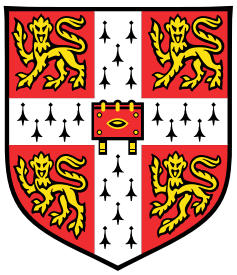
Direct Preference Optimisation for Natural Language Generation with Minimum Bayes Risk Decoding



Fouad Khnaisser

Department of Engineering
University of Cambridge

This dissertation is submitted for the degree of
Master of Philosophy in Machine Learning and Machine Intelligence

Homerton College

February 2025

I would like to dedicate this thesis to my loving parents.

Declaration

I, Fouad Khnaisser of Homerton, being a candidate for the MPhil in Machine Learning and Machine Intelligence, hereby declare that this report and the work described in it are my own work, unaided except as may be specified below, and that the report does not contain material that has already been used to any substantial extent for a comparable purpose.

All software used in this thesis was written in Python and PyTorch. The HuggingFace transformers library[1] was used to download the instruction tuned models. We used the implementation of preference learning algorithms from the HuggingFace trl library[2]. The Summarization CNN/DM and the Question Generation SquadV2 datasets were installed through the datasets library[3]. The Question Answering StrategyQA dataset was downloaded form GitHub[4]. We used the HuggingFace evaluate library[5] for the evaluation metrics. We used the following GitHub repository[6] for the MBR implementation. All remaining scripts were written by me with standard Python packages. Codes are available here.

The word count, including footnotes, figure and table captions is 14033

Fouad Khnaisser
February 2025

[1] version 4.44.0 https://pypi.org/project/transformers/
[2] version 0.9.6 https://pypi.org/project/trl/
[3] version 2.21.0 https://pypi.org/project/datasets/
[4] https://github.com/suzgunmirac/crowd-sampling/blob/main/data/bigbench_strategyqa_test.txt
[5] version 0.4.2 https://pypi.org/project/evaluate/
[6] https://github.com/CyberAgentAILab/model-based-mbr

Acknowledgements

First and foremost, I would like to express my sincere gratitude and my heartfelt thanks to my supervisor Professor Bill Byrne and my co-supervisor Eric Chen. I'm grateful for the opportunity I was given to conduct my thesis under their supervision. Their insights and guidance have been instrumental in shaping the trajectory of the thesis. Their patience and constructive feedback was critical in the early stages of the project, as I was building a solid foundation for the thesis. Moreover, I sincerely appreciate their crucial feedback, that helped me refine my dissertation

Finally, I would like to thank all my college friends that have supported me through this year.

Abstract

In this project we leverage recent advances in Large Language Models (LLM) preference optimisation to align the LLM using a preference set generated through Minimum Bayes risk (MBR) decoding. MBR decoding is a two pass decoding method, that effectively addresses some of the weaknesses of commonly used single pass decoding methods. MBR decoding has shown to enhance the performance and diversity of the generated outputs compared to single pass decoding methods. Yet MBR's superior performance comes at a cost, as a it scales quadratically with the number of samples. This renders MBR impractical to use at inference time.

Unsupervised preference learning using a preference set generated through MBR enables the model to learn the MBR preferences, and match the MBR performance through a single pass decoding algorithm. We validate the efficacy of this method on three different tasks: Question Answering on StrategyQA, Summarization on CNN/DM, and Question Generation on SquadV2. To align the models, we use Direct Preference Optimization (DPO) and Kahneman-Tversky Optimization (KTO). We investigate the performance of the chosen alignment methods with different degrees of regularization and different preference set generation strategies. We find that DPO achieves on all three tasks the best performance. Guided by MBR data properties, we explain why DPO outperforms KTO.

Our work shows the efficacy of unsupervised preference learning using MBR data, as it works across tasks and alignment techniques.

Table of contents

List of figures

List of tables

Chapter 1

Introduction

Large Language models (LLM) demonstrate strong ability to generate human-like text on Natural Language Generation (NLG) tasks. The decoding algorithm used to generate texts from LLMs is critical to their performance. Single pass decoding algorithms are commonly used, yet they suffer from known issues. For instance, Beam Search, a popular single pass decoding algorithm, can generate repetitive, bland and less diverse outputs [28], and favours shorter sentences [11]. Other sampling based approaches, like temperature sampling, may produce more diverse outputs of lower quality.

Minimum Bayes risk (MBR) decoding is a powerful method first introduced for speech recognition [50, 23], but has gained increasing popularity in machine translation [20, 36, 15]. It is a two pass decoding algorithm that chooses the output based on the lowest risk among a list of samples. MBR's gains are not restricted to machine translation, as it has been proven to work on a wide variety of NLG tasks [52, 32, 33]. Despite its superiority to standard decoding methods, MBR remains prohibitively expensive to run at inference time with a time complexity of $O(n^2)$. Although recent advances have made the risk calculation of MBR more efficient [31, 9, 59], it remains impractical to run at inference time, especially that sampling multiple hypotheses from LLMs can be time consuming.

Yang et al. [62] has shown that Direct Preference Optimization (DPO), a recent preference optimization algorithm, can be used to improve the models' single-pass decoding performance by curating preference pairs from MBR rankings. The recent preference optimization literature has proposed various techniques that draw inspiration from DPO to build algorithms that fix some of DPO's shortcomings. For example, Ethayarajh et al. [16] identified some weaknesses inherent to DPO like loss aversion. They design Kahneman-Tversky Optimization (KTO) to address the identified limitations of DPO.

In this project, we aim to extend Yang et al. [62] method, which was only experimented on Machine Translation, to different NLG tasks, specifically, Summarization on CNN/DM

[26, 39], Question Answering on StrategyQA [48] and Question Generation on SquadV2 [43]. Additionally, we investigate whether KTO, an alternative preference optimization algorithm to DPO, can be applied in this pipeline.

Firstly we replicate the literature's baseline results for MBR [33, 32, 52]. After validating the efficacy of MBR, we conduct an investigation on the effect that key decoding hyper-parameters have on the output. We showcase the behaviour of MBR ranking compared to the Oracle ranking, a ranking based on the reference hypothesis.

In the second part we validate the efficacy of the proposed method on our selected NLG tasks. We investigate both DPO and KTO's performance with different dataset generation methods, and multiple degrees of regularization. We show that both algorithms improve the model's single pass decoding performance on all three tasks. We notice that KTO underperforms compared to DPO. Finally, we explain these conflicting results through the nature of the MBR data. We identify several properties of the MBR data that makes it unsuitable for KTO.

We summarize our main contributions like so:

1. We validate the enhanced performance of MBR compared to single pass decoding on the three selected tasks, and analyse MBR's behaviour.
2. We confirm that DPO and KTO with MBR works on the three tasks.
3. We investigate the efficacy of MBR+preference optimization with different dataset generation methods and different degrees of regularization for both alignment techniques.
4. We provide an analysis of KTO and DPO's behaviour as training progresses.
5. We provide an explanation based on MBR data properties, why DPO works better than KTO.

Chapter 2

Background and Literature Review

In this chapter, we present the relevant concepts used in this project and discuss related literature pertinent to our research. Firstly, we delve into sampling methods in Section 2.1, where we introduce deterministic and stochastic sampling strategies along with discussions about their limitations. We end the section with Minimum Bayes Risk Decoding as a decoding method. Section 2.2 presents three different methods for preference optimization from human feedback. Finally, Section 2.3 explains how MBR can be used with preference optimization techniques.

2.1 Decoding Approaches

Once training of the Large Language Model (LLM) is done, we need a way to decode the tokens and extract the output for a given input. Most decoding methods can be classified into deterministic and stochastic methods. While both methods decode a token at a certain time step conditioned on all previous tokens, deterministic approaches pick the token according to some rule and stochastic approaches sample a token from the probability distribution. Notably, different decoding methods impact the quality of the generation in significant ways. For example, deterministic methods produce less diverse outputs, while stochastic methods might produce less optimal ones. Despite being standard, these methods suffer from additional problems such as degeneration [28], where the output text can be incoherent, dull and repetitive. Eikema and Aziz [15] showed that Minimum Bayes risk (MBR) decoding with an adequate utility function can be a good alternative. In this section we will introduce deterministic and stochastic decoding methods, along with MBR decoding as an alternative.

2.1.1 Deterministic Decoding Methods

Maximum-a-posteriori (MAP) decoding shown in Equation 2.1 selects the sequence with highest probability under the model. In practice, this y^{MAP} can be intractable to compute [51, 24] as it requires passing over all the possible sequences. Consequently, multiple decoding approaches like Beam Search [24] and Greedy decoding have been proposed to approximate y^{MAP}.

$$\underset{y \in \Sigma^*}{argmax}\, P_T(\mathbf{y}|\mathbf{x}, \theta) = \underset{y \in \Sigma^*}{argmax} \prod_i P_T(y_i|y_{<i}, \mathbf{x}, \theta) \tag{2.1}$$

Greedy decoding is a simple and fast approximation to MAP. Given a partial hypothesis it picks the word that has the highest probability until it hits the end of sentence token. In other words, Greedy decoding, shown in Equation 2.2 picks the local optimal choice at each time step.

$$y_t = \underset{y \in \Sigma}{argmax} \prod_i P_T(y_i|y_{<t}, \mathbf{x}, \theta) \tag{2.2}$$

While greedy decoding is computationally efficient, it doesn't guarantee the global optimal solution. Moreover, Greedy decoding doesn't consider the whole sentence probability which can lead to ambiguous outputs compared to MAP.

Beam Search is another popular algorithm that applies a breadth first approach to decoding. Given a beam size N, beam search maintains N partial hypotheses at time step t $y^k_{<t}, k = 1, ..., N$. Then for each partial hypothesis, that has not ended with the end of sentence token, it picks the N most likely words in a greedy way. Out of the possible N^2 resulting outputs, it retains only the N most probable sequences. By the end of time step t, the selected N hypotheses are $y^k_{<t+1}, k = 1, ..., N$. When all the sequences have ended with the stop token, Beam Search selects one hypothesis with the highest probability. Since it selects at each time step the next word with the highest probability, a beam size of 1 will be equivalent to greedy decoding. In comparison to Greedy decoding, Beam search generates more diverse sentences, has a higher awareness of the global context and leads to more optimal results. Additionally, Beam search suffers from a length bias [11] as it performs better for sequences with a smaller target length, and bias towards frequent words as it over represents them in the output [40]. Although it produces less optimal outputs than MAP, Beam Search strikes a convenient balance between performance and computational demand.

2.1.2 Stochastic Decoding Strategies

While deterministic decoding methods are very popular and work well, they can suffer from a lack of diversity. On the other hand, stochastic methods generate more diverse outputs

due to the randomness introduced by sampling. However, sampling tokens from probability distributions can lead to sub-optimal or low probability tokens in the final sequence. As all the tokens in the vocabulary are allocated non zero-probability, this leads to unwanted tokens receiving high cumulative probability mass. To resolve this issue, various sampling algorithms that change the distribution have been introduced.

Top-K sampling [17] is a simple fix that excludes low probability tokens by picking out of the top K tokens, where K is a hyper parameter, with the highest score. The probability of each selected token is normalized with respect to the sum of the original probabilities. The tokens are then sampled from this new distribution.

Top-p sampling [28] is very similar to Top-K. As it also prunes away tokens with low probability, but the selection process is different. As seen in Equation 2.3 Top-p selects the minimum number of tokens whose probabilities sum to or greater then the hyper-parameter p. Afterwards, we sample from the normalized probability distribution as in Top-K

$$i = argmin_{i'} \sum_{j=1}^{i'} P_\theta(x^{(j)}|x_{<j}) \geq p \tag{2.3}$$

Although both Top-p and Top-k sampling aim to cut out the tail of the distribution, they can fail at only sampling high probability tokens. For example, the number of tokens that are pruned away by Top-k does not consider the confidence of the model. If the distribution is more or less uniform for a number higher than K, potential good candidates may be truncated. Additionally, if most of the probability mass falls on a number smaller than K, then the model will be exposed to additional very low probability tokens, that should have been pruned away. The same analysis holds for Top-p. If most words that constitute the selected set have similar probabilities, the next most likely word with slightly lower probability will be excluded from the set. Moreover, if most of the tokens have low probability then it is possible to have hundreds of tokens populating the set, some with very low probability. Both behaviours are undesirable and negatively affect decoding.

To solve these problems Hewitt et al. [27] introduced ε sampling a method that prunes away the tokens whose probability is smaller than ε. Where $0 \leq \varepsilon \leq 1$. Therefore, epsilon sampling samples tokens with probability higher than ε

Unlike previous sampling approaches that truncate the distribution and exclude tokens with low probability, Temperature sampling [1] shown in Equation 2.4 introduces a parameter τ that controls the peakiness of the distribution. For smaller τ the distribution is sharper, likely tokens will see their probability rise, while less likely tokens will have their probability reduced. This induces a more deterministic behaviour and less diverse behaviour. For $\tau = 0$ temperature sampling reduces to greedy decoding. For bigger τ the distribution is smoother,

probabilities are distributed more uniformly across the tokens.

$$\tilde{p}(y_t|y_{<t},\mathbf{x},\theta) = \frac{exp(z_\theta(y_t|y_{<t},\mathbf{x})/\tau)}{\sum_{y\in\mathscr{Y}} exp(z_\theta(y|y_{<t},\mathbf{x})/\tau)} \tag{2.4}$$

While Temperature sampling has been very popular for NLG tasks [25, 8, 18], Freitag et al. [19] has demonstrated that epsilon sampling [27] with MBR achieves better results than the aforementioned sampling techniques.

2.1.3 MBR

As stated previously, deterministic decoding strategies suffer from a lack of diversity, word frequency bias, and sequence length bias, while stochastic ones produce diverse outputs with a potentially lower quality. In contrast, Minimum Bayes risk decoding [36] has been shown to be a viable alternative. It provides substantial improvements on multiple metrics, for a wide range of NLG tasks [52]. Out of a set of hypotheses, MBR selects the output with the lowest expected risk, instead of the most probable. MBR can be framed as a variant of crowd sampling, as multiple hypothesis are compared and the least risky one is picked. The expected risk of a hypothesis is defined in Equation 2.5. y^{MBR} (Equation 2.6) is computed by choosing the hypothesis that has the minimum risk.

$$R(\mathbf{y}|\mathbf{x},\theta) = \mathbb{E}_{y'\sim p(.|\mathbf{x})}[L(\mathbf{y},\mathbf{y}^{'})] = \sum_{y\in C} L(\mathbf{y},\mathbf{y}^{'})p(\mathbf{y}^{'}|\mathbf{x}) \tag{2.5}$$

$$y^{MBR} = \underset{y\in H}{argmin}\, R(\mathbf{y}|\mathbf{x},\theta) \tag{2.6}$$

H and C are the hypothesis and reference sets. While Eikema and Aziz [15] argues that there is benefit in sampling different sentences for H and C, for practical reasons they are often kept the same. Furthermore, sampling all possible sequences to form H is impossible. Thus, MBR relies on a finite subset of samples. The usage of a finite set of samples prohibits us from using $p(\mathbf{y}^{'}|\mathbf{x},\theta)$. Therefore, we use the Monte Carlo approximation of the Risk in Equation 2.5:

$$R(\mathbf{y}|\mathbf{x},\theta) = \mathbb{E}_{y'\sim p(.|\mathbf{x})}[L(\mathbf{y},\mathbf{y}^{'})] = \frac{1}{|H|}\sum_{y\in H} L(\mathbf{y},\mathbf{y}^{'}) \tag{2.7}$$

As $|H| \to \infty$ the model's probability distribution is recovered. Moreover, we can replace $L(\mathbf{y},\mathbf{y}^{'})$ by $U(\mathbf{y},\mathbf{y}^{'})$ a utility function that measures the semantic similarity and closeness of

the hypothesis instead of the risk. y^{MBR} can therefore, be expressed in terms of Gain and similarity seen in Equation 2.8:

$$y^{MBR} = \underset{y \in H}{argmax}\, G(\mathbf{y}|\mathbf{x}, \theta) = \underset{y \in H}{argmax} \frac{1}{|H|} \sum_{y \in H} U(\mathbf{y}, \mathbf{y}^{'}) \tag{2.8}$$

The similarity metric is usually a neural metric like BLEURT [46] or BertScore [65]. Thus, MBR picks the sequence that is the most similar to all other in the hypothesis set with relatively high probability. MBR with a sufficiently large hypothesis set outperforms standard decoding practices, although it heavily depends on the $|H|$.

Since MBR chooses the sample that is the closest to all others, it will ignore the presence of outliers that would score higher on evaluation metrics and picks the safest choice even if it under performs. However, MBR tends to over-fit the utility function [19]. Consequently, a utility function that does not reflect human judgment will under perform on evaluation metrics. Additionally, the samples chosen by MBR will reflect the limitations, biases and strength of the utility function. For example, if the utility function does not suffer from word frequency bias and length bias, MBR will be less affected by them [38].

While MBR is a simple decoding method, it remains prohibitively expensive to run at inference time. With the wide adoption of LLM, extensive sampling to build an adequate hypothesis set is very time consuming. In addition, selecting y^{MBR} scales quadratically with $|H|$.

Current works aim to improve various aspects of the MBR algorithm. Jinnai and Ariu [31] re-frames MBR as a medoid identification problem, they use the correlated sequence halving algorithm [3] to find the medoid of the hypothesis set, which is the sequence that minimizes the total distance to all other points. This approach reduces the run time of MBR for a slight decrease in performance. Jinnai et al. [33] suggest replacing the Monte Carlo estimate in Equation 2.7 with the normalized probability of the samples in the hypothesis set $\hat{p}(\mathbf{y}^{'}|\mathbf{x})$. They found that their method provides slight improvements over the classical approach of MBR. Jinnai et al. [32] notices that the sequences that get the highest score under MBR tend to be very lexically similar. This might be a problem if we want the top-k samples to be lexically diverse, yet semantically similar. To fix this they introduce two methods, the first one clusters the samples in the hypothesis set then picks the medoid of each cluster. The second method introduces to the MBR objective in Equation 2.8 a diversity metric like pairwise-BLEU [47] or pairwise-sentence BERT [44].

In this work, we will employ the standard MBR approach defined above and strive to distill the benefits of MBR into single pass decoding methodologies through preference optimization techniques that we will introduce next.

