

Quantifying Uncertainties in Natural Language Processing Tasks

Yijun Xiao, William Yang Wang

University of California, Santa Barbara

{yijunxiao,william}@cs.ucsb.edu

Abstract

Reliable uncertainty quantification is a first step towards building explainable, transparent, and accountable artificial intelligent systems. Recent progress in Bayesian deep learning has made such quantification realizable. In this paper, we propose novel methods to study the benefits of characterizing model and data uncertainties for natural language processing (NLP) tasks. With empirical experiments on sentiment analysis, named entity recognition, and language modeling using convolutional and recurrent neural network models, we show that explicitly modeling uncertainties is not only necessary to measure output confidence levels, but also useful at enhancing model performances in various NLP tasks.

Introduction

With advancement of modern machine learning algorithms and systems, they are applied in various applications that, in some scenarios, impact human wellbeing. Many of such algorithms learn black-box mappings between input and output. If the overall performance is satisfactory, these learned mappings are assumed to be correct and are used in real-life applications. It is hard to quantify how confident a certain mapping is with respect to different inputs. These deficiencies cause many AI safety and social bias issues with the most notable example being failures of auto-piloting systems. We need systems that can not only learn accurate mappings, but also quantify confidence levels or uncertainties of their predictions. With uncertainty information available, many issues mentioned above can be effectively handled.

There are many situations where uncertainties arise when applying machine learning models. First, we are uncertain about whether the structure choice and model parameters can best describe the data distribution. This is referred to as *model uncertainty*, also known as epistemic uncertainty. Bayesian neural networks (BNN) (Buntine and Weigend 1991; Denker and Lecun 1991; MacKay 1992; 1995; Neal 2012) is one approach to quantify uncertainty associated with model parameters. BNNs represent all model weights as probability distributions over possible values instead of fixed scalars. In this setting, learned mapping of a BNN model must be robust under different samples of

weights. We can easily quantify model uncertainties with BNNs by, for example, sampling weights and forward inputs through the network multiple times. Quantifying model uncertainty using a BNN learns potentially better representations and predictions due to the ensemble nature of BNNs. It is also showed in (Blundell et al. 2015) that it is beneficial for exploration in reinforcement learning (RL) problems such as contextual bandits.

Another situation where uncertainty arises is when collected data is noisy. This is often the case when we rely on observations and measurements to obtain the data. Even when the observations and measurements are precise, noises might exist within the data generation process. Such uncertainties are referred to as *data uncertainties* in this paper and is also called aleatoric uncertainty (Der Kiureghian and Ditlevsen 2009). Depending on whether the uncertainty is input independent, data uncertainty is further divided into *homoscedastic uncertainty* and *heteroscedastic uncertainty*. Homoscedastic uncertainty is the same across the input space which can be caused by systematic observation noise. Heteroscedastic uncertainty, on the contrary, is dependent on the input. For example, when predicting the sentiment of a Yelp review, single-word review “good” is possible to have 3, 4 or 5-star ratings while a lengthened review with strong positive emotion phrases is definitely a 5-star rating. In the rest of the paper, we also refer to heteroscedastic uncertainty as input-dependent data uncertainty.

Recently, there are increasing number of studies investigating the effects of quantifying uncertainties in different applications (Kendall, Badrinarayanan, and Cipolla 2015; Gal and Ghahramani 2016b; Kendall and Gal 2017; Zhu and Laptev 2017). In this paper, we focus on exploring the benefits of quantifying both model and data uncertainties in the context of various natural language processing (NLP) tasks. Specifically, we study the effects of quantifying model uncertainty and input-dependent data uncertainty in sentiment analysis, named entity recognition, and language modeling tasks. We show that there is a potential performance increase when including both uncertainties in the model. We also analyze the characteristics of the quantified uncertainties.

The main contributions of this work are:

1. We mathematically define model and data uncertainties via the law of total variance;

2. Our empirical experiments show that by accounting for model and data uncertainties, we observe significant improvements in three important NLP tasks;
3. We show that our model outputs higher data uncertainties for more difficult predictions in sentiment analysis and named entity recognition tasks.

Related Work

Bayesian Neural Networks

Modern neural networks are parameterized by a set of model weights \mathbf{W} . In the supervised setting, for a dataset $D = \{(\mathbf{x}_1, y_i)\}_{i=1}^N$, a point estimate for \mathbf{W} is obtained by maximizing certain objective function. Bayesian neural networks (Buntine and Weigend 1991; Denker and Lecun 1991; MacKay 1992; 1995; Neal 2012) introduce model uncertainties by putting a prior on the network parameters $p(\mathbf{W})$. Bayesian inference is adopted in training aiming to find the posterior distribution of the parameters $p(\mathbf{W}|D)$ instead of a point estimate. This posterior distribution describes possible values for the model weights given the dataset. Predictive function $f^{\mathbf{W}}(\mathbf{x})$ is used to predict the corresponding y value. Given the posterior distribution for \mathbf{W} , the function is marginalized over \mathbf{W} to obtain the expected prediction.

