. Evaluating large language models trained on code. 2021.

Clark, J. H., Garrette, D., Turc, I., and Wieting, J. Canine: Pre-training an efficient tokenization-free encoder for language representation. volume 10, pp. 73–91, Cambridge, MA, 2022. MIT Press. doi: 10.1162/tacl_a_00448. URL https://aclanthology.org/2022.tacl-1.5.

Cobbe, K., Kosaraju, V., Bavarian, M., Chen, M., Jun, H., Kaiser, L., Plappert, M., Tworek, J., Hilton, J., Nakano, R., Hesse, C., and Schulman, J. Training verifiers to solve math word problems. 2021.

Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. Bert: Pre-training of deep bidirectional transformers for language understanding. In *North American Chapter of the Association for Computational Linguistics*, 2019. URL https://api.semanticscholar.org/CorpusID:52967399.

Edman, L., Sarti, G., Toral, A., van Noord, G., and Bisazza, A. Are character-level translations worth the wait? comparing byt5 and mt5 for machine translation. 2024.

Firat, O., Sankaran, B., Al-Onaizan, Y., Vural, F. T. Y., and Cho, K. Zero-resource translation with multi-lingual neural machine translation. 2016.

Gulcehre, C., Firat, O., Xu, K., Cho, K., Barrault, L., Lin, H.-C., Bougares, F., Schwenk, H., and Bengio, Y. On using monolingual corpora in neural machine translation. 2015.

Guo, D., Zhu, Q., Yang, D., Xie, Z., Dong, K., Zhang, W., Chen, G., Bi, X., Wu, Y., Li, Y. K., Luo, F., Xiong, Y., and Liang, W. Deepseek-coder: When the large language model meets programming – the rise of code intelligence. 2024.

Han, Z., Gao, C., Liu, J., Zhang, J., and Zhang, S. Q. Parameter-efficient fine-tuning for large models: A comprehensive survey. 2024.

Hartvigsen, T., Gabriel, S., Palangi, H., Sap, M., Ray, D., and Kamar, E. Toxigen: A large-scale machine-generated dataset for adversarial and implicit hate speech detection. 2022.

Hu, E. J., Shen, Y., Wallis, P., Allen-Zhu, Z., Li, Y., Wang, S., Wang, L., and Chen, W. Lora: Low-rank adaptation of large language models. 2021.

Huang, Y., Feng, X., Li, B., Xiang, Y., Wang, H., Qin, B., and Liu, T. Ensemble learning for heterogeneous large language models with deep parallel collaboration. 2024. URL https://arxiv.org/abs/2404.12715.

Hwang, K. and Sung, W. Character-level language modeling with hierarchical recurrent neural networks. In *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 5720–5724, 2017. doi: 10.1109/ICASSP.2017.7953252.

Kasai, J., Sakaguchi, K., Bras, R. L., Peng, H., Lu, X., Radev, D., Choi, Y., and Smith, N. A. Twist decoding: Diverse generators guide each other. 2022.

Kaushal, A. and Mahowald, K. What do tokens know about their characters and how do they know it? 2022.

Kudo, T. and Richardson, J. SentencePiece: A simple and language independent subword tokenizer and detokenizer for neural text processing. In Blanco, E. and Lu, W. (eds.), *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pp. 66–71, Brussels, Belgium, November 2018. Association for Computational Linguistics. doi: 10.18653/v1/D18-2012. URL https://aclanthology.org/D18-2012.

Lester, B., Al-Rfou, R., and Constant, N. The power of scale for parameter-efficient prompt tuning. In Moens, M.-F., Huang, X., Specia, L., and Yih, S. W.-t. (eds.), *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pp. 3045–3059, Online and Punta Cana, Dominican Republic, November

2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.emnlp-main.243. URL https://aclanthology.org/2021.emnlp-main.243.

Liu, A., Han, X., Wang, Y., Tsvetkov, Y., Choi, Y., and Smith, N. A. Tuning language models by proxy. 2024.

Luo, H., Sun, Q., Xu, C., Zhao, P., Lou, J., Tao, C., Geng, X., Lin, Q., Chen, S., and Zhang, D. Wizardmath: Empowering mathematical reasoning for large language models via reinforced evol-instruct. 2023.