2.2 Preference Optimization of LLMs

Although LLMs have acquired impressive abilities [7, 12, 30, 54], aligning them to generate desirable outputs remains a challenging task. During pre-training the models are exposed to a vast amount of user generated data ranging across multiple domains. It is inevitable that they will be exposed to erroneous or harmful content, that might appear in the output [41, 4]. While it is desirable for the LLM to be exposed to wrong answers, as it provides knowledge about potential human mistakes, we do not want the model to output such answers. Instead, we want to bias the model towards these correct generations. To solve this issue, techniques have been successfully developed to align the models to human preferences, ensuring their reliability, helpfulness, harmlessness and accuracy [53]. In this section, we present three preference learning algorithms: Reinforcement Learning from Human Feedback (RLHF), Direct Preference Optimization (DPO) and Kahneman-Tversky Optimization (KTO).

2.2.1 RLHF

RLHF [13] is a very successful alignment method, as it leverages human preference data to steer the LM to generate the desirable output. Ziegler et al. [66] defined three stages to RLHF pipeline that succeed the unsupervised pre-training.

The first step is Supervised Fine Tuning (SFT): This step fine-tunes the pre-trained LLM on desired tasks (for example summarization, question answering and question generation). Given a task-specific dataset of high quality answers the model is tuned to obtain π^{SFT}.

The second step is to fit a reward model $r_\phi(x,y)$ that generates a reward for the answer y given the prompt x. It can be broken down into the following steps:

1. Generate a set of output pairs given a prompt such as $(y_1,y_2) \sim \pi^{SFT}(y|x)$
2. Human annotators are asked to rate the pairs in term of preference, where $y_1 \succ y_2|x$ denotes that y_1 is the preferred sample and y_2 is the dispreferred one. We obtain a preference dataset $D = \{x^{(i)}, y_w^{(i)}, y_l^{(i)}\}_{i=1}^N$ where $y_w^{(i)}$ is the preferred sample and $y_l^{(i)}$ is the dispreferred.

If we assume that their exists a ground truth reward model $r^*(y,x)$ from which the preferences are obtained, we can model the human preferences according to the Bradley-Terry (BT) model [6] shown in Equation 2.9:

$$p^*(y_1 \succ y_2|x) = \frac{exp(r^*(x,y_1))}{exp(r^*(x,y_1)) + exp(r^*(x,y_2))} \tag{2.9}$$

Assuming that our preference dataset was sampled from p^* we can fit a reward model $r_\phi(x,y)$ through maximum likelihood. Equation 2.10 represents the negative log-likelihood loss:

$$\mathscr{L}_R(r_\phi, D) = -\mathbb{E}_{(x,y_w,y_l)\sim D}[log\sigma(r_\phi(x,y_w) - r_\phi(x,y_l))] \tag{2.10}$$

where σ is the logistic function. It is common practice to initialise the reward model from π^{SFT} with a single Linear Layer at the end that predicts the reward .

RLHF methods fit a reward model to a dataset of human preferences and then use RL to optimize a language model policy to produce responses assigned high reward without drifting excessively far from the original model.

The third and final stage is RL fine tuning, where the reward model is used to rate the answers generated by the LLM and steer it towards generating answers with high reward. The optimization problem can be seen in Equation 2.11.

$$max_{\pi_\theta}\mathbb{E}_{x\sim D, y\sim\pi_\theta(y|x)}[r_\phi(x,y)] - \beta\mathbb{D}_{KL}[\pi_\theta(y|x)||\pi_{ref}(y|x)] \tag{2.11}$$

Where $\pi_\theta(y|x)$ and $\pi_{ref}(y|x)$ have been initialised from π^{SFT}. Where β is the term that controls the strength of the KL term. It prevents the model from deviating too much from the base model. The Bigger the beta the closer the policy has to be to the reference. However, Equation 2.11 is not differentiable when working on language generation, thus, it is optimized through RL. In practice [66, 49], the reward function in Equation 2.12 is maximized using Proximal Policy Optimization (PPO) [45].

$$r(x,y) = r_\phi(x,y) - \beta(log\pi_\theta(y|x) - log\pi_{ref}(y|x)) \tag{2.12}$$

2.2.2 DPO

While RLHF works well at aligning LLM with human preferences, it remains prohibitively complex [42]. RLHF involves training a reward model and tuning the model by sampling from the LM during training and incentivizing it to produce answers that yield maximal reward. This leads to a notable computational cost compared to directly optimizing the LM. Moreover, PPO[45] suffers from hyper-parameter sensitivity, and it can be tricky to implement when scaling up to larger models [64]. To avoid explicitly fitting a reward model Rafailov et al. [42] introduce Direct Preference Optimization (DPO), an algorithm that trains the LM on human preferences without the need for RL or a reward model. Rafailov et al. [42] leverages a change of variables to shift the loss in Equation 2.10 from a loss over rewards to a loss over policies.

Staring from Equation 2.11 we obtain the solution, as optimal policy, to this KL-constrained problem in Equation 2.13.

$$\pi_r(y|x) = \frac{1}{Z(x)}\pi_{ref}(y|x)exp(\frac{1}{\beta}r(x,y)) \tag{2.13}$$

where $Z(x) = \sum_y \pi_{ref}(y|x)exp(\frac{1}{\beta}r(x,y))$ is the partition function. In practice $Z(x)$ is computationally expensive to approximate [35, 22]. By rearranging Equation 2.13 we can express $r(x,y)$ in terms of policies π_{ref} and π_r yielding Equation 2.14.

$$r(y,x) = \beta log(\frac{\pi_r(y|x)}{\pi_{ref}(y|x)}) + \beta logZ(x) \tag{2.14}$$

As RLHF, DPO assumes that the human preferences are modeled according to the Bradley-Terry model [6]. Therefore, when substituting the expression of the reward obtained in Equation 2.14 into the Bradley-Terry model in Equation 2.9 we get:

$$p^*(y_1 \succ y_2|x) = \frac{1}{1 + exp(\beta log(\frac{\pi^*(y_2|x)}{\pi_{ref}(y_2|x)}) - \beta log(\frac{\pi^*(y_1|x)}{\pi_{ref}(y_1|x)}))} \tag{2.15}$$

As we can see in Equation 2.15 the partition function cancels out.
Applying the same change of variable to Equation 2.10 we obtain the following DPO loss in Equation 2.16.

$$\mathscr{L}_{DPO}(\pi_\theta, \pi_{ref}) = -\mathbb{E}_{(x,y_w,y_l)\sim\mathscr{D}}[log\sigma(\beta log\frac{\pi_\theta(x,y_w)}{\pi_{ref}(x,y_w)} - \beta log\frac{\pi_\theta(x,y_l)}{\pi_{ref}(x,y_l)}))] \tag{2.16}$$

where π_θ is the policy being trained, and θ are the parameters of the LM.
By analyzing the DPO gradient in Equation 2.17 we can extract several insights on how DPO works.

$$\nabla_\theta\mathscr{L}_{DPO}(\pi_\theta;\pi_{ref}) = \\ -\beta\mathbb{E}_{(x,y_w,y_l)\sim D}[\sigma(\hat{r}_\theta(x,y_l) - \hat{r}_\theta(x,y_w))[\nabla_\theta log\pi(y_w|x) - \nabla_\theta log\pi(y_l|x)]] \tag{2.17}$$

where $\hat{r}_\theta(x,y) = \beta log\frac{\pi_\theta(x,y)}{\pi_{ref}(x,y)}$. The term $\nabla_\theta log\pi(y_w|x)$ increases the likelihood of y_w while the term $\nabla_\theta log\pi(y_l|x)$ decreases the likelihood of y_l. Additionally, the $\sigma(.)$ controls how big the gradient step should be. If $\hat{r}_\theta(x,y_w) < \hat{r}_\theta(x,y_l)$ then the implicit reward is incorrect leading to a bigger $\sigma(.)$ term and a bigger importance on increasing the likelihood of the

preferred term.

In contrast with RLHF where the reward model steers the LM into generating high reward completions, DPO is more straightforward as it increases the likelihood of the winning samples and decreases the likelihood of the losing samples. Moreover, through DPO's framing, "your language model becomes an implicit reward model" [42]. DPO's utilization of preference pairs is well-suited for our project. We will generate preference pairs for each prompt using MBR, then apply DPO to transfer the MBR performance onto the model.

2.2.3 KTO

Prospect theory introduced by Kahneman and Tversky [34] describes how humans judge the outcome of random events. Tversky and Kahneman [58] found that when faced with decisions about random variables, humans tend to be biased in an explicit way. In other words, humans are loss averse and they tend to prioritize events that lead to less loss rather than more gains. Ethayarajh et al. [16] showed that DPO implicitly incorporates such biases, and introduce a loss function derived from prospect theory, that fixes the loss aversion problem. Prospect theory introduces the value function Equation 2.18 that provides the subjective value (in the LLM setup it can be thought of the reward) that humans assign to an event compared to a reference point.

$$v(z, z_{ref}, \lambda, \alpha) = \begin{cases} (z - z_{ref})^{\alpha} & \text{if } z > z_{ref} \\ -\lambda(z_{ref} - z)^{\alpha} & \text{if } z < z_{ref} \end{cases} \tag{2.18}$$

Where $z > z_{ref}$ and $z < z_{ref}$ correspond to positive and negative event. α controls the speed at which the value changes when z moves away from z_{ref}, and λ controls how loss averse the judgment is. Empirically, the median values are 0.88 and 2.25 respectively. We can see from Equation 2.18 that human judgment is more loss averse due to $\lambda > 1$.

Based on Equation 2.18 Ethayarajh et al. [16] introduces the value function in Equation 2.21 that differs in significant ways from the original value function. Firstly, α is replaced by the sigmoid function, as the exponent is hard to optimize. While DPO only considers one dispreferred example, the new value function considers the relative value of one example compared to all others. KTO assumes that humans rate the quality of one sample based on all the data they have already seen. Finally, we can see from the expression of the loss in Equation 2.23 both dispreferred and preferred samples are scaled by a coefficient. This provides the users with greater control to loss aversion and gain sensitivity. For example, if

$\lambda_U = 1$ and $\lambda_D = 1.5$ KTO will be more gain sensitive .

$$r_{TKO}(x,y) = \beta log \frac{\pi_\theta(y|x)}{\pi_{ref}(y|x)} \tag{2.19}$$

$$z_{ref} = \mathbb{E}_{x' \sim D}[\beta \mathbf{KL}(\pi_\theta(y'|x')||\pi_{ref}(y'|x'))] \tag{2.20}$$

$$v_{KTO}(x,y;\beta) = \begin{cases} \sigma(r_{KTO}(x,y) - z_{ref}) & \text{if } y \sim y_{desirable}|x \\ \sigma(z_{ref} - r_{KTO}(x,y)) & \text{if } y \sim y_{undesirable}|x \end{cases} \tag{2.21}$$

$$w(y) = \begin{cases} \lambda_D & \text{if } y \sim y_{desirable}|x \\ \lambda_U & \text{if } y \sim y_{undesirable}|x \end{cases} \tag{2.22}$$

$$\mathscr{L}_{KTO}(\pi_{theta}, \pi_{ref}) = \mathbb{E}_{x,y \sim D}[w(y)(1 - v_{KTO}(x,y;\beta))] \tag{2.23}$$

Moreover, KTO does not need preference pairs, it only needs to know which samples are accepted or rejected for a certain prompt. Therefore, if a dataset is imbalanced in one of the classes, KTO can leverage the whole dataset by changing the weights in equation 2.22 to reflect the class proportions.

Finally, while KTO and DPO optimize the LLM directly compared to RLHF, KTO's loss derived from prospect theory avoids the inherent biases introduced by humans judgment. KTO's ability to leverage unbalanced datasets is ideal in our case, as the MBR ranking provides us with one desirable sample y^{MBR} and multiple rejected samples.

2.3 MBR with RL

MBR outputs a ranking of the samples in the hypothesis set, this ranking could be used to build a preference set for DPO and KTO. Yang et al. [62] has shown the validity of this method on machine translation. He uses the MBR ranking to extract preference triplets and runs DPO. After DPO fine-tuning, he observes an improvement in the performance using Beam Search. His results suggest that MBR+DPO teaches the model to learn from MBR preferences, and the model is able to replicate the MBR behaviour through a single pass decoding method. Moreover, he observes a positive and increasing reward margin as training progresses. The reward margin is the difference between the reward for the winning hypothesis and the losing one:

$$\text{Margin} = \beta * (log(\frac{\pi_\theta(y_w|x)}{\pi_{ref}(y_w|x)}) - log(\frac{\pi_\theta(y_l|x)}{\pi_{ref}(y_l|x)})) \tag{2.24}$$

Increase in the reward margin indicates that, as training proceeds, the model is preferring the winning hypothesis and dispreferring the losing one.

Motivated by his results, we aim to establish the effectiveness of this method on different tasks and if different alignment methods benefit from a preference set extracted through MBR. In the next chapter, we will delve into the experimental settings of each alignment method.

2.4 Summary

In this chapter, we have presented the necessary background on decoding methods and preference optimization techniques and highlighted the contributions of our work. We explained various decoding methods and their limitations, and identified MBR decoding as an appropriate way to build a preference dataset in an unsupervised manner. Finally, we introduced popular optimization techniques, that will be used to align the LLM to the MBR preference data. In the next chapter we will discuss the methods used to generate the preference data and align the LLM.

Chapter 3

Methodology and Experiment Design

In this chapter, we introduce the methodology used to validate that MBR coupled with preference optimization works across our three selected tasks. We, first, introduce the dataset for each task. We detail how we split the available data into a test set for evaluation and a preference set to align the models. We, then, present the models used along with the appropriate prompt format for each task. Furthermore, we delve into the decoding strategies used to evaluate our models along with generating the MBR samples. Moreover, we give a detailed explanation of each metric used, with their strengths and weaknesses. Finally, we present our approach for building the different preference sets used in aligning the models, and the parameter setups for all the experiments.

3.1 Datasets

As stated earlier we will extract preference data using MBR on three different tasks. The tasks have been carefully picked, as MBR has shown to improve their performance on evaluation metrics. For all selected datasets, we strive to only use the test data when enough samples are available. Doing so, prevents possible overlaps with the training data. Since the open source models could have been trained on these datasets. The preference data set that we extract will range in the low thousands, as Yang et al. [62] showed that DPO+MBR works well using 5000 samples.

Summarization: We use the openly available CNN, Daily Mail dataset [26, 39]. It is a dataset for abstractive text summarization, where the target output is a one sentence summary of the articles. The test set contains 11500 samples, we use the first 1000 samples as held out data, to reproduce the MBR results in [33] and compare the improvements obtained from preference optimization. We use the next 4000 samples to build the preference dataset.

Question answering: We use the StartegyQA dataset from BigBench [48]. It is a dataset

that contains questions with their one sentence answer. This a task that aims to test the reasoning capabilities of the models. The model is required to provide an appropriate and concise answer in natural language. For comparison with previous works, we use the curated dataset in [52] that consists of 2289 samples. The dataset is split into test and train with 457 and 1832 respective samples. We use the test set for evaluation. As the dataset is relatively small we will use the train set to build preference pairs.

Question Generation: Given a paragraph, we aim to generate a question that is relevant to its content. We use the SquadV2 dataset [43], it consists of paragraphs with corresponding questions. During evaluation, we generate the question and compare it with the set of available questions, and pick the highest possible score. SquadV2's test data consists of 1204 samples, we use the first 1000, as in [32], for evaluation and use 4000 samples from the training set to build the preference dataset.

Due to the lack of samples in the test set for StartegyQA and SquadV2, we had to use the training set to construct preference pairs. A better way, would have used the evaluation set to build preference pairs. However both datasets do not have evaluation sets. Although it is not optimal, as the model could have been trained on them. It is the only way to proceed for these two tasks, to remain consistent with the available literature enabling us to compare our results to existing baselines.

3.2 Models

For consistency with the existing literature [32, 33], we will use Mistral-7B-Instruct-v0.1 [30] for text summarization and Zephyr-7b-beta [57] for question generation. Suzgun et al. [52] used text-davinci-002 a model from openAI to generate the answers for the question generation task. This model is not publicly available, therefore we decided to proceed with Mistral-7B-Instruct-v0.1.

Mistral-7B-Instruct-v0.1 [30]: is an instruction tuned version of Mistral-7B-v0.1. Jiang et al. [30] use openly available high quality instruction datasets on hugging face, where each instruction prompt is matched with a carefully written answer. Mistral-7B-Instruct-v0.1 is carefully engineered to provide fast inference without loss in performance, outperforming bigger LLMS like LLaMa 34B [54] or LLama2 13B [55] on a wide range of tasks. It has 32 transformer layers [60] with hidden size 14336, 32 attention heads per layer with size of 128. Instead of vanilla attention, they leverage sliding window attention [10] and Group Query attention[2] for faster inference and improved attention for longer sequences.

Zephyr-7b-beta [57] is both instruction tuned and aligned to human preferences, it uses Mistral-7B-v0.1 [30] as a base model. Tunstall et al. [57] first instruction tune Mistral-7B-

v0.1 using distilled SFT. Distilled SFT [61] consists of using a more powerful teacher model to generate answers for a set of prompts and run SFT on the generated dataset. Secondly, they use distilled DPO to align the model to human preferences. Distilled DPO is classical DPO introduced in Section 2.2, with preference pairs generated form teacher models. They query four LLMs and rank the answers using GPT4. The most preferred answer by GPT4 will be the winning sample and any of the other three will be the losing one.

3.3 Prompt format

As the models we are using are instruction tuned, we need to carefully design the prompt format for each task.
Mistral-7B-Instruct-v0.1 prompt format differs from the classical chat template, in that it uses a specific format where the prompt should be enclosed with the following tags :
<s>[INST] PROMPT [/INST].
For question answering, the question it self is used as the prompt in the following fashion:
<s>[INST] Is it common to see frost during some college commencements? [/INST].
For abstractive summarization we use the same prompt as in [33]:
<s>[INST] Given the following article, write a short summary in one sentence. Article: [[QUESTION]][/INST], Where [[QUESTION]] is replaced by the article.
Since we are using Zephyr-7b-beta for question generation, we have to use a different prompt format. Zephyr-7b-beta follows this chat template:
<|system|>
System Prompt
<|user|>
User Prompt
<|assistant|>
Where the System Prompt is the general information provided to the LLM about the task that it needs to perform. We will use the same system prompt as [32]:
"Given a paragraph provided by the user, generate a very short question one can answer by a word to test the understanding of the paragraph. Make sure that the question is very short. Do NOT include the answer".
The User Prompt will be the paragraph.

3.4 Decoding Strategies

Previous works [33, 32, 52] have shown the superior performance of MBR compared to standard decoding methodologies. Thus, the model's output quality is significantly impacted by the choice of decoding algorithm we use. In this project, we will contrast the performance of various decoding strategies on the non-aligned models. We will aim to reproduce the benchmarks set in the literature. Finally, we will use Beam Search on the models after alignment, to validate that the MBR gains have been passed down to the model.

