

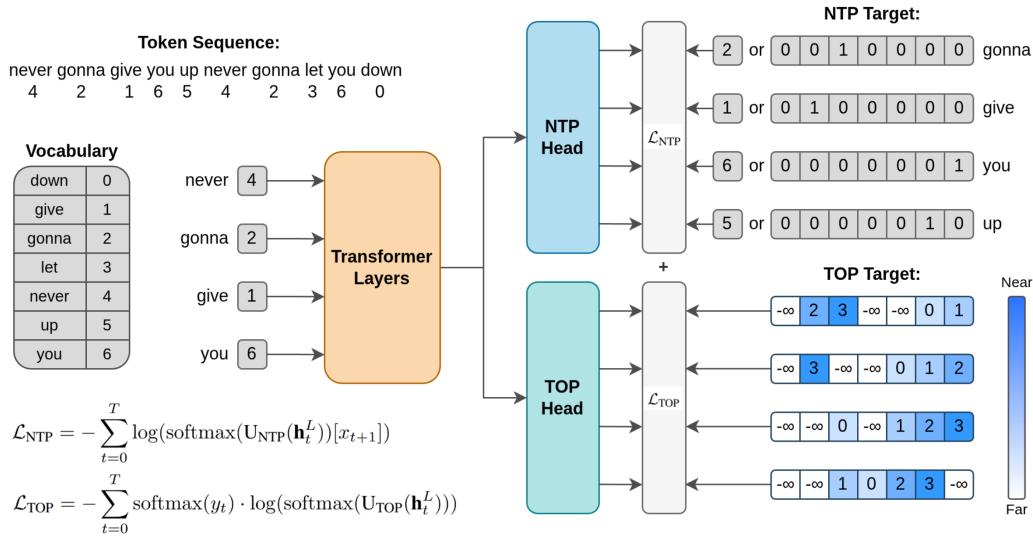
PREDICTING THE ORDER OF UPCOMING TOKENS IMPROVES LANGUAGE MODELING

000
001
002
003
004
005 **Anonymous authors**
006 Paper under double-blind review
007
008
009
010

ABSTRACT

011 Multi-Token Prediction (MTP) has been proposed as an auxiliary objective to improve next-token
012 prediction (NTP) in language model training but shows inconsistent improvements, underperforming
013 in standard NLP benchmarks. We found MTP’s exact future token prediction to be too difficult as
014 an auxiliary loss. Instead, we propose Token Order Prediction (TOP), which trains models to order
015 upcoming tokens by their proximity using a learning-to-rank loss. TOP requires only a single additional
016 unembedding layer compared to MTP’s multiple transformer layers. We pretrain models of
017 340M, 1.8B, and 7B parameters using NTP, MTP, DeepSeek MTP and TOP objectives. The results
018 of eight standard NLP benchmarks show that TOP overall outperforms NTP, MTP, and DeepSeek
019 MTP even at scale. On the synthetic star graph task, TOP enables pathfinding on graphs where NTP
020 and MTP fail.
021

1 INTRODUCTION



042
043 Figure 1: An overview of Token Order Prediction (TOP). Given an input token sequence, a vocabulary, a sequence
044 length of 4 and window size of 4, a TOP target sequence is constructed via Algorithm 1. The output hidden represen-
045 tation of the final layer goes to two separate unembedding heads for NTP and TOP. The final loss to optimize is a sum
046 of the NTP and TOP loss.
047

048 Current large language models (LLMs) are trained to predict the next token in a sequence during training, an unsupervised
049 learning task often referred to as next-token prediction (NTP). Although simple, NTP has been very successful
050 in creating powerful language models that can solve complex tasks and even reason over their context.

051 However, NTP has received various criticisms in recent years. A notable argument by LeCun (2024) claims that NTP
052 at inference time accumulates errors over every time step and inevitably falls off greatly in accuracy. This was however
053 refuted by Bachmann & Nagarajan (2024), in which they argue that the main issue of NTP lies not in inference time
error accumulation; rather, that teacher-forcing is unable to learn an accurate next-token predictor in the first place.

Building off ideas such as ProphetNet (Qi et al., 2020), Multi-Token Prediction (MTP) (Gloeckle et al., 2024) has emerged as a relatively successful auxiliary learning task to improve NTP in LLM training. MTP adds multiple heads to the end of a transformer that each predict a different offset of tokens ahead. All MTP heads share the same trunk of transformer layers, with the hope that having these auxiliary heads leads to the model learning better internal representations that are considerate of not only the next immediate token, but also future tokens that may come after it. It has been shown that MTP improves performance of language models on certain generative tasks that require some level of look-ahead, such as coding and summarization. This method was used in the training of DeepSeek-V3 (DeepSeek-AI et al., 2024), although with sequential instead of parallel MTP heads.

However, the original MTP paper (Gloeckle et al., 2024) showed that MTP does not generally improve language modeling performance on every standard NLP task, as evident from the downstream task performance seen in their Appendix G. Important to note is that MTP does not seem to work on smaller models. In generative tasks, MTP harms performance and only starts to gain advantage over NTP for models with over 1-3 billion parameters. Additionally, the number of future tokens to predict is a hyperparameter that needs to be set before training. This is critical because increasing the future token count requires adding more heads, which adds more parameters to train and more compute required. Furthermore, the paper shows that increasing the number of future tokens does not guarantee better performance even on the benefiting tasks, where it is shown that 4 future tokens performs better than 8 in coding.

We aim to improve upon MTP by introducing a different auxiliary training objective with the same goal as MTP: enhancing next-token prediction performance through better internal representations. However, instead of exactly predicting multiple future tokens, we propose that a better training objective is to predict the order of upcoming tokens in the sequence with a learning-to-rank loss. In this paper, we contribute the following:

1. We introduce Token Order Prediction (TOP), a novel auxiliary training loss in addition to NTP to improve language modeling in general.
2. For each of the four training strategies NTP, MTP, DeepSeek MTP (DS-MTP) and TOP, we pretrain language models with sizes of 340M, 1.8B, and 7B parameters on up to 104B tokens.
3. We evaluate these models on standard NLP benchmarks and show that TOP improves upon NTP, MTP, and DS-MTP even at scale. Additionally, the synthetic star graph pathfinding task shows TOP can solve lookahead problems that NTP, MTP, and DS-MTP cannot.

