Efficiently Acquiring Human Feedback with Bayesian Deep Learning

Haishuo Fang[†§*], **Jeet Gor**[†], **Edwin Simpson**[†]
[†] Intelligent Systems Lab, University of Bristol
[§] UKP Lab, Technical University of Darmstadt
haishuo.fang@tu-darmstadt.de
jeetgor06@gmail.com
edwin.simpson@bristol.ac.uk

Abstract

Learning from human feedback can improve models for text generation or passage ranking, aligning them better to a user's needs. Data is often collected by asking users to compare alternative outputs to a given input, which may require a large number of comparisons to learn a ranking function. The amount of comparisons needed can be reduced using Bayesian Optimisation (BO) to query the user about only the most promising candidate outputs. Previous applications of BO to text ranking relied on shallow surrogate models to learn ranking functions over candidate outputs, and were therefore unable to fine-tune rankers based on deep, pretrained language models. This paper leverages Bayesian deep learning (BDL) to adapt pretrained language models to highly specialised text ranking tasks, using BO to tune the model with a small number of pairwise preferences between candidate outputs. We apply our approach to community question answering (cQA) and extractive multi-document summarisation (MDS) with simulated noisy users, finding that our BDL approach significantly outperforms both a shallow Gaussian process model and traditional active learning with a standard deep neural network, while remaining robust to noise in the user feedback.

1 Introduction

In many NLP tasks, the ideal output is highly user or topic-specific, presenting a challenge to general-purpose models. For example, in community question answering (cQA), the system must identify the most helpful answer to present to a user, by processing a complex question on a niche topic, which assumes substantial background knowledge, and selecting between long, multi-sentence answers (Tran et al., 2015; Rücklé et al., 2019; Deng et al., 2020). Likewise, to provide an optimal summary of a set of documents, we may need to know what information a particular user will find beneficial and how best to present a topic to them (López et al., 1999). In these situations, the challenge is to identify which output a user will prefer.

One way to adapt NLP models to these specialised tasks is to acquire data through human-in-the-loop interactive learning, in which a user is presented with pairs of candidate outputs, and asked to select the most appropriate candidate in each pair (Simpson et al., 2020). The pairwise labels can then be used to train ranking models, which predict a score for each candidate that can be viewed as its *utility* to the user. Typically, such *preference learning* approaches use either the Bradley-Terry (Bradley and Terry, 1952; Luce, 1959; Plackett, 1975) or Thurstone-Mosteller (Thurstone, 1927; Mosteller, 1951) model to map the utilities to pairwise labels. Pairwise labelling typically reduces the user's cognitive burden compared with scoring candidates directly (Yang and Chen, 2011).

In most practical applications of preference learning, the user effort required to read and compare different candidates needs to be minimal. Therefore, Simpson et al. (2020) use *Bayesian optimisation (BO)* (Močkus, 1975) to reduce the number of pairwise labels required to find the optimal output by actively choosing promising candidates for the user to compare. The Bayesian approach accounts for *epistemic uncertainty*, i.e., the uncertainty in the parameters of the model when learning from a finite dataset. BO uses this uncertainty to identify candidates that the model may currently be underrating – those for which its judgement is uncertain and could be increased given suitable training samples. It then asks the user to compare these selected candidates with its current best candidate. BO thus avoids asking the user to compare outputs that are unlikely to satisfy the user.

A limitation of the approach by Simpson et al. (2020) is the use of Gaussian process preference learning (GPPL) as a ranking model, which relies

[*] Work done during Intellignet Systems Lab of the University of Bristol.

Proceedings of the 1st Workshop on Uncertainty-Aware NLP (UncertaiNLP 2024), pages 70–80

Q1: Why is an album in Apple Music marked with an " E " and grayed out ?
A1 (GPPL): If you want your **music** to be uploaded to iCloud , you must sign up for iTunes Match. Make sure you're running the latest version of OS X and iOS. (El Capitan - iOS 9 are the stable releases right now, although you can get the public beta of macOS Sierra and iOS 10). Then make sure the Apple ID you've entered in the Music App on iOS is the ...
A2 (non-Bayesian deep ranker): In my experience, **the cover photo is simply the first photo in the album**. When you 're within an album, hit the button in the upper-right corner to manipulate the photos; then tap and hold a photo until it "inflates", at which point you can drag it to move it around. You 'll just have to be happy with your album cover being the first photo...
A3 (Bayesian deep ranker): The E symbol means " Explicit ". You can enable or disable explicit content within Settings on your device . You'll find the "ALLOW MUSIC; PODCASTS RATED "option under General $->$ Restrictions $->$ Music Podcasts . You may need to enter a PIN code to access the Restrictions settings. You can disable the ...

Table 1: Example outputs for different methods after 4 interactions in cQA. Only A3 is correct.

on fixed embeddings of questions and answers as input that cannot be fine-tuned to the task in hand. GPPL also ignores any model uncertainty in the embeddings themselves, which may limit the ability of BO to identify suitable candidates for comparison. We therefore investigate techniques that allow deep neural networks (DNNs), which are powerful representation learners, to be used in their place. Standard DNNs cannot be used as ranking models for BO, as they do not account for any model uncertainty: they output point estimates of the utility of a candidate, whereas a Bayesian approach will provide a posterior distribution, including the posterior variance, which quantifies uncertainty in the prediction. Furthermore, in classification tasks, the predictive probabilities that DNNs output are often not well-calibrated, especially when generalising outside the training distribution (Guo et al., 2017). We therefore turn to *Bayesian deep learning (BDL)*, which measures model uncertainty, and improves calibration and generalisation, thereby enabling BO while keeping the representation learning ability of neural networks (Maddox et al., 2019).