We will first investigate Beam Search performance with varying beam size from 1 to 5. Theoretically, Beam Search benefits from larger beam sizes. As more sequences are generated, we can find more optimal sequences. However, it has been observed that higher beam sizes lead to lower performances, a phenomenon known as the beam search curse [63]. This happens because more shorter answers will be generated, since it is easier to hit end of sentence token. Since beam search has length bias, it will assign higher probability to smaller sequences which leads to a drop in quality. We follow previous works [21, 56], that found that quality degrades for a beam size bigger than 5.

The performance of MBR introduced in Section 2.1, relies on the following three design choices.

1. Sampling the hypothesis set: a higher quality hypothesis set will lead to better performance. Following the current literature, we will use epsilon sampling, that has proven to sample higher quality sentences, and does not suffer from the limitations of previous sampling techniques. We set $\varepsilon = 0.01$ for Zephyr-7b-beta and $\varepsilon = 0.02$ for Mistral-7B-Instruct-v0.1.

2. Size of the hypothesis set: A larger |H| can potentially lead to better results. We investigate the performance of MBR with respect to |H|. For computational reasons, we restrict |H| to the following three values 8,16,32.

3. Choice of the utility function: Since MBR selects the hypothesis that is the most similar to all others, it is imperative that $U(x,x^{'})$ handles semantic similarities appropriately. Moreover, since the MBR decoding process optimises the metric used, we should choose a metric that doesn't suffer from obvious limitations like word frequency or length bias. Finally, for optimal results, we should choose a utility function that correlates well with human judgment. Taking all these criteria into consideration, a neural metric like BertScore lends it self as a natural choice. Other neural metrics like

BleuRT have also been popular, but Suzgun et al. [52] and Jinnai et al. [32] shows that BertScore leads to a higher overall score on our desired tasks.

3.5 Evaluation Metrics

In this project we will use the same metrics employed in the literature to evaluate the quality of the samples. These metrics are categorized into non-neural metrics based on n-gram matching, and neural metrics based on the similarity of the sentence embedding. Additionally, human and GPT4 evaluation are popular, but due to resource constraints we will not conduct such evaluations.

Rouge-n [37] was originally designed to measure the quality of machine summarised text to human references. It measures the number of similar n_grams between the reference and the hypothesis. Equation 3.1 depicts how rouge_n is computed. Where S_x^n and $S_{\hat{x}}^n$ are the set of n-grams in the reference and hypothesis set.

$$R_n = \frac{\sum_{S_x^n \in References} \sum_{w \in S_x^n} \mathbb{I}[w \in S_{\hat{x}}^n]}{\sum_{S_x^n \in References} |S_x^n|} \tag{3.1}$$

Rouge_L is a variant of Rouge_n that works on the longest common sub-sequence rather than the common n-grams. Although Rouge is a popular method, it suffers from obvious limitations. Since we only have one reference, rouge's ability to leverage multiple references is not utilized. This can lead to under estimating the model performance as multiple answers can be true. Secondly, Rouge doesn't consider semantic similarities. Synonyms or paraphrases of some terms will not be considered as it operates on exact n-gram matching. This leads to a lower rouge score, while the answer can be semantically equivalent to the target. Rouge will assign a higher score to longer hypothesis, as they will have more n-grams that matches with the reference. In our project, we will use Rouge_L for the summarization and question answering tasks, and Rouge_1 for question answering

METEOR [5], who was originally developed for machine translation, employs a flexible uni-gram matching. In addition to matching exact uni-grams, it allows stemming, paraphrasing and synonym matching. It uses external synonym dictionaries to match words with similar meaning. It relies on a stemmer to match words with the same root word. And it employs a paraphrase table to identify potential matches. While METEOR addresses some of Rouge's issues, it remains far from perfect. METEOR is reliant on the external resources it uses, any limitations they introduce will directly mirror into the score. Moreover, METEOR has limited semantic consideration, and contextual understanding. A word or its synonym's meaning depends on its placement in the sentence and neighbouring words. Thus,

METEOR might match words even though they have different meanings. In this work, we use METEOR for all three tasks.

Rouge and METEOR's main limitation is that they do not properly consider semantic equivalence. Instead, they rely on shallow similarities between hypothesis and the reference set through n-gram matching. This leads to sentences/expressions that are equivalent in meaning but different syntactically to be underestimated. In contrast, BertScore [65] solves these problems, by leveraging contextual word embedding. Contextual word embedding, correctly handles dependencies between words, semantic equivalence, paraphrases and structural differences. Every word's representation is impacted by the neighbouring words. This leads to the same word having a different representation based on its overall function in the sentence. BertScore uses the token embedding computed by Bert[14] and calculates the cosine similarity between each word in both sentences. Therefore, BertScore measures the similarity based on the meaning rather than the words matching. We will use BertScore for all three tasks. While Bertscore is a better measure of performance, we use it as utility metric for MBR. Since MBR tends to over-fit the utility function, and the preference sets are extracted through MBR. Therefore, we need other unbiased evaluation metrics to help us validate our experiments. Although, Rouge and Meteor suffer from known limitations, they remain in standard use throughout the literature along with BertScore.

3.6 Preference Optimization of LLMS

While MBR is a powerful technique, it is a two pass decoding method that remains prohibitively expensive to run at inference time. Since MBR outputs a ranking of the hypothesis based on utility, we can leverage this ranking to build a preference dataset. For each task, we sample 32 sequences for each prompt, and rank them through MBR. Afterwards, we build preference pairs for DPO alignment, where the preferred sample has a higher utility than the dispreferred one. For KTO alignment, we build a set of rejected and accepted samples for each prompt. We will align the model using this preference dataset, then evaluate it on the test set using Beam Search. After the model is aligned, our aim is to observe improvements on the evaluation metrics for sequences generated using Beam Search. If improvements are observed, on the post-aligned models from the pre-aligned models, we can successfully claim that MBR gains have been distilled onto the post-aligned models. In this section we will outline the preference set generation methods for both KTO and DPO, along with hyper-parameter set up of our experiments.

3.6.1 DPO Dataset Generation

DPO's preference set is formed using triplets (x, y_w, y_l), where x is the user prompt, y_w is the wining hypothesis, and y_l is the losing one. In MBR terms, the winning hypothesis corresponds the the one with the higher utility, while the losing one has the lower utility. Sampling the hypothesis set H follows the same setup as in Section 3.4. Once we acquired the ranked hypothesis set, we need to build a dataset of triplets. Analogous to [62], we adopt three different approaches for triplet selection.

1. Best-Worst (BW): as its name indicates we select only one pair for each prompt. y_w has the highest utility among the 32 samples and y_l has the lowest utility:

$$y_w = \underset{y \in H}{argmax}\ G(\mathbf{y}|\mathbf{x}, \theta), y_l = \underset{y \in H}{argmin}\ G(\mathbf{y}|\mathbf{x}, \theta) \tag{3.2}$$

   This is the simplest setup, as it leads to smallest dataset. As a consequence of this setup, the pairs have the highest difference in utility among the samples. DPO might benefit from a more nuanced approach where the difference between the utility is smaller. This might lead the model to understand on a more granular fashion what is desirable in the sentence and what is not.

2. Best-Middle-Worst (BMW): In this setup for every prompt we have two triplets (x, y_w, y_m) and (x, y_m, y_l). Where y_l and y_w are the same as in BW setup, but y_m represents the sample that falls in the middle of the MBR ranked list $m = \lceil (|H|/2) \rceil$. This setup provides a way for the model to understand what exactly makes y_w the preferred sample. If we look back at Equation 2.17 of the DPO gradient, the likelihood of y_m will be increased when it is the winning hypothesis and it will be decreased when it is the losing hypothesis. The DPO algorithm will eventually settle in a region where:

$$\hat{r}_\theta(x, y_w) > \hat{r}_\theta(x, y_m) > \hat{r}_\theta(x, y_l) \tag{3.3}$$

   In this sens the algorithm will increase the likelihood of y_m over y_l but not over y_w. The model will, thus, implicitly learn the MBR ranking order. This provides the model with greater knowledge on what makes each hypothesis better than the other.

3. Consecutive Pairs with strides (CPS): This method selects the pairs where y_w and y_l are separated by a stride. We choose a stride of 2. Therefore, we get 16 preference pairs for each prompt.

$$(x, y_1, y_3), (x, y_3, y_5), ..., (x, y_{30}, y_{32}) \tag{3.4}$$

A possible variation would be to use a stride of 1, but consecutive samples in the MBR ranking are too similar. As a result, the model might struggle to learn the nuances between the pairs.

3.6.2 KTO Dataset Generation

KTO does not use preference pairs rather a signal if the corresponding output is desirable or not. Moreover, KTO introduces a weight for each of the desirable and undesirable classes. This allows the user to control how biased the algorithm should be to a specific class. Which grants us the ability to handle unbalanced datasets. In the MBR setting, the sample with the highest utility is the accepted one and all the others are rejected. Due to this property, KTO can leverage the MBR ranking to build preference sets that are unbalanced in the rejected class. For consistency and comparison with DPO, we use the same MBR samples to build the different preference sets for KTO. We adopt 5 different approaches to build the preference sets.

1. 1:1 ratio: This setup is similar to the BW setup in DPO, as each prompt has two corresponding sequences one rejected and one accepted. The accepted one is the sample with the highest utility and the rejected one has the lowest utility.

2. Ethayarajh et al. [16] postulates that KTO with a 1:1 ratio of accepted and rejected samples coming from preference pairs, secretly benefits the algorithm. To test it's robustness and performance on unbalanced datasets they gradually discard 10% of the accepted samples until they have discarded 90%. They notice that for all unbalanced datasets, KTO outperforms DPO. Since we only have one accepted sample per prompt, and an abundance of rejected samples we test KTO's robustness by adding rejected samples rather than discarding accepted ones. Starting from the bottom of the ranked list, we add n rejected examples with a stride of 2. In other words if we want a ratio of 1:2 we add y_{32} and y_{30} to the rejected list and y_1 to the accepted list. We will report results for the following ratios: 1:2, 1:3, 1:4. Experimentally, we found that higher ratios lead to excessive model hallucinations and a sharp drop in model performance.

3. 3:3 ratio: In order to test if KTO benefits from additional lower quality desirable data, we add 2 more examples to the desired set. Starting from the top we select n samples with a stride of 2. For the rejected samples we follow the same approach but starting

from the bottom. The final dataset can be shown below:

$$x => \begin{cases} y_{desirable} = [y_1, y_3, y_5] \\ y_{undesirable} = [y_{28}, y_{30}, y_{32}] \end{cases} \tag{3.5}$$

3.6.3 Experiment Settings

For each task we build the preference datasets, and use 10% for validation and 90% for training. For DPO trained models, we will explore the performance with respect to different β values. As stated earlier β controls how far the model is allowed to drift from the reference model. We will experiment and report the results for $\beta = 0.01, 0.1, 0.5, 1.0$ on all three preference datasets for all three tasks. On the other hand, we start by experimenting with KTO on the different preference datasets for $\beta = 0.1$. Afterwards, we select the preference set that has led to the highest performance, and align the models with different β values. As in DPO, we will end up comparing KTO's performance with $\beta = 0.01, 0.1, 0.5, 1.0$. Additionally, we need to set the class weights for KTO. We follow the same procedure as in [16]:

$$\frac{\lambda_D n_D}{\lambda_U n_U} \in [1, \frac{4}{3}] \tag{3.6}$$

Where at least one class weight in Equation 3.6 should be set to one. In our case, since we want the model to be more sensitive to desirable samples we always set $\lambda_U = 1$ and $\lambda_D = \frac{n_U * 4}{n_D * 3}$.

$$1:1 \text{ and } 3:3 => \begin{cases} \lambda_D = \frac{4}{3} \\ \lambda_U = 1 \end{cases} \tag{3.7}$$

$$1:2 => \begin{cases} \lambda_D = \frac{8}{3} \\ \lambda_U = 1 \end{cases} \tag{3.8}$$

$$1:3 => \begin{cases} \lambda_D = \frac{12}{3} \\ \lambda_U = 1 \end{cases} \tag{3.9}$$

$$1:4 => \begin{cases} \lambda_D = \frac{16}{3} \\ \lambda_U = 1 \end{cases} \tag{3.10}$$

We use the HuggingFace trl library[1] to run all our experiments. For both preference algorithm, we use RMSprop with learning rate 0.000001. We apply a linear-warmup phase for

[1] trl version 0.9.6 https://pypi.org/project/trl/

the first 10% of the training steps. All the experiments on the three selected tasks are trained on 2 A100 GPUs for one epoch. For the summarization task we use a per_device_batch_size of 1, and a per_device_batch_size of 2 for question generation and question answering. We set π_{ref} to the reference model, without an SFT phase, as we noticed that for all three tasks the post-SFT model performs worse then the pre-SFT model. Furthermore, we investigated the use of Low-Rank Adaptation method (LoRA) [29] along with preference optimization. LoRA is a method that trains a subset of the model's parameters. After training with LoRA, we did not notice any improvements on the evaluation metrics. Consequently, all the experiments reported have had all the parameters of the models updated.

3.7 Summary

The methods adopted in this project, first aim to investigate the performance of decoding strategies on various tasks. They aim to validate that multiple tasks benefit from MBR decoding, and that MBR's performance is dependent on the size of the hypothesis set. The second part, aims to leverage the MBR samples to build multiple preference sets, and to align the models using KTO and DPO on the the preference sets, with appropriate exploration of hyperparameters for each algorithm. We aim to establish if the aligned models are able to accurately learn MBR preferences, and improve the quality of the output using efficient single pass decoding methods.

Chapter 4

Results and Discussion

In this chapter, we will showcase our results. We firstly compare Beam Search and MBR, and explore how their respective parameters affect the decoding performance. We explain the obtained results on the three tasks, demonstrating MBR's superior performance. Additionally we provide some insight on the behaviour of MBR. We then move to align the models through DPO on the simplest preference split. After validating the efficacy of the fine-tuning technique, we move to analyse the performance on different preference splits. We also provide a qualitative analysis based on training statistics why one specific split works best. Additionally, we align the models using KTO on different splits and we compare its performance relative to DPO. Finally, we propose reasons why one algorithm outperforms the other on MBR preference data.

4.1 Analysis of Decoding Methods

In this section we investigate the performance of decoding strategies for the following three tasks:

1. Question answering on StrategyQA's test set, using Mistral-7B-Instruct-v0.1 as our model
2. Summarization on the first 1000 samples from CNN/DM's test set, using Mistral-7B-Instruct-v0.1 as our model
3. Question generation on the first 1000 samples from SquadV2's test set, using Zephyr-7b-beta as our model.

4.1.1 Establishing Baselines

We first aim to reproduce the MBR baselines reported in the literature for each dataset. We can observe from Table 4.1 that MBR successfully improves on the performance of Beam Search. For the CNN/DM dataset our results are almost equal to the Jinnai et al. [33]'s baseline. For SquadV2, Jinnai et al. [32] achieves better performance, as they chose the hypothesis set size |H| to be 128. For practical reasons we cannot reproduce that. For StrategyQA, Suzgun et al. [52] used davinci-002, as we do not have access to this model we could not reproduce the baseline results. Although we failed to meet the baseline, with openly available models, for SquadV2 and StrategyQA, our MBR results show the same trend across all three tasks. MBR improves the decoding performance compared to Beam Search on all three selected tasks. In the next section we will delve into the impact of the Beam Size and |H| on decoding performance.

Decoding Method	CNN/DM	StrategyQA		SquadV2
	RougeL↑	RougeL↑	Rouge1↑	Meteor↑
Beam Search	14.0	18.1	25.3	38.4
MBR |H|=32	17.5	21.7	28.8	40.5
BaseLine MBR	17.6	32.1	39.1	41.1

Table 4.1 MBR and Beam Search compared to the MBR literature baseline. We used BertScore as MBR utility function. All the following results will be reported with BertScore as utility.

4.1.2 Evaluation of Decoding Mechanisms

The metrics reported for each task in Table 4.1 are those used in the literature. In this section we add BertScore and METEOR for each task. As we discussed earlier, these two metrics provide additional semantic understanding compared to RougeL and Rouge1. Therefore, they will help us get a more holistic measure of the performance.

Summarization on CNN/DM:

Beam Search: As we observe from Table 4.2, as the Beam Size gets bigger the RougeL values decrease. Due to the exact string matching mechanism employed by RougeL, this is to be expected, as bigger beam sizes explore a more diverse set of candidates. We can conclude that the greedy search solution has more exact words as the optimal solution. Looking at METEOR and BertScore, we notice that METEOR's values fluctuate around 26,

while BertScore remains the same. These results show that while a bigger Beam size leads to more diversity, the samples remain semantically very similar. Therefore the greedy search solution (beam size = 1) is semantically equivalent and lexically superior to Beam Search.

MBR: On the other hand, MBR shows a different trend. As the hypothesis set size increases, the performance on all the evaluation metrics increase. Even for our smallest hypothesis set of |H| = 8, we achieve better performance than Beam Search. Since Both non-neural and neural metrics improved, y^{MBR} is both more lexically and semantically similar to the ground truth than the Beam Search output.

Systems	**RougeL** ↑	**METEOR** ↑	**BertScore** ↑
Beam Search			
Beam Size = 1	16	**26.6**	85.6
Beam Size = 2	15.2	26.5	85.7
Beam Size = 3	14.6	25.9	85.7
Beam Size = 4	14.2	26.0	85.6
Beam Size = 5	14.0	25.6	85.6
MBR			
|H| = 8	17.1	26.0	86.0
|H| = 16	17.3	26.3	86.2
|H| = 32	**17.5**	26.4	**86.3**

Table 4.2 Summarization's decoding performance of MBR with different |H| size, and Beam search with different Beam sizes on CNN/DM's test set.

Question Answering on StrategyQA:

Beam Search: As we can see from Table 4.3 the performance of beam search on METEOR steadily increases as the beam size increases. The increase witnessed in METEOR, declines for beam sizes beyond 2. On the other hand, RougeL and Rouge1 steadily decrease as we increase the beam size. This is attributed to the nature of these metrics, as they are based on exact string matching, and a larger beam size introduces more diversity in the generation. Therefore, they are incapable of accounting for this diversity. In contrast, METEOR increases as it includes mechanisms to handle lexical diversity and paraphrases. Bertscore's performance remains the same. While Beam search is able to generate more diverse and globally optimal solutions as the beam size increases, these solutions remain semantically similar under the BertScore metric. Thus, for this task the Greedy Search solution (beam size =1) is as good as beam search.