2 BACKGROUND

Next-token prediction (NTP) is the standard training objective for present-day language models. This task is learned by optimizing the cross-entropy loss over the sequence length. Given sequence length T , model dimension D , vocabulary size V and $\mathbf{x} = \{x_0, \dots, x_{T+1} \mid x_i \in \mathbb{Z}\}$ as the input token sequence, this loss is written as

$$\mathcal{L}_{\text{NTP}} = - \sum_{t=0}^T \log(\mathbf{P}_\theta(x_{t+1} | x_{0:t})) \quad (1)$$

where \mathbf{P}_θ is the output probability given by the language model with parameters θ . The probability of the next token x_{t+1} given this model is written as

$$\mathbf{P}_\theta(x_{t+1} | x_{0:t}) = \text{softmax}(\mathbf{U}_{\text{NTP}}(\mathbf{h}_t^L))[x_{t+1}] \quad (2)$$

where the hidden representation $\mathbf{h}_t^L \in \mathbb{R}^D$ is generated by a transformer up to the final layer L conditioned on $x_{0:t}$, and the NTP head $\mathbf{U}_{\text{NTP}} : \mathbb{R}^D \rightarrow \mathbb{R}^V$ is a linear unembedding layer to project \mathbf{h}_t^L onto the vocabulary. The probability is taken at the index of the target token $[x_{t+1}]$.

Multi-Token Prediction (MTP) (Gloeckle et al., 2024) was proposed as an architectural modification that adds additional MTP heads¹ in the form of parallel, singular transformer layers that each output a future token prediction at offset positions. Given N as the number of future tokens to predict (including the next token), the MTP loss can be written as

$$\mathcal{L}_{\text{MTP}} = - \sum_{t=0}^T \log(\mathbf{P}_\theta(x_{t+1:t+N} | x_{0:t})) = - \sum_{t=0}^T \sum_{n=1}^N \log(\mathbf{P}_\theta(x_{t+n} | x_{0:t})) \quad (3)$$

¹Disambiguation: Architecturally, MTP heads are transformer blocks, as we follow the terminology from Gloeckle et al. (2024). Meanwhile, NTP head and TOP head are linear unembedding layers.

If we define \mathbf{h}_t^{L-1} as the hidden representation before the last transformer layer and have F_i for $i = 1, \dots, N$ as the MTP heads in the form of singular transformer layers for each future token, and all heads share the same unembedding layer or NTP head U_{NTP} , then

$$P_\theta(x_{t+n}|x_{0:t}) = \text{softmax}(U_{\text{NTP}}(F_n(\mathbf{h}_t^{L-1}))[x_{t+n}]) \quad (4)$$

MTP promises better performance on generative tasks such as coding and summarization that benefit from the look-ahead nature of MTP. MTP also allows the model to do a form of self-speculative decoding, which speeds up inference to some degree. However, as mentioned earlier, MTP does not seem to improve overall language modeling performance on downstream tasks other than those mentioned above, struggling on standard NLP benchmarks.

There have also been other MTP variants such as the one used for training DeepSeek V3 (DeepSeek-AI et al., 2024). We refer to this variant as DS-MTP. Unlike the original MTP, DS-MTP arranges the MTP heads sequentially. Each head after the first one, receives concatenated, normalized embeddings from the original tokens up to that offset. We define $E: \mathbb{R}^V \rightarrow \mathbb{R}^D$ as the embedding layer also shared with the main transformer trunk.

$$h_t^{L-1+n} = \begin{cases} F_n(\mathbf{h}_t^{L-1}) & \text{if } n = 1 \\ F_n([\text{RMSNorm}(\mathbf{h}_t^{L-1+n-1}); \text{RMSNorm}(E(x_{0:t+n-1}))]) & \text{if } 2 \leq n \leq N \end{cases} \quad (5)$$

$$P_\theta(x_{t+n}|x_{0:t+n-1}) = \text{softmax}(U_{\text{NTP}}(\mathbf{h}_t^{L-1+n})[x_{t+n}]) \quad (6)$$

$$\mathcal{L}_{\text{DS-MTP}} = \sum_{t=0}^T \sum_{n=1}^N \log(P_\theta(x_{t+n}|x_{0:t})) \quad (7)$$

Notably, DeepSeek V3 only used $N = 2$ for their MTP training, which means the model only learns to predict the next two tokens. This might be related to a key point of our paper, which is that MTP is too difficult an objective to be an effective auxiliary loss to NTP, especially for larger look-ahead values N .

3 MOTIVATION

Our hypothesis for why MTP only partially improves language modeling is that MTP is too difficult as a learning objective. If we look at the original MTP paper (Gloekle et al., 2024), there are two empirical results that support this argument. First, MTP does not improve performance of small language models on generative tasks, such as coding. This suggests that a certain capability threshold is required for MTP’s multi-token modeling to be effective, which they observe to be in the 1B-3B parameter range. Second, increasing the number of future tokens in MTP does not guarantee better performance overall. The ideal number of future tokens varies across different tasks. Not only does this make it difficult to determine the optimal number beforehand, it also indicates that there are thresholds of look-ahead distance where the difficulty of prediction starts to hurt learning instead of helping it.

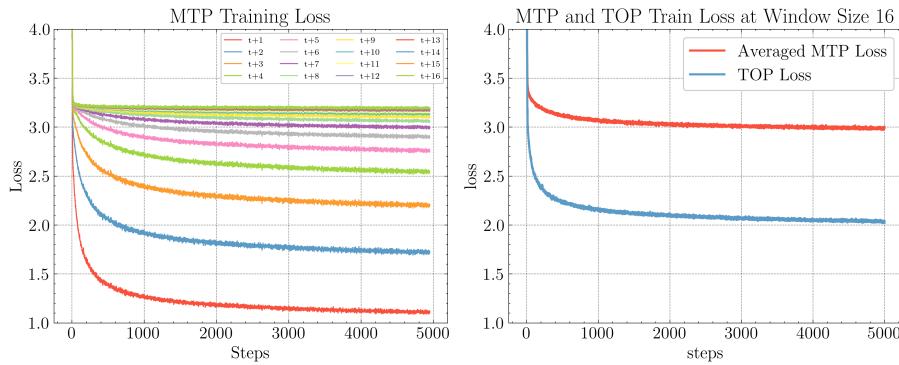


Figure 2: Left: Training loss of a MTP transformer with 16 MTP heads predicting tokens at $t + 1, \dots, t + 16$ offsets. Right: Training loss of the MTP model averaged over all 16 heads, compared to the training loss of the TOP model with window size 16.

To illustrate our argument, we train a small 16M parameter transformer with 16 MTP heads and visualize the training loss of each MTP head in Figure 2. A clear pattern emerges where the losses of predicting the tokens at positions $t + 1, \dots, t + 16$ arrange themselves from bottom to top. Each future token farther away significantly worsens in loss compared to the immediate next token loss and shows a decreased rate of loss descent, indicating the difficulty of exactly predicting far ahead. We believe that relaxing this MTP objective will make it more useful as an auxiliary loss. Compared to a similarly sized model with the TOP objective at window size 16, we see that the TOP loss is lower.