In this work, we propose BDL methods with interactive preference learning for non-factoid answer selection (select the appropriate answer from a list of candidate answers) and summary ranking (rank candidate summaries by quality). An illustration of the cQA task is shown in Table 1. Our approach leverages pretrained models to embed text and can be fine-tuned end-to-end with user feedback, but is able to provide not just a prediction of the utility score for each candidate output, but also an estimate of the model's uncertainty, represented by its posterior variance.

Experiments using simulated noisy users on an English cQA dataset (Rücklé et al., 2019) show that BDL outperforms the shallow GPPL and non-Bayesian DNN with only 4 interactions, achieving on average a 15% improvement in accuracy over the non-Bayesian method. Results for extractive multi-document English news summarisation corroborate these results, with BDL outperforming the non-Bayesian approach by 10-12% with 6 interactions. We also show that our Bayesian approach is robust to noise in the user feedback and make a preliminary comparison of two BDL techniques on cQA, Monte Carlo Dropout (MCD) (Gal and Ghahramani, 2016) and stochastic weight averaging Gaussian (SWAG) (Maddox et al., 2019), finding that MCD performs best with limited computational costs. This work highlights the potential of BO for adapting NLP models to individual users or novel tasks without the need to collect large amounts of human feedback, and demonstrates the benefits of modelling epistemic uncertainty in NLP. Please find our code available at `https://github.com/edwinrobots/BayesianOpt_uncertaiNLP2024`.

2 Related Work

Recent large language models (LLMs) have been trained to follow user instructions by acquiring human preference feedback, training a ranking model, and using the ranker as a reward function for reinforcement learning (Ouyang et al., 2022). This process, *reinforcement learning from human feedback (RLHF)*, is a powerful example of using preference learning to optimise a latent objective function, but the prior work does not discuss how to acquire the labels efficiently. We address this by investigating a BO method with much smaller NLP models, which could be applied to RLHF in future to reduce the data acquisition cost for fine-tuning LLMs.

Many previous works on interactive learning, such as P.V.S and Meyer (2017), Lin and Parikh (2017) and Peris and Casacuberta (2018) use uncertainty sampling to select unlabelled data points to query users for labels, but measure uncertainty using conventional DNNs, which are miscalibrated and overconfident (Guo et al., 2017), or measure only predictive rather than model uncertainty (Ein-

Dor et al., 2020). Siddhant and Lipton (2018) conducted a large empirical study of deep active learning across multiple tasks, showing deep Bayesian active learning significantly outperforms classical uncertainty sampling. For text ranking, Simpson et al. (2020) replaced uncertainty sampling with BO to find the best solution from a pool of candidates, achieving state-of-the-art performance in both interactive cQA and summarisation. However, their shallow GPPL ranker cannot fine-tune the input embeddings nor quantify the model uncertainty in the embeddings.

3 Background

BO with Expected Improvement (EI) BO aims at finding the maximum or minimum of a function. For text ranking tasks, this function maps an input text, x, to a score, $f(x)$, called the *utility*. BO uses an acquisition function to decide which input the user should evaluate next, as part of an iterative process of gathering pairwise preference labels. In effect, BO is an active learning process that focuses on finding the extremum, rather than learning the function across the whole input space. It is therefore suited to NLP tasks where the model must learn how to produce the best output for a user, e.g., the most suitable answer to a question or the most fitting summary for a topic.

Here, we adopt EI as the acquisition function (Močkus, 1975), which was previously shown to outperform other acquisition functions for cQA and MDS (Simpson et al., 2020). To compute EI, we require a ranking model that outputs not a point prediction of the utility of each candidate, $f(x)$, but a posterior distribution over $f(x)$. For a set of candidates, $\boldsymbol{x}$, we assume a Gaussian posterior distribution over their utilities, $\mathcal{N}(\boldsymbol{\mu}(\boldsymbol{x}), \mathbf{C})$, where $\boldsymbol{\mu}(\boldsymbol{x})$ is the posterior mean vector and $\mathbf{C}$ is the posterior covariance. Within our set of candidates, $\boldsymbol{x}$, we can find the current best candidate, x^*, according to the model's current posterior distribution, by finding the candidate with the highest posterior mean, $\mu^* = max\{\mu(x) \in \boldsymbol{\mu}(\boldsymbol{x}))\}$. EI compares the posterior distribution for each candidate text, x, to that of the current best candidate, x^*, and determines which x has the most potential to improve over x^*. To do this, EI considers the probability that $f(x)$ is higher than $f(x^*)$, and by how much. The process for computing EI is as follows:

