Bayesian Deep Learning for Interactive Community Question Answering

Anonymous ACL submission

Abstract

Human-in-the-loop interactive learning has been shown to be effective for best solution selection tasks. Bayesian Optimisation (BO) reduces the amount of user interaction required but has so far relied on shallow models rather than end-to-end deep learning. This paper leverages recent advances in Bayesian deep learning (BDL) to more accurately identify the best solution from a few rounds of interaction. We apply our approach to community question answering (cQA), finding that our BDL approach significantly outperforms existing methods while remaining robust to noise in the user feedback.

1 Introduction

Selecting the correct, multi-sentence answer to a complex question poses a challenge for cQA systems (Tran et al., 2015; Rücklé et al., 2019; Deng et al., 2020). They can be assisted by human-in-the-loop interactive learning, which presents the user with pairs of candidate answers, and asks them to select the most appropriate answer in each pair (Simpson et al., 2020). This kind of pairwise labelling (Thurstone, 1927) reduces the user's cognitive burden compared with rating candidates (Yang and Chen, 2011). To minimise the number of interactions required, Simpson et al. (2020) use BO (Močkus, 1975), which actively chooses promising candidates for the user to compare. However, their model uses Gaussian process preference learning (GPPL), which relies on fixed embeddings of questions and answers as input that cannot be fine-tuned to the task in hand. Besides, the Bayesian approach relies on estimating the *epistemic* or *model uncertainty* due to the inability to determine exact values for the model's parameters from a finite training set. However, GPPL ignores model uncertainty in the question and answer representations.

While deep neural networks (DNNs) are powerful representation learners, standard DNN probabilities are not well-calibrated, especially when generalising out of training distribution (Guo et al., 2017), and do not account for model uncertainty. We therefore turn to BDL, which measures model uncertainty, and improves calibration and generalisation while keeping the representation learning ability of neural networks (Maddox et al., 2019).

In this work, we propose a BDL method with interactive preference learning for non-factoid answer selection. Our approach leverages pretrained models to embed text and can be fine-tuned end-to-end with user feedback. An example output is shown in Table 1. Experiments 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. We also show that the BDL approach is robust to noise in the user feedback and compare two recent BDL approaches, 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. We will make all code publicly available on acceptance.

2 Related Work

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 to query users for labels, but measure uncertainty using conventional DNNs, which are miscalibrated and overconfident (Guo et al., 2017), or measure 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, achiev-

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. Only A3 is correct.

ing state-of-the-art performance in both interactive cQA and summarisation. However, their shallow GPPL ranker cannot fine-tune the 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 should be evaluated next. In effect, BO is an active learning process that focuses on finding the extremum, rather than learning the function.

Here, we adopt EI as the acquisition function, which outperformed other acquisition functions for answer selection (Simpson et al., 2020). To compute EI, first estimate the posterior distribution over the utility of each candidate, $\mathcal{N}(\mu(x), \mathbf{C})$. Then, find the current best candidate $\mu^* = max\{\mu(x)\}$. For each candidate x, use $\mathbf{C}$ to compute the variance v of $(f(x) - f^*)$ and the normalised difference, $z = \frac{\mu(x)-\mu^*}{\sqrt{v}}$, then use this to compute EI:

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

EI selects candidates with both a high expected utility, $\mu(x)$, and high uncertainty, v, and therefore trades off exploitation and exploration.

Monte Carlo Dropout EI requires us to estimate posteriors over utilities, but standard DNN inference outputs point estimates rather than distributions. Gal and Ghahramani (2016) proposed Monte Carlo Dropout (MCD), a computationally efficient approximate Bayesian inference method. 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, thereby generating samples of utilities. We then compute the empirical mean and variance 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 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 SGD iterations. SWAG approximates the covariance by summing a diagonal covariance and a low-rank covariance term, also computed from the sampled weights. To estimate the posterior over the utilities, sample from the Gaussian weight posterior, predict the utilities with each sample of weights, then compute the empirical mean and variance of the predicted utilities.

