UC Santa Barbara
UC Santa Barbara Electronic Theses and Dissertations

Title
On Uncertainty and Robustness in Deep Learning for Natural Language Processing

Permalink
https://escholarship.org/uc/item/6td9p2d2

Author
Xiao, Yijun

Publication Date
2022

Supplemental Material
https://escholarship.org/uc/item/6td9p2d2#supplemental

Peer reviewed|Thesis/dissertation

eScholarship.org Powered by the California Digital Library
University of California

University of California
Santa Barbara

On Uncertainty and Robustness in Deep Learning for Natural Language Processing

A dissertation submitted in partial satisfaction
of the requirements for the degree

Doctor of Philosophy
in
Computer Science

by

Yijun Xiao

Committee in charge:

Professor William Wang, Chair
Professor Xifeng Yan
Professor Sang-Yun Oh

June 2022

The Dissertation of Yijun Xiao is approved.

Professor Xifeng Yan

Professor Sang-Yun Oh

Professor William Wang, Committee Chair

June 2022

On Uncertainty and Robustness in Deep Learning for Natural Language Processing

Copyright © 2022

by

Yijun Xiao

To my parents,
who supported me unconditionally throughout this journey.

Acknowledgements

I am incredibly grateful to my advisor William Yang Wang for the support and guidance throughout my study and during difficult times. He helps me enormously in navigating the field of natural language processing and provides me with knowledgeable suggestions and great freedom to explore. I would like to thank Professor Xifeng Yan and Professor Sang-Yun Oh for serving on my committee and for their insightful comments and feedback on my research.

I owe great thanks to all my collaborators, colleagues, and friends (in alphabetical order): Alon Albalak, Rob Chambers, Wenhu Chen, Kyunghyun Cho, Kevin Knight, Peter Li, Yasumasa Onoe, Luo Ping, Jing Qian, Jiyuan Qian, Xing Shi, Yi-Lin Tuan, Hong Wang, Hongmin Wang, Xin Wang, Wenhan Xiong, Tiancheng Zhao, Ganbin Zhou for their company, support and discussions throughout the years.

I am truely thankful to my partner Zhi for her company and encouragement through my Ph.D. journey. Last and foremost, my deepest love to my parents.

Curriculum Vitæ

Yijun Xiao

Education

2017 - 2022	Ph.D. in Computer Science, University of California, Santa Barbara.
2014 - 2016	M.S. in Data Science, New York University.
2011 - 2013	M.S. in Civil Engineering, University of California, Davis.
2007 - 2011	B.E. in Civil Engineering, Tsinghua University.

Publications

EACL 2021	Yijun Xiao, William Yang Wang. On Hallucination and Predictive Uncertainty in Conditional Language Generation.
arXiv:1912.12818	Yijun Xiao, William Yang Wang. Disentangled Representation Learning with Wasserstein Total Correlation.
AAAI 2019	Yijun Xiao, William Yang Wang. Quantifying Uncertainties in Natural Language Processing Tasks.
arXiv:1811.00135	Yijun Xiao, Tiancheng Zhao, William Yang Wang. Dirichlet Variational Autoencoder for Text Modeling.
AAAI 2018	Ganbin Zhou, Ping Luo, Yijun Xiao, Fen Lin, Bo Chen, Qing He. Elastic Responding Machine for Dialog Generation with Dynamically Mechanism Selecting.
AAAI 2018	Ganbin Zhou, Ping Luo, Rongyu Cao, Yijun Xiao, Fen Lin, Bo Chen, Qing He. Tree-Structured Neural Machine for Linguistics-Aware Sentence Generation.
arXiv:1602.00367	Yijun Xiao, Kyunghyun Cho. Efficient Character-level Document Classification by Combining Convolution and Recurrent Layers.

Experience

2021/06 - 2021/09	Software Engineer Intern, Machine Learning, Facebook
2019/06 - 2019/09	Research Intern, DiDi Labs
2016/06 - 2017/06	NLP Engineer, WeChat

Abstract

On Uncertainty and Robustness in Deep Learning for Natural Language Processing

by

Yijun Xiao

With the recent success of deep learning methods, neural-based models have achieved superior performances and since dominated across natural language understanding and generation tasks. Due to the fact that many of such models are black-box mappings from the input to the output, it is increasingly important to understand how confident a model is about certain predictions and how robust the model is under distribution shift. Uncertainty estimation methods provide us a way to separately quantify epistemic and aleatoric uncertainty where the former arises due to inadequate knowledge about the model and the latter is the inherent irreducible uncertainty in data. We could then develop uncertainty-aware approaches that improve the robustness of a model. A closely related concept called calibration measures how aligned model confidence is with the prediction accuracy. A better calibrated model is more robust because we could better interpret the predicted confidence scores from the model. Another important aspect concerning the robustness of a deep learning model is its ability to adapt to distribution shift. When the test distribution differs significantly from the training distribution, the ability to detect and adjust accordingly is vital in practical applications.

In this dissertation, we first examine the benefits of applying uncertainty quantification methods to sentiment analysis, named entity recognition and language modeling tasks. We show that by incorporating uncertainty estimation in the modeling process, we observe significant improvements in the three important NLP tasks. We then draw connections between hallucination and predictive uncertainty and empirically investigate their relationship in image captioning and data-to-text generation tasks. Next, we investigate the relationship between model calibration and label smoothing in document classification. We further acknowledge the importance of learning under distribution shift by introducing a benchmark that evaluates models on their abilities to estimate the change of label distributions in classification settings. Finally, we summarize the findings and discuss potential future research directions for uncertainty aware learning and model robustness for NLP.

The methods and analyses in this dissertation allow for a better understanding of uncertainty and robustness in deep learning for natural language processing tasks. We envision a future where AI systems are explainable and accountable.

Contents

Curriculum Vitae **vi**

Abstract **vii**

1 Introduction **1**
- 1.1 Uncertainty Estimation 2
- 1.2 Model Calibration 4
- 1.3 Distribution Shift and Quantification Learning 4

2 Uncertainty Quantification in Natural Language Processing **6**
- 2.1 Introduction 6
- 2.2 Related Work 8
- 2.3 Methods 10
- 2.4 Experiments and Results 16
- 2.5 Analysis 22
- 2.6 Conclusion 25

3 Hallucination and Uncertainty in Conditional Language Generation **27**
- 3.1 Introduction 27
- 3.2 Hallucination and Predictive Uncertainty 29
- 3.3 Case Study: Image Captioning 32
- 3.4 Case Study: Data-to-text Generation 36
- 3.5 Reducing Hallucination 39
- 3.6 Related Work 44
- 3.7 Discussion and Conclusions 46

4 Label Smoothing and Model Calibration in Text Classification **47**
- 4.1 Introduction 47
- 4.2 Preliminaries 48
- 4.3 Does Label Smoothing Always Help with Calibration? 50
- 4.4 Why Does Label Smoothing (Not) Help? 51
- 4.5 When Does Label Smoothing Help? 53

4.6 Label Sharpening 54
4.7 Discussion and Conclusions 55

5 Label Distribution Estimation Under Real-World Temporal Shift 56
5.1 Introduction 56
5.2 Related Work 59
5.3 The Text Quantification Benchmark 61
5.4 Baseline Algorithms 68
5.5 Experiments and Results 69
5.6 Conclusions 74

6 Conclusion 75

Bibliography 77

Chapter 1

Introduction

The last decade witnessed an enormous success of deep neural networks. They have been adapted in various research fields where complex systems need to be modeled. Natural language processing (NLP) is no exception. Especially with the invention of the Transformer model [1], significant improvements have been further achieved across many natural language understanding and generation tasks. Despite superior performances compared to traditional machine learning models, deep neural models remain limited in many critical real-world applications. Some famous failures[1] of image captioning and machine translation systems have caused controversy and raised discussions on the robustness of deep learning models. Some major factors limiting the deep learning model robustness are:

- How a deep learning model produces outputs are often less transparent, making it less explainable compared to some of the traditional machine learning models such as linear models and decision trees [2].

- Many deep models produce point estimates and are unable to provide reliable uncer-

---

[1] https://www.bbc.com/news/technology-33347866

tainty estimates for its decisions. There are frequent observations of over-confident predictions [3].

- Deep models are often developed under the i.i.d. assumption where test data are sampled from the same distribution as the training data. They are sensitive to distribution shifts [4].

At the core of tackling these limitations are reliable uncertainty estimation, model calibration, and distribution shift detection / adaptation. In this dissertation, we conduct analyses on all three fronts in the context of different natural language processing tasks.

1.1 Uncertainty Estimation

The motivation behind uncertainty quantification is straightforward: a robust model should know when it does not know. An accurate measure of how uncertain a model is given some specific inputs is beneficial in many ways. For example, predictions that are deemed uncertain can be ignored or passed to human experts for verification to prevent harm in a production AI system; Uncertainty can be used as a signal to acquire new samples in active learning [5]; uncertainty measures can be used to balance between exploitation and exploration in deep reinforcement learning [6, 7].