1. Obtain the posterior means and covariances from the model.
2. Identify the current best candidate, x^*, as described above.
3. For each candidate x, the difference in utility to the current best candidate, $(f(x) - f^*)$ has a Gaussian posterior distribution. Compute the posterior variance v of $(f(x) - f^*)$, $v = \boldsymbol{C}_{x,x} + \boldsymbol{C}_{x^*,x^*} - 2\boldsymbol{C}_{x,x^*}$, where $\boldsymbol{C}_{a,b}$ is the element of $\boldsymbol{C}$ at the row corresponding to text a and column corresponding to text b.
4. Compute the difference between the posterior means for x and x^*, normalised by its posterior standard deviation, $\sqrt{v}$, $z = \frac{\mu(x)-\mu^*}{\sqrt{v}}$.
5. Compute EI as follows:

$$a_{EI}(x) = \sqrt{v}z\Phi(z) + \sqrt{v}\mathcal{N}(z; 0, 1), \quad (1)$$

where a_{EI} is the EI acquisition function, and $\Phi()$ is the cumulative density function of a standard Gaussian distribution. The terms involving z give higher scores to candidates with a high expected utility, $\mu(x)$, and terms involving v give more weight to candidates with uncertain utilities. As part of an iterative active learning process, the candidate x with highest EI is selected, and the user is asked to compare this candidate to the current best, x^*, to provide a new pairwise training label. EI therefore trades off *exploitation* of known good candidates and *exploration* of uncertain candidates.

Monte Carlo Dropout EI requires us to estimate posteriors over utilities, but standard DNN inference outputs point estimates of $f(x)$ rather than distributions. Gal and Ghahramani (2016) proposed Monte Carlo Dropout (MCD), a computationally efficient approximate Bayesian inference method that applies dropout at inference time to obtain a set of samples of $f(x)$. MCD samples T times from a variational posterior distribution over model weights as follows. For each weight j in the ith layer, sample $z_{i,j} \sim Bernouilli(p_i)$ to determine whether dropout is applied to that weight. Then compute $W_i = M_i \cdot \text{diag}(z_i)$, where M_i represents the weight matrix before dropout and W_i is a sample of weights with dropout applied. We use each sample, W_i, to predict the utilities for all candidates, $f(x) \forall x \in \boldsymbol{x}$, thereby generating samples of utilities with potentially different values for each candidate x. We then compute the empirical mean and covariance of these sample utilities to estimate the posterior distribution over the utilities.

SWAG Another variational inference method, SWAG can provide a better approximation to the posterior than MCD (Maddox et al., 2019). It calculates the first two moments (mean and covariance) of an approximate Gaussian distribution over the weights using SGD iterates. To estimate the mean of the Gaussian, it adopts SWA (Izmailov et al., 2018), which averages the weights at selected iterations of SGD. SWAG approximates the covariance by summing a diagonal covariance and a low-rank covariance term, also computed from the sampled weights of the network at chosen iterations. To estimate the posterior over the utilities, sample weights from the Gaussian weight posterior, predict the utilities with each sample of weights, then compute the empirical mean and covariance of the predicted utilities.

4 Interactive Learning Process

Figure 1 provides an overview of the interactive learning process studied in this paper. For a given input (e.g., a question for QA tasks; a set of source documents for summarisation) and multiple candidate outputs, the model needs to return the best matched output to a user after several rounds of interaction. During each interaction, the query strategy selects a pair of candidates for the user to compare, based on the acquisition function. Once the user's feedback is obtained, it is added into the training set to train the ranking model. The process will be repeated a predefined number of times.

Learning the ranking model from scratch would cause a cold start problem, where we have no information to guide the query process and may require a lot of user interactions to learn a reasonable model. We avoid this by introducing a warm-start phase, which pretrains the ranking model using in-domain data for different inputs before the interactive learning process begins. This means that the ranking model learns a general-purpose ranking function in the warm-start phase, which is then fine-tuned to a specific topic or user through the interactive learning phase depicted in Figure 1.

5 Proposed Bayesian Ranking Model

Figure 2 shows the architecture of the deep ranker used as a surrogate model in the interactive learning process. In our experiments, we use this same architecture for both Bayesian and non-Bayesian deep rankers. It consists of a pretrained encoder, two fully connected layers and an output layer. We train

