RankGen: Improving Text Generation with Large Ranking Models

Kalpesh Krishna♠ * **Yapei Chang**♠ **John Wieting**◇ **Mohit Iyyer**♠

♠University of Massachusetts Amherst, ◇Google Research
{kalpesh,miyyer}@cs.umass.edu
jwieting@google.com

Abstract

Given an input sequence (or *prefix*), modern language models often assign high probabilities to output sequences that are repetitive, incoherent, or irrelevant to the prefix; as such, model-generated text also contains such artifacts. To address these issues, we present RankGen, an encoder model (1.2B parameters) that scores model generations given a prefix. RankGen can be flexibly incorporated as a scoring function in beam search and used to decode from any pretrained language model. We train RankGen using large-scale contrastive learning to map a prefix close to the ground-truth sequence that follows it and far away from two types of negatives: (1) random sequences from the same document as the prefix, and (2) sequences generated from a large language model conditioned on the prefix. Experiments across four different language models (345M-11B parameters) and two domains show that RankGen significantly outperforms decoding algorithms like nucleus, top-k, and typical sampling on both automatic metrics (85.0 vs 77.3 MAUVE) as well as human evaluations with English writers (74.5% human preference over nucleus sampling). Analysis reveals that RankGen outputs are more relevant to the prefix and improve continuity and coherence compared to baselines. We open source our model checkpoints, code, and human preferences with detailed explanations for future research.[1]

1 Introduction

Despite exciting recent progress in large-scale language modeling (Radford et al., 2019; Brown et al., 2020), text generated from these language models (LMs) continues to be riddled with artifacts. Modern LMs suffer from the "likelihood trap" (See et al., 2019; Zhang et al., 2021), in which high likelihood (low perplexity) sequences produced by greedy decoding or beam search tend to be dull and repetitive. While truncated sampling methods such as top-k (Fan et al., 2018), nucleus (Holtzman et al., 2020), and typical sampling (Meister et al., 2022) alleviate these issues, they can also produce text with inconsistencies, hallucinations, factual errors, or commonsense issues (Massarelli et al., 2020; Dou et al., 2022; Krishna et al., 2021).

Part of the problem is that LMs are trained using "teacher forcing", where they are always given the ground-truth prefix[2] and asked to predict the next token. At test-time, however, the prefix can contain model-generated text, allowing errors to propagate during decoding (Bengio et al., 2015). This issue, combined with the observation that LMs overly rely on *local* context (Khandelwal et al., 2018; Sun et al., 2021), results in LMs generating sequences that break coherence or consistency within a larger discourse-level context (Wang et al., 2022).

To address this issue, we present RankGen, a 1.2 billion parameter encoder model that maps both human-written prefixes and model-generated continuations of those prefixes (generations) to a shared vector space. RankGen efficiently measures the compatibility between a given prefix and generations from any external LM by ranking the generations via their dot product with the prefix (Figure 2). We train RankGen using large-scale contrastive learning, encouraging prefixes to be closer to their gold continuation and far away from incorrect negatives. Since our objective considers two *sequences* rather than just single token prediction, it encourages RankGen to consider longer-distance relationships between the prefix and continuation rather than just rely on local context.

We devise two different strategies (shown in Fig-

[1] All resources for the paper to be available at https://github.com/martiansideofthemoon/rankgen

*Work done as a student researcher at Google Research.

[2] A *prefix* is a sequence of tokens fed as input to an LM, which then generates continuations conditioned on the prefix. A prefix is also called a *prompt* in prior work (Fan et al., 2018).

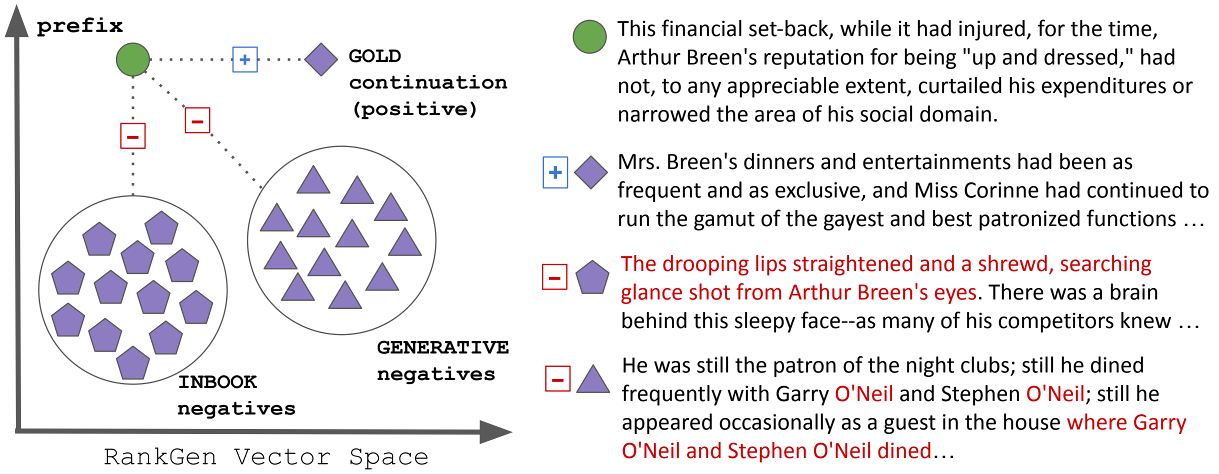

Figure 1: A datapoint from the novel "*Peter*" (Smith, 1911) used to train RANKGEN with contrastive learning. The prefix vector is pushed towards the gold continuation and away from the vectors of several incorrect continuation with errors (shown in red). These negative samples are either human-written **INBOOK** sequences taken from random locations in the same document (fluent and sometimes topically-similar, but irrelevant and incoherent), or **GENERATIVE** samples from a pretrained LM (relevant, but potentially containing hallucination or repetition).

ure 1) for selecting challenging negative samples, and empirically show that current large LMs cannot distinguish gold continuations from the negatives via perplexity (Section 2.1). In the first strategy, **INBOOK**, we select random sequences that occur within the same document as the prefix. While these human-written negatives are fluent and might contain topic or entity overlap, they are irrelevant as continuations to the prefix. In the second strategy, **GENERATIVE**, we generate continuations by conditioning a large pretrained LM on a given prefix. Compared to INBOOK negatives, these negatives are much more relevant to the prefix, but they suffer from issues like hallucination and repetition.

While RANKGEN can be easily used to rerank full-length samples from any external LM, we demonstrate further improvements in generation quality when it is integrated as a scoring function into beam search. On automatic and human evaluations across four large pretrained models (345M to 11B parameters) and two datasets, we observe that RANKGEN significantly and consistently outperforms sampling-based methods (nucleus, typical, top-k) as well as perplexity-based reranking (85.0 vs 77.3 MAUVE, 74.5% human preference over nucleus sampling[3]). Qualitative analysis from our human annotators (English writers) suggests that most of the improvements stem from increased relevance and continuity between the generated text and the prefix. Finally, we explore applications of our RANKGEN retriever outside of text generation and report state-of-the-art results on two complex literary retrieval benchmarks: RELiC (Thai et al., 2022) and ChapterBreak (Sun et al., 2022). We open source code with multiple RANKGEN checkpoints of different model sizes for future research.[1]

2 RANKGEN: a generation ranker

RANKGEN is a deep encoder network that projects prefixes and generations to a shared vector space. Given a prefix vector and a generation vector, we compute a *score* for the generation via the dot product between the two vectors. To ensure that these scores are meaningful, we train RANKGEN using large-scale contrastive learning (Radford et al., 2021), pushing the prefix vector close to the gold completion and away from the vectors of negative samples (Figure 1). We use two types of negative samples for learning the metric space: (1) sequences at random locations in the same document (INBOOK), and (2) model generations (GENERATIVE). The rest of this section first empirically justifies our choice of negative samples (Section 2.1) before presenting a more precise RANKGEN formulation (Section 2.2).

2.1 Language models cannot distinguish gold continuations from negatives

We explicitly choose our RANKGEN negatives to focus on a weak point of modern language models: namely, their tendency to assign high probabilities

[3]See Table 3, 4 for all results. MAUVE (Pillutla et al., 2021) is a recently introduced automatic metric for open-ended generation which has high correlation with human judgements.

InBook neg type →	Random		Hard	
	PG19	Wiki	PG19	Wiki
Random	50.0	50.0	50.0	50.0
Unigram Overlap	79.4	69.1	55.9	51.6
GPT2-medium (2019)	69.8	61.8	53.2	50.3
GPT2-XL (2019)	72.3	63.2	54.5	50.7
T5-base (f.t. PG19)	73.0	64.0	54.0	50.5
T5-XXL (f.t. PG19)	79.6	68.6	58.5	53.1
T5-XXL-C4 (2021)	76.4	66.2	57.4	52.2
GPT3 170B* (2020)	78.2	67.7	62.8	63.5
RankGen (ours)				
PG-XL-InBook	**99.1**	92.7	77.4	72.0
PG-XL-Generative	80.2	68.3	52.5	53.5
PG-XL-both	**99.1**	92.3	78.0	71.4
all-XL-both	98.7	**97.3**	61.3†	77.2†
Humans	94.5	91.0	**82.0**	**90.5**

Table 1: Given a prefix, how often do different models prefer the gold continuation over an InBook negative (i.e., a sequence from a different location in the same document)? Overall, large LMs (via perplexity) perform poorly compared to both our RankGen model as well as humans. See C.1 for results with multiple InBook negatives. *GPT3 scores computed on 1000 datapoints; †hard sets adversarially built with this model.

Discriminator	PG19	Wikipedia	Average
Random	50.0	50.0	50.0
Unigram Overlap	40.2	44.4	42.3
GPT2-medium (2019)	14.7	23.3	19.0
GPT2-XL (2019)	21.5	31.5	26.5
T5-XXL (f.t. PG19)	32.4	33.7	33.1
T5-XXL-C4 (2021)	19.0	39.1	29.1
RankGen (ours)			
PG-XL-Generative	**94.7**	**89.2**	**91.9**
PG-XL-InBook	69.8	59.7	64.8
PG-XL-both	92.0	74.9	83.5
all-XL-both	86.2	81.3	83.7

Table 2: Given a prefix, how often do different models prefer the gold continuation over a Generative negative (i.e., a model-generated continuation of the prefix)? LM perplexity strongly prefers Generative over gold continuations, while RankGen is able to accurately distinguish the gold. Negatives were generated from all four GPT2 and T5-XXL checkpoints using nucleus sampling (Holtzman et al., 2020) with $p = 0.9$ and then pooled (Appendix C.2 breaks down scores by LM).

to implausible or irrelevant continuations of a prefix. In particular, we demonstrate that even large LMs fail to consistently assign higher probability to gold continuations over InBook and Generative negatives.

InBook negatives: Our first type of negative samples are sequences from random locations in the same document as the prefix, whose lengths are controlled to match those of the ground-truth continuations. As these negatives are written by humans, they are always fluent and coherent, and they are often topically similar to the prefix and contain overlapping entities. However, they are irrelevant to the prefix, and as such they break discourse-level continuity and coherence (Hobbs, 1979; Grosz et al., 1995).

LMs struggle to distinguish gold continuations from InBook negatives: Given a prefix of 256 tokens from Wikipedia or a book in the PG19 dataset (Rae et al., 2019), we measure how often LMs assign higher probability (lower perplexity) to the gold 128-token continuation over a single InBook negative.[4] We break all prefixes and continuations at sentence boundaries to make the task less reliant on local syntactic patterns. Table 1 shows that even large LMs like GPT2-XL and GPT-3 perform poorly at this task and well below human[5] estimates (for 2-way classification with GPT2-XL, 72.3% vs 94.5% on PG19, 63.2% vs 91.0% on Wikipedia). While we randomly selected the InBook negative in these experiments, we could select a harder negative from the document using a trained RankGen. Specifically, we use RankGen to score the compatibility of each sequence in the document to the prefix, and take the highest scoring sequence that is not the gold continuation ("Hard" negative). Table 1 shows that LM performance drops significantly and is barely above chance (50.7% for GPT2-XL on Wikipedia), while human performance continues to be high (90.5%). We hypothesize that LMs perform poorly because (1) they overly focus on local context instead of long-range dependencies from the prefix (Khandelwal et al., 2018; Sun et al., 2021); and (2) LMs prefer words with high frequency in their training data (Holtzman et al., 2021) which may occur in InBook but not in the gold continuation.

LMs also struggle to distinguish gold continuations from Generative negatives: Our second type of negative samples are continuations to a prefix that are generated by a pretrained language model. Machine-generated text is known to differ significantly from human text: specifically, LM

[4]We experiment with more than one InBook negative in Appendix C.1. Also note that this task is similar to suffix identification benchmarks like ROCStories (2016); see Section 5.2 for experiments on them.

[5]Human study done on Upwork; details in Appendix B.

outputs contain repetitions (Holtzman et al., 2020), hallucinations (Maynez et al., 2020), and other artifacts (Zellers et al., 2019b). We choose GENERATIVE negatives to encourage RANKGEN to prefer generations closer to the human distribution, similar in spirit to GAN discriminators (Goodfellow et al., 2014). GENERATIVE negatives have also been used in previous energy-based language models (Deng et al., 2020), although not at this scale; see Section 6 for more related work. In Table 2, we show that language model perplexity is poor at identifying human text over GENERATIVE negatives. For example, GPT2-XL obtains just 26.5% average binary accuracy (well below random chance of 50%) across two domains. LMs tends to have high confidence in machine-generated text (Gehrmann et al., 2019), especially its own (Appendix C.2).

2.2 Training RANKGEN

Having motivated our negative sampling strategies, we now describe RANKGEN's training process. We train RANKGEN using large-scale contrastive learning with in-batch negative sampling, which is a popular metric learning technique (Sohn, 2016) previously used for dense retrieval (DPR, Karpukhin et al., 2020), image classification (SimCLR, Chen et al., 2020), and multimodal representation learning (CLIP, Radford et al., 2021).

A single RANKGEN training instance consists of a triple (p_i, c_i, g_i), where p_i is a prefix, c_i is the ground-truth continuation of that prefix, and g_i is a continuation generated by an LM. We prepend a special token (`pre`) to each prefix, and we also prepend a special token (`suf` for *suffix*) to each continuation and generation. We then pass each element of the triple through a shared Transformer encoder (Vaswani et al., 2017), projecting them to fixed-size vectors ($\mathbf{p}_i$, $\mathbf{c}_i$, $\mathbf{g}_i$) using the representation of the corresponding special token. To train this model, we use a contrastive objective that pushes the prefix vector $\mathbf{p}_i$ close to the gold continuation vector $\mathbf{c}_i$, but away from both the generation vector $\mathbf{g}_i$ as well as all other continuation vectors $\mathbf{c}_j$ in the same minibatch ("in-batch negative sampling"). Concretely,

$$Z(\mathbf{p}_i) = \sum_{c_j \in B} \exp \mathbf{p}_i \cdot \mathbf{c}_j + \sum_{g_j \in B} \exp \mathbf{p}_i \cdot \mathbf{g}_j$$

$$P(c_i|p_i) = \exp(\mathbf{p}_i \cdot \mathbf{c}_i) \ / \ Z(\mathbf{p}_i)$$

$$\text{loss} = - \sum_{(p_i, c_i) \in \mathcal{B}} \log P(c_i|p_i)$$

where $\mathcal{B}$ is a minibatch. All minibatch elements are sampled from the *same document*, which provides the INBOOK negatives. Note that the minibatch size $|\mathcal{B}|$ is an important hyperparameter since it determines the number of negative samples; we set $|\mathcal{B}| = 1536$ for our XL variant. We provide the training details of model variants in Appendix A.1.

Dataset construction: We consider all possible 256-word prefixes p_i in our document, ensuring that prefixes begin and end at sentence boundaries. We then select continuations c_i of variable length (10-128 words long) for each prefix p_i so that RANKGEN can re-rank candidates of different lengths at test-time. To produce GENERATIVE negatives, we first use 50% of our (p_i, c_i) training data pairs to fine-tune T5-XXL (Raffel et al., 2020) for causal language modeling (one per domain). For the remaining half of the dataset, we use this LM to generate a single continuation g_i to the prefix p_i of variable length (10-128 words) using nucleus sampling (Holtzman et al., 2020) with $p = 0.9$.

2.3 Using RANKGEN at inference

After model training, the dot product between the RANKGEN vectors of a prefix and a candidate continuation denotes their compatibility score. How can we integrate these scores into a decoding algorithm for text generation? We experiment with two strategies: (1) over-generation and reranking, in which we use any pretrained LM and existing decoding algorithm to generate multiple samples (20 in our experiments) and then re-rank them via RANKGEN scores; and (2) beam search (Figure 2), in which we generate N samples of length L via nucleus or ancestral sampling, compute the top B highest-scoring samples, and concatenate them to the prefix to continue generation. Note that unlike standard beam search decoding, in which the scores are the LM's predicted probabilities, here we use scores computed by RANKGEN. There are three hyperparameters for our beam search during inference: (i) the rerank length L, or the number of tokens generated before each re-ranking; (ii) the beam size B; and (iii) the number of samples generated per beam N. Setting N=20, B=1, L=128 (maximum generation length) is equivalent to the first decoding strategy of over-generation and reranking. We provide implementation details and specifics of our hyperparameter grid search in Appendix A.2, A.3. Overall we found N=10, B=2, L=20 performs best, but improvements over base-

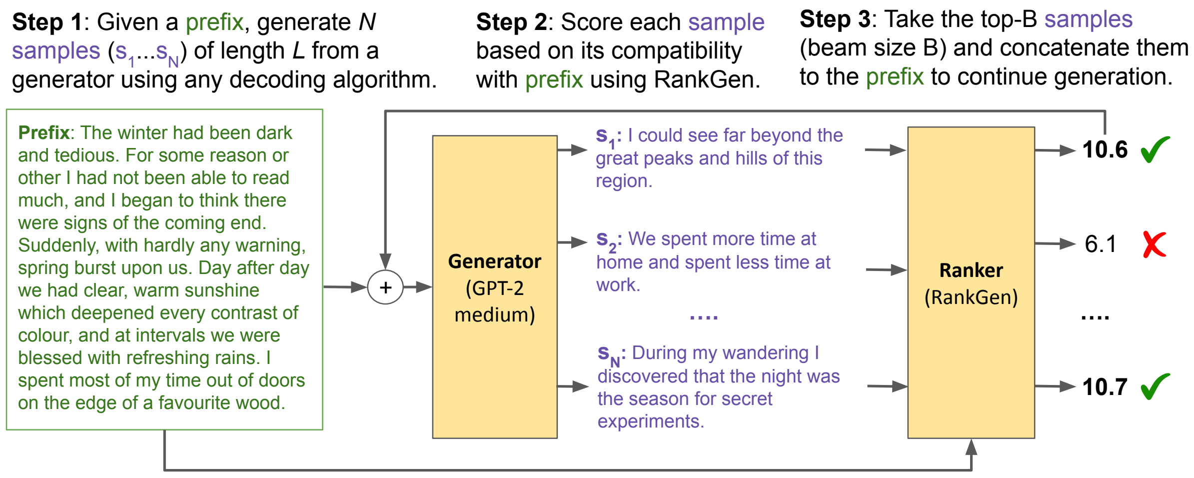

Figure 2: The RANKGEN setup during inference. RANKGEN can be flexibly plugged into any generative model (like GPT2) using any decoding algorithm (like nucleus sampling) during inference in a beam-search like setup. The examples shown here are actual generations from GPT2-md (with nucleus p=0.9) and scores from RANKGEN.

lines are consistent across hyperparameter choices.