Recent studies in uncertainty quantification often categorize uncertainty into two parts: epistemic uncertainty and aleatoric uncertainty [8, 9]. Epistemic uncertainty, also referred to as model uncertainty, captures uncertainty about the model in the functional space. Model uncertainty is reducible as more data becomes available and the training distribution approaches the ground-truth data distribution. On the other hand, aleatoric uncertainty, also called data uncertainty, is data inherent and cannot be reduced with more training data.

Popular methods to measure both parts of the uncertainty during model prediction include Bayesian neural networks (BNNs) [10, 11], Monte Carlo Dropout (MC Dropout) [12], Deep ensembles [13], Posterior networks [14], etc. BNN weights are modeled as Gaussian random variables instead of scalars. The variances of the weights directly describe the epistemic uncertainty of the model; MC Dropout applies dropout between network layers and measures the uncertainty in the output space by comparing results from multiple forward passes; Instead of injecting randomness through dropout layers, Deep ensembles train multiple copies of a network using different hyper-parameters or random seeds and measure the discrepancy in the output space; Posterior network aims to use a single network to quantify uncertainty. In the context of classification, Posterior network produces a Dirichlet distribution as its prediction instead of a categorical distribution and uses the dispersion of the prediction to represent the estimated model uncertainty.

In terms of applications, probabilistic pixel-wise semantic segmentation has been studied in [15]; [16] studied model uncertainty in recurrent neural networks in the context of language modeling and sentiment analysis; [9] researched both model and data uncertainties in various vision tasks and achieved higher performances; [17] used similar approaches to perform time series prediction and anomaly detection with Uber trip data. In Chapter 2, 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. In Chapter 3, we investigate the relationship between hallucination and predictive uncertainty in image captioning and data-to-text generation tasks. We propose an uncertainty-aware beam search algorithm to reduce the chance of hallucination by penalizing parts or the entirety of the predictive uncertainty during model decoding.

1.2 Model Calibration

In a supervised classification task, model calibration measures how closely the model prediction confidence reflects the actual classification accuracy. For example, when a sentiment analysis model predicts 60% probability that a tweet is positive, ideally there are actually 60% such tweets expressing positive sentiment. The benefits of a calibrated model are mainly two-folds: it is less risky and easier to control in safety-critical applications since low model confidence can be interpreted as low accuracy; it is important for general model interpretability and trustworthiness especially for black-box models like deep neural networks.

[3] show that post calibration methods such as Platt scaling [18] are effective in recalibrating modern neural networks. [19] conduct a set of experiments discussing the effects of label smoothing on representation learning and knowledge distillation. They find that label smoothing has an implicit model calibration effect. Models trained with label smoothing are better calibrated on two image classification tasks and the EN-DE translation task. In Chapter 4, we investigate the role label smoothing plays with respect to model calibration in text classification settings and find that label smoothing achieves calibration effects in some scenarios by artificially suppressing model uncertainties at the cost of biasing the predictions.

1.3 Distribution Shift and Quantification Learning

There has been an increasing interest in studying the challenges arising from data distribution changes in the machine learning community [20, 21, 22, 23]. The focus of these studies is mainly on adapting to covariate shift and improving the performance of the underlying learner in a shifted domain. Another line of research [24, 25] aims to estimate and adjust for the label prevalence changes in the target domain, assuming the feature distribution of each class stays the same.

Quantification learning deals with the task of estimating the label distribution when test distributions differ from the training. Different from classification tasks, quantification learning focuses on accuracy on the aggregate level. For example, while classifiers can be adopted to identify and demote individual contents categorized as hate speech on a social platform, quantifiers are more useful in estimating the prevalence of hate speech for a collection of contents. With the help of a robust quantifier, such a platform could evaluate the effectiveness of a newly developed anti-hate speech feature using A/B testing and closely monitor future hate speech prevalence on the platform.

Despite the many applications, quantification is relatively understudied in the NLP community. One main problem with recent studies on text quantification is the dataset, especially the testing sets. In Chapter 5, we introduce the first text quantification benchmark with naturally occurred temporal distribution shift. We focus on temporal distribution shift to tackle a main use case for text quantifiers: monitoring future label prevalence changes given historic labeled data. A total of eight datasets are included in the benchmark spanning sentiment analysis, document categorization, and toxicity classification. We evaluate different quantification algorithms on the benchmark and find that no algorithm consistently outperforms others.

Chapter 2

Uncertainty Quantification in Natural Language Processing

2.1 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) [26, 27, 28, 29, 30] 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 natural of BNNs. It is also showed in [10] 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 chapter and is also called aleatoric uncertainty [8]. 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 chapter, 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 [15, 16, 9, 17]. In this chapter, 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 chapter 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.

2.2 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 [26, 27, 28, 29, 30] 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 [31, 11, 10, 12]. In particular, Monte Carlo dropout (MC dropout) [12] 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 [32, 33] are more suitable.

Recently, quantifications of model and data uncertainties are gaining researchers' attentions. Probabilistic pixel-wise semantic segmentation has been studied in [15]; Gal and Ghahramani [16] studied model uncertainty in recurrent neural networks in the context of language modeling and sentiment analysis; Kendall and Gal [9] researched both model and data uncertainties in various vision tasks and achieved higher performances; Zhu and Laptev [17] 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.

2.3 Methods

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

$$\mathrm{Var}(y) = \mathrm{Var}\left(\mathbb{E}[y|x]\right) + \mathbb{E}\left[\mathrm{Var}(y|x)\right] \tag{2.1}$$

We mathematically define model uncertainty and data uncertainty as:

$$U_m(y|x) = \mathrm{Var}\left(\mathbb{E}[y|x]\right) \tag{2.2}$$

$$U_d(y|x) = \mathbb{E}\left[\mathrm{Var}(y|x)\right] \tag{2.3}$$

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 $\mathrm{Var}(y|x)$. 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:

$$y|\mathbf{W} \sim \mathcal{N}(f^{\mathbf{W}}(\mathbf{x}), \sigma^2) \tag{2.4}$$

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

$$p(y^*|\mathbf{x}^*, D) = \int_{\mathbf{W}} p\left(y^*|f^{\mathbf{W}}(\mathbf{x}^*)\right) p(\mathbf{W}|D) d\mathbf{W} \tag{2.5}$$

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 [11, 10, 12]. In particular, dropout variational inference method [12], 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, 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 2.5 and perform

Monte Carlo integration as follows:

$$\mathbb{E}(y^*|\mathbf{x}^*) \approx \frac{1}{M}\sum_{j=1}^{M} f^{\widehat{\mathbf{W}}_j}(\mathbf{x}^*) \tag{2.6}$$

Predictive variance can also be approximated as:

$$\mathrm{Var}\,(y^*) \approx \frac{1}{M}\sum_{j=1}^{M} f^{\widehat{\mathbf{W}}_j}(\mathbf{x}^*)^2 - \mathbb{E}(y^*|\mathbf{x}^*)^2 + \sigma^2 \tag{2.7}$$

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.:

$$U_m(y^*|\mathbf{x}^*) = \frac{1}{M}\sum_{j=1}^{M} f^{\widehat{\mathbf{W}}_j}(\mathbf{x}^*)^2 - \mathbb{E}(y^*|\mathbf{x}^*)^2 \tag{2.8}$$

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 $\mathrm{Var}(y|x)$ mentioned in Equation 2.3.

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:

$$y \sim \mathcal{N}\left(\mu(\mathbf{x}), \sigma(\mathbf{x})^2\right) \tag{2.9}$$

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

$$\begin{aligned}\mathcal{L}_{\text{rgs}}(\mathbf{W}) =& -\frac{1}{N}\sum_{i=1}^{N} \log p(y_i|\mu(\mathbf{x}_i), \sigma(\mathbf{x}_i)) \\ =& \frac{1}{N}\sum_{i=1}^{N}\left(\frac{1}{2}\left|\frac{y_i - \mu(\mathbf{x}_i)}{\sigma(\mathbf{x}_i)}\right|^2 + \right. \\ & \left. \frac{1}{2}\log\sigma(\mathbf{x}_i)^2 + \frac{1}{2}\log 2\pi\right)\end{aligned} \tag{2.10}$$

Comparing Equation 2.10 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 2.10 is referred to as learned loss attenuation in [9].