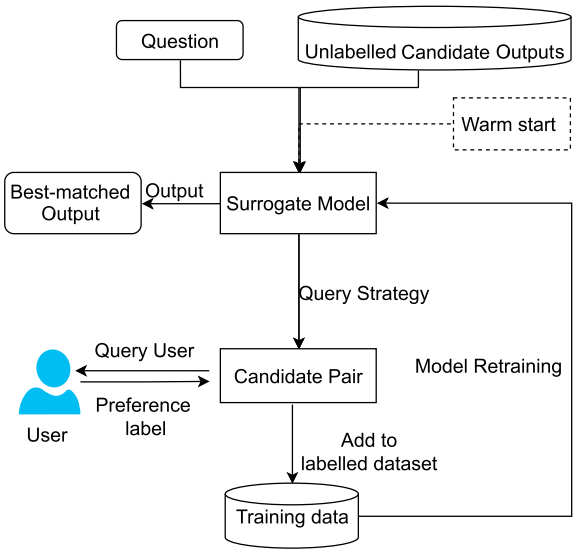

Figure 1: Workflow of our Proposed Approach

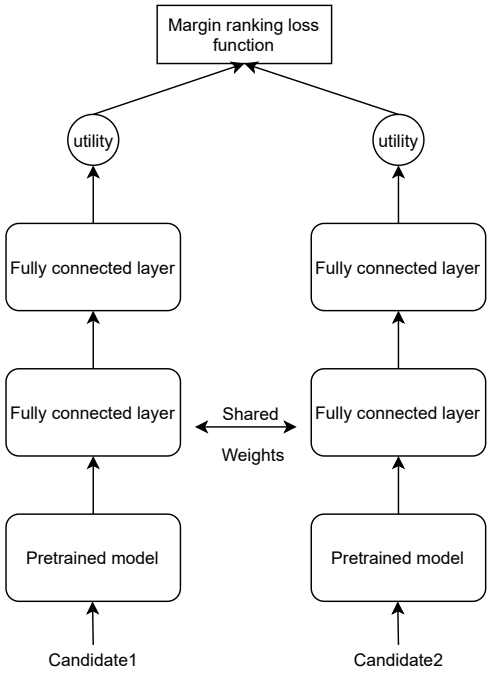

Figure 2: Architecture of the proposed ranking model

with pairwise labels, with all weights shared between candidates 1 and 2 in each pair (as a Siamese architecture), using margin ranking loss:

$$L(f_1, f_2, y) = \max(0, -y(f_1 - f_2 + m)), \quad (2)$$

where f_1 and f_2 represent predicted utilities of two input texts, $y \in \{-1, 1\}$, indicates which input should be ranked higher, and $y = 1$ means candidate 1 ranks higher. During inference, each candidate is processed individually to predict its utility, $f(x)$. For the Bayesian variants of the deep ranker, we use MCD and SWAG to approximate posteriors

over utilities. For prior distributions over the neural network weights, recent experiments provide strong evidence that vague Gaussian priors can induce useful inductive bias (Dmitry et al., 2020). Therefore, we add L2 regularisation to the loss function since it can be interpreted as a Gaussian prior.

6 Experiments

6.1 Experimental Setup

cQA Datasets We conduct experiments on an English cQA dataset consisting of questions posted on StackExchange in the communities Apple, Cooking and Travel (Rücklé et al., 2019). For a given question, there is one accepted answer marked by the user and 99 candidate answers collected from answers to similar questions. For the questions, the dataset retains only the title and discards the detailed description in the question body. For the encoder, we use *distilRoBERTa* (Liu et al., 2019).

To train the initial model in the warm start phase, we use the original training data and tune hyperparameters on part of the original validation set. The hyperparameters used for fine-tuning in the interaction phase are tuned separately, on the remaining portion of the validation set (Table 2). For the interaction phase, we use the original test set (Table 3). The experiments simulate a setting where the user provides labels to fine-tune a model for the test question only. In other words, the cQA system helps the user narrow their search for the right answer with the help of user feedback. In this setting, the model sees pairwise labels for a small subset of test answers. Our experimental setup therefore compares active learning methods that choose which pairwise comparisons the model gets to see, but does not test the general performance of the resulting cQA model on other questions.

Topics	Train	Warm start validation	Interaction phase validation
Apple	5,831	1,249	831
Cooking	3,692	791	692
Travel	3,572	765	572

Table 2: Number of cQA questions in the datasets for training and hyperparameter tuning.

Topics	#questions	#accepted answers	#candidate answers	#cand. per topic
Apple	1,250	1,250	125,000	100
Cooking	792	792	79,200	100
Travel	766	766	76,600	100

Table 3: Statistics for the test dataset used in the cQA interaction phase.

MDS Datasets For multi-document summarisation, we use the DUC datasets[1]. There are three DUC datasets, i.e., DUC'01, DUC'02, DUC'04, containing a number of news topics. Each topic has three distinct model summaries, each of which was penned by a different expert, offering a varied perspective on what makes an effective summary. This multi-document summarisation approach poses a challenge, especially given the diverse themes within a document collection, as it is not straightforward to pinpoint a singular, succinct summary that would cater to all users.

With MDS, we use *SUPERT* (Gao et al., 2020) as the encoder, as also used by Simpson et al. (2020). For each topic in the dataset, we followed the approach of Gao et al. (2018) and generated 10,000 candidate summaries, each with no more than 100 words, by randomly selecting sentences from the source documents, to enable comparison with their prior work. The 10,000 summaries for each topic are then split into train (6,000 summaries), validation (2,000) and test sets (2,000), hence all topics appear in every split. The validation set is used to tune hyperparameters in both the warm-start and interaction phases (refer to Table 4 and Table 5).

As in the cQA task, the goal of the interaction is to fine-tune the model for a specific topic, rather than to learn a user's preferences over summaries in general. Hence, the pairwise labels obtained in the interaction phase compare test examples, and the experiment aims to compare methods for selecting these examples, rather than learning a model that can generalise to other topics.

Dataset	Train	Validation
DUC'01	180,000	60,000
DUC'02	354,000	118,000
DUC'04	300,000	100,000

Table 4: Number of MDS summaries in the datasets for training and hyperparameter tuning.

[1] https://duc.nist.gov/

Data -set	#top -ics	#source docs	#model summ -aries	#cand. summ -aries	#cand. per topic
DUC'01	30	300	90	300000	2,000
DUC'02	59	567	177	590000	2,000
DUC'04	50	500	150	500000	2,000

Table 5: Statistics for the test dataset used in the MDS interaction phase.