Exact inference for BNNs is rarely available given the complex nonlinear structures and high dimension of model parameters \mathbf{W} of modern neural networks. Various approximate inference methods are proposed (Graves 2011; Hernández-Lobato and Adams 2015; Blundell et al. 2015; Gal and Ghahramani 2016a). In particular, Monte Carlo dropout (MC dropout) (Gal and Ghahramani 2016a) requires minimum modification to the original model. Dropouts are applied between nonlinearity layers in the network and are activated at test time which is different from a regular dropout. They showed that this process is equivalent to variational Bayesian approximation where the approximating distribution is a mixture of a zero mean Gaussian and a Gaussian with small variances. When sampling dropout masks, model outputs can be seen as samples from the posterior predictive function $f^{\widehat{\mathbf{W}}}(\mathbf{x})$ where $\widehat{\mathbf{W}} \sim p(\mathbf{W}|D)$. As a result, model uncertainty can be approximately evaluated by finding the variance of the model outputs from multiple forward passes.

Uncertainty Quantification

Model uncertainty can be quantified using BNNs which captures uncertainty about model parameters. Data uncertainty describes noises within the data distribution. When such noises are homogeneous across the input space, it can be modeled as a parameter. In the cases where such noises are input-dependent, i.e. observation noise varies with input \mathbf{x} , heteroscedastic models (Nix and Weigend 1994; Le, Smola, and Canu 2005) are more suitable.

Recently, quantifications of model and data uncertainties are gaining researchers' attentions. Probabilistic pixel-wise semantic segmentation has been studied in (Kendall, Badrinarayanan, and Cipolla 2015); Gal and Ghahramani (2016b) studied model uncertainty in recurrent neural networks in

the context of language modeling and sentiment analysis; Kendall and Gal (2017) researched both model and data uncertainties in various vision tasks and achieved higher performances; Zhu and Laptev (2017) used similar approaches to perform time series prediction and anomaly detection with Uber trip data. This study focuses on the benefits of quantifying model and data uncertainties with popular neural network structures on various NLP tasks.

Methods

First of all, we start with the law of total variance. Given an input variable x and its corresponding output variable y , the variance in y can be decomposed as:

$$\text{Var}(y) = \text{Var}(\mathbb{E}[y|x]) + \mathbb{E}[\text{Var}(y|x)] \quad (1)$$

We mathematically define model uncertainty and data uncertainty as:

$$U_m(y|x) = \text{Var}(\mathbb{E}[y|x]) \quad (2)$$

$$U_d(y|x) = \mathbb{E}[\text{Var}(y|x)] \quad (3)$$

where U_m and U_d are model and data uncertainties respectively. We can see that both uncertainties partially explain the variance in the observation. In particular, model uncertainty explains the part related to the mapping process $\mathbb{E}[y|x]$ and data uncertainty describes the variance inherent to the conditional distribution $\text{Var}(y|x)$. By quantifying both uncertainties, we essentially are trying to explain different parts of the observation noise in y .

In the following sections, we introduce the methods employed in this study to quantify uncertainties.

Model Uncertainty

Recall that Bayesian neural networks aim to find the posterior distribution of \mathbf{W} given the dataset $D = \{(\mathbf{x}_1, y_i)\}_{i=1}^N$. We also specify the data generating process in the regression case as:

$$y|\mathbf{W} \sim \mathcal{N}(f^{\mathbf{W}}(\mathbf{x}), \sigma^2) \quad (4)$$

With the posterior distribution $p(\mathbf{W}|D)$, given a new input vector \mathbf{x}^* , the prediction is obtained by marginalizing over the posterior:

$$p(y^*|\mathbf{x}^*, D) = \int_{\mathbf{W}} p(y^*|f^{\mathbf{W}}(\mathbf{x}^*)) p(\mathbf{W}|D) d\mathbf{W} \quad (5)$$

As exact inference is intractable in this case, we can use variational inference approach to find an approximation $q_{\theta}(\mathbf{W})$ to the true posterior $p(\mathbf{W}|D)$ parameterized by a different set of weights θ where the Kullback-Leibler (KL) divergence of the two distributions is minimized.