While Equation 2.10 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:

$$\mathbf{u} \sim \mathcal{N}\left(\boldsymbol{\mu}(\mathbf{x}), \mathrm{diag}(\boldsymbol{\sigma}(\mathbf{x})^2)\right) \tag{2.11}$$

$$\mathbf{p} = \mathrm{softmax}(\mathbf{u}) \tag{2.12}$$

$$y \sim \mathrm{Categorical}(\mathbf{p}) \tag{2.13}$$

where diag() function takes a vector and output a diagonal matrix by putting the elements on the main diagonal. Note here in Equation 2.13, 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:

$$\mathbf{u}^{(k)} \sim \mathcal{N}\left(\boldsymbol{\mu}(\mathbf{x}), \mathrm{diag}(\boldsymbol{\sigma}(\mathbf{x})^2)\right) \tag{2.14}$$

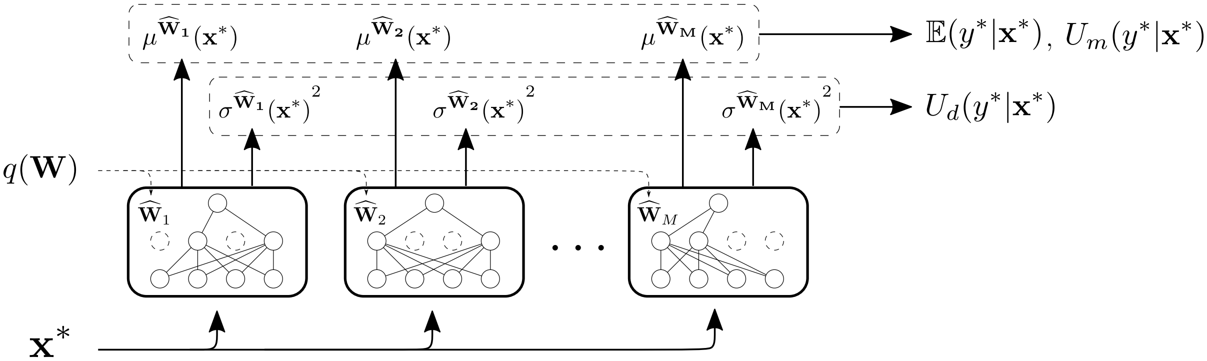
$$\mathbb{E}[\mathbf{p}] \approx \frac{1}{K}\sum_{k=1}^{K} \mathrm{softmax}(\mathbf{u}^{(k)}) \tag{2.15}$$

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

$$\begin{aligned}\mathcal{L}_{\mathrm{clf}}(\mathbf{W}) =& \frac{1}{N}\sum_{i=1}^{N} \log \sum_{k=1}^{K} \exp\left(u_{i,y_i}^{(k)} - \log \sum_{c} \exp u_{i,c}^{(k)}\right) \\ & - \log K \end{aligned} \tag{2.16}$$

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



$\mu^{\widehat{\mathbf{W}}_1}(\mathbf{x}^*)$
$\mu^{\widehat{\mathbf{W}}_2}(\mathbf{x}^*)$
$\mu^{\widehat{\mathbf{W}}_M}(\mathbf{x}^*)$
$\mathbb{E}(y^*|\mathbf{x}^*),\ U_m(y^*|\mathbf{x}^*)$
$\sigma^{\widehat{\mathbf{W}}_1}(\mathbf{x}^*)^2$
$\sigma^{\widehat{\mathbf{W}}_2}(\mathbf{x}^*)^2$
$\sigma^{\widehat{\mathbf{W}}_M}(\mathbf{x}^*)^2$
$U_d(y^*|\mathbf{x}^*)$
$q(\mathbf{W})$
$\widehat{\mathbf{W}}_1$
$\widehat{\mathbf{W}}_2$
$\widehat{\mathbf{W}}_M$
$\mathbf{x}^*$

Figure 2.1: 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.

input $\mathbf{x}^*$ in the regression case:

$$U_d(y^*|\mathbf{x}^*) = \sigma(\mathbf{x}^*)^2 \tag{2.17}$$

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 2.10,2.16 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:

$$\mathbb{E}(y^*|\mathbf{x}^*) \approx \frac{1}{M}\sum_{j=1}^{M} \mu^{\widehat{\mathbf{W}}_j}(\mathbf{x}^*) \tag{2.18}$$

Model uncertainty can be measured with:

$$U_m(y^*|\mathbf{x}^*) = \frac{1}{M}\sum_{j=1}^{M} \mu^{\widehat{\mathbf{W}}_j}(\mathbf{x}^*)^2 - \mathbb{E}(y^*|\mathbf{x}^*)^2 \tag{2.19}$$

and data uncertainty is quantified with:

$$U_d(y^*|\mathbf{x}^*) = \frac{1}{M}\sum_{j=1}^{M} \sigma^{\widehat{\mathbf{W}}_j}(\mathbf{x}^*)^2 \tag{2.20}$$

where again $\widehat{\mathbf{W}}_j$ is sampled from $q(\mathbf{W})$. Figure 2.1 is an illustration of the evaluation process of predictive value and different uncertainty measures.

2.4 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.

Corpus	**Size**	**Average Tokens**	$\|V\|$	**Classes**
Yelp 2013	335,018	151.6	211,245	5
Yelp 2014	1,125,457	156.9	476,191	5
Yelp 2015	1,569,264	151.9	612,636	5
IMDB	348,415	325.6	115,831	10

Table 2.1: Summaries of Yelp 2013/2014/2015 and IMDB datasets. $|V|$ represents the vocabulary size.

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 [34]. Specifically, we use IMDB movie review data [35] and Yelp restaurant review datasets from Yelp Dataset Challenge in 2013, 2014 and 2015. Summaries of the four datasets are given in Table 2.1. Data splits are the same as in [34, 35].

Experiment Setup We implement convolutional neural network (CNN) baselines in both regression and classification settings. CNN model structure follows [36]. 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 [37] 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 [38] 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 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

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 2.2: 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 [36]; MU and DU denote model uncertainty and data uncertainty respectively. Classification results have a similar pattern but the improvements are less obvious.

scores from the true ratings and is defined as:

$$\mathrm{MSE} = \frac{\sum_{i=1}^{N}(\mathrm{gold}_i - \mathrm{predicted}_i)^2}{N} \tag{2.21}$$

Results Experiment results are shown in Table 2.2. 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.

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) [39] neural network as the baseline model and measure the effects of quantifying model

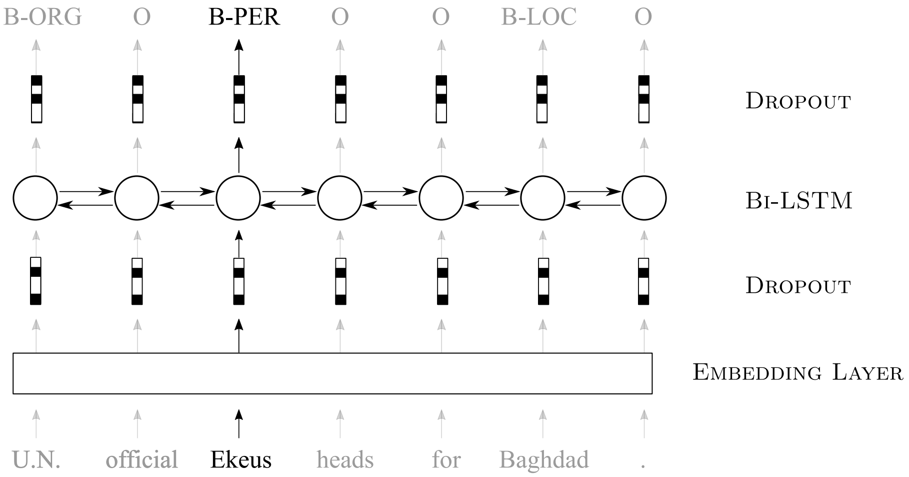


Figure 2.2: An illustration of the bidirectional LSTM model used for named entity recognition. Two dropout layers independently sample their masks while masks are the same across time steps.

and input-dependent uncertainties on the test performances.

Datasets For the NER experiments, we use the CoNLL 2003 dataset [40]. 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 [16]. An illustration of the model is shown in Figure 2.2. 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 proba-

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 2.3: 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

bility is fixed at 0.5; other hyper-parameters related to quantifying uncertainties are the same with previous experiment setups.

For training, we use Adam optimizer [38]. 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:

$$\text{F1} = \frac{2 \cdot \text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}} \tag{2.22}$$

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 2.3. 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.

Experiment Setting We follow the medium model setting in [41]. 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 [16], 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 2.4. We can observe performance improvements when quantifying either model uncertainty or data uncertainty. We observe less performance improvements compared to [16] possibly due to the fact that we use simpler dropout formulation that only applies dropout between layers.

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

Table 2.4: Test set perplexities of LSTM language models trained on PTB dataset. PPL represents perplexity. Baseline is the baseline medium two-layer LSTM model in [41]; MU and DU denote model uncertainty and data uncertainty respectively.

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.

2.5 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 [9].

What Does Data Uncertainty Measure

In Equation 2.3, 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 du
should game automatic doors !
i 've bought tires from discount tire for years at different locations and have had a good 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
great sports bar ! brian always goes out of his way to make sure we are good to go ! great people , great food , great music ! great bartenders and even great bouncers ! always accommodating ! all the best _unk !
great _unk burger ! amazing service ! brilliant interior ! the burger was delicious but it was a little big . it 's a great restaurant good for any occasion .

