DEPARTMENT OF ENGINEERING MATHEMATICS

Bayesian Deep Learing for Interactive Question Answering

Haishuo Fang

A dissertation submitted to the University of Bristol in accordance with the requirements of the degree of Master of Science in the Faculty of Engineering.

Monday 20$^{\text{th}}$ September, 2021

Declaration

This dissertation is submitted to the University of Bristol in accordance with the requirements of the degree of MSc in the Faculty of Engineering. It has not been submitted for any other degree or diploma of any examining body. Except where specifically acknowledged, it is all the work of the Author.

Haishuo Fang, Monday 20th September, 2021

Contents

List of Figures

List of Tables

List of Algorithms

Abstract

Question Answering(QA) systems can automatically answer human questions based on machine learning and information retrieval techniques. A variety of intelligent AI assistants like Siri, Alexa, Xiaoice are emerging and blooming due to rapidly growing QA systems. These QA systems are good at answering factoid and task-oriented questions. Users, for instance, often uses them to query weather or some facts. The common character of these questions is that answers are very short, typically a phrase or a sentence. However, when it comes to non-factoid questions like how, why, these assistants are unable to obtain satisfactory answers since those answers tend to be exploratory or descriptive in the format of passages or documents, posing a big challenge for current QA matching algorithms.

Most of current approaches are based on conventional deep learning methods. In our paper, we turn to interactive learning, which puts users or humans in the loop to iteratively instruct model learning within very small annotation effort. The widely adopted method to interact with humans is active learning in which the most informative instances will be presented to human for labelling. For informativeness measurement, we consider two factors: uncertainty measurement and representation ability of text. Classical deep learning models are not well-calibrated especially when generalising out of training distribution and do not quantify the epistemic uncertainty. Conversely, the shallow Bayesian method in the previous work [71], Gaussian Process Preference Learning(GPPL), is able to model epistemic uncertainty but lack strong representation ability. Therefore, we adopt Bayesian deep learning which is able to measure epistemic uncertainty while keeping the representation ability of neural networks. Specifically, we build a Bayesian version of pretrained language models and apply MCDropout and SWA-Gaussian(SWAG), simple, scalable approximate Bayesian inference methods, to estimate posterior distribution over model weights. Then, we put it into the active learning framework to measure the informativeness of instances. The adopted query strategy is expected improvement. To evaluate the performance of our approach, we compare it with other methods, including a classical non-Bayesian deep learning model and a shallower Bayesian model, Gaussian Process Preference Learning(GPPL), on a community Question Answering(cQA) dataset. It shows our models can achieve higher accuracy with fewer interactions.

The main contributions of our paper are as follows:

- We found that Bayesian deep learning methods achieve the better average and worst-case performance over a shallow Gaussian-process-based method, GPPL, within fewer interactions for interactive question answering.
- We demonstrated Bayesian methods outperform non-Bayesian methods in selecting informative instances to query.
- We found Bayesian deep learning methods are robust under circumstance of noisy users.
- We developed software for this project with different Bayesian deep learning methods.

Supporting Technologies

- I used Python as the main programming language.
- PyTorch is used as the deep learning framework and pretrained models are provided by the library Transformers.
- I used the Perl script of Rouge to obtain rouge scores for simulating users' preference.
- I used LaTeX to format my thesis, via the online service *Overleaf*.

Acknowledgements

I would like to express my deepest appreciation to my family. Without their support, I could not obtain this Master degree, especially in this remarkably difficult year. It is never too late to say 谢谢你，老爸老妈。

I am extremely grateful to my supervisor Dr. Edwin Simpson. He never wavered in his support for any resource and help I need. His patience and professional instruction helped me finish my dissertation and achieve some very useful research results. And many thanks to all lecturers in this master programme. They make this dark year bright again.

Last not least, I'd like to acknowledge the computational resources provided by the HPC of the University of Bristol.

Chapter 1

Introduction

This chapter starts with an introduction to backgrounds of QA systems before providing motivations for interactive QA and demonstrating why it is an important study field. Next, we illustrate the limitations of previous interactive learning methods and give a description of our proposed approaches and highlight development challenges. Finally, research aims of the project are outlined.

1.1 Background

Developing question answering systems to automatically answer human questions has been a long-standing research field, which is a challenging topic in AI, requiring synthesis of machine learning, natural language processing (NLP), knowledge representation, reasoning and human interactions. The QA system can be defined as given a question in natural language, the system returns one or more answers to the question. In this case, QA systems can be considered as an advanced form of information retrieval whose aims is to retrieve more specific and coherent answers[49].

QA systems can be classified into two categories: factoid and non-factoid according to the type of answers[72]. For factoid questions, QA systems are expected to return short and precise answers to given questions. For instance, *When was the University of Bristol founded?* expects exact time as an answer. Regarding to the information source used to extract answers, factoid QA systems can be further divided into two types: QA with knowledge bases and QA with the web[77]. QA with knowledge bases retrieves facts from existing, structured databases while QA with the web derives answers from unstructured web documents [10, 18, 4].

For non-factoid QA systems, questions are open-ended that requires long and complex answers like advice, explanation or description[30]. Questions like *How do question answering systems work?* are non-factoid. Answers to this question tend to be explanatory and descriptive long documents. Therefore, non-factoid answers are always multi-sentence texts rather than an individual entity or a single sentence. This proposes challenges for current QA systems which performs poorly on capturing relevance and similarity between long texts[62].

1.2 Motivations

Against the above background, our project focuses on answering selection or ranking methods for non-factoid QA systems.

1.2.1 Community Question Answering

The exemplary example of non-factoid QA is community Question Answering(cQA), like StackExchange, Quora, Stack Overflow where people can freely ask any questions. There are two characters distinguishing cQA from common QA. First, questions are composed of subjects and bodies. Subjects often express key information and bodies are used to give more detailed descriptions. Second, the majority of questions and answers are at the passage-level which means they are multi-sentence texts, making it difficult for questioners to select the good answer. Therefore, building automatic QA systems to help questioners select best answers is important.

1.2.2 Search engine

Intelligent search engines need to understand users' queries and directly return answers rather than relevant links which require users to extract information from relevant documents by themselves [11]. Google search engines are good at answering factoid questions but not non-factoid questions. This project is critical for improving the ability of search engines to answer non-factoid questions.

1.2.3 Current Approaches

Current approaches can be categorized into representation-based and interaction-based approaches[28]. Representation-based approaches focus on learning complex representation functions to obtain high-level representation of questions and answers and then use simple evaluation function to get relevant scores, like Siamese Network[50], DSSM[33]. This approach is more suitable for short sentence representation since high-level representations of long sentences are difficult to learn. Interaction-based approaches directly model relevance between questions and answers using complex interaction function, like Match-SRNN [74], DRMM[27]. Although this approach performs well on long sentence matching via complex interactions, it requires sophisticated model design and large datasets to obtain good results.

1.2.4 Interactive learning

Compared with traditional machine learning models that are learned from offline data without human interaction. Interactive learning, which has gained more and more research interests these days[2, 59, 42], involves humans in the loop to instruct models training, fine-tuning or adaptation especially when the quality of train data is poor or the number of labelled data is limited. For this project, a large amount of labelled domain-specific data is unavailable and there exists some noisy data that are unable to be eliminated. Therefore, it is a good choice to directly obtain a few labelled data from end users via interactive learning. Another difficulty of this project comes from the contrast between two comparable candidates which can both answer given questions but with different emphasis and details, making models confused. In this case, end users' feedback can efficiently help models capture preference features. Moreover, methods used in non-factoid QA problems could be generalized to other tasks where interacting with humans to get more data is needed, like text summarisation[25], recommendation systems.

1.3 Our Approach

We propose a Bayesian deep interactive ranking method, which combines interactive learning and Bayesian deep learning, to solve the non-factoid QA matching task. Unlike traditional deep learning methods which requires enormous training data, this approach can achieve high performance with only a few interactions within the user. Given a question and a pool of candidate answers, we can get the best matched answer via a small amount of users' feedback. In order to interactive with the user and minimise the number of labelling, Active Learning(AL) is chosen as the framework that is illustrated in 2.2. During each iteration of AL, there is a query strategy to decide which candidate should be labelled by the user. Simpson et al.[71]

firstly introduce Bayesian Optimisation integrating with Gaussian Process to efficiently select candidate to query and dramatically reduce the number of interactions. However, this method lack strong textual representation ability and the cold start is also a problem. This project researches whether Bayesian deep learning is able to improve textual representation ability and retain the ability to measure epistemic uncertainty. Specifically, for the surrogate model of Bayesian deep learning, we choose RoBERTa[44], a state-of-the-art pretrained language model based on BERT, which provides strong prior knowledge for downstream tasks, especially for small datasests. SWA-Gaussian(SWAG)[46] and MC Dropout[23] are used as optimization approaches to approximate posterior distribution over the whole model weights.

It should be noted that the user is required in interaction sessions. To obtain a user's feedback, we use a noisy user-response model[73] to simulate it. So the noisy degree is also researched to figure out how the noise affects performance of our approach. With regards to efficacy of Bayesian methods, we also compare Bayesian methods with non-Bayesian methods. Details of our approach are described in Chapter 3.

1.3.1 Research Aim

To develop a new answer selection method based on interactive learning, our aims are summarised as follows.

- Validate whether Bayesian deep learning methods are necessary compared with non-Bayesian deep learning methods.
- Research whether advanced Bayesian deep learning [45, 16] can improve the accuracy of top-ranked answers within fewer interactions(labelled data) compared with existed shallower Bayesian models(GPPL).
- Compare the performance of different Bayesian deep learning methods.
- Investigate the robustness of Bayesian deep learning methods under noisy circumstances.
- Build software for this project, including framework for interactive learning simulations, Bayesian BERT with SWAG and MC Dropout.

1.3.2 Challenges

The main challenge to this project is how to integrate Bayesian deep learning into the interactive learning framework as it is the first trial to design Bayesian deep interactive ranking methods. There three concerns are as follows.

- The first concern is how to approximate posterior distributions over enormous weights of neural networks. The approximation methods should be scalable, simple and fast since the end user is unable to wait too long to interact with the system.
- Another concern is related to model design. How to suitably set the number of epochs, learning rate and the number of interactions needs to be explored.
- The third one is related to the efficiency. The most time-consuming part is the time to select the most informative instances and the time for multiple training phases of this model. Whether computational time can meet the user's needs should be tested.

Chapter 2

Background

This chapter provides contextual backgrounds of interactive QA systems, techniques of interactive learning and its application in NLP (section 2.1). Active learning is introduced as the framework in section 2.2. Next, we describe Bayesian Optimisation in section 2.3 as it is the main sampling strategy for our project. Section 2.4 gives the description of current Bayesian deep learning which provides calibrated epistemic uncertainty measurement. Finally, we integrate them together and introduce deep active learning in section 2.5.

2.1 Interactive learning for NLP

Interactive learning has been applied in many NLP tasks, including question answering, machine translation, text summarisation. PVS et al.[59] use active learning with users' feedback to develop multi-document summarization systems. Lin et al. [42] develop visual question answering models based on active learning with different types of scoring functions. Peris et al. [57] adapt active learning to select effective source sentences in unbound data stream to improve quality of translation systems. Radlinski et al. [60] design a new method to obtain natural conversational preferences for conversational recommendation systems in a dialogue framework. These works use uncertainty sampling strategies to select unlabelled data to query users, which leads to a large number of interactions and increases end users' cognitive load.

For interactive question answering, Simpson et al.[71] propose an interactive ranking method based on Bayesian optimisation and Gaussian Process Preference Learning (GPPL) which dramatically reduce the number of interactions with users and substantially improve model performance in non-factoid QA systems and text summarisation. The key insight behind this method is that ranking all candidates is unnecessary and wasteful since what we want to find is the best candidate. To get the best candidate more efficiently, the GPPL-based interactive ranking method uses Bayesian Optimisation(BO) with effective acquisition functions, like Expected Improvement(EI) to select the best answer within only a few interactions. BO aims at finding extrema of an objective function by prioritising candidates that appear more promising from prior results. We discuss more details of BO in section 2.3. However, the surrogate model of this method, GPPL, is not good at representing features of text. To solve the cold start problem and obtain better textual representation, Simpson et al.[71] train an initial ranker using a deep learning model and inject it to GPPL as the prior information.