3 Experiments

We compare RANKGEN to existing decoding algorithms (e.g., nucleus sampling, greedy decoding) across a wide range of pretrained LM decoders and RANKGEN configurations. In our experiments, we provide a human-written prefix to a pretrained LM and decode continuations from the model using RANKGEN as well as competing approaches. Our automatic evaluations measure the quality of the model-generated continuations using the MAUVE metric proposed by Pillutla et al. (2021), on which RANKGEN outperforms all competing decoding approaches when used as a re-ranker of multiple samples. Furthermore, integrating RANKGEN into beam search yields the highest-quality outputs, measured both by MAUVE and human A/B tests. This section details our experimental setup and then highlights the main takeaways of our results.

3.1 Model configurations

RANKGEN variants: We study four configurations of RANKGEN, each of which has 1.2B parameters (XL size) and is trained with a minibatch size of 1536. Three of our models are trained on the PG19 dataset (Rae et al., 2019), which consists of long-form books, using (1) only INBOOK negatives, (2) only GENERATIVE negatives, and (3) both types of negatives. Since PG-19 contains mainly historical literature, we also experiment with different data sources by training RANKGEN on the union of four domains ("all") — PG19, Wikipedia, C4-NewsLike and C4-WebTextLike (Raffel et al., 2020). This last model is trained using both INBOOK and GENERATIVE negatives. Additional ablations varying the model size and minibatch size (number of negatives) are provided in Appendix E.

Pretrained language models: Does RANKGEN improve generation quality regardless of the size (and pretraining dataset composition) of the base LM? To answer this question, we evaluate four different pretrained LMs whose sizes vary considerably from that of RANKGEN (1.5B parameters). We experiment with two variants of GPT-2 (Radford et al., 2019): GPT2-medium (345M parameters) and GPT2-XL (1.5B parameters). We also evaluate a pretrained T5-XXL-v1.1 (Raffel et al., 2020) model (11B parameters) that we fine-tune to perform language modeling on the training set of PG19 (Rae et al., 2019). Finally, to experiment with a large LM trained on out-of-domain data for RANKGEN, we also evaluate the T5-XXL model from Lester et al. (2021) (11B parameters) that was fine-tuned for language modeling on the C4 corpus.

3.2 Open-ended text generation

Following previous work on text generation (Welleck et al., 2019; Holtzman et al., 2020; Su et al., 2022), we primarily focus on open-ended text generation, which has wide applications for tasks such as generating stories (Fan et al., 2018), poetry (Zhang and Lapata, 2014), and dialog (Miller et al., 2017). Additionally, recent work on zero-shot and few-shot NLP (Brown

Decoding method	Language Model / Dataset								Average
	T5-XXL-C4		GPT2-md		GPT2-XL		T5-XXL-PG19		
	PG19	wiki	PG19	wiki	PG19	wiki	PG19	wiki	
Greedy decoding	6.6	15.2	3.8	11.2	6.4	18.3	23.4	38.5	15.4
Ancestral sampling	67.7	71.6	75.5	73.2	77.4	75.0	90.2	67.7	74.8
Nucleus, $p = 0.9$ (2020)	69.7	77.9	73.0	74.6	74.4	75.0	92.6	81.8	77.3
Top-k, $k = 40$ (2018)	68.3	77.3	74.8	73.4	76.0	75.2	92.2	81.8	77.4
Typical, $p = 0.9$ (2022)	69.5	77.4	73.2	73.5	73.6	76.4	92.7	81.1	77.1
Re-ranking 20 full-length ancestral samples									
RankGen PG19-XL-both	79.9	83.3	78.8	78.5	78.2	79.6	92.2	79.2	81.2
RankGen all-XL-both	71.0	85.8	**79.0**	84.9	**79.0**	86.4	92.1	82.9	82.6
Re-ranking 20 full-length nucleus samples									
Unigram overlap	65.6	80.7	74.8	78.7	73.9	79.4	93.6	90.6	79.7
LM perplexity	62.6	55.1	55.5	63.1	58.3	61.6	88.4	77.1	65.2
RankGen PG19-XL-Generative	78.3	82.4	76.2	73.8	76.2	73.0	**95.0**	87.1	80.2
RankGen PG19-XL-InBook	70.7	83.4	76.7	81.7	76.0	83.6	93.3	85.9	81.4
RankGen PG19-XL-both	**80.7**	86.4	76.3	79.4	75.2	81.3	**94.3**	87.3	82.6
RankGen all-XL-both	73.0	88.1	74.8	83.9	75.9	85.7	93.6	91.8	83.4
+ beam search (B=2, L=20, N=10)	74.0	**89.4**	76.2	**88.9**	77.0	**89.4**	92.2	**93.0**	**85.0**

Table 3: A comparison between RankGen variants and baseline decoding algorithms using MAUVE (Pillutla et al., 2021), an automatic text generation metric with high correlation to human judgments. RankGen significantly outperforms baselines like nucleus and typical sampling. Additionally, it outperforms other re-ranking strategies like those based on LM perplexity and unigram overlap. Incorporating RankGen into beam search (last row) results in the highest average MAUVE score.

et al., 2020; Chowdhery et al., 2022) relies heavily on decoding from pretrained language models. We consider **two domains** in our study: (1) prefixes from Wikipedia, and (2) literary text from PG19 (Rae et al., 2019). Because it is difficult to conduct human evaluations of long sequences of machine-generated text (Karpinska et al., 2021; Krishna et al., 2021), our main experiments consider a 256-token prefix and 128-token generations. We analyze generation quality given varying prefix lengths in Section 4.3.

Decoding algorithms: For each of the four pretrained LMs considered in our experiments, we decode outputs using greedy decoding, ancestral sampling, nucleus sampling (Holtzman et al., 2020), top-k sampling (Fan et al., 2018), and typical sampling (Meister et al., 2022). Since RankGen is fundamentally a re-ranker of multiple samples, we also compare to two other re-ranking methods that use LM perplexity and unigram overlap, respectively. In all re-ranking settings, we generate 20 nucleus or ancestral samples and then re-rank them with each method. For RankGen, we additionally have a beam search configuration that iteratively re-ranks partially generated hypotheses.

Automatic & human evaluation metrics: We use MAUVE (Pillutla et al., 2021) as our primary metric for automatic evaluation. MAUVE computes the similarity of the distribution of human-written text and machine-generated text, and it has been shown to have high correlation with human judgments. We provide details about our MAUVE setup in Appendix D.1. More automatic evaluations with token overlap methods such as REP (Welleck et al., 2020) are provided in Appendix D.3. Since automatic metrics are insufficient for text generation evaluation (Celikyilmaz et al., 2020), we also conduct a human evaluation by hiring English teachers and writers from Upwork;[6] see Appendix B for more details. For each of GPT2-medium and T5-XXL-C4 we choose 50 Wikipedia and 50 PG19 prefixes, and show *three* annotators a pair of continuations from different decoding strategies in a random order (blind A/B testing). Annotators are asked to choose the better continuation and provide a 1-3 sentence explanation for their choice. This gives us a total of 600 annotations, which we analyze in Section 3.4, 4.1.

3.3 Results from automatic evaluations

Table 3 contains MAUVE scores for all configurations of RankGen variants, pretrained LMs, decoding algorithms, and datasets studied in this work. Overall, we observe that:

[6] https://www.upwork.com

	PG19	Wiki	Overall
gpt2-md	80.0 (72.0)	82.0 (78.3)	81.0 (75.1)
t5-xxl-c4	68.0 (63.3)	68.0 (65.3)	68.0 (64.3)
Overall	74.0 (67.8)	75.0 (71.9)	74.5 (69.8)

Table 4: Percentage of instances for which English writers prefer RANKGEN outputs over nucleus samples in a blind A/B test. Scores shown are majority vote, with average accuracy in subscript. Overall, humans significantly prefer RANKGEN outputs ($p < 0.001$). Note that the identical PG19 / Wiki majority vote scores in T5 is coincidental. See Table 5 for agreement stats.

RANKGEN re-ranking and beam search significantly improve MAUVE: Re-ranking full-length samples with RANKGEN yields an average MAUVE score of 83.4 across all configurations, significantly outperforming other decoding strategies like greedy decoding (15.4), ancestral sampling (74.8), and nucleus / top-k / typical sampling (77.1-77.4). Adding beam search with best hyperparameters[7] further boosts average performance to 85.0 MAUVE. Surprisingly, re-ranking 20 full-length ancestral samples with RANKGEN performs better than standard nucleus sampling (77.3 vs 82.6). However, re-ranking 20 ancestral samples is slightly worse than re-ranking 20 nucleus samples (82.6 vs 83.4) due to worse inherent quality of ancestral samples compared to nucleus samples (74.8 vs 77.3). Re-ranking generations by unigram overlap to the prefix is a surprisingly good baseline (79.7), while re-ranking by LM perplexity reduces MAUVE to 65.2, since it emulates likelihood-based methods like greedy decoding. Finally, we observe that RANKGEN performs best on in-domain data, with the PG19-XL-both variant obtaining better scores (80.7 on T5-XXL-C4 with PG19) than the model trained on four domains (all-XL-both, 73.0).

INBOOK negatives help more than GENERATIVE, but using both maximizes MAUVE: In Table 3 (bottom), we perform ablations by removing the INBOOK and GENERATIVE for RANKGEN variants trained on PG19. All three variants outperform the nucleus sampling baseline (77.3), but keeping both objectives performs best (82.6). A model trained with only INBOOK samples is more effective (81.4) than one trained with only GENERATIVE samples (80.2).

[7]Hyperparameter grid search details in Appendix A.3.

	PG19	Wiki	Overall
GPT2-md	0.31, 48%	0.49, 60%	0.40, 54%
T5-XXL-C4	0.27, 46%	0.30, 48%	0.29, 47%
Overall	0.29, 47%	0.40, 54%	0.35, 51%

Table 5: Inter-annotator agreement statistics for the human evaluation in Table 4. Scores shown are Fleiss κ (Fleiss, 1971) and the % of instances with unanimous agreement among 3 annotators. Overall, we see moderate agreement, higher for GPT2-md and Wikipedia.

3.4 Human evaluation with A/B tests

Despite the effectiveness of metrics such as MAUVE, human evaluation remains critical for open-ended generation (Celikyilmaz et al., 2020; Gehrmann et al., 2022). Since human evaluation is expensive, we decide to focus on comparing our best performing RANKGEN variant (RANKGEN-XL-all with beam search) to nucleus sampling, which is one of the most popular decoding algorithms in use today. To determine if humans prefer continuations decoded via RANKGEN versus those from nucleus sampling, we conduct blind A/B testing by hiring English teachers / writers from the Upwork platform. Table 4 shows that humans significantly prefer outputs from RANKGEN over nucleus sampling (74.5% preference by majority vote, $p < 0.001$). We notice higher preference and better inter-annotator agreement (Table 5) for outputs from the smaller GPT2-medium (82% vs 68%). Additionally, humans show slightly higher preference and agreement for Wikipedia instances, consistent with our MAUVE evaluation (Table 3).

4 Analysis

Our experiments confirm that in terms of both MAUVE score and human A/B tests, continuations decoded using RANKGEN are of higher quality than those generated via other methods. In this section, we explore what properties of the generated text improve with RANKGEN, and we also examine its speed-quality trade-off.

4.1 In what ways does RANKGEN improve generation quality?

To get more insight into the human preference judgments made in Section 3.4, we also asked our annotators to provide a 1-3 sentence free-form explanation for each of their choices.[8] We manually categorized each of 600 explanations into nine broad

[8]All 600 human explanations are provided in our codebase.

Reasons relating the prefix with the generation (81%)	
More topically relevant to the prefix	37.7%
Better continuity / flow / chronology	31.6%
Does not contradict prefix	6.8%
Stylistically closer to prefix	4.7%
Reasons related only to the generated text (19%)	
Better commonsense understanding	8.0%
Less repetitive	4.7%
More grammatical	3.1%
Less contradictions	1.7%
More coherent / other	1.7%

Table 6: Distribution of reasons given by our human evaluators (English writers/teachers) for preferring RANKGEN outputs over nucleus samples. Relevance / continuity to prefix was a common explanation.

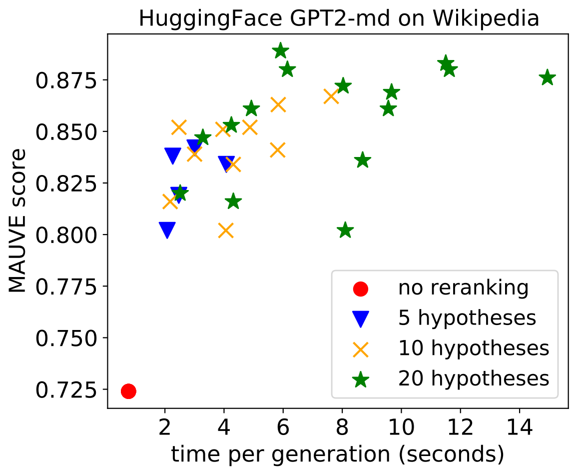

Figure 3: Performance / time trade-off for different RANKGEN hyperparameters (Appendix A.3). Overgeneration and RANKGEN re-ranking significantly improve MAUVE scores, but need an order of magnitude more time, primarily due to overgeneration (Table 7).

	HuggingFace (GPT2) medium	XL	T5X / seqio (T5) base	XXL
secs / gen	7.7e-1	2.9e0	8.1e-3	7.4e-2
RANKGEN calls in same time as one generation				
base	108.5	408.5	8.4	77.0
large	42.8	161.1	4.3	38.9
XL	16.4	61.7	1.7	15.7

Table 7: Number of possible RANKGEN calls in the same amount of time as one model generation. Across implementations (HuggingFace/T5X) and model sizes, RANKGEN needs only a fraction of time vs generation.

categories loosely based on the schema designed by Dou et al. (2022). In Table 6 we present the distribution of explanations whenever annotators preferred RANKGEN. Interestingly, 81% of the explanations mentioned some aspect of the relationship between the prefix and the generated text as the determining factor, including relevance, continuity, and stylistic similarity. 8.0% of the explanations said that RANKGEN outputs displayed fewer commonsense errors, while 4.7% said that they were less repetitive. We provide representative generations and corresponding explanations in Table 8 and more full-length generations in Appendix F.

4.2 How fast is decoding with RANKGEN?

Decoding with our method requires over-generation followed by re-ranking with RANKGEN. How much extra decoding time does this process add? In Figure 3, we show the trade-off between MAUVE score and decoding time across different beam search hyperparameters.[9] While decoding a single nucleus sample takes just 0.77 seconds, generating 20 samples followed by full-length sample re-ranking with RANKGEN requires 2.52 seconds. Our best-performing beam search configuration, which uses multiple re-ranking steps, needs 5.91 seconds.[10] We dig a little deeper into these numbers: is the extra compute time due to over-generation or RANKGEN re-ranking? In Table 7, we measure the time taken to generate and score an individual instance. Overall, we see that re-ranking with RANKGEN takes only a fraction of the time (1-10%) compared to generation, which means that over-generation tends to be the bottleneck. Thus, developing methods that avoid the need for over-generation (via distillation or RANKGEN gradient based sampling) is an exciting direction for future work.

4.3 How does RANKGEN work with different length prefixes?

Our RANKGEN model is trained with a fixed prefix length of 256 tokens, and all of the evaluations in Section 3 also assume a prefix length of 256 tokens. However, many text generation applications take shorter prefixes as input, like short writing prompts in story generation (Fan et al., 2018). How well does RANKGEN generalize to shorter and longer prefixes? Figure 4 compares nucleus sampling to RANKGEN across varying prefix lengths. We observe that RANKGEN consistently outperforms

[9]Decoding time depends on the library and hardware. We analyze two implementations - HuggingFace library on RTX3090, and T5X on TPU-v3; details in Appendix A.2.

[10]See Appendix A.3.2 for more plots on speed tradeoffs across hyperparameters.