Table 2.5: 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.

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 2.5 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

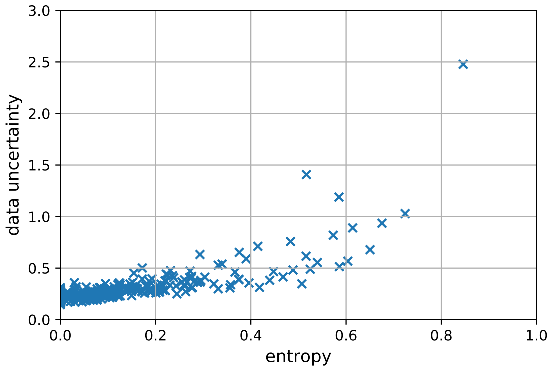


Figure 2.3: Scatter plot of evaluated data uncertainty against entropy of annotated NER tag distribution for all tokens in CoNLL 2003 dataset. Higher input-dependent data uncertainties are estimated for input tokens that have higher tag entropies.

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:

$$H(p_1, p_2, \cdots, p_m) = -\sum_{i=1}^{m} p_i \log p_i \tag{2.23}$$

where $p_1, p_2, \cdots, p_m$ 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 entropy for the tokens in Figure 2.3. It is clear that for tokens with

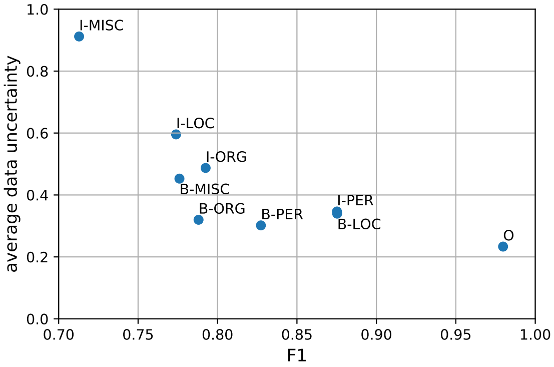


Figure 2.4: Scatter plot of average evaluated data uncertainty against test set F1 score for different tags. Higher data uncertainties are observed when predicting tags with lower F1 score.

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 2.4. 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.

2.6 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. In the next chapter, we investigate the relationship between predictive uncertainty and hallucination. We introduce an uncertainty-aware decoding method to help reduce hallucination in image captioning and table-to-text tasks.

Chapter 3

Hallucination and Uncertainty in Conditional Language Generation

3.1 Introduction

Modern deep neural network models have brought drastic improvements of generation quality measured by standard metrics on different natural language generation (NLG) tasks. However, along with these improvements, researchers find that neural models are more prone to a phenomenon called hallucination, where models generate description tokens that are not supported by the source inputs. This phenomenon seriously damages the applicability of neural language generation models in practice where information accuracy is vital.

Hallucination has been observed in various conditional NLG tasks such as image captioning [42], data-to-text generation [43, 44, 45], abstractive summarization [46, 47], and neural machine translation (NMT) [48]. These studies tackle hallucinations within a specific task and give possible explanations of why hallucinations occur. For example, [42] attributes object hallucination in image captioning to visual misclassification and over-reliance on language priors; [44] believes hallucination in neural surface realization comes from the misalignment between meaning representations and their corresponding references in the dataset; [48] claims that hallucinations in NMT are mainly due to domain shift.

We believe that there is a common theme across all the hallucination explanations in conditional NLG tasks: predictive uncertainty. In language generation, predictive uncertainty quantifies the entropy of the token probability distributions a model predicts. There are multiple sources of uncertainty. Two major ones frequently studied are aleatoric and epistemic uncertainties, where the former comes from the data or measurements, and the latter is concerned with the model. With recent progress in Bayesian neural networks (BNNs) [49, 50] and uncertainty quantification [10, 12, 13], we are able to quantify both parts of predictive uncertainty in neural NLG.

This chapter draws connections between hallucination and predictive uncertainty and empirically investigates their relationship in image captioning and data-to-text generation tasks. We propose an uncertainty-aware beam search algorithm to reduce the chance of hallucination by penalizing parts or the entirety of the predictive uncertainty during model decoding. We find that the choice of uncertainty matters, and penalizing epistemic uncertainty yields better results compared to penalizing aleatoric or total uncertainty. Our contributions are:

- We draw connections between hallucination and predictive uncertainty across various conditional natural language generation tasks and empirically investigate their relationship.
- We propose an uncertainty-aware beam search approach for hallucination reduction to demonstrate that lowering uncertainty can lead to less hallucination.
- We show that uncertainty decomposition helps to achieve better trade-offs between

hallucination and performance.

3.2 Hallucination and Predictive Uncertainty

Hallucination Probability

In general, hallucination refers to the phenomenon where the model generates false information not supported by the input. For example, in the context of image captioning, hallucination can be defined as generating captions that contain descriptions not present in the given image. Let (x, y) be the pair of variables at interest where x is some structured data containing facts and y is a natural language sentence based on the facts. The task is to learn the conditional distribution of $p(y|x)$ in order to generate sentence y given any new input x. Most neural approaches break the probability into a sequence of single token predictions:

$$p(y|x) = p(y_1|x) \prod_{i=2}^{k} p(y_i|x, y_1, \ldots, y_{i-1}) \tag{3.1}$$

where $\{y_1, \ldots, y_k\}$ is the collection of tokens in sentence y. We denote $c_i = \{x, y_1, \ldots, y_{i-1}\}$ as the context of the i-th prediction in the following sections for simplicity.

Apparently, hallucination is context-dependent which means we need to look at a certain context c_i and determine whether the next token prediction y_i is hallucinated or not. Let $\mathcal{V}_h^{(c_i)}$ denote the set of tokens that are considered false information given the current context c_i and $\mathcal{V}$ the whole vocabulary. Consider a random sampling decoder where a token is generated based on the predicted categorical distribution. i.e. $\mathrm{Cat}(|\mathcal{V}|, p(y_i|c_i))$.

The probability of hallucination at the current step is simply:

$$P(y_i \in \mathcal{V}_h^{(c_i)}) = \sum_{v \in \mathcal{V}_h^{(c_i)}} p(y_i = v|c_i) \tag{3.2}$$

Practically, it is hard to automatically determine the context-dependent set $\mathcal{V}_h^{(c_i)}$. Task-specific heuristics are often used to determine which tokens are hallucinated. In specific restrictive applications, the context-dependent set can be relaxed to a context-independent one to reduce the complexity of determining hallucination.

Relationship with Predictive Uncertainty

We use entropy to measure the predictive uncertainty in this chapter. The total uncertainty of predicting token y_i is:

$$\begin{aligned}
&H(y_i|c_i) \\
=& -\sum_{v \in \mathcal{V}} p(y_i = v|c_i) \log p(y_i = v|c_i) \\
=& -\sum_{v \in \mathcal{V} \setminus \mathcal{V}_h^{(c_i)}} p(y_i = v|c_i) \log p(y_i = v|c_i) \\
&- \sum_{v \in \mathcal{V}_h^{(c_i)}} p(y_i = v|c_i) \log p(y_i = v|c_i)
\end{aligned} \tag{3.3}$$

From Equation 3.3, we can see that there are two sources of uncertainty for the token predictions: one from the uncertainty of choosing suitable tokens to describe the input; another from some unsuitable tokens attaining considerable probability mass either by being confusing in the current context or due to an insufficiently trained system.

The second source of uncertainty is directly related to hallucination probability. Although no monotonic relationship can be derived, a near-zero hallucination probability

$$\begin{array}{c|ccc|}
 & \text{Token 1} & \text{Token 2} & \text{Token 3} \\
\text{Prediction 1} & 0.33 & 0.33 & 0.33 \\
\text{Prediction 2} & 0.33 & 0.33 & 0.33 \\
\text{Prediction 3} & 0.33 & 0.33 & 0.33
\end{array}$$

(a)

$$\begin{array}{c|ccc|}
 & \text{Token 1} & \text{Token 2} & \text{Token 3} \\
\text{Prediction 1} & 0.98 & 0.01 & 0.01 \\
\text{Prediction 2} & 0.01 & 0.98 & 0.01 \\
\text{Prediction 3} & 0.01 & 0.01 & 0.98
\end{array}$$

(b)

Figure 3.1: Examples of predictions with (a) high aleatoric but low epistemic uncertainty; and (b) high epistemic but low aleatoric uncertainty.

requires a near-zero value of the second source of uncertainty. This observation prompts us to investigate the relationship between hallucination and predictive uncertainty in practice. Intuitively, the higher the predictive uncertainty is, the more probable some of the probability mass gets assigned to unsuitable tokens.