2.2 Active Learning

This section introduces the framework of interactive learning, active Learning(AL), and different scenarios of AL. After that, methods used to select the most informative data points to query are explained.

Active learning is a machine learning method in which models can actively select informative data points to get labels from oracles(e.g. annotators or end users). It aims at achieving better performance by using fewer labelling data via actively interactions with oracles. It is suitable for tasks where there are lots of abundant data and labels are difficult to obtain.

2.2.1 Scenarios of Active Learning

The general AL setting consists of a small set of labelled data and a large set of unlabelled data. Models are initially trained on small labelled data. From unlabelled dataset, query strategies are used to select data points for oracles to label. Then, these new labelled data points are added into training data and retrain the model again. This process would be repeated until achieving good performance. The process is shown in Algorithm 1.

Algorithm 2.1: Active learning

Input : A pool of unlabelled data $\mathcal{U} = \{x^1, x^2, x^3, x^4, ...\}$, labelled dataset $\mathcal{D}$, the predefined number of training data N, a query strategy f, a model m.
Output: The trained model m
$\mathcal{D} \leftarrow \Phi$
while $|\mathcal{D}| < N$ **do**
  Select most informative instances $X \in \mathcal{U}$ according to f
  Query the Oracle to obtain labels Y of X
  $D \leftarrow D \cup \{X, Y\}$
  $\mathcal{U} \leftarrow \mathcal{U} \setminus \{X\}$
end

According to data source, AL could be divided into three categories[65]: Membership query synthesis[3], Stream-based selective sampling[13], Pool-based active learning[38].

Membership query synthesis describes a setting where queries are generated by learners(models) rather than sampled from natural distribution. In this scenario, obtaining a real data point is expensive and difficult. The idea of this method is that trained models are able to capture underlying data distribution well to generate reasonable data points, so human annotators can label these data points to help model learning. For instance, given an image, the model needs to construct the image for the oracle to label. However, there is a problem that the constructed data points may not be understandable by humans, making this method infeasible in many areas. Schumann et.al [63] applied Variational Autoencoders for query generation, making membership query synthesis feasible in NLP field and achieving competitive performance to pool-based AL strategies in text classification.

Stream-based selective sampling obtains unlabelled instances from real data source but it can derive only one instance at a time. Then, it decides whether to query instance based on informative measurement or some query strategies. The assumption is that getting an unlabelled instance is inexpensive so that it can be sampled from real data rather than synthesized. This method is powerful especially when memory or computing resources are limited. This approach has been adapted in part of speech tagging[15], learning to rank[20]. The workflow of this method is as follows.

Pool-based active learning is the most widely used AL strategy. In pool-based active learning setup, there is a small set of labelled data χ and a large pool of unlabelled data Φ. Initially, the model has acquired the small set of labelled data χ. A large pool of unlabelled data Φ would be ranked by the acquired model where the most informative data point would be chosen to label. Once labelled, these instances will be added intoχ to retrain models. The overall objective is to maximize model performance while keeping the number of data or the cost at a minimum. This method is suitable for those gathering data is simple but labelling is difficult to obtain. However, this method is computationally expensive since it needs to measure each unlabelled data to select the most suitable candidate to query during

each iteration. The difference between stream-based and pool-based active learning is that the former individually decide whether to query data points while the latter compare all unlabelled data. The pipeline of pool-based active learning is below.

2.2.2 Acquisition Function

Measuring informativeness of unlabelled instances is critical for queries selection. As mentioned in [56], acquisition functions (query strategies) can be classified into three categories: heterogeneity-based methods, representativeness-based methods and performance-based methods.

Heterogeneity-based Methods

The idea of this method is to select the most heterogeneous areas to obtain labels from oracles. Heterogeneity means the most uncertain data points in the unlabelled pool or those examples that are dissimilar or disagree with the current model. This approach tends to explore unknown but informative examples to query labels. However, it may lead models to only capture noisy and unrepresentative data especially when data are more noisy.

Uncertainty sampling [38] is an explicit and most commonly used strategy where instances that models are most uncertain about will be presented to the oracle for labelling. The uncertainty is often measured by probabilistic models but some models use deterministic models to measure uncertainty, which cannot distinguish aleatoric and epistemic uncertainty. A more generalised measurement is entropy. In information theory, entropy measures average information of a random variable. The more uncertain a random variable is, the more information is needed. Entropy-based acquisition functions are adopted in many fields, including semantic segmentation[70], sequence labelling[66], objective recognition[32].

$$x^* = argmax_x - \sum_i P(y_i|x;\theta) log P(y_i|x;\theta) \tag{2.1}$$

where $P(y_i)$ represents the probability of class i.

An alternative way to entropy is Least Confidence(LC)[39].

$$x^* = argmin_x P(y^*|x;\theta) \tag{2.2}$$

where $y^* = argmax_y P(y|x;\theta)$ is the label with the maximum probability. This method has been shown to work well in different tasks [39, 1, 80].

Query-by-Committee(QBC) [67] generates a committee of models on the same labelled dataset with different hypotheses and consider the most informative instances as those with the highest disagreement among these models. In order to implement QBC, measure of disagreement is needed. There are two approaches that are widely used. The first one is vote entropy[15] which calculates entropy of agreement percentage.

$$x^* = argmax_x - \sum_i \frac{V(y_i)}{C} log \frac{V(y_i)}{C} \tag{2.3}$$

Another measurement is average KullbackLeibler (KL) divergence[48]

$$x^* = argmax_x \frac{1}{C} \sum_{c=1}^{C} D(P_{\theta^{(c)}} || P_C) \tag{2.4}$$

where:

$$D(P_{\theta^{(c)}}||P_C) = \sum iP(y_i|x;\theta^{(c)})log\frac{P(y_i|x;\theta^{(c)})}{P(y_i|x;C)} \tag{2.5}$$

$$P(y_i|x;C) = \frac{1}{C}\sum_{c=1}^{C} P(y_i|x;\theta^{(c)}) \tag{2.6}$$

where $\theta^{(c)}$ represents a certain model and C represents the whole committee. In this setup, the disagreement is described as the average difference between the label distributions of any committee member and the overall committee. QBC has been applied to many machine learning algorithms like Navie Bayes[52], random forest[37], SVM[7].

Expected model change [9] aims at selecting data points which could lead to the largest change to the current model at the level of model parameters. Specifically, this method choose the data point at which the gradient of loss is the largest among all unlabelled data. However, true labels of unlablled data points are unavailable in advance. The author uses expected changes over all labels to approximate the true change. The potential problem is it can be computationally expensive if features space and label space is expensive.

Performance-based Methods

The idea of this method is simple and straightforward that looks at whether the added queried data point can improve the performance of the model on unlabelled data. This method focuses on reducing future error on unlabelled data, which could be considered as valid dataset, avoiding the bad effect of outliers or noise. Expected Error Reduction and Expected Variance Reduction are discussed here.

Expected Error Reduction [61] selects examples which can minimize expected future error. For each unlabelled example x and possible label y, this method add the pair (x, y) to the training set and retrain the model. Then, entropy of the posterior class distribution (2.7) are used to estimate expected future error. The data point has lowest expected error on all other examples is selected.

$$E_{P_{D^*}} = \frac{1}{|P|}\sum_{x\in p}\sum_{y\in Y} P_{D^*}(y|x)log(P_{D*}(y|x)) \tag{2.7}$$

The drawback of this method is that its computational cost is very high since it needs to retrain models for each label for each unlabelled data. In this case, this method is more suitable for simple binary classification tasks.

Expected Variance Reduction is to search data points that has the minimal expected variance reduction in unlabelled data. Geman et.al [26] have proven that the generalization error could be expressed as the sum of model bias, variance and inherent noise. Model variance is affected by examples selection. Compared to expected error reduction, this method can decrease computational cost via approximating the variance[55]

Representativeness-based Methods

It has been shown uncertainty-based methods are easy to query noise and outliers, which is the main motivation behind representativeness-based methods. This method consider not only uncertainty but also representativeness. Queries should not only be those which are uncertain, but also those that are representative of the whole data distribution. The representativeness is often measured using density-weighting. The equation is as follows.

$$x = argmax_x \phi_A(x) \times (\frac{1}{U} \sum_{u=1}^{U} sim(x, x^u))^\beta \tag{2.8}$$

Here, ϕ_A represents the base informativeness of x according to some base query strategies. The second measure representativeness of x by its average similarity to all other unlabelled data in the input distribution.

There are some other ways to measure density and representativeness. Fujii et al.[20] selects queries that are unlike data points in the training set but most similar to the unlabelled data. Nguyen and Smeulders.[51] propose a density-based method that clusters datapionts and avoid querying outliers by propagating label information in the same cluster.

2.3 Bayesian Optimisation

This section points out the advantage of Bayesian Optimisation in the ranking problem especially when the aim is to select the best candidate among a pool of unlabelled data points.

Bayesian Optimisation(BO) is a classical optimisation method, aiming at finding the maximum or minimum value of the objective function without assuming any functional forms. It is more useful when the objective functions are complex, "black-box" or expensive-to-evaluate. In the setting of Bayesian optimisation, a surrogate model is built for objective and quantifies uncertainty using Bayesian techniques and then acquisition functions are used to decide where to evaluate next based on the informativeness of each candidate.

The whole workflow of BO is much like that of AL but they have the essential difference. BO focuses on finding the extremum of the complex or expensive objective function while active learning wants to determine an accurate model via exploring the most uncertain point. Therefore, acquisition functions of BO focus more on extrema finding.

2.3.1 Acquisition function

Probability of Improvement (PI) choose the next point to query which has the highest improvement over the current maximum $f(x^+)$ where $x^+ = argmax_{x_i \in x_{1:t}} f_x$. It can be expressed as follows,

$$x^* = argmax(\alpha(x)) = argmax(P(f(x) \geq f(x^+) + \epsilon)), \tag{2.9}$$

$$\text{where } P(f(x) \geq f(x^+) + \epsilon)) = \Phi(\frac{\mu(x) - f(x^+) - \epsilon}{\sigma(x)}) \tag{2.10}$$

ϵ is used to control the balance between exploration and exploitation. With the increase of ϵ, the strategy tend to explore new points far away from current maximum.

Expected Improvement PI only considers the probability of improvement but does not quantify the improvement. Expected Improvement(EI) is proposed to solved this problem. Expected Improvement(EI) determines where to evaluate the model next according to improvement value. Specifically, improvement is defined as $max\{0, f(x) - f^*\}$, where $f^* = max_x f(x)$ is maximum value of data collected so far and $f(x)$ is the value of new candidates of current function f. This means we can receive the improvement $f(x) - f^*$ if $f(x)$ larger than f^* and no improvement otherwise. This kind of improvement can be computed as follows. Firstly, we need to estimate the posterior distribution over the candidate utility, which is a Gaussian Distribution $\mathcal{N}(\mu(x), \mathbf{C})$. Then, we can find the current best candidate $\mu^* = max\{\mu(x)\}$. For

any candidate x, the different between $\mu(x)$ and μ^* can be computed by cumulative density function $\Phi(z)$ where $z = \frac{\mu(x)-\mu^*}{\sqrt{v}}$. The ultimate form of expected improvement can be written as follows.

$$a_{EI}(x) = \sqrt{v}z\Phi(z) + \sqrt{v}\mathcal{N}(z; 0, 1) \tag{2.11}$$

Like PI, EI considers trade-off between exploitation and exploration. From this expression, we can see the EI can be large when $\mu(x) - \mu^*$ is high or the standard deviation $\sqrt{v}$ around a point is large. When $\mu(x) - \mu^*$ is large, points around the current best point will be evaluated(exploitation) while points with high uncertainty would be selected when $\sqrt{v}$ is high. The Expected Improvement captures both of them when querying data points.