Prefix	Generations	Annotator Preference
PG19, *Half a Life-time Ago*, by *Elizabeth Gaskell*: ... If thou doesn't choose to marry me on those terms–why! I can snap my fingers at thee, never fear. I'm not so far gone in love as that. But I will not have thee, if thou say'st in such a hectoring way that Willie must go out of the house–and the house his own too–before thou'lt set foot in it. Willie bides here, and I bide with him." "Thou hast may-be spoken a word too much," said Michael, pale with rage.	Text 1 (**Nucleus**): "How on earth could it be? He must be part of the marriage, my love. But he can't go away–he can't go away with me. And if he does go away, he'll have to pay ... Text 2 (**RankGen**): "Take heed how thou handlest thy words in the future." "I haven't spoken a word too much," she murmured, coolly. "I meant–I meant the last word!" "Swear it, thou hastna sworn a word yet. I'll go to talk with ...	**Text 2**. Text 1 has a completely different style, way more modern. First sentence in Text 2 fits the tone of enraged character, Michael. **Text 2**. The writing style is more similar, Text 1 sounds too modern. Plus, the atmosphere of the fragment is more consistent. The characters seem to be having an argument, so "My love" in doesn't quite make sense.
Wiki, *American Indian Movement*: ... named three lower-level AIM members involved in her death: Arlo Looking Cloud, John Graham, and Theda Nelson Clarke. This was the first time that an AIM leader active at the time of the Aquash murder had publicly accused AIM of having been involved. Looking Cloud and Graham were convicted of murder in 2004 and 2010, by federal and South Dakota state juries, respectively. By then Clark was being cared for in a nursing home and was not indicted.	Text 1 (**Nucleus**): Clarke was the second AIM leader to be charged. On July 19, 2009, her defense attorney, Michael Kranz had filed a motion ... His request for a new trial failed in December 2009. In 2009, the Los Angeles Times ... Text 2 (**RankGen**): Clarke has also denied any involvement in Aquash's murder. In the early months of 2001 the FBI began an effort to break through AIM's defenses, to try to identify and bring charges against all three AIM members...	**Text 2** - The last sentence of the prefix paragraph ("By then Clark was being cared for in a nursing home and was not indicted") flows well with Text 2, implying that Clarke was absolved of guilt. **Text 2**. Text 2 further goes into Clark's involvement in the case and Aquash's murder while Text 1 contradicts part of the original paragraph.
PG19, *The Fall of a Nation* by *Thomas Dixon*: The grim gray wave of destruction from the sand dunes had rolled into battle-line and spread out over the green clothed hills and valleys of the Island–swiftly, remorselessly, with an uncanny precision ... a puff of black smoke streamed downward and the distant officer, ... gunners of his battery. Our rifles cracked in vain. The birdmen laughed and paid no attention.	Text 1 (**RankGen**): They raced across the plains and away. Our artillery fell silent and rested. It would have to be our last salvo. "Are they coming down here?" shouted an American, as he watched ... Text 2 (**Nucleus**): With a bark of laughter, a group of strong men fell among the men and laughed with them. And with the general smile on his face he began to wave his finger in the air at them and ...	**Text 1**. The jolly atmosphere of Text 2 really doesn't fit with the prefix. The prefix read together with Text 2 has kind of a Monty Python vibe to it. **Text 1.** Chose the first one, because of the stark change in the tone in the second text, which has fair amount of laughter and sleeping during a battle.
Wiki, *Tim Richmond*: ... Richmond raced in a 1978 Mini Indy car event at Phoenix International Raceway, winning the Formula Super Vee support event in a Lola T620. The win attracted sponsors and attention from ... He also competed in USAC's Silver Crown series. Richmond's father bought an Eagle Indy Car chassis and an Offenhauser engine for the 1979 race at Michigan International Speedway. Richmond qualified 21st fastest with a lap, significantly slower than Bobby Unser's pole position speed.	Text 1 (**RankGen**): However, his effort earned him an invitation to join the 1979 Indy 500 at Phoenix International Raceway. After finishing sixth, Richmond was called upon to replace the injured Jimmy Corder. A rookie, Richmond began his race in a three-car pack ... Text 2 (**Nucleus**): In 1982 the pair switched to the SuperCar chassis that year. As a result of the change, Richmond's driving style evolved somewhat. At age 42 he returned to IndyCar. At the 1977 ...	**Text 1** - Chronology - the events of text 1 follow prefix text and are before the events of text 2. It makes sense the text would be written in the correct order. **Text 1** - Text 1 continues the idea of the final sentence of the prefix paragraph. Despite his unimpressive position in the qualifying race, his effort earned him an invitation into the Indy 500.

Table 8: Representative model generations using RankGen vs nucleus sampling (Holtzman et al., 2020), along with human explanations (from English teachers/writers) for preferring RankGen outputs. For every row the color coding is used to ground the annotator explanation in the prefix and generation. See Appendix F or our codebase (link on page 1) for several more *full-length* model generations and detailed human explanations.

nucleus sampling in terms of MAUVE, and beam search with RankGen always provides further gains, suggesting robustness to the prefix length.

5 Using RankGen for other tasks

While we designed RankGen specifically to improve text generation, we also find that it is an effective zero-shot retriever, despite being trained on specific negatives. This section compares the RankGen model to retriever and LM baselines on several retrieval and suffix identification tasks.

5.1 RankGen as a retriever

RankGen follows a dual encoder architecture similar to those of several recent dense retrieval approaches like DPR (Karpukhin et al., 2020) and REALM (Guu et al., 2020). How does our model fare on complex retrieval tasks? We test RankGen on RELiC (Thai et al., 2022), a complex retrieval

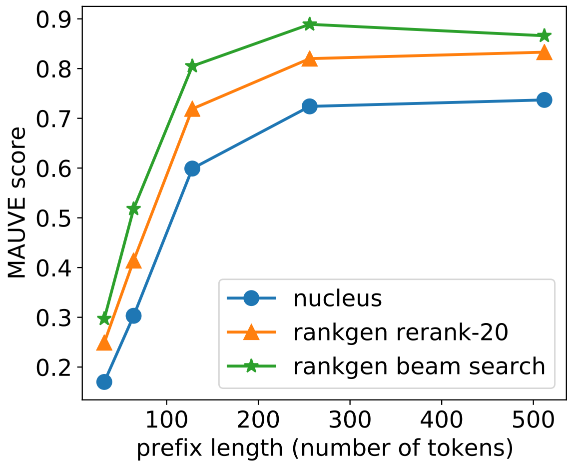

Figure 4: MAUVE score variation with change in prefix length for GPT2-medium on Wikipedia. Across prefix lengths re-ranking with RANKGEN-XL-all boosts performance, and using it in beam search does best.

Model	Recall@k (↑)				
	1	3	5	10	50
Random	0.0	0.1	0.1	0.2	1.2
BM25 (1995)	1.3	2.9	4.1	6.7	14.5
SIM (2019)	1.3	2.8	3.8	5.6	13.4
DPR (2020)	1.3	3.0	4.3	6.6	15.4
c-REALM (2021)	1.6	3.5	4.8	7.1	15.9
ColBERT (2020)	2.9	6.0	7.8	11.0	21.4
RANKGEN (ours)					
PG-XL-GEN	0.7	1.9	2.7	4.1	9.1
PG-XL-INBOOK	**6.0**	**12.2**	**15.4**	**20.7**	**37.3**
PG-base-both	3.8	8.2	10.8	15.4	31.6
PG-XL-both	4.5	8.4	11.0	15.1	27.9
all-XL-both	4.9	9.2	11.9	16.5	31.5
Supervised (upperbound)	9.4	18.3	24.0	32.4	51.3

Table 9: RANKGEN performance on RELiC (Thai et al., 2022) compared to other retrievers. Our model achieves state-of-the-art on the *zero-shot* RELiC setting, approaching the upperbound supervised model.

task for literary analysis. Given an excerpt of literary analysis, systems must retrieve a quotation from a novel which is most relevant to the context. RELiC requires a deep understanding of literary and linguistic phenomena (like irony, metaphors, co-reference, paraphrasing, and stylistics), and current retrievers struggle on it. Note that we use models in a **zero-shot** setting, without finetuning on RELiC training data. Also, as RANKGEN is partially trained on literary data from the PG-19 dataset, RELiC is in some ways an in-domain task.

RANKGEN is the best performing retriever on RELiC: Table 9 shows that RANKGEN models significantly outperform other retrievers and achieve a new state of the art on RELiC.[11] PG-XL-INBOOK performs best (6.0 vs 2.9 recall@1 against the next-best ColBERT), approaching the R@1 of a supervised upperbound (9.4). While our XL model has many more parameters than all baselines, even the smaller-sized PG-base-both outperforms all baselines (3.8 vs 2.9 R@1). Dropping INBOOK leads to poor performance (0.7 R@1 for GENERATIVE), further confirming the efficacy of INBOOK. Besides RELiC, we also investigate retrieval over PG19 books in Appendix C.1.

5.2 RANKGEN for suffix identification

RANKGEN is trained on a *suffix identification* objective: given a prefix, choose the gold continuation over INBOOK and GENERATIVE negatives. How well does RANKGEN learn this task? How does

[11]"Anonymous Razorbill" on the official leaderboard.

	ChapterBreak PG19	ChapterBreak AO3	StoryCloze 2016	StoryCloze 2018	HSw
prefix length	240.3	241.6	35.4	35.3	39.5
suffix length	152.9	156.1	7.4	7.4	26.0
Random	16.7	16.7	50.0	50.0	25.0
Token overlap	37.3	28.7	39.9	40.9	27.4
GPT2-md	20.8	21.3	66.0	66.0	37.0
GPT2-XL	20.8	22.9	71.2	72.2	47.9
T5-base-PG	23.2	23.4	59.0	61.9	33.1
T5-XXL-PG	28.6	25.3	69.3	73.5	62.3
T5-XXL-C4	24.1	24.3	76.0	**77.8**	63.6
GPT3 (170B)	26.0	23.8	83.2	-	78.9
PaLM (540B)	-	-	**84.6**	-	**83.4**
RANKGEN (1.2B, ours)					
PG-XL-GEN	33.6	21.8	57.9	57.9	35.0
PG-XL-INBK	**64.3**	**39.5**	73.4	72.6	39.3
PG-XL-both	63.5	36.9	71.1	72.6	40.7
all-XL-both	59.3	32.8	75.4	75.8	46.3

Table 10: Zero-shot suffix identification results on existing datasets. RANKGEN significantly outperforms all LMs on ChapterBreak which has long prefix/suffix lengths. RANKGEN performs similar to similar-sized GPT2-XL on StoryCloze and HellaSwag, with shorter inputs and more local dependencies.

RANKGEN fare on existing suffix identification benchmarks?

Performance on INBOOK / GENERATIVE: In Section 2.1 we motivated the RANKGEN design by showing the inability of LM perplexity to prefer the gold continuations over negatives. How does RANKGEN fare on these negatives? In Table 1 and Table 2 we evaluate the performance at distinguishing gold continuations from negatives, and compare RANKGEN to large LMs. Since RANKGEN is directly optimized on this objective, it significantly outperforms large LMs (99.1% vs 78.2% with GPT-3 for INBOOK). RANKGEN variants trained on just INBOOK or just GENERATIVE perform best at their respective tasks, but we observe some generalization (INBOOK model gets 69.8% on GENERATIVE PG19 negatives, GENERATIVE model gets 80.2% on INBOOK negatives, both higher than all LMs). Strong performance on GENERATIVE could have several applications like fake news detection (Zellers et al., 2019b; Gehrmann et al., 2019), and is an interesting future work direction.

Performance on existing suffix identification benchmarks: We test RANKGEN on three existing suffix identification datasets — ChapterBreak (Sun et al., 2022), ROCStories cloze test (Mostafazadeh et al., 2016) and HellaSwag (Zellers et al., 2019a); dataset details are provided in Appendix C.3. To measure their intrinsic capability, models are evaluated **zero-shot**, without finetuning on training sets.[12]

In Table 10 we find that RANKGEN significantly outperforms all LMs on ChapterBreak (64.3 vs 28.6). RANKGEN performs comparably to similar-sized GPT2-XL (1.5B parameters) on other tasks, beating it on StoryCloze (75.8 vs 72.2), but slightly worse on HellaSwag (46.3 vs 47.9). Much larger LMs like GPT3 170B (Brown et al., 2020) and PaLM 540B (Chowdhery et al., 2022) perform best on StoryCloze and HellaSwag. Scaling also benefits RANKGEN (30.4 vs 40.7 on HellaSwag for base vs XL), and we believe further scaling RANKGEN is a promising direction for future work. We also find INBOOK negatives are more beneficial than GENERATIVE negatives (64.3 vs 33.6 on ChapterBreak PG19). We hypothesize that the different trends on different datasets can be attributed to input length. As seen in Table 10, ChapterBreak has much longer inputs (240 prefix, 153 suffix tokens) than other datasets (35 prefix, 7 suffix tokens for ROCStories). The focus on local context in LMs (Khandelwal et al., 2018; Sharan et al., 2018; Sun et al., 2021) helps with short-range tasks but also likely contributes to their underperformance on complex long-range tasks like ChapterBreak.

[12]Zellers et al. (2019a) also describe *zero-shot* HellaSwag experiments, testing models on unseen WikiHow / ActivityNet categories; however they still finetune models on HellaSwag data for seen categories, while we do no such finetuning.

6 Related Work

Our INBOOK negative sampling is related to popular **self-supervised representation learning algorithms** utilizing multiple sentences in discourse, which have proven effective for learning sentence embeddings like SkipThought & FastSent (Kiros et al., 2015; Hill et al., 2016; Jernite et al., 2017). Our specific formulation is most similar to QuickThought (Logeswaran and Lee, 2018), who use in-batch negative sampling on a contiguous set of sentences. More recently, next sentence prediction has been used for pretraining BERT (Devlin et al., 2019) and ALBERT (Lan et al., 2020) and is effective for pretraining large language models (Aroca-Ouellette and Rudzicz, 2020). Unlike these works, the focus of our research is text generation rather than self-supervised pretraining for NLU tasks.

Our work is closely related to efforts in **energy-based methods** (LeCun et al., 2006) for generative modeling (Grover et al., 2019; Parshakova et al., 2019), speech recognition (Wang and Ou, 2018), open-ended text generation (Bakhtin et al., 2019; Deng et al., 2020), machine translation (Shen et al., 2004; Lee et al., 2021; Bhattacharyya et al., 2021), and constrained generation (Qin et al., 2022; Mireshghallah et al., 2022), and models for specific attributes like style (Dathathri et al., 2020; Yang and Klein, 2021), length (Li et al., 2017), or repetition & relevance (Holtzman et al., 2018). Unlike prior work, we use human-written text from the same document as negative samples (INBOOK) in addition to machine generated text, to create a general-purpose generation re-ranking model. RANKGEN is also trained at a much larger scale than prior energy-based models for text (1.2B parameters, large-scale contrastive learning with 3K negatives and 4 domains).

Finally our work is related to efforts in **modeling non-local dependencies** in generation, by predicting multiple tokens (Oord et al., 2018; Qi et al., 2020), using retrieval (Khandelwal et al., 2020), using bidirectional LMs (Serdyuk et al., 2018), contrastive learning for making isotropic LM representations (Su et al., 2022), using BERT for sentence-level language modeling (Ippolito et al., 2020), or sequence-level losses (Wiseman and Rush, 2016; Edunov et al., 2018; Welleck et al., 2020; Liu et al., 2022) for reducing exposure bias (Bengio et al., 2015; Ranzato et al., 2016). While the RANKGEN approach is significantly different from these prior works, it could be considered a "k-

word sequence-level" language modeling approach, which is discriminative rather than generative.

7 Limitations

An important limitation of RANKGEN compared to other decoding methods is the need for over-generation, which we discuss in Section 4.2. While RANKGEN itself is efficient, generating multiple samples increases decoding time by an order of magnitude. RANKGEN is a re-ranking method, so it relies on other decoding methods to produce the candidate output set. Biases in the output candidate set from existing decoding algorithms may be present in RANKGEN outputs. Besides this, RANKGEN may be vulnerable to adversarial examples (Szegedy et al., 2013) — gibberish text which gets high RANKGEN score, obtained by white-box attacks (Ebrahimi et al., 2018; Wallace et al., 2019).

This study is limited to open-ended text generation, which has a large space of possible outputs. RANKGEN or our findings may not be directly applicable to other generation tasks which have a more constrained output space like summarization, long-form QA or machine translation.

8 Conclusion & Future Work

We present RANKGEN, a large encoder neural network which scores continuations given a prefix, and can be flexibly incorporated into text generation systems. RANKGEN significantly outperforms sampling methods (nucleus / top-k / typical) in terms of automatic and human evaluations.

Future directions include (1) training larger RANKGEN models (T5-XXL size or bigger), with longer prefix / suffix lengths; (2) exploring the utility of RANKGEN in other generation tasks like dialog generation, summarization or long-form question answering; (3) RANKGEN re-ranking of significantly larger hypothesis sets generated using search algorithms like Xu and Durrett (2021); (4) a more direct incorporation of RANKGEN into generative modeling to eliminate the need for over-generation, either via direct gradient-based sampling (Qin et al., 2022) or distilling RANKGEN knowledge into language models via unlikelihood training (Welleck et al., 2020); (5) using RANKGEN as a retriever in knowledge retrieval augmented generation (Nakano et al., 2021; Komeili et al., 2021); (6) further exploring the capability of RANKGEN as a retriever, either zero-shot or by fine-tuning on retrieval benchmarks like BEIR (Thakur et al., 2021); (7) applying RANKGEN for fake news detection (Zellers et al., 2019b) since it scores human-written text highly; and (8) studying the utility of RANKGEN as a text generation evaluation metric like CARP (Matiana et al., 2021) or CLIPScore (Hessel et al., 2021).

Acknowledgements

We are very grateful to the freelancers on Upwork and volunteers who helped us evaluate generated text. We thank Xavier Garcia and the T5X team for helping us with technical issues related to the T5X library. We are grateful to William Cohen, Elizabeth Clark, Marzena Karpinska, Tu Vu, Simeng Sun, Ari Holtzman, Slav Petrov, Ciprian Chelba, Nader Akoury, Neha Kennard, Dung Thai, the UMass NLP group and the Google AI language research group in Pittsburgh for several useful discussions during the course of the project. This work was mostly done while Kalpesh Krishna (KK) was a student researcher at Google Research hosted by John Wieting. KK was partly supported by a Google PhD Fellowship awarded in 2021.

Ethical Considerations

Current text generation technology produces fluent outputs but suffer from several issues like factual inaccuracies, lack of faithfulness to the input prefix, commonsense issues etc., which makes their real-world deployment difficult. RANKGEN is an effort at rectifying some of these issues, with a focus faithfulness to input prompts. However, RANKGEN outputs continue to be factually inaccurate at times, as noted by some of our human annotators. This should be strongly considered before any direct deployment of this system. To tackle this issue, using RANKGEN for retrieval augmented generation (Nakano et al., 2021) is a promising direction for future work. We have also open-sourced all 600 human annotations, which have detailed explanations highlighting the strengths / weaknesses of RANKGEN compared to nucleus sampling.