162 4 METHOD
163

164 We propose Token Order Prediction (TOP), a novel auxiliary training loss for language modeling. Given a sequence
165 of tokens $\mathbf{x} = \{x_0, \dots, x_T \mid x_i \in \mathbb{Z}\}$, we construct a TOP target sequence $\mathbf{y} = \{y_0, \dots, y_T \mid y_i \in \mathbb{Z}^V\}$ where V
166 is a vocabulary size, in which every index contains a score ranking each token in the vocabulary based on their order
167 in the sequence \mathbf{x} , going in descending order from closest to furthest first appearance after x_t . We also introduce a
168 hyperparameter, window size, within which the token order is evaluated. To better understand this target sequence,
169 please refer to the pseudocode in Algorithm 1 and the visualization in Figure 1 on how to construct it. In practice,
170 we have an optimized Triton kernel for this function that creates the target sequence on the fly during training and
171 practically incurs no overhead. Alternatively, one could also pre-process an entire dataset beforehand.

172 **Algorithm 1** Convert a token sequence to a TOP target sequence
173

174 **Goal:** For each position, compute a "proximity" score to the next occurrence of every token within a window of
175 size W .

176 **Require:** Token sequence \mathbf{x} of length $T + W$, vocab size V , window size W .

177 **Ensure:** Tensor \mathbf{y} of shape (T, V)

```

178 1: Initialize  $\mathbf{y} \leftarrow -\infty$                                  $\triangleright$  By default, tokens not appearing in the sequence are set to  $-\infty$ 
179 2: Initialize  $\mathbf{n}[v] \leftarrow T + W$  for all  $v \in [0, V - 1]$        $\triangleright$   $\mathbf{n}$  keeps track of next occurrence of every token
180 3: for  $t \leftarrow T + W - 1$  down to 0 do                          $\triangleright$  Iterate backwards through sequence
181 4:   if  $\mathbf{x}[t]$  is valid then                                 $\triangleright$  Check if token  $\mathbf{x}[t]$  is in vocabulary
182 5:      $\mathbf{n}[\mathbf{x}[t]] \leftarrow t$                                  $\triangleright$  Record most recent position of token  $\mathbf{x}[t]$ 
183 6:   end if
184 7:   if  $t < T$  then                                          $\triangleright$  Length of output is only  $T$ 
185 8:     for each  $v \in [0, V - 1]$  do                          $\triangleright$  Update each token in the vocabulary
186 9:        $d \leftarrow \mathbf{n}[v] - t$                                  $\triangleright$   $d$  equals steps ahead until next occurrence of token  $v$ 
187 10:      if  $0 < d \leq W$  then                                 $\triangleright$  If the distance to next occurrence of token  $v$  is within the window
188 11:         $\mathbf{y}[t, v] \leftarrow W - d$                                  $\triangleright$  Closer tokens get assigned a larger proximity score, with maximum  $W$ 
189 12:      end if
190 13:    end for
191 14:  end if
192 15: end for

```

193 To train the model to order upcoming tokens as in the target sequence, we borrow a loss formulation from the learning-
194 to-rank literature (Pobrotyn et al., 2020), more specifically from ListNet (Cao et al., 2007). This listwise ranking loss
195 is formulated as the distance between the top-one probability of two lists of scores, where the distance metric is
196 cross-entropy. The TOP auxiliary loss is defined as follows:

$$\mathcal{L}_{\text{TOP}} = - \sum_{t=0}^T \text{softmax}(y_t) \cdot \log(\text{softmax}(\mathbf{U}_{\text{TOP}}(\mathbf{h}_t^L))) \quad (8)$$

200 Note that $\text{softmax}(\mathbf{U}_{\text{TOP}}(\mathbf{h}_t^L))$ is not a probability distribution by definition, hence why we do not write it as \mathbf{P}_θ . The
201 correct way to think about it is to view $\mathbf{U}_{\text{TOP}}(\mathbf{h}_t^L)$ as the model prediction of the ranking in the form of proximity
202 scores, and $\text{softmax}(\mathbf{y}) \cdot \log(\text{softmax}(\mathbf{y}))$ as the ranking loss defined in ListNet. Also note that there are no additional
203 transformer layers needed for TOP. There is however an additional linear unembedding layer $\mathbf{U}_{\text{TOP}} : \mathbb{R}^D \rightarrow \mathbb{R}^V$ in
204 parallel to the NTP head. Both unembedding heads \mathbf{U}_{NTP} and \mathbf{U}_{TOP} receive the same hidden state \mathbf{h}_t^L , which is the
205 output of the final transformer layer. We refer to these unembedding layers as NTP head and TOP head, respectively.
206 The final loss being optimized is simply a sum of the NTP loss and the TOP loss:

$$\mathcal{L} = \mathcal{L}_{\text{NTP}} + \mathcal{L}_{\text{TOP}} \quad (9)$$

207 Through this specific target sequence formulation and ranking loss function, the model is expected to learn an internal
208 representation that can approximately *construct* the future sequence by returning the most probable order of upcoming
209 tokens. This is expected to be an easier task than trying to exactly predict a future token at some offset. It is also much
210 more scalable compared to MTP, requiring only one additional unembedding matrix. While this matrix can be large,
211 no additional parameters are needed even when adjusting window size, unlike MTP where each additional future token
212 requires a full extra transformer layer. Specifically, while a TOP head requires DV extra parameters, N number of
213 MTP heads require $N(4D^2 + 12D^2 + 2D)$ assuming a standard transformer block with MLP hidden size $4D$ and 2
214 RMSNorms. Moreover, the cost of a single unembedding layer gets amortized as the model size scales up. In practice,
215

216
217 Table 1: Training configuration and hyperparameters for {340M, 1.8B, 7B} models.
218
219

| Parameter | Value | Parameter | Value |
|-------------------------------|--------------------|--------------------|----------------------|
| Hidden size | {1024, 2048, 4096} | Optimizer | AdamW |
| Num. layers | {24, 32, 30} | Learning rate | {3e-4, 2e-4, 1.2e-4} |
| Num. heads | {16, 32, 32} | LR schedule | Cosine |
| Num. KV heads | {16, 32, 8} | Minimum LR (%) | 10% |
| Sequence length | 4096 | Warmup steps | {1000, 2000, 2000} |
| RoPE θ | 10,000 | Device batch size | {16, 16, 8} |
| Tied embeddings | False | Grad. accum. steps | {1, 1, 2} |
| Vocab size | 32,000 | Global batch size | 128 |
| TOP window size | 4096 | Training steps | {100k, 200k, 200k} |
| MTP/DS-MTP num. future tokens | 4 | Gradient clip max. | 1.0 |

231 we use a fused Triton kernel that readily performs both the unembedding and loss calculation block-wise in one pass,
232 thus making the overhead minimal. This kernel is a modification to fused linear cross-entropy loss kernels from Yang
233 & Zhang (2024), resulting in the same performance as the non-modified version.