This acquisition function is adopted in the interactive ranking method by Simpson et al.[71] which achieves the best performance among various acquisition functions.

2.4 Bayesian Deep Learning

Deep Learning has achieved significant breakthroughs in perception tasks like image recognition, text classification, speech recognition. However, when it comes to tasks involve inference and uncertainty measurement, conventional deep learning is unable to obtain good results due to the lack of the ability to capture model uncertainty, causal inference, logic deduction. Current deep learning methods are sometimes overconfident about their predictions and very prone to be overfitting. The first fatality related to AI systems is caused by an assisted driving system in 2016 which confused the white side of a trailer for bright sky. If the algorithm can give a high level of uncertainty to its wrong predictions, this disaster could be likely avoided. When the algorithm is uncertain about its own prediction, human can give instructions to make better results.

The reason why current deep learning models cannot give confidence interval of predictions is that deep learning often uses point estimation methods, like maximum likelihood estimation, to estimate a single value which could minimize the loss function without any uncertainty estimation. However, in Bayesian deep learning paradigm, all variables or weights are treated as stochastic variables, which means each variable has their corresponding distribution. So, during the prediction stage, Bayesian models can give confidence interval for predictions. Almost any model used for point estimate in conventional deep learning can be adopted in Bayesian deep learning, like Bayesian Recurrent neural networks[19], Bayesian Convolution networks[22].

2.4.1 Framework of Bayesian Deep Learning

Bayesian Deep Learning(BDL) is a type of stochastic deep learning model built by introducing stochastic components into neural networks (like stochastic activation functions and stochastic weights). In this case, it can capture multiple possible models weights θ with probability distribution $p(\theta)$, which could be considered as a case of ensemble learning. In the setting of BDL, a deep learning model or a functional model should be selected firstly. Then, one has to choose a prior distribution over the model parameterization $p(\theta)$ and likelihood function $p(y|x, \theta)$. Training(observed) data is added as new knowledge to calibrate distribution over model weights. According to Bayesian theorem, the above components can be written as

$$p(\theta|D) = \frac{p(D_y|D_x, \theta)p(\theta)}{\int_{\theta'} p(D_y|D_x, \theta')p(\theta')d_{\theta'}} \tag{2.12}$$

where, $p(\theta)$ is prior distribution over model parameters. $p(D_y|D_x, \theta)$ is likelihood function while $\int_{\theta'} p(D_y|D_x, \theta')d_{\theta'}$ is marginalization part. $p(\theta|D)$ is posterior distribution over model parameters.

Once obtaining posterior distribution, it is possible to get marginal probability distribution of the output, which gives the model uncertainty.

$$p(y^*|x^*, D) = \int_\theta p(y^*|x^*, \theta)p(\theta|, D)d\theta \tag{2.13}$$

The integral in equation 2.13 is also called Bayesian model average. θ is sampled from posterior distribution and $p(y|x, \theta)$ is the output of deep learning model, given sampled model weights θ. Equation (2.12) can be explained as the weighted average of all settings of model parameters in which weights are their posterior distribution.

2.4.2 Inference Methods

The most difficult part of equation (2.12) is the integral of evidence term $\int_H P(D|H)P(H)dH$ in Bayes' theorem, which is always intractable. Many inference approaches have been researched to directly approximate posterior distribution.

Markov Chain Monte Carlo(MCMC) methods are most popular ones. MCMC samples data points in a sequential process(Markov chain) where new samples only depends on previous samples so that elements in the sequence are finally distributed in a desired distribution. Metropolis-Hastings algorithms is a representative sampling method which could be used to estimate posterior distribution. The attribute makes this method is suitable for Bayesian deep learning is that it does not need to know exact posterior probability distribution to sample from. It only requires a function $f(x)$ that is proportional to target distribution. The basic idea is that using rejection mechanism with a local proposal distribution to update the elements of a Markov chain in turn. At each updating step, if the candidate is more likely than the previous datapoint, it would be accepted. Otherwise, it would be rejected.

Variational Inference(VI) is another method to approximate posterior distribution via defining a new distribution(variational distribution) among a parametrised family and optimising over parameters to make the variational distribution and the target distribution as close as possible. Compared to MCMC which samples data points from the real distribution, VI is less accurate but produces results much faster, making it more scalable. The measure of closeness most adopted is the KL divergence, which measures the difference between the exact posterior distribution and variational distribution.

$$KL(Q_\phi(H)||P(H|D)) = \int_H Q_\phi(H) log \frac{Q_\phi(H)}{P(H|D)} dH \tag{2.14}$$

where $Q_\phi(H)$ is variational distribution and ϕ is parameters that needs to learn. $P(H|D)$ is posterior distribution. According to Bayes' theorem in which $P(H|D) = \frac{P(H,D)}{P(D)}$, we can derive evidence lower bound or ELBO[76],

$$ELBO = \int_H Q_\phi(H) log \frac{P(H, D)}{Q_\phi(H)} dH = log(P(D)) - KL(Q||P) \tag{2.15}$$

We can observe to minimize $KL(Q||P)$ is equivalent to maximizing ELBO. For scalablity, Stochastic Variational Inference[31] is the most popular method.

Bayes-by-backpropagation in Bayesian neural networks, a variational inference based method is extended to update parameters called Bayes-by-backpropagation[8]. The core idea is to approximate true posterior distribution p with variational distribution q with parameter θ and estimate it via Monte Carlo sampling. The loss function is

$$\theta_{opt} = argmin_\theta \sum_{i=1}^{n} log q_\theta(w^i|D) - log p(w^i) - log p(D|w^i) \tag{2.16}$$

Then, it would use backpropagation to update parameter θ.

Monte Carlo Dropout In order to adapt most complicated neural networks, many new, efficient dedicated algorithms have been proposed. Monte Carlo Dropout has been shown to be a good method to approximate posterior distribution during the test phase. Gal and Ghahramani [23] develop a new theoretical framework proving Dropout can be considered as a type of Bayesian inference method and it has been assessed on classification, regression and reinforcement learning tasks, achieving state-of-the-art performance. It is a simple and fast way to do Bayesian approximation without any additional computational complexity. Monte Carlo Dropout can be considered as variational inference, with variation posterior distribution for each weight matrix as below,

$$z_{i,j} \sim Bernouilli(p_i) \tag{2.17}$$
$$W_i = M_i \cdot diag(z_i) \tag{2.18}$$

Where, p_i represents activation probability for layer i, $z_{i,j}$ decides whether dropout should be applied to the jth weight in layer i. M_i represents weight matrix before dropout.

For uncertainty estimation of predictive distribution(equation (2.13)), Monte Carlo estimate is used to sample T times from posterior distribution(equation (2.17)) and first two moments are estimated empirically. Specifically, T sets of vectors of realisations from Bernoulli distribution are sampled $\{z_1^t, z_2^t, ...z_L^t\}_1^T$, giving $\{W_1^t, ..., W_L^t\}_{t=1}^T$. So the expectation can be estimated as follows,

$$\mathbb{E} \approx \frac{1}{T} \sum_{t=1}^{T} \hat{y^*}(x^*, W_1^t, ..., W_L^t) \tag{2.19}$$

Variance can be estimated in the same way.

SGD-Based methods SGD can be reinterpreted as a Markov-chain[47] which provides theory foundation for many SGD dynamic-based methods,like SGLD[47], RECAST[64], SWAG[46].

2.5 Deep Active Learning

Deep active learning takes Bayesian deep learning as surrogate models to measure uncertainty. One of main work of introducing Bayesian deep learning into active learning is done by Gal and Ghahramani[24]. They firstly put Bayesian prior on the filters of convolution neural networks as surrogate models of active learning framework for image processing. For large-scale problems, it is infeasible to retrain the whole model after each acquired instance, especially when there are millions of parameters of models. Batch active learning approaches are proposed[58] to reduce the computational complexity caused by repeated model updates. The core idea is to consider batch construction as sparse approximation so that data in a batch can cover the entire data manifold. Specifically, the authors select the optimal batch whose log posterior $log(\theta|D \cup D')$ best approximates the complete data log posterior $log(\theta|D \cup D_p)$. Gal and Ghahramani[22] proposed an efficient Bayesian CNN, prevent overfitting on small dataset via placing a probability distribution over CNN's kernel. More interestingly, they prove that dropout is a type of Bernoulli approximate variational inference method which can be used to approximate variational distributions without any additional model parameters. References [75] adapt Bayesian active learning into distance metric learning to solve inefficient instances selection problems on small dataset.

In NLP field, references [68] incorporate active learning with NER models to build a lightweight architecture which can match state-of-the-art performance with only 25% original data. References [79] propose an AL method for text classification with efficient instances selection. This method focuses on embedding space rather than output space, which means sentence with words whose embeddings are more likely to be updated with the greatest magnitude will be selected. It is evaluated in sentence classification and document classification, showing better results than baseline AL approaches. For knowledge graph construction, ActiveLink[54] was proposed to predict entity relations. It optimises data query strategies and uses a meta-learning approach for incremental training.

Chapter 3

Design

This chapter elaborates on our proposed model architecture used to fill the research gap, as stated in Chapter 1.3.1. We applied Bayesian deep learning models to get the best answer for a given question within fewer interactions with the end user compared to current approaches[71]. In order to evaluate the effectiveness of our approaches in increasing the accuracy of top-ranked answers for all given questions and reducing the number of interactions, we compared the approach with a non-Bayesian model and GPPL, a shallow Bayesian model, in different topics of a cQA dataset. Firstly, the overview of our approach to doing interactive question answering is described in section 3.1. After that, the proposed Bayesian deep learning model and two comparable methods are explained in section 3.2. Moreover, query strategies used to select samples are introduced in section 3.3. In section 3.4, the warm start phase is discussed as well.

3.1 Overview of Our Approach

For a given non-factoid question and multiple candidate answers, the model needs to return the best matched answer to a user after several rounds of interactions. Specifically, this task could be considered as a sub task of learning-to-rank problem, in which a learning model needs to learn a ranking function based on the relevance with the question. In this case, more relevant answers should be ranked higher and the top one answer would be the most matched answer. Based on input space, hypothesis space and loss functions, approaches to learning-to-rank could be categorised into three groups: pointwise, pairwise, listwise[43].

For this project, we turn to pairwise methods, which is also called preference learning[21], in view of interacting with users. For listwise approaches, lists of candidates are required to help model learning which takes users large effort and time. Pointwise approaches are unable to achieve better results than pairwise and listwise methods. Pairwise comparison is a much suitable choice, which can reduce effort and time the user must expend to train a ranking model while keeping good model performance. In the setting of pairwise comparison, models take a pair of candidate answers as input and need to learn which one is better. For the preference information between one candidate pair $A > B$, there are two popular ways to depict it. One is assigning A and B real values a and b such that $a > b$, showing A is better than B. The task is converted to find a utility function which can map data to utilities(real numbers) so that the ranking problem can be solved by ranking those utilities. The other one is called preference relations which directly do binary classification of given a pair of candidates(instances or labels) which could solve the multiple labels ranking problem. However, preference relation is not transitive since it cannot give us overall ranking among all candidates only with pairwise labels. Therefore, we adopt the method based on utility functions to conduct pairwise(preference) learning.

The workflow of our proposed approach to fast and effectively finding best answers is stated in Figure 3.1. In the beginning, an existed surrogate model, which is described in detail in section 3.2, would be used to predict utilities of unlabelled candidates. Then, some query strategies would be adopted to

measure the informativeness of those unlabelled candidates. The most informative candidate pair would be presented to the user for preference labelling. Once the user's feedback of a new candidate pair is obtained, it will be added into the training set to train the surrogate model incrementally. For deep learning surrogate models, we found incremental training is more efficient and effective than retraining the model from scratch again. The whole pipeline is repeated several times until it converges or the predefined number of interactions has been reached.