Our final XL-sized models were trained using a Google Cloud TPUv3 Pod slice with 128 chips for a total of 2 days per model. Several similar sized models were trained during the development of this project, roughly one XL-size model every week from October 2021 to February 2022. Due to expensive training costs, we will open source our model checkpoints for the community to use and build upon. Note that "TPUs are highly efficient chips which have been specifically designed for machine learning applications" as mentioned in the Google 2020 environment report.[13] These accelerators run on Google Cloud, which is "carbon neutral today, but aiming higher: our goal is to run on carbon-free energy, 24/7, at all of our data centers by 2030." (https://cloud.google.com/sustainability). More details on model size and training are provided in Appendix A.1.

References

Stéphane Aroca-Ouellette and Frank Rudzicz. 2020. On Losses for Modern Language Models. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 4970–4981, Online. Association for Computational Linguistics.

Anton Bakhtin, Sam Gross, Myle Ott, Yuntian Deng, Marc'Aurelio Ranzato, and Arthur Szlam. 2019. Real or fake? learning to discriminate machine from human generated text. *arXiv preprint arXiv:1906.03351*.

Samy Bengio, Oriol Vinyals, Navdeep Jaitly, and Noam Shazeer. 2015. Scheduled sampling for sequence prediction with recurrent neural networks. *Advances in neural information processing systems*, 28.

Sumanta Bhattacharyya, Amirmohammad Rooshenas, Subhajit Naskar, Simeng Sun, Mohit Iyyer, and Andrew McCallum. 2021. Energy-based reranking: Improving neural machine translation using energy-based models. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 4528–4537, Online. Association for Computational Linguistics.

James Bradbury, Roy Frostig, Peter Hawkins, Matthew James Johnson, Chris Leary, Dougal Maclaurin, George Necula, Adam Paszke, Jake VanderPlas, Skye Wanderman-Milne, and Qiao Zhang. 2018. JAX: composable transformations of Python+NumPy programs.

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

Asli Celikyilmaz, Elizabeth Clark, and Jianfeng Gao. 2020. Evaluation of text generation: A survey. *arXiv preprint arXiv:2006.14799*.

[13] https://www.gstatic.com/gumdrop/sustainability/google-2020-environmental-report.pdf

Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. 2020. A simple framework for contrastive learning of visual representations. In *Proceedings of the International Conference of Machine Learning*.

Aakanksha Chowdhery, Sharan Narang, Jacob Devlin, Maarten Bosma, Gaurav Mishra, Adam Roberts, Paul Barham, Hyung Won Chung, Charles Sutton, Sebastian Gehrmann, Parker Schuh, Kensen Shi, Sasha Tsvyashchenko, Joshua Maynez, Abhishek Baindoor Rao, Parker Barnes, Yi Tay, Noam M. Shazeer, Vinodkumar Prabhakaran, Emily Reif, Nan Du, Benton C. Hutchinson, Reiner Pope, James Bradbury, Jacob Austin, Michael Isard, Guy Gur-Ari, Pengcheng Yin, Toju Duke, Anselm Levskaya, Sanjay Ghemawat, Sunipa Dev, Henryk Michalewski, Xavier García, Vedant Misra, Kevin Robinson, Liam Fedus, Denny Zhou, Daphne Ippolito, David Luan, Hyeontaek Lim, Barret Zoph, Alexander Spiridonov, Ryan Sepassi, David Dohan, Shivani Agrawal, Mark Omernick, Andrew M. Dai, Thanumalayan Sankaranarayana Pillai, Marie Pellat, Aitor Lewkowycz, Erica Oliveira Moreira, Rewon Child, Oleksandr Polozov, Katherine Lee, Zongwei Zhou, Xuezhi Wang, Brennan Saeta, Mark Diaz, Orhan Firat, Michele Catasta, Jason Wei, Kathleen S. Meier-Hellstern, Douglas Eck, Jeff Dean, Slav Petrov, and Noah Fiedel. 2022. Palm: Scaling language modeling with pathways. *arXiv preprint arXiv:2204.02311*.

Elizabeth Clark, Tal August, Sofia Serrano, Nikita Haduong, Suchin Gururangan, and Noah A. Smith. 2021. All that's 'human' is not gold: Evaluating human evaluation of generated text. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 7282–7296, Online. Association for Computational Linguistics.

Sumanth Dathathri, Andrea Madotto, Janice Lan, Jane Hung, Eric Frank, Piero Molino, Jason Yosinski, and Rosanne Liu. 2020. Plug and play language models: a simple approach to controlled text generation. In *Proceedings of the International Conference on Learning Representations*.

Mostafa Dehghani, Anurag Arnab, Lucas Beyer, Ashish Vaswani, and Yi Tay. 2021. The efficiency misnomer. *arXiv preprint arXiv:2110.12894*.

Yuntian Deng, Anton Bakhtin, Myle Ott, Arthur Szlam, and Marc'Aurelio Ranzato. 2020. Residual energy-based models for text generation. In *International Conference on Learning Representations*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Conference of the North American Chapter of the Association for Computational Linguistics*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Yao Dou, Maxwell Forbes, Rik Koncel-Kedziorski, Noah Smith, and Yejin Choi. 2022. Is GPT-3 text indistinguishable from human text? scarecrow: A framework for scrutinizing machine text. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 7250–7274, Dublin, Ireland. Association for Computational Linguistics.

Javid Ebrahimi, Anyi Rao, Daniel Lowd, and Dejing Dou. 2018. HotFlip: White-box adversarial examples for text classification. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 31–36, Melbourne, Australia. Association for Computational Linguistics.

Sergey Edunov, Myle Ott, Michael Auli, David Grangier, and Marc'Aurelio Ranzato. 2018. Classical structured prediction losses for sequence to sequence learning. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 355–364, New Orleans, Louisiana. Association for Computational Linguistics.

Angela Fan, Mike Lewis, and Yann Dauphin. 2018. Hierarchical neural story generation. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 889–898, Melbourne, Australia. Association for Computational Linguistics.

Joseph L Fleiss. 1971. Measuring nominal scale agreement among many raters. *Psychological bulletin*, 76(5):378.

Sebastian Gehrmann, Elizabeth Clark, and Thibault Sellam. 2022. Repairing the cracked foundation: A survey of obstacles in evaluation practices for generated text. *arXiv preprint arXiv:2202.06935*.

Sebastian Gehrmann, Hendrik Strobelt, and Alexander Rush. 2019. GLTR: Statistical detection and visualization of generated text. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 111–116, Florence, Italy. Association for Computational Linguistics.

Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. 2014. Generative adversarial nets. *Advances in neural information processing systems*, 27.

Barbara J Grosz, Scott Weinstein, and Aravind K Joshi. 1995. Centering: a framework for modeling the local coherence of discourse. *Computational Linguistics*, 21(2):203–225.

Aditya Grover, Jiaming Song, Ashish Kapoor, Kenneth Tran, Alekh Agarwal, Eric J Horvitz, and Stefano Ermon. 2019. Bias correction of learned generative models using likelihood-free importance weighting. *Advances in neural information processing systems*, 32.

Kelvin Guu, Kenton Lee, Zora Tung, Panupong Pasupat, and Mingwei Chang. 2020. Retrieval augmented language model pre-training. In *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pages 3929–3938. PMLR.

Jack Hessel, Ari Holtzman, Maxwell Forbes, Ronan Le Bras, and Yejin Choi. 2021. CLIPScore: A reference-free evaluation metric for image captioning. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 7514–7528, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Felix Hill, Kyunghyun Cho, and Anna Korhonen. 2016. Learning distributed representations of sentences from unlabelled data. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1367–1377, San Diego, California. Association for Computational Linguistics.

Jerry R Hobbs. 1979. Coherence and coreference. *Cognitive science*, 3(1):67–90.

Ari Holtzman, Jan Buys, Li Du, Maxwell Forbes, and Yejin Choi. 2020. The curious case of neural text degeneration. In *Proceedings of the International Conference on Learning Representations*.

Ari Holtzman, Jan Buys, Maxwell Forbes, Antoine Bosselut, David Golub, and Yejin Choi. 2018. Learning to write with cooperative discriminators. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1638–1649, Melbourne, Australia. Association for Computational Linguistics.

Ari Holtzman, Peter West, Vered Shwartz, Yejin Choi, and Luke Zettlemoyer. 2021. Surface form competition: Why the highest probability answer isn't always right. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 7038–7051, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Daphne Ippolito, David Grangier, Douglas Eck, and Chris Callison-Burch. 2020. Toward better storylines with sentence-level language models. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7472–7478, Online. Association for Computational Linguistics.

Yacine Jernite, Samuel R Bowman, and David Sontag. 2017. Discourse-based objectives for fast unsupervised sentence representation learning. *arXiv preprint arXiv:1705.00557*.

Marzena Karpinska, Nader Akoury, and Mohit Iyyer. 2021. The perils of using Mechanical Turk to evaluate open-ended text generation. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 1265–1285, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Vladimir Karpukhin, Barlas Oguz, Sewon Min, Patrick Lewis, Ledell Wu, Sergey Edunov, Danqi Chen, and Wen-tau Yih. 2020. Dense passage retrieval for open-domain question answering. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 6769–6781, Online. Association for Computational Linguistics.

Urvashi Khandelwal, He He, Peng Qi, and Dan Jurafsky. 2018. Sharp nearby, fuzzy far away: How neural language models use context. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 284–294, Melbourne, Australia. Association for Computational Linguistics.

Urvashi Khandelwal, Omer Levy, Dan Jurafsky, Luke Zettlemoyer, and Mike Lewis. 2020. Generalization through memorization: Nearest neighbor language models. In *International Conference on Learning Representations*.

Omar Khattab and Matei Zaharia. 2020. Colbert: Efficient and effective passage search via contextualized late interaction over bert. In *Proceedings of the 43rd International ACM SIGIR conference on research and development in Information Retrieval*, pages 39–48.

Ryan Kiros, Yukun Zhu, Russ R Salakhutdinov, Richard Zemel, Raquel Urtasun, Antonio Torralba, and Sanja Fidler. 2015. Skip-thought vectors. *Advances in neural information processing systems*, 28.

Mojtaba Komeili, Kurt Shuster, and Jason Weston. 2021. Internet-augmented dialogue generation. *arXiv preprint arXiv:2107.07566*.

Kalpesh Krishna, Aurko Roy, and Mohit Iyyer. 2021. Hurdles to progress in long-form question answering. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 4940–4957, Online. Association for Computational Linguistics.

Ranjay Krishna, Kenji Hata, Frederic Ren, Li Fei-Fei, and Juan Carlos Niebles. 2017. Dense-captioning events in videos. In *Proceedings of the IEEE international conference on computer vision*, pages 706–715.

Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. 2020. Albert: A lite bert for self-supervised learning of language representations. In *International Conference on Learning Representations*.

Yann LeCun, Sumit Chopra, Raia Hadsell, Marc'Aurelio Ranzato, and Fu Jie Huang. 2006. A tutorial on energy-based learning. *To appear in "Predicting Structured Data*, 1:0.

Ann Lee, Michael Auli, and Marc'Aurelio Ranzato. 2021. Discriminative reranking for neural machine translation. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 7250–7264, Online. Association for Computational Linguistics.

Brian Lester, Rami Al-Rfou, and Noah Constant. 2021. The power of scale for parameter-efficient prompt tuning. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 3045–3059, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Jiwei Li, Will Monroe, and Dan Jurafsky. 2017. Learning to decode for future success. *arXiv preprint arXiv:1701.06549*.

Yixin Liu, Pengfei Liu, Dragomir Radev, and Graham Neubig. 2022. Brio: Bringing order to abstractive summarization. *arXiv preprint arXiv:2203.16804*.

Lajanugen Logeswaran and Honglak Lee. 2018. An efficient framework for learning sentence representations. In *International Conference on Learning Representations*.

Luca Massarelli, Fabio Petroni, Aleksandra Piktus, Myle Ott, Tim Rocktäschel, Vassilis Plachouras, Fabrizio Silvestri, and Sebastian Riedel. 2020. How decoding strategies affect the verifiability of generated text. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 223–235, Online. Association for Computational Linguistics.

Shahbuland Matiana, JR Smith, Ryan Teehan, Louis Castricato, Stella Biderman, Leo Gao, and Spencer Frazier. 2021. Cut the carp: Fishing for zero-shot story evaluation. *arXiv preprint arXiv:2110.03111*.

Joshua Maynez, Shashi Narayan, Bernd Bohnet, and Ryan McDonald. 2020. On faithfulness and factuality in abstractive summarization. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 1906–1919, Online. Association for Computational Linguistics.

Clara Meister, Tiago Pimentel, Gian Wiher, and Ryan Cotterell. 2022. Typical decoding for natural language generation. *arXiv preprint arXiv:2202.00666*.

Alexander Miller, Will Feng, Dhruv Batra, Antoine Bordes, Adam Fisch, Jiasen Lu, Devi Parikh, and Jason Weston. 2017. ParlAI: A dialog research software platform. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 79–84, Copenhagen, Denmark. Association for Computational Linguistics.

Fatemehsadat Mireshghallah, Kartik Goyal, and Taylor Berg-Kirkpatrick. 2022. Mix and match: Learning-free controllable text generation using energy language models. In *Proceedings of the Association for Computational Linguistics*.

Nasrin Mostafazadeh, Nathanael Chambers, Xiaodong He, Devi Parikh, Dhruv Batra, Lucy Vanderwende, Pushmeet Kohli, and James Allen. 2016. A corpus and cloze evaluation for deeper understanding of commonsense stories. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 839–849, San Diego, California. Association for Computational Linguistics.

Reiichiro Nakano, Jacob Hilton, Suchir Balaji, Jeff Wu, Long Ouyang, Christina Kim, Christopher Hesse, Shantanu Jain, Vineet Kosaraju, William Saunders, et al. 2021. Webgpt: Browser-assisted question-answering with human feedback. *arXiv preprint arXiv:2112.09332*.

Aaron van den Oord, Yazhe Li, and Oriol Vinyals. 2018. Representation learning with contrastive predictive coding. *arXiv preprint arXiv:1807.03748*.

Tetiana Parshakova, Jean-Marc Andreoli, and Marc Dymetman. 2019. Global autoregressive models for data-efficient sequence learning. In *Proceedings of the 23rd Conference on Computational Natural Language Learning (CoNLL)*, pages 900–909, Hong Kong, China. Association for Computational Linguistics.

Krishna Pillutla, Swabha Swayamdipta, Rowan Zellers, John Thickstun, Sean Welleck, Yejin Choi, and Zaid Harchaoui. 2021. Mauve: Measuring the gap between neural text and human text using divergence frontiers. *Advances in Neural Information Processing Systems*, 34.

Weizhen Qi, Yu Yan, Yeyun Gong, Dayiheng Liu, Nan Duan, Jiusheng Chen, Ruofei Zhang, and Ming Zhou. 2020. ProphetNet: Predicting future n-gram for sequence-to-SequencePre-training. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 2401–2410, Online. Association for Computational Linguistics.

Lianhui Qin, Sean Welleck, Daniel Khashabi, and Yejin Choi. 2022. Cold decoding: Energy-based constrained text generation with langevin dynamics. *arXiv preprint arXiv:2202.11705*.

Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, and Ilya Sutskever. 2021. Learning transferable visual models from natural language supervision. In *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pages 8748–8763. PMLR.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners. *OpenAI Blog*, 1(8).

Jack W Rae, Anna Potapenko, Siddhant M Jayakumar, Chloe Hillier, and Timothy P Lillicrap. 2019. Compressive transformers for long-range sequence modelling. In *International Conference on Learning Representations*.

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

Marc'Aurelio Ranzato, Sumit Chopra, Michael Auli, and Wojciech Zaremba. 2016. Sequence level training with recurrent neural networks. In *4th International Conference on Learning Representations, ICLR 2016*.

Adam Roberts, Hyung Won Chung, Anselm Levskaya, Gaurav Mishra, James Bradbury, Daniel Andor, Sharan Narang, Brian Lester, Colin Gaffney, Afroz Mohiuddin, Curtis Hawthorne, Aitor Lewkowycz, Alex Salcianu, Marc van Zee, Jacob Austin, Sebastian Goodman, Livio Baldini Soares, Haitang Hu, Sasha Tsvyashchenko, Aakanksha Chowdhery, Jasmijn Bastings, Jannis Bulian, Xavier Garcia, Jianmo Ni, Andrew Chen, Kathleen Kenealy, Jonathan H. Clark, Stephan Lee, Dan Garrette, James Lee-Thorp, Colin Raffel, Noam Shazeer, Marvin Ritter, Maarten Bosma, Alexandre Passos, Jeremy Maitin-Shepard, Noah Fiedel, Mark Omernick, Brennan Saeta, Ryan Sepassi, Alexander Spiridonov, Joshua Newlan, and Andrea Gesmundo. 2022. Scaling up models and data with `t5x` and `seqio`. *arXiv preprint arXiv:2203.17189*.

Stephen E Robertson, Steve Walker, Susan Jones, Micheline M Hancock-Beaulieu, Mike Gatford, et al. 1995. Okapi at trec-3. *Nist Special Publication Sp*, 109:109.

Abigail See, Aneesh Pappu, Rohun Saxena, Akhila Yerukola, and Christopher D. Manning. 2019. Do massively pretrained language models make better storytellers? In *Proceedings of the 23rd Conference on Computational Natural Language Learning (CoNLL)*, pages 843–861, Hong Kong, China. Association for Computational Linguistics.

Dmitriy Serdyuk, Nan Rosemary Ke, Alessandro Sordoni, Adam Trischler, Chris Pal, and Yoshua Bengio. 2018. Twin networks: Matching the future for sequence generation. In *International Conference on Learning Representations*.