4 Problem Definition

As shown in Figure 1, for a given question and multiple candidate answers, the model needs to return the best matched answer to a user after several rounds of interaction. During each interaction, the query strategy selects a pair of candidates for the user to compare. Once the user's feedback is obtained, it is added into the training set to train the surrogate model. The process will be repeated a predefined number of times. To solve the cold start problem, a warm start phase is introduced to pretrain the model using in-domain data.

5 Proposed Bayesian Surrogate Model

The architecture (Appendix A) consists of a pretrained encoder (distilRoBERTa (Liu et al., 2019)), two fully connected layers and one output layer. A

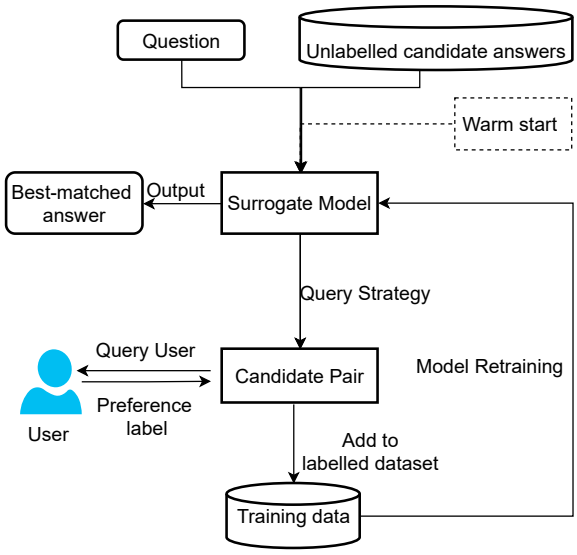

Figure 1: Workflow of our Proposed Approach

Siamese architecture with margin ranking loss is used to learn a utility function for each candidate from pairwise comparisons:

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

where f_1 and f_2 represent predicted utilities of two input texts and $y \in \{-1, 1\}$, indicates which input should be ranked higher, where $y = 1$ means candidate 1 ranks higher. 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

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 interaction phase, we use the original test set (Table 2). To train the initial model in the warm start phase, we use the original training and part of the original validation set, while the remainder is used to tune hyperparameters in the interaction phase (Table 3).

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

Table 2: Statistics for the dataset at the interaction phase.

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 3: Number of questions in the datasets for the warm start and interaction phase hyperparameter tuning.

Simulate 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 6.2. We estimate the utilities f_a and f_b with ROUGE-L, which calculates the longest common subsequence between candidates and gold answers.

Evaluation Metrics We compute *matching accuracy*, which is the fraction of top-ranked answers that match the gold answers, and *normalized discounted cumulative gain (NDCG@5)*, which compares the ROUGE-L score of the top five candidates to the gold answer.

Hyperparameters We set the margin m (Equation 2) to 0.1 and tuned hyperparameters at both the warm start and interaction phases (Appendix B). As shown in table 5, 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 accurately approximate the posterior. 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 model has over 82M parameters. All models are trained incrementally.

Model	Query Strategy	Cooking Acc	Cooking NDCG@5	Apple Acc	Apple NDCG@5	Travel Acc	Travel NDCG@5
Initial ranker with no interactions		0.539	0.64	0.458	0.593	0.643	0.667
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	**0.736**	0.683	**0.677**	0.654	**0.833**	0.702

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

#Samples	10	20	30	40
Accuracy	0.828	0.836	0.838	0.841
Time per interaction (s)	7	14	21	28

Table 5: Number of MCD samples on the validation set.

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 (P.V.S and Meyer, 2017) (UNC). 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 a good candidate and f_a is the predicted utility of a. We query the candidate pair (a, b) with the highest uncertainty values, $unc(a)$ and $unc(b)$.