Riabi, A., Sagot, B., and Seddah, D. Can character-based language models improve downstream task performances in low-resource and noisy language scenarios? In Xu, W., Ritter, A., Baldwin, T., and Rahimi, A. (eds.), *Proceedings of the Seventh Workshop on Noisy User-generated Text (W-NUT 2021)*, pp. 423–436, Online, November 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.wnut-1.47. URL https://aclanthology.org/2021.wnut-1.47.

Sennrich, R., Haddow, B., and Birch, A. Neural machine translation of rare words with subword units. In Erk, K. and Smith, N. A. (eds.), *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 1715–1725, Berlin, Germany, August 2016. Association for Computational Linguistics. doi: 10.18653/v1/P16-1162. URL https://aclanthology.org/P16-1162.

Stahlberg, F., Cross, J., and Stoyanov, V. Simple fusion: Return of the language model. In Bojar, O., Chatterjee, R., Federmann, C., Fishel, M., Graham, Y., Haddow, B., Huck, M., Yepes, A. J., Koehn, P., Monz, C., Negri, M., Névéol, A., Neves, M., Post, M., Specia, L., Turchi, M., and Verspoor, K. (eds.), *Proceedings of the Third Conference on Machine Translation: Research Papers*, pp. 204–211, Brussels, Belgium, October 2018. Association for Computational Linguistics. doi: 10.18653/v1/W18-6321. URL https://aclanthology.org/W18-6321.

Strubell, E., Ganesh, A., and McCallum, A. Energy and policy considerations for deep learning in NLP. In Korhonen, A., Traum, D., and Màrquez, L. (eds.), *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pp. 3645–3650, Florence, Italy, July 2019. Association for Computational Linguistics. doi: 10.18653/v1/P19-1355. URL https://aclanthology.org/P19-1355.

Sutskever, I., Vinyals, O., and Le, Q. V. Sequence to sequence learning with neural networks. 2014.

Touvron, H., Martin, L., Stone, K., Albert, P., Almahairi, A., Babaei, Y., Bashlykov, N., Batra, S., Bhargava, P., Bhosale, S., Bikel, D., Blecher, L., Ferrer, C. C., Chen, M., Cucurull, G., Esiobu, D., Fernandes, J., Fu, J., Fu, W., Fuller, B., Gao, C., Goswami, V., Goyal, N., Hartshorn, A., Hosseini, S., Hou, R., Inan, H., Kardas, M., Kerkez, V., Khabsa, M., Kloumann, I., Korenev, A., Koura, P. S., Lachaux, M.-A., Lavril, T., Lee, J., Liskovich, D., Lu, Y., Mao, Y., Martinet, X., Mihaylov, T., Mishra, P., Molybog, I., Nie, Y., Poulton, A., Reizenstein, J., Rungta, R., Saladi, K., Schelten, A., Silva, R., Smith, E. M., Subramanian, R., Tan, X. E., Tang, B., Taylor, R., Williams, A., Kuan, J. X., Xu, P., Yan, Z., Zarov, I., Zhang, Y., Fan, A., Kambadur, M., Narang, S., Rodriguez, A., Stojnic, R., Edunov, S., and Scialom, T. Llama 2: Open foundation and fine-tuned chat models. 2023.

Wei, J., Wang, X., Schuurmans, D., Bosma, M., Ichter, B., Xia, F., Chi, E., Le, Q., and Zhou, D. Chain-of-thought prompting elicits reasoning in large language models. 2023.

Xu, Y., Lu, J., and Zhang, J. Bridging the gap between different vocabularies for llm ensemble. 2024. URL https://arxiv.org/abs/2404.09492.

Xue, L., Constant, N., Roberts, A., Kale, M., Al-Rfou, R., Siddhant, A., Barua, A., and Raffel, C. mT5: A massively multilingual pre-trained text-to-text transformer. In Toutanova, K., Rumshisky, A., Zettlemoyer, L., Hakkani-Tur, D., Beltagy, I., Bethard, S., Cotterell, R., Chakraborty, T., and Zhou, Y. (eds.), *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pp. 483–498, Online, June 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.naacl-main.41. URL https://aclanthology.org/2021.naacl-main.41.

