Decoding Schemes for Conditional Language Generation: An Exploration of Bi-directional Decoding and Sparse Attention

Daniel Saggau
Department of Statistical Science, LMU Munich
`daniel.saggau@campus.lmu.de`

Abstract

This paper examines different two problems within conditional text generation. I focus on two issues: (1) Embedding past and future context rather than just constraining on the past and (2) Embedding long sequences. The first paper [9] deals with non-monotonic reasoning with bi-directional decoding, referred to as DeLorean decoding. [9] explore the performance of their decoding scheme on two tasks: (a) abductive reasoning and (b) counterfactual reasoning. I replicate this study for both tasks. I add to the evaluation by adding the Bleurt metric, an evelauation metrics that combines signals from various performance metrics. The second approach I look at is text summarization for long documents [13]. [13] focus on sparse attention. I replicate existing results for BigBird-Pegasus for 3 datasets. I look at *'Pubmed', 'Arxiv', 'Big Patent'* and also apply the model to a new summarization dataset: *'Billsum'*. The benchmark also adds the Bleurt metrics for evaluation. Further, I adjust the decoding scheme and benchmark the performance for various hyper-parameter specifications. This research proposes further research in the a combination of DeLorean decoding and text summarization strategies for improved text summarization.

1 Introduction

One of the fundamental challenges in conditional text generation is the correct representation of context to condition on. One issue is the incorporation of bi-directional context. Most existing models only account for the past context. [9] focus on understanding non-monotonic reasoning, looking at abductive reasoning and counterfactual reasoning. The abductive reasoning task here is the abductive reasoning for natural language generation task. [3] proposed the abductive NLG task. [3] use the *ART* dataset for abductive NLG and NLI. [8] proposed the counterfactual reasoning task, using the *TimeTravel* dataset. Backward and forward passes allow for bi-directional constraints. The backward pass ensure consistency and the forward pass ensures fluent continuation. Another existing problem in conditional text generation task is the incorporation of long-sequence context. The attention mechanism in the transformer architecture scales quadratically O(n^2), making long sequences troublesome. Text summarization methods explore approaches to include longer context. One approach is using sparse attention, leading to computational cost of O($n \times r$) and allowing for the incorporation of longer context. Further, recent studies suggest task specific pre-training. [14] argue that masking sentences rather than tokens leads to better performance on text summarization. In this paper, I explore decoding schemes for long text summarization. I examine the performance of BigBird Pegasus on three summarization tasks: '*pubmed*', '*Big Patent*' and '*arxiv*'. This study proposes further work on a combination of these decoding methods for better contextual language modeling. By utilizing backward and forward passes as proposed by [4] and [9] could allow for bidirectional conditioning. Combining this with sparse attention from [13] and/or a divide and conquer strategy as proposed by [5] could lead to further improvements in conditional text generation.

2 Literature

2.1 Bi-Directional Decoding

Most language models only condition on the past. One strategy to condition both on the past and future is bi-directional decoding by updating logit weights through multiple forward and backward passes. Subsequently, text is sampled based on a combination of these passes. [4] look at forward and backward passes for supervised conditional language generation. Similarly to [9], [4] update the logit weights.

[9] contribute to the existing literature by looking at unsupervised conditional language generation. Note that neither approach is confined to a specific transformer structure. The input of the model is a pre-trained language model (LM). Further one needs to provide the model with Context X and future constraints Z. The model undertakes forward and backward passes and then combines the representation in one vector. Thereafter, the model samples from this vector. The authors merely update the logit weights and not the actual word embeddings.[1] To accommodate the surrounding context in a group of sentences [9] use back propagation to re-calibrate the logits in the decoding step. Logits are probability weights between 0 an 1 for the different words in the word embedding.