Simulated Users Here, we follow previous work (Simpson et al., 2020; Gao et al., 2019) to simulate user preferences with the user-response model (Viappiani and Boutilier, 2010). Given utilities of two candidates, f_a and f_b, the simulated user will prefer document a with probability:

$$p(y_{a,b}|f_a, f_b) = \frac{1}{1 + \exp(f_a - f_b)/t}, \quad (3)$$

where t controls the noise in the user's preferences. We set the default value of t to 0.3, as per Simpson et al. (2020) and investigate its effect in Section 6.4. For cQA, we estimate the utilities f_a and f_b with ROUGE-L, which calculates the longest common sub-sequence between candidates and gold answers. For MDS, we use a combination of ROUGE_1, ROUGE_2, and ROUGE_{SU4} that showed high correlation with human preferences in previous summarisation work (P.V.S and Meyer, 2017).

Simulated users allow us to test the proposed method more rapidly at a greater scale. However, the simulation assumes a consistent latent preference function, from which we observe noisy preferences. It is possible that real human feedback may sometimes violate these assumptions, so we view our experiments as an initial exploration to establish whether further experimentation with real users is warranted.

Evaluation Metrics For cQA, we compute *matching accuracy*, which is the fraction of top-ranked answers that match the gold answers. To evaluate the first few highest-ranked answers, we use *normalized discounted cumulative gain at 5 (NDCG@5)*, which uses ROUGE-L as the relevance score to compare the top five candidates to the gold answer. For MDS, NDCG@100 is used to compare the top 1% with the reference summary. The combined ROUGE score for simulating users is adopted here as the relevance metric. We do not use ranking metrics that evaluate the entire candidate ranking, since the goal of this application is to find the most appropriate top-ranked items only – we do not care about the order of unsuitable outputs.

Hyperparameters For cQA, we set the margin m (Equation 2) to 0.1 and tuned hyperparameters at both the warm start and interaction phases (Appendix A). As shown in Table 6, doubling the number of MCD samples led to a small improvement in accuracy, while doubling computation time. Given limited compute time in interactive settings, we fixed the number of samples to 20.

SWAG requires sufficient epochs before starting sampling to approximate the posterior accurately. Therefore, maintaining the same computation time for SWAG as MCD limited us to using only three epochs to compute the weight posteriors in only the last four layers since the entire cQA ranking model has over 82M parameters. With the non-Bayesian method, the time per round of interactive learning was ~2 seconds.

For the MDS task, we observed a similar pattern with increasing numbers of MCD samples and fixed this value to 30. After tuning, the margin loss for MDS was set to 0.5 and other hyperparameters are given in Appendix A.

Baselines We compared our models with a non-Bayesian deep ranker and GPPL (using the implementation of Simpson et al. (2020)). BO cannot be used for the non-Bayesian ranker because it does not compute the variance of the utility, so we use an established uncertainty sampling approach (UNC) (P.V.S and Meyer, 2017). For each candidate, a, we compute $unc(a) = 0.5 - |p(a) - 0.5|$, where $p(a) = (1 + exp(-f_a))^{-1}$ is the probability that a is accepted by the user and f_a is the predicted utility of a. We query the candidate pair (a, b) with the highest uncertainty values, $unc(a)$ and $unc(b)$.

#Samples	10	20	30	40
cQA Accuracy	0.828	0.836	0.838	0.841
Time per interaction (s)	7	14	21	28
MDS NDCG@1%	0.684	0.665	0.641	0.675
Time per interaction (s)	4	8	12	16

Table 6: The accuracy and computation times versus the number of MCD samples on the interaction phase validation sets. Computation times are for training with a single NVIDIA RTX2080Ti GPU.

6.2 Results: Performance Comparison

On the cQA task, Table 7 shows that Bayesian deep ranker outperforms the non-Bayesian deep

Model	Query Strategy	Cooking Acc	Cooking NDCG@5	Apple Acc	Apple NDCG@5	Travel Acc	Travel NDCG@5
Non-Bayesian deep ranker	UNC	0.685	0.683	0.417	0.614	0.686	0.634
GPPL	EI	0.573	0.644	0.465	0.647	0.702	0.691
Bayesian deep ranker with SWAG	EI	0.692	0.67	0.540	0.637	0.713	0.668
Bayesian deep ranker with MCD	EI	**0.726**	0.612	**0.650**	0.616	**0.863**	0.675

Table 7: Results of different interactive ranking methods for cQA with 4 interactions.

Model	Query Strat.	#inter-actions	DUC '01	DUC '02	DUC '04
Non-Bayesian deep ranker	UNC	6	0.524	0.551	0.579
GPPL	EI	20	0.624	0.630	0.653
Bayesian deep ranker with MCD	EI	6	**0.637**	**0.661**	**0.681**

Table 8: MDS results, showing NDCG@1%. The results for GPPL were obtained from Simpson et al. (2020), which uses the same experimental setup.