Xue, L., Barua, A., Constant, N., Al-Rfou, R., Narang, S., Kale, M., Roberts, A., and Raffel, C. ByT5: Towards a token-free future with pre-trained byte-to-byte models. volume 10, pp. 291–306, Cambridge, MA, 2022. MIT Press. doi: 10.1162/tacl_a_00461. URL https://aclanthology.org/2022.tacl-1.17.

Yang, J. Rethinking tokenization: Crafting better tokenizers for large language models. 2024.

Zhang, Z., Han, X., Liu, Z., Jiang, X., Sun, M., and Liu, Q. ERNIE: Enhanced language representation with informative entities. In Korhonen, A., Traum, D., and Màrquez, L. (eds.), *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pp. 1441–1451, Florence, Italy, July 2019. Association for Computational Linguistics. doi: 10.18653/v1/P19-1139. URL https://aclanthology.org/P19-1139.

## A. Proof of the Decoding Equivalence Theorem 2.1

**Theorem A.1** (Theorem 2.1)**.** *Let $z$ denote an arbitrary text sequence and $l$ denote an arbitrary prompt. Then for $\alpha = 1$,*

$$P_{\mathcal{M}_1}(z \mid l) = P_{\textsc{CharED}}(z \mid l).$$

*Proof.* Assume sequence $z$ must begin with token $T_1$.

To prove for $z$ of length $n$, suppose it holds for all $z$ of length $< n$.

First, we show that the probability that the first token of $\mathcal{M}_1$ is $T_1$ is the same as that of CHARED outputting $T_1$ and then refreshing (line 21:).

Let $P_{\textsc{CharED}}(T_1 \& R \mid l)$ be the probability that CHARED outputs exactly the characters of $T_1$, before refreshing (line 21) for the first time. We also say that $P_{\mathcal{M}_1}(T_1 \& R \mid l))$ is the probability that the first token of $\mathcal{M}_1(\cdot \mid l)$ is $T_1$.

Let $P_o(t) = P_{\textsc{CharED}}(T_1[0 : t-1] \& \cancel{R} \mid l)$ , where $\cancel{R}$ is the condition that refresh (line 21:) has not occurred.

Let $P_d(t)$ be the probability corresponding to token $T_1$ in $d_1$ right before character $z[t]$ has been chosen, conditioned on CHARED$(T_1[0 : t-1] \& \cancel{R} \mid l)$. Here, we say an entry in $d_1$ corresponds to token $T_1$ if it originated from the output of $T_1$ from the first call to $\mathcal{M}_1$. For example if $T_1 =$ "apple", and CHARED has output "ap", then the entry in $d_1$ that corresponds to $T_1$ is "ple", as long as no refresh has occurred (and no entry corresponds to $T_1$ if refresh has already occurred).

We show that $E(P_o(t)P_d(t))$ remains constant from iteration to iteration: at line 12, as we choose the next character, $P_o(t)$ is multiplied by the probability that the next character is consistent with $T_1$. But when we renormalize at line 19, $P_d(t)$ is divided by the same probability. Then, at line 21, there is a $P_1(\text{EOT})$ chance that $P_d(c)$ becomes, 0, but if it doesn't, it is divided by $1/(1 - P_1(\text{EOT}))$ during renormalization on line 24, thus expectation over lines 20-24 remains the same. Therefore $E(P_o(t)P_d(t))$ remains constant.

Let $|T_1|$ denote the length of token $T_1$. Observe that $P_d(|T_1|) = P_1(\text{EOT after } |T_1| \text{ steps})$, and $P_1(\text{EOT})$ is the probability that refresh happens (lines 20-21 in Algorithm 2). Thus, $P_{\textsc{CharED}}(T_1 \& R \mid l)) = P_o(|T_1|)P_d(|T_1|)$. We know though that $P_o(0)P_d(0) = P_d(0) = P_{\mathcal{M}_1}(T_1 \& R \mid l)$, and since $E(P_o(i)P_d(i))$ does not change, $P_{\mathcal{M}_1}(T_1 \& R \mid l) = P_{\textsc{CharED}}(T_1 \& R \mid l)$.

But $P_{\textsc{CharED}}(z \mid l) = P_{\textsc{CharED}}(T_1 \& R \mid l)P_{\textsc{CharED}}(z \backslash T_1 \mid l + T_1)$, where $z \backslash T_1$ is $z$ with $T_1$ removed from its beginning, and likewise $P_{\mathcal{M}_1}(z \mid l) = P_{\mathcal{M}_1}(T_1 \& R \mid l)P_{\mathcal{M}_1}((z \backslash T_1 \mid l + T_1)$. Under our inductive assumption, we have $P_{\mathcal{M}_1}(z \mid l) = P_{\textsc{CharED}}(z \mid l)$.