There are several variational inference methods proposed for Bayesian neural networks (Hernández-Lobato and Adams 2015; Blundell et al. 2015; Gal and Ghahramani 2016a). In particular, dropout variational inference method (Gal and Ghahramani 2016a), when applied to models with dropout layers, requires no retraining and can be applied with minimum changes. The only requirement is dropouts have to be added between nonlinear layers. At test time,

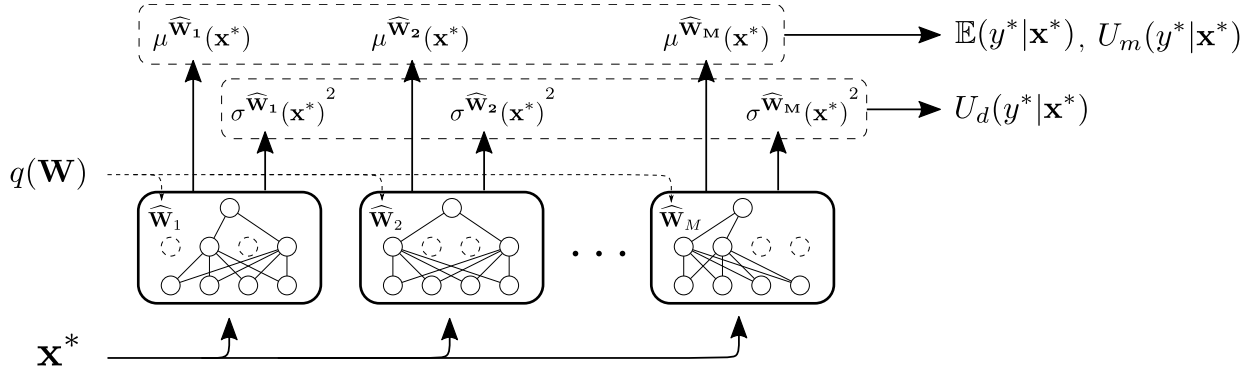


Figure 1: Illustration of the evaluation process of predicted output and both model uncertainty and data uncertainty. $\mathbb{E}(y^*|\mathbf{x}^*)$ denotes the expected value of model prediction; $U_m(y^*)$ is the model uncertainty with respect to the output; $U_d(y^*)$ is the input-dependent data uncertainty. Dotted arrows represent sampling processes.

dropouts are activated to allow sampling from the approximate posterior. We use MC dropout in this study to evaluate model uncertainty.

At test time, we have the optimized approximated posterior $q(\mathbf{W})$. Prediction distribution can be approximated by switching $p(\mathbf{W}|D)$ to $q(\mathbf{W})$ in Equation 5 and perform Monte Carlo integration as follows:

$$\mathbb{E}(y^*|\mathbf{x}^*) \approx \frac{1}{M} \sum_{j=1}^M f^{\widehat{\mathbf{W}}_j}(\mathbf{x}^*) \quad (6)$$

Predictive variance can also be approximated as:

$$\text{Var}(y^*) \approx \frac{1}{M} \sum_{j=1}^M f^{\widehat{\mathbf{W}}_j}(\mathbf{x}^*)^2 - \mathbb{E}(y^*|\mathbf{x}^*)^2 + \sigma^2 \quad (7)$$

where $\widehat{\mathbf{W}}_j$ is sampled from $q(\mathbf{W})$.

Note here σ^2 is the inherent noise associated with the inputs which is homogeneous across the input space. This is often considered by adding a weight decay term in the loss function. We will discuss the modeling of input-dependent data uncertainty in the next section. The rest part of the variance arises because of the uncertainty about the model parameters \mathbf{W} . We use this to quantify model uncertainty in the study, i.e.:

$$U_m(y^*|\mathbf{x}^*) = \frac{1}{M} \sum_{j=1}^M f^{\widehat{\mathbf{W}}_j}(\mathbf{x}^*)^2 - \mathbb{E}(y^*|\mathbf{x}^*)^2 \quad (8)$$

Data Uncertainty

Data uncertainty can be either modeled homogeneous across input space or input-dependent. We take the second option and make the assumption that data uncertainty is dependent on the input. To achieve this, we need to have a model that not only predicts the output values, but also estimates the output variances given some input. In other words, the model needs to give an estimation of $\text{Var}(y|x)$ mentioned in Equation 3.

Denote $\mu(\mathbf{x})$ and $\sigma(\mathbf{x})$ as functions parameterized by \mathbf{W} that calculate output mean and standard deviation for input \mathbf{x} (in practice, logarithm of the variance is calculated for an improvement on stability). We make the following assumption on the data generating process:

$$y \sim \mathcal{N}(\mu(\mathbf{x}), \sigma(\mathbf{x})^2) \quad (9)$$

Given the setting and the assumption, the negative data log likelihood can be written as follows:

$$\begin{aligned} \mathcal{L}_{\text{rgs}}(\mathbf{W}) &= -\frac{1}{N} \sum_{i=1}^N \log p(y_i | \mu(\mathbf{x}_i), \sigma(\mathbf{x}_i)) \\ &= \frac{1}{N} \sum_{i=1}^N \left(\frac{1}{2} \left| \frac{y_i - \mu(\mathbf{x}_i)}{\sigma(\mathbf{x}_i)} \right|^2 + \right. \\ &\quad \left. \frac{1}{2} \log \sigma(\mathbf{x}_i)^2 + \frac{1}{2} \log 2\pi \right) \end{aligned} \quad (10)$$

Comparing Equation 10 to a standard mean squared loss used in regression, we can see that the model encourages higher variances estimated for inputs where the predicted mean $\mu(\mathbf{x}_i)$ is more deviated from the true observation y_i . On the other hand, a regularization term on the $\sigma(\mathbf{x}_i)$ prevents the model from estimating meaninglessly high variances for all inputs. Equation 10 is referred to as learned loss attenuation in (Kendall and Gal 2017).