Uncertainty Decomposition

There are often two types of uncertainties frequently mentioned in uncertainty quantification literature: epistemic and aleatoric uncertainty [8, 9, 51]. Epistemic uncertainty reflects the uncertainty on model weights, and aleatoric uncertainty concerns inherent uncertainty in the data or measurement. We are interested in whether the relationship with hallucination is the same for both types of uncertainties.

Bayesian deep learning approaches [10, 12, 13] are widely studied for uncertainty quantification with neural networks. Following the notations in Section 3.2, the predictive distribution of $p(y_i|c_i)$ can be written as:

$$p(y_i|c_i) = \int_w p(y_i|c_i, w)q(w)dw \tag{3.4}$$

where w parameterizes the neural network that makes predictions and $q(w)$ denotes the approximate posterior distribution of the weights w given the training data. Notice that if we fix the weights w, $H(y_i|c_i, w)$ represents the entropy that is unrelated to the uncertainty of the model weights. Therefore the aleatoric part of the predictive uncertainty can be calculated with $\mathbb{E}_{q(w)}[H(y_i|c_i, w)]$. The epistemic part of the uncertainty is the difference between the total and the aleatoric uncertainty as shown below:

$$u_{al}(y_i|c_i) = \mathbb{E}_{q(w)}[H(y_i|c_i, w)] \tag{3.5}$$

$$u_{ep}(y_i|c_i) = H(y_i|c_i) - \mathbb{E}_{q(w)}[H(y_i|c_i, w)] \tag{3.6}$$

In this chapter, the aleatoric and epistemic parts of predictive uncertainty are estimated using deep ensembles [13]. More concretely, denote the model predictions as $\{p_m(y_i|c_i)\}_{m=1}^{M}$ and the aggregated prediction as $p(y_i|c_i) = \frac{1}{M}\sum_{m=1}^{M} p_m(y_i = v|c_i)$, aleatoric and epistemic uncertainties are calculated as:

$$u_{al}(y_i|c_i) = \frac{1}{M}\sum_{m=1}^{M} H_m(y_i|c_i) \tag{3.7}$$

$$u_{ep}(y_i|c_i) = H(y_i|c_i) - u_{al}(y_i|c_i) \tag{3.8}$$

where $H_m(y_i|c_i)$ and $H(y_i|c_i)$ are the entropy of $p_m(y_i|c_i)$ and $p(y_i|c_i)$ respectively.

Intuitively, in the case of deep ensembles, aleatoric uncertainty measures the average spread of all model predictions, while epistemic uncertainty measures the agreement among all model predictions. Examples with three possible tokens are illustrated in Figure 3.1.

3.3 Case Study: Image Captioning

In this section, we analyze image captioning models trained on MSCOCO [52] data set.

Model	Action hallucination % at uncertainty level					
	≤ 0.8	0.8 - 1.6	1.6 - 2.4	2.4 - 3.2	3.2 - 4.0	> 4.0
FC	0.00	0.00	2.27	12.86	15.71	31.03
Att2In	0.00	0.00	3.39	6.58	12.07	22.03
BUTD	0.00	2.94	1.92	12.77	17.24	25.53
Transformer	2.99	5.48	6.58	8.82	12.00	43.75

Table 3.1: Action hallucination percentages at different levels of predictive uncertainty. Action predictions with higher uncertainty are more prone to hallucination.

Hallucination Probability at Different Uncertainty Levels

The first question we want to investigate is whether hallucination probabilities change at different predictive uncertainty levels. Some experimental settings are listed below.

Model architecture We consider four different image captioning models: **FC** model [53] where image features are used to initialize the RNN decoder; **Att2In** model from [53] applies attention on image features and feeds it into the decoder LSTM [39] cell gate; **BUTD** model from [54] uses bottom-up attention which operates at the level of objects and other salient image regions; **Transformer** model where transformers [1] are used in the encoder-decoder structure for generation. All models are implemented in the open source framework by [55][1].

Training We consider the same data split from [56]. All models are trained with batch size 50 for 30 epochs with Adam optimizer [38]. Evaluations are done on the Karpathy test set.

Hallucination and uncertainty evaluation As in [42], synonyms for all possible MSCOCO objects are used to determine whether an object generated by the captioning model is hallucinated. Hallucination probabilities are calculated by binning all object

---

[1] https://github.com/ruotianluo/self-critical.pytorch

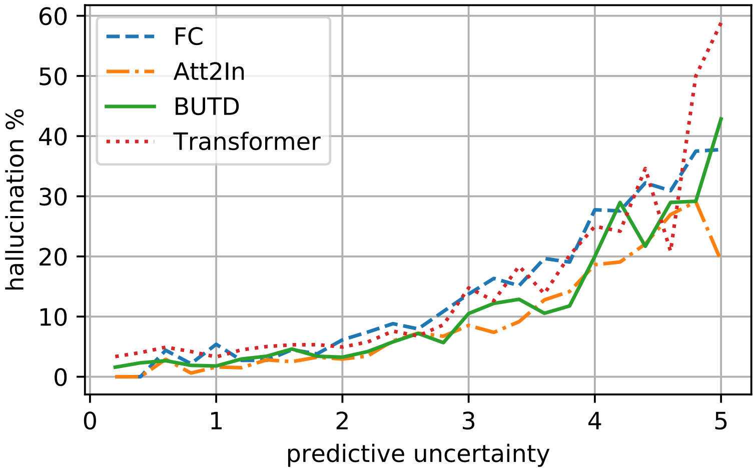


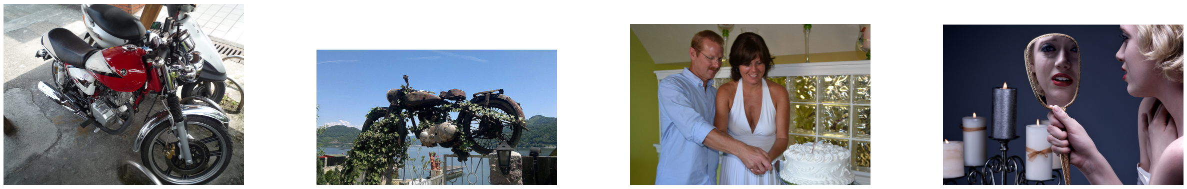
Figure 3.2: Object hallucination chance at different predictive uncertainty levels. Higher predictive uncertainty corresponds to a higher level of hallucination percentage across all models.

token prediction entropy and counting the percentage of hallucinated objects in each bin.

Results and Discussions

Figure 3.2 shows the object hallucination percentages at different predictive uncertainty levels. At higher uncertainty levels, the generated objects are more likely to be hallucinated. The results are consistent across four different models. The transformer model seems to have a higher hallucination chance at high uncertainty levels than the other three models. However, this does not indicate Transformer models hallucinate more. In fact, the transformer model has an overall lowest hallucination percentage among all four models.

Beyond object hallucination Aside from object hallucination, we also analyze verbs generated by the models to see whether a similar relationship holds for other types of token generations. The same models and training procedures are adopted. We extract all

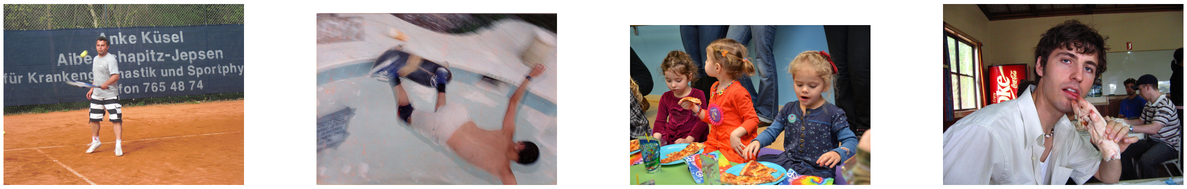


(a) a red and black **motorcycle** (*0.58*) parked in a parking lot

(b) a **motorcycle** (*4.80*) is parked on a dock with a bird perched on top of it

(c) a bride and groom cutting their wedding **cake** (*0.09*)

(d) a woman holding a cup and a **cake** (*5.29*)



(e) a man standing on a tennis court **holding** (*0.81*) a racquet

(f) a young man is **holding** (*4.76*) a skateboard in his hand

(g) a group of children sitting at a table **eating** (*1.00*) pizza

(h) a man is **eating** (*4.01*) a hot dog at a restaurant

Figure 3.3: Examples of token predictions generated with the BUTD model with high and low uncertainty values for objects (top) and actions (bottom). Numbers in italic are predictive uncertainty values for the token predictions preceding them. The examples are cherry-picked.

present continuous tense verbs from the generated captions using spaCy part-of-speech tagger[2] and manually label whether they are suitable to describe the corresponding images. There are approximately 3500 generated captions containing verbs, and 400 are annotated for each model. We refer to unsuitable verbs generated in the captions as action hallucinations.