Counterfactual Reasoning: Counterfactuals deal with hypothetical scenarios. In this case we deal with what would happen, when a different sentence occurs, changing the context. We feed the model 3 sentences. The first two sentences are the original story. The third sentence is the modified context. This modification results in causal change in the future context. The goal of this conditional generation task to minimally edit the ending, but generates new sentences that account for this causal change. Equation 1 shows the loss function for the counterfactual reasoning task. Here we are using a softmax loss. The authors use the Kullback–Leibler divergence to approximate the original ending. This method ensures that we get a similar but not identical ending. X is the story context. Y is the rewritten ending. Z is the original ending.

$$\mathcal{L}(X, \tilde{Y}, Z) := \mathrm{KL}(Z \| softmax(\tilde{Y}/\tau)) \tag{1}$$

Abductive Reasoning: The abductive reasoning task provides 2 sentences and the model is supposed to fill the sentence in the middle. The causal reasoning looks at counterfactuals. For the abductive reasoning, the authors use the loss function in equation 2. X is the past observation/context. Y is the Hypothesis. Z is the future observation.

$$\mathcal{L}(X, \tilde{Y}, Z) := -\sum_{n=1}^{N_Z} \log P_{\mathrm{LM}}\left(z_n \mid X, \tilde{Y}, Z_{1:n-1}\right) \tag{2}$$

Another pivotal contribution of this paper is the ranking step. This paper uses the BertScore for the next sentence prediction task to rank output and then subset. The original measure that was used in the study is the next sentence prediction task or in short NSP. BERT uses the NSP and masked LM tasks for their training. NSP is a binary classification loss. The purpose of this task was improve performance on downstream tasks. [7] and [12] propose to omit NSP. [12] undertake an ablation study and find that next sentence prediction does not improve their permutation language model performance. [6] propose sentence order prediction. Further research could look at whether a different ranking metric could led to better downstream performance.

2.2 Text Summarization

There are two types of text summarization: extractive and abstractive text summarization. Abstrative text summarization is about creating summaries with new words. Extractive summaries focus on summarizing, based on paraphrasing sentences from the original document. Current practice is that

[1]Future work could also examine performance on updated word embeddings.

we pre-train langauge models and predict masked out tokens. This is referred to as Auto-regressive (self-supervised) language modelling. The underlying assumption of this approach is that this task will translate into better downstream task performance. Recent research suggests, that one size fits all pre-training does not trickle down equally well for all tasks. [10] explore different pre-training tasks and led as an inspiration for Pegasus. [14] propose a novel pre-training method for abstractive text summarization, Pegasus. Abstractive summarization describes a model that creates a summary with new sentences rather than paraphrasing. [14] introduce Gap sentence generation. Gap sentence generation masks out entire sentences rather than just mask some tokens. Further, the authors suggest that mask language modelling does not contribute to their results and omitted the task altogether in their pre-training. Their results propose that one can fine-tune on a much smaller dataset and still result in substantial performance improvements. The model performed SOTA in several tasks including text summarization. This method was combined with a sparse attention mechanism by [13]. BigBird Pegasus is currently ranked second on the arxiv and pubmed text summarization tasks, behind HAT-BART. BigBird combines three different attention types. Random attention, Window attention and global attention. Window attention means that we fix the corners (neighbors) of our attention mechanism. Global Attention means that we have fixed tokens that are connected to everything else. Random attention means that we include random blocks, which fully connect in log time. This approach is Turing complete. The Longformer [2] also used window attention and global attention hence random attention was the major contribution here. Instead of fully connected graph, we are working with a sparsely connected attention. The paper makes up for full attention by adding more layers to the attention mechanism. In a nutshell, the BigBird structure is basically a Longformer with an additional random attention mechanism. The Attention mechanism ends up being O($r \times n$) instead of O(n^2) BigBird Pegasus can handle sequences of up to 4096 tokens. [13] provide pre-trained models for three summarization datasets (Big Patent, Arxiv, Pubmed). BigBird-Pegasus used Adafactor as an optimizer. Further, recent methods look at combinations of this method. One recent approach [5] looks at a divide and conquer strategy. This approach breaks down a long document into smaller chunks and thereafter generates summaries for these chunks. In the final step, this method combines these summaries and creates a large summary using Pegasus. This eases the computational burden of including long sequences and essentially is a recursive solution to long text summarization. [1] apply a novel rewriting approach to modify text summaries. The approach uses the original document as a context. Using information from the original document, the rewriting method modifies the existing summary, adding more information and refining elements. [1] apply this method to the CNN/Dailymail dataset.

3 Modifications

I undertake a number of modifications. To contribute to the existing implementation, I modify the sampling strategy. Therefore, I briefly explore the three core decoding strategies namely greedy decoding, Beam search and random sampling (top-p and tok-k). I strongly build upon the documentation in this huggingface guide: `https://huggingface.co/blog/how-to-generate`. I also add BLEURT as an evaluation metric that correlates stronger with human judgement compared to other traditional metrics. Hence, the upcoming subsection explores different automatic evaluation metrics for reference.