Vatsal Sharan, Sham Kakade, Percy Liang, and Gregory Valiant. 2018. Prediction with a short memory. In *Proceedings of the 50th Annual ACM SIGACT Symposium on Theory of Computing*, pages 1074–1087.

Rishi Sharma, James Allen, Omid Bakhshandeh, and Nasrin Mostafazadeh. 2018. Tackling the story ending biases in the story cloze test. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 752–757, Melbourne, Australia. Association for Computational Linguistics.

Noam Shazeer and Mitchell Stern. 2018. Adafactor: Adaptive learning rates with sublinear memory cost. In *International Conference on Machine Learning*, pages 4596–4604. PMLR.

Libin Shen, Anoop Sarkar, and Franz Josef Och. 2004. Discriminative reranking for machine translation. In *Proceedings of the Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics: HLT-NAACL 2004*, pages 177–184, Boston, Massachusetts, USA. Association for Computational Linguistics.

Francis Hopkinson Smith. 1911. *Peter: A Novel of which He is Not the Hero*. C. Scribner's sons.

Kihyuk Sohn. 2016. Improved deep metric learning with multi-class n-pair loss objective. *Advances in neural information processing systems*, 29.

Yixuan Su, Tian Lan, Yan Wang, Dani Yogatama, Lingpeng Kong, and Nigel Collier. 2022. A contrastive framework for neural text generation. *arXiv preprint arXiv:2202.06417*.

Simeng Sun, Kalpesh Krishna, Andrew Mattarella-Micke, and Mohit Iyyer. 2021. Do long-range language models actually use long-range context? In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 807–822, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Simeng Sun, Katherine Thai, and Mohit Iyyer. 2022. Chapterbreak: A challenge dataset for long-range language models. In *North American Association for Computational Linguistics*.

Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. 2013. Intriguing properties of neural networks. *arXiv preprint arXiv:1312.6199*.

Katherine Thai, Yapei Chang, Kalpesh Krishna, and Mohit Iyyer. 2022. Relic: Retrieving evidence for literary claims. In *Proceedings of the Association for Computational Linguistics*.

Nandan Thakur, Nils Reimers, Andreas Rücklé, Abhishek Srivastava, and Iryna Gurevych. 2021. Beir: A heterogeneous benchmark for zero-shot evaluation of information retrieval models. In *Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track (Round 2)*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Proceedings of Advances in Neural Information Processing Systems*, pages 5998–6008.

Eric Wallace, Shi Feng, Nikhil Kandpal, Matt Gardner, and Sameer Singh. 2019. Universal adversarial triggers for attacking and analyzing NLP. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 2153–2162, Hong Kong, China. Association for Computational Linguistics.

Bin Wang and Zhijian Ou. 2018. Learning neural trans-dimensional random field language models with noise-contrastive estimation. In *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 6134–6138. IEEE.

Rose E Wang, Esin Durmus, Noah Goodman, and Tatsunori Hashimoto. 2022. Language modeling via stochastic processes. In *International Conference on Learning Representations*.

Sean Welleck, Kianté Brantley, Hal Daumé Iii, and Kyunghyun Cho. 2019. Non-monotonic sequential text generation. In *International Conference on Machine Learning*, pages 6716–6726. PMLR.

Sean Welleck, Ilia Kulikov, Stephen Roller, Emily Dinan, Kyunghyun Cho, and Jason Weston. 2020. Neural text generation with unlikelihood training. In *International Conference on Learning Representations*.

John Wieting, Taylor Berg-Kirkpatrick, Kevin Gimpel, and Graham Neubig. 2019. Beyond BLEU:training neural machine translation with semantic similarity. In *Proceedings of the Association for Computational Linguistics*.

Sam Wiseman and Alexander M. Rush. 2016. Sequence-to-sequence learning as beam-search optimization. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1296–1306, Austin, Texas. Association for Computational Linguistics.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Remi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander Rush. 2020. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing:*

System Demonstrations, pages 38–45, Online. Association for Computational Linguistics.

Jiacheng Xu and Greg Durrett. 2021. Massive-scale decoding for text generation using lattices. *arXiv preprint arXiv:2112.07660*.

Kevin Yang and Dan Klein. 2021. FUDGE: Controlled text generation with future discriminators. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 3511–3535, Online. Association for Computational Linguistics.

Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. 2019a. HellaSwag: Can a machine really finish your sentence? In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4791–4800, Florence, Italy. Association for Computational Linguistics.

Rowan Zellers, Ari Holtzman, Hannah Rashkin, Yonatan Bisk, Ali Farhadi, Franziska Roesner, and Yejin Choi. 2019b. Defending against neural fake news. *Advances in neural information processing systems*, 32.

Hugh Zhang, Daniel Duckworth, Daphne Ippolito, and Arvind Neelakantan. 2021. Trading off diversity and quality in natural language generation. In *Proceedings of the Workshop on Human Evaluation of NLP Systems (HumEval)*, pages 25–33, Online. Association for Computational Linguistics.

Xingxing Zhang and Mirella Lapata. 2014. Chinese poetry generation with recurrent neural networks. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 670–680, Doha, Qatar. Association for Computational Linguistics.

Appendices accompanying "RankGen: Improving Text Generation with Large Ranking Models"

A More RankGen details

A.1 RankGen training details

We fine-tune the encoder of the T5 v1.1 models from Raffel et al. (2020) using large minibatches (see Table 11 for sizes) on a Cloud TPU v3 Pod slice with 128 chips. Our models are implemented in JAX (Bradbury et al., 2018) using the T5X library (Roberts et al., 2022). Each model was fine-tuned for 100k steps, using a constant learning rate of 0.002 using the Adafactor optimizer (Shazeer and Stern, 2018).

Model	Batch Size	Parameters
RankGen-base	4096	110.2M
RankGen-large	4096	342.3M
RankGen-XL	1536	1.2B

Table 11: Minibatch size and number of trainable parameters across different RankGen variants. See Appendix E for ablation studies justifying scale.

A.2 Implementation and timing details

In Figure 5 we provided a simplified Python implementation (without minibatching) of our RankGen beam search algorithm. We implement this algorithm in two libraries — the first uses PyTorch with the popular HuggingFace Transformers library (Wolf et al., 2020), which we test on a RTX 3090 GPU with 25GB memory. The second uses JAX (Bradbury et al., 2018) with the T5X library (Roberts et al., 2022), and is tested on a single Cloud TPU v3 board with 32GB memory.[14] While measuring decoding time for various hyperparameters (Appendix A.3.2), we focus on *throughput* (Dehghani et al., 2021), measuring wall-clock time after minibatching to the extent the hardware permits. We ensure consistent experimental settings across hyperparameters, using the same machine and making sure no other computationally expensive process is running on it.

[14] https://cloud.google.com/tpu/docs/system-architecture-tpu-vm#single_tpu_board

A.3 RankGen hyperparameter grid search

Our hyperparameter grid search is conducted on Wikipedia data with the smallest model considered (GPT2-medium), using MAUVE as our hill-climbing criteria. Our RankGen algorithm has three main hyperparameters — rerank length L, beam size B and number of samples per beam N. The rerank length denotes the number of new tokens which are generated before a re-ranking step takes place. Number of samples denotes the number of generated sequences for each beam. The number of samples retained across different re-ranking cycles is the beam size (see Figure 5 for exact implementation). Our RankGen grid search is conducted over the following configurations —
rerank length L: 5, 10, 20, 50, max_length tokens
number of samples (beam size B * number of samples in every beam N):
1 sample — (1 * 1);
5 samples — (1 * 5);
10 samples — (1 * 10); (2 * 5);
20 samples — (1 * 20); (2 * 10); (4 * 5);
40 samples — (1 * 40); (2 * 20);

Additionally, we measure the extent to which full-length reranking works (L = max length, B = 1) by simply increasing the number of samples N over-generated and then for re-ranking.

A.3.1 MAUVE score tradeoffs

In Figure 6 we study the MAUVE performance tradeoffs for different hyperparameter configurations for the GPT2-medium model evaluated on Wikipedia data. Overall, we observe —

- Across all hyperparameter configurations, RankGen significantly improves MAUVE score over a no re-ranking baseline.
- MAUVE scores improve for shorter rerank lengths, justifying the benefit of beam search over re-ranking of complete generations.
- For cases of full re-ranking (re-rank length = max length), increasing number of samples improves the MAUVE score (since RankGen has more generations to choose from), but improvements saturates after 60 samples (for both model sizes), with the largest gain from 1 to 10 samples.
- We find that rerank length = 20 with 20 samples (beam size 2, samples per beam 10) performs best across all configurations.

A.3.2 Speed tradeoffs

In Figure 7 we study the average time taken (in seconds) for a single generation on Wikipedia. Overall, in both our implementations we observe that —

- Decoding a single sample is an order of magnitude faster than decoding multiple samples ("over-generation"), which is needed before any re-ranking with RANKGEN is possible.
- Reducing the rerank length increases decoding time, since more generate / re-rank cycles are needed. These cycles cannot be parallelized since the generate and re-rank steps are dependent on each other.
- Overall, we see observe that decoding time is roughly $\mathcal{O}(BN/L)$, where B is beam size, N is the number of samples per beam and L is rerank length. This is especially true for the T5X implementation.

Also see Section 4.2 in the main body of the paper for a performance / time tradeoff plot, and analysis showing that over-generation is the speed bottleneck for decoding with RANKGEN.

B Human Evaluation Details

We hired freelancers from Upwork[15] as well as two volunteers to perform our human evaluation. In total, our human evaluation had eight annotators. Following recent recommendations from Karpinska et al. (2021), we ensured that each annotator (except one) was either an English teacher or an English writer. To avoid bias, we ensured that none of the annotators were computer science researchers, making them unaware of text generation research / RANKGEN.

Setup: Annotators were shown a 200-250 word prefix, and were asked to choose one of two 80-100 word continuations. Annotators were not told which model generated each continuation, and we shuffled the continuations in a random order to avoid position biases ("*blind A/B testing*"). We used Amazon Mechanical Turk Sanbox[16] to collect our annotations, using the interface shown in Figure 10. Note that we used the MTurk Sandbox *interface* only — no MTurk *workers* are recruited in our human study due to poor annotation quality for open-ended text generation (Karpinska et al., 2021; Clark et al., 2021).

Screening: To ensure high annotation quality, we first asked annotators to complete a small screening test of 20 pairs with INBOOK distractors, keeping 80% accuracy as our passing criteria (estimated human performance on this set is 90-95%). We paid annotators 10$ for the screening test. Around half the interviewed Upworkers passed the test.

Main Task (comparing generations): In our main task comparing generations from RANKGEN with nucleus sampling, we asked annotators to choose the better continuation as well as provide a 1-3 sentence free-form explanation for their choice. We paid annotators 1$ for each pair, and provided a 10$ bonus at the end of a 100 pairs. Each annotator was provided with 100 instances (50 each from Wikipedia and PG19) either generated by the T5-XXL-C4 model (Lester et al., 2021) or GPT2-medium (Radford et al., 2019), with beam search outputs from RANKGEN-XL-all. Three annotators rate each model, giving us a total of 600 human annotations with explanations.

Main Task (INBOOK human estimate): Our second main task involved choosing the gold human-written continuation vs random INBOOK negatives. We paid annotators 0.5$ for this task, and did not ask them to explain their choices. This main task was similar in nature to our screening task.

C Suffix Identification

C.1 Gold vs INBOOK - more negatives

In Section 2.1, we used a single INBOOK to test models. How do models fare when they need to choose the gold continuation over multiple INBOOK negatives? In Table 12 we perform experiments on a 11-way classification task (10 INBOOK negatives). Overall, we find that most LMs do barely above chance, whereas RANKGEN significantly outperforms large LMs (even GPT3).

Gold vs all INBOOK negatives ("retrieval"): What if instead of 10 negatives, we used all possible INBOOK negatives in the book? This task could be framed as a *retrieval* problem akin to RELiC (Section 5.1): given a prefix, find the correct continuation from all possible continuations in the same book. Since PG19 books can be quite long, retrievers needs to search among 2538 candidates on

[15] https://www.upwork.com
[16] https://requestersandbox.mturk.com/

INBOOK neg type →	Random		Hard	
	PG	Wiki	PG	Wiki
Random	9.1	9.1	9.1	9.1
Unigram Overlap	42.3	18.5	8.6	5.0
GPT2-medium (2019)	25.2	11.8	7.6	5.0
GPT2-XL (2019)	28.2	12.4	8.2	5.0
T5-base (f.t. PG19)	28.8	14.3	7.8	5.1
T5-XXL (f.t. PG19)	38.8	17.5	9.8	6.0
T5-XXL-C4 (2021)	34.3	14.6	9.2	5.5
GPT3 170B* (2020)	32.0	14.0	14.0	8.0
RANKGEN (ours)				
PG19-XL-INBOOK	**94.4**	69.8	49.1	36.5
PG19-XL-GENERATE	45.0	28.5	11.7	11.8
PG19-XL-both	**94.4**	69.0	**49.5**	35.7
all-XL-both	92.6	**84.6**	39.5†	**52.1**†

Table 12: A version of Table 1 with 10 distractors (11-way classification). Like Table 1, large LMs perform poorly and close to chance on hard sets. *GPT3 scores computed using 100 datapoints. †The hard sets were adversarially constructed using this RANKGEN variant.

Model Size	Batch Size	Retrieval over PG19 books R@1	R@3	R@5	R@10
(RANKGEN models trained on PG19)					
base	4096	34.9	52.6	60.6	70.5
large	4096	45.2	62.8	69.9	78.1
XL	1536	48.1	65.4	72.1	79.7
XL-inbook	1536	48.2	65.5	72.1	79.7
XL-gen	1536	4.4	10.4	14.4	20.5
(RANKGEN models trained on all 4 domains)					
base	4096	28.4	44.4	52.1	62.4
large	4096	39.6	56.8	64.0	72.9
XL	256	24.3	38.7	45.7	55.4
XL	512	31.7	47.5	54.6	64.1
XL	768	34.6	51.0	58.5	67.5
XL	1536	41.5	58.8	65.7	74.3

Table 13: RANKGEN retrieval performance on PG19 validation books. On average, retrieval takes place over 2538 candidates. RANKGEN gets high performance on this task, and scaling model size, scaling minibatch size, training on just PG19 and using just INBOOK negatives improves recall@1 (R@1).

average in the PG19 validation set. We present results on this retrieval task in Table 13. Overall, we find that RANKGEN is quite successful at this task, getting a recall@1 of 48.2% with a model trained on just PG19 data and INBOOK negatives. Training on just PG19, increase model size, increasing minibatch size and using just INBOOK negatives helps improve retrieval performance. In initial experiments, we extensively used performance on this task to hill-climb and justify our design choices. Note that we do not test LMs on this retrieval task, since it is computationally expensive to do a forward pass for each of the 2538 candidates for each of the 100K datapoints.

C.2 Gold vs GENERATIVE - breakdown by generative model

See Table 14 for a breakdown by the model used to create the GENERATIVE negatives.

C.3 Details of Suffix Identification Datasets

ChapterBreak (Sun et al., 2022) is a 6-way classification task in which models are provided as input a long segment from a narrative that ends in a chapter boundary. Models must then identify the correct ground-truth chapter beginning from a set of negatives sampled from the same narrative — a task requiring global narrative understanding. ChapterBreak has two settings — (1) PG19 — the validation set of the Project Gutenberg language modeling benchmark (Rae et al., 2019); (2) AO3 — a ChapterBreak split adapted from fan-fiction posted to Archive of Our Own (AO3).[17] Although Sun et al. (2022) provide prefixes up to 8192 tokens, we study ChapterBreak in the setting using just 256 tokens of prefix to ensure compatibility with the input lengths of RANKGEN. The ChapterBreak dataset is not divided into validation / test splits, so we simply use the single available split.

HellaSwag (Zellers et al., 2019a) is a 4-way classification task focusing on commonsense natural language inference. For each question, a prefix from a video caption is provided as input and a model must choose the correct continuation for this prefix. Only one out of the four choices is correct – the actual next caption of the video. HellaSwag is scraped from the video captions in ActivityNet (Krishna et al., 2017) and how-to paragraph instructions on WikiHow. We study the setting where each of the 4 endings are complete sentences, which is constructed by prepending `ctx_b` to the given endings). We use the validation set of the HellaSwag corpus since the test set answers are hidden.

StoryCloze (Mostafazadeh et al., 2016; Sharma et al., 2018) is a 2-way classification task designed to test commonsense reasoning. Systems are provided with the first four sentences of a five-sentence

[17] https://archive.org/download/AO3_story_dump_continuing

Discriminator	GPT2-md		GPT2-XL		T5-XXL-PG19		T5-XXL-C4		Average
	PG19	wiki	PG19	wiki	PG19	wiki	PG19	wiki	
Random	50.0	50.0	50.0	50.0	50.0	50.0	50.0	50.0	50.0
Unigram Overlap	38.4	43.6	36.7	39.8	48.5	56.8	37.2	37.4	42.3
GPT2-medium (2019)	2.1	4.9	3.0	6.6	36.1	59.1	17.2	22.7	19.0
GPT2-XL (2019)	12.7	23.3	1.7	4.6	45.1	68.7	26.5	29.3	26.5
T5-XXL (f.t. PG19)	46.2	54.6	23.5	29.7	28.5	26.3	31.5	24.1	33.1
T5-XXL-C4 (2021)	24.7	52.2	10.9	26.1	31.9	65.2	8.5	13.0	29.1
RankGen (ours)									
PG-XL-Generative	**96.9**	**91.4**	**95.7**	**88.8**	**91.8**	92.3	**94.3**	**84.4**	**91.9**
PG-XL-InBook	78.4	66.3	69.7	60.3	65.9	60.1	65.2	52.2	64.8
PG-XL-both	97.4	81.3	93.7	74.0	87.4	79.4	89.7	65.0	83.5
all-XL-both	94.3	84.5	88.8	78.0	80.3	**95.3**	81.3	67.3	83.7

Table 14: A version of Table 2 breaking down performance by domain (Project Gutenberg PG19, Wikipedia) and model used to generate Generative negatives using nucleus sampling (Holtzman et al., 2020) with $p = 0.9$. Language model perplexity prefers Generative sequences over human text (as previously noted by Gehrmann et al., 2019), especially when the Generative negative is generated by the same language model.

commonsense story, and must choose the correct ending to the story. We used the test set for the Spring 2016 split and the validation set for the Winter 2018 split (due to the hidden test set).