Action predictions are binned according to their uncertainty values, and the results are shown in Table 3.1. We can observe that action tokens with higher predictive uncertainty are also more likely to be hallucinated. Noticeably, the transformer model also has a higher action hallucination rate at high uncertainty levels.

Examples of predictions with high and low uncertainty Figure 3.3 shows some example images and their captions generated from a BUTD model on the test set. The

---

[2] https://spacy.io

Model	**Correlation coefficient** epistemic	 aleatoric
FC	0.313	0.299
BUTD	0.334	0.228
Att2In	0.360	0.268
Transformer	0.269	0.131

Table 3.2: Pearson correlation coefficients between hallucination and epistemic/aleatoric uncertainty in image captioning task. Epistemic uncertainty is more indicative of hallucination across four models.

token predictions of interests and the corresponding uncertainty values are highlighted in bold and italic, respectively. We observe that highly uncertain predictions often correspond to unusual textures, features resembling the predicted tokens, or blurred images. For example, Figure 3.3(b) shows a motorcycle covered in vines; Figure 3.3(d) shows candles in the background which resemble cakes; Figure 3.3(f) is blurred.

Epistemic and aleatoric uncertainties As we could decompose the total uncertainty into two parts, we are interested in which part is more indicative of hallucination. Table 3.2 shows the Pearson correlation coefficients between hallucination (binary) and epistemic/aleatoric uncertainty for all four models. We can see that both parts of uncertainty are weakly correlated with hallucination, while epistemic uncertainty is more indicative of hallucination across all four models compared to aleatoric uncertainty.

3.4 Case Study: Data-to-text Generation

Data-to-text generation [57, 58] is a task to generate textual content conditioned on input content in the form of structured data such as tables. Neural models are prone to hallucination in data-to-text generation tasks compared to traditional template-based systems, and methods are proposed to improve faithfulness [43, 44, 59]. In this section, we discuss the relationship between predictive uncertainty and hallucination in data-to-text generation with ToTTo dataset [45].

Generation Quality and Average Uncertainty

We conduct token-level analysis in Section 3.3. Now we take a different route and analyze sentence-level quality with different average predictive uncertainty values. Experiment settings are described below.

Dataset ToTTo dataset consists of tables from English Wikipedia articles with their corresponding metadata, such as page title and section title. Candidate description texts are modified by annotators to pair with each table. Relevant table cells supporting the description texts are highlighted by the annotators as well. There are 120,761 table-text pairs in training, 7,700 in validation, and 7,700 in test. We use the baseline standard linearization approach to represent the highlighted portions of the tables along with their corresponding metadata (referred to as subtable with metadata in [45]).

Model architecture and training We use a standard sequence-to-sequence model with attention [60, 55] for analysis. LSTM with 512 hidden size is used for both the encoder and the decoder. Adam optimizer with learning rate 1e-3 is used for the optimization. The model is trained with cross-entropy loss for 20 epochs. The checkpoint with the best validation loss is chosen for the evaluation. The implementation is done using fairseq [61][3].

Evaluation We evaluate the average predictive uncertainty for all generated sentences in the validation set and select the top, bottom, and middle 5% for comparison. BLEU score [62] is used as an automatic metric to evaluate the similarity to the references;

---

[3] https://github.com/pytorch/fairseq

Unc. Level	Avg Unc.	BLEU	Faithfulness (%)	Less/Neutral/More Coverage
High	1.83 - 3.74	10.2	41.3	79.4 / 15.9 / 04.7
Medium	0.83 - 0.89	31.5	78.9	35.2 / 47.9 / 16.9
Low	0.04 - 0.27	72.8	99.0	22.2 / 70.1 / 07.7

Table 3.3: Evaluation results for candidates with high, medium, and low average predictive uncertainty values for ToTTo validation set. Unc. denotes uncertainty. Higher uncertainty candidates have lower quality and higher chance of being hallucinated/unfaithful w.r.t. the input tables.

further manual annotations are done to evaluate the *fluency*, *faithfulness* (precision), and *coverage with respect to reference* (recall) of the generated sentences. Particularly, faithfulness reflects how likely the generated sentences hallucinate facts that are not supported by the tables. More details of the human evaluation metrics are described in [45]. The goal is to measure how different the generation qualities are for candidates with varying average predictive uncertainties.

Results and Discussions

Table 3.3 summarizes the evaluation results for candidates with varying uncertainty values. It is obvious that candidates with higher average predictive uncertainty values are less fluent and more likely to contain hallucinations. Another interesting observation from Table 3.3 is that the generated sentences with medium average uncertainty are more likely (16.9%) to cover more table facts than the references compared to the ones with high (4.7%) and low (7.7%) average uncertainty. One possible explanation is that some table facts that are not always included in the references, when generated, have higher predictive uncertainty values than the facts that are almost always included in the references. Therefore, generated sentences with low uncertainty tend to include less but more confident facts considered by the model.

3.5 Reducing Hallucination

Uncertainty-Aware Beam Search

Because of the positive correlation between hallucination probability and predictive uncertainty, it is straightforward to incorporate uncertainty into the caption generation process to reduce hallucination. Beam search is the most used approximate decoding method in language generation. It keeps track of the top-B scored candidates at each generation step and considers all single token extensions of the current candidates.

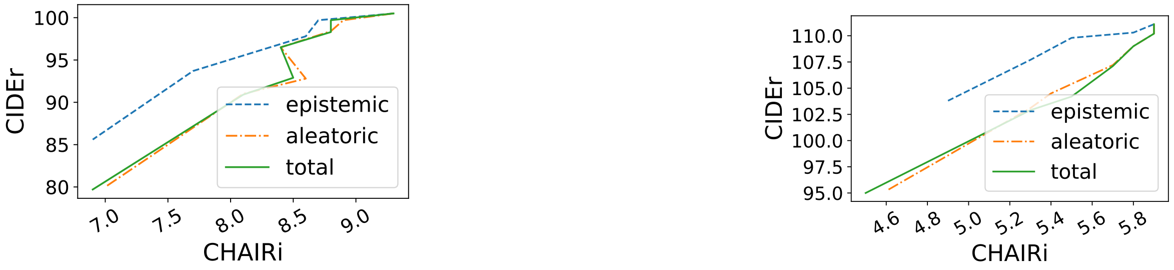
More formally, denote the set of B candidates in the beam at time step $t-1$ as $\mathcal{Y}_{t-1} = \{\mathbf{y}_{t-1}^{(b)}\}_{b=1}^{B}$. All possible single token extensions of the candidates in $\mathcal{Y}_{t-1}$ form a set $\mathcal{C}_t = \{\mathbf{y} \mid \mathbf{y}_{t-1} \in \mathcal{Y}_{t-1} \wedge y_t \in \mathcal{V}\}$. Beam at step t is then formed as:

$$\begin{aligned} \mathcal{Y}_t = & \underset{\mathbf{y}_1 \cdots \mathbf{y}_B \in \mathcal{C}_t}{\arg\max} \sum_{b=1}^{B} \log p(\mathbf{y}_b|\mathbf{x}) \\ & s.t. \quad \mathbf{y}_i \neq \mathbf{y}_j \quad \forall i \neq j \end{aligned} \tag{3.9}$$

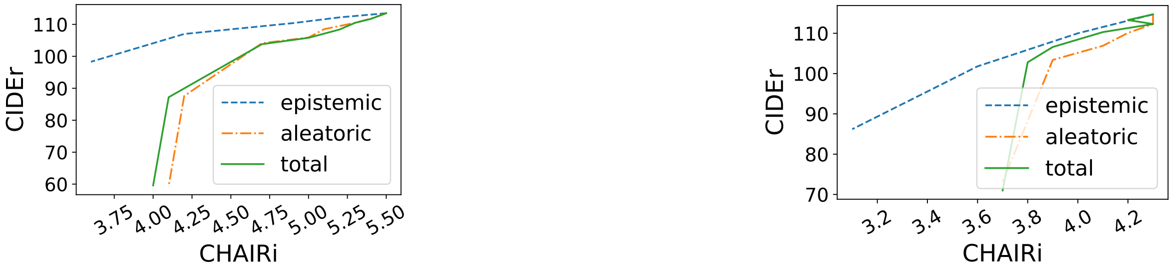
Uncertainty-aware beam search (UABS) adds a weighted penalty term in the beam search objective to balance between log probability and predictive uncertainty of the selected candidates. Let $u(\mathbf{y}|\mathbf{x})$ be the function to measure the aggregated predictive uncertainty of candidate $\mathbf{y}$ given input $\mathbf{x}$, uncertainty-aware beam search updates the beam at step t according to the following equation:

$$\begin{aligned} \mathcal{Y}_t = & \underset{\mathbf{y}_1 \cdots \mathbf{y}_B \in \mathcal{C}_t}{\arg\max} \sum_{b=1}^{B} \log p(\mathbf{y}_b|\mathbf{x}) - \lambda u(\mathbf{y}_b|\mathbf{x}) \\ & s.t. \quad \mathbf{y}_i \neq \mathbf{y}_j \quad \forall i \neq j \end{aligned} \tag{3.10}$$

where $\lambda \geq 0$ is the weight controlling the degree to which we want to penalize decoding



(a) FC (b) Att2In



(c) BUTD (d) Transformer

Figure 3.4: CIDEr plotted against CHAIRi scores of captions generated with UABS with different uncertainty penalty weights. Lower CHAIRi score indicates less hallucination. Upper-left is better. Penalizing epistemic uncertainty in UABS achieves the best results.

uncertainty. Larger λ leads to candidates with smaller predictive uncertainty. In practice, this can be done by subtracting the weighted uncertainty term from the aggregated log probability scores at each decoding step before choosing top-B candidates.