While Equation 10 works desirably for regression, it is based on the assumption that $y \sim \mathcal{N}(\mu(\mathbf{x}), \sigma(\mathbf{x})^2)$. This assumption clearly does not hold in the classification context. We can however adapt the same formulation in the logit space. In detail, define $\boldsymbol{\mu}(\mathbf{x})$ and $\boldsymbol{\sigma}(\mathbf{x})$ as functions that maps input \mathbf{x} to the logit space. Logit vector is sampled and thereafter transformed into probabilities using softmax operation. This process can be described as:

$$\mathbf{u} \sim \mathcal{N}(\boldsymbol{\mu}(\mathbf{x}), \text{diag}(\boldsymbol{\sigma}(\mathbf{x})^2)) \quad (11)$$

$$\mathbf{p} = \text{softmax}(\mathbf{u}) \quad (12)$$

$$y \sim \text{Categorical}(\mathbf{p}) \quad (13)$$

Corpus	Size	Average Tokens	$ V $	Classes	Class Distribution
Yelp 2013	335,018	151.6	211,245	5	.09/.09/.14/.33/.36
Yelp 2014	1,125,457	156.9	476,191	5	.10/.09/.15/.30/.36
Yelp 2015	1,569,264	151.9	612,636	5	.10/.09/.14/.30/.37
IMDB	348,415	325.6	115,831	10	.07/.04/.05/.05/.08/.11/.15/.17/.12/.18

Table 1: Summaries of Yelp 2013/2014/2015 and IMDB datasets. $|V|$ represents the vocabulary size.

where $\text{diag}()$ function takes a vector and output a diagonal matrix by putting the elements on the main diagonal. Note here in Equation 13, y is a single label. This formulation can be easily extended to multi-way Categorical labels.

During training, we seek to maximize the expected data likelihood. Here we approximate the expected distribution for \mathbf{p} using Monte Carlo approximation as follows:

$$\mathbf{u}^{(k)} \sim \mathcal{N}(\boldsymbol{\mu}(\mathbf{x}), \text{diag}(\boldsymbol{\sigma}(\mathbf{x})^2)) \quad (14)$$

$$\mathbb{E}[\mathbf{p}] \approx \frac{1}{K} \sum_{k=1}^K \text{softmax}(\mathbf{u}^{(k)}) \quad (15)$$

The negative log-likelihood for the dataset can be written as:

$$\mathcal{L}_{\text{clf}}(\mathbf{W}) = \frac{1}{N} \sum_{i=1}^N \log \sum_{k=1}^K \exp \left(u_{i,y_i}^{(k)} - \log \sum_c \exp u_{i,c}^{(k)} \right) - \log K \quad (16)$$

where $u_{i,c}$ is the c -th element in \mathbf{u}_i .

After the model is optimized, we use $\sigma(\mathbf{x}^*)^2$ to estimate the data uncertainty given input \mathbf{x}^* in the regression case:

$$U_d(y^*|\mathbf{x}^*) = \sigma(\mathbf{x}^*)^2 \quad (17)$$

For classification, we use the average variance of the logits as a surrogate to quantify the data uncertainty. This does not directly measures data uncertainty in the output space but can reflect to a certain extent the variance caused by the input.

Combining Both Uncertainties

To simultaneously quantify both uncertainties, we can simply use Equation 10,16 in the training stage and adopt MC dropout during evaluation as described in the model uncertainty section.

Take the regression setting as an example, prediction can be approximated as:

$$\mathbb{E}(y^*|\mathbf{x}^*) \approx \frac{1}{M} \sum_{j=1}^M \mu^{\widehat{\mathbf{W}}_j}(\mathbf{x}^*) \quad (18)$$

Model uncertainty can be measured with:

$$U_m(y^*|\mathbf{x}^*) = \frac{1}{M} \sum_{j=1}^M \mu^{\widehat{\mathbf{W}}_j}(\mathbf{x}^*)^2 - \mathbb{E}(y^*|\mathbf{x}^*)^2 \quad (19)$$

and data uncertainty is quantified with:

$$U_d(y^*|\mathbf{x}^*) = \frac{1}{M} \sum_{j=1}^M \sigma^{\widehat{\mathbf{W}}_j}(\mathbf{x}^*)^2 \quad (20)$$

where again $\widehat{\mathbf{W}}_j$ is sampled from $q(\mathbf{W})$. Figure 1 is an illustration of the evaluation process of predictive value and different uncertainty measures.