D More Evaluation Details & Results

D.1 MAUVE setup

We extensively use the MAUVE metric from Pillutla et al. (2021) for automatic evaluation of our model. MAUVE is shown to have high correlation with human judgements of the quality of generated text. We closely follow the best practices listed in the official MAUVE repository,[18] which we found critical in preliminary experiments. Specifically,

1. We ensure that each run has the exact same hyperparameters — using the default hyperparameters in the official MAUVE library.

2. We use 7713 generations per run, which is the size of our Wikipedia validation set. This follows the suggestion in the official codebase README of having at least 5000 generations for comparing models. While our PG19 validation set is much bigger, we truncate it to 7713 generations since MAUVE scores tend to reduce with more generations.

3. Since MAUVE scores are higher for shorter generations, we ensure that all tested methods have roughly equal generation lengths, between 70-80 words / 120-130 tokens. We also truncate human text / generations to ensure that each instance ends at a sentence boundary. In initial experiments we observed that truncating consistently for human text and machine text leads to lower MAUVE variation.

4. Due to variation in MAUVE score from run to run, we average the MAUVE score for nucleus / top-k / typical sampling over five runs. For the T5-XXL-C4 model on Wikipedia with nucleus sampling, the MAUVE scores were [0.803, 0.778, 0.759, 0.785, 0.768], giving a standard deviation of 0.015.

[18] https://github.com/krishnap25/mauve#best-practices-for-mauve

D.2 MAUVE Divergence Curves

The MAUVE metric is the area under a divergence curve, a curve which attempts to analyze the type of errors the model is making. Given P is the distribution of human text and Q is the distribution of machine-generated text, Pillutla et al. (2021) describe two types of errors made by models —

Type I: $\mathrm{KL}(Q|P)$ — False positives, or cases where models generate text which is unlikely to be written by humans, like semantic repetitions common in neural text generators (Holtzman et al., 2020; Zhang et al., 2021).

Type II: $\mathrm{KL}(P|Q)$ — False negatives, or cases where models cannot generate text which is likely to be written by humans, sometimes seen with truncation strategies (See et al., 2019).

In Figure 8 and Figure 9 we plot the divergence curves comparing greedy decoding, nucleus sampling, and full sample re-ranking with perplexity and RankGen. We observe that re-ranking with RankGen increases the area under the curve, whereas re-ranking with model perplexity reduces the area. Re-ranking with RankGen reduces both Type I (bigger intercept on $y = 1$) and Type II errors (bigger intercept on $x = 1$). Re-ranking with perplexity leads to higher Type I errors, or more repetition (as also observed in Appendix D.3).

D.3 Token Overlap metrics

In addition to the MAUVE scores calculated in Section 3, we measure token overlap statistics comparing different decoding methods. First, we measure the **rep** metric from Welleck et al. (2020), which is an approximate measurement of the amount of repetition in generated text. We measure the percentage of generated tokens which are exactly copied from the immediate local prefix of 20 tokens. In Table 15 we find that re-ranking with RankGen slightly reduces **rep** compared to nucleus sampling (18.9 vs 19.5). We get even lower repetition on the RankGen trained on just generative negatives (17.8), while RankGen trained on just inbook negatives gets 20.0 — thus generative negatives are better at reducing repetition. Re-ranking with perplexity increases **rep** to 23.9, whereas greedy decoding has the highest repetition of 59.5. This is consistent with recent findings of repetition in greedy decoded outputs (Holtzman et al., 2020; Zhang et al., 2021). Human text is the least repetitive, with a **rep** score of 15.4.

Next, we measure the fraction of unigrams in the generation which are also present in the prefix. Higher scores could either imply more faithfulness to the prefix (less hallucination), or lower amounts of abstraction. We present two versions of this metric — (1) considering all tokens (Table 16); (2) considering only only lemmatized nouns and numbers (Table 17). Overall, we find that re-ranking samples with RankGen slightly increases this overlap score (19.5 vs 21.7), but re-ranking by token overlap (38.4) or perplexity (25.0) leads to a much higher score. Given the lower MAUVE scores for these two approaches (Table 3), we suspect that token overlap / perplexity re-ranking leads to lower amounts of abstraction / repetitiveness. Human written text has the lowest overlap, perhaps indicating more abstractive text.

E Ablation Studies

We conduct several ablation studies studying the importance of three aspects — (1) model size; (2) minibatch size, or number of negative samples during contrastive learning; (3) the type of negative samples (inbook, generative or both). Overall, we see clear benefits of increasing model size and increasing minibatch size for suffix identification (Table 18, Table 19) and human-text identification (Table 21). We see a similar, but less prominent trend on MAUVE scores after re-ranking generations (Table 20). For some settings we find that the RankGen-large variant produces slightly better generations than RankGen-XL. We hypothesize this is due to the much larger minibatch used to train RankGen-large models (4096) compared to RankGen-XL (1536) due to memory constraints.

F More Model Generations

More model generations with human explanations are provided in Table 22 to Table 26. See our official Github repository (link on Page 1) for all 600 annotations for the 200 generation pairs.

```python
def rankgen_search(prefix, scorer, generator,
                   rerank_length, beam_size, samples_per_beam):
  all_beams = [""]
  for _ in range(0, MAX_LENGTH, rerank_length):
    # concatenate input prefix with current beams
    all_inputs = [prefix + " " + beam for beam in all_beams]
    # for each beam, generate next rerank_length tokens.
    # samples_per_beam hypotheses are generated per beam,
    # making a total of (num_beams * samples_per_beam) hypotheses
    hypotheses = generator(all_inputs,
                           num_new_tokens=rerank_length,
                           num_samples=samples_per_beam)
    # measure RankGen score between prefix and each hypothesis
    scores = scorer(prefix, hypotheses)
    # take top-K scores where K=beam size
    top_indices = np.argsort(-1 * scores)[:beam_size]
    all_beams = [outputs[x] for x in top_indices]
  return all_beams
```

Figure 5: A simplified Python implementation showing our RANKGEN beam search algorithm (without minibatching). For every `rerank_length` tokens, a generator suggests hypotheses and the RANKGEN scorer ranks them. The top `beam_size` hypotheses are retained for the next stage of generation and re-ranking.

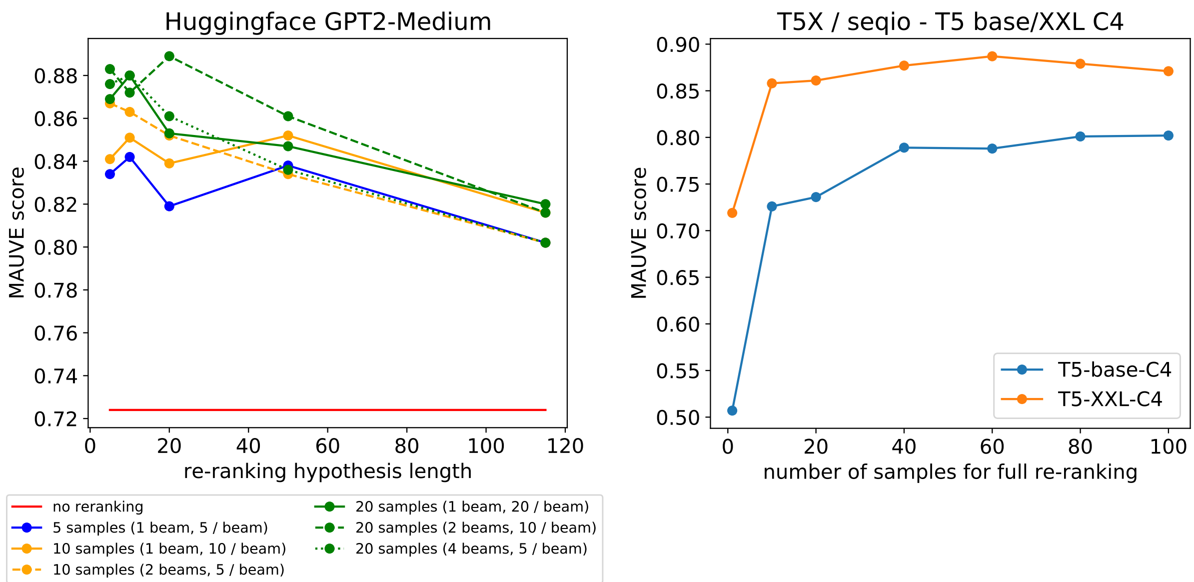

Figure 6: Variation in MAUVE score across different RANKGEN hyperparameters on Wikipedia data (Appendix A.3.1). **Left**: Experiments on GPT2-medium show that RANKGEN improvements are robust to hyperparameter choice, re-ranking shorter hypotheses improves performances over full re-ranking, re-ranking more samples improves performance. **Right**: Full re-ranking performance generally improves with more samples, but this improvement saturates after a point, especially for larger models (T5-XXL).

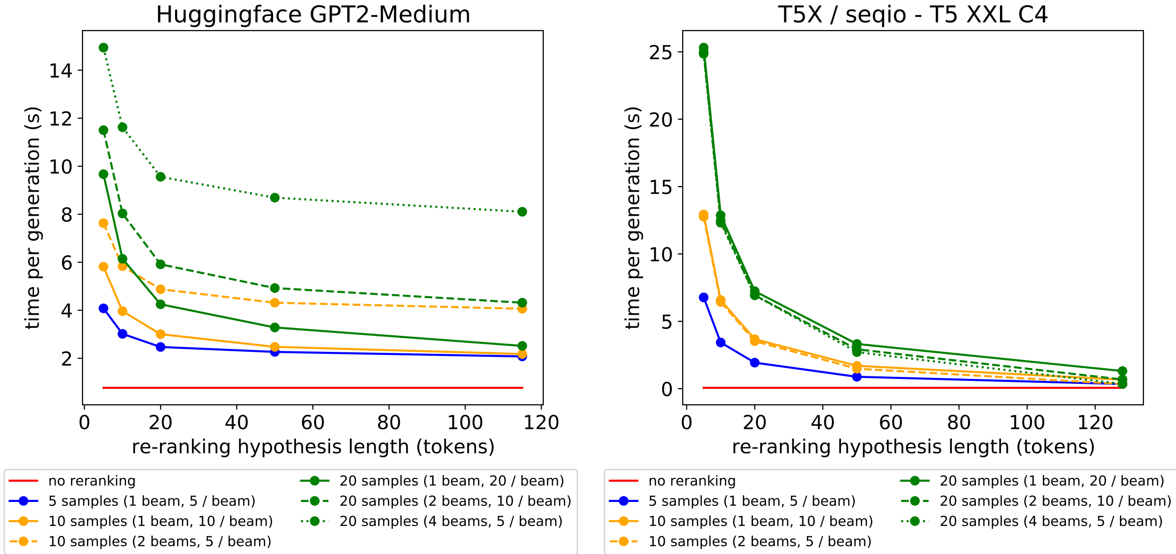

Figure 7: Time taken (in seconds) for a single generation across different hyperparameter settings in both our implementations (HuggingFace / T5X). We see roughly linear increase in decoding time with number of samples, and linear increase with number of re-ranking steps (1 / rerank_length).

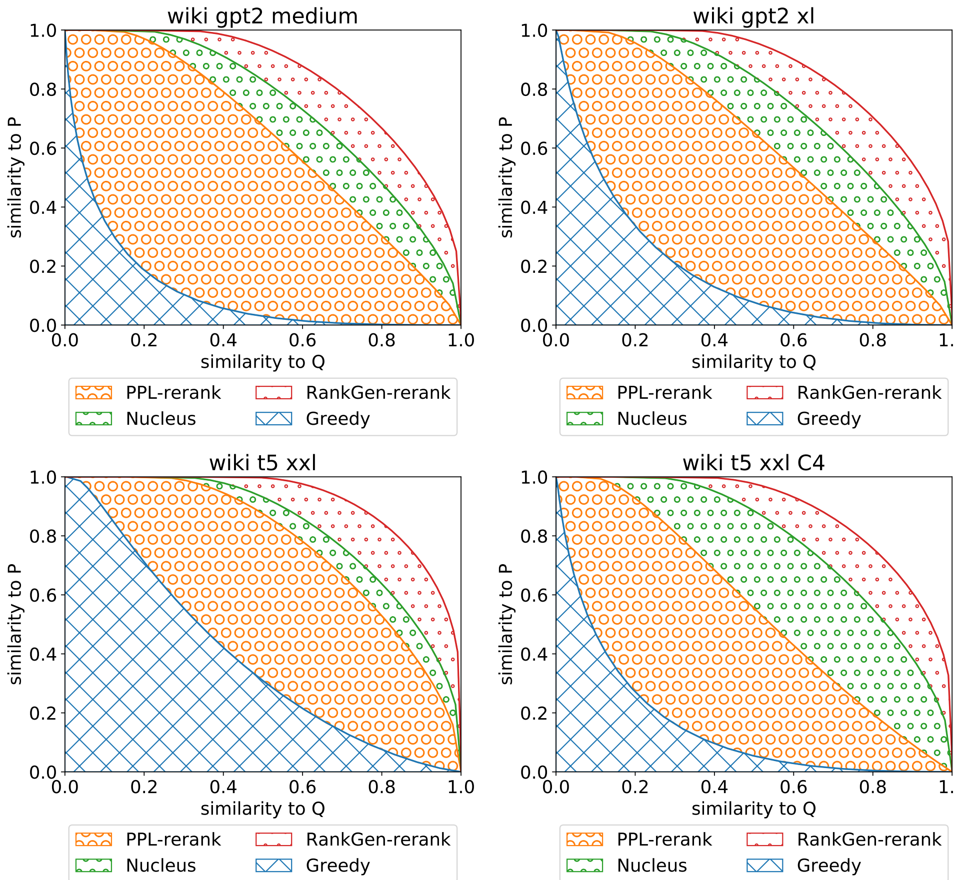

Figure 8: Divergence curves (Pillutla et al., 2021) after full sample re-ranking on Wikipedia inputs using RANKGEN-XL trained on all four domains. The area under this curve is the MAUVE score. Overall, we see that RANKGEN makes fewer Type I (bigger intercept with $y = 1$ line) and Type II style errors (bigger intercept with $x = 1$ line). PPL re-ranking increases the amount of repetition in generated text (Table 15), leading to more Type I errors (smaller intercept with $y = 1$ line).

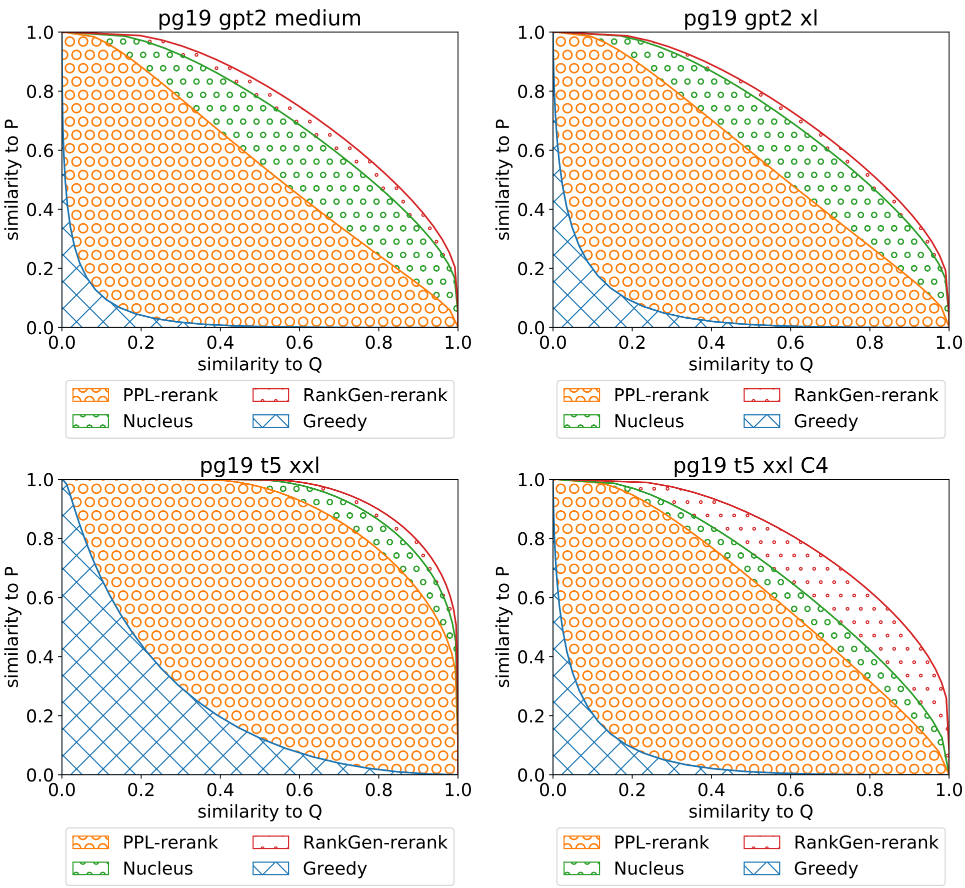

Figure 9: Divergence curves (Pillutla et al., 2021) after full sample re-ranking on PG19 inputs using RANKGEN-XL trained on PG19. The area under this curve is the MAUVE score. Overall, we see that RANKGEN makes fewer Type I (bigger intercept with $y = 1$ line) and Type II style errors (bigger intercept with $x = 1$). PPL re-ranking increases the amount of repetition in generated text (Table 15), leading to more Type I errors (smaller intercept with $y = 1$).