6.1 Results: Performance Comparison

Table 4 shows that all Bayesian methods outperform the non-Bayesian deep ranker in most cases. For the topic *Apple*, the accuracy of the deep ranker with MCD is 26% 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, which 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, but selects predicted utilities close to zero, so may select middling candidates and thereby waste labelling effort.

As shown in Table 4, 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 limited MCD appears more effective.

6.2 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 the accuracy of the Bayesian deep ranker with MCD for the topic *Travel*. Table 6 shows that as noise increases, the matching accuracy decreases, but there is no significant drop 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.

Noise Level	0.3	1	2.5
Accuracy	0.833	0.828	0.795

Table 6: Effect of simulated user noise on the Bayesian deep ranker with MCD.

6.2.1 Computational Cost

The time per round of interactive learning increases from ~2 seconds on a single NVIDIA RTX2080Ti GPU with the non-Bayesian method, to ~16 seconds with MCD or SWAG due to weight sampling.

7 Conclusion

We proposed an interactive Bayesian deep learning method for cQA, which keeps the strong representation learning ability of neural networks while considering the uncertainty in the model itself. Our method can be generalised to any interactive ranking task whose aim is to select the best candidate. Experiments showed our method achieves state-of-the-art performance compared with a non-Bayesian deep ranker and a shallow Bayesian method, and is robust to noise in the user feedback. We also showed when approximating posterior distributions under tight compute time constraints, MCD can outperform SWAG. Evaluation with real users remains a task for future work, along with parallel sampling from posteriors to speed up Bayesian inference.

8 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 systems on the answers 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 cQA data as an initial investigation into the potential for BDL in answer 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 and to users of different backgrounds.

References

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. 2019. Preference-based interactive multi-document summarisation. *Information Retrieval Journal*, pages 1–31.

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*.

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.

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.

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.

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 Appendix: Architecture

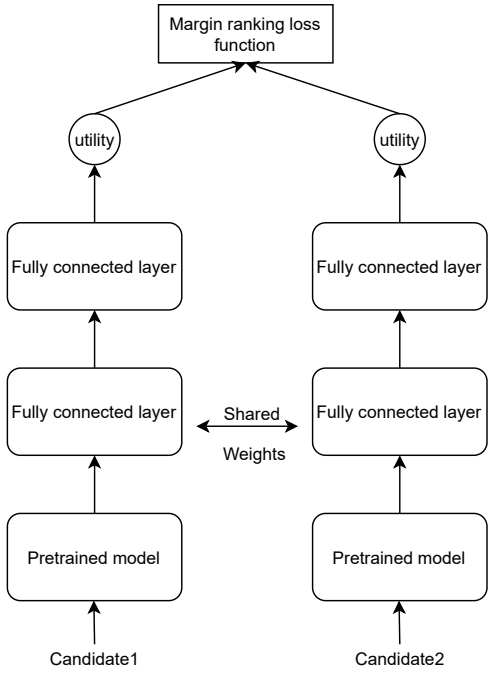

Figure 2: Architecture of the proposed surrogate model

Figure 2 shows the architecture of the deep ranker used as a surrogate model in the interactive learning process. All weights are shared between candidates 1 and 2 (Siamese architecture) and the architecture is used for both the Bayesian and non-Bayesian deep rankers.

B Appendix: Hyperparameter Tuning

For hidden sizes of our model, we followed the setup of previous work to make our work comparable as shown in table 7. 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	orignial size
Fully-connected layer 1	100
Fuly-connected layer 2	10
Output layer	1

Table 7: Hidden size for different layers in our model

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 results are shown in Appendix for different topics. For other hyperparameters, the training epoch is fixed to 3 epochs for conventional deep ranker and 6 epochs for SWAG-based ranker. The Dropout rate is the default value, 0.1.