Experiments and Results

We conduct experiments on three different NLP tasks: sentiment analysis, named entity recognition, and language modeling. In the following sections, we will introduce the datasets, experiment setups, evaluation metrics for each task, and experimental results.

Sentiment Analysis

Conventionally, sentiment analysis is done with classification. In this study, to explore the effect of quantifying uncertainties, we consider both regression and classification settings for sentiment analysis. In the regression setting, we treat the class labels as numerical values and aim to predict the real value score given a review document. We introduce the datasets and setups in both settings in this section.

Datasets We use four large scale datasets containing document reviews as in (Tang, Qin, and Liu 2015). Specifically, we use IMDB movie review data (Diao et al. 2014) and Yelp restaurant review datasets from Yelp Dataset Challenge in 2013, 2014 and 2015. Summaries of the four datasets are given in Table 1. Data splits are the same as in (Tang, Qin, and Liu 2015; Diao et al. 2014).

Experiment Setup We implement convolutional neural network (CNN) baselines in both regression and classification settings. CNN model structure follows (Kim 2014). We use a maximum vocabulary size of 20,000; embedding size is set to 300; three different kernel sizes are used in all models and they are chosen from [(1,2,3), (2,3,4), (3,4,5)]; number of feature maps for each kernel is 100; dropout (Srivastava et al. 2014) is applied between layers and dropout rate is 0.5. To evaluate model uncertainty and input uncertainty, 10 samples are drawn from the approximated posterior to estimate the output mean and variance.

Adam (Kingma and Ba 2014) is adopted in all experiments with learning rate chosen from [3e-4, 1e-3, 3e-3] and weight decay from [3e-5, 1e-4, 3e-4]. Batch size is set to 32 and training runs for 48 epochs with 2,000 iterations per

Model	Yelp 2013	Yelp 2014	Yelp 2015	IMDB
(RGS MSE)				
Baseline	0.71	0.72	0.72	3.62
Baseline + MU	0.57	0.55	0.55	3.20
Baseline + DU	0.84	0.75	0.73	3.74
Baseline + both	0.57	0.54	0.53	3.13
Relative Improvement (%)	19.7	25.0	26.4	13.5

Table 2: Test set mean squared error of CNN regressors trained on four sentiment analysis datasets. RGS MSE represents regression MSE. Baseline is the baseline CNN model (Kim 2014); MU and DU denote model uncertainty and data uncertainty respectively. Classification results have a similar pattern but the improvements are less obvious.

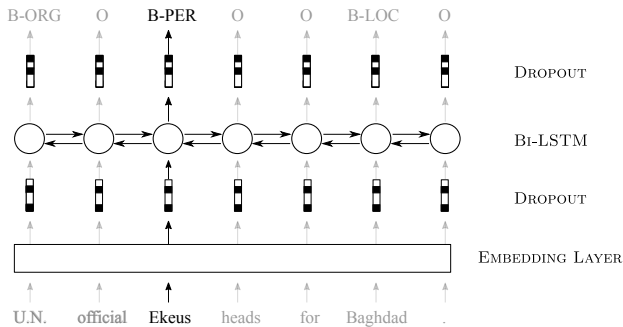


Figure 2: An illustration of the bidirectional LSTM model used for named entity recognition. Two dropout layers independently sample their masks while masks are the same across time steps.

epoch for Yelp 2013 and IMDB, and 5,000 iterations per epoch for Yelp 2014 and 2015. Model with best performance on the validation set is chosen to be evaluated on the test set.

Evaluation We use accuracy in the classification setting and mean squared error (MSE) in the regression setting to evaluate model performances. Accuracy is a standard metric to measure classification performance. MSE measures the average deviation of the predicted scores from the true ratings and is defined as:

$$\text{MSE} = \frac{\sum_{i=1}^N (\text{gold}_i - \text{predicted}_i)^2}{N} \quad (21)$$

Results Experiment results are shown in Table 2. We can see that BNN models (i.e. model w/ MU and w/ both) outperform non-Bayesian models. Quantifying both model and data uncertainties boosts performances by 13.5%-26.4% in the regression setting. Most of the performance gain is from quantifying model uncertainty. Modeling input-dependent uncertainty alone marginally hurts prediction performances. The performances for classification increase marginally with added uncertainty measures. We conjecture that this might be due to the limited output space in the classification setting.

Model	CoNLL 2003
(F1 SCORE)	
Baseline	77.5
Baseline + MU	76.5
Baseline + DU	79.6
Baseline + both	78.5
Relative Improvement (%)	2.7

Table 3: Test set F1 scores (%) of bidirectional LSTM taggers trained on CoNLL 2003 dataset. Baseline is the baseline bidirectional LSTM model; MU and DU denote model uncertainty and data uncertainty respectively. Modeling data uncertainty boosts performances

Named Entity Recognition

We conduct experiments on named entity recognition (NER) task which essentially is a sequence tagging problem. We adopt a bidirectional long-short term memory (LSTM) (Hochreiter and Schmidhuber 1997) neural network as the baseline model and measure the effects of quantifying model and input-dependent uncertainties on the test performances.