MBR: In contrast to Beam Search, as we increase the size of the hypothesis set we

notice improvements across all the metrics. Moreover, compared to Beam Search, we notice non-trivial improvements on all the metrics except for METEOR which decreased by 2 points. When METEOR and BertScore's values are in conflict, we rely on Bertscore as it has better semantic coverage. We conclusively claim that y^{MBR} is both more lexically and semantically similar to the ground truth than the Beam Search output.

Systems	**RougeL** ↑	**Rouge1** ↑	**METEOR** ↑	**BertScore** ↑
Beam Search				
Beam Size = 1	20.6	27.8	25.0	86.0
Beam Size = 2	20.4	28.2	26.3	86.0
Beam Size = 3	20.1	28.0	26.3	86.0
Beam Size = 4	20.0	28.0	26.5	86.0
Beam Size = 5	18.1	25.3	**27.0**	86.0
MBR				
|H| = 8	20.4	27.6	24.9	86.9
|H| = 16	21.0	28.2	25.2	87.1
|H| = 32	**21.7**	**28.8**	25.0	**87.3**

Table 4.3 Question answering's decoding performance of MBR with different |H| size, and Beam search with different Beam sizes on StrategyQA's test set.

Question generation on SquadV2:

Beam Search: We can see from Table 4.4 that the performance of beam search, measured by BertScore, steadily increases up to a beam size of 4, after which we notice a marginal drop of 0.1 for beam size of 5, while the METEOR values oscillates from slightly increasing to slightly decreasing. Although, more likely outputs do not correlate with a better METEOR score, they do with BertScore's values. Since METEOR has limited semantic coverage compared to BertScore, it might not properly handle the small lexical and structural discrepancies introduced by a larger beam size. In contrast, BertScore accurately handles these differences as semantically superior or equivalent. Thus, assigning a higher score.

MBR: In contrast to Beam Search, as we increase the size of the hypothesis set we notice improvements across all the metrics. A bigger hypothesis set correlates well with improvements on the metrics. y^{MBR} is more semantically similar to the ground truth than the Beam Search output.

Observing all these results we can ascertain that, independent of |H|, MBR successfully generates higher quality outputs compared to Beam search. And MBR performance improves as |H| gets bigger. Additionally, for the Summarization and Question answering task Beam

Systems	METEOR ↑	BertScore ↑
Beam Search		
Beam Width = 1	39.0	87.5
Beam Width = 2	39.5	87.7
Beam Width = 3	39.1	87.8
Beam Width = 4	39.4	88.0
Beam Width = 5	38.4	87.9
MBR		
|H| = 8	40.1	89.3
|H| = 16	40.3	89.5
|H| = 32	**40.5**	**89.6**

Table 4.4 Question generation's decoding performance of MBR with different |H| size, and Beam search with different Beam sizes on SquadV2's test set.

search is incapable of generating higher quality outputs. This is possibly due to the confidence of the model in its top candidates leading to little change in the answer.

4.1.3 Behaviour of MBR

We have observed the improvements introduced by MBR on all three tasks compared to Beam Search. Notably, the most significant improvements come from shifting from Beam Search to MBR with the smallest |H|. Although larger |H| lead to higher scores, the difference in performance between the biggest and the smallest |H| is smaller than the difference from Beam search to MBR. This suggests the presence of an upper bound on the performance of MBR, as simply scaling the hypothesis set will eventually lead to stagnation in the MBR score.

MBR chooses the safest hypothesis, rather than the one that maximises the evaluation metric compared to the ground truth. To analyse this behaviour, we present MBR Oracle. Instead of calculating the MBR score as in Equation 2.6, we now get the score for each sample as a measure of the utility function with the ground truth (reference hypothesis), and pick the one with highest score (Equation 4.1). MBR neglects outliers, this property is a double edged sword, as we are certain not to pick a sample that significantly under-performs we also are guaranteed to ignore samples that significantly out-perform the rest. The purpose of the oracle score is to showcase this property. As expected, we can see from Table 4.5 that for all three tasks the Oracle score is higher than both MBR and Beam search. The Oracle score validates the importance of the hypothesis set size, as bigger |H| leads to better outputs.

The MBR procedure generates higher quality outputs for bigger |H|.

$$y^{Best} = \underset{y \in H}{argmax}\, U(y, y^{ref}) \tag{4.1}$$

Systems	CNN/DM			StrategyQA				SquadV2	
	RougeL↑	Meteor↑	BertScore↑	RougeL↑	Rouge1↑	Meteor↑	BertScore↑	Meteor↑	BertScore↑
Beam Search	14.0	25.6	85.6	18.1	25.3	27.0	86.0	38.4	87.9
MBR |H|=32	17.5	26.4	86.3	21.7	28.8	25.0	87.3	40.5	89.6
MBR Oracle |H|=8	19.4	29.4	86.8	23.0	32.2	28.9	87.9	44.6	90.5
MBR Oracle |H|=16	20.0	30.0	87.1	24.8	33.7	29.7	88.1	47.0	91.1
MBR Oracle |H|=32	**20.0**	**31.0**	**87.4**	**26.1**	**35.0**	**30.0**	**88.6**	**48.0**	**91.5**

Table 4.5 Oracle score for all three tasks.

We are now interested in how well does MBR rank the hypothesis compared to the ground truth. We extract the MBR and Oracle ranking list. While MBR ranks all the hypothesis based on their expected utility, Oracle ranks them based on their utility with the reference hypothesis. Table 4.6 shows the Spearman's rank correlation coefficient for each task's test set. While we notice for both question generation and summarization a decent correlation, question answering shows little to no correlation with the oracle rankings. We will explore in the subsequent section if this lack of correlation has any effect on preference learning.

Dataset	**Correlation ↑**
Question Generation	0.48
Question Answering	0.05
Summarization	0.44

Table 4.6 Spearman's rank correlation between the oracle rankings and the MBR rankings of the hypothesis samples with |H| = 32 across the tests sets of the selected tasks

4.2 Preference optimization with DPO

In the previous section, we showed that MBR decoding with a varying hypothesis set size is an effective method that improves the evaluation metrics across all three tasks. Despite its performance, MBR remains prohibitively expensive to run at inference time. In this section, we will first demonstrate that a model trained with DPO on the BW preference set, extracted from MBR rankings with |H| = 32, successfully replicates MBR's performance through a single decoding pass. Additionally, we examine the effect that different preference sets have on performance of the post-DPO models.

4.2.1 Training Statistics for BW split

In this section we align the models using the BW split where each prompt has one preference pair. π_{ref} is the original model as we do not conduct SFT. In order to check the validity of the method, we rely on two signals. The first one is improvements on the evaluation metrics for the post-DPO models, the second one is an increasing positive reward margin (Equation 2.24).
Moreover, to get a deeper insight on how the margin works, we will analyse the reward progression of both the winning and losing hypothesis. This provides us with knowledge about the behaviour of each algorithm .

$$Reward_w = \beta * log(\frac{\pi_\theta(y_w|x)}{\pi_{ref}(y_w|x)}), Reward_l = \beta * log(\frac{\pi_\theta(y_l|x)}{\pi_{ref}(y_l|x)}) \tag{4.2}$$

Summarization on CNN/DM

Table 4.7 shows the post-DPO beam search performance on CNN/DM. We see marked improvement in all three metrics compared to pre-DPO beam search. For the metrics that measure semantic similarity wee see that post-DPO models roughly matches the MBR performance. The β value seems to have little effect on the model. The only noticeable effect is for RougeL as the bigger the beta the closer the RougeL value is to the pre-DPO beam search. This lexical diversity observed remains semantically equivalent as both METEOR and BertScore are very similar across β.

Figure 4.1 shows that the reward margins increase as training progresses. This shows that the DPO objective of maximizing reward margin is successfully optimized in training. The reward margins are closely clustered together, this explains the small effect of β on performance. Looking at the reward progression for both y_l Figure 4.2b and y_w Figure 4.2a we can see that they both decrease as training progresses, although the reward of of y_l decreases at much higher rate than the reward of y_w. We attempt to explain this behaviour in Section 4.5.

Systems	RougeL ↑	METEOR ↑	BertScore ↑
Beam Search			
Beam Width = 5	14.0	25.6	85.6
MBR			
|H| = 32	17.5	26.3	86.2
DPO			
$\beta = 0.01$		Hallucinating	
$\beta = 0.1$	**15.3**	26.0	**86.0**
$\beta = 0.5$	15.0	25.9	**86.0**
$\beta = 1.0$	14.7	**26.1**	**86.0**

Table 4.7 Summarization's decoding performance of Beam search with Beam size = 5 on the post-DPO model. The second and fourth row show the performance of the pre-DPO model.

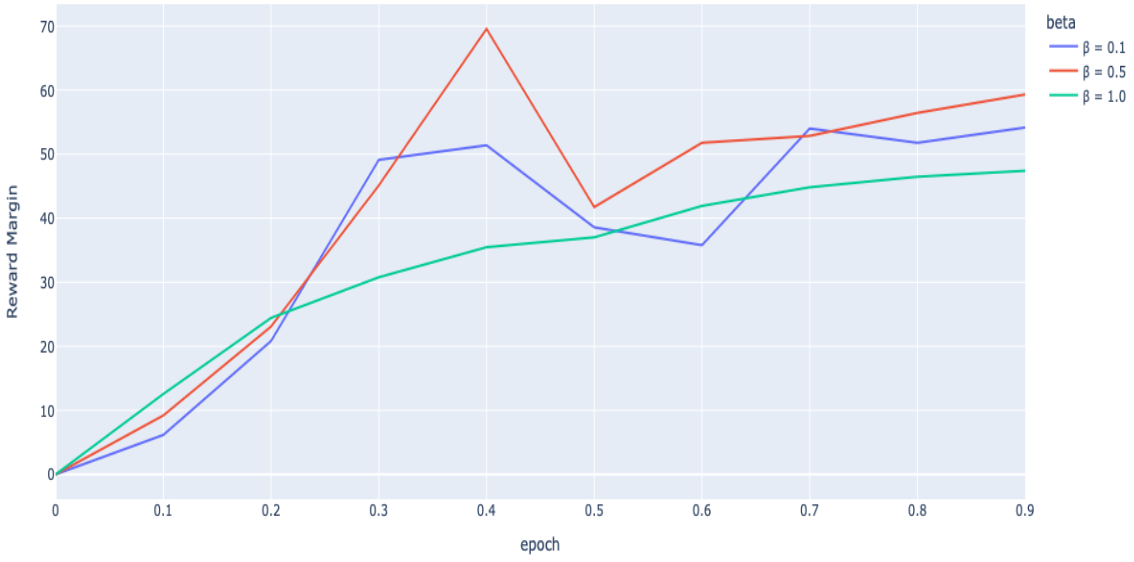
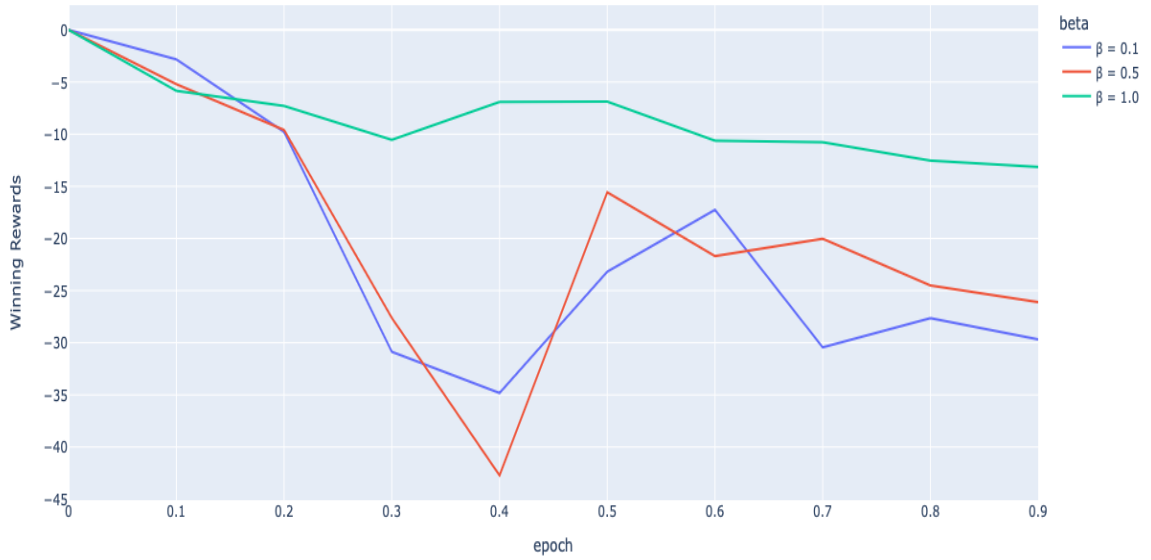
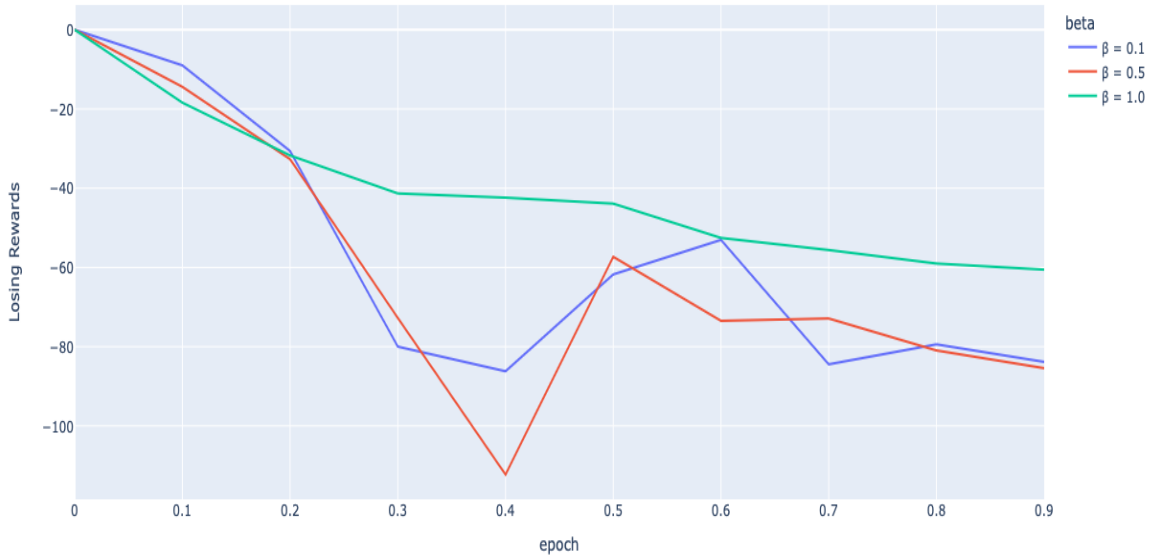


Fig. 4.1 Average Reward margin on CNN/DM as a function of the percentage of the epoch completed for the BW validation set.



(a) $Reward_w$



(b) $Reward_l$

Fig. 4.2 Average Rewards of the winning hypothesis 4.2a and the losing hypothesis 4.2b on CNN/DM as a function of the percentage of the epoch completed for the BW validation set.

Question Answering on StartegyQA

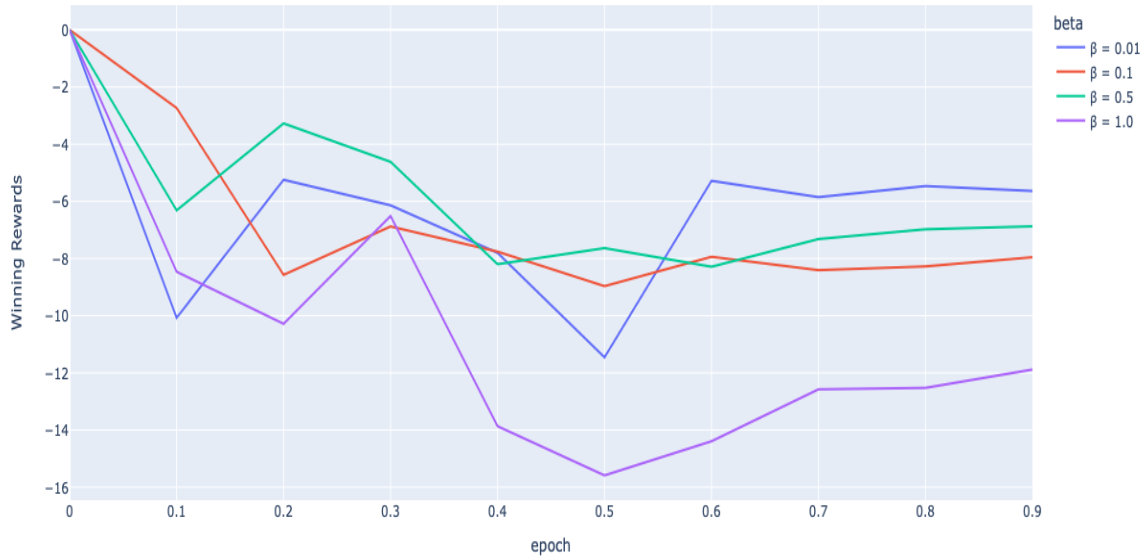
Table 4.8 shows us that the performance of the post-DPO models have improved across all the metrics compared to the pre-DPO Beam Search. METEOR score has seen a sizable improvement compared to the MBR setup. BertScore for $\beta = 0.1$ matches the MBR BertScore. For $\beta = 0.01, 0.1, 0.5$ the evaluation metric scores are close to each other, we notice marginal change between them. While we notice a significant drop in performance for $\beta = 1.0$. As β increases π_θ remains closer to π_{ref}, this explains the difference in the scores.

Figure 4.3, shows the reward margins as training progresses. The reward margins are increasing regardless of β, therefore DPO is working. Despite having different β values, the reward margins are similar in magnitude for $\beta = 0.01, 0.1, 0.5$, explaining why the models perform similarly. Although, the $\beta = 1.0$ system has a higher reward margin it under-performs compared to the rest. Looking at the winning and losing rewards in Figure 4.4, we see that $\beta = 0.01, 0.1, 0.5$ again yield similar behavior, whereas for $\beta = 1.0$, the winning reward and the losing reward decrease by substantial margins. This may explain why the low performance of $\beta = 1.0$ system compared to systems with other β values.