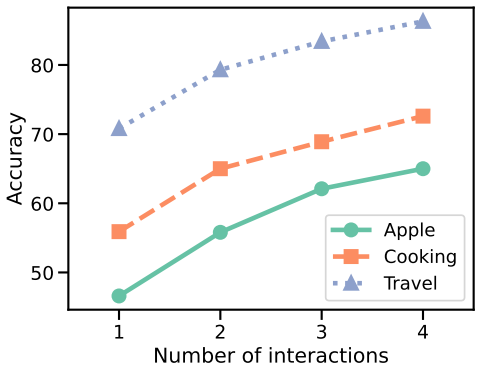

Figure 3: The performance change on cQA topics in relation to the number of interactions.

ranker and GPPL in terms of top-1 matching accuracy. For the topic *Apple*, the accuracy of the deep ranker with MCD is 23% higher than that of the non-Bayesian deep ranker. This gain comes from the ability to quantify uncertainty in the model weights due to a lack of knowledge as *posterior variance* in the utilities. This enables us to apply BO to find the optimal candidate as quickly as possible. In contrast, the non-Bayesian ranker does not quantify model uncertainty: rather than outputting a distribution over utility, it simply provides a point estimate, its "best guess". Since it lacks information about the model's epistemic uncertainty, the non-Bayesian query strategy cannot select pairs on this basis. Instead, it relies on a heuristic, whereby it selects predicted utilities close to zero, assuming that these candidate answers have a probability close to 0.5 of being accepted by the user, and hence the highest uncertainty. The issue is that many of these candidates could be of middling quality, rather than simply of uncertain utility, so labelling effort could be wasted in selecting such candidates. The shallow GPPL method also outperforms the non-Bayesian deep ranker with UNC by 3.3% and 5.7% (Accuracy) under the topic Apple and Travel.

As shown in Table 7, MCD outperforms SWAG under our computation time constraints. For SWAG, the weight covariance obtained from just three samples is relatively small, so the posterior tends to be sharply peaked. Moreover, with SWAG we only update the posteriors for the last four layers, while MCD is applied to all layers. The variance is thereby underestimated, impeding EI from exploring new points. Therefore, when computation time is very limited, MCD appears more effective.

For the MDS task, Table 8 demonstrates the Bayesian deep ranker again out-performs the non-Bayesian deep ranker with UNC sampling method, and is able to outperform GPPL despite using far fewer simulated user interactions.

6.3 The Impact of number of interactions

We investigated how the performance changes in relation to the number of interactions for the cQA task. We varied the number of simulated human interactions from 1 to 4 and conducted experiments with the MCD-based Bayesian ranker. As visualised in Figure 3, we observe that as the number of interactions increases, the performance improves across all topics. For questions under the topic Apple and Travel, with only one interaction, the MCD-based Bayesian deep ranker outperforms both the Non-Bayesian deep ranker and shallow Bayesian ranker with GPPL which adopts 10 interactions.

6.4 Noisy User Experiment

To test the robustness of our proposed models, we vary the noise level t in the simulated labels (Equation (3)) and observe its effect on the Bayesian deep ranker with MCD, examining accuracy for the cQA topic *Travel* and NDCG@1% for the DUC'01 topics. Table 9 shows that as noise increases, the matching accuracy decreases, but does not drop substantially when the noise parameter t rises from 0.3 to 2.5. This indicates that the Bayesian deep ranker with MCD is robust under noisy circumstances.

cQA, *Travel*: Noise Level	0.3	1	2.5
Accuracy	0.833	0.828	0.795
MDS, *DUC'01*: Noise Level	0.5	1	2.14
NDCG@1%	0.643	0.632	0.618

Table 9: Effect of simulated user noise on the Bayesian deep ranker with MCD.

7 Conclusion

We proposed a Bayesian approach to learning from human feedback in the form of pairwise preferences, which combines pretrained language models with end-to-end Bayesian deep learning to gauge uncertainty in the model's predictions. We showed how this approach enables a Bayesian optimisation strategy for active learning, which selects pairs to be labelled by the user to find the most appropriate solution with only a few round of user interaction. We applied our method to community question answering and multi-document summarisation, but the method can be generalised to any interactive ranking task where the aim is to identify strong candidate outputs for a given input. Our experiments showed the Bayesian deep learning can outperform a non-Bayesian deep ranker and a shallow Bayesian method in an active preference learning setting, and performs well when there is a high level of noise in the user feedback. We also showed when approximating posterior distributions under tight compute time constraints, MCD can outperform SWAG, although further investigation is needed to evaluate SWAG under different conditions and on other tasks.

There is a wealth of other tasks that this approach could be evaluated on, including for fine-tuning large language models with human feedback (Ouyang et al., 2022), which we plan to investigate in future work. The experiments in this paper used interaction to learn models specialised to particular questions or summary topics, so the application of Bayesian optimisation to learning general-purpose models is yet to be explored. A key benefit that merits further research is that more efficient sampling of training data for the reward model in RLHF could help to make fine-tuning language models more accessible to smaller organisations by reducing annotation costs.

8 Limitations

Although we demonstrate that Bayesian deep learning-based preference learning can efficiently acquire human feedback for text ranking tasks, i.e., cQA and multi-document summarisation, there are several limitations that call for future research. The primary limitation of our work is that we use simulated users to approximate human preferences. In future work, we aim to evaluate the approach with real users to determine whether the labelling efficiency we found with simulated users is observed with human labellers.

The experimental setup was constrained to learning models for a specific question or summary topic. As such, the pairwise feedback was obtained for examples in the test set. The experimental results are therefore not a reflection of how well the models generalise to new questions or news topics, nor how well the interactive learning method helped the models' question answering or summarisation performance in general, as this was not within the scope of this paper. This is a limitation that we plan to address in future work on BO for personalising and fine-tuning more versatile models.