Datasets For the NER experiments, we use the CoNLL 2003 dataset (Tjong Kim Sang and De Meulder 2003). This corpus consists of news articles from the Reuters RCV1 corpus annotated with four types of named entities: location, organization, person, and miscellaneous. The annotation scheme is IOB (which stands for inside, outside, begin, indicating the position of the token in an entity). The original dataset includes annotations for part of speech (POS) tags and chunking results, we do not include these features in the training and use only the text information to train the NER model.

Experiment Setup Our baseline model is a bidirectional LSTM with dropout applied after the embedding layer and before the output layer. We apply dropout with the same mask for all time steps following (Gal and Ghahramani 2016b). An illustration of the model is shown in Figure 2. Note that the dropout mask is the same across time steps. Different examples in the same mini-batch have different dropout masks.

Word embedding size is 200 and hidden size in each direction is 200; dropout probability is fixed at 0.5; other hyper-

Model	PTB
(PPL)	
Baseline	82.7
Baseline + MU	81.3
Baseline + DU	80.5
Baseline + both	79.2
Relative Improvement (%)	4.2

Table 4: Test set perplexities of LSTM language models trained on PTB dataset. PPL represents perplexity. Baseline is the baseline medium two-layer LSTM model in (Zaremba, Sutskever, and Vinyals 2014); MU and DU denote model uncertainty and data uncertainty respectively.

parameters related to quantifying uncertainties are the same with previous experiment setups.

For training, we use Adam optimizer (Kingma and Ba 2014). Learn rate is selected from [3e-4, 1e-3, 3e-4] and weight decay is chosen from [0, 1e-5, 1e-4]. Training runs for 100 epochs with each epoch consisting of 2,000 randomly sampled mini-batches. Batch size is 32.

Evaluation The performances of the taggers are measured with F1 score:

$$F1 = \frac{2 \cdot \text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}} \quad (22)$$

where precision is the percentage of entities tagged by the model that are correct; recall is the percentage of entities in the gold annotation that are tagged by the model. A named entity is correct only if it is an exact match of the corresponding entity in the data.

Results Test set performances of the models trained with and without uncertainties are listed in Table 3. We observe that much different from the sentiment analysis case, models that quantify data uncertainty improves performances by 2.7% in F1 score. Quantifying model uncertainty, on the other hand, under-performs by approximately 1% absolute F1 score. One possible explanation for worse results with model uncertainty is due to the use of MC dropout and chunk based evaluation. More specifically, predicted tag at each time step is taken to be the argmax of the average tag probability across multiple passes with the same inputs. This operation might break some temporal dynamics captured with a single pass of the inputs.

Language Modeling

We introduce the experiments conducted on the language modeling task.

Datasets We use the standard Penn Treebank (PTB), a standard benchmark in the field. The dataset contains 887,521 tokens (words) in total.

High DU
should game automatic doors !
i've bought tires from discount tire for years at different locations and have had a <i>good</i> experience , but this location was different . i went in to get some new tires with my fiancé . john the sales guy pushed a certain brand , specifically because they were running a rebate special . tires are tires , especially on a prius (the rest 134 tokens not shown here due to space)
Low DU
<i>great</i> sports bar ! brian always goes out of his way to make sure we are <i>good</i> to go ! <i>great</i> people , <i>great</i> food , <i>great</i> music ! <i>great</i> bartenders and even <i>great</i> bouncers ! always accommodating ! all the <i>best</i> _unk ! <i>great</i> _unk burger ! <i>amazing</i> service ! <i>brilliant</i> interior ! the burger was <i>delicious</i> but it was a little big . it 's a <i>great</i> restaurant <i>good</i> for any occasion .

Table 5: Examples of inputs in Yelp 2013 dataset with high and low data uncertainties. They are taken from the top and bottom 10 examples with respect to measured data uncertainty. High DU is around 0.80 and low is around 0.52. Italic tokens are highly indicative tokens for higher ratings.

Experiment Setting We follow the medium model setting in (Zaremba, Sutskever, and Vinyals 2014). The model is a two-layer LSTM with hidden size 650. Dropout rate is fixed at 0.5. Dropout is applied after the embedding layer, before the output layer, and between two LSTM layers. Similar to the NER setting, dropout mask is the same across time steps. Unlike (Gal and Ghahramani 2016b), we do not apply dropout between time steps. Weight tying is also not applied in our experiments. Number of samples for MC dropout is set to 50.

Evaluation We use the standard perplexity to evaluate the trained language models.