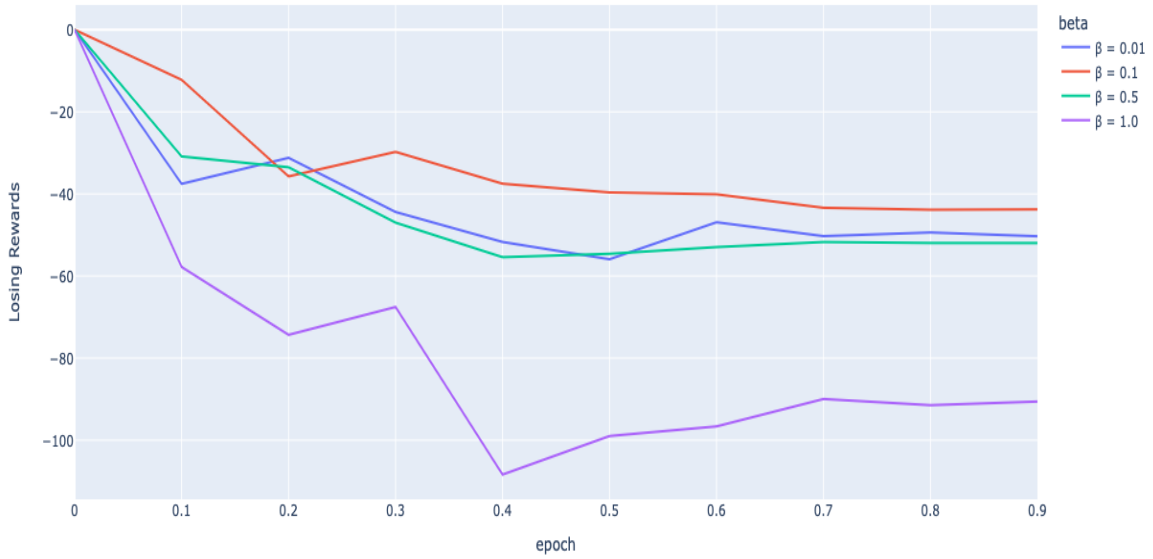
Systems	**RougeL** ↑	**Rouge1** ↑	**METEOR** ↑	**BertScore** ↑
Beam Search				
Beam Width = 5	18.1	25.3	27.0	86.0
MBR				
|H| = 32	21.4	28.8	25.0	87.2
DPO				
$\beta = 0.01$	20.4	28.4	28.1	87.1
$\beta = 0.1$	**20.5**	**28.5**	28.2	**87.2**
$\beta = 0.5$	20.4	28.4	**28.5**	87.1
$\beta = 1.0$	19.0	26.4	27.07	86.7

Table 4.8 Question answering's decoding performance of Beam search with Beam size = 5 on the post-DPO model. The second and fourth row show the performance of the pre-DPO model.

Fig. 4.3 Average Reward margin on StrategyQA as a function of the percentage of the epoch completed for the BW validation set.



(a) $Reward_w$



(b) $Reward_l$

Fig. 4.4 Average Rewards of the winning hypothesis 4.4a and the losing hypothesis 4.4b on StrategyQA in function of the percentage of the epoch completed for the BW validation set.

Question Generation on SquadV2

Table 4.9 shows that the post DPO-models outperform pre-DPO Beam-Search and MBR scores in terms of Bertscore. While the $\beta = 0.5$ model outperforms them in terms of Meteor.

Figure 4.5 shows the positive increasing reward margins for both systems, therefore DPO is working. For $\beta = 1.0$ the reward margin is significantly higher than $\beta = 0.5$, while it under-performs in the metrics. A closer look at Figure 4.6 we can see that both $Reward_w$ and $Reward_l$ are smaller for $\beta = 1.0$ which leads to a higher margin but worse performance. Which explains the difference in performance, as y_w is more dispreferred. The discrepancy in the performance of the post-DPO models is expected, as for $\beta = 0.5$ the model has more freedom to deviate from the reference policy, and accurately learn from the preferences.

Systems	**METEOR ↑**	**BertScore ↑**
Beam Search		
Beam Width = 5	38.4	87.9
MBR		
|H| = 32	40.5	89.6
DPO		
$\beta = 0.01$	Hallucinating	
$\beta = 0.1$	Hallucinating	
$\beta = 0.5$	**42.5**	**90.8**
$\beta = 1.0$	37.2	89.7

Table 4.9 Question generation's decoding performance of Beam search with Beam size = 5 on the post-DPO model. The second and fourth row show the performance of the pre-DPO model.

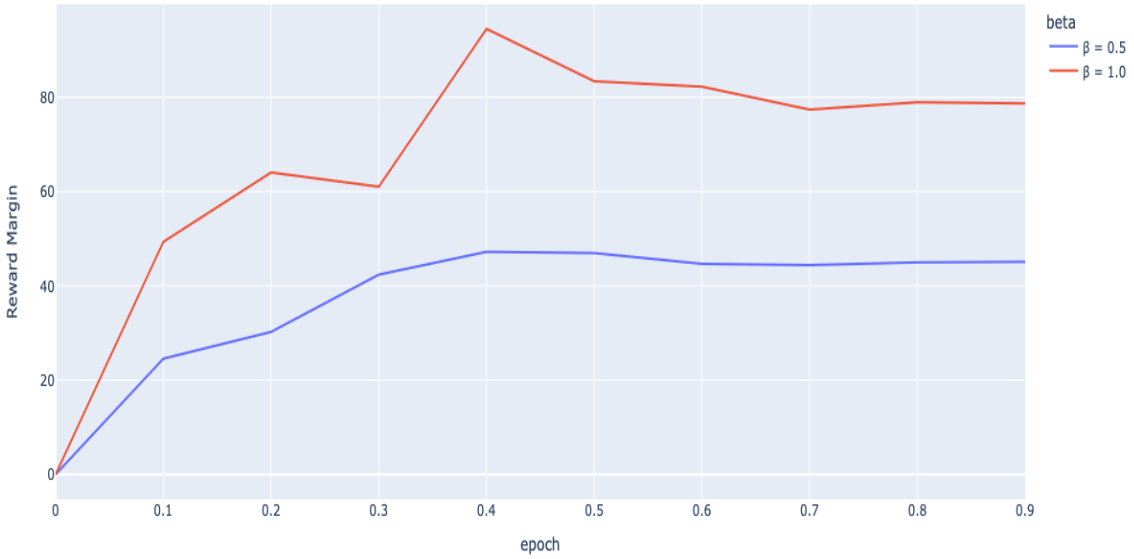
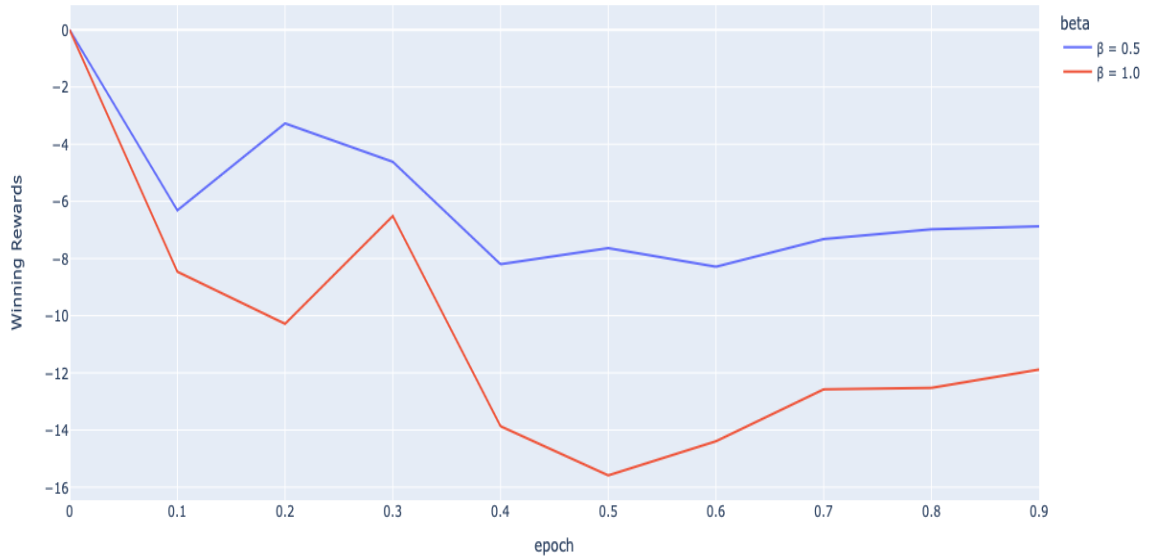
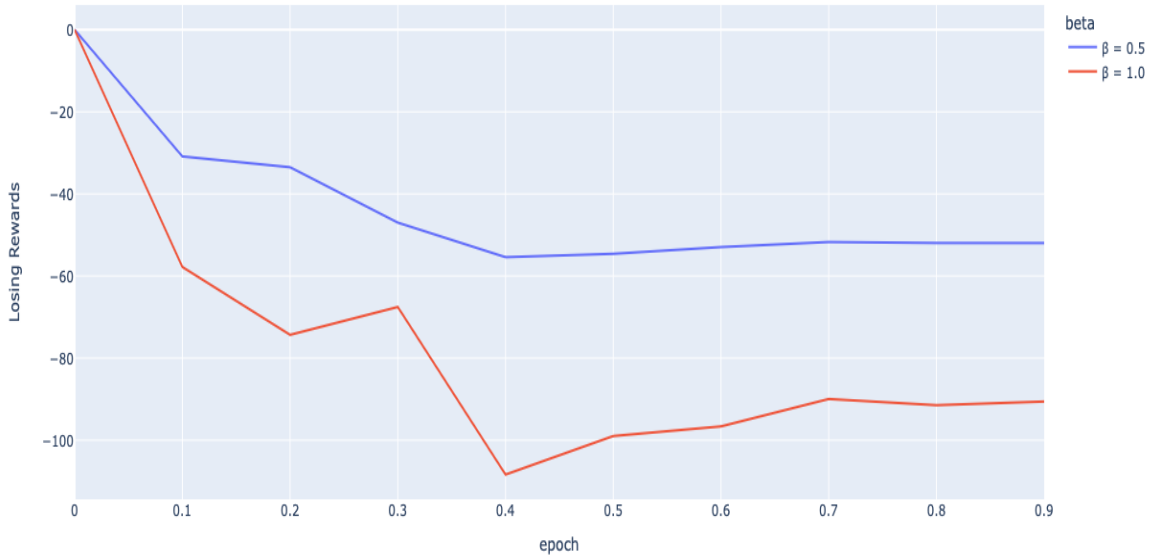


Fig. 4.5 Average Reward margin on SquadV2 as a function of the percentage of the epoch completed for the BW validation set.



(a) $Reward_w$



(b) $Reward_l$

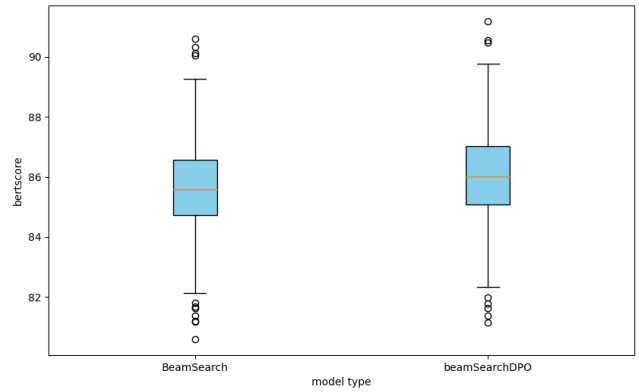
Fig. 4.6 Average Rewards of the winning hypothesis 4.6a and the losing hypothesis 4.6b on SquadV2 in function of the percentage of the epoch completed for the BW validation set.

We have empirically demonstrated that DPO with a preference set extracted from MBR samples successfully learns the MBR preferences, establishing that unsupervised preference learning with MBR works on a wide range of tasks. For all three tasks we observe $Reward_w$ is decreasing, albeit at a lower rate then $Reward_l$, leading to higher reward margin. We attribute this phenomenon to the loss aversion property of DPO discussed in Section 2.2. We will address this in Section 4.5. We also found that lower β values lead to a better performance, as the model is free to deviate from the reference policy. Moreover the correlation with the Oracle Ranking has no bearing on the performance. Question answering on StrategyQA had no correlation and it did achieve higher performance.

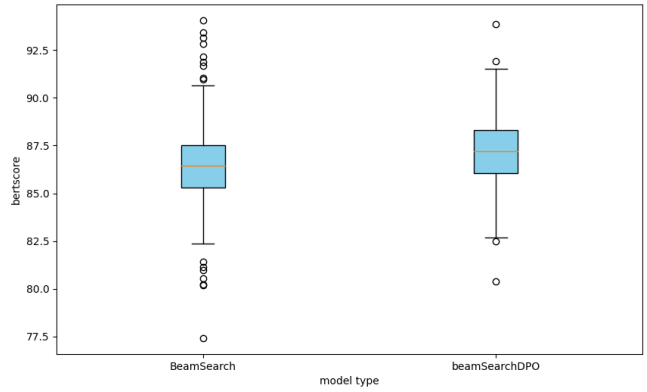
MBR chooses the hypothesis based on the lowest risk, we expect that this behaviour is passed down onto the model. We examine the Box plot distribution of the post-DPO model over the test set. We aim to notice a tighter distribution, with fewer outliers and a smaller variance then the pre-DPO model. Figure 4.7 shows that for all three tasks we have a higher median and less outliers. Observing the variance is less clear form the plots, therefore we include their values in Table 4.10. Table 4.10 shows that both question answering and question generation achieve a lower standard deviation, yet for the summarization task the standard deviation remains slightly higher. Looking at the standard deviation of the MBR hypotheses gives a deeper insight into this behaviour. The post-DPO models inherit the limitation of the MBR hypotheses. If MBR successfully generated hypotheses with a tighter distribution than Beam Search, so will the post-DPO models. In the next section we investigate the effect of different preference sets.

Systems	**std** $\downarrow$
Question Generation	
pre-DPO Beam search	2.40
MBR (32)	2.10
post-DPO Beam search	**2.00**
Question Answering	
pre-DPO Beam search	2.00
MBR (32)	1.95
post-DPO Beam search	**1.72**
Summarization	
pre-DPO Beam search	**1.40**
MBR (32)	1.54
post-DPO Beam search	1.45

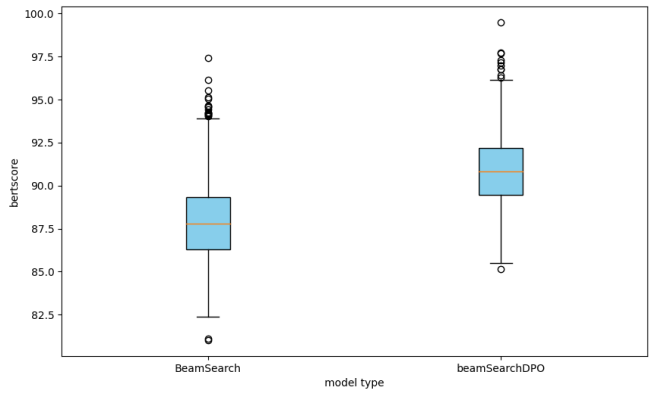
Table 4.10 Standard deviation of the test set scores on all three tasks



(a) Summarization



(b) Question Answering



(c) Question Generation

Fig. 4.7 The Box Plot distribution of the test set scores between the pre-DPO and post-DPO models using Beam Search with a Beam size = 5.

4.2.2 Performance of Different Preference Sets

For the summarization task, Table 4.11 shows that the BMW split with $\beta = 0.1$ achieved the highest performance for BertScore and METEOR. It matched the MBR performance in BertScore and exceeded it in METEOR. Overall, the BMW split shows comparative results with BW split. On the other hand, we notice a drop in performance across all the metrics for the models trained on the CPS split. Their performance degrades below the pre-DPO beam search values. All data splits show the same pattern for CNN/DM, where the β values have little effect on the performance, as they are all more or less similar on the evaluation metrics.

Systems	**RougeL** ↑	**METEOR** ↑	**BertScore** ↑
Beam Search			
Beam Width = 5	14.0	25.6	85.6
MBR			
|H| = 32	17.5	26.3	86.2
DPO BW			
$\beta = 0.1$	15.3	26.0	86.0
DPO BMW			
$\beta = 0.1$	**15.6**	**26.6**	**86.2**
$\beta = 0.5$	14.8	26.6	86.0
$\beta = 1.0$	14.9	26.5	86.0
DPO CPS			
$\beta = 0.1$	13.1	24.8	85.4
$\beta = 0.5$	13.2	24.8	85.5
$\beta = 1.0$	13.0	24.5	85.4

Table 4.11 Summarization's decoding performance of Beam search with Beam size = 5 on the post-DPO models trained on BMW and CPS preference sets.

For the question answering task, Table 4.12 shows that the BMW split with $\beta = 0.01$ achieved the highest performance across all metrics, surpassing the MBR scores. Overall, the performance of both BMW and CPS splits outperform the pre-DPO beam search setup in both METEOR and BertScore. Although, BMW outperforms CPS for similar β values. Both CPS and BMW split notice a decrease in performance as β becomes larger. It is expected, since the bigger the β the less freedom the model has to learn from the preference set.

Systems	**RougeL ↑**	**Rouge1 ↑**	**METEOR ↑**	**BertScore ↑**
Beam Search				
Beam Width = 5	18.1	25.3	27.0	86.0
MBR				
|H| = 32	21.4	28.8	25.0	87.2
DPO BW				
$\beta = 0.1$	20.5	28.5	28.2	87.2
DPO BMW				
$\beta = 0.01$	**21.6**	**29.6**	**28.5**	**87.4**
$\beta = 0.1$	20.1	27.7	28.2	86.9
$\beta = 0.5$	19.4	26.9	27.4	86.8
$\beta = 1.0$	19.0	26.5	27.2	86.6
DPO CPS				
$\beta = 0.01$	21.2	28.5	27.1	87.1
$\beta = 0.1$	17.2	24.0	27.6	86.2
$\beta = 0.5$	17.1	23.9	27.6	86.2
$\beta = 1.0$	17.1	23.8	27.4	86.1

Table 4.12 Question answering's decoding performance of Beam search with Beam size = 5 on the post-DPO models trained on BMW and CPS preference sets.

Concerning the question generation, the previous BW setup had already achieved better results than the MBR setup. Table 4.13 shows that the enhanced performance observed previously holds here. As the BMW split with $\beta = 0.5$ achieves the highest performance on BertScore, and $\beta = 1.0$ the highest on METEOR. Although, the METEOR metric for BMW with $\beta = 0.5$ might not have been able to handle some more complex structures, that are semantically similar to the ground truth. The performance on all BMW models surpasses Beam Search on a pre-DPO model. CPS does deliver improvements over pre-DPO setup, but these improvements are less significant than the BMW case.

