Literature Review

*Intro*

Since the crux of this project is to assess the suitability of applying Bayesian deep learning (BDL) techniques to passage ranking (PR) problems, this chapter explores the relevant literature that discusses previous approaches to passage ranking solutions. Once this assessment has been done, we will then also examine literature that assesses BDL as opposed to classical deep learning techniques.

*Text Summarisation Models*

*Previous works*

* Interactive Learning

Interactive learning is a machine learning workflow involving directed experimentation with inputs and output \cite{Amershi14}. Rapid change in response to user input facilitates interactive inspection on impact of user input. This workflow format is commonly used to solve NLP problems; related works include literature in passage ranking (PR) of generated text in the context of translations, question answering and text summarisation \cite{Peris18, Lin17, PVS17}. These works had a focus on interactionally-expensive uncertainty sampling to learn the rankings of \emph{all} candidate passages \cite{Simpson19}. Gao et al. \cite{Gao18} researched how to reduce the number of user interactions for uncertainty sampling techniques with some success using an active learner. A positive step towards reasonable interactive learning.

Simpson et al \cite{Simpson19} take an alternative approach to uncertainty sampling by proposing a Bayesian optimisation (BO) strategy instead \cite{Simpson19}. With Gaussian process (GPs) displaying some success in error reduction for NLP tasks with noisy labels \cite{Cohn13, Beck14}, Simpson and Gurevych \cite{Simpson18} proposed using Gaussian process preference learning (GPPL) with uncertainty sampling. This approach has been further built upon by Simpson et al. \cite{Simpson19} to a BO framework. This approach showed a markable improvement in the accuracy of chosen answers in a community question answering (cQA) task with a small number of interactions required \cite{Simpson19}. The methodology used Expected Improvement (IMP) as the acquisition function for AL which twisted the focus of optimisation to find the best candidate, as opposed to the ranks of all candidates \cite{Simpson19}. The switch to exploitation of promising candidates showed to be massively influential on performance \cite{Simpson19}. Simpson et al. \cite{Simpson19} furthered the performance enhancement gained from using the BO framework by using prior predictions from a deep learner as an informative prior for GPPL \cite{Simpson19} to address the cold-start problem for recommender systems \cite{Bobadilla12}.

* Active Learning

*What is AL?*

Alongside unsupervised and supervised learning, active learning (AL) is a machine learning framework whereby queries are asked of an oracle – such as a human annotator – in the form of labelling unlabelled observations \cite{Settles09}. The active interactions with oracles allow better performance with few labelled data points. AL is beneficial in the cases where labelled data is scarce due to high costs; for speech recognition problems \cite{Zhu05} details a scale factor of ten times between the length of a speech extract and the time taken to annotate such as extract.

*AL Scenarios*

Settles \cite{Settles09} describes three scenarios that are considered in literature to categories AL problems.

\paragraph{Membership query synthesis.} Labels are requested by the learner for any unlabelled instance in the input space. This includes queries that are generated as if for the first time rather than from some causal distribution \cite{Angluin88}. A considerable limitation of this scenario occurs when the oracle is a human annotator. Baum and Lang \cite{Baum92} employed membership query learning to classify handwritten characters using a human oracle. They found that many query images that were generated were unrecognisable symbols. This limitation could feasibly produce nonsense summaries when tasked with a PR situation; something we should be cautious of.

\paragraph{Stream-based selective sampling.} In this setting, unlabelled observations are selected sequentially and the learner determines whether to query or discard it to reduce annotation effort \cite{Cohn94}. This is under the major assumption that acquiring unlabelled instances is low-cost since the learner needs to be able to decide it can discard the unlabelled observation with minimal opportunity cost. The most common way of defining if a sample should be queried or discarded is by creating a \emph{version space} \cite{Mitchell82} using two models with different parameter choices; for those instances that the models agree on, we can discard as there is little uncertainty. However, with regards to the cases of disagreement, these unlabelled instances fall in the region of uncertainty \cite{Settles09}. This region of uncertainty is computationally expensive to calculate; thus, it is common to use approximations in practice \cite{Seung92, Cohn94, Dasgupta07}.

\paragraph{Pool-based active learning.} A common approach for many real world examples such as text classification \cite{Lewis94}, information extraction \cite{Thompson99} and speech recognition \cite{Tur05} since it is common to find large groups of unlabelled data collected at once. The \emph{pool-based active learning} workflow starts with a learner trained on a small set of labelled data, $ labelled\_instances $, which is then used to \emph{greedily} rank instances in a large collection of unlabelled instances, $unlaballed\_instances$ \cite{Lewis94}. The highest-ranked instance is then labelled by an oracle and then used within the learner retrain. In comparison to a stream-based active learner, a greater computational cost is associated with a pool-based learner since it ranks the entire set $unlaballed\_instances$ before making a query as opposed to making sequential decisions.