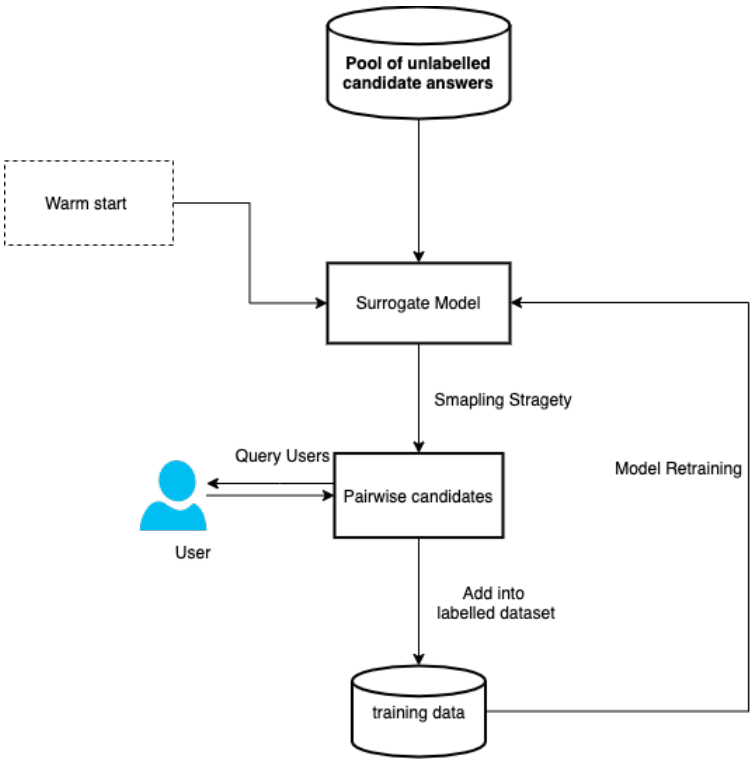


Figure 3.1: Overview of the Workflow of our Proposed Approach

We develop a deep interactive ranking algorithm for non-factoid question answering systems. Specifically, a Bayesian deep learning model is designed to compare with the previous work, GPPL and non-Bayesian methods. The corresponding sampling strategies are introduced as well.

3.2 Surrogate Model

This section introduces the Bayesian deep learning model we designed to measure the informativeness of unlabelled data. Firstly, the essential model architecture is illustrated. Then, the Bayesian version of it with two different inference methods are described.

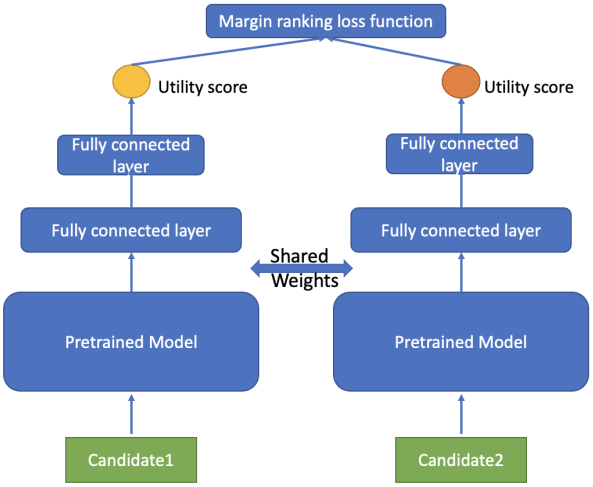


Figure 3.2: Architecture of Surrogate Models

3.2.1 Model Architecture

As shown in Fig 3.2, the model architecture follows the two-fold Siamese structure with components of a pretrained model, two fully connected layers and one output layer. It should be noted that the weights of two neural networks of the Siamese architecture are shared. The Siamese architecture is introduced to learn a utility function for each candidate in the setting of pairwise comparison. Siamese architectures[12] are widely used to find similarity or difference between a pair of inputs by comparing their feature vectors. This architecture receives a pair of inputs that are encoded by the same encoder and compared by subsequent layers. They have been successfully applied in tasks of textual similarity[50], face recognition[14], learning to rank[53].

For the pretrained model, we adopt BERT/RoBERTa, pretrained language models[41], which can provide strong prior knowledge for downstream tasks especially when the dataset is limited. Two fully connected layers are added to the output of BERT/RoBERTa to improve non-linear representation ability. Then, an output layer is used to get utility scores of input texts. Finally, a margin ranking loss function is used to optimize the whole model, which is stated as follows.

$$loss(x_1, x_2, y) = max(0, -y(x_1 - x_2 + margin)) \tag{3.1}$$

Here, x_1 and x_2 represents utility scores of input1 and input2 respectively. $y \in \{-1, 1\}$, indicates which input should be ranked higher. If $y = 1$, x_1 ranks higher than x_2 and vice versa for $y = -1$.

3.2.2 Bayesian deep learning Models

Based on the above model architecture, Bayesian deep learning models assume all weights are sampled from corresponding distributions rather than a single value. What the model needs to learn is parameters of distributions over weights like mean, variance for a Gaussian distribution. In this case, one model architecture can represent multiple models that are consistent with observations. In the classical procedure, only one model is selected, losing the ability to measure uncertainty and causing miscalibration. They are always overconfident about their predictions without epistemic uncertainty. A classical deep learning model uses maximum likelihood to approximate true weights $\theta = argmax_\theta P(D|\theta)$. It can be considered as a special case of Bayesian model average (2.13).

$$p(y^*|x^*, D) = p(y^*|x^*, \theta^*) \text{ with } \theta^* = argmax_\theta P(D|\theta) \tag{3.2}$$

Compared equation(2.13) and (3.2), we can see Bayesian models integrate all predictions weighted by probability of corresponding model weights, leading to better generalization and calibration. In order to evaluate the efficacy of Bayesian deep learning, we also conduct a classical(non-Bayesian) deep learning model in the same model architecture.

Setting Priors

Setting priors for small probabilistic models is simple and intuitive. For Bayesian deep learning models, it seems not easy to configure priors over millions of parameters since it is not explicit how a large neural network will generalize for given parametrization. Fortunately, recent experiments provide strong evidence that vague Gaussian priors over parameters in a neural network can induce useful inductive bias. Ulyanov et al.[17] shows randomly initialized convolution neural networks without training provide powerful results in image denoising, super-resolution. Zhang et al.[78] also prove that applying randomly initialized convolution neural networks without training to preprocess CIFAR-10 can improve the test result of a simple Gaussian kernel on pixels. These papers indicate Gaussian priors is a good default choice for large neural networks.

In practice, we only need to add L2 regularization to the loss function since L2 regularization could be interpreted as the Gaussian prior. According to the Bayesian rule(equation (2.12)), Maximum A Posterior(MAP) can be written as follows,

$$\hat{w} = argmax P(D|W)P(W) = argmax \prod_{i=1}^{n} P(D_i|W)P(W) \tag{3.3}$$

By taking the negative of the logarithm, we can get

$$\hat{w} = argmax P(D|W)P(W) = argmin - \sum_{i}^{n} logP(D_i|W) - logP(W) \tag{3.4}$$

We assume each weight w_j is sampled from a Gaussian Distribution with mean zero and variance λ.

$$P(W) = \prod_{j=1}^{d} P(w_j) \propto \prod_{j=1}^{d} exp(-\frac{\lambda}{2} w_j^2) = exp(-\frac{\lambda}{2} \sum_{j=1}^{d} w_j^2) \tag{3.5}$$

Based on equation (3.4) and equation (3.5), the MAP estimate would be

$$\hat{w} = argmin - \sum_{i}^{n} logP(D_i|W) + \frac{\lambda}{2}||W||^2 \tag{3.6}$$

As we can see, Gaussian priors can be explained as L2 regularization (the second term). Therefore, optimizers with explicit L2 regularization or weight decay can be used to approximate posterior distribution. In our project, we adopt AdamW and SGD with weight decay as optimizers.

Inference methods

Given prior distribution $P(\theta)$ and likelihood function $P(D|\theta)$, it only needs to compute the evidence term $\int_\theta P(D|\theta)P(\theta)d_\theta$ according to equation (2.12) to obtain posterior distribution over weights. However, this integral tend to be computationally intractable. Therefore, inference methods need to be applied to approximate posterior distribution. As described in 2.4.2, there are two main approaches: simulation-based methods(MCMC) and variational inference methods. For Bayesian deep learning models, scalability and efficiency of inference methods are the biggest concern due to complex model architectures and enormous weights. We select a simulated-based method, SWAG, and a variational inference method, MCDropout and compare their results in Table 5.11.

MC Dropout is adopted in this project. As stated in 2.4.2, it is a fast and easy method to approximate the posterior distribution without increasing any time complexity and computational cost. In practical, there is no difference from non-Bayesian Ranker during training. But for testing period, dropout should not be switched off in order to sample from variational posterior distribution.

Specifically, in variation inference, the aim is to minimize KL divergence between the real posterior distribution $P(\theta|D)$ and variational distribution $Q(\theta)$. Referring to [23], $Q(\theta)$ is selected as the distribution over all weights matrices whose columns are randomly set to zero based on Bernoulli distribution, which is described by equation (2.17).

For the uncertainty measurement for utility of candidate answers, we adopt Monte Carlo estimate that is described by equation (2.19).

SWAG Stochastic Weight Averaging Gaussian(SWAG)[46] is a SGD-based method, which leverages information about trajectory of SGD to efficiently approximate posterior distribution over model weights. Mandt et al.[47] showed the stationary distribution of SGD iterates can approximate posterior distributions in probabilistic models. However, for deep neural networks, some assumptions in [47] are not applicable due to non-convexity and over-parameterization. Fortunately, there are some evidences showing that posterior distribution is approximately Gaussian distribution in low-dimensional subspace spanned by SGD iterates and that is what SWAG does.[46]

In more detail, SWAG calculates the first two moments of the approximate Gaussian distribution via useful information of SGD iterates.

For the mean of Gaussian distribution, it adopts SWA[36], which directly averages weights along the SGD trajectory to estimate the true mean value. For instance, after T epochs, $\bar{\theta} = \sum_{i=1}^{T} \theta_i$.

For covariance approximation, it is estimated from two parts: diagonal covariance and low-rank covariance. The diagonal covariance is $\Sigma_{diag} = diag(\bar{\theta^2} - \bar{\theta}^2)$. The full covariance matrix can be written as $\Sigma = \frac{1}{T-1}\sum_{i=1}^{T}(\theta_i - \bar{\theta_i})(\theta_i - \bar{\theta_i})^T = \frac{1}{T-1}DD^T$, where D is the deviation matrix comprised of columns $D_i = (\theta_i - \bar{\theta_i})$, and $\bar{\theta_i}$ is the running average of weights obtained from the first i samples. To obtain low-rank covariance, covariance of the last K epochs are retained, forming $\Sigma_{low_rank} = \frac{1}{K-1}\hat{D}\hat{D}^T$, where $\hat{D}$ only consists the last K of D_i.

Finally, the approxiamte posterior combines the above components:

$$P(\theta|D) \sim \mathcal{N}(\bar{\theta}, \frac{1}{2}(\Sigma_{diag} + \Sigma_{low_rank})) \tag{3.7}$$

For uncertainty estimate, Monte Carlo sampling is applied following the below formula.

$$\tilde{\theta} = \bar{\theta} + \frac{1}{\sqrt{2}} \cdot \Sigma_{diag}^{\frac{1}{2}} z_1 + \frac{1}{\sqrt{2(K-1)}}\hat{D_{z_2}}, \text{where} z_1 \sim \mathcal{N}(0, I_d), z_2 \sim \mathcal{N}(0, I_k) \tag{3.8}$$

Where, d is the number of model weights, K is rank of the covariance matrix.

3.3 Query strategy

As discussed in 2.2.2, query strategies decide which candidate would be labelled by the user based on the informativeness of candidates. For this project, we adopt two different types of query methods: uncertainty-based strategy and expected improvement.

3.3.1 Uncertainty-based Strategy

For preference-based interactive learning, the most uncertain pair of candidates would be queried. P.V.S. and Meyer [5] firstly proposed the measurement of uncertainty for interactive document summarisation,

which defines the uncertainty of a candidate as follows,

$$unc(a|D) = \begin{cases} p(a|D), & p(a|D) \leq 0.5, \\ 1 - p(a|D), & p(a|D) > 0.5. \end{cases} \tag{3.9}$$

where $p(a|D) = (1 + exp(-f_a))^{-1}$ is the probability that a is a good candidate and f_a is the utility of the candidate a. Those candidates whose $unc(a|D)$ are close to 0.5 will get the maximum value. A pair of candidates(a and b) with the highest uncertainty values $unc(a|D)$ and $unc(b|D)$ will be selected to query.