3.1 Decoding Strategies

Greedy Decoding: Greedy search misses high probability words hidden behind a low probability word. Therefore it is usually not recommended as a decoding strategy. We simply use the next most probable word at every step as seen in equation 3.

$$w_t = argmax_w P\left(w \mid w_{1:t-1}\right) \tag{3}$$

Beam Search: This methods selects a number of Beams of hypothesis and then subsequently picks the one with the highest probability. One major issue is repetition. Especially for generation without fixed length, Beam search can be troublesome. Further, there is little variation using Beam search. For further information see `https://huggingface.co/blog/how-to-generate`.

Sampling: In top-k sampling one sort by top-k probabilities and subset the most likely top-k terms. Everything below the kth token is set to 0. Nucleus sampling (top-p sampling) in comparison looks at the smallest possible set of terms. The probabilities are cut off via a threshold looking at the probability of a sum of terms. The probability mass is then redistributed among the remaining terms. The size of the set of words can adjust with respect to the probability distribution of the next word. Mixing top-p and top-k sampling allows one to avoid very low ranked words. This approach simultaneously enabling some dynamic selection. Note, that sampling is not deterministic. Additionally, we can specify a temperature parameter to make the probability densities sharper. [9] use greedy decoding with random sampling via top-k sampling. [13] use Beam search. [9] already use a fairly efficient decoding strategy. Experiments on subsets suggested that performance difference were negligible. Henceforth, I focus on sampling for text summarization as done by [13].

3.2 Automatic Evaluation Metrics

Bleurt: The Bleurt score uses various different metrics and combines their signals to create a new score. [11] recommend the usage of quantiles for the evalaution and subsequent interpretation.[2] Rather than only looking at aggregate scores, this measure lets us look at single observations and then examine in what part of the distribution the score is placed. The score is not normalized and hence also can have negative values. Whether these scores are good or bad depends on the relative position rather than the absolute value. [11] recommend comparing performance by looking at the score for different specifications/models as opposed to evaluating performance in isolation. I report the density charts for the different Bleurt score plots. You can find the chart for ARXIV in the pain paper and the charts for the other tasks in the appendix.

ROUGE: The ROUGE score is the main metric for text summarization. But not every ROGUE score is equally applicable. Therefore, I briefly elaborate on on the different types for further interpretation. The standard ROUGE-score 1 is of little use if we look at abstractive text summarization based on novel sentences as it just reports similarity of unigrams between the prediction and reference. ROUGE-2 looks at bigrams. ROUGE-L looks at the longest common sub-sequence. This measure accounts for sentence level structure similarity.

4 Results: Bidirectional Decoding for Commonsense Reasoning

The first section of the results examines the performance of [9] I replicate the results by [9] using subsets for both tasks. For the counterfactual tasks I generate approximately 500 endings using random samples from the test set. Using a single GPU this took approximately 4 hours. For the abductive reasoning task I generate 1000 endings based on randomly sampled prompts. On a single GPU this took around 8-9 hours. For the quantitative assessment I also use the ROUGE-L Score, Bleurt and the BertScore. BertScore and Bleurt correlates stronger with human judgement as opposed to conservative measures such as BLEU or ROUGE.

	ROUGE-L	BertScore	Bleurt
TimeTravel			
Reported DeLorean	40.73	63.36	-
Replicated DeLorean (Non-Deprecated)	31.19	0.886	-.8552
Art			
Reported DeLorean	18.94	42.86	-
Replicated DeLorean (Non-Deprecated)	17.89	0.874	-0.970

Table 1: Replicated Results on the Test Dataset for ART and TimeTravel

In the table we can see that even just looking at the performance on the non-deprecated sentences, the ROUGE-L score is similar. Here I report the average of the BertScores using Roberta as the BertScore specification. Unfortunately, [9] do not provide further context on how these scores were normalized/computed. The official coding implementation also excludes the performance evaluation step. Therefore, I just left BertScores as is without normalization which explains the different score. Further, we can see that the performance for counterfactual reasoning is slightly better than for abductive reasoning. Note that here I dont only evaluate the best ranked sentence but all sentences that are non-deprecated. Generated Endings that are poorly written because of e.g. repetitions are already removed as part of the ranking step within the model. It is unclear from the paper whether

[2] `https://github.com/google-research/Bleurt/issues/1`

the original paper evaluated all viable generated sentences or only the best one. Possibly, by only taking the best ending, performance would be better.