We have shown that the preference set does matter, as we observed improvements using the BMW split over the BW for all three tasks. In contrast the CPS split under performed compared to BW and BMW, and in the summarization case it even worsened the generation compared to the pre-DPO model.

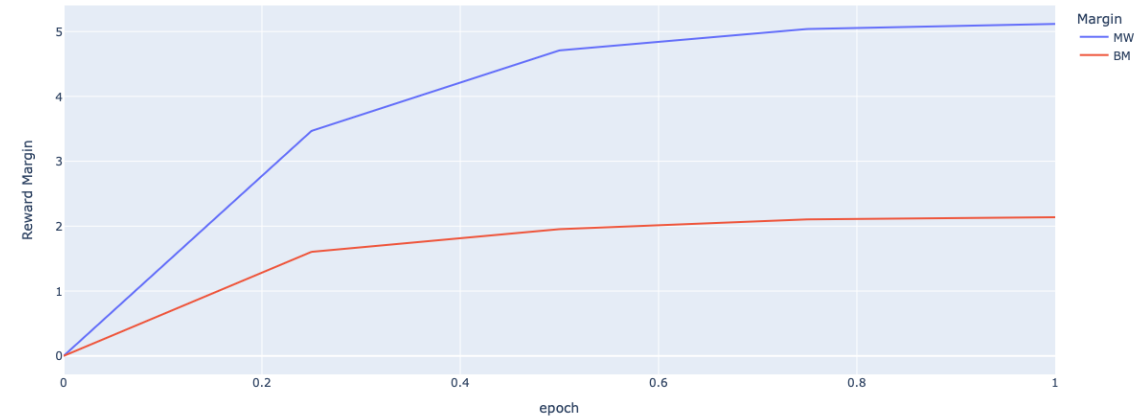
The performance of BMW may be associated with the ability of DPO to implicitly learn the MBR ranking, and learn exactly what makes one sample better than the other. To visualize this behavior we plot the average reward margins for the best-middle $Reward_{BM}$ pairs and the middle-worst $Reward_{MW}$ pairs. Figure 4.8 shows that all three tasks display the same pattern. All the reward margins are positive and increasing, and $Reward_{MW} > Reward_{BM}$. This shows that the model accurately learns to prefer y_w over y_m, and y_m over y_l, reflecting the MBR

Systems	METEOR ↑	BertScore ↑
Beam Search		
Beam Width = 5	38.4	87.9
MBR		
|H| = 32	40.5	89.6
DPO BW		
$\beta = 0.5$	42.5	90.8
DPO BMW		
$\beta = 0.1$	40.3	90.3
$\beta = 0.5$	42.8	**91.1**
$\beta = 1.0$	**44.0**	90.8
DPO CPS		
$\beta = 0.1$	35.5	87.2
$\beta = 0.5$	39.5	88.6
$\beta = 1.0$	40.2	88.5

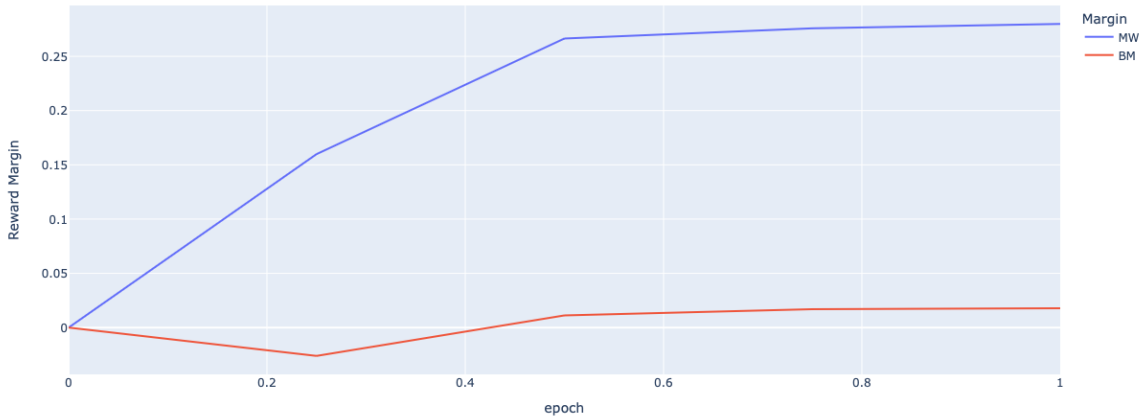
Table 4.13 Question generation's decoding performance of Beam search with Beam size = 5 on the post-DPO models trained on BMW and CPS preference sets.

ranking. In an ideal scenario we would see that both reward margins are equal, since y_m falls in the middle of the MBR ranking. Here we observe that $Reward_{MW} > Reward_{BM}$. Rather than maximizing the margin between y_w and y_m, DPO is maximizing the margin between y_m and y_l. We attribute this behaviour to the loss aversion property, that we will explain in Section 4.5. We can confidently claim that DPO benefits from a more granular dataset, whose preference pairs reflect the MBR ranking.

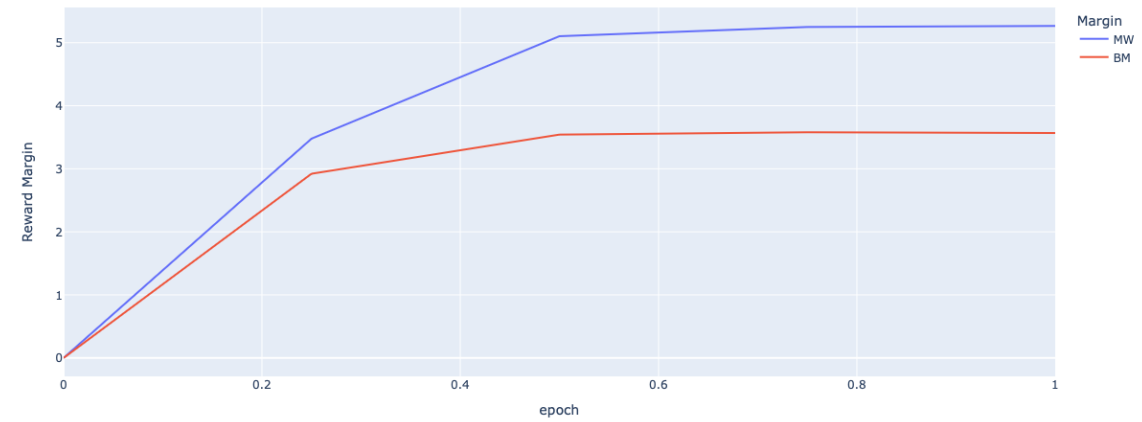
Despite CPS, being a more detailed version of BMW, it fails to match the performance of BW. This phenomenon suggests two things. Either DPO benefits up to a certain point from additional comparative pairs, or the model needs more epochs to accurately capture the fine grained distinctions. Due to the computational and time consuming demands of the CPS split, we leave this exploration to future work.



(a) Summarization



(b) Question Answering



(c) Question Generation

Fig. 4.8 Average reward margins of the $Reward_{BM}$ (red) and $Reward_{MW}$ (blue) on the test set of the BMW data split for all three tasks in function of the percentage of the epoch completed.

4.3 Preference optimization with KTO

In the previous secion, we demonstrated the effectiveness of DPO, and concluded that a model trained with DPO successfully learns from MBR preference pairs. However, we observed for all the tasks, DPO's loss aversion problem indicated by KTO. In this section, we first examine the performance of KTO on all three tasks for $\beta = 0.1$ with the 1:1 preference set. We, also, investigate KTO's performance on an extended and unbalanced dataset, to observe if additional data points provide any improvements. Finally, we will explore the effect of different β values on the performance.

4.3.1 Performance of KTO with $\beta = 0.1$

We chose $\beta = 0.1$ to first probe the performance of KTO, as Ethayarajh et al. [16] recommends that this β value achieves optimal performance. A 1:1 preference set is the closest setup to DPO, as it uses the same samples present in the BW setup. Namely, the N preference pairs that form the BW split were separated into $2N$ samples. Looking at Tables 4.14, 4.16, and 4.15, we notice a pattern across all three tasks. KTO at best matches the performance of DPO with the BW split. It matches its performance in terms of BertScore for the summarization task, while slightly under-performing on the question generation and answering tasks.

For all three tasks KTO with an unbalanced dataset split, under-performs compared to the 1:1 split. Both METEOR and Bertscore decrease as the imbalance gets bigger. This observation shows us that for these tasks KTO cannot leverage the abundance of rejected samples. On the contrary, it lowers the performance.

Adding additional positive samples, albeit of lower quality, matches or slightly under-performs compared the the 1:1 split. Ethayarajh et al. [16] pointed out that one of KTO's property is that the algorithm does not learn from positive samples with relatively small reward. In other words, if it is too difficult to learn what makes this sample desirable the KTO algorithm will not learn from it. This might be the reason why these lower quality additional positive examples did not impact positively the performance.

To analyse the behaviour of KTO as training progresses we plot the reward margins, the reward of the desired samples $Reward_d$, and the reward of the undesired samples $Reward_u$. Figure 4.9 shows that the margin is positively increasing. The KTO objective of increasing the reward of the desirable samples and decreasing the reward of the undesirable ones is successfully working. We do notice significant stagnation in the Margin for both question answering and summarization, and the margins are considerably smaller than their DPO counterpart. Figure 4.10a shows that the reward associated with the desirable hypothesis

(previously the winning) are positive and non decreasing, and the rewards for the undesirable samples are negative and decreasing.

The notable difference between KTO and DPO's reward progressions is that the winning hypothesis has its reward increase under KTO, while its reward decreases under DPO. Both algorithms reduce the losing hypothesis's reward, but in DPO it decreases at much higher rate.

Systems	**RougeL** ↑	**METEOR** ↑	**BertScore** ↑
DPO BW			
$\beta = 0.1$	15.3	26.0	86.0
DPO BMW (Best)			
$\beta = 0.1$	**15.6**	26.6	**86.2**
KTO $\beta = 0.1$			
split = 1:1	15.2	26.7	86.0
split = 1:2	14.3	**26.8**	85.8
split = 1:3	13.4	22.2	83.2
split = 1:4	15.2	22.3	85.0
split = 3:3	15.0	26.5	86.0

Table 4.14 Summarization's decoding performance of Beam search with Beam size = 5 on the post-KTO models trained on the difference preference sets.

Systems	**RougeL** ↑	**Rouge1** ↑	**METEOR** ↑	**BertScore** ↑
DPO BW				
$\beta = 0.1$	20.5	28.5	28.2	87.2
DPO BMW				
$\beta = 0.01$	**21.6**	**29.6**	28.5	**87.4**
KTO $\beta = 0.1$				
split = 1:1	19.7	27.6	**28.6**	87.0
split = 1:2	20.8	29.0	27.4	86.9
split = 1:3	20.8	29.0	27.5	86.8
split = 1:4		Hallucinating		
split = 3:3	20.2	28.5	27.0	86.8

Table 4.15 Question answering's decoding performance of Beam search with Beam size = 5 on the post-KTO models trained on the difference preference sets.

Systems	METEOR ↑	BertScore ↑
DPO BW		
$\beta = 0.5$	42.5	90.8
DPO BMW		
$\beta = 0.5$	42.8	**91.1**
KTO $\beta = 0.1$		
split = 1:1	**44.0**	90.0
split = 1:2	43.3	89.5
split = 1:3	43.2	89.4
split = 1:4	43.1	89.3
split = 3:3	42.3	89.5

Table 4.16 Question generation's decoding performance of Beam search with Beam size = 5 on the post-KTO models trained on the difference preference sets.

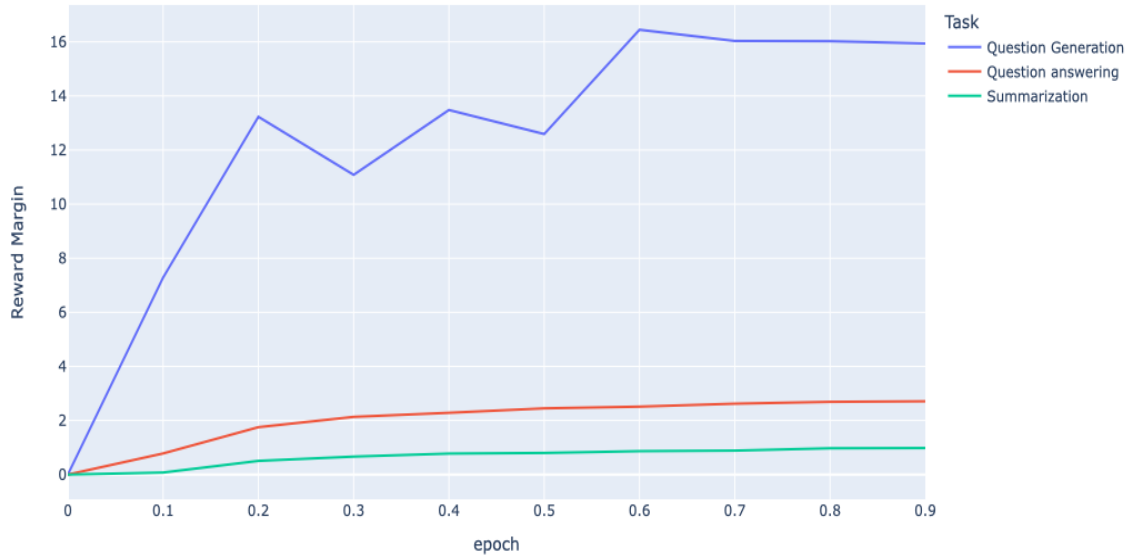
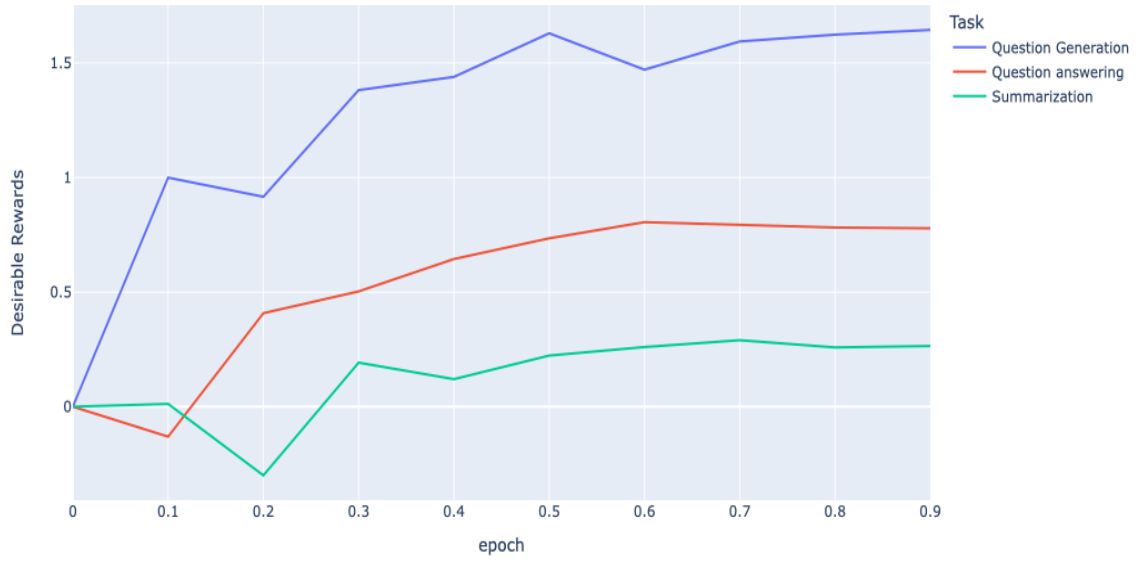
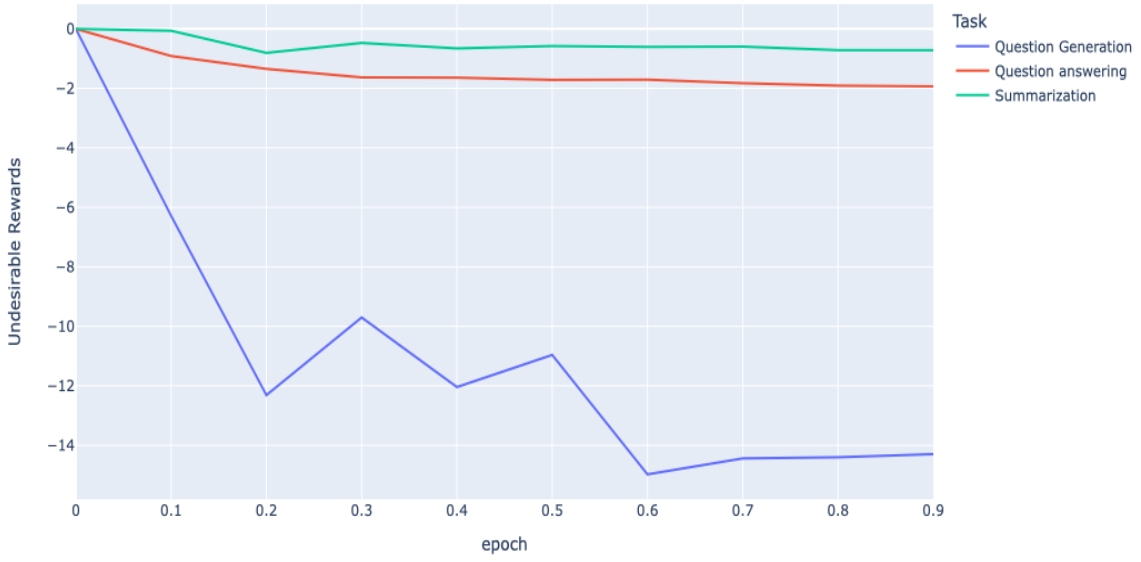


Fig. 4.9 Average Reward margin on the three tasks as a function of the percentage of the epoch completed for KTO trained models with $\beta = 0.1$ and 1:1 data split.



(a) $Reward_d$



(b) $Reward_u$

Fig. 4.10 Average Rewards of the y_d 4.10a and y_u 4.10b on the three tasks in function of the percentage of the epoch completed for KTO trained models with $\beta = 0.1$ and 1:1 data split.

4.3.2 Performance for different β

In the previous section we demonstrated that both algorithm successfully leverage the MBR rankings to align the models. Although for $\beta = 0.1$, KTO either matches or slightly under-performs the DPO results. In this section, we examine if different values of β lead to a better performance. We will only train the subsequent KTO models on 1:1 split, as it outperformed all the others.