	Generator Language Model								
	GPT2-md		GPT2-XL		T5-XXL-PG19		T5-XXL-C4		Average
Decoding method	PG19	wiki	PG19	wiki	PG19	wiki	PG19	wiki	
Human Text	15.8	15.0	15.8	15.0	15.8	15.0	15.8	15.0	15.4
Greedy decoding	71.4	56.6	66.8	51.6	55.6	52.7	67.6	53.7	59.5
Nucleus, $p = 0.9$ (2020)	21.8	18.8	22.4	19.5	17.7	17.4	20.3	18.4	19.5
Top-k, $k = 40$ (2018)	19.4	17.0	19.9	19.7	17.9	17.9	20.4	18.6	18.9
Typical, $p = 0.9$ (2022)	21.6	18.6	22.2	19.5	17.6	17.4	20.3	18.5	19.5
Re-ranking 20 nucleus samples									
Unigram overlap	22.2	19.9	22.9	20.6	19.0	18.7	21.5	19.8	20.6
LM perplexity	26.9	23.2	27.9	24.3	20.4	21.5	24.6	22.5	23.9
RANKGEN PG-XL-gen	20.0	17.2	20.5	17.9	16.3	15.8	18.3	16.6	17.8
RANKGEN PG-XL-inbook	22.1	19.5	22.7	20.0	18.2	17.8	20.7	18.6	20.0
RANKGEN PG-XL-both	20.9	18.4	21.6	19.2	17.4	16.9	19.7	18.2	19.0
RANKGEN all-XL-both	20.5	18.6	21.1	19.4	17.3	16.6	19.5	18.2	18.9

Table 15: Fraction of generated tokens which are copied from the previous 20 tokens, *roughly measuring the amount of repetition* in text (the **rep** metric from Welleck et al., 2020). Overall we find that ranking samples with RANKGEN reduces repetition, whereas ranking with perplexity increases repetition. Greedy decoded outputs are the most repetitive, whereas human-written text is the least repetitive.

Decoding method	GPT2-md PG19	GPT2-md wiki	GPT2-XL PG19	GPT2-XL wiki	T5-XXL-PG19 PG19	T5-XXL-PG19 wiki	T5-XXL-C4 PG19	T5-XXL-C4 wiki	Average
	Generator Language Model								
Human Text	14.0	20.7	14.0	20.7	14.0	20.7	14.0	20.7	17.4
Greedy decoding	16.1	25.5	15.9	25.0	15.8	21.0	20.0	27.3	20.8
Nucleus, $p = 0.9$ (2020)	16.7	22.8	17.3	23.7	14.0	19.0	17.8	24.8	19.5
Top-k, $k = 40$ (2018)	15.6	21.0	15.8	15.9	15.1	20.2	19.3	25.7	18.6
Typical, $p = 0.9$ (2022)	16.6	22.5	17.2	23.8	14.1	18.8	18.0	25.0	19.5
Re-ranking 20 nucleus samples									
Unigram overlap	33.6	43.5	34.4	45.7	28.9	34.1	39.9	47.0	38.4
LM perplexity	19.9	29.4	20.2	30.2	16.9	22.7	27.3	33.1	25.0
RankGen PG-XL-gen	18.8	25.5	19.3	26.5	14.6	20.0	20.9	26.6	21.5
RankGen PG-XL-inbook	18.8	25.1	19.4	26.4	15.9	21.0	19.7	26.5	21.6
RankGen PG-XL-both	19.4	25.2	19.7	26.5	15.7	21.3	21.2	26.7	22.0
RankGen all-XL-both	19.1	24.8	19.5	26.1	15.7	21.3	20.4	26.3	21.7

Table 16: Percentage of unigrams in generation also present in the prefix. Overall, we see that re-ranking nucleus samples with RankGen increases this overlap, but not as much as re-ranking with LM perplexity. Human text has the lowest overlap, which we hypothesize is due to higher amounts of abstraction.

Decoding method	GPT2-md PG19	GPT2-md wiki	GPT2-XL PG19	GPT2-XL wiki	T5-XXL-PG19 PG19	T5-XXL-PG19 wiki	T5-XXL-C4 PG19	T5-XXL-C4 wiki	Average
	Generator Language Model								
Human Text	19.6	27.3	19.6	27.3	19.6	27.3	19.6	27.3	23.4
Greedy decoding	23.8	31.1	23.0	30.5	21.8	26.2	26.5	33.2	27.0
Nucleus, $p = 0.9$ (2020)	23.8	29.7	24.2	30.3	19.3	24.4	24.6	31.6	26.0
Top-k, $k = 40$ (2018)	22.0	27.6	22.2	28.7	21.0	26.4	27.1	33.2	26.0
Typical, $p = 0.9$ (2022)	23.7	29.2	24.2	30.3	19.4	24.5	24.8	32.0	26.0
Re-ranking 20 nucleus samples									
Unigram overlap	42.0	51.0	42.4	52.9	35.1	41.0	47.4	54.7	45.8
LM perplexity	27.8	35.1	27.1	35.4	23.0	28.9	35.2	39.2	31.4
RankGen PG-XL-gen	26.3	32.6	26.5	33.4	20.4	26.5	28.6	34.2	28.6
RankGen PG-XL-inbook	26.5	32.7	26.9	34.1	21.8	27.7	27.4	34.2	28.9
RankGen PG-XL-both	27.0	32.8	27.5	33.9	21.8	28.0	29.2	34.5	29.3
RankGen all-XL-both	27.0	32.6	27.3	33.7	21.7	28.0	28.4	34.0	29.1

Table 17: A version of Table 16 considering only lemmatized nouns, proper nouns and numbers, with similar trends.

Model Size	Batch Size	ChapterBreak PG19	ChapterBreak AO3	StoryCloze 2016	StoryCloze 2018	Hella Swag	RELiC (Recall@k) 1	RELiC 3	RELiC 5	RELiC 10	RELiC 50
(RankGen models trained on PG19)											
base	4096	57.7	36.0	67.6	68.7	30.7	3.8	8.2	10.8	15.4	31.6
large	4096	60.6	31.9	69.3	69.8	34.2	**5.7**	**11.0**	**14.5**	**20.0**	**36.6**
XL	1536	**63.5**	**36.9**	**71.1**	**72.6**	**40.7**	4.5	8.4	11.0	15.1	27.9
(RankGen models trained on all 4 domains)											
base	4096	48.1	33.0	69.0	69.1	34.0	3.1	6.2	8.3	11.8	25.6
large	4096	51.4	31.1	70.3	71.7	40.6	3.7	7.3	9.5	13.1	25.8
XL	256	38.2	28.3	70.6	68.5	35.9	2.8	5.6	7.4	10.8	22.9
XL	512	47.3	31.3	72.3	69.8	39.3	3.3	7.1	9.7	13.6	26.5
XL	768	45.2	30.1	72.5	71.2	41.4	3.8	7.2	9.6	13.7	27.5
XL	1536	**59.3**	**32.8**	**75.4**	**75.8**	**46.3**	**4.9**	**9.2**	**11.9**	**16.5**	**31.5**

Table 18: Variation in performance on existing suffix identification and literary retrieval datasets with model size and minibatch size (number of negative samples). Overall, we see that scaling both model size and minibatch size improves suffix identification performance. See Table 10 for comparisons with non-RankGen baselines.

Model Size	Batch Size	pg19-random 2-way	pg19-random 11-way	pg19-hard 2-way	pg19-hard 11-way	wiki-random 2-way	wiki-random 11-way	wiki-hard 2-way	wiki-hard 11-way
*(*RANKGEN *models trained on PG19)*									
base	4096	98.6	91.7	69.4	36.8	88.4	57.0	65.6	25.7
large	4096	99.0	94.2	76.0	46.4	91.3	66.3	69.7	32.7
XL	1536	**99.1**	**94.4**	**78.0**	**49.5**	**92.3**	**69.0**	**71.4**	**35.7**
*(*RANKGEN *models trained on all 4 domains)*									
base	4096	97.9	88.4	63.5	29.8	95.6	77.8	74.7	42.3
large	4096	98.6	92.1	**68.6**	39.3	97.0	83.7	**79.1**	50.7
XL	256	96.8	83.7	60.3	26.0	95.0	75.9	73.5	39.8
XL	512	97.7	87.8	63.1	31.6	96.1	80.0	76.0	45.0
XL	768	98.1	89.7	64.7	34.2	96.6	82.1	77.6	48.2
XL	1536	**98.7**	**92.6**	61.3*	**39.5***	**97.3**	**84.6**	77.2*	**52.1***

Table 19: Variation in performance on our PG19 / Wikipedia suffix identification datasets with model size and minibatch size (number of negative samples). Overall, we see that scaling both model size and minibatch size improves suffix identification performance. See Table 1 for comparisons with non-RANKGEN baselines. * Note that these numbers are lower since hard sets were adversarially constructed using this RANKGEN variant.

		Generator Language Model (re-ranking 20 nucleus samples)				
	batch size	GPT2-md	GPT2-XL	T5-XXL-PG19	T5-XXL-C4	Average
*(*RANKGEN *models trained on PG19 and evaluated on PG19 prefixes)*						
base	4096	**0.784**	**0.775**	**0.946**	0.722	0.807
large	4096	0.771	**0.776**	0.934	0.734	0.804
XL	1536	0.763	0.752	**0.943**	**0.807**	**0.816**
*(*RANKGEN *models trained on all 4 domains and evaluated on Wikipedia prefixes)*						
base	4096	83.8	83.0	90.1	87.4	86.1
large	4096	**86.3**	**85.8**	**92.0**	**88.5**	**88.1**
XL	256	81.5	0.842	89.7	87.9	85.8
XL	512	82.5	0.845	90.2	87.3	86.1
XL	768	81.0	**0.851**	89.7	87.8	85.9
XL	1536	83.9	**85.7**	**91.8**	**88.1**	**87.3**

Table 20: Variation in MAUVE score of top-ranked generation (among 20 nucleus samples with $p = 0.9$) using RANKGEN variants having a different model / minibatch size. On average, increasing model size and minibatch size boosts performance, but the trend is less prominent than in other tasks. However, all RANKGEN variants outperform baselines like nucleus sampling (see Table 3 for details).

Model	batch size	GPT2-md	GPT2-XL	T5-XXL-PG19	T5-XXL-C4	Average
*(*RANKGEN *models trained on PG19 and evaluated on PG19 prefixes)*						
PG19-base	4096	84.4	78.3	68.3	70.9	75.5
PG19-large	4096	93.7	87.9	79.1	81.3	85.5
PG19-XL	1536	**97.4**	**93.7**	**87.4**	**89.7**	**92.1**
*(*RANKGEN *models trained on all 4 domains and evaluated on Wikipedia prefixes)*						
all-base	4096	71.9	68.2	88.2	60.0	72.1
all-large	4096	80.4	74.7	93.0	64.7	78.2
all-XL	256	73.4	68.8	88.8	60.7	72.9
all-XL	512	78.5	73.6	93.1	64.3	77.4
all-XL	768	81.9	76.1	**95.4**	65.8	79.8
all-XL	1536	**84.5**	**78.0**	95.3	**67.3**	**83.7**

Table 21: Variation in human-written text identification (vs machine generated with $p = 0.9$) performance with model size and minibatch size (number of negative samples). Overall, we see that scaling both model size and minibatch size improves human text identification performance. See Table 2 for comparisons with causal LMs.

View instructions

Context: "Now then," she continued, as the doctor joined their party. "I don't quite know what I ought to say to you, Admiral. You want some very plain speaking to." "'Pon my word, ma'am, I don't know what you are talking about." "The idea of you at your age talking of going to sea, and leaving that dear, patient little wife of yours at home, who has seen nothing of you all her life! It's all very well for you. You have the life, and the change, and the excitement, but you don't think of her eating her heart out in a dreary London lodging. You men are all the same." "Well, ma'am, since you know so much, you probably know also that I have sold my pension. How am I to live if I do not turn my hand to work?" Mrs. Westmacott produced a large registered envelope from beneath the sheets and tossed it over to the old seaman. "That excuse won't do. There are your pension papers. Just see if they are right."

Text 1: "And just what is this?" "Mr. Westmacott told me. You will have to bring me some more to make my pension." "Well, then, let me take them." "That's all right. You won't need anything else to live on. I should go straight for your daughters, and I'm sure you will learn how to cope with the Navy in their turn." "And you'll be able to bring her home," she urged. "Ah! You are going to meet my little daughter. Is she not going to be quite spoiled?

Text 2: "Oh, yes, ma'am, they are right. I have only counted them a hundred times. There is £1,400 in them." "Of course it is right, my dear old Admiral. I can live out of it. In fact I never knew it was in there before I went back to sea. There were no monies in there, and I assumed that it would have been there if you had gone to sea in the first place, and that is the reason I sold it. It was always better to take the money out of the army than the navy.

Enter 'Text 1' or 'Text 2' followed by a 1-3 sentence explanation.

Submit

Figure 10: The interface shown to Upwork freelancers for human evaluation. We used Amazon Mechanical Turk Sandbox to collect our annotations (note that we use the MTurk Sandbox interface only; we do not hire any workers from MTurk due to poor annotation quality (Karpinska et al., 2021).

Prefix (**PG19**) *A Young Inventor's Pluck* by *Arthur M. Winfield*: "Oh, I only wanted to know," replied the young man, apparently unconcerned. "Pooler said I was never to speak of it," returned Meg. "He thinks I don't know where it is, but I do." "Show it to us, will you?" asked Mont, eagerly. Meg shrank back. "It's haunted!" she exclaimed. "Besides, Pooler would kill me if he found it out." "We'll never tell him, will we, Jack?" "Certainly not," replied the young machinist, to whom this new move was only another mystery. "Besides, we intend to be your friends, you know," continued Mont. "Well, then, come on," replied Meg, finally. "I don't care if he likes it or not. 'Taint a very nice place though." "Never mind. We can't find any worse or more dangerous places than we have already gone through." Meg picked her way along one bank of the island for nearly a quarter of a mile. Jack and Mont followed closely.

Text 1 (RankGen, T5-XXL-C4): The path led through an area of woods, shrubbery and scrub oak. Neither meadow nor open ground was seen. Jack called out as they scampered around an occasional rough rock, often from the height of a dozen feet. "I think I can see it now," said Meg, suddenly, as they rounded a grassy slope and followed another bank. A few yards farther on she stopped, and glared at the clear sky and the undisturbed valley. "That's the place," she said. The young man could tell she had a point.
Text 2 (Nucleus, T5-XXL-C4): Jack would not let her out of his sight. He made a sharp look at Meg as she walked forward. Then, his expression became grave. "Meg," he whispered, "are we going to the haunted house or not?" "This is the haunted house, Jack," she replied. "Is it haunted?" "Of course it's haunted," returned Mont, "but if you were so scared you would be shaking all over." "Of course it's haunted," replied the young man.

Human Annotation 1: Text 1. Second text appears to omit a part when they get to their destination having them appear at their destination instantly. Although a haunted hous sounds more in-line with the sample text than a haunted valley, the first text fits more naturally with the sample text.
Human Annotation 2: Text 1 - Meg is taking them somewhere - so leading them would make sense. 'That's the place' indicates she had found it. Jack does not sound worried in prefix text, so it would not really fit that he doesn't let Meg out of his sight as indicated in Text 2.
Human Annotation 3: Text 2. The collocations and style in Text 1 sound too colloquial and off when compared to the prefix. "(...) he had a point. "

Prefix (**PG19**) *A Young Inventor's Pluck* by *Arthur M. Winfield*: But at that moment, he did not stop to question how the young man had gotten into a position similar to his own. His one thought was whether or not his friend was alive. He placed the body as comfortably as possible on the fork of the tree, and then watched eagerly for some sign of life. There was a cut upon Mont's brow, and presently a few drops of blood oozed from the wound, and trickled down his cheek. Jack took this as a good sign, and he was not mistaken, for a few seconds later the young man gave a deep sigh, and slowly opened his eyes. "Where am I?" he asked, faintly. "Here with me, Mont," replied the young machinist, bending over him. "Don't you know me?" "Jack!" was the low response. "Oh, how my head hurts! Where are we?" "On the river." "We are?" Mont attempted to rise, but Jack pushed him back. "Oh, I remember now!" he continued, shuddering. "Remember what?"

Text 1 (Nucleus, GPT2-md): "Mont, when I was twenty-two, I met some girls who took care of my grandfather, and when I wanted to give a hand in the shop, I got him to promise to do it for me for free, if he liked me, or give me money. And now, when I tell you I feel a sense of belonging, Jack, I have been a slave for four hundred years. I have been born of slavery, and yet I feel like a slave." Jack listened to this.
Text 2 (RankGen, GPT2-md): "That time I tried to drag you off my back and then went back on to kill myself." "That is wrong," replied the young machinist, trembling. "I was in your arms. I never intended to kill myself. It happened." "It was a mistake, then," said Jack with a shake of his head. "That was just the first of many," said Mont. The young man looked at him sadly. "Don't lie, Jack," he said gently. "You are not dead. You are alive. What's going on?

Human Annotation 1: Text 2 - Text 1 seems too mundane to be the memory on waking after being knocked out. Text 2 would fit a little better perhaps.
Human Annotation 2: Text 2 - Text 2 was chosen via deductive reasoning. Text 1 mentions ideas that have nothing to do with the prefix paragraph.
Human Annotation 3: Text 2. Text 2 answers the question that was asked at the end of the paragraph and continues with the same characters.

Table 22: More model generations (Part 1) from each decoding method with human preferences and explanations. See our official Github repository (link on Page 1) for all annotations.

Prefix (PG19) *Life in the Iron-Mills* by *Rebecca Harding Davis*: While she was skinning the potatoes, and munching them, a noise behind her made her stop. "Janey!" she called, lifting the candle and peering into the darkness. "Janey, are you there?" A heap of ragged coats was heaved up, and the face of a young girl emerged, staring sleepily at the woman. "Deborah," she said, at last, "I'm here the night." "Yes, child. Hur's welcome," she said, quietly eating on. The girl's face was haggard and sickly; her eyes were heavy with sleep and hunger: real Milesian eyes they were, dark, delicate blue, glooming out from black shadows with a pitiful fright. "I was alone," she said, timidly. "Where's the father?" asked Deborah, holding out a potato, which the girl greedily seized. "He's beyant,–wid Haley,–in the stone house." (Did you ever hear the word _jail_ from an Irish mouth?) "I came here.