*Acquisition Functions*

Whilst introducing \emph{active learning}, a lot is spoken about measuring the usefulness of each instance and whether to query it. This is measured using \emph{acquisition functions}.

*DEEP LEARNING*

*What is deep learning?*

Deep learning methods form a subset of machine learning, based on neural networks with at least three hidden layers. These techniques have dramatically increased capabilities of model recognition in many domains including visual object recognition, question answering and text summarisation \cite{Lecun15, Sharma18, Azar17}.

[NN Deep Learning Image]

*MAP – what is done classically.*

In classical training, one typically uses maximum a-posteriori (MAP) optimisation to choose the set of parameters, $\hat{w}$, for our model that maximises the posterior probability from our parameter distribution \cite{Wilson20}. MAP does not require computationally-costly calculations of the marginal distribution \cite{Hero14}; however, since MAP is a point estimate, it cannot be fully considered a Bayesian approach \cite{Hero15}.

[Hero15 plot]

* + Pre-trained Models - BERT

Pre-trained, deep learning, language models are useful in unsupervised learning problems due to the lack of major architectural modifications required and the high performance levels that are delivered \cite{Mridha19}.

*What is BERT?*

One popular pre-trained language model is the Bidirectional Encoder Representations from Transformers (BERT) which takes an entire sequence of words, bidirectionally, to produce significantly improved results. The input is augmented by three embeddings – position, segment and token embeddings – and padded by a [CLS] token at the beginning of the first sentence to ensure BERT has lots of useful information \cite{Navin21}.

BERT is trained on two tasks in parallel: Masked Language Modelling, prediction of hidden words in sentences, and Next Sentence Prediction \cite{Navin21}. However BERT can be applied to many NLP tasks \cite{Mridha19} such as question answering and text classification tasks with some minor fine-tuning; we add an additional small layer on the top of the transformer output for the [CLS] token \cite{Navin21} to adapt the core model to different tasks.

[classification example]

*How does it do?*

Recent publications have found BERT-based models \cite{Devlin18} to be extremely effective when tasked with passage ranking situations across the question answering and text summarisation disciplines \cite{Xu19, Qiao19}. Xu et al. \cite{Xu19} explored a query-passage set up when applying BERT to cQA such that the BERT final hidden state fed into an MLP module to produce relevance scores in a supervised way. Since this technique outperformed the baseline, it may be a useful structure to consider adapting to the text summarisation domain.

The limitation of utilising an interactive learning framework such as one outlined by Simpson et al. \cite{Simpson19} as that it does not utilise the vast performance capabilities of newer, pre-trained techniques such as BERT. Although the framework presented does limit the number of interactions required from a user – allowing the user to tailor the summary – Ein-Dor et al. \cite{EinDor20} look to take this idea further with the incorporation of a BERT component in an AL framework.

* Deep Active Learning

Ein-Dor et al. \cite{EinDor20} developed a framework that used an AL approach with BERT-based classification. Zhang and Zhang also explored an alike ensemble of AL strategies \cite{Zhang19}; however, the task is less relatable to PR since framework proposed by Ein-Dor et al. had experimentation on data with high class imbalance, scarce labelling and a small annotation budget \cite{EinDor20}, attributes of an interactive PR context. This structure consisted of pool-based AL in batch mode in conjunction with BERT as the classification scheme. Different AL strategies were examined – Monte-Carlo Dropout (MCD) \cite{Gal15}, a Bayesian approach, and Discriminative Active Learning (DAL) \cite{Gissin19} – with Al proving an excellent boost to helping BERT emerge from its poor initial model \cite{EinDor19}. Although DAL would not be appropriate for the PR context due to its focus on querying batches, using MCD as a strategy is a technique we could consider.

* *Bayesian Deep Learning*

*What is BDL?*

Bayesian Deep Learning is a deep learning approach which uses a probabilistic framework – whether that be in the model acquisition function or model parameters – to improve model performance. Bayesian acquisition functions are something we have mentioned previously; however, with regards to a probabilistic approach to selection of model parameters, $\theta$, marginalisation is used to replace optimisation. This is so we can utilise the effect of several models using different $\theta$ with probability distribution $p(\theta)$.

To allow us to marginalise over $\theta$, we require Bayes Theorem to link the \emph{prior distribution}, $p(w)$, for weights $w$; the likelihood, $p(D|w)$ of such weights being suitable for data, $D$; and the \emph{posterior distribution}, $p(w|D)$, of the weights.