Results The results are shown in Table 4. We can observe performance improvements when quantifying either model uncertainty or data uncertainty. We observe less performance improvements compared to (Gal and Ghahramani 2016b) possibly due to the fact that we use simpler dropout formulation that only applies dropout between layers.

Summary of Results

We can observe from the results that accounting for uncertainties improves model performances in all three NLP tasks. In detail, for the sentiment analysis setting with CNN models, quantifying both uncertainties gives the best performance and improves upon baseline by up to 26.4%. For named entity recognition, input-dependent data uncertainty improves F1 scores by 2.7% in CoNLL 2003. For language modeling, perplexity improves 4.2% when both uncertainties are quantified.

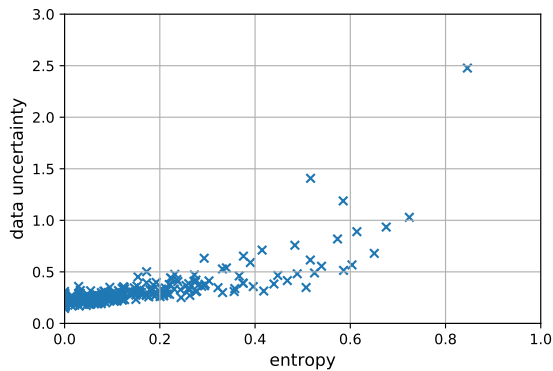


Figure 3: Scatter plot of evaluated data uncertainty against entropy of annotated NER tag distribution for all tokens in CoNLL 2003 dataset. Higher input-dependent data uncertainties are estimated for input tokens that have higher tag entropies.

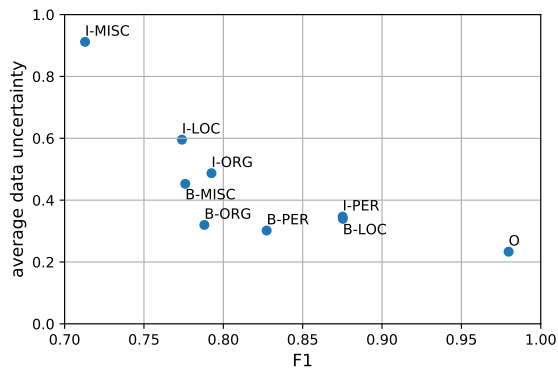


Figure 4: Scatter plot of average evaluated data uncertainty against test set F1 score for different tags. Higher data uncertainties are observed when predicting tags with lower F1 score.

Analysis

In the previous section, we empirically show that by modeling uncertainties we could get better performances for various NLP tasks. In this section, we turn to analyze the uncertainties quantified by our approach. We mainly focus on the analysis of data uncertainty. For model uncertainty, we have similar observations to (Kendall and Gal 2017).

What Does Data Uncertainty Measure

In Equation 3, we define data uncertainty as the proportion of observation noise or variance that is caused by the inputs. Conceptually, input-dependent data uncertainty is high if it is hard to predict its corresponding output given an input. We explore in both sentiment analysis and named entity recognition tasks and analyze the characteristics of inputs with high and low data uncertainties measured by our model.

Table 5 shows examples with high and low data uncertainties taken from the Yelp 2013 test set. Due to space limit, we only show four typical examples. Examples with high data uncertainties are either short or very long with extensive descriptions of actions instead of opinions. On the other hand, examples with low data uncertainties are of relatively medium length and contain large amount of strong opinion tokens. These observations are consistent with our intuition.

For the CoNLL 2003 dataset, we take all tokens and measure their average quantified data uncertainty. We use the following strategy to measure how difficult the prediction for each token is: 1. calculate the distribution of NER tags the token is annotated in the training data; 2. use entropy to measure the difficulty level of the prediction defined as:

$$H(p_1, p_2, \dots, p_m) = - \sum_{i=1}^m p_i \log p_i \quad (23)$$

where p_1, p_2, \dots, p_m is the distribution of NER tags assigned to a particular token in the training set. The higher the entropy, the more tags a token can be assigned and the more even these possibilities are. For example, in the training data, the token *Hong* has been annotated with tag B-LOC (first token in *Hong Kong*), B-ORG, B-PER, B-MISC. Therefore *Hong* has a high entropy with respect to its tag distribution. In contrast, the token *defended* has only been assigned tag O representing outside of any named entities. Therefore *defended* has a low entropy of 0.

We plot the relationship between the average quantified data uncertainty and NER tag distribution for the tokens in Figure 3. It is clear that for tokens with higher entropy values, data uncertainties measured by our model are indeed higher.

We also analyze the data uncertainty differences among NER tags. For each NER tag, we evaluate its test set F1 score and average data uncertainty quantified by our model. The relationship is shown in Figure 4. We observe that when predicting more difficult tags, higher average data uncertainties are estimated by the model. These observations indicate that data uncertainty quantified by our model is highly correlated with prediction confidence.