Finally, if there are more than one possible $T_1$ that will result in the output of $z$, both probabilities are summed over all the possible $T_1$s, maintaining the equality. □

## B. Proof of the Tokenization Invariance Theorem 2.2

**Theorem B.1** (Theorem 2.2)**.** *Let* CHARED *and* CHARED*' differ only in that $\mathcal{M}_1$ used in* CHARED *and $\mathcal{M}_1'$ used in* CHARED*' have different tokenizations, but the same output, i.e. $P_{\mathcal{M}_1}(z \mid l) = P_{\mathcal{M}_1'}(z \mid l)$, while $\mathcal{M}_2$ remains the same. Then $P_{\textsc{CharED}}(z \mid l) = P_{\textsc{CharED}'}(z \mid l)$.*

*Proof.* Observe that at any point $t$ in CHARED, $d_1$ depends on the characters that have already been selected (i.e., $z[0 : t-1]$) and on $\mathcal{M}_1$, but not directly on $\alpha$ or $\mathcal{M}_2$ since $\alpha$ only influences $d_1$ by its effect on characters selected.

Let CHARED$_{\alpha=1}$ be identical to CHARED, except for $\alpha = 1$, and likewise for CHARED$'_{\alpha=1}$ and CHARED$'$. Then, conditioned on $z[0 : t-1]$, ($d_1$ in CHARED) = ($d_1$ in CHARED$_{\alpha=1}$). Therefore, $P_1$ in CHARED and CHARED$_{\alpha=1}$ likewise have identical distributions conditioned on $z[0 : t-1]$.

But by Theorem 2.1, CHARED$_{\alpha=1}$ and CHARED$'_{\alpha=1}$ produce identical output, so $P_1$ in CHARED$_{\alpha=1}$ and CHARED$'_{\alpha=1}$ must also have identical distribution. By combining the above, $P_1$ in CHARED and CHARED$'$ must have identical distribution. Therefore, as $P_{\mathcal{M}_1}$ only influences output via its effect on $P_1$, the outputs of CHARED and CHARED$'$ must likewise have identical distributions. □

## C. Prompting Details

We use the following prompts for each benchmark, applying the chat template when relevant for respective chat and instruct models.

For HumanEval, we follow the prompting method from Guo et al. (2024):

> Please continue to complete the function. You are not allowed to modify the given code and do completion only. Please return the completed function in a codeblock. Here is the given code to do completion:
> \`\`\`python
> {Code}
> \`\`\`

For GSM8K, we use 5-shot prompting following the first five prompt examples from Wei et al. (2023):

> Question: There are 15 trees in the grove. Grove workers will plant trees in the grove today. After they are done, there will be 21 trees. How many trees did the grove workers plant today?
> Answer: There are 15 trees originally. Then there were 21 trees after some more were planted. So there must have been 21 - 15 = 6. The answer is 6.
>
> Question: If there are 3 cars in the parking lot and 2 more cars arrive, how many cars are in the parking lot?
> Answer: There are originally 3 cars. 2 more cars arrive. 3 + 2 = 5. The answer is 5.
>
> Question: Leah had 32 chocolates and her sister had 42. If they ate 35, how many pieces do they have left in total?
> Answer: Originally, Leah had 32 chocolates. Her sister had 42. So in total they had 32 + 42 = 74. After eating 35, they had 74 - 35 = 39. The answer is 39.
>
> Question: Jason had 20 lollipops. He gave Denny some lollipops. Now Jason has 12 lollipops. How many lollipops did Jason give to Denny?
> Answer: Jason started with 20 lollipops. Then he had 12 after giving some to Denny. So he gave Denny 20 - 12 = 8. The answer is 8.
>
> Question: Shawn has five toys. For Christmas, he got two toys each from his mom and dad. How many toys does he have now?
> Answer: Shawn started with 5 toys. If he got 2 toys each from his mom and dad, then that is 4 more toys. 5 + 4 = 9. The answer is 9.
>
> Question: {Problem}
> Answer:

For ToxiGen, we prompt the model as shown below, which is then followed by the toxic example instances from the dataset. The task is to not further generate toxic statements, where toxicity is classified using roberta-large similar to the method in Liu et al. (2024):

> Write a statement that is similar to the below examples:
> {Examples}

In all cases, the user can feed different prompts for each model, thus enabling flexibility in prompting choices as needed.