Text 1 (Nucleus, T5-XXL-C4): I needed to find my feet." "What have you done here?" Deborah was amazed. "Whoever came in here yesterday found his feet," she replied, "and went away,–not hearing my voice." "What a wonder! What chance do you have of hearing your own voice?" She went over to the fire, lowered it, and began to put the potatoes on it, by degrees, till she had covered all the floor. "Why did you come here?" Deborah inquired, as she looked at the girl's face. "Just for the night. Where's the father?
Text 2 (RankGen, T5-XXL-C4): I came to ask for food. Hur git me?" "You don't see it so bad," said Deborah, leaning her long arms about the girl's small shoulders. "You're poor." "That's true," the girl said, weeping softly. "Hus poor is no better than a beast's. I'd be well if they'd only set food on the table," she added, trembling again. "Evil grows in the dark," said Deborah sadly.

Human Annotation 1: Text 2. Both bit chaotic and funny, but the second one seems at least more outlandish. Being poor or saying Hur and Hus isn't as bad as putting potatos in fire that's covering the floor.
Human Annotation 2: Text 2 - Prefix text says she is hungry, so it would make sense that she would say she came to ask for food.
Human Annotation 3: Text 2. Deborah already told the girl in the prefix part where the father is, but she asks again in Text 1. So Text 2 makes more sense.

Prefix (Wikipedia) *Homarus Gammarus*: It may grow to a length of and a mass of , and bears a conspicuous pair of claws. In life the lobsters are blue, only becoming "lobster red" on cooking. Mating occurs in the summer, producing eggs which are carried by the females for up to a year before hatching into planktonic larvae. Homarus gammarus is a highly esteemed food, and is widely caught using lobster pots, mostly around the British Isles. Homarus gammarus is a large crustacean, with a body length up to and weighing up to , although the lobsters caught in lobster pots are usually long and weigh . Like other crustaceans, lobsters have a hard exoskeleton which they must shed in order to grow, in a process called ecdysis (molting). This may occur several times a year for young lobsters, but decreases to once every 1–2 years for larger animals. The first pair of pereiopods is armed with a large, asymmetrical pair of feet.

Text 1 (Nucleus, T5-XXL-C4): They are armed with two large claws, whose upper surfaces are split into segmented claws, to allow them to grasp with maximum force. The claws are bent backwards when not in use, to keep them from rolling off the lobster's body, and are hooked into the shell. One claw can also bear a thick, sharp biting claw (known as the "hammer claw"). Both claws contain a fan-shaped set of “feathered” hairs, arranged around a central spine.
Text 2 (RankGen, T5-XXL-C4): The foot is covered in spines, and each skeletal segment has a claw (which is a pair of claws in an asymmetrical arrangement). The claws bear nails (trapezoids) and powerful radulae, and are able to claw prey through a small hole in the ground. The second pair of pereiopods has a small foot, with three claws arranged symmetrically. The foot of the second pair is covered in concave spines which are longer and thinner than those of the first pair.

Human Annotation 1: Text 2. Because "The first pair of pereiopods is armed with a large, asymmetrical pair of feet. They (the feet???) are armed with two large claws..) just sounds weird.
Human Annotation 2: Text 2 - prefix mentions the feet and text 2 describes the foot. Text 1 just sounds odd as a follow on from prefix.
Human Annotation 3: Text 2. Both text seem to be very factually incorrect, but the second one at least appears more like a continuation of the sample text. The first text goes back to talk about the claws, while the second talks about feet and second pair of pereiopods.

Table 23: More model generations (Part 2) from each decoding method with human preferences and explanations. See our official Github repository (link on Page 1) for all annotations.

Prefix (**PG19**) *The Horse in History* by *Basil Tozer*: Two years after James I. had ascended the throne there set in one of the coldest winters this country has ever known, with the result that a long stretch of the River Ouse became frozen over and so afforded the king an opportunity, of which he was quick to avail himself, of organising a race-meeting on the ice. Drake tells us that the course extended "from the tower at the end of Marygate, under the great arch of the bridge, to the crane at Skeldergate Postern." But even so early as this in the reign of King James the opponents of horse racing began to raise indignant protests against "the folly and wickedness of betting on running horses," protests to which but scant attention was paid. Not until some years later did the extremely zealous clergyman named Hinde set seriously to work to denounce the practice of gambling in any and every form, and he appears then to have spoken and written so forcibly that many persons of intelligence and education—I quote from a trustworthy source—gathered round and strove to encourage him to the best of their ability.

Text 1 (RankGen, GPT2-md): He did not, however, heed their advice and instead turned his attention to betting. "The king, who at that time was of a temperance and reform bent, had taken into his royal commission the recommendation that betting and horse-trading, while in principle harmless, be avoided. Such was the feeling among the clergy at that time as well as among the people, that in their opinions bettors and dealers, when dealing openly with men, did more harm than good.
Text 2 (Nucleus, GPT2-md): " Here again I have a record that can be said to be worthy of mention of a high level. There is one point which the facts of the situation require me to address, namely that from that day till our own day it is an object of our interest to try to remove all evidence of a race-course. That purpose is, I believe, the greatest thing that we can hope to attain in the course of this study.

Human Annotation 1: Text 1. The original paragraph mentions a "zealous clergyman" and "denounce the practice of gambling", and text 1 continues the discourse about "the clergy at the time" and "their opinions betters and dealers". Text 2 does not follow the same discourse, nor does it provide more information in prefix.
Human Annotation 2: Text 2 - prefix text is setting up for a quote with 'I quote from a trustworthy source, then Text 2 is the start of a quote as it begins with quotation marks.
Human Annotation 3: Text 1; Text 1 continues the narrative established around horse-race betting established in the prefix. The prefix mentions, "the opponents of horse racing began to raise indignant protests against [betting]", then, Text 1 continues this idea when it states, "He did not, however, heed their advice and instead turned his attention to betting".

Prefix (**Wikipedia**) *Thelnetham Windmill*: The mill is a small four storey tower mill with Patent sails and winded by a fantail. It drives two pairs of millstones, with a third pair driven by an auxiliary engine. The earliest record of a windmill in Thelnetham was a post mill which was shown of Joseph Hodgkinson's map of Suffolk of 1778. In 1818, the post mill was moved to Sandy Lane, Diss, Norfolk The post mill was replaced by a tower mill, which was built by millwright George Bloomfield for William Button in 1819. The mill was set to work on Christmas Day 1819. In 1832, the mill was modernised by the fitting of a cast-iron windshaft, Patent sails and a fantail. The new windshaft was fitted on 16 July 1832. It was made by J Aickman, the King's Lynn millwright, and weighs 1 3/4 tons (1,780 kg). A new stock was fitted in September 1836. William Button died on 11 February 1837. The mill passed jointly to his widow Rebecca and their son Richard.

Text 1 (Nucleus, GPT2-md): In 1841, it was renamed The Postmill Mill in memory of its deceased owner. The mill has been restored to its original function. It is now used to run turbines for a turbine company. The mill was built for William Button. He used to work on it during the day, but he would also travel by carriage. In the years around 1816-17, he visited several other mills in the town, including his father's old mill. He bought it in 1810. This mill was not as modern as the mill on the left.
Text 2 (RankGen, GPT2-md): Richard built the mill and sold it to John Beattie on 21 May 1845. On 8 June 1854, Thomas Moore purchased the mill from the Beattie family. In 1861 the mill was owned by the Earl of Dartmouth who rebuilt it as a tenement mill for a number of years, then sold it to a local paper company for £60 a week to pay for repairs. In 1913, the mill was purchased by David Cottrell. There is a stone chimney and iron grate set into the front wall of the mill.

Human Annotation 1: Text 2. Text 2 follows the original style of the paragraph by continuing the history of the mill, while Text 1 is redundant and repeats some things from the original text. Due to following the same pattern as the original paragraph, text 2 is the best choice.
Human Annotation 2: Text 2 - I really don't know, neither seem to fit. Text 1 says ' The mill was built for William Button' but we've just been told William Button is dead so that seems to be a little odd to mention, so Text 2 is a better choice.
Human Annotation 3: Text 2; Text 2 is the better contintuation of the prefix. Prefix ends with mentioned that, "The mill passed jointly to his widow Rebecca and their son Richard. Then, the Text 2 begins with, "Richard built the mill" - this connects the prefix to Text 2.

Table 24: More model generations (Part 3) from each decoding method with human preferences and explanations. See our official Github repository (link on Page 1) for all annotations.

Prefix (Wikipedia) *Daniel Radcliffe*: Radcliffe's performance in the show earned him Drama Desk Award, Drama League Award and Outer Critics Circle Award nominations. The production itself later received nine Tony Award nominations. Radcliffe left the show on 1 January 2012. Radcliffe's first post-Harry Potter project was the 2012 horror film The Woman in Black, adapted from the 1983 novel by Susan Hill. The film was released on 3 February 2012 in the United States and Canada, and was released on 10 February in the UK. Radcliffe portrays a man sent to deal with the legal matters of a mysterious woman who has just died, and soon after he begins to experience strange events and hauntings from the ghost of a woman dressed in black. He has said he was "incredibly excited" to be part of the film and described the script as "beautifully written". In 2013, he portrayed American beat poet Allen Ginsberg in the thriller drama Kill Your Darlings, directed by John Krokidas.

Text 1 (RankGen, T5-XXL-C4): The film premiered at Sundance Film Festival in January 2013, where it won two awards. The film follows Ginsberg during the 1959 American poets' conference at Bennington College. Radcliffe next appeared in the film adaptation of Mary Shelley's Frankenstein. Following the film's premiere at the Venice Film Festival on 21 August 2013, it was released on 5 October 2013. Radcliffe also starred as William H. Macy in the animated adaptation of Bram Stoker's Dracula in 2013. The film was directed by Christian Rivers and was released on 27 July 2014.
Text 2 (Nucleus, T5-XXL-C4): He described the film as "so much fun, and a reminder that life after Hogwarts is exciting and fun". He also starred in the 2016 thriller The Woman in the Window, which was produced by his agent Lydia Dean Pilcher, in which he plays the titular role of a boy who is sent to the country to be married to his pregnant aunt. In March 2018, Radcliffe appeared in the romantic comedy/drama Imposter, co-starring Rooney Mara.

Human Annotation 1: Text 1. The second text says he plays a titular role, while the title is "The Woman in the Window", being a contradiction. First text has no major issues and follows very similar structure of listing acting jobs Radcliffe had after the unnamed show (be that true or not).
Human Annotation 2: Text 1. The quotation at the beginning of Text 2 seems pretty off, as it follows the exact same structure as the quote in the prefix section - the movie + a quote in the movie. Also, the two quotes together make Radcliffs vocabulary seem limited to the word "exciting" and its cognates. Text 2 sounds automatical and forced. Plus, the "so much fun" bit sounds pretty childish for a grown man, especially together with the Hogwarts remark.
Human Annotation 3: Text 1 - This refers to the film date from prefix, 2013, the same film presumably as it mentioned Ginsberg, and with it being a thriller, it may not perhaps be described as being 'so much fun' .

Prefix (PG19) *The Making of Arguments* by *J.H. Gardiner*: But besides the omissions there were only six among the forty which did not contain positively wrong statements; in twenty-four papers up to ten per cent of the statements were free inventions, and in ten answers–that is, in one fourth of the papers–more than ten per cent of the statements were absolutely false, in spite of the fact that they all came from scientifically trained observers. Only four persons, for instance, among forty noticed that the <DW64> had nothing on his head; the others gave him a derby, or a high hat, and so on. In addition to this, a red suit, a brown one, a striped one, a coffee- jacket, shirt sleeves, and similar costume were invented for him. He wore in reality white trousers and a black jacket with a large red neck-tie. The scientific commission which reported the details of the inquiry came to the general statement that the majority of the observers omitted or falsified about half of the processes which occurred completely in their field of vision.

Text 1 (RankGen, T5-XXL-C4): That is, they gave the impression that they had not seen a thing; or they had only seen a part of it, and put it down wrongly; or they only saw what they wanted to see, and presented a falsified impression of a given process. Only one third of the papers gave correct information, but that figure is based on the assumption that an observer watches more than half the processes at a time. How can it be explained that the observers give a false impression of processes?
Text 2 (Nucleus, T5-XXL-C4): It also came to the conclusion that the <DW64>, since he was not standing still in a given position, did not exert his full influence on the proceedings of the people within his area of activity, and that there was no need for special observations on the phenomena. But the whole suggestion of the Commission was not well received. There was, the commission declared, a noticeable scepticism about their conduct. Thus a report of the commission was sent to the official newspaper, and three days later the secretary of the observatory came to it and spoke frankly about the commission's conduct.

Human Annotation 1: Text 1. It seems to be closer aligned with the prefix, covering pretty much the same points. It still sounds off, with the numbers not adding up (Only one third of the papers gave correct information, but that figure is based on the assumption that an observer watches more than half the processes at a time. «< different numbers in the prefix). Also the rhetorical question at the end discrupts the style. However, Text 2 with this fragments: " did not exert his full influence on the proceedings of the people within his area of activity, and that there was no need for special observations on the phenomena" makes even less sense.
Human Annotation 2: Text 1. Second text has an awkward "word" in it followed by a convoluted explanation of something that appears only vaguely related to the sample text. Latter part of text 2 talks about the conduct of the commission while most of the text 1 sticks to talking about the results.
Human Annotation 3: Text 1 - This continues the concept that observers are not remembering what they saw, or not truly seeing but guessing. Text 2 is about something else.

Table 25: More model generations (Part 4) from each decoding method with human preferences and explanations. See our official Github repository (link on Page 1) for all annotations.

Prefix (**PG19**) *Letters of Lord Acton* by *Lord Acton*: In that character he showed, when occasion came, that his lxvi long silence in Parliament had not been due to incapacity for public speaking. At Windsor he was agreeable to the Queen from his German tastes and sympathies, not to mention the fact that he could speak German as fluently as English. Every moment of leisure during his "wait" there was spent in the Castle library. Yet the position was an unnatural one, and Lord Acton soon became anxious to escape from it. His thoughts turned to his favourite Bavaria, and he humbly suggested the Legation at Stuttgart as a possible sphere. But something infinitely better than any political or diplomatic post remained for this born student and truly learned man. In 1895, just a year after Mr. Gladstone's resignation, Sir John Seeley, Professor of Modern History at Cambridge, departed this life. The Chair was in the gift of the Crown, that is, of the Prime Minister, and Lord Rosebery appointed Lord Acton. The appointment was singularly felicitous, and the opportunity came in the nick of time.

Text 1 (RANKGEN, GPT2-md): Lord Acton entered upon his first year in the Chair, and the year was filled with extraordinary events, the first of which was the assassination of King George V, which had so disastrous an effect upon the world that it has never since been fully restored. Then came the attempt on the life of Sir Edmund Burke at St. Petersburg, the downfall of William of Orange, and the beginning of the Spanish War.
Text 2 (Nucleus, GPT2-md): The "youngest and most handsome man of his class", as they called him, had been nominated to succeed him. Mr. Gladstone, whose wit had been so well known for many years, was now at last able to be seen in public with dignity, and his manner was becoming more relaxed. The Prince felt confident enough to nominate Sir John, who was the most distinguished scholar in Bavaria, and he received his nomination in person. There were some curious features of the office.

Human Annotation 1: Text 1 - The prefix paragraph focuses on the life and career of Lord Acton, thus, Text 1 is the best selection to continue exposition of Lord Acton's career.
Human Annotation 2: Text 1 - Prefix states that Lord Acton was appointed as chair, Text 1 then mentions Lord Acton entering his first year as chair. This seems to follow more closely than Text 2.
Human Annotation 3: Text 1. At the end of the paragraph, Lord Acton is appointed Chair and text 1 delves into what occurred once that was fact. Text 2 does not closely follow the narrative of the paragraph.

Prefix (**Wikipedia**, *Battle of Roanoke Island*): They arrived too late to take part in the battle, but were involved in the attack on the fortress at Pao-ting Fu, where the Chinese government was believed to have found asylum after Peking was captured by western forces. The Victorians joined a force of 7,500 men on a ten-day march to the fort, once again only to find that it had already surrendered. The Victorians then garrisoned Tientsin and the New South Wales contingent undertook garrison duties in Peking. HMCS Protector was mostly used for survey, transport, and courier duties in the Gulf of Chihli, before departing in November. The naval brigades remained during the winter, unhappily performing policing and guard duties, as well as working as railwaymen and fire-fighters. They left China in March 1901, having played only a minor role in a few offensives and punitive expeditions and in the restoration of civil order. Six Australians died from sickness and injury, but none were killed as a result of enemy action.

Text 1 (Nucleus, GPT2-md): In July 1901 they began their work in the Indian Ocean, arriving in Hong Kong in November 1901. In January 1902, the Australian Expeditionary Force joined the Chinese in a massive invasion of Peking. While Australian naval forces took an active part in this event, the Australian Navy was still engaged in military operations in the Gulf of Hainan. When the Chinese invaded Hong Kong, the navy withdrew from the mainland and its warships were deployed at the port of Hong Kong until December 1902.
Text 2 (RANKGEN, GPT2-md): A second invasion of the Chinese mainland was attempted by British naval forces on 23 June 1901 when Victoria signed a treaty with China. Following the treaty signing, a force led by HMAS Sault Ste. Marie was sent to occupy Peking and Tientsin. HMAS San Francisco, HMAS Mackellar and HMAS Melbourne returned to Hong Kong after a two-month deployment in China in early 1902 and were replaced by a group of 14,000 men under HMCS Lendl, which was formed on 24 November 1902 as part of the second invasion.

Human Annotation 1: Text 2; Text 2 is the better continuation of the prefix. In Text 1, it isn't clear who "they" is in the phrase, "they began their work in the Indian Ocean" which makes Text 1 appear disjointed when reading directly after the prefix whereas Text 2's introduction flows more seamlessly even though it's introduction brings a slight change in idea.
Human Annotation 2: Text 1. Although both texts could follow the paragraph, Text 1 follows along with the timeline set in the paragraph.
Human Annotation 3: Text 2 - very difficult without more knowledge of these events. I'm picking text 2 just because the date mentioned, 23 June 1901, is closest to the date mentioned in prefix text - march 1901

Table 26: More model generations (Part 5) from each decoding method with human preferences and explanations. See our official Github repository (link on Page 1) for all annotations.