234 We find that we do not need additional transformer blocks like MTP for TOP because both the NTP and TOP heads
235 are mainly aligned on the same objective: assigning the highest score to the next token. Although it is possible to train
236 a language model on only the TOP objective, the resulting model will only be able to do greedy generation. An NTP
237 head is still needed for non-greedy, probability sampling-based inference. At inference time, we remove the TOP head
238 and use only the NTP head, making the model equivalent to the original transformer architecture.

240 5 EXPERIMENTS AND RESULTS

242 5.1 GENERAL LANGUAGE MODELING

244 We train transformer models for the training methods NTP, MTP, DS-MTP, and TOP in 3 sizes each: 340M, 1.8B, and
245 7B. These sizes are an approximate naming scheme; each model of each training method will have slightly different
246 parameter counts. We try to match the parameter count at training time, excluding embedding parameters. This means
247 that by setting the MTP or DS-MTP number of future tokens to 4, the shared trunk will be reduced by 3 layers to
248 account for the added MTP heads, as is done in the original MTP paper. We train all models on the sample-100BT
249 subset of FineWeb-Edu (Lozhkov et al., 2024). The 340M models are trained on 52B tokens, while the 1.8B and 7B
250 models are trained on 104B tokens. We use the Flame framework and flash-linear-attention (Yang & Zhang, 2024)
251 repository to implement and train our models. The full training configuration and hyperparameters for all model sizes
252 are detailed in Table 1. We want the TOP window size to be as large as possible but still tractable to compute the target
253 sequence with. Here, we set it to be equal to the sequence length, which means Algorithm 1 will receive an input with
254 twice the sequence length.

255 We evaluate our models on eight standard NLP benchmarks: ARC challenge (Clark et al., 2018), Lambada (Paperno
256 et al., 2016), PIQA (Bisk et al., 2020), SciQ (Welbl et al., 2017), Social IQa (Sap et al., 2019), TriviaQA (Joshi
257 et al., 2017), NaturalQuestions Open (Kwiatkowski et al., 2019), and HellaSwag (Zellers et al., 2019), with full results
258 presented in Table 2. Across all model sizes, TOP shows overall better performance over MTP, DS-MTP, and the
259 baseline NTP models on most tasks.

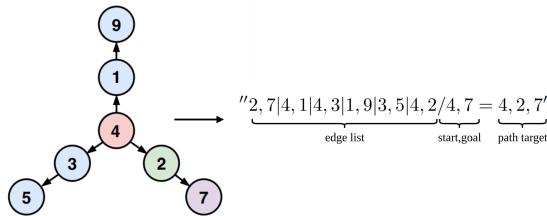
260 Our reproduction of MTP shows smaller MTP models achieve competitive results. This finding complements the
261 original MTP paper, which did not report on models smaller than 7B on the standard NLP benchmarks. Consistent
262 with the original study however, the 7B MTP model underperforms in these tasks. While the MTP paper suggests that
263 it scales effectively on coding tasks, our findings indicate that this scalability does not extend to non-coding tasks. In
264 contrast, our TOP model improves in performance as it scales to 7B and surpasses the 7B NTP and MTP baseline. This
265 suggests that in more general tasks, TOP performs and scales better than MTP. We also do not see an improvement in
266 performance from DS-MTP compared to MTP in our reproduction.

267 We also report the training loss recorded only on the NTP head of each model (i.e. the first MTP head for MTP
268 models). Interestingly, TOP exhibits a higher training loss than NTP, yet achieves lower Lambada perplexity and
269 better benchmark scores. We argue that TOP may act as a regularizer, mitigating overfitting on the limited FineWeb-
Edu subset we used for training. However, this claim may need further investigation.

270
271 Table 2: General language modeling evaluation results of NTP vs MTP vs DS-MTP vs TOP on standard NLP bench-
272 marks. We report the NTP head only final training loss, the accuracy and perplexity on Lambada, the normalized
273 accuracy on HellaSwag, ARC Challenge, PIQA, and SciQ, the accuracy on Social IQa, and the exact match score on
274 NaturalQuestions Open and TriviaQA. Green/red values indicate the performance difference from the NTP baseline.

| 275 Size | 276 Model | 277 NTP Loss | 278 Lambada Acc. \uparrow | 279 Lambada PPL \downarrow | 280 HellaSwag N. Acc. \uparrow | 281 ARC N. Acc. \uparrow | 282 PIQA N. Acc. \uparrow | 283 SciQ N. Acc. \uparrow | 284 Social IQa Acc. \uparrow | 285 NQ Open E.M. \uparrow | 286 TriviaQA E.M. \uparrow |
|----------|-----------|--------------|-----------------------------|------------------------------|----------------------------------|----------------------------|-----------------------------|-----------------------------|--------------------------------|-----------------------------|------------------------------|
| 277 340M | NTP | 2.39 | 36.35 | 30.34 | 42.53 | 28.84 | 66.65 | 74.90 | 39.82 | 1.94 | 4.93 |
| | MTP | 2.47 | 35.32 <small>-1.03</small> | 35.31 <small>+4.96</small> | 42.73 <small>+0.20</small> | 29.86 <small>+1.02</small> | 66.49 <small>-0.16</small> | 77.40 <small>+2.50</small> | 39.00 <small>-0.82</small> | 2.35 <small>+0.42</small> | 2.55 <small>-2.37</small> |
| | DS-MTP | 2.50 | 34.66 <small>-1.69</small> | 40.99 <small>+10.65</small> | 40.29 <small>-2.24</small> | 27.56 <small>-1.28</small> | 63.76 <small>-2.88</small> | 73.70 <small>-1.20</small> | 37.92 <small>-1.89</small> | 0.36 <small>-1.58</small> | 0.87 <small>-4.05</small> |
| | TOP | 2.40 | 37.07 <small>+0.72</small> | 28.76 <small>-1.58</small> | 43.57 <small>+1.04</small> | 29.35 <small>+0.51</small> | 67.57 <small>+0.92</small> | 79.80 <small>+4.90</small> | 39.00 <small>-0.82</small> | 2.22 <small>+0.28</small> | 4.37 <small>-0.55</small> |
| 281 1.8B | NTP | 2.06 | 49.58 | 11.38 | 60.05 | 38.65 | 73.50 | 86.40 | 41.56 | 4.54 | 11.85 |
| | MTP | 2.14 | 47.93 <small>-1.65</small> | 13.69 <small>+2.31</small> | 58.29 <small>-1.76</small> | 40.61 <small>+1.96</small> | 73.07 <small>-0.44</small> | 87.20 <small>+0.80</small> | 42.12 <small>+0.56</small> | 4.46 <small>-0.08</small> | 15.98 <small>+4.13</small> |
| | DS-MTP | 2.14 | 48.71 <small>-0.87</small> | 13.32 <small>+1.94</small> | 57.48 <small>-2.57</small> | 40.44 <small>+1.79</small> | 71.87 <small>-1.63</small> | 86.40 | 42.84 <small>+1.28</small> | 4.21 <small>-0.33</small> | 12.06 <small>+0.21</small> |
| | TOP | 2.08 | 50.34 <small>+0.76</small> | 11.19 <small>-0.19</small> | 60.45 <small>+0.40</small> | 42.32 <small>+3.67</small> | 74.16 <small>+0.65</small> | 87.90 <small>+1.50</small> | 42.53 <small>+0.97</small> | 5.37 <small>+0.83</small> | 18.93 <small>+7.07</small> |
| 285 7B | NTP | 1.88 | 55.89 | 7.97 | 67.44 | 45.65 | 76.99 | 88.60 | 44.37 | 7.31 | 24.28 |
| | MTP | 1.95 | 53.13 <small>-2.76</small> | 8.99 <small>+1.03</small> | 65.85 <small>-1.58</small> | 45.56 <small>-0.09</small> | 75.73 <small>-1.25</small> | 89.30 <small>+0.70</small> | 44.11 <small>-0.26</small> | 7.40 <small>+0.08</small> | 23.36 <small>-0.92</small> |
| | DS-MTP | 1.96 | 55.62 <small>-0.27</small> | 8.52 <small>+0.55</small> | 66.03 <small>-1.40</small> | 44.37 <small>-1.28</small> | 75.79 <small>-1.20</small> | 88.70 <small>+0.10</small> | 43.76 <small>-0.61</small> | 6.57 <small>-0.75</small> | 18.54 <small>-5.73</small> |
| | TOP | 1.89 | 57.03 <small>+1.14</small> | 7.64 <small>-0.32</small> | 68.73 <small>+1.29</small> | 46.42 <small>+0.77</small> | 76.39 <small>-0.60</small> | 91.60 <small>+3.00</small> | 43.91 <small>-0.46</small> | 7.70 <small>+0.39</small> | 30.90 <small>+6.63</small> |