This method dose not require any additional uncertainty measure. It is suitable for non-Bayesian methods to select queries.

3.3.2 Expected Improvement

Expected Improvement(EI) works very well with GPPL in the previous work and our research aim is not to evaluate different acquisition functions. So we directly adopt EI as the sampling strategy for Bayesian deep interactive methods. The details of EI could be found in section 2.3.1. For pairwise comparison, we follow the setting in the previous work, pairing candidate with the highest utility and the one with the maximum improvement together for preference labelling.

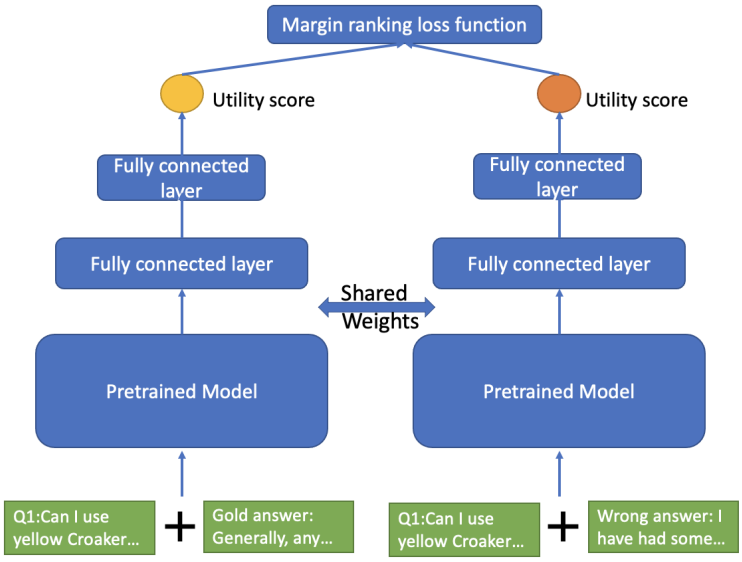
As shown in equation (2.11), the mean and variance of utilities of all candidates need to be calculated firstly. In Gaussian Process, it can be directly obtained. For Bayesian deep learning, we apply Monte Carlo sampling to obtain predictions of utilities multiple times and estimate the mean and variance based on those samples.

3.4 Warm Start

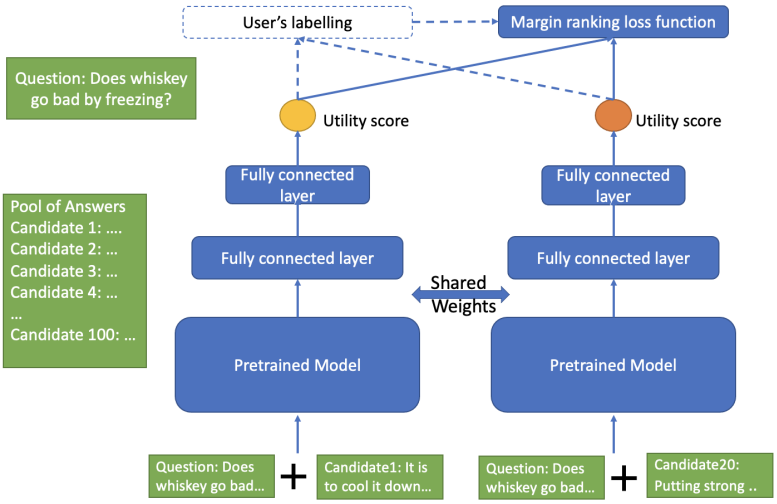
In order to solve the cold start problem, a warm start phase is introduced to tailor the pretrained model in domain. We split the dataset into train and validation as stated in 4.2 to train an initial ranker to provide prior knowledge for the interaction phase. Compared with the GPPL method, the initial ranker is directly fine-tuned at the interaction phase rather than taking it as prior information. Gururangan et al. [29] show continuous pretraining on a large corpus of unlabeled domain-specific or task-specific text can lead to big performance gains. Whether this kind of self-supervised pretraining can used as the warm start for this task can be researched in future work.

For training data construction, we randomly select 10 candidate answers as negative examples for each question and feed them into the ranker together with the corrected answer to learn general information of ranking. In line with the format of input of pretrained models, we concatenate the question and the candidate answer.

It should be noted that this training process at the warm start phase is different from that at the interaction phase. There is no label querying process during the warm start phase(3.3a). All questions are mixed up to form the dataset to learn a coarse-grained ranking. However, at the interaction phase, for each question, we need to acquire preference labels from the user to learn a fine-grained ranking. Figure 3.3 shows training process at the warm start phase and interactive phase.



(a) Model architecture in the warm start phase



(b) Model architecture in the interaction phase

Figure 3.3: Training difference between warm start phase and interaction phase

Chapter 4

Implementation

In this chapter, details of the implementation of approaches in chapter 3 are demonstrated. Section 4.1 introduces the basic experiment environment setup and datasets used to evaluate approaches are described in section 4.2. Users simulation is necessary for this project which is clarified in section 4.3. We then introduce evaluation metrics in section 4.4. Finally, the components of the built software and computation resources are stated in section 4.5 and 4.6.

4.1 Experimental Environment

For this project, the main programming language we adopt is Python which is very suitable for developing machine learning models due to its simplicity, understandability and compatibility. Moreover, its various libraries can dramatically boost software development. As the deep learning framework for this project, PyTorch is selected to conduct experiments as its pythonic programming syntax is handy to use. Besides, its dynamic computational graph makes it easy to debug compared with other deep learning frameworks with static computational graphs. Another programming language Perl is also used to calculate Rouge metric, which estimates scores of candidate answers when simulating users' preference. Specifically, we directly use the library developed by Chin-Yew Lin [40] for Rouge calculation. The details of Rogue is discussed in section 4.3.

4.2 Dataset Description

In order to compare our results with previous work on interactive question answering[71], we perform experiments on cQA datasets which consist of questions posted on StackExchange in the communities Apple, Cooking and Travel[62]. For a given question, there are one accepted answer marked by the user and 99 candidate answers collected from answers of similar questions. For the question part, it only retains the title and discards the detail description in the question body. The following is an example of the cQA dataset.

Q: Does whiskey go bad by freezing?
Candidate Answer1: It is to cool it down without diluting it–ice cubes would melt. And yes, you could simply cool the entire bottle, but it wouldn't look that fancy. Note that some purists would wrinkle their noses and insist that whisky is best enjoyed at room temperature and perhaps with a small dash of spring water. And I'm soooo not going into a whisky vs. whiskey debate here.
Candidate Answer2: Putting strong spirits in the freezer should not harm them. The solubility of air gases increases at low temperature, which is why you see bubbles as it warms up. Drinks with a lower alcohol content will be affected in the freezer. There is potential to freeze water out of anything with an alcohol content of 28% or lower. Many people use the freezer to increase the alcohol content of their home brew in UK, by freezing water out of it–the alcohol stays in the liquid portion.
Candidate Answer3: ...

The statistics of the dataset used for interaction is stated in Table 4.1. As we can see, the average answer length of these topics are over 100 words, which increases difficulty for current question answering techniques to process. Moreover, selecting one best matched answer from one hundred multi-sentence documents is also a hassle.

Topics	#questions	#accepted answers	#candidate answers	Avg. Answer Length
Apple	1,250	1,250	125,000	114
Cooking	792	792	79,200	189
Travel	766	766	76,600	214

Table 4.1: Statistics for Dataset for Interaction

In more detail, Table 4.1 is the test part of the original dataset, which is used for interactions. For the original training and validation dataset, we split them into two parts. The majority of it shown in Table 4.2 is used to train the warm start model. The remaining data shown in Table 4.3 is used to tune hyperparameters of the interactive model.

Topics	# Train questions	# Valid questions
Apple	5,831	1,249
Cooking	3,692	791
Travel	3,572	765

Table 4.2: Statistics for Dataset for Warm start

Topics	#Development questions
Apple	831
Cooking	692
Travel	572

Table 4.3: Statistics for Validation Dataset for Interaction

4.3 Simulated Users

Users' feedback is indispensable part of this project. However, it is difficult to get real users' feedback due to unavailability of this resource. A user-response model is introduced to simulate users' preference[73].The formula is as follows.

$$p(y_{a,b}|g_a, g_b) = \frac{1}{1 + exp(g_a - g_b)/t} \tag{4.1}$$

This formula means given utilities of two candidates, g_a and g_b, the simulated user will prefer document a with probability $p(y_{a,b}|g_a, g_b)$. t is a parameter used to control the noise degree of the user's preference. When t gets smaller, the utilities of a and b would be separate from each other so that the user is more likely to make the corrected choice. This user-response model becomes noiseless. In noiseless model, users would always select the candidate with the higher utility, which is not realistic. A noise model admits users' mistakes via allowing the user to select a candidate that does not maximize utility, which is more in line with reality. In this case, larger values of t would make a and b are close to each other, the preference probability would become close to 0.5 after transforming by the sigmoid function and the user's choice would be random as well. In order to investigate the effect of noisy users and test the robustness of proposed models, we tune the value of t and conduct experiments in section 5.3.

When it comes to getting utility for each pair of candidate answers, the Recall-Oriented Understudy for Gisting Evaluation(ROUGE) scoring algorithm[40] is introduced. ROUGE score calculates the lexical similarity between a candidate document and a reference or a collection of references. It is often used to evaluate automatic text summrisation and machine translation. In more detail, there are five different evaluation metrics.

- ROUGE-N calculates the overlap of n-grams between candidates and references. ROUGE-1(unigram) and ROUGE-2(bigrams) are common metrics.
- ROUGE-L calculates Longest Common Subsequence(LCS) between candidates and references. LCS considers the whole sentence structure and automatically identify the largest co-occurring n-grams strings.
- ROUGE-W is a weighted version of ROUGE-L and extend it to consecutive substrings.
- ROUGE-S calculates the overlap of skip-bigram, which means any pair of words in the same order would be counted.
- ROUGE-SU combines skip-gram statistics and unigram-based co-occurrence.

We choose the ROUGE-L as the function to calculate a utility for each candidate making our approaches comparable to previous work since it has been widely adopted to evaluate question answering systems[71, 6, 35]. Concretely, we estimate a utility of a candidate answer via calculating ROUGE-L score of it against the gold answer accepted by the user. Once obtaining utility scores of a pair of candidates, the user-response model of can work to simulate a user's preference.

4.4 Evaluation Metrics

Model evaluation directly reflects model performance and research objectives. Usually, generalization performance of a model on unseen datasets are estimated to evaluate a machine learning model. Widely used metrics to assess a classification algorithm is accuracy, confusion matrix, F1 score while Root Mean Squared Error(RMSE), R-squared are applied to regression models.

For interactive question answering, evaluation metrics would be different from conventional machine learning tasks. Our aim is to find the best answer to a specific question in as few interactions as possible rather than generalizing the model to a new question. Therefore, the primary metric is the matched accuracy which is the ratio of the number of top one answers matching gold answers across all questions divided by the number of questions. Users' cognition is always limited so another metric is the number of interactions. We hope to achieve the best performance using the least human effort. Another metric could be evaluated is the Normalized Discounted Cumulative Gain(NDCG), a measure of ranking quality. It is derived from Discounted Cumulative Gain(DCG). The expression is as follows.

$$DCG_p = \sum_{i=1}^{p} \frac{2^{rel_i} - 1}{log_2(i+1)} \tag{4.2}$$

where rel_i is the relevance of the document at position i. The higher the highly relevant documents rank , the larger the DCG_p. But this metric cannot be used across different queries since it ignores variety of queries. Normalized DCG(NDCG) is proposed to normalise the result between different queries.

$$nDCG_p = \frac{DCG_p}{IDCG_p}, \text{where } IDCG_p = \sum_{i=1}^{|REL_p|} \frac{2^{rel_i} - 1}{log_2(i+1)} \tag{4.3}$$

NDCG measures the average performance of the ranking method via averaging values for all queries.