Conclusion

In this work, we evaluate the benefits of quantifying uncertainties in modern neural network models applied in the context of three different natural language processing tasks. We conduct experiments on sentiment analysis, named entity recognition, and language modeling tasks with convolutional and recurrent neural network models. We show that by quantifying both uncertainties, model performances are improved across the three tasks. We further investigate the characteristics of inputs with high and low data uncertainty measures in Yelp 2013 and CoNLL 2003 datasets. For both datasets, our model estimates higher data uncertainties for more difficult predictions. Future research directions include possible ways to fully utilize the estimated uncertainties.

References

- Blundell, C.; Cornebise, J.; Kavukcuoglu, K.; and Wierstra, D. 2015. Weight uncertainty in neural networks. *arXiv preprint arXiv:1505.05424*.
- Buntine, W. L., and Weigend, A. S. 1991. Bayesian backpropagation. *Complex systems* 5(6):603–643.
- Denker, J. S., and Lecun, Y. 1991. Transforming neural-net output levels to probability distributions. In *Advances in neural information processing systems*, 853–859.
- Der Kiureghian, A., and Ditlevsen, O. 2009. Aleatory or epistemic? does it matter? *Structural Safety* 31(2):105–112.
- Diao, Q.; Qiu, M.; Wu, C.-Y.; Smola, A. J.; Jiang, J.; and Wang, C. 2014. Jointly modeling aspects, ratings and sentiments for movie recommendation (jmars). In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, 193–202. ACM.
- Gal, Y., and Ghahramani, Z. 2016a. Dropout as a bayesian approximation: Representing model uncertainty in deep learning. In *international conference on machine learning*, 1050–1059.
- Gal, Y., and Ghahramani, Z. 2016b. A theoretically grounded application of dropout in recurrent neural networks. In *Advances in neural information processing systems*, 1019–1027.
- Graves, A. 2011. Practical variational inference for neural networks. In *Advances in neural information processing systems*, 2348–2356.
- Hernández-Lobato, J. M., and Adams, R. 2015. Probabilistic backpropagation for scalable learning of bayesian neural networks. In *International Conference on Machine Learning*, 1861–1869.
- Hochreiter, S., and Schmidhuber, J. 1997. Long short-term memory. *Neural computation* 9(8):1735–1780.
- Kendall, A., and Gal, Y. 2017. What uncertainties do we need in bayesian deep learning for computer vision? In *Advances in neural information processing systems*, 5574–5584.
- Kendall, A.; Badrinarayanan, V.; and Cipolla, R. 2015. Bayesian segnet: Model uncertainty in deep convolutional encoder-decoder architectures for scene understanding. *arXiv preprint arXiv:1511.02680*.
- Kim, Y. 2014. Convolutional neural networks for sentence classification. *arXiv preprint arXiv:1408.5882*.
- Kingma, D. P., and Ba, J. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.
- Le, Q. V.; Smola, A. J.; and Canu, S. 2005. Heteroscedastic gaussian process regression. In *Proceedings of the 22nd international conference on Machine learning*, 489–496. ACM.
- MacKay, D. J. 1992. A practical bayesian framework for backpropagation networks. *Neural computation* 4(3):448–472.
- MacKay, D. J. 1995. Probable networks and plausible predictions—a review of practical bayesian methods for supervised neural networks. *Network: Computation in Neural Systems* 6(3):469–505.
- Neal, R. M. 2012. *Bayesian learning for neural networks*, volume 118. Springer Science & Business Media.
- Nix, D. A., and Weigend, A. S. 1994. Estimating the mean and variance of the target probability distribution. In *Neural Networks, 1994. IEEE World Congress on Computational Intelligence., 1994 IEEE International Conference On*, volume 1, 55–60. IEEE.
- Srivastava, N.; Hinton, G.; Krizhevsky, A.; Sutskever, I.; and Salakhutdinov, R. 2014. Dropout: a simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research* 15(1):1929–1958.
- Tang, D.; Qin, B.; and Liu, T. 2015. Document modeling with gated recurrent neural network for sentiment classification. In *Proceedings of the 2015 conference on empirical methods in natural language processing*, 1422–1432.
- Tjong Kim Sang, E. F., and De Meulder, F. 2003. Introduction to the conll-2003 shared task: Language-independent named entity recognition. In *Proceedings of the seventh conference on Natural language learning at HLT-NAACL 2003-Volume 4*, 142–147. Association for Computational Linguistics.
- Zaremba, W.; Sutskever, I.; and Vinyals, O. 2014. Recurrent neural network regularization. *arXiv preprint arXiv:1409.2329*.
- Zhu, L., and Laptev, N. 2017. Deep and confident prediction for time series at uber. In *Data Mining Workshops (ICDMW), 2017 IEEE International Conference on*, 103–110. IEEE.