289 5.2 THE STAR GRAPH TASK



303 Figure 3: Illustration of a star graph training sample with $d = 3$ and $l = 3$ due to Bachmann & Nagarajan (2024).

304
305
306 In addition to general language modeling, we also evaluate TOP on a synthetic task, the star graph pathfinding problem.
307 It was proposed by Bachmann & Nagarajan (2024) to highlight the weakness of training with NTP where the model
308 fails to learn the correct look-ahead solution. The task is to find a path from a starting node to a goal node, given a
309 star graph $G(d, l)$ with d paths branching from the starting node and path length l . Refer to Figure 3 for an example
310 training sample of this task. We train standard transformers with 8 layers, embedding size 384, and 6 attention heads
311 for 6 training objectives: NTP, TOP, MTP-2, MTP-4, DS-MTP-2, and DS-MTP-4. The MTP and DS-MTP models
312 have different number of future tokens i.e. two and four MTP heads. Note that in the case of MTP and DS-MTP in
313 this task, we only subtract one layer from the main trunk and add the MTP heads on top because the models are too
314 small, resulting in MTP models with larger parameter counts compared to NTP and TOP. We train these models on
315 4 star graph setups: $G(3, 3)$, $G(3, 5)$, $G(5, 3)$, and $G(5, 5)$ with $N = 30$ which means each node label is sampled
316 uniformly from 30 labels i.e. tokens. In each setup we generate 300,000 training samples and 10,000 test samples and
317 train for 100 epochs to convergence. We use a batch size of 4096, learning rate of 0.003 with warmup of 1500 and
318 cosine decay down to a learning rate of 0.001 for all models.

319 We present the results of the star graph task in Table 3 and Figure 4. Similar to the results in (Bachmann & Nagarajan,
320 2024), NTP performs poorly across the board and fails to learn the appropriate look-ahead mechanism. We also
321 observe that the effectiveness of MTP is dependent on its number of heads. For instance, a configuration with four
322 heads is effective, while one with only two is insufficient, particularly for graphs with longer paths. However, the TOP
323 model demonstrates the most robust performance. On the $G(5, 5)$ star graph, the TOP model is the only model with
perfect test set accuracy, where even the MTP and DS-MTP model with 4 heads fail.

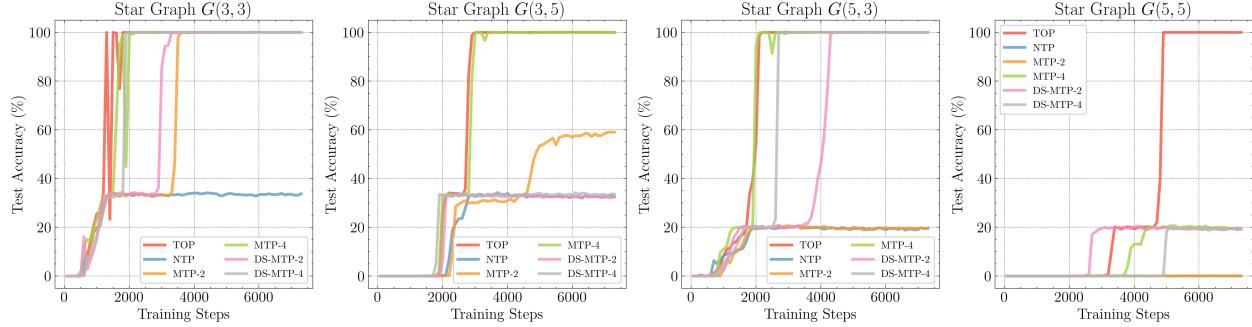


Figure 4: Test set accuracy during training of NTP, TOP and MTP models on the star graph pathfinding task.

Table 3: Parameter count and test set accuracy of star graph task models.

| Model | Param. Count | $G(3,3)$ | $G(3,5)$ | $G(5,3)$ | $G(5,5)$ |
|----------|--------------|------------|------------|------------|------------|
| NTP | 14.22M | 33.77 | 32.53 | 19.49 | 0.06 |
| MTP-2 | 16.00M | 100 | 59.02 | 19.58 | 0.05 |
| MTP-4 | 19.55M | 100 | 100 | 100 | 19.53 |
| DS-MTP-2 | 16.59M | 100 | 32.46 | 100 | 19.20 |
| DS-MTP-4 | 20.73M | 100 | 33.62 | 100 | 19.26 |
| TOP | 14.24M | 100 | 100 | 100 | 100 |

5.3 SELF-SPECULATIVE DECODING

Speculative decoding Stern et al. (2018) is a technique for accelerating inference by first using a smaller model to generate predictions and then employing the original model as a validator. In MTP, all of the heads can be used simultaneously to predict future tokens, with each head generating one token. The predicted tokens are then validated in a second forward pass using the original model. Because MTP uses the same model, the technique is referred to as self-speculative decoding. This requires performing inference twice: once for generation and once for validation.