Considering the technical limitations of BO, there is a time cost to the sampling steps in Bayesian inference, which could be sped up in future implementations by using parallel sampling from posteriors. Our investigation into SWAG was highly constrained by computational resources, and should be seen as a preliminary investigation only – further work is needeed to investigate alternative BDL methods in comparison with SWAG and MCD. A promising approach is Bayesian Layers (Tran et al., 2019), which offers more efficient Bayesian inference over larger transformer models.

9 Ethical Considerations

There is a risk with automatic answer selection and recommender systems that the content directed can have undesirable effects, for instance if the user

receives conspiracies or propaganda as answers to a question. This may happen unintentionally when user goals diverge from system goals, such as distributing advertisements. To some extent, interactive systems, such as that proposed here, hand more control to users and could reduce these issues. Nonetheless, further investigation is needed to determine the effect of interactive cQA and MDS systems on the answers and summaries that users get to see, to determine how this biases their content consumption or to identify other negative consequences.

Our experiments evaluate the method on English data as an initial investigation into the potential for BDL in answer or summaries selection tasks. The proposed method can be applied to other languages and tasks, but will require further evaluation to determine whether users can provide suitable labels in such domains, how many interactions are needed for the chosen language, and whether the mode of interaction is equally accessible to users of different backgrounds.

References

Ralph Allan Bradley and Milton E. Terry. 1952. Rank analysis of incomplete block designs: I. The method of paired comparisons. *Biometrika*, 39(3/4):324–345.

Yang Deng, Wai Lam, Yuexiang Xie, Daoyuan Chen, Yaliang Li, Min Yang, and Ying Shen. 2020. Joint learning of answer selection and answer summary generation in community question answering. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 7651–7658.

Ulyanov Dmitry, Andrea Vedaldi, and Lempitsky Victor. 2020. Deep image prior. *International Journal of Computer Vision*, 128(7):1867–1888.

Liat Ein-Dor, Alon Halfon, Ariel Gera, Eyal Shnarch, Lena Dankin, Leshem Choshen, Marina Danilevsky, Ranit Aharonov, Yoav Katz, and Noam Slonim. 2020. Active Learning for BERT: An Empirical Study. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 7949–7962, Online. Association for Computational Linguistics.

Yarin Gal and Zoubin Ghahramani. 2016. Dropout as a Bayesian approximation: Representing model uncertainty in deep learning. In *international conference on machine learning*, pages 1050–1059. PMLR.

Yang Gao, Christian M. Meyer, and Iryna Gurevych. 2018. APRIL: Interactively learning to summarise by combining active preference learning and reinforcement learning. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 4120–4130, Brussels, Belgium. Association for Computational Linguistics.

Yang Gao, Christian M Meyer, and Iryna Gurevych. 2019. Preference-based interactive multi-document summarisation. *Information Retrieval Journal*, pages 1–31.

Yang Gao, Wei Zhao, and Steffen Eger. 2020. SUPERT: Towards new frontiers in unsupervised evaluation metrics for multi-document summarization. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 1347–1354, Online. Association for Computational Linguistics.

Chuan Guo, Geoff Pleiss, Yu Sun, and Kilian Q Weinberger. 2017. On calibration of modern neural networks. In *International Conference on Machine Learning*, pages 1321–1330. PMLR.

Pavel Izmailov, Dmitrii Podoprikhin, Timur Garipov, Dmitry Vetrov, and Andrew Gordon Wilson. 2018. Averaging weights leads to wider optima and better generalization. In *Conference on Uncertainty in Artificial Intelligence*.

Xiao Lin and Devi Parikh. 2017. Active learning for visual question answering: An empirical study. *arXiv preprint arXiv:1711.01732*.

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

Manuel J Maña López, Manuel de Buenaga Rodríguez, and José María Gómez Hidalgo. 1999. Using and evaluating user directed summaries to improve information access. In *International Conference on Theory and Practice of Digital Libraries*, pages 198–214. Springer.

R. Duncan Luce. 1959. On the possible psychophysical laws. *Psychological Review*, 66(2):81–95.

Wesley J Maddox, Pavel Izmailov, Timur Garipov, Dmitry P Vetrov, and Andrew Gordon Wilson. 2019. A simple baseline for Bayesian uncertainty in deep learning. *Advances in Neural Information Processing Systems*, 32:13153–13164.

Jonas Močkus. 1975. On Bayesian methods for seeking the extremum. In *Optimization techniques IFIP technical conference*, pages 400–404. Springer.

Frederick Mosteller. 1951. Remarks on the method of paired comparisons: I. The least squares solution assuming equal standard deviations and equal correlations. *Psychometrika*, 16(1):3–9.

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. 2022. Training language models to follow instructions with human feedback. *Advances in Neural Information Processing Systems*, 35:27730–27744.

Álvaro Peris and Francisco Casacuberta. 2018. Active learning for interactive neural machine translation of data streams. In *Proceedings of the 22nd Conference on Computational Natural Language Learning*, pages 151–160, Brussels, Belgium. Association for Computational Linguistics.

R. L. Plackett. 1975. The analysis of permutations. *Journal of the Royal Statistical Society, Series C (Applied Statistics)*, 24(2):193–202.

Avinesh P.V.S and Christian M. Meyer. 2017. Joint optimization of user-desired content in multi-document summaries by learning from user feedback. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1353–1363, Vancouver, Canada. Association for Computational Linguistics.