At the interaction stage, we empirically set both the number of iteration rounds and updating epochs to 4. The optimizer we used here is SGD with momentum. Early stopping is applied. Learning rate, weight decay, and the number of sampling from posterior distributions are tunable parameters. The search space of these parameters is as follows.

Parameters	Value ranges
learning rate	[1e-4, 5e-5]
sampling nums	[10, 20, 40]
weight decay	[0.01, 0.001]

Table 8: Search Space of hyperparameters at the interaction stage

Learning rate	Batch size	Weight decay	Validation Accuracy	
			Vanilla ranker	SWAG ranker
5e-5	16	0.001	0.964	0.913
5e-5	32	0.001	0.964	0.923
2e-5(1e-4)[1]	16	0.001	0.969	0.932
2e-5(1e-4)	32	0.001	0.968	**0.958**
5e-5	16	0.01	**0.970**	0.954
5e-5	32	0.01	0.967	0.957
2e-5(1e-4)	16	0.01	0.969	0.943
2e-5	32	0.01	**0.970**	0.955

Table 9: Validation Accuracy of different combinations of hyperparameters of different models on Topic Cooking

[1] hyperparameters in parentheses are tuned for the SWAG-based model

Learning rate	Batch size	Weight decay	Validation Accuracy	
			Vanilla ranker	SWAG ranker
5e-5	16	0.001	0.936	0.921
5e-5	32	0.001	0.936	0.916
2e-5(1e-4)[1]	16	0.001	0.941	**0.922**
2e-5(1e-4)	32	0.001	**0.943**	0.913
5e-5	16	0.01	0.929	0.912
5e-5	32	0.01	0.938	0.920
2e-5(1e-4)	16	0.01	0.939	0.914
2e-5(1e-4)	32	0.01	0.935	0.922

Table 10: Validation Accuracy of different combinations of hyperparameters of different models on Topic apple

[1] hyperparameters in parentheses are tuned for the SWAG-based model

Learning rate	Batch size	Weight decay	Validation Accuracy	
			Vanilla ranker	SWAG ranker
5e-5	16	0.001	0.975	0.548
5e-5	32	0.001	0.974	0.889
2e-5(1e-4)[1]	16	0.001	**0.981**	0.541
2e-5(1e-4)	32	0.001	**0.981**	0.968
5e-5	16	0.01	0.977	0.968
5e-5	32	0.01	0.979	0.837
2e-5(1e-4)	16	0.01	**0.981**	**0.971**
2e-5(1e-4)	32	0.01	0.980	0.967

[1] hyperparameters in parentheses are tuned for the SWAG-based model

Table 11: Validation Accuracy of different combinations of hyperparameters of different models on Topic travel

Learning rate	Weight decay	Acc of Bayesian deep ranker		Acc of Non-Bayesian deep ranker
		Dropout	SWAG	
1e-4	0.01	0.777	0.71	0.7
1e-4	0.001	**0.765**	**0.73**	0.71
5e-5	0.01	0.74	0.70	0.698
5e-5	0.001	0.749	0.72	**0.715**

Table 12: Results of different combinations of hyperparameters on Topic Cooking

Learning rate	Weight decay	Acc of Bayesian deep ranker		Acc of Non-Bayesian deep ranker
		Dropout	SWAG	
1e-4	0.01	0.644	0.524	**0.451**
1e-4	0.001	**0.656**	**0.552**	0.437
5e-5	0.01	0.641	0.551	0.44
5e-5	0.001	0.633	0.541	0.432

Table 13: Results of different combinations of hyperparameters on Topic Apple

Learning rate	Weight decay	Acc of Bayesian deep ranker		Acc of Non-Bayesian deep ranker
		Dropout	SWAG	
1e-4	0.01	**0.836**	0.75	0.68
1e-4	0.001	0.833	0.751	0.668
5e-5	0.01	0.831	0.744	0.675
5e-5	0.001	0.833	**0.76**	**0.678**

Table 14: Results of different combinations of hyperparameters on Topic Travel