Although TOP is intended to be used only at training time to improve learning, we also explore the possibility of using the TOP head for self-speculative decoding. To do this, we construct a future sequence by ordering the TOP head’s predicted tokens by proximity score, appending them to the input, and running another forward pass to check the longest common prefix i.e. acceptance rate with the NTP head’s predictions. Importantly, the acceptance rate or average number of accepted tokens does not imply a directly equivalent inference speed-up because two forward passes are required.

We take all TOP, MTP, and DS-MTP models and evaluate their self-speculative decoding potential on texts of different domains. For each domain, we take 5000 random snippets of text and calculate the average number of accepted tokens per validation forward pass. We present the results in Table 4. Evidently, TOP does not perform as well as MTP nor DS-MTP for self-speculative decoding. A maximum acceptance rate of 1.54 on the 7B TOP model might translate to a 1.2-1.4x inference speed-up, depending on the implementation. Meanwhile the 7B MTP model goes up to 2.49 acceptance rate, still slightly below the numbers reported in the original paper given 4 MTP heads, while the DS-MTP 7B model achieves up to an impressive 3.03 acceptance rate.

5.4 WINDOW SIZE ABLATION

We investigate the effect of varying window size when training using the TOP objective. We pretrain 340M parameter models with the same setup as in Section 5.1, only changing the TOP window size with values 4, 16, 128, 1024, and 4096. We report the downstream benchmark results in Table 5, comparing each setup to the baseline NTP model as well. We observe varying results in each benchmark. We hypothesize that every task might benefit from different amounts of lookahead. All window sizes however outperform the NTP baseline.

378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396

Table 4: Average number of accepted tokens per forward pass when doing self-speculative decoding.

| Model | Size | Domain | | | |
|--------|------|-----------|-------|-------|-------|
| | | Wikipedia | Books | Code | Math |
| MTP | 340M | 2.145 | 1.883 | 2.232 | 2.208 |
| | 1.8B | 2.268 | 2.023 | 2.381 | 2.420 |
| | 7B | 2.392 | 2.090 | 2.460 | 2.488 |
| DS-MTP | 340M | 2.806 | 2.719 | 2.596 | 2.691 |
| | 1.8B | 2.963 | 2.821 | 2.754 | 2.805 |
| | 7B | 3.030 | 2.912 | 2.921 | 2.968 |
| TOP | 340M | 1.379 | 1.333 | 1.366 | 1.368 |
| | 1.8B | 1.472 | 1.414 | 1.485 | 1.507 |
| | 7B | 1.517 | 1.415 | 1.529 | 1.546 |

Table 5: Token order prediction window size ablation results on 340M models. We report the accuracy and perplexity on Lambada, the normalized accuracy on HellaSwag, ARC Challenge, PIQA, and SciQ, and the exact match score on NaturalQuestions Open and TriviaQA. Best scores are bolded.

| Window Size | Lambada | | HellaSwag | | ARC | | PIQA | | SciQ | | NQ Open | | TriviaQA | |
|-------------|-----------------|------------------|--------------------|--------------------|--------------------|--------------------|--------------------|--------------------|--------------------|-----------------|-----------------|-----------------|-----------------|-----------------|
| | Acc. \uparrow | PPL \downarrow | N. Acc. \uparrow | E.M. \uparrow |
| 4 | 37.36 | 27.42 | 43.22 | 29.78 | 66.81 | 77.40 | 3.05 | | 3.05 | 3.05 | | 6.38 | | |
| 16 | 38.68 | 25.30 | 43.43 | 30.55 | 68.66 | 75.70 | | | | 2.08 | | 3.72 | | |
| 128 | 37.98 | 26.69 | 43.91 | 28.50 | 69.04 | 78.10 | | | | 2.22 | | 4.07 | | |
| 1024 | 36.95 | 27.80 | 43.74 | 30.12 | 67.85 | 76.60 | | | | 2.66 | | 4.15 | | |
| 4096 | 37.07 | 28.76 | 43.57 | 29.35 | 67.57 | 79.80 | | | | 2.22 | | 4.37 | | |
| NTP | 36.35 | 30.34 | 42.53 | 28.84 | 66.65 | 74.90 | | | | 1.94 | | 4.93 | | |

406
407 6 RELATED WORK
408409
410 **Language Model Losses** Many previous works have explored variations of the language modeling loss, most of
411 them for use in training encoder models. Masked language modeling (MLM) randomly masks input tokens and trains
412 the model to recover them from bidirectional context (Devlin et al., 2019). For example, T5 uses a denoising span-
413 corruption objective in which contiguous spans are replaced by sentinel tokens and the model must reconstruct the
414 spans (Raffel et al., 2020); this yields shorter target sequences and faster training. XLNet’s permutation language
415 modeling samples random autoregressive orderings to capture bidirectional dependencies while remaining autoregressive
416 (Yang et al., 2019). Similarly, denoising autoencoder pretraining as in BART corrupts text (e.g., by shuffling sentences
417 or infilling masked spans) and learns to reconstruct the original text (Lewis et al., 2020). UL2 (Tay et al., 2022) further
418 unifies these ideas with a mixture-of-denoisers objective, interleaving various span- and prefix-corruption schemes to
419 improve robustness across tasks. Retrieval-augmented models like RETRO add a nearest-neighbor retrieval step during
420 pretraining, conditioning generation on retrieved document chunks to reduce perplexity and enable easy knowledge
421 updates (Borgeaud et al., 2022). Other alternatives include replaced-token detection as in ELECTRA (Clark et al.,
422 2020), where the model sees plausible substitutes in place of masked tokens and must identify which tokens were
423 replaced.424
425 **Multi-Token Prediction** Next-token prediction (NTP), the standard training objective for language models, has been
426 criticized for its limitations in long-range planning. The teacher-forcing approach may fail to learn an accurate next
427 token predictor, which hinders the model’s ability to plan beyond several tokens (Bachmann & Nagarajan, 2024). This
428 issue motivates the exploration of alternative or auxiliary training objectives.429
430 An enhancement to NTP is Multi-Token Prediction (MTP) (Gloekle et al., 2024), an auxiliary objective designed to
431 improve a model’s look-ahead capabilities. MTP adds parallel heads that predict several future tokens simultaneously.
432 This approach has shown significant improvements on planning-intensive tasks such as coding and has been adopted
433 in state-of-the-art models like DeepSeek-V3 (DeepSeek-AI et al., 2024) and Ling-V2 (inclusionAI, 2025). As a
434 complementary feature, MTP can also be used to speed up inference through self-speculative decoding.