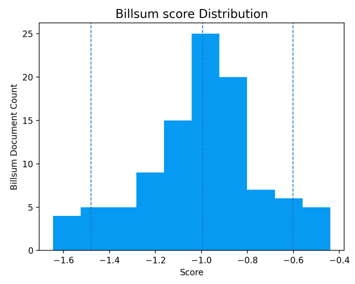
To further understand the model performance, I also provide a brief qualitative assessment. For reference, I include an example of a counterfactual generated ending:
Premise: I received a message on facebook from a random girl. Initial:She loved the Beatles just like me. Counterfactual: She hates the Beatles. Original ending: We went to a Paul McCartney concert together. We started dating shortly after. We have been together for 4 years. **Generated Ending**: She said to me, "I'm going to go to the Beatles concert in London and I'm going to sing the song 'I Want to Hold Your Hand' and I'm going to hold your hand."I said, "I'm not going to do that."

While the model understood that someone hates the Beatles, entity recognition is still troublesome. Further, looking deeper into the structure of the model, one issue is that we cannot really assess the performance using automatic metrics. This premise holds true especially in the unsupervised case. Here the authors include these metrics to examine the minimum edits objective. This means that the generated endings should be similar to the original endings. But, we cannot really ensure that the ending truly accounts for the counterfactual for the first task. Also, looking at the evaluation of the abductive reasoning task, we solely examine that the imputed sentence is similar to the contextual sentences. While using this ranking approach, performance did improve significantly, there are still limitations. Apart from the issue of entity recognition, repetition was a really big issue and led to a lot of deprecated endings for the counterfactual task. One possible solution could be the usage of repetition penalties. I explore this and similar hyper-parameters for the summarization benchmark.

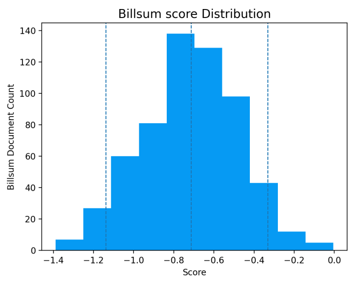
5 Text Summarization

This section looks at the performance of BigBird-Pegasus for 4 different datasets. The selected hyper-parameters can be found in the appendix. I use 600 observations/documents per dataset due to computational constraints. All of the datasets can be found on huggingface. The performance on the subset is slightly worse for the given datasets. The first 3 performance scores are based on the respective pre-trained models. All the replicated models use Beam search. All the sampling models use top-p sampling.[3] Therefore, i omitted it from the table. The fourth model does not have a pre-trained version for this specific task. The authors recommend the usage of the pre-trained BigBird-Pegasus for similar tasks. Henceforth, I here look at two different pre-trained models for the Billsum dataset. The Billsum dataset has not been included in [13]. I use the BigBird-Pegasus-large-arxiv and BigBird-Pegasus-large-Big Patent checkpoints and then evaluate the performance. As one can see in the table 2, performance for billsum is much worse than for the pre-trained models. It seems that using the legal dataset big-patent allows for better performance. Adding the repetition penalty worsens performance, Adding a n-gram repitition penalty to Beam decoding did led improvement, but still performance worse than the big-patent model. I also add the different decoding schemes and benchmark them, reporting the F1 scores for ROUGE-1, ROUGE-2, ROUGE-L and Bleurt score. I also include all further results (recall, precision) in the Github Repository which can be found here: `https://github.com/danielsaggau/deep_unsupervised_learning`

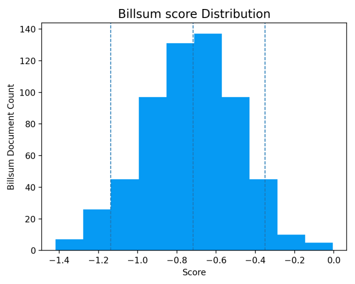
[3]I also experimented with mixed sampling but the performance was identical in this case.