In order to evaluating the performance of different surrogate models, the following combination of query strategies and surrogate models have been implemented and compared. All surrogate models have the same base architecture on all datasets as stated in section 3.2.

- **Non-Bayesian Deep Ranker and Uncertainty Sampling**: Non-Bayesian model do not have posterior distributions over weights which means Bayesian-related sampling methods is not applicable.
- **Bayesian Deep Ranker with SWAG and Expected Improvement Sampling**: Simpson et al. [71] show Expected improvement function outperforms other query strategies.
- **Bayesian Deep Ranker with MCDropout and Expected Improvement Sampling**: The only difference with the above one is that MCDropout is used as the inference method.

These combinations would be evaluated in chapter 5.

4.5 Software Development

This section introduces different modules of the software for this project. It can be broken into five different modules:

- Model: the essential model architecture described in 3.2 is implemented in the this module.
- Queriers: The queriers module includes different query strategies used in this project: uncertainty-based strategy and expected improvement.
- Oracle: The oracle module is responsible for simulating the user's preference.

It should mentioned that the above modules follow the framework of the GPPL interactive ranking method[1].

- Learners: we have a non-Bayesian learner and a Bayesian learner with SWAG and MCDropout as inference methods.
  - For the implementation of SWAG, we refer to the implementation of the original paper[2].
  - As the implementation of dropout is quite easy, we directly set dropout to be true during test to approximate posterior distribution.

  The class of each learner has a method for training, a method for updating parameters at the interaction phase and a method for predictions based on posterior distribution.
- Main.py: this script is the entrance of the whole project and conducts the whole pipeline for this project.
- Shell scripts: Scripts of experiments are used to submit experiments with different setups to computing nodes of High Performance Computing Systems(HPC).

[1] https://github.com/UKPLab/tacl2020-interactive-ranking
[2] https://github.com/wjmaddox/swa$_g aussian$

4.6 Computation Resources

All experiments were conducted on the resources provided by High Performance Computing Systems(HPC) of the University of Bristol. For each interaction, there are several components with high time complexity.

- Firstly, getting utilities for a pool of unlabelled candidates is really time-consuming. The main reason is that utilities come from the average of multiple predictions of tens of samples from posterior distributions over all weights.
- After that, selecting the most informative pair of candidates among them is a concern as well. It is required to calculate informativeness of candidate pairs and select the maximum one.
- Then, the new labelled pair candidates would be added into the training dataset to incrementally train the model within several epochs, which increases overall time.

This pipeline will be repeated several times for a single question. It is necessary to accelerate the whole workflow. We apply multiple GPUs to incrementally train our surrogate models. For different questions, parallel computing is used to speed up by submitting an array job with multiple sub jobs to the computing cluster. The specific resources are listed as below.

- GPU: 4× NVIDIA RTX2080Ti
- CPU: Intel E5-2680 v4 (Broadwell)
- ARM: 20G

Chapter 5

Critical Evaluation

This chapter presents experimental results with regard to research aims stated in Chapter 1.3.1. Section 5.1 conducts exhaustive experiments to select the optimal set of hyperparameters for models at the warm start stage and models at the interaction stage on different topics respectively. The speed of one round of interactions of these models is discussed as well. Then, performance of different interactive models are compared from the perspective of accuracy and NDCG@5 in section 5.2. Finally, in section 5.3, the robustness of Bayesian deep learning methods are investigated under noisy circumstances.

5.1 Hyperparameters Selection

Hyperparameters selection is a vital step for improving the performance of machine learning algorithms. Compared with model parameters, hyperparameters cannot be learned by the objective function. When developing machine learning algorithms for a specific task, lots of design choices need to be considered and tested to find the optimal set of hyperparameters.

For this project, we design our model based on a pretrained model and fine tune it according to the user's feedback for each given question. The pretrained model we chose is RoBERTa[44], which is a robustly optimized BERT. It was trained on 160GB of text with a much larger batch size and learning rates and introduces the dynamic masking mechanism to improve model representation ability. These architecture choices make RoBERTa generalize better to downstream tasks compared with BERT. In order to reduce computational times and model sizes, we choose DistilRoBERTa, a distilled version of RoBERTa, which is smaller than the original model but twice as fast as RoBERTa. It should be mentioned that other pretrained models could be also used to evaluate our approaches.

For hidden sizes of our model, we followed the setup of previous work to make our work comparable. Table 5.1 shows the setup of hidden sizes. We keep original weight sizes for each layer in DistilRoBERTa. For the first fully-connected layer, the number of neurons is 100 and the size of the second one is 10. The ouput layer only has one neuron so as to obtain the utility of the input text.

Layer name	# Hidden size
DistilRoBERTa	orignial size
Fully-connected layer 1	100
Fuly-connected layer 2	10
Output layer	1

Table 5.1: Hidden size for different layers in our model

It should be noted that the Bayesian deep interactive ranker with Dropout as the inference method and the non-Bayesian deep interactive ranker adopt the same model architecture. The only difference is if

its query strategy is Bayesian or not. However, for the SWAG inference method, the first two moments of the distribution over all weights are approximated. To speed up the sampling from posterior distributions, we only approximate posterior distributions over weights of the last 4 layers in our model since the entire model has over 82M parameters which is quite huge for sampling.

At the warm start stage, we use the AdamW optimizer for the conventional deep ranker which has the correct and implicit L2 regularization reflected by the weight decay. For the SWAG-based model, we use SGD with momentum as the optimizer since it will update the parameters of posterior distribution along the SGD trajectory. To separate each candidate from each other, we set the margin of the marginal loss function to be 0.1.

5.1.1 Hyperparameters tuning for the warm start phase

There are several important hyperparameters needing to be tuned, like learning rate, batch size and weight decay. The search space for the conventional deep learning and SWAG-based Bayesian deep learning is stated in 5.2.

Parameters	Value range
learning rate	[2e-5, 5e-5]
learning rate(SWAG)	[1e-4, 5e-5]
weight decay	[0.01, 0.001]

Table 5.2: Search space of hyperparameters for warm start models

We empirically set learning rate $\in \{2e-5, 5e-5\}$ for conventional deep learning, $\{1e-4, 5e-5\}$ for the SWAG-based model. batch size $\in \{16, 32\}$, weight decay $\in \{0.01, 0.001\}$. We exhaustively searched these combinations to find the optimal combination and the results are shown in Table 5.3 5.4 5.5 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.

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 5.3: 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 5.4: 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 5.5: Validation Accuracy of different combinations of hyperparameters of different models on Topic travel

As we can see, the optimal set of hyperparameters are in bold which would be used as the base model in the interaction phase for fine-tuning a specific model for each question.

5.1.2 Hyperparameters tuning for interaction

In each interaction, the surrogate model needs to select a pair of candidates according to sampling strategies and update model parameters based on acquired data so far. There are some hyperparameters need to be tuned as well. We empirically set the number of iterations to 4 and batch size to 1. The optimizer we used here is SGD with momentum. The number of epochs is fixed to 10. Early stopping is applied to early stop training if there is no improvements. 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 5.6: Search Space of hyperparameters

We hypothesize as the number of samplings increases, the result of Bayesian model averaging would be better. However, sampling from posterior distributions over millions of weights is computationally

expensive. In this case, we firstly investigate the relationship between accuracy and the number of samplings as well as the time each round of iteration needs in table 5.7. The definition of one round of interaction includes sampling a pair of candidates from a pool of unlabelled candidates, getting the preference label from the user, adding it into train data and doing incremental training. We use the Bayesian deep ranker with Dropout to conduct our experiment on the topic Travel.

Parameters	10	20	30	40
Accuracy on Validation dataset	0.828	0.836	0.838	0.841
Time of each of interaction(second)	7	14	21	28

Table 5.7: Relationship between the number of samples and accuracy on the validation dataset

As we can see, with the increase in the number of samplings, there is a small improvement in the accuracy while the computational time is doubled. With the limited compute resources, we have to make a trade-off between the number of samplings and the accuracy so we fix the number of samplings to 20. For future work, how to sample in parallel is a promising direction.

The dataset used to search the optimal hyperparameters has been introduced in section 4.2. It should be noted that there are three different methods compared at the interaction phase. We report results of those combinations of different models for different datasets in Table 5.8 5.9, 5.10. These optimal models would be used on the test dataset.

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 5.8: 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 5.9: 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 5.10: Results of different combinations of hyperparameters on Topic Travel

5.2 Analysis of performance of different models in cQA datasets

5.2.1 Results of different models in cQA datasets

It is a non-trivial problem to estimate posterior distributions over weights in Bayesian deep learning. Why do we need to take much effort and time to train Bayesian deep learning models? To evaluate the efficacy and necessity of Bayesian deep learning models, we conduct experiments to compare them with a classical deep learning model described in 3.2.1. GPPL with BO, the state-of-the-art interactive ranking method is compared as well. Results in Table 5.11 show Bayesian deep learning models achieve the highest matched accuracy among these models across all datasets within 4 interactions.

Model	Query Strategy	Cooking		Apple		Travel		# Interactions
		Acc	NDCG@5	Acc	NDCG@5	Acc	NDCG@5	
Initial ranker		0.539	0.64	0.458	0.593	0.643	0.667	0
Non-Bayesian deep ranker	UNC[2]	0.685	0.683	0.417	0.614	0.686	0.634	4
GPPL[1]	EI[3]	0.722	0.7331	0.614	0.715	0.792	0.744	10
Bayesian deep ranker_SWAG	EI	0.692	0.67	0.540	0.637	0.713	0.668	4
Bayesian deep ranker_Dropout	EI	**0.736**	0.683	**0.677**	0.654	**0.833**	0.702	4

[1] This result is obtained from the paper of GPPL with BO [71]
[2] UNC represents uncertainty-based strategy
[3] EI represents Expected Improvement

Table 5.11: Results of different interactive ranking methods for cQA

This observation demonstrates the necessity of Bayesian deep learning methods. As we can see, all Bayesian methods outperform the non-Bayesian deep learning model especially when the datasest is complex, like topic apple. The accuracy of Bayesian deep ranker with Dropout as the inference method are 26% higher than that of the non-Bayesian deep ranker. This huge gain comes from the ability to quantify epistemic uncertainty. Modelling epistemic uncertainty is imperative for this project since the labelling budget is quite limited in interactive learning. Epistemic uncertainty is caused by lack of knowledge, which could be reduced by additional resource of information(more training data). Opposed to this, aleatoric uncertainty is due to the inherently random effects. The model can give the probability for the possible outcomes but not definite answers. For example, models of coin flipping can only give probabilities of heads and tails rather than deterministic results. It is not reducible by additional data. The non-Bayesian deep ranker is unable to distinguish these two types of uncertainties [34].

To illustrate the problem it caused, an example is given. Saying there are two candidate answers: the real rank of one is medium and the other one is a new data point that is inconsistent with instances learned before. For the non-Bayesian method with the uncertainty-based query strategy stated in equation (3.9), both candidate answers would be considered as the most uncertain instances to acquire labels. However, candidates with intermediate utility would be repeatedly selected since the model does not know if its uncertainty is caused by its inherent features(mediocre utility), leading to wasteful labelling effort and poor model performance. This non-Bayesian ranker is uncalibrated as it is only one type of model among all possible models with the same architecture. In contrast, Bayesian methods can give epistemic uncertainty measurement when a new point is inconsistent with observed data.

Another benefit of our Bayesian deep rankers is the adaptation of Bayesian Optimisation, whose efficacy has been shown in the previous work[71]. Compared with uncertainty-based sampling strategies, BO strikes a balance between exploitation and exploration so as to finding extrema as soon as possible.