Looking closely at Tables 4.17, 4.19, and 4.18 we see very little difference between different β values. Therefore, the previous observation on KTO's performance compared to DPO stands. On a deeper level, for the summarization task Table 4.17 Bertscore remains unchanged, while METEOR decreases as β increases. Moreover, from Table 4.18 we see a slight decrease in both BertScore and METEOR as β increases. While, BertScore decreases and METEOR marginally increases for the question generation task (Table 4.19). After thorough investigation into KTO, we fail to observe the claimed improvements on evaluation metrics that Ethayarajh et al. [16] obtained for different tasks. This contradiction might be due to the nature of the MBR data, we aim to answer this in Section 4.6.

Systems	**RougeL** ↑	**METEOR** ↑	**BertScore** ↑
DPO BW			
$\beta = 0.1$	15.3	26.0	86.0
DPO BMW (Best)			
$\beta = 0.1$	**15.6**	26.6	**86.2**
KTO split=1:1			
$\beta = 0.01$	15.0	**27.0**	86.0
$\beta = 0.1$	15.2	26.7	86.0
$\beta = 0.5$	15.2	26.0	86.0
$\beta = 1.0$	14.8	26.0	86.0

Table 4.17 Summarization's decoding performance of Beam search with Beam size = 5 on the post-KTO models trained on 1:1 split with different β values

Systems	**RougeL** ↑	**Rouge1** ↑	**METEOR** ↑	**BertScore** ↑
DPO BW				
$\beta = 0.1$	20.5	28.5	28.2	87.2
DPO BMW				
$\beta = 0.01$	**21.6**	29.6	28.5	**87.4**
KTO split=1:1				
$\beta = 0.01$	21.4	**30.0**	28.2	87.1
$\beta = 0.1$	19.7	27.6	**28.6**	87.0
$\beta = 0.5$	20.6	28.5	27.0	86.9
$\beta = 1.0$	20.8	29.0	27.1	86.9

Table 4.18 Question answering's decoding performance of Beam search with Beam size = 5 on the post-KTO models trained on 1:1 split with different β values

Systems	**METEOR** ↑	**BertScore** ↑
DPO BW		
$\beta = 0.5$	42.5	90.8
DPO BMW		
$\beta = 0.5$	42.8	**91.1**
KTO split=1:1		
$\beta = 0.01$	Hallucinating	
$\beta = 0.1$	44.0	90.0
$\beta = 0.5$	44.1	89.1
$\beta = 1.0$	**44.2**	89.3

Table 4.19 Question generation's decoding performance of Beam search with Beam size = 5 on the post-KTO models trained on 1:1 split with different β values

4.4 KL-divergence

One way to evaluate how far a model diverges from the reference given it's β value, would be to estimate the KL divergence on the generated samples from the post-DPO and post-KTO models like in Equation 4.3, where N is the number of samples in the test set.

$$KL = \frac{1}{N}\sum_{i}^{N} log(\frac{\pi_\theta(y_i|x)}{\pi_{ref}(y_i|x)} \tag{4.3}$$

We could then estimate which alignment method is more efficient, since we could compare the metric performance with respect to the KL-divergence. We would aim to see higher/equal performance for lower KL. This would mean that the model learns from the preferences without excessively drifting from the reference model.

While estimating the KL values for the DPO models, we notice that most of them are negative. Since the DPO models reduce the rewards of the winning hypothesis rather than improving it, we obtain for the majority of the generated samples a $\pi_{ref}(y_i|x) > \pi_\theta(y_i|x)$, which leads to a negative KL. Therefore, estimating the KL divergence for DPO models is not feasible through the limited samples we generate. On the other hand, since KTO improves the reward of the generated samples we do not observe this estimation problem. For reference we report the KL-divergences of all KTO models for the three tasks. From Table 4.20 we can clearly see that the lower the β the higher the KL-divergence. Therefore, we can experimentally validate that with lower betas the model drifts more from the reference.

Systems	**KL-divergence**
Question answering	
$\beta = 0.01$	14
$\beta = 0.1$	13
$\beta = 0.5$	8.12
$\beta = 1.0$	6.2
Summarization	
$\beta = 0.01$	42
$\beta = 0.1$	36.19
$\beta = 0.5$	30.2
$\beta = 1.0$	30.04
Question generation	
$\beta = 0.1$	12.4
$\beta = 0.5$	12
$\beta = 1.0$	11

Table 4.20 KL-divergence of KTO trained models for all three tasks.

4.5 Loss aversion

Using DPO, we observe for all three datasets that the reward margin is increasing, but both rewards for the winning and losing hypothesis decrease. The reward for the losing hypothesis decreases at a much higher rate, which leads to a positive margin. As DPO's objective is to maximize the reward margin, it is prioritizing decreasing the $Reward_l$ rather than increasing $Reward_w$. DPO is also decreasing $Reward_w$ and making it less likely for the winning hypothesis to be generated. DPO is therefore avoiding to learn what makes the hypothesis desirable, it is rather focusing on preventing risky hypothesises from being generated.

It could be that the winning hypothesis as picked by MBR cannot be learned from so DPO focuses on reducing the reward for the losing hypothesis. But as we saw from KTO, the progression of the $Reward_d$ is positive and increasing. KTO with $\lambda_d > \lambda_u$ [16] is a gain sensitive algorithm that prioritizes learning what makes the sample desirable, rather than trying to avoid generating undesirable outputs. Therefore, it successfully assigns higher $Reward_d$ as training progresses.

We find that the loss aversion property does not necessarily lead to a lower performance. It, in fact, can be positive as the model learns to avoid high risk hypotheses, like outliers that negatively affect the mean metric performance. And as we observed, DPO outperformed KTO on all datasets.

4.6 KTO or DPO ?

KTO relies on a signal that the sample is desirable or not. Therefore, the quality of each sample should clearly reflect why it is suitable or not. When choosing these samples from MBR, we base our choice on the lowest risk rather than the highest performance under the metrics. We cannot claim that most of the undesirable samples chosen by MBR contain clear errors, that would allow KTO to leverage them. The notion of risk is not represented in the KTO loss while it is in DPO's loss, as it contains $\pi_\theta(y_w|x)$ and $\pi_\theta(y_l|x)$. Preference pairs are, therefore, a blessing in disguise. They provide the model with a way to learn the notion of risk, that KTO is oblivious to.

Ethayarajh et al. [16] states that KTO does not learn from negative samples that are hard to learn from. KTO ends up ignoring samples that are hard to learn what makes them bad. In our case the notion of risk is hard to identify from a sample on its own. As the decision for rejected sample was not made on the quality of the sample it self, rather in comparison to another.

Ethayarajh et al. [16] states that their is a potential for KTO to under-fit datasets with little noise and intransitivity, and that DPO might perform better. In other words, for a preference set extracted by humans, it is inevitable that two human annotators have conflicting preference. One might prefer y_1 over y_2 and the other the opposite. This set of contradictions is the intransitivity and noise of the data. In our case, our preference dataset has no intransitivity. Therefore, KTO runs the risk of under-fitting on preference data extracted through MBR. Whilst we cannot assert that KTO has under-fit, as it provided improvements over the pre-KTO models.

Empirically we have seen that DPO outperforms all KTO models. And because of properties of the MBR data that we are using, that do not favor KTO. We can confidently assert that DPO is more suited to MBR preference data.

4.7 Summary

In summary, we presented our results showcasing the effectiveness of MBR with bigger hypothesis sizes. We proved that aligning models from preference data extracted through MBR, works on a wide range of tasks and across optimization algorithms. We presented an explanation why the BMW split performs the best, as it allows the model to inherently learn the MBR ranking. Additionally, we provided an explanation based on empirical data for the loss aversion phenomenon that DPO suffers from. Finally, we compared KTO, a gain sensitive optimization method, with DPO and explained why given the properties of the MBR dataset KTO under-performs compared to DPO.

Chapter 5

Future Work

We have shown the efficacy of unsupervised preference learning on our three selected tasks. Although it is a promising line of work, much remains to be addressed and explored.

1. **Dataset restrictions:** Due to dataset limitations, we used the training set to build preference pairs for both the question generation and question answering tasks. The models might have been exposed to these datasets during training. Therefore, investigating this method on larger datasets could be a useful avenue to explore.

2. **Pair selection methods:** Our results showed that the model benefits from pairs that reflect the MBR rankings. Examining the number of preference pairs, after which the model's performance degrades/stagnates regardless of training time, is an interesting avenue of research.

3. **Using Variations of MBR:** Classical MBR has shown remarkable performance. As highlighted in Section 2.1.3 work has been conducted on improving various aspects of MBR. An investigation into the performance of these methods coupled with preference learning is a promising area yet to be explored.

4. **Which alignment method to use:** As we showcased in our project, KTO does not work as well as DPO on MBR data. There have been a flurry of new preference algorithms, that aim to improve on DPO. Future work could include a more rigorous investigation into which algorithm suits MBR preference data the best.

5. **Beyond Preference Pairs:** The Bradley-Terry model operates on pair-wise comparison only. Development of preference algorithms that go beyond pairs, leveraging a fully ranked list of candidates has been burgeoning in the literature. As MBR outputs a ranked list of candidates such algorithms are well suited to make the most out of the MBR data.

6. **MBR on top of post-aligned models:** Since MBR enhances the decoding performance of pre-aligned models, it could be that it also enhances the decoding performance of post-aligned model. Investigating if further alignment of the post-aligned models on MBR data extracted from the post-aligned models enhances the single pass decoding performance. And analysing if this method has a ceiling, beyond which further alignment does not benefit the model.

Chapter 6

Conclusion

In this project we explored whether Preference optimization with a preference set generated from the MBR ranking works on three natural language generation tasks: Summarization on CNN/DM, Question Generation on SquadV2 and Question Answering on StrategyQA.

We first establish that MBR decoding out-performs Beam Search on all three tasks. Then, we demonstrate that DPO can be applied to learn from the MBR preference data. Our post-DPO systems match pre-DPO MBR performance with beam search, a single-pass decoding algorithm.

In addition, we experimented with KTO showing that it also benefits from a dataset compiled from MBR rankings. Although we did not notice the advertised superior performance over DPO, post-KTO models outperform non-aligned ones. We attribute KTO's inferior performance, compared to DPO, to the properties of the MBR data. Specifically, the ranking is not based on the quality of the samples, it is based on the notion of risk in comparison to other samples. KTO, whose loss does not incorporate direct comparison, is ill-suited to learn from this MBR data.

We conduct extensive ablation study on the effect of key decoding hyper-parameters. Firstly, we find that increasing beam size does not help improve performance on summarization and question answering. Additionally, increasing the MBR hypothesis set from 8 to 16 to 32 yields diminishing performance gain. Significant improvements are seen when transitioning from Beam Search to MBR with a hypothesis set of 8. We also observed that the output obtained through Oracle ranking consistently outperformed the MBR output by a significant margin. This showcases the MBR property to ignore outliers and select the safest option.

We analyze the effect of DPO and KTO hyper-parameters. In particular, we show that the preference set selection strategy matters for both DPO and KTO. For DPO, the BMW split outperforms all others. The BMW split enables the model to learn the MBR ranking from

the dataset. Moreover, the CPS split under performed across all three tasks. An interesting line of work would be to identify why given more preference pairs DPO under performs. For KTO, a 1:1 split leads to the best performance, additional lower quality samples did not benefit the algorithm, and additional rejected samples degraded the performance of the post-KTO model. We consistently find that the smallest, non-hallucinating beta value leads to the best-performing models on all three tasks.

Through a close analysis of the individual reward progression for the winning/losing hypotheses of the evaluation set during training, we identify the loss aversion property of DPO as identified in the literature. DPO is emphasizing on avoiding to generate risky hypothesis rather than incentivizing the generation of safe hypothesis. Remarkably, the loss aversion is not detrimental to the performance of the post-DPO models.
To summarize, we:

1. Show that the MBR+Preference Optimization methodology works on Summarization, Question Answering and Question Generation for both DPO and KTO (Sections 4.2, 4.3).

2. Analyze the effect of key decoding hyper-parameters for beam search and MBR on the three tasks (Section 4.1.2).

3. We inspect the behaviour of the MBR ranking compared to the Oracle ranking in Section 4.1.3.

4. Study the DPO/KTO performance with different preference set selection strategy (Sections 4.2, 4.3.1) and with varying degrees of regularization (Sections 4.2, 4.3.2).

5. We showcase the loss aversion property of DPO in Section 4.5.

6. We explain why MBR data is ill suited for KTO (Section 4.6).

References

[1] Ackley, D. H., Hinton, G. E., and Sejnowski, T. J. (1985). A learning algorithm for boltzmann machines. *Cognitive Science*, 9(1):147–169.

[2] Ainslie, J., Lee-Thorp, J., de Jong, M., Zemlyanskiy, Y., Lebrón, F., and Sanghai, S. (2023). Gqa: Training generalized multi-query transformer models from multi-head checkpoints.

[3] Baharav, T. and Tse, D. (2019). Ultra fast medoid identification via correlated sequential halving. In Wallach, H., Larochelle, H., Beygelzimer, A., d'Alché-Buc, F., Fox, E., and Garnett, R., editors, *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc.

[4] Bai, Y., Kadavath, S., Kundu, S., Askell, A., Kernion, J., Jones, A., Chen, A., Goldie, A., Mirhoseini, A., McKinnon, C., Chen, C., Olsson, C., Olah, C., Hernandez, D., Drain, D., Ganguli, D., Li, D., Tran-Johnson, E., Perez, E., Kerr, J., Mueller, J., Ladish, J., Landau, J., Ndousse, K., Lukosuite, K., Lovitt, L., Sellitto, M., Elhage, N., Schiefer, N., Mercado, N., DasSarma, N., Lasenby, R., Larson, R., Ringer, S., Johnston, S., Kravec, S., Showk, S. E., Fort, S., Lanham, T., Telleen-Lawton, T., Conerly, T., Henighan, T., Hume, T., Bowman, S. R., Hatfield-Dodds, Z., Mann, B., Amodei, D., Joseph, N., McCandlish, S., Brown, T., and Kaplan, J. (2022). Constitutional ai: Harmlessness from ai feedback.

[5] Banerjee, S. and Lavie, A. (2005). METEOR: An automatic metric for MT evaluation with improved correlation with human judgments. In Goldstein, J., Lavie, A., Lin, C.-Y., and Voss, C., editors, *Proceedings of the ACL Workshop on Intrinsic and Extrinsic Evaluation Measures for Machine Translation and/or Summarization*, pages 65–72, Ann Arbor, Michigan. Association for Computational Linguistics.

[6] Bradley, R. A. and Terry, M. E. (1952). Rank analysis of incomplete block designs: I. the method of paired comparisons. *Biometrika*, 39(3/4):324–345.

[7] Bubeck, S., Chandrasekaran, V., Eldan, R., Gehrke, J., Horvitz, E., Kamar, E., Lee, P., Lee, Y. T., Li, Y., Lundberg, S., Nori, H., Palangi, H., Ribeiro, M. T., and Zhang, Y. (2023). Sparks of artificial general intelligence: Early experiments with gpt-4.

[8] Caccia, M., Caccia, L., Fedus, W., Larochelle, H., Pineau, J., and Charlin, L. (2020). Language gans falling short.

[9] Cheng, J. and Vlachos, A. (2023). Faster minimum bayes risk decoding with confidence-based pruning.

[10] Child, R., Gray, S., Radford, A., and Sutskever, I. (2019). Generating long sequences with sparse transformers.

[11] Cho, K., van Merriënboer, B., Bahdanau, D., and Bengio, Y. (2014). On the properties of neural machine translation: Encoder–decoder approaches. In Wu, D., Carpuat, M., Carreras, X., and Vecchi, E. M., editors, *Proceedings of SSST-8, Eighth Workshop on Syntax, Semantics and Structure in Statistical Translation*, pages 103–111, Doha, Qatar. Association for Computational Linguistics.

[12] Chowdhery, A., Narang, S., Devlin, J., Bosma, M., Mishra, G., Roberts, A., Barham, P., Chung, H. W., Sutton, C., Gehrmann, S., Schuh, P., Shi, K., Tsvyashchenko, S., Maynez, J., Rao, A., Barnes, P., Tay, Y., Shazeer, N., Prabhakaran, V., Reif, E., Du, N., Hutchinson, B., Pope, R., Bradbury, J., Austin, J., Isard, M., Gur-Ari, G., Yin, P., Duke, T., Levskaya, A., Ghemawat, S., Dev, S., Michalewski, H., Garcia, X., Misra, V., Robinson, K., Fedus, L., Zhou, D., Ippolito, D., Luan, D., Lim, H., Zoph, B., Spiridonov, A., Sepassi, R., Dohan, D., Agrawal, S., Omernick, M., Dai, A. M., Pillai, T. S., Pellat, M., Lewkowycz, A., Moreira, E., Child, R., Polozov, O., Lee, K., Zhou, Z., Wang, X., Saeta, B., Diaz, M., Firat, O., Catasta, M., Wei, J., Meier-Hellstern, K., Eck, D., Dean, J., Petrov, S., and Fiedel, N. (2022). Palm: Scaling language modeling with pathways.

[13] Christiano, P. F., Leike, J., Brown, T. B., Martic, M., Legg, S., and Amodei, D. (2017). Deep reinforcement learning from human preferences. In Guyon, I., von Luxburg, U., Bengio, S., Wallach, H. M., Fergus, R., Vishwanathan, S. V. N., and Garnett, R., editors, *NIPS*, pages 4299–4307.

[14] Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. (2019). BERT: Pre-training of deep bidirectional transformers for language understanding. In Burstein, J., Doran, C., and Solorio, T., editors, *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

[15] Eikema, B. and Aziz, W. (2022). Sampling-based approximations to minimum Bayes risk decoding for neural machine translation. In Goldberg, Y., Kozareva, Z., and Zhang, Y., editors, *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 10978–10993, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

[16] Ethayarajh, K., Xu, W., Muennighoff, N., Jurafsky, D., and Kiela, D. (2024). Kto: Model alignment as prospect theoretic optimization.

[17] Fan, A., Lewis, M., and Dauphin, Y. (2018). Hierarchical neural story generation. In Gurevych, I. and Miyao, Y., editors, *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.

[18] Ficler, J. and Goldberg, Y. (2017). Controlling linguistic style aspects in neural language generation. In Brooke, J., Solorio, T., and Koppel, M., editors, *Proceedings of the Workshop on Stylistic Variation*, pages 94–104, Copenhagen, Denmark. Association for Computational Linguistics.

[19] Freitag, M., Ghorbani, B., and Fernandes, P. (2023). Epsilon sampling rocks: Investigating sampling strategies for minimum Bayes risk decoding for machine translation. In Bouamor, H., Pino, J., and Bali, K., editors, *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 9198–9209, Singapore. Association for Computational Linguistics.