(a) Billsum: BigBird Pegasus fine-tuned on ARXIV using Beam search



(b) Billsum: BigBird Pegasus fine-tuned on Big Patent using Beam search



(c) Billsum: BigBird Pegasus fine-tuned on Big Patent using random sampling

Figure 1: Bleurt Distribution for Billsum Text Summarization

	ROUGE-1	ROUGE-2	ROUGE-L	Bleurt
Pubmed				
Reported BigBird	46.32	20.65	42.33	-
Replicated BigBird	43.94	19.60	26.90	-0.448
BigBird + Sampling	43.93	19.62	26.91	-0.159
BigBird + Sampling + Repetition Penalty	43.94	19.55	26.82	-0.161
Arxiv				
Reported BigBird	46.63	19.02	41.77	-
Replicated BigBird	43.50	17.74	25.77	-0.4993
BigBird + Sampling	43.50	17.74	25.77	-0.4993
BigBird + Sampling + Repetition Penalty	43.47	17.77	25.70	-0.498
Big Patent				
Reported BigBird	60.64	42.46	50.01	-
Replicated BigBird	37.88	15.11	26.12	-0.473
BigBird + Sampling	37.88	15.11	26.12	-0.473
BigBird + Sampling + Repetition Penalty	37.88	15.21	26.09	-0.474
billsum				
BigBird Arxiv	16.06	01.73	12.45	-1.001
BigBird Big Patent	26.92	09.44	19.22	-0.716
BigBird Arxiv + Ngram-Penalty	22.22	03.24	13.20	-0.9698
BigBird Big Patent + Sampling + Repetition Penalty	26.47	09.38	18.94	-0.724

Table 2: Replication results of BigBird Using F1 Measure.

One can see that the performance is significantly worse on billsum compared to the other tasks. Also the performance for Big Patent is significantly worse, despite using the respective pre-trained model. Possibly, one needs to further adjust hyperparameters. Regardless, we can also see that the Big Patent model performance a lot better than the ARXIV model. To further improve performance, one could fine-tune BigBird-Pegasus on the respective dataset.[4] As one can see in figure one, can see a slight improvement by adding the penalty. Also, the range has become sharper when looking at the score on the X-axis. So, while we see some variation in the aggregated scores, we can also see a shift in the distribution looking at the plot.

6 Conclusion and further Research

This paper examines recent advancements in conditional text generation. I focus on two common challenges Firstly, I explore the performance of bi-directional decoding which conditions on forward and backward context. [9] apply this method to unsupervised conditional text generation. Specifically, the authors focus on abductive reasoning and counterfactual reasoning. This paper replicates both tasks and adds Bleurt to evaluate the performance. Thereafter, I examine the challenge of incorporating

[4] I included copied reference code for this task from huggingface in the github repository.

longer context for text summarization. I replicated the performance on three text summarization datasets that were reported in the original paper [13] and one new summarization dataset, *'billsum'*. Further, I explore different decoding schemes and benchmark the performance. Additionally, I also add the BLEURT metrics and plot respective performance difference based on the distribution chart as opposed to only looking at the change in the aggregate score.

Further research could combine these different decoding schemes. [5] suggest a divide and conquer strategy to break down large text into smaller chunks. One could potentially combine DeLorean decoding with a divide and conquer strategy, leading to bidirectional summarization on long sequences. Further, one could also combine [14] pre-training with the idea behind DeLorean decoding. Both the abductive reasoning task and the Gap sentence generation are very similar. Perhaps, one could improve performance by training the model using the ranking step as proposed by [9] and [4]. Moreover, one could apply this combined approach to shared tasks such as Laysumm2020 in future applications.

References

[1] Guangsheng Bao and Yue Zhang. Contextualized rewriting for text summarization. *arXiv preprint arXiv:2102.00385*, 2021.

[2] Iz Beltagy, Matthew E Peters, and Arman Cohan. Longformer: The long-document transformer. *arXiv preprint arXiv:2004.05150*, 2020.

[3] Chandra Bhagavatula, Ronan Le Bras, Chaitanya Malaviya, Keisuke Sakaguchi, Ari Holtzman, Hannah Rashkin, Doug Downey, Scott Wen-tau Yih, and Yejin Choi. Abductive commonsense reasoning. *arXiv preprint arXiv:1908.05739*, 2019.

[4] Sumanth Dathathri, Andrea Madotto, Janice Lan, Jane Hung, Eric Frank, Piero Molino, Jason Yosinski, and Rosanne Liu. Plug and play language models: A simple approach to controlled text generation. *arXiv preprint arXiv:1912.02164*, 2019.