An important decision in using uncertainty-aware beam search is the choice of uncertainty term $u(\mathbf{y}|\mathbf{x})$. We could use either the aleatoric or epistemic part of the predictive uncertainty or both. We compare these choices and discuss the results in the next section.

Image Captioning Results

With larger weights on the uncertainty penalty term, log probabilities of the decoded sentences drop. Therefore, we expect to see a trade-off between the quality of generated captions and the chance of hallucination.

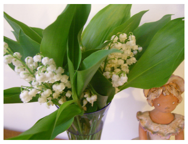
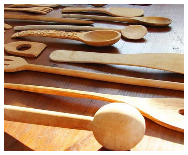
Image	**UABS results with weight λ**		
	0	20	80
	a vase filled with flowers sitting on top of a table	a vase filled with lots of white flowers	there is a vase that has flowers in it
	a wooden cutting board topped with lots of food	a wooden cutting board topped with lots of food	a cutting board that has a bunch on it

Table 3.4: Two examples of epistemic UABS results with varying penalty weights on the image captioning data set. In the first example the model successfully avoids hallucination of a table with $\lambda = 20$ while in the second example it is unable to change the generated caption until larger penalty weight is set.

We empirically examine the trade-offs on the image captioning models with different uncertainty choices for the penalty term. We use a five-model ensemble for each of the four model architectures to estimate aleatoric and epistemic uncertainties. Due to the different magnitudes of aleatoric and epistemic uncertainties, we choose penalty weight λ from $[0.1, 0.2, 0.4, 0.8, 1.0, 2.0, 4.0]$ for aleatoric and total uncertainty and $[10, 20, 40, 80]$ for epistemic uncertainty.

Figure 3.4 shows the trade-offs between CIDEr [63] and CHAIRi [42] scores of captions generated with uncertainty-aware beam search with different uncertainty choices and penalty weights. A smaller value of CHAIRi indicates the model is less likely to generate hallucinated objects, and a higher CIDEr indicates better caption quality. Therefore an approach that is to the upper left of another is better. As the penalty weight increases, we observe a decrease in both the CHAIRi and the CIDEr scores across all models.

Table 3.4 shows two examples of different generated captions using epistemic UABS with varying penalty weights. In the first example, we can see that a medium penalty weight of 20 not only helps avoid the hallucination of a table but also adds correct information about the color of the flowers. In the second example, a medium penalty

	λ	**avg. len.**	**# obj.**	**hal. %**	**gen. %**
ref.		10.44	6114	0	-
base	0	9.31	7328	5.5	0
epist.	10	9.21	7195	5.2	0
	20	9.16	7078	4.9	0.2
	40	9.15	6912	4.2	1.5
	80	9.12	6493	3.6	4.6
aleat.	0.1	9.32	7250	5.4	0
	0.4	9.32	7051	5.1	0
	1.0	9.33	6800	4.7	1.0
	4.0	9.43	4349	4.1	28.4

Table 3.5: Average sentence length and total number of objects detected in the captions generated by BUTD model with varying uncertainty penalty weight λ. Penalizing epistemic uncertainty leads to slightly shorter lengths. Number of objects mentioned by the captions decreases with increasing λ. **gen. %** denotes percentage of generic responses. It is moderate with epistemic penalized results but can be very high if aleatoric uncertainty is heavily penalized.

λ	**BLEU**	**Fluency (%)**	**Faithfulness (%)**	**Less/Neutral/More Coverage**
0	40.1	92	79	34 / 60 / 6
10	33.6	83	84	41 / 51 / 8
20	27.4	73	80	52 / 42 / 6

Table 3.6: Evaluation results for candidates decoded with different penalty weights for UABS on ToTTo validation set. Epistemic uncertainty is used for uncertainty penalization. Faithfulness first increases, then decreases to the same level as regular beam search results as we increase the penalty weight λ.

weight is unable to change the generated caption.

Regarding the choice of uncertainty, it is notable that when penalizing epistemic uncertainty, the generated captions achieve higher CIDEr scores than penalizing aleatoric or total uncertainty. We hypothesize that epistemic uncertainty indicates the uncertainty of model weights. By penalizing epistemic uncertainty, we encourage the model to take the prediction path where it is well-calibrated. On the other hand, penalizing aleatoric uncertainty encourages the model to make low entropy predictions in all contexts regardless of the actual data distributions.

Reference	**UABS results with weight λ**		
	0	10	20
barrows scored 164 net points in virgin islands at the 2008 summer olympics.	in virgin islands at the 2008 summer olympics, barrows iii received 164 points.	in virgin islands at the 2008 summer olympics, barrows received 164 points.	thomas barrows received a total score of 164.
janet gaynor won the first academy award for best actress for her performance in the 7th heaven (1927 film).	janet gaynor won the academy award for best actress for his performance in janet gaynor.	janet gaynor won the academy award for best actress.	janet gaynor won an academy award for best actress.

Table 3.7: Two examples of UABS results with varying penalty weights on the ToTTo validation set. Blue tokens are correct table facts that are dropped by candidates generated with larger penalty weights; red tokens are incorrect/hallucinated facts that are dropped with larger penalty weights. In general, UABS with larger weights tend to produce sentences with less information that the model is more confident with.

Table 3.5 shows the average sentence length, the number of objects, the percentage of hallucinations, and the percentage of generic responses in the captions generated by the BUTD model with different uncertainty choices and penalty weights on the test set. We can see that when penalizing epistemic uncertainty, UABS results in slightly shorter caption candidates. Both the number of objects and hallucination percentage decrease as we increase the weight λ. Interestingly, when penalizing aleatoric uncertainty, sentence length stays approximately the same despite lower CIDEr scores, as shown in Figure 3.4. Further investigation shows that this is partly due to an increasing number of generic captions such as "*there is no image here to provide a caption for*". Penalizing epistemic uncertainty is much less likely to result in such generic captions. We can see that when increasing λ from 1.0 to 4.0 with aleatoric UABS, the percentage of generic responses jumps drastically from 1.0% to 28.4%. In comparison, epistemic UABS keeps the generic response rates low while achieving lower hallucination rates.

Data-to-text Results

We also evaluate the effect of UABS on the ToTTo dataset. We choose to penalize epistemic uncertainty due to its better performances than aleatoric uncertainty, as shown in the previous section. A five-model deep ensemble is used to quantify the epistemic uncertainty and generate results with UABS. We compare the BLEU score and three human evaluation metrics among results generated with different uncertainty penalty weights. 100 generation results are randomly selected and evaluated for each penalty weight choice. The results are shown in Table 3.6. We can see that a relatively small penalty weight leads to a reduced hallucination chance (hence more faithful) with a cost on the BLEU score and fluency.

To qualitatively examine the sentences generated with different λ values, we show example results on the ToTTo validation set in Table 3.7. We can see that with larger penalty weights, the UABS results drop certain statements that the model deems less confident regardless of the correctness. This results in shorter but more confident predictions for UABS results with a larger uncertainty penalty.

3.6 Related Work

Hallucination There are many pieces of anecdotal evidence of hallucination presented in various NLG tasks. Most recently, researchers started investigating the phenomenon systematically. [42] analyzes object hallucination focusing on the objects that appeared in the MSCOCO segmentation challenge. They propose the CHAIR metric to quantify the severity of object hallucination. They find that the models tend to make predictions consistent with a language model trained on the captions instead of a model trained to predict objects in an image. Therefore hallucination is caused by an over-reliance on the language priors. [44] believes that the origin of the hallucination problem in neural surface realization comes from the data side. More specifically, datasets used for NLG systems often include instances with information misalignment between the input structure and the output text. They propose integrating a language understanding module for iterative data refinement to better align meaning representations and output text. [48] examines hallucination in neural machine translation and observes that the phenomenon is most common in out-of-domain settings. They empirically compare several strategies to improve domain robustness in NMT and find that a combination of reconstruction and a noisy channel model for reranking is most effective.

These observations are consistent with our findings. For example, domain shift and data misalignment are known to lead to a higher level of epistemic uncertainty [9] which makes hallucination a more severe problem.