For the performance of different Bayesian methods in Table 5.11, the Bayesian deep ranker with MCDropout achieved the highest accuracy across all topics. The Bayesian deep ranker with SWAG has comparable performance but not as well as MCDropout in this task. The main reason is that SWAG approximates the mean and variance along SGD trajectory which means it requires sufficient epochs to update running parameters to capture the loss geometry. With the limitation of computation resources, we only average results of three epochs to approximate posterior distributions which is proven to be

not enough to get a good approximation. We found the covariance of weights derived from SWAG is relatively small so that posterior distributions tend to be sharply peaked, making it similar to conventional approaches. This limits the expected improvement strategy to explore new points. Moreover, we only update the posterior distributions over last four layers to speed up predicting which restricts the performance of SWAG.

For the metric of NDCG@5, we can see non-Bayesian deep ranker has higher value than its accuracy while accuracy of Bayesian deep rankers is larger than NDCG@5. This situation is attributed to different sampling strategies. Uncertainty-based sampling selects the most uncertain examples to fit the exact form of the objective function so that NDCG@5 is higher than accuracy in uncertainty-sampling strategies. However, Bayesian optimisation is to find the maximum or minimum points without fitting the exact form of functions so the accuracy of Bayesian deep rankers is much higher.

5.2.2 Case study

We give two concrete examples to illustrate that why Bayesian deep rankers have a better understanding of QA matching than GPPL and the non-Bayesian deep ranker.

Q1: Why is an album in Apple Music marked with an " E " and grayed out ?

Answer1(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 exact one on the Mac 's iTunes. If it didn't work, make sure both devices have been connected to the internet since setting the Apple ID. By the way, " Songs of Innocence " is **an album** by Irish rock band U2 , which was a 'gift' from Apple to you. If you want it removed, see URL Of course, connecting your iOS device to your Mac can help you sync them. URL webpage has got a table which lets you know how to do deal with some of these similar problems.

Answer2(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 in the album. (Edited to clarify the tap-and-hold gesture.

Answer3(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 restrictions entirely by selecting the "Disable Restrictions" option at the top of the Restrictions window.

Q2: If I buy an app from the app store , do I have to stay logged into the app store ?

Answer1(GPPL): Realistically, you need an Intel Mac, the cheapest possible one will do , and access to at least 1 to 3 iOS devices. One device for the current OS release, one device running the OS release of the oldest OS you plan on supporting, and perhaps one dedicated test device for doing beta OS **app** testing . If you want to do Universal **apps**, at least 1 iPad and one non-iPad , as there are many things that can't be fully tested on the iOS Simulator. Going forward, a Mac that can run Lion may end up being required to keep up with the latest SDK frameworks, codesigning tools and app submission tools . You need to register any iOS devices that you want to use for app testing ..... You don't need to **"register"** any Macs. You do need to create and/or copy private

keys and certificates to each Mac that you want to use for device testing development or **app store** submission. Unrealistically, you could get by with none...

Answer2(non-Bayesian deep ranker): Try searching your email for the string: Here's an example email receipt I received from the **iTunes App Store** which included an **in-app purchase** (in this case, an X-Men comic from the comiXology app)

Answer3(Bayesian deep ranker): Yes, although there are that cover the same ground already on SE. You can **DL app store apps** (including Lion) on your girlfriend 's computer. Then she can **log into her own apple account and buy other apps**. The only thing you need to be aware of is that for updates, you can only update the apps you bought with that particular account. So if you want to update Lion (not likely as it will come through software update) or other apps, you will have to re-log into your account. I believe most apps have a 5 computer limit.

Answers selected by the GPPL of the above two questions only captures some keywords in the question, like Music, Album, App store, apps but not understand the intention of questions. This is because GPPL lacks of strong textual representation ability of neural networks. From the answers obtained by non-Bayesian deeper rankers, we can see the non-Bayesian deep ranker is better at capturing semantic information than GPPL since it can get the topic of a question - album or app store purchase but still not hit the point of the question in only 4 interactions. However, Bayesian deep rankers equipped with Bayesian Optimisation can quickly get best matched answers.

5.3 Noisy users Experiments

In real-world applications, acquiring completely noise-free labels from Oracles is almost impossible. To test the robustness of our proposed models under noisy circumstances, we conduct experiments on the topic travel only using the Bayesian deep ranker with MCDropout due to the limited GPU resources. Based on equation (4.1), there is a parameter t could be tuned to simulate the noise level. When t increases, the difference between two candidates would be narrower and the user's choice tend to be random and vice versa when t decreases. In our experiment, we set t to be [0.3, 1, 2.5].

Noise Level	0.3	1	2.5
Accuracy	0.833	0.828	0.795

Table 5.12: Effect of the noise level of the simulated user on model performance

Table 5.12 shows as the noise level increases, the matched accuracy decreases as expected. However, there is no significant drop when the noise parameter t rises eightfold from 0.3 to 2.5. This indicates that the developed Bayesian deep ranker is robust under noise circumstances.

Chapter 6

Conclusion

The goal of this research is to investigate whether Bayesian deep learning models are effective for the interactive question answering task especially for non-factoid QA. The study followed an in-depth and time-consuming experimental design, evaluating main elements in interactive question answering, including matched accuracy of Bayesian and non-Bayesian models, NDCG@5, the number of interactions, and robustness of models under the noise circumstance. In this chapter, we discuss and reflect our findings. Furthermore, limitations of our study is described and we underline what needs to be done in the future based on our findings.

6.1 Project Status

This dissertation focuses on Bayesian deep learning and how to use them to improve the performance of interactive QA systems. We propose a Bayesian deep interactive ranking approach with different inference methods to improve model performance with fewer labels from the user compared with the previous work. We completed all research aims outlined before and several findings are summarised in line with research aims in section 1.3.1.

The original research aims are as follow.

- Validate whether Bayesian deep learning methods are necessary compared with non-Bayesian deep learning methods.
- Research whether advanced Bayesian deep learning [45, 16] can improve the accuracy of top-ranked answers within fewer interactions(labelled data) compared with shallower Bayesian models.
- Explore performance of different Bayesian deep learning methods.
- Investigate the robustness of Bayesian deep rankers under noisy circumstances.
- Build software for this project, including framework for interactive learning simulations, Bayesian BERT with SWAG and MC Dropout.

For the first aim, we proved Bayesian methods are superior to non-Bayesian methods based on results shown in section 5.2. We compared them on three different datasets, which showed Bayesian deep learning methods significantly outperform conventional deep learning methods with 26% improvements in accuracy in the Topic Apple. This observations is consistent with the previous work [69]. The main improvement comes from the ability to quantify epistemic uncertainty so that Bayesian optimisation could be used to fast and effectively find extremums.

With regard to the second aim, we observed Bayesian deep rankers outperform the shallow ranker across all topics(Table 5.11). We have shown Bayesian deep rankers can achieve better performance

with only 4 interactions while the Bayesian shallow ranker(GPPL-based) needs 10 interactions or even more[71]. Interestingly, conventional deep learning can achieve competitive accuracy with the GPPL-based interactive ranker although its acquisition function is non-Bayesian. This shows that the powerful textual representation ability is also imperative for this task.

For the third research objective, we found the Bayesian deep ranker with MCDropout performs better than the SWAG-based model(Table 5.11). As we analysed before, SWAG needs more epochs to capture the loss geometry and approximate posterior distributions. To speed up the predicting process, we only estimate posterior distributions over the last four layers, which also hurts performance of SWAG.

For the fourth research aim, as shown in Table 5.12, we demonstrated the Bayesian deep ranker with Dropout is robust under noisy circumstance.

Finally, we developed the software for this project. It includes how to train an initial ranker and interact with users to fine-tune Bayesian deep rankers for a given question[1].

Overall, all of these research aims have been achieved via model design(Chapter 3), various experiments and analysis(Chapter 4 and Chapter 5).

6.2 Limitations

There are some limitations that affect our model performance. The first thing is related to simulated users. Simulated users simulate the user's preference based on the utility score of a pair of candidates. However, the utility score is derived from Rouge metric, which only considers literal similarity but ignores semantic similarity. Therefore, It can not fully reflect the real user's preference. Another problem has been discussed by Simpson et al.[71] that the errors real users make tend to be systematic rather than random errors. To further this work, interactions with real users need to be investigated. The second issue is randomness of Monte Carlo sampling. Different random seeds would lead to slightly different sampled weights. The last problem is related to the limited computation resources. For hyperparameters tuning in section 5.1, we empirically set several combinations whose search space could be extended to get more optimal sets if GPU resources are abundant. For the noisy user experiment in section5.3, the Bayesian deep ranker with MCDropout is evaluated. Technically, the SWAG method should be evaluated as well to demonstrate the robustness of deep Bayesian rankers.

6.3 Future work

A human evaluation study is needed to further demonstrate performance of our proposed Bayesian deep rankers. But it should be noted that it needs to be carried out at a large scale due to the variations of user preferences. For the randomness of Monte Carlo sampling, there are two ways could be applied to alleviate this problem. One is to sample multiple times and average the results of all experiments and the other way is to increase the number of sampling to reduce the effect of randomness.

Our findings indicate Bayesian deep learning are effective to quickly select the best answer among a pool of unlabelled candidates based on the user's feedback. Naturally, this method could be generalized to other tasks involving text matching, document ranking or humans' preference. For instance, it can be used in recommendation systems to recommend items based on a few user's feedback or text summarisation task to rank extracted summaries.

In order to find the optimal hyperparameters of Bayesian deep learning models, we split a development dataset at the interaction phase to explore search space. However, in the real scenario, there is unavailable validation dataset for us to tune hyperparameters. The label budget is only available for interactions. How to set the hyperparameters to have a better induction bias without labelling datasets is still an open question. Besides, the way to train an initial ranker could be researched as well since the current

[1]The Github repository: https://github.com/HaishuoFang/BDL

work still requires a lot of in-domain labelled dataset. As mentioned before, in-domain or task-specific continuous pretraining methods could be useful to provide a basic ranker.

In Table 5.7, we show that as the number of samplings increases, the accuracy would be improved but computation time raises as well. How to make predictions of Bayesian deep learning with large amounts of weights more scalable is also a promising field.

This project can be integrated with labelling platforms to create more industrial value. In more detail, the platform is responsible for presenting a pair of candidates to the end user and instructs them to label. The developed Bayesian deep ranker would be used to search the best matched candidate for the user in several interactions.

Bibliography

[1] Ankit Agrawal, Sarsij Tripathi, and Manu Vardhan. Active learning approach using a modified least confidence sampling strategy for named entity recognition. *Progress in Artificial Intelligence*, 10(2):113–128, 2021.

[2] Saleema Amershi, Maya Cakmak, William Bradley Knox, and Todd Kulesza. Power to the people: The role of humans in interactive machine learning. *Ai Magazine*, 35(4):105–120, 2014.

[3] Dana Angluin. Queries and concept learning. *Machine learning*, 2(4):319–342, 1988.

[4] Ahlam Ansari, Moonish Maknojia, and Altamash Shaikh. Intelligent question answering system based on artificial neural network. In *2016 IEEE International Conference on Engineering and Technology (ICETECH)*, pages 758–763. IEEE, 2016.

[5] PVS Avinesh and Christian M Meyer. 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, 2017.

[6] Lisa Bauer, Yicheng Wang, and Mohit Bansal. Commonsense for generative multi-hop question answering tasks. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 4220–4230, Brussels, Belgium, October-November 2018. Association for Computational Linguistics.

[7] Michael Bloodgood. Support vector machine active learning algorithms with query-by-committee versus closest-to-hyperplane selection. In *2018 IEEE 12th International Conference on Semantic Computing (ICSC)*, pages 148–155. IEEE, 2018.

[8] Charles Blundell, Julien Cornebise, Koray Kavukcuoglu, and Daan Wierstra. Weight uncertainty in neural network. In *International Conference on Machine Learning*, pages 1613–1622. PMLR, 2015.

[9] Wenbin Cai, Ya Zhang, and Jun Zhou. Maximizing expected model change for active learning in regression. In *2013 IEEE 13th international conference on data mining*, pages 51–60. IEEE, 2013.

[10] Yllias Chali, Sadid A Hasan, and Shafiq R Joty. Improving graph-based random walks for complex question answering using syntactic, shallow semantic and extended string subsequence kernels. *Information Processing & Management*, 47(6):843–855, 2011.