[Bayes Formula]

The marginalisation stage forms the integral over all possible $\theta$ on the numerator. The posterior distribution is incredibly useful to calculate the predictive distribution (or marginal probability distribution) of the output. The \emph{predictive distribution}, $p(y|D, x)$, defines the probability of label $y$ given additional input $x$ and dataset $D$ \cite{Izmailov20}.

[formula]

This integral is called \emph{Bayesian Model Averaging (BMA)} and can be thought of as the weighted average (using probability distributions) of all parameters and defines the probability for label $y$ given input $x$ and data $D$ \cite{Izmailov20}. Wilson and Izmailov \cite{Izmailov20} argue that using a BMA increases accuracy as well as obtaining a realistic expression of uncertainty with classical neural networks exhibiting overconfident predictions \cite{Xu19}. Unfortunately, calculating the posterior distribution is a computationally expensive task, due to the marginalisation step in the denominator, so approximate posterior distributions are used.

*Strategies to approximate posterior distribution*

Firstly, Wilson and Izamailov \cite{Izmailov20} comment that taking a selection of possible $\theta$ and combining the resulting models to approximate BMA – named Monte Carlo approximation – evocative of frequentist deep ensembles. However, there are modern approaches one can take.

A common practical method is using Monte Carlo Markov Chains (MCMC) to approximate the posterior \cite{Izmailov20}. MCMCs are used to approximate variable distributions for an idealised system \cite{Brooks11} and there are two common algorithms that have been tailored to approximate posterior distributions: Gibbs Sampling and the Metropolis-Hastings Algorithm. However, Gibbs Sampling is not appropriate for neural networks with conditional posterior distributions due to the interdependency of weights \cite{Neal95}. Simple forms of the Metropolis-Hasting algorithm (MH) can be more appropriate; however, again due to the high interdependence of states, MH can be costly and prone to random walks. Duane et al. \cite{Daune87) propose an alternative \emph{hybrid Monte Carlo} which is a combination of MH with sampling techniques from dynamical simulation.

Second, Graves \cite{Graves11} proposed fitting a Gaussian variational posterior approximation over the weights of neural networks and optimising over the weights to ensure the variational distribution is as good an estimate of the posterior distribution as possible. This method works well for networks of a moderate size, but supplies training difficulties when working with larger architectures \cite{He15}.

Thirdly, Gal and Ghahramani \cite{Gal15} present Monte Carlo Dropout (MCD); a dropout framework which integrates stochasticity into a neural network, by randomly removing weights \emph{during training}. We can interpret dropout as approximate Bayesian inference, leading to a range of weightings. It is intuitive to see the link between this and sampling weights from a posterior to approximate a predictive distribution.

[MCD image]

Denoting the neural network weight matrices for layer $i$ as $W\_i$ alongside input and output sets $X, Y$, we again suffer from an intractable posterior distribution $p(y|x, X, Y)$. Thus $q(\omega)$ is as an approximation defined as below

[formula]

A simple Bernoulli distribution is used to determine which states are set to zero given some probability $p\_i$ and variational weights $M\_i$. Note here that $z\_{I,j}$ denotes unit $j$ in layer $i-1$.

To obtain the model uncertainty obtained through dropout in neural networks, we take our approximate predictive distribution given by [formula]. Though $T$ sample sets of realisations from our posterior distribution $z\_{I,j}$, we get $T$ weight matrices ${W\_t…}$, we get the following estimate by which we call our Monte Carlo Dropout.

[Formula]

Another popular technique is \emph{Stochastic Weight Averaging – Gaussian (SWAG)} \cite{Maddox19}. This builds on the idea of \emph{Stochastic Weight Averaging (SWA)} which combines weights of the same neural network at different stages in training \cite{Dmitrii18}. SWAG uses Stochastic Gradient Descent (SGD) information to estimate the shape of the posterior distribution by fitting a Gaussian distribution to the first two moments of the SGD iterate \cite{Maddox19}. We use these fitted Gaussian distributions for BMA. The benefits of SWAG are grounded in its practicality, stability and accuracy which are essential attributes when working with large neural networks \cite{Maddox19}.

Wang and Yeung \cite{Wang20} posit a general BDL probabilistic framework made up of two components: a perception component and a task-specific component. The perceptron component would have multiple, non-linear layers in a chain structure to represent the probabilistic element of the deep learning model. The task-specific component is often more complex in structure and aim to learn the more intricate relationships between parameters. Due to this structure, Wang and Yeung \cite {Wang20} outline three types of parameters: perception parameters, hinge parameters and task parameters. With perception and task parameters referring to the parameters of the two components, the hinge parameters look after combining the two components smoothly.