432 The MTP framework has several variants. DeepSeek-V3 uses a sequential prediction mechanism with a small look-
 433 ahead window ($N=2$) to enhance decoding efficiency (DeepSeek-AI et al., 2024). Other approaches for multi-token
 434 awareness include converting NTP models to MTP models using register tokens (Gerontopoulos et al., 2025) and
 435 exploring parallel reasoning in a continuous space (Gozeten et al., 2025). Ahn et al. (2025) proposes to predict the
 436 joint probability of future tokens by carefully bottlenecking the architecture of the MTP heads.
 437

438 7 LIMITATIONS

439
 440 Due to limitations in compute and time, we are unable to pretrain on more data or larger models. The 7B models
 441 require 2 weeks of training time each on the 8xH200 node available to us. While our work demonstrates the potential
 442 of token order prediction for LLM pretraining, it remains to be seen whether TOP scales well to the standard of larger
 443 models and longer training runs of today.
 444

445 8 CONCLUSION

446 In this paper, we propose Token Order Prediction (TOP) as a novel auxiliary training loss for LLM pretraining. Our
 447 approach addresses some limitations of Multi-Token Prediction (MTP) by replacing the difficult task of exact future
 448 token prediction with the more tractable objective of ranking upcoming tokens by their proximity. TOP requires only
 449 a single additional unembedding layer compared to MTP’s multiple transformer layers, making it more parameter-
 450 efficient and scalable.
 451

452 Based on the results of our general language modeling experiments across three model sizes (340M, 1.8B, and 7B
 453 parameters), TOP overall improves performance over NTP, MTP, and DS-MTP on standard NLP benchmarks. The
 454 method shows positive gains as parameter count grows, suggesting its potential value for larger-scale language models.
 455 Additionally, we further verify the power of the TOP objective by evaluating on the synthetic star graph pathfinding
 456 task. Here, TOP learns the correct look-ahead solution on graphs where NTP and MTP do not. Despite not being as
 457 effective as MTP for self-speculative decoding, these preliminary results indicate that TOP offers another promising
 458 direction for improving language model training through effective auxiliary objectives.
 459

460 REPRODUCIBILITY

461 To reproduce all of our experiments we provide the code repositories in the supplementary materials. The repositories
 462 include instructions on what commands to run and the configurations of each training run. We have detailed the needed
 463 datasets, training hyperparameters, and configurations for the experiments sections 5.1, 5.2, 5.3, and more specifically
 464 in Table 2. All datasets used are publicly accessible or readily able to be synthetically generated using the code. All
 465 trained models are also uploaded to public repositories for further research.
 466

467 468 REFERENCES

469 Kwangjun Ahn, Alex Lamb, and John Langford. Efficient joint prediction of multiple future tokens, 2025.

470 Gregor Bachmann and Vaishnav Nagarajan. The pitfalls of next-token prediction, 2024.

471 Yonatan Bisk, Rowan Zellers, Ronan Le Bras, Jianfeng Gao, and Yejin Choi. PIQA: Reasoning about physical
 472 commonsense in natural language. In *Thirty-Fourth AAAI Conference on Artificial Intelligence*, 2020.

473 Sebastian Borgeaud, Arthur Mensch, Jordan Hoffmann, Trevor Cai, Eliza Rutherford, Kathrin Millican, George
 474 van den Driessche, Jean-Baptiste Lespiau, Bogdan Damoc, Aidan Clark, et al. Improving language models by
 475 retrieving from trillions of tokens. In *Proceedings of the 39th International Conference on Machine Learning*
 476 (*ICML*), pp. 2206–2240, 2022.

477 Zhe Cao, Tao Qin, Tie-Yan Liu, Ming-Feng Tsai, and Hang Li. Learning to rank: from pairwise approach to listwise
 478 approach. In *Proceedings of the 24th International Conference on Machine Learning*, ICML ’07, pp. 129–136,
 479 New York, NY, USA, 2007. Association for Computing Machinery. ISBN 9781595937933. doi: 10.1145/1273496.
 480 1273513. URL <https://doi.org/10.1145/1273496.1273513>.

481 Kevin Clark, Minh-Thang Luong, Quoc V. Le, and Christopher D. Manning. ELECTRA: Pre-training text encoders
 482 as discriminators rather than generators. In *International Conference on Learning Representations (ICLR)*, 2020.

486 Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord.
 487 Think you have solved question answering? try arc, the ai2 reasoning challenge. *ArXiv*, abs/1803.05457, 2018.
 488

489 DeepSeek-AI, Aixin Liu, Bei Feng, Bing Xue, Bingxuan Wang, Bochao Wu, Chengda Lu, Chenggang Zhao, Chengqi
 490 Deng, Chenyu Zhang, Chong Ruan, Damai Dai, Daya Guo, Dejian Yang, Deli Chen, Dongjie Ji, Erhang Li, Fangyun
 491 Lin, Fucong Dai, Fuli Luo, Guangbo Hao, Guanting Chen, Guowei Li, H. Zhang, Han Bao, Hanwei Xu, Haocheng
 492 Wang, Haowei Zhang, Honghui Ding, Huajian Xin, Huazuo Gao, Hui Li, Hui Qu, J. L. Cai, Jian Liang, Jianzhong
 493 Guo, Jiaqi Ni, Jiashi Li, Jiawei Wang, Jin Chen, Jingchang Chen, Jingyang Yuan, Junjie Qiu, Junlong Li, Junxiao
 494 Song, Kai Dong, Kai Hu, Kaige Gao, Kang Guan, Kexin Huang, Kuai Yu, Lean Wang, Lecong Zhang, Lei Xu,
 495 Leyi Xia, Liang Zhao, Litong Wang, Liyue Zhang, Meng Li, Miaojun Wang, Mingchuan Zhang, Minghua Zhang,
 496 Minghui Tang, Mingming Li, Ning Tian, Panpan Huang, Peiyi Wang, Peng Zhang, Qiancheng Wang, Qihao Zhu,
 497 Qinyu Chen, Qiushi Du, R. J. Chen, R. L. Jin, Ruiqi Ge, Ruisong Zhang, Ruizhe Pan, Runji Wang, Runxin Xu,
 498 Ruoyu Zhang, Ruyi Chen, S. S. Li, Shanghao Lu, Shangyan Zhou, Shanhua Chen, Shaoqing Wu, Shengfeng Ye,
 499 Shengfeng Ye, Shirong Ma, Shiyu Wang, Shuang Zhou, Shuiping Yu, Shunfeng Zhou, Shuting Pan, T. Wang, Tao
 500 Yun, Tian Pei, Tianyu Sun, W. L. Xiao, Wangding Zeng, Wanja Zhao, Wei An, Wen Liu, Wenfeng Liang, Wenjun
 501 Gao, Wenqin Yu, Wentao Zhang, X. Q. Li, Xiangyue Jin, Xianzu Wang, Xiao Bi, Xiaodong Liu, Xiaohan Wang,
 502 Xiaojin Shen, Xiaokang Chen, Xiaokang Zhang, Xiaosha Chen, Xiaotao Nie, Xiaowen Sun, Xiaoxiang Wang, Xin
 503 Cheng, Xin Liu, Xin Xie, Xingchao Liu, Xingkai Yu, Xinnan Song, Xinxia Shan, Xinyi Zhou, Xinyu Yang, Xinyuan
 504 Li, Xuecheng Su, Xuheng Lin, Y. K. Li, Y. Q. Wang, Y. X. Wei, Y. X. Zhu, Yang Zhang, Yanhong Xu, Yanhong Xu,
 505 Yanping Huang, Yao Li, Yao Zhao, Yaofeng Sun, Yaohui Li, Yaohui Wang, Yi Yu, Yi Zheng, Yichao Zhang, Yifan
 506 Shi, Yiliang Xiong, Ying He, Ying Tang, Yishi Piao, Yisong Wang, Yixuan Tan, Yiyang Ma, Yiyuan Liu, Yongqiang
 507 Guo, Yu Wu, Yuan Ou, Yuchen Zhu, Yuduan Wang, Yue Gong, Yuheng Zou, Yujia He, Yukun Zha, Yunfan Xiong,
 508 Yunxian Ma, Yuting Yan, Yuxiang Luo, Yuxiang You, Yuxuan Liu, Yuyang Zhou, Z. F. Wu, Z. Z. Ren, Zehui Ren,
 509 Zhangli Sha, Zhe Fu, Zhean Xu, Zhen Huang, Zhen Zhang, Zhenda Xie, Zhengyan Zhang, Zhewen Hao, Zhibin
 510 Gou, Zhicheng Ma, Zhigang Yan, Zhihong Shao, Zhipeng Xu, Zhiyu Wu, Zhongyu Zhang, Zhuoshu Li, Zihui Gu,
 511 Zijia Zhu, Zijun Liu, Zilin Li, Ziwei Xie, Ziyang Song, Ziyi Gao, and Zizheng Pan. DeepSeek-V3 technical report,
 512 2024.