[5] Alexios Gidiotis and Grigorios Tsoumakas. A divide-and-conquer approach to the summarization of long documents. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 28:3029–3040, 2020.

[6] Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. Albert: A lite bert for self-supervised learning of language representations. *arXiv preprint arXiv:1909.11942*, 2019.

[7] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*, 2019.

[8] Lianhui Qin, Antoine Bosselut, Ari Holtzman, Chandra Bhagavatula, Elizabeth Clark, and Yejin Choi. Counterfactual story reasoning and generation. *arXiv preprint arXiv:1909.04076*, 2019.

[9] Lianhui Qin, Vered Shwartz, Peter West, Chandra Bhagavatula, Jena Hwang, Ronan Le Bras, Antoine Bosselut, and Yejin Choi. Back to the future: Unsupervised backprop-based decoding for counterfactual and abductive commonsense reasoning. *arXiv preprint arXiv:2010.05906*, 2020.

[10] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *arXiv preprint arXiv:1910.10683*, 2019.

[11] Thibault Sellam, Dipanjan Das, and Ankur P Parikh. Bleurt: Learning robust metrics for text generation. *arXiv preprint arXiv:2004.04696*, 2020.

[12] Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Russ R Salakhutdinov, and Quoc V Le. Xlnet: Generalized autoregressive pretraining for language understanding. *Advances in neural information processing systems*, 32, 2019.

[13] Manzil Zaheer, Guru Guruganesh, Kumar Avinava Dubey, Joshua Ainslie, Chris Alberti, Santiago Ontanon, Philip Pham, Anirudh Ravula, Qifan Wang, Li Yang, et al. Big bird: Transformers for longer sequences. In *NeurIPS*, 2020.

[14] Jingqing Zhang, Yao Zhao, Mohammad Saleh, and Peter Liu. Pegasus: Pre-training with extracted gap-sentences for abstractive summarization. In *International Conference on Machine Learning*, pages 11328–11339. PMLR, 2020.

A Hyperparameters

Abductive Reasoning: As the pretrained langauge model I pick GPT2-medium. I set the maximum length to 30. The original paper uses top k sampling set to 1. I undertake 20 passes and 20 backward iterations per instance. I select a step size of 0.0003. The authors use a temperature parameter of 1 for their initial rate and for the forward and backward passes.

Counterfactual Reasoning: The reference code suggests 5 and 10 as a number of passes. For both options, one computes 5,8,10 and 15 backward iterations. As the pre-trained language model the authors propose gpt-2 medium. Here the authors use a mixture rate of 0.92. The forward temperature parameter is set to 1. For the sampling strategy, the authors use top k sampling set to 1. The authors propose a step-size of 0.0004.

Text Summarization Task: I use a maximum length of 4096. I set the length of the generated abstract to 256. For the Beam search, I set the number of Beams to 5. For the n-gram penalization, I set the number to 2. For the random sampling, I use the top-p sampling set to 0.95. I also experimented with top k sampling and a mixture between top p and top-k sampling but results were identical to using top-p sampling in isolation. I set the repetition penalty to 1.1. I use a length penalty of 0.8 which is slightly different to the original implementation.

B Appendix - Coding Instructions

For the replication of the DeLorean approach, I use the offical implementation. This decoding strategy is coded in Pytorch. For further information, see: `https://github.com/qkaren/unsup_gen_for_cms_reasoning` The authors do not submit a complete evaluation of the model. Henceforth, I added code for the evaluation. Further, the existing repository does not include the original data but only small snippets for replication. I add the test data from the *'TIMETRAVEL'* Repository which can be found here:`https://paperswithcode.com/dataset/timetravel` The test data for *'ART'* can be found here: `http://abductivecommonsense.xyz` For the BigBird implementation, my code heavily builds upon `https://colab.research.google.com/github/vasudevgupta7/BigBird/blob/main/notebooks/BigBird_pegasus_evaluation.ipynb`.

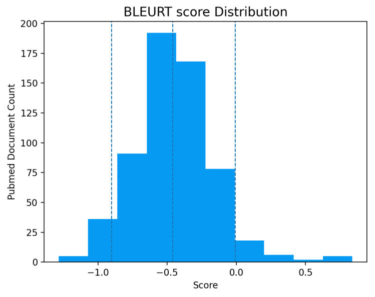
C Appendix

Example Counterfactual Reasoning:
S1: Zeke was throwing a party.
S2: All his friends were dressed up for this Halloween party.
S3: Zeke thought about being a vampire or a wizard.
S4: Then he decided on a scarier costume.
S5: Zeke dressed up like a skeleton.