[11] Yi Chang and Hongbo Deng. *Query Understanding for Search Engines*. Springer.

[12] Davide Chicco. Siamese neural networks: An overview. *Artificial Neural Networks*, pages 73–94, 2021.

[13] David Cohn, Les Atlas, and Richard Ladner. Improving generalization with active learning. *Machine learning*, 15(2):201–221, 1994.

[14] Wanxin Cui, Wei Zhan, Jingjing Yu, Chenfan Sun, and Yangyang Zhang. Face recognition via convolutional neural networks and siamese neural networks. In *2019 International Conference on Intelligent Computing, Automation and Systems (ICICAS)*, pages 746–750. IEEE, 2019.

[15] Ido Dagan and Sean P Engelson. Committee-based sampling for training probabilistic classifiers. In *Machine Learning Proceedings 1995*, pages 150–157. Elsevier, 1995.

[16] John S Denker and Yann LeCun. Transforming neural-net output levels to probability distributions. In *Proceedings of the 3rd International Conference on Neural Information Processing Systems*, pages 853–859, 1990.

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

[18] Sanjay K Dwivedi and Vaishali Singh. Research and reviews in question answering system. *Procedia Technology*, 10:417–424, 2013.

[19] Meire Fortunato, Charles Blundell, and Oriol Vinyals. Bayesian recurrent neural networks. *arXiv preprint arXiv:1704.02798*, 2017.

[20] Atsushi Fujii, Kentaro Inui, Takenobu Tokunaga, and Hozumi Tanaka. Selective sampling for example-based word sense disambiguation. *arXiv preprint cs/9910020*, 1999.

[21] Johannes Fürnkranz and Eyke Hüllermeier. Preference learning and ranking by pairwise comparison. In *Preference learning*, pages 65–82. Springer, 2010.

[22] Yarin Gal and Zoubin Ghahramani. Bayesian convolutional neural networks with bernoulli approximate variational inference. *arXiv preprint arXiv:1506.02158*, 2015.

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

[24] Yarin Gal, Riashat Islam, and Zoubin Ghahramani. Deep bayesian active learning with image data. In *International Conference on Machine Learning*, pages 1183–1192. PMLR, 2017.

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

[26] Stuart Geman, Elie Bienenstock, and René Doursat. Neural networks and the bias/variance dilemma. *Neural computation*, 4(1):1–58, 1992.

[27] Jiafeng Guo, Yixing Fan, Qingyao Ai, and W Bruce Croft. A deep relevance matching model for ad-hoc retrieval. In *Proceedings of the 25th ACM international on conference on information and knowledge management*, pages 55–64, 2016.

[28] Jiafeng Guo, Yixing Fan, Liang Pang, Liu Yang, Qingyao Ai, Hamed Zamani, Chen Wu, W Bruce Croft, and Xueqi Cheng. A deep look into neural ranking models for information retrieval. *Information Processing & Management*, 57(6):102067, 2020.

[29] Suchin Gururangan, Ana Marasović, Swabha Swayamdipta, Kyle Lo, Iz Beltagy, Doug Downey, and Noah A Smith. Don't stop pretraining: adapt language models to domains and tasks. *arXiv preprint arXiv:2004.10964*, 2020.

[30] Helia Hashemi, Mohammad Aliannejadi, Hamed Zamani, and W Bruce Croft. Antique: A non-factoid question answering benchmark. In *European Conference on Information Retrieval*, pages 166–173. Springer, 2020.

[31] Matthew D Hoffman, David M Blei, Chong Wang, and John Paisley. Stochastic variational inference. *Journal of Machine Learning Research*, 14(5), 2013.

[32] Alex Holub, Pietro Perona, and Michael C Burl. Entropy-based active learning for object recognition. In *2008 IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops*, pages 1–8. IEEE, 2008.

[33] Po-Sen Huang, Xiaodong He, Jianfeng Gao, Li Deng, Alex Acero, and Larry Heck. Learning deep structured semantic models for web search using clickthrough data. In *Proceedings of the 22nd ACM international conference on Information & Knowledge Management*, pages 2333–2338, 2013.

[34] Eyke Hüllermeier and Willem Waegeman. Aleatoric and epistemic uncertainty in machine learning: An introduction to concepts and methods. *Machine Learning*, 110(3):457–506, 2021.

[35] Sathish Reddy Indurthi, Seunghak Yu, Seohyun Back, and Heriberto Cuayáhuitl. Cut to the chase: A context zoom-in network for reading comprehension. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 570–575, Brussels, Belgium, October-November 2018. Association for Computational Linguistics.

[36] Pavel Izmailov, Dmitrii Podoprikhin, Timur Garipov, Dmitry Vetrov, and Andrew Gordon Wilson. Averaging weights leads to wider optima and better generalization. *arXiv preprint arXiv:1803.05407*, 2018.

[37] Seho Kee, Enrique del Castillo, and George Runger. Query-by-committee improvement with diversity and density in batch active learning. *Information Sciences*, 454:401–418, 2018.

[38] David D Lewis and William A Gale. A sequential algorithm for training text classifiers. In *SIGIR'94*, pages 3–12. Springer, 1994.

[39] Mingkun Li and Ishwar K Sethi. Confidence-based active learning. *IEEE transactions on pattern analysis and machine intelligence*, 28(8):1251–1261, 2006.

[40] Chin-Yew Lin. Rouge: A package for automatic evaluation of summaries. In *Text summarization branches out*, pages 74–81, 2004.

[41] Tianyang Lin, Yuxin Wang, Xiangyang Liu, and Xipeng Qiu. A survey of transformers. *arXiv preprint arXiv:2106.04554*, 2021.

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

[43] Tie-Yan Liu. Learning to rank for information retrieval. 2011.

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

[45] David JC MacKay. A practical bayesian framework for backpropagation networks. *Neural computation*, 4(3):448–472, 1992.

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

[47] Stephan Mandt, Matthew D Hoffman, and David M Blei. Stochastic gradient descent as approximate bayesian inference. *arXiv preprint arXiv:1704.04289*, 2017.

[48] Andrew Kachites McCallumzy and Kamal Nigamy. Employing em and pool-based active learning for text classification. In *Proc. International Conference on Machine Learning (ICML)*, pages 359–367. Citeseer, 1998.

[49] Dan Moldovan and Mihai Surdeanu. On the role of information retrieval and information extraction in question answering systems. In *International Summer School on Information Extraction*, pages 129–147. Springer, 2002.

[50] Paul Neculoiu, Maarten Versteegh, and Mihai Rotaru. Learning text similarity with siamese recurrent networks. In *Proceedings of the 1st Workshop on Representation Learning for NLP*, pages 148–157, 2016.

[51] Hieu T Nguyen and Arnold Smeulders. Active learning using pre-clustering. In *Proceedings of the twenty-first international conference on Machine learning*, page 79, 2004.

[52] Kamal Nigam and Andrew McCallum. Pool-based active learning for text classification. In *Conference on Automated Learning and Discovery (CONALD)*, 1998.

[53] Yuzhen Niu, Dong Huang, Yiqing Shi, and Xiao Ke. Siamese-network-based learning to rank for no-reference 2d and 3d image quality assessment. *IEEE Access*, 7:101583–101595, 2019.

[54] Natalia Ostapuk, Jie Yang, and Philippe Cudré-Mauroux. Activelink: deep active learning for link prediction in knowledge graphs. In *The World Wide Web Conference*, pages 1398–1408, 2019.

[55] Gerhard Paass and Jörg Kindermann. Bayesian query construction for neural network models. In *Proceedings of the 7th International Conference on Neural Information Processing Systems*, pages 443–450, 1994.

[56] Emmeleia Panagiota Mastoropoulou. Enhancing deep active learning using selective self-training for image classification, 2019.

[57] Álvaro Peris and Francisco Casacuberta. 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, October 2018. Association for Computational Linguistics.

[58] Robert Pinsler, Jonathan Gordon, Eric Nalisnick, and José Miguel Hernández-Lobato. Bayesian batch active learning as sparse subset approximation. *arXiv preprint arXiv:1908.02144*, 2019.

[59] Avinesh P.V.S and Christian M. Meyer. 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, July 2017. Association for Computational Linguistics.

[60] Filip Radlinski, Krisztian Balog, Bill Byrne, and Karthik Krishnamoorthi. Coached conversational preference elicitation: A case study in understanding movie preferences. 2019.

[61] Nicholas Roy and Andrew McCallum. Toward optimal active learning through monte carlo estimation of error reduction. *ICML, Williamstown*, 2:441–448, 2001.

[62] Andreas Rücklé, Nafise Sadat Moosavi, and Iryna Gurevych. 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, 2019.

[63] Raphael Schumann and Ines Rehbein. Active learning via membership query synthesis for semi-supervised sentence classification. In *Proceedings of the 23rd conference on computational natural language learning (CoNLL)*, pages 472–481, 2019.

[64] Nabeel Seedat and Christopher Kanan. Towards calibrated and scalable uncertainty representations for neural networks. *arXiv preprint arXiv:1911.00104*, 2019.

[65] Burr Settles. Active learning literature survey. 2009.

[66] Burr Settles and Mark Craven. An analysis of active learning strategies for sequence labeling tasks. In *Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing*, pages 1070–1079, 2008.

[67] H Sebastian Seung, Manfred Opper, and Haim Sompolinsky. Query by committee. In *Proceedings of the fifth annual workshop on Computational learning theory*, pages 287–294, 1992.

[68] Yanyao Shen, Hyokun Yun, Zachary C Lipton, Yakov Kronrod, and Animashree Anandkumar. Deep active learning for named entity recognition. *arXiv preprint arXiv:1707.05928*, 2017.

[69] Aditya Siddhant and Zachary C Lipton. Deep bayesian active learning for natural language processing: Results of a large-scale empirical study. *arXiv preprint arXiv:1808.05697*, 2018.

[70] Yawar Siddiqui, Julien Valentin, and Matthias Nießner. Viewal: Active learning with viewpoint entropy for semantic segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9433–9443, 2020.

[71] Edwin Simpson, Yang Gao, and Iryna Gurevych. Interactive text ranking with bayesian optimisation: a case study on community qa and summarisation. *arXiv preprint arXiv:1911.10183*, 2019.

[72] Marco Antonio Calijorne Soares and Fernando Silva Parreiras. A literature review on question answering techniques, paradigms and systems. *Journal of King Saud University-Computer and Information Sciences*, 32(6):635–646, 2020.

[73] Paolo Viappiani and Craig Boutilier. Optimal bayesian recommendation sets and myopically optimal choice query sets. In *NIPS*, pages 2352–2360, 2010.

[74] Shengxian Wan, Yanyan Lan, Jun Xu, Jiafeng Guo, Liang Pang, and Xueqi Cheng. Match-srnn: Modeling the recursive matching structure with spatial rnn. *arXiv preprint arXiv:1604.04378*, 2016.

[75] Liu Yang, Rong Jin, and Rahul Sukthankar. Bayesian active distance metric learning. *arXiv preprint arXiv:1206.5283*, 2012.

[76] Xitong Yang. Understanding the variational lower bound, 2017.

[77] Wen-tau Yih and Hao Ma. Question answering with knowledge base, web and beyond. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Tutorial Abstracts*, pages 8–10, San Diego, California, June 2016. Association for Computational Linguistics.

[78] Chiyuan Zhang, Samy Bengio, Moritz Hardt, Benjamin Recht, and Oriol Vinyals. Understanding deep learning (still) requires rethinking generalization. *Communications of the ACM*, 64(3):107–115, 2021.

[79] Ye Zhang, Matthew Lease, and Byron Wallace. Active discriminative text representation learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 31, 2017.

[80] Jingbo Zhu, Huizhen Wang, Eduard Hovy, and Matthew Ma. Confidence-based stopping criteria for active learning for data annotation. *ACM Transactions on Speech and Language Processing (TSLP)*, 6(3):1–24, 2010.

Appendix A

Github Repo

Github Repository: https://github.com/HaishuoFang/BDL