[20] Freitag, M., Grangier, D., Tan, Q., and Liang, B. (2022). High Quality Rather than High Model Probability: Minimum Bayes Risk Decoding with Neural Metrics. *Transactions of the Association for Computational Linguistics*, 10:811–825.

[21] Gehring, J., Auli, M., Grangier, D., Yarats, D., and Dauphin, Y. N. (2017). Convolutional sequence to sequence learning. In Precup, D. and Teh, Y. W., editors, *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 1243–1252. PMLR.

[22] Go, D., Korbak, T., Kruszewski, G., Rozen, J., Ryu, N., and Dymetman, M. (2023). Aligning language models with preferences through f-divergence minimization.

[23] Goel, V. and Byrne, W. J. (2000). Minimum bayes-risk automatic speech recognition. *Comput. Speech Lang.*, 14:115–135.

[24] Graves, A. (2012). Sequence transduction with recurrent neural networks.

[25] Hashimoto, T. B., Zhang, H., and Liang, P. (2019). Unifying human and statistical evaluation for natural language generation. In Burstein, J., Doran, C., and Solorio, T., editors, *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 1689–1701, Minneapolis, Minnesota. Association for Computational Linguistics.

[26] Hermann, K. M., Kocisky, T., Grefenstette, E., Espeholt, L., Kay, W., Suleyman, M., and Blunsom, P. (2015). Teaching machines to read and comprehend. In Cortes, C., Lawrence, N., Lee, D., Sugiyama, M., and Garnett, R., editors, *Advances in Neural Information Processing Systems*, volume 28. Curran Associates, Inc.

[27] Hewitt, J., Manning, C. D., and Liang, P. (2022). Truncation sampling as language model desmoothing.

[28] Holtzman, A., Buys, J., Du, L., Forbes, M., and Choi, Y. (2020). The curious case of neural text degeneration.

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

[30] Jiang, A. Q., Sablayrolles, A., Mensch, A., Bamford, C., Chaplot, D. S., de las Casas, D., Bressand, F., Lengyel, G., Lample, G., Saulnier, L., Lavaud, L. R., Lachaux, M.-A., Stock, P., Scao, T. L., Lavril, T., Wang, T., Lacroix, T., and Sayed, W. E. (2023). Mistral 7b.

[31] Jinnai, Y. and Ariu, K. (2024). Hyperparameter-free approach for faster minimum bayes risk decoding.

[32] Jinnai, Y., Honda, U., Morimura, T., and Zhang, P. (2024a). Generating diverse and high-quality texts by minimum bayes risk decoding.

[33] Jinnai, Y., Morimura, T., Honda, U., Ariu, K., and Abe, K. (2024b). Model-based minimum bayes risk decoding for text generation.

[34] Kahneman, D. and Tversky, A. (1979). Prospect theory: An analysis of decision under risk. *Econometrica*, 47(2):263–291.

[35] Korbak, T., Elsahar, H., Kruszewski, G., and Dymetmant, M. (2024). On reinforcement learning and distribution matching for fine-tuning language models with no catastrophic forgetting. In *Proceedings of the 36th International Conference on Neural Information Processing Systems*, NIPS '22, Red Hook, NY, USA. Curran Associates Inc.

[36] Kumar, S. and Byrne, W. (2004). Minimum Bayes-risk decoding for statistical 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 169–176, Boston, Massachusetts, USA. Association for Computational Linguistics.

[37] Lin, C.-Y. (2004). ROUGE: A package for automatic evaluation of summaries. In *Text Summarization Branches Out*, pages 74–81, Barcelona, Spain. Association for Computational Linguistics.

[38] Müller, M. and Sennrich, R. (2021). Understanding the properties of minimum Bayes risk decoding in neural machine translation. In Zong, C., Xia, F., Li, W., and Navigli, R., editors, *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 259–272, Online. Association for Computational Linguistics.

[39] Nallapati, R., Zhou, B., dos Santos, C., Gulçehre, Ç., and Xiang, B. (2016). Abstractive text summarization using sequence-to-sequence RNNs and beyond. In Riezler, S. and Goldberg, Y., editors, *Proceedings of the 20th SIGNLL Conference on Computational Natural Language Learning*, pages 280–290, Berlin, Germany. Association for Computational Linguistics.

[40] Ott, M., Auli, M., Grangier, D., and Ranzato, M. (2018). Analyzing uncertainty in neural machine translation. In Dy, J. and Krause, A., editors, *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 3956–3965. PMLR.

[41] Ouyang, L., Wu, J., Jiang, X., Almeida, D., Wainwright, C. L., Mishkin, P., Zhang, C., Agarwal, S., Slama, K., Ray, A., Schulman, J., Hilton, J., Kelton, F., Miller, L., Simens, M., Askell, A., Welinder, P., Christiano, P., Leike, J., and Lowe, R. (2022). Training language models to follow instructions with human feedback.

[42] Rafailov, R., Sharma, A., Mitchell, E., Ermon, S., Manning, C. D., and Finn, C. (2023). Direct preference optimization: Your language model is secretly a reward model.

[43] Rajpurkar, P., Zhang, J., Lopyrev, K., and Liang, P. (2016). SQuAD: 100,000+ questions for machine comprehension of text. In Su, J., Duh, K., and Carreras, X., editors, *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2383–2392, Austin, Texas. Association for Computational Linguistics.

[44] Reimers, N. and Gurevych, I. (2019). Sentence-BERT: Sentence embeddings using Siamese BERT-networks. In Inui, K., Jiang, J., Ng, V., and Wan, X., editors, *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 3982–3992, Hong Kong, China. Association for Computational Linguistics.

[45] Schulman, J., Wolski, F., Dhariwal, P., Radford, A., and Klimov, O. (2017). Proximal policy optimization algorithms.

[46] Sellam, T., Das, D., and Parikh, A. (2020). BLEURT: Learning robust metrics for text generation. In Jurafsky, D., Chai, J., Schluter, N., and Tetreault, J., editors, *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7881–7892, Online. Association for Computational Linguistics.

[47] Shen, T., Ott, M., Auli, M., and Ranzato, M. (2019). Mixture models for diverse machine translation: Tricks of the trade. In Chaudhuri, K. and Salakhutdinov, R., editors, *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 5719–5728. PMLR.

[48] Srivastava, A., Rastogi, A., Rao, A., Shoeb, A. A. M., Abid, A., Fisch, A., Brown, A. R., Santoro, A., Gupta, A., Garriga-Alonso, A., Kluska, A., Lewkowycz, A., Agarwal, A., Power, A., Ray, A., Warstadt, A., Kocurek, A. W., Safaya, A., Tazarv, A., Xiang, A., Parrish, A., Nie, A., Hussain, A., Askell, A., Dsouza, A., Slone, A., Rahane, A., Iyer, A. S., Andreassen, A., Madotto, A., Santilli, A., Stuhlmüller, A., Dai, A., La, A., Lampinen, A., Zou, A., Jiang, A., Chen, A., Vuong, A., Gupta, A., Gottardi, A., Norelli, A., Venkatesh, A., Gholamidavoodi, A., Tabassum, A., Menezes, A., Kirubarajan, A., Mullokandov, A., Sabharwal, A., Herrick, A., Efrat, A., Erdem, A., Karakaş, A., Roberts, B. R., Loe, B. S., Zoph, B., Bojanowski, B., Özyurt, B., Hedayatnia, B., Neyshabur, B., Inden, B., Stein, B., Ekmekci, B., Lin, B. Y., Howald, B., Orinion, B., Diao, C., Dour, C., Stinson, C., Argueta, C., Ramírez, C. F., Singh, C., Rathkopf, C., Meng, C., Baral, C., Wu, C., Callison-Burch, C., Waites, C., Voigt, C., Manning, C. D., Potts, C., Ramirez, C., Rivera, C. E., Siro, C., Raffel, C., Ashcraft, C., Garbacea, C., Sileo, D., Garrette, D., Hendrycks, D., Kilman, D., Roth, D., Freeman, D., Khashabi, D., Levy, D., González, D. M., Perszyk, D., Hernandez, D., Chen, D., Ippolito, D., Gilboa, D., Dohan, D., Drakard, D., Jurgens, D., Datta, D., Ganguli, D., Emelin, D., Kleyko, D., Yuret, D., Chen, D., Tam, D., Hupkes, D., Misra, D., Buzan, D., Mollo, D. C., Yang, D., Lee, D.-H., Schrader, D., Shutova, E., Cubuk, E. D., Segal, E., Hagerman, E., Barnes, E., Donoway, E., Pavlick, E., Rodola, E., Lam, E., Chu, E., Tang, E., Erdem, E., Chang, E., Chi, E. A., Dyer, E., Jerzak, E., Kim, E., Manyasi, E. E., Zheltonozhskii, E., Xia, F., Siar, F., Martínez-Plumed, F., Happé, F., Chollet, F., Rong, F., Mishra, G., Winata, G. I., de Melo, G., Kruszewski, G., Parascandolo, G., Mariani, G., Wang, G., Jaimovitch-López, G., Betz, G., Gur-Ari, G., Galijasevic, H., Kim, H., Rashkin, H., Hajishirzi, H., Mehta, H., Bogar, H., Shevlin, H., Schütze, H., Yakura, H., Zhang, H., Wong, H. M., Ng, I., Noble, I., Jumelet, J., Geissinger, J., Kernion, J., Hilton, J., Lee, J., Fisac, J. F., Simon, J. B., Koppel, J., Zheng, J., Zou, J., Kocoń, J., Thompson, J., Wingfield, J., Kaplan, J., Radom, J., Sohl-Dickstein, J., Phang, J.,

Wei, J., Yosinski, J., Novikova, J., Bosscher, J., Marsh, J., Kim, J., Taal, J., Engel, J., Alabi, J., Xu, J., Song, J., Tang, J., Waweru, J., Burden, J., Miller, J., Balis, J. U., Batchelder, J., Berant, J., Frohberg, J., Rozen, J., Hernandez-Orallo, J., Boudeman, J., Guerr, J., Jones, J., Tenenbaum, J. B., Rule, J. S., Chua, J., Kanclerz, K., Livescu, K., Krauth, K., Gopalakrishnan, K., Ignatyeva, K., Markert, K., Dhole, K. D., Gimpel, K., Omondi, K., Mathewson, K., Chiafullo, K., Shkaruta, K., Shridhar, K., McDonell, K., Richardson, K., Reynolds, L., Gao, L., Zhang, L., Dugan, L., Qin, L., Contreras-Ochando, L., Morency, L.-P., Moschella, L., Lam, L., Noble, L., Schmidt, L., He, L., Colón, L. O., Metz, L., Şenel, L. K., Bosma, M., Sap, M., ter Hoeve, M., Farooqi, M., Faruqui, M., Mazeika, M., Baturan, M., Marelli, M., Maru, M., Quintana, M. J. R., Tolkiehn, M., Giulianelli, M., Lewis, M., Potthast, M., Leavitt, M. L., Hagen, M., Schubert, M., Baitemirova, M. O., Arnaud, M., McElrath, M., Yee, M. A., Cohen, M., Gu, M., Ivanitskiy, M., Starritt, M., Strube, M., Swędrowski, M., Bevilacqua, M., Yasunaga, M., Kale, M., Cain, M., Xu, M., Suzgun, M., Walker, M., Tiwari, M., Bansal, M., Aminnaseri, M., Geva, M., Gheini, M., T, M. V., Peng, N., Chi, N. A., Lee, N., Krakover, N. G.-A., Cameron, N., Roberts, N., Doiron, N., Martinez, N., Nangia, N., Deckers, N., Muennighoff, N., Keskar, N. S., Iyer, N. S., Constant, N., Fiedel, N., Wen, N., Zhang, O., Agha, O., Elbaghdadi, O., Levy, O., Evans, O., Casares, P. A. M., Doshi, P., Fung, P., Liang, P. P., Vicol, P., Alipoormolabashi, P., Liao, P., Liang, P., Chang, P., Eckersley, P., Htut, P. M., Hwang, P., Miłkowski, P., Patil, P., Pezeshkpour, P., Oli, P., Mei, Q., Lyu, Q., Chen, Q., Banjade, R., Rudolph, R. E., Gabriel, R., Habacker, R., Risco, R., Millière, R., Garg, R., Barnes, R., Saurous, R. A., Arakawa, R., Raymaekers, R., Frank, R., Sikand, R., Novak, R., Sitelew, R., LeBras, R., Liu, R., Jacobs, R., Zhang, R., Salakhutdinov, R., Chi, R., Lee, R., Stovall, R., Teehan, R., Yang, R., Singh, S., Mohammad, S. M., Anand, S., Dillavou, S., Shleifer, S., Wiseman, S., Gruetter, S., Bowman, S. R., Schoenholz, S. S., Han, S., Kwatra, S., Rous, S. A., Ghazarian, S., Ghosh, S., Casey, S., Bischoff, S., Gehrmann, S., Schuster, S., Sadeghi, S., Hamdan, S., Zhou, S., Srivastava, S., Shi, S., Singh, S., Asaadi, S., Gu, S. S., Pachchigar, S., Toshniwal, S., Upadhyay, S., Shyamolima, Debnath, Shakeri, S., Thormeyer, S., Melzi, S., Reddy, S., Makini, S. P., Lee, S.-H., Torene, S., Hatwar, S., Dehaene, S., Divic, S., Ermon, S., Biderman, S., Lin, S., Prasad, S., Piantadosi, S. T., Shieber, S. M., Misherghi, S., Kiritchenko, S., Mishra, S., Linzen, T., Schuster, T., Li, T., Yu, T., Ali, T., Hashimoto, T., Wu, T.-L., Desbordes, T., Rothschild, T., Phan, T., Wang, T., Nkinyili, T., Schick, T., Kornev, T., Tunduny, T., Gerstenberg, T., Chang, T., Neeraj, T., Khot, T., Shultz, T., Shaham, U., Misra, V., Demberg, V., Nyamai, V., Raunak, V., Ramasesh, V., Prabhu, V. U., Padmakumar, V., Srikumar, V., Fedus, W., Saunders, W., Zhang, W., Vossen, W., Ren, X., Tong, X., Zhao, X., Wu, X., Shen, X., Yaghoobzadeh, Y., Lakretz, Y., Song, Y., Bahri, Y., Choi, Y., Yang, Y., Hao, Y., Chen, Y., Belinkov, Y., Hou, Y., Hou, Y., Bai, Y., Seid, Z., Zhao, Z., Wang, Z., Wang, Z. J., Wang, Z., and Wu, Z. (2023). Beyond the imitation game: Quantifying and extrapolating the capabilities of language models.

[49] Stiennon, N., Ouyang, L., Wu, J., Ziegler, D. M., Lowe, R., Voss, C., Radford, A., Amodei, D., and Christiano, P. (2022). Learning to summarize from human feedback.

[50] Stolcke, A., Konig, Y., and Weintraub, M. (1997). Explicit word error minimization in n-best list rescoring. In *Proc. 5th European Conference on Speech Communication and Technology (Eurospeech 1997)*, pages 163–166.

[51] Sutskever, I., Vinyals, O., and Le, Q. V. (2014). Sequence to sequence learning with neural networks. In Ghahramani, Z., Welling, M., Cortes, C., Lawrence, N., and

Weinberger, K., editors, *Advances in Neural Information Processing Systems*, volume 27. Curran Associates, Inc.

[52] Suzgun, M., Melas-Kyriazi, L., and Jurafsky, D. (2023). Follow the wisdom of the crowd: Effective text generation via minimum Bayes risk decoding. In Rogers, A., Boyd-Graber, J., and Okazaki, N., editors, *Findings of the Association for Computational Linguistics: ACL 2023*, pages 4265–4293, Toronto, Canada. Association for Computational Linguistics.

[53] Tian, K., Mitchell, E., Yao, H., Manning, C. D., and Finn, C. (2023). Fine-tuning language models for factuality.

[54] Touvron, H., Lavril, T., Izacard, G., Martinet, X., Lachaux, M.-A., Lacroix, T., Rozière, B., Goyal, N., Hambro, E., Azhar, F., Rodriguez, A., Joulin, A., Grave, E., and Lample, G. (2023a). Llama: Open and efficient foundation language models.

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

[56] Tu, Z., Liu, Y., Shang, L., Liu, X., and Li, H. (2016). Neural machine translation with reconstruction.

[57] Tunstall, L., Beeching, E., Lambert, N., Rajani, N., Rasul, K., Belkada, Y., Huang, S., von Werra, L., Fourrier, C., Habib, N., Sarrazin, N., Sanseviero, O., Rush, A. M., and Wolf, T. (2023). Zephyr: Direct distillation of lm alignment.

[58] Tversky, A. and Kahneman, D. (1992). Advances in prospect theory: Cumulative representation of uncertainty. *Journal of Risk and Uncertainty*, 5(4):297–323.

[59] Vamvas, J. and Sennrich, R. (2024). Linear-time minimum bayes risk decoding with reference aggregation.

[60] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L. u., and Polosukhin, I. (2017). Attention is all you need. In Guyon, I., Luxburg, U. V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., and Garnett, R., editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc.

[61] Wang, Y., Kordi, Y., Mishra, S., Liu, A., Smith, N. A., Khashabi, D., and Hajishirzi, H. (2023). Self-instruct: Aligning language models with self-generated instructions. In Rogers, A., Boyd-Graber, J., and Okazaki, N., editors, *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 13484–13508, Toronto, Canada. Association for Computational Linguistics.

[62] Yang, G., Chen, J., Lin, W., and Byrne, B. (2024). Direct preference optimization for neural machine translation with minimum Bayes risk decoding. In Duh, K., Gomez, H., and Bethard, S., editors, *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 2: Short Papers)*, pages 391–398, Mexico City, Mexico. Association for Computational Linguistics.

[63] Yang, Y., Huang, L., and Ma, M. (2018). Breaking the beam search curse: A study of (re-)scoring methods and stopping criteria for neural machine translation. In Riloff, E., Chiang, D., Hockenmaier, J., and Tsujii, J., editors, *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 3054–3059, Brussels, Belgium. Association for Computational Linguistics.

[64] Yuan, Z., Yuan, H., Tan, C., Wang, W., Huang, S., and Huang, F. (2023). Rrhf: Rank responses to align language models with human feedback without tears.

[65] Zhang, T., Kishore, V., Wu, F., Weinberger, K. Q., and Artzi, Y. (2020). Bertscore: Evaluating text generation with bert.

[66] Ziegler, D. M., Stiennon, N., Wu, J., Brown, T. B., Radford, A., Amodei, D., Christiano, P., and Irving, G. (2020). Fine-tuning language models from human preferences.