513 Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of deep bidirectional
 514 transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of
 515 the Association for Computational Linguistics: Human Language Technologies*, pp. 4171–4186, 2019.

516 Anastasios Gerontopoulos, Spyros Gidaris, and Nikos Komodakis. Multi-Token Prediction Needs Registers, May
 517 2025. URL <http://arxiv.org/abs/2505.10518>. arXiv:2505.10518 [cs].

518 Fabian Gloeckle, Badr Youbi Idrissi, Baptiste Rozière, David Lopez-Paz, and Gabriel Synnaeve. Better & faster large
 519 language models via multi-token prediction, 2024.

520 Halil Alperen Gozeten, M. Emrullah Ildiz, Xuechen Zhang, Hravir Harutyunyan, Ankit Singh Rawat, and Samet
 521 Oymak. Continuous Chain of Thought Enables Parallel Exploration and Reasoning, May 2025. URL <http://arxiv.org/abs/2505.23648> [cs].

522 inclusionAI. Ling-v2. <https://github.com/inclusionAI/Ling-V2>, 2025. MIT License.

523 Mandar Joshi, Eunsol Choi, Daniel S. Weld, and Luke Zettlemoyer. TriviaQA: A large scale distantly supervised
 524 challenge dataset for reading comprehension. In *Proceedings of the 55th Annual Meeting of the Association for
 525 Computational Linguistics*, Vancouver, Canada, July 2017. Association for Computational Linguistics.

526 Tom Kwiatkowski, Jennimaria Palomaki, Olivia Redfield, Michael Collins, Ankur Parikh, Chris Alberti, Danielle
 527 Epstein, Illia Polosukhin, Matthew Kelcey, Jacob Devlin, Kenton Lee, Kristina N. Toutanova, Llion Jones, Ming-
 528 Wei Chang, Andrew Dai, Jakob Uszkoreit, Quoc Le, and Slav Petrov. Natural questions: a benchmark for question
 529 answering research. *Transactions of the Association of Computational Linguistics*, 2019.

530 Yann LeCun. Do large language models need sensory grounding for meaning and understanding? University Lecture,
 531 2024.

532 Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoy-
 533 anov, and Luke Zettlemoyer. BART: Denoising sequence-to-sequence pre-training for natural language generation,
 534 translation, and comprehension. In *Proceedings of the 58th Annual Meeting of the Association for Computational
 535 Linguistics*, pp. 7871–7880, 2020.

536 Anton Lozhkov, Loubna Ben Allal, Leandro von Werra, and Thomas Wolf. Fineweb-edu: the finest collection of edu-
 537 cational content, 2024. URL <https://huggingface.co/datasets/HuggingFaceFW/fineweb-edu>.

540 Denis Paperno, Germán Kruszewski, Angeliki Lazaridou, Quan Ngoc Pham, Raffaella Bernardi, Sandro Pezzelle,
 541 Marco Baroni, Gemma Boleda, and Raquel Fernández. The LAMBADA dataset, Aug 2016.

542

543 Przemyslaw Pobrotyn, Tomasz Bartczak, Mikolaj Synowiec, Radoslaw Bialobrzeski, and Jaroslaw Bojar. Context-
 544 aware learning to rank with self-attention. *ArXiv*, abs/2005.10084, 2020.

545 Weizhen Qi, Yu Yan, Yeyun Gong, Dayiheng Liu, Nan Duan, Jiusheng Chen, Ruofei Zhang, and Ming Zhou. Prophet-
 546 Net: Predicting future n-gram for sequence-to-sequence pre-training, 2020.

547

548 Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li,
 549 and Peter J. Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. In *Journal of
 550 Machine Learning Research*, volume 21, pp. 1–67, 2020.

551 Maarten Sap, Hannah Rashkin, Derek Chen, Ronan Le Bras, and Yejin Choi. Social IQa: Commonsense reasoning
 552 about social interactions. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language
 553 Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pp.
 554 4463–4473, 2019.

555

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

558

559 Yi Tay, Mostafa Dehghani, Vinh Q. Tran, Xavier Garcia, Jason Wei, Xuezhi Wang, Hyung Won Chung, Siamak
 560 Shakeri, Dara Bahri, Tal Schuster, et al. UL2: Unifying language learning paradigms. In *International Conference
 561 on Learning Representations (ICLR)*, 2022.

562

563 Johannes Welbl, Nelson F. Liu, and Matt Gardner. Crowdsourcing multiple choice science questions. In
 564 *NUT@EMNLP*, 2017.

565

566 Songlin Yang and Yu Zhang. FLA: A triton-based library for hardware-efficient implementations of linear attention
 567 mechanism, January 2024. URL <https://github.com/fla-org/flash-linear-attention>.

568

569 Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Ruslan Salakhutdinov, and Quoc V. Le. XLNet: Generalized
 570 autoregressive pretraining for language understanding. In *Advances in Neural Information Processing Systems
 571 (NeurIPS)*, pp. 5753–5763, 2019.

572

573

574

575

576

577

578

579

580

581

582

583

584

585

586

587

588

589

590

591

592

593