Uncertainty quantification Uncertainty quantification has attracted more attention recently due to the progress in Bayesian deep learning. Bayes by backprop [10], Monte Carlo dropout [12], and deep ensembles [13] are examples of popular Bayesian approaches to evaluate uncertainty with deep neural models. [9] investigates the benefits of modeling epistemic and aleatoric uncertainty in vision tasks such as semantic segmentation and depth regression. They show that it is important to model aleatoric uncertainty with large datasets and real-time applications and epistemic uncertainty with small datasets and safety-critical applications. Other applications of uncertainty quantification have been explored in the context of time series predictions [17], natural language processing tasks [64], etc. More broadly, prediction entropy has been analyzed in different neural language generation tasks [65, 66]. [51] shows how to extract and decompose uncertainty in Bayesian neural networks with latent variables for decision-making purposes. They show that active learning and risk-sensitive reinforcement learning both benefit from uncertainty decomposition.

3.7 Discussion and Conclusions

We investigate the relationship between hallucination and predictive uncertainty in image captioning and data-to-text generation tasks and show that predictions with higher uncertainty are more prone to hallucination. In particular, epistemic uncertainty is more indicative of hallucination than aleatoric uncertainty. We propose uncertainty-aware beam search to incorporate uncertainty into the decoding process to reduce hallucination. We show that uncertainty decomposition helps the proposed beam search variant to achieve a better performance-hallucination trade-off. Specifically, penalizing epistemic uncertainty yields better results compared to penalizing aleatoric or total uncertainty.

In this chapter, we analyze uncertainty from the token level. This might be restrictive because uncertainty corresponds to the current prediction context instead of the predicted token. The relationship between hallucination and uncertainty, therefore, can be much more complicated than a linear one. It is still possible to produce hallucinated information with a very confident model. The proposed UABS reduces hallucination by limiting the total uncertainty of the generated text. As a result, it might lead to shorter generations and lower generation quality. Devising more sophisticated uncertainty-aware training and decoding methods with less adverse effects on the generation quality is a future direction to explore.

In addition to predictive uncertainty, model calibration is another concept that affects the reliability and interpretability of model predictions. In the next chapter, we discuss the relationship between model calibration and label smoothing.

Chapter 4

Label Smoothing and Model Calibration in Text Classification

4.1 Introduction

Label smoothing has been widely used in deep learning models across various tasks, including image classification [67] and machine translation [1] to improve model performance. Recently, there are several studies analyzing various properties of label smoothing [19, 68, 69]. Specifically, [19] conduct a set of experiments discussing the effects of label smoothing on representation learning and knowledge distillation. They find that label smoothing has an implicit model calibration effect. Models trained with label smoothing are better calibrated on two image classification tasks and the EN-DE translation task. However, it is unclear whether this implicit calibration effect is universal. How and when does label smoothing help with model calibration are not evident.

In this chapter, we investigate the role label smoothing plays with respect to model calibration in text classification settings. We train BERT [70] and deep averaging network (DAN) [71] models on four text classification data sets with different label smoothing factors and further analyze the effects of label smoothing on the model predictions. Our contributions are:

- We show that label smoothing does not always improve model calibration with empirical experiments on several text classification data sets.
- We demonstrate that label smoothing artificially suppresses model uncertainties and pushes the model to produce higher entropy predictions. This helps to calibrate the model when there is a significant gap between training and validation losses.
- We find that in the settings where label smoothing yields worse calibrated models, it is still possible to produce better-calibrated models by reversing the effect with label sharpening.

4.2 Preliminaries

Model Calibration

In a supervised classification setting, the accuracy of the predictions is often the most critical metric. However, calibrating the model prediction confidence to better represent the actual prediction correctness is becoming an essential task when accurate measurements of the test-time prediction correctness are required.

Let (x, y) be the input-label pair that follows a ground-truth distribution $\pi(x, y) = \pi(x)\pi(y|x)$; $q(y|x, w)$ denotes the conditional model prediction given input x and parameter w. The ultimate goal of model calibration in classification tasks is to match $q(y|x, w)$ with $\pi(y|x)$. In practice, the discrepancy between the model predictions and the ground-truth distribution can be approximated using negative log-likelihood (NLL) of the data samples.

Expected Calibration Error (ECE) [72, 3] is a popular metric to measure miscalibration. It calculates the expected difference between model confidence and accuracy, i.e.

$$\mathbb{E}_p\big[\big|P\big(y=\hat{y}\mid q(y|\hat{x},w)=p\big)-p\big|\big] \tag{4.1}$$

where $(\hat{x},\hat{y})$ are data examples. ECE is often approximated by binning all model predictions by confidence scores and calculating the difference between confidence and average accuracy within each bin.

Label Smoothing

Cross-entropy loss is widely used to train neural network models on classification tasks. For K-class classification and an input-label pair $(\hat{x},\hat{y})$, the cross-entropy loss is calculated as

$$\sum_{k=1}^{K} -1_{[k=\hat{y}]}\log q(y=k|\hat{x},w) \tag{4.2}$$

where $1_{[k=\hat{y}]}$ is 1 when $k=\hat{y}$ and 0 otherwise.

When applying label smoothing with a factor of α, we instead minimize the cross-entropy loss between the prediction and the modified soft target $y_k^{LS}=1_{[k=\hat{y}]}(1-\alpha)+\alpha/K$:

$$\sum_{k=1}^{K} -y_k^{LS}\log q(y=k|\hat{x},w) \tag{4.3}$$

Label smoothing has been found to boost model performance, especially in machine translation tasks with complex neural models.

4.3 Does Label Smoothing Always Help with Calibration?

We first investigate the question of whether label smoothing improves model calibration across different data sets and models. We experiment on the following data sets:

AG's News [73]: 4 topic classes. 120000 training and 7600 test examples. We randomly select 7600 examples from the training set for validation.

CoLA [74]: binary classification data set for linguistic acceptability. 8551/1043/1063 examples for training/validation/test.

20 Newsgroups : newsgroup articles from 20 different newsgroups. 9034/2259/7528 documents for train/validation/test.

Stanford Sentiment Treebank (SST) [75]: Movie reviews represented as sentiment annotated parse trees. We use the original sentences as input. There are 8544/1101/2210 examples for train/validation/test with fine-grained sentence-level sentiment labels (SST-5), and 67349/872/1821 for train/validation/test with binary labels (SST-2).

For CoLA, we use BERT models [70] trained from scratch. For 20 Newsgroups, we train Deep Averaging Networks (DANs) [71] with three feed-forward layers. To compare different models on the same data set, we train both types of models on AG's news, SST-2 and SST-5. Models are trained with AdamW [76, 77]. Early stopping is applied monitoring the validation F1 score.

We evaluate the calibration performances of the models under different levels of label smoothing factors using ECE. The test set results are listed in Table 4.1. We observe an advantage of applying label smoothing when training BERT models on most of the data

Model	**Dataset**	**# classes**	**ECE (%) with label smoothing** $\alpha = 0$	$\alpha = 0.1$	$\alpha = 0.2$	$\alpha = 0.4$	**Better calibrated**
BERT	AG's News	4	1.37	2.70	6.06	12.56	No
	CoLA	2	23.63	16.43	14.23	8.32	Yes
	SST-2	2	14.15	10.09	6.26	6.13	Yes
	SST-5	5	15.57	14.10	9.79	6.56	Yes
DAN	AG's News	4	4.59	8.28	11.54	17.18	No
	20 Newsgroups	20	2.47	3.28	4.07	4.97	No
	SST-2	2	8.63	11.75	14.39	18.19	No
	SST-5	5	3.18	4.19	4.97	6.44	No

Table 4.1: Comparison of ECE scores under different levels of label smoothing strengths. α is the label smoothing factor. Results are averaged over 5 random runs. Label smoothing hurts model calibration for DAN models and helps in most cases for BERT models.

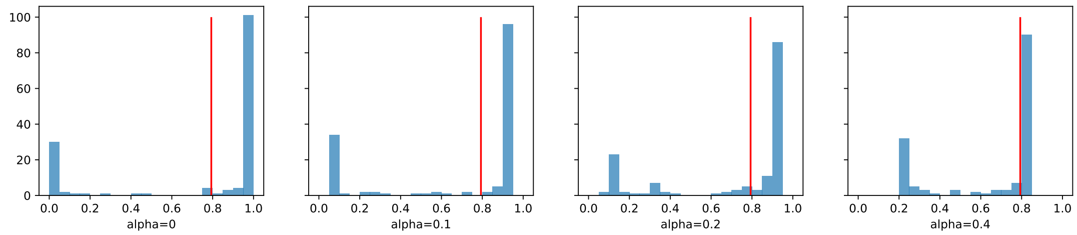
sets. It is also obvious that it hurts model calibration to apply label smoothing when training DANs.

4.4 Why Does Label Smoothing (Not) Help?