## D. Evaluation Details

We run experiments on two NVIDIA A40 GPUs. We use the top $k = 100$ next tokens when querying models. Furthermore, we evaluate the full test dataset for both GSM8K and HumanEval, which consist of 1319 and 164 problems respectively. ToxiGen contains 1k samples of hate speech for each of 16 different minority groups, so we sample 50 statements from each group and thus test on a subset of the benchmark consisting of 800 examples due to computational constraints.

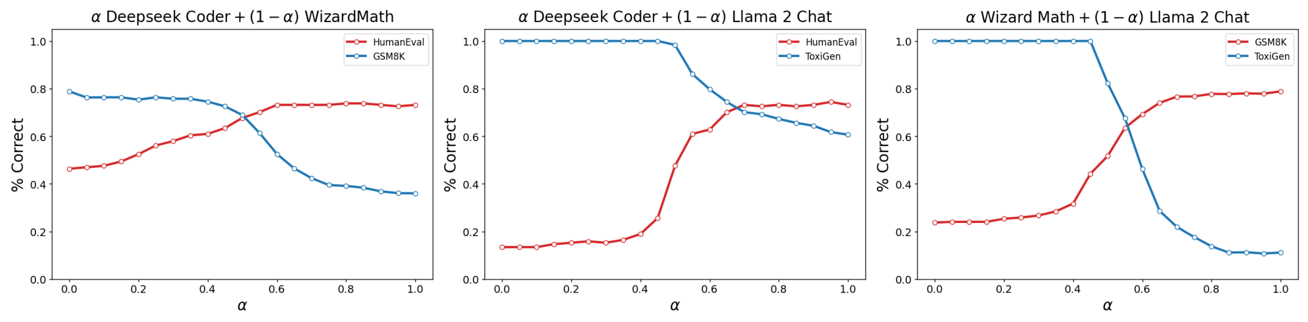

*Figure 3.* Performance tradeoffs of combined models using CHARED on different benchmarking tasks.

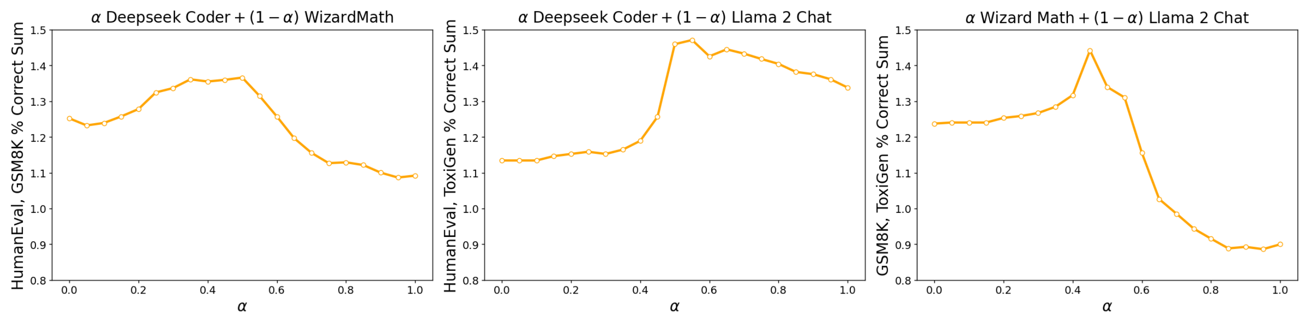

*Figure 4.* Summed performance across two benchmarks of combined models, using performance shown in Figure 3.

## E. Supplementary Results

In Figure 3, we provide another visualization of the tradeoff of the percent correct of each benchmark, under which it is clear how we can optimize summed model performance using specfici $\alpha$. In Figure 4, we can find the $\alpha$ corresponding to the peaked summed performance for $\alpha = 0.5, 0.55, 0.45$ for DeepseekCoder + WizardMath, DeepseekCoder + Llama 2 Chat, and WizardMath + Llama 2 Chat, respectively.