Andreas Rücklé, Nafise Sadat Moosavi, and Iryna Gurevych. 2019. Coala: A neural coverage-based approach for long answer selection with small data. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 6932–6939.

Aditya Siddhant and Zachary C. Lipton. 2018. Deep Bayesian active learning for natural language processing: Results of a large-scale empirical study. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2904–2909, Brussels, Belgium. Association for Computational Linguistics.

Edwin Simpson, Yang Gao, and Iryna Gurevych. 2020. Interactive text ranking with Bayesian optimization: A case study on community QA and summarization. *Transactions of the Association for Computational Linguistics*, 8:759–775.

Louis L Thurstone. 1927. A law of comparative judgment. *Psychological review*, 34(4):273.

Dustin Tran, Mike Dusenberry, Mark Van Der Wilk, and Danijar Hafner. 2019. Bayesian layers: A module for neural network uncertainty. *Advances in neural information processing systems*, 32.

Quan Hung Tran, Duc-Vu Tran, Tu Vu, Minh Le Nguyen, and Son Bao Pham. 2015. Jaist: Combining multiple features for answer selection in community question answering. In *Proceedings of the 9th International Workshop on Semantic Evaluation (SemEval 2015)*, pages 215–219.

Paolo Viappiani and Craig Boutilier. 2010. Optimal bayesian recommendation sets and myopically optimal choice query sets. In *Advances in Neural Information Processing Systems*, pages 2352–2360.

Yi-Hsuan Yang and Homer H. Chen. 2011. Ranking-based emotion recognition for music organization and retrieval. *IEEE Transactions on Audio, Speech, and Language Processing*, 19(4):762–774.

A Hyperparameter Tuning for cQA

We fixed the hidden layer sizes of our model as shown in Table 10, and did not tune them. At the warm start stage, we use the AdamW optimizer for the deep ranker which includes implicit L2 regularization using weight decay. For the SWAG-based model, we use SGD with momentum as the optimizer since it will update the parameters of posterior distributions along the SGD trajectory.

Layer name	# Hidden size
DistilRoBERTa	original size, 768
Fully-connected layer 1	100
Fully-connected layer 2	10

Table 10: Hidden layer sizes in the cQA model

For the warm-start phase, we empirically set the search space of learning rate $\in \{2e-5, 5e-5\}$ for conventional deep learning, $\{1e-4, 5e-5\}$ for the SWAG-based model, batch size $\in \{16, 32\}$ and weight decay $\in \{0.01, 0.001\}$. We exhaustively searched these combinations to find the optimal combination and the selected values are shown in Table 11. The number of training epochs during warm-start was fixed to 3 for the non-Bayesian deep ranker and the deep ranker with MCD, and 6 epochs for the SWAG-based ranker. The Dropout rate was the default value, 0.1.

At the interaction phase, all models were trained with SGD with momentum and the number of epochs was fixed to 4 with early stopping. We tuned learning rate $\in \{1e-4, 5e-5\}$ and weight decay $\in \{0.01, 0.001\}$. We did not tune batch size as we fine-tune only with the 4 data points obtained from the simulated user. The selected values are shown in Table 12.

B Hyperparameter Tuning for MDS

For summarisation, Table 13 shows the hidden layer sizes of our model. At the warm start stage, we again used AdamW optimizer.

Topic/Dataset	Model	Learning rate	Batch size	Weight decay
Cooking	Non-Bayesian DR	5e-5	16	0.01
Cooking	DR + MCD	5e-5	16	0.01
Cooking	DR + SWAG	1e-4	32	0.001
Apple	Non-Bayesian DR	2e-5	32	0.001
Apple	DR + MCD	2e-5	32	0.001
Apple	DR + SWAG	1e-4	16	0.001
Travel	Non-Bayesian DR	2e-5	16	0.01
Travel	DR + MCD	2e-5	16	0.01
Travel	DR + SWAG	1e-4	16	0.01

Table 11: Hyperparameters selected at the warm-start phase for each cQA topic.

Topic/Dataset	Model	Learning rate	Weight decay
Cooking	Non-Bayesian DR	5e-5	0.001
Cooking	DR + MCD	1e-4	0.01
Cooking	DR + SWAG	1e-4	0.001
Apple	Non-Bayesian DR	1e-4	0.01
Apple	DR + MCD	1e-4	0.001
Apple	DR + SWAG	1e-4	0.001
Travel	Non-Bayesian DR	1e-4	0.01
Travel	DR + MCD	1e-4	0.01
Travel	DR + SWAG	5e-5	0.001

Table 12: Hyperparameters selected for the interaction phase for each cQA topic.

Layer name	# Hidden size
SUPERT	original size, 1024
Fully-connected layer 1	256
Fuly-connected layer 2	128

Table 13: Hidden layer sizes in the MDS model

Hyperparameters for the warm-start phase were tuned, considering learning rate $\in \{1e-4, 2e-5, 5e-5\}$ and weight decay $\in \{0.01, 0.001\}$. For all MDS models, we found the best choice to be learning rate = 5e-5 and weight decay = 0.001.

For the interaction phase, the learning rate was tuned $\in \{1e-4, 5e-5\}$ and weight decay $\in \{0.01, 0.001\}$, finding optimal values of learning rate = 1e-4 for all cases, and weight decay = 0.001 for all cases except the non-Bayesian deep ranker on DUC'02 and DUC'04, which used weight decay = 0.01.