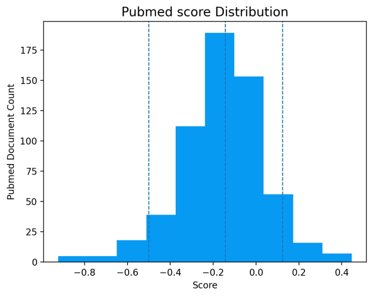
S2': All his friends were dressed up for this Game of Thrones themed party.
S3': Zeke thought about Lannister, but he didn't want to look like a Lannister.
S4':He wanted to look like a Stark.
S5':Zeke dressed up like a Stark.

Example Abductive Reasoning:
Past Observation: Ray hung a tire on a rope to make his daughter swing.
Hypothesis: She hit the rope and the tire fell on top of her.
Future Observations Z: Ray ran to his daughter to make sure she was okay.

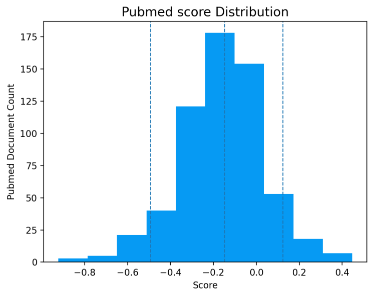
D Appendix - Bleurt Charts



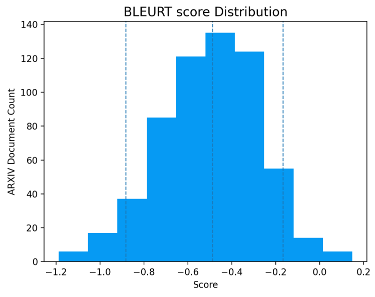
(a) Pubmed: Beam Search



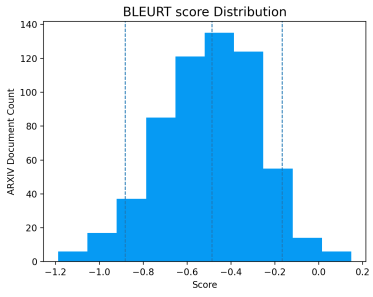
(b) Pubmed: Random Sampling



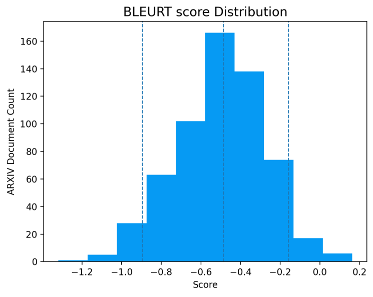
(c) Pubmed: Random Sampling + penalty



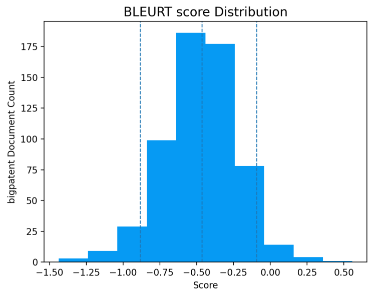
(a) Arxiv: Beam Search



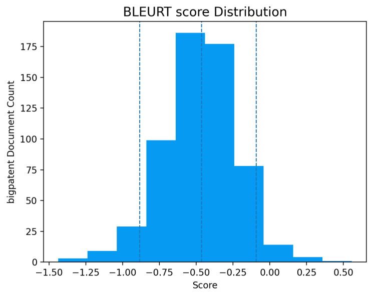
(b) Arxiv: Random Sampling



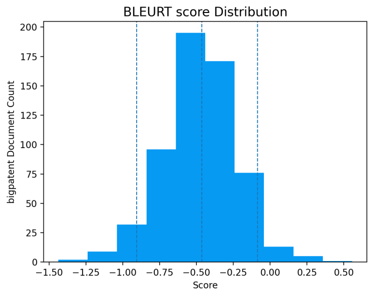
(c) Arxiv: Random Sampling and Repetition penalty



(a) Big Patent: Beam Search



(b) Big Patent: Sampling



(c) Big Patent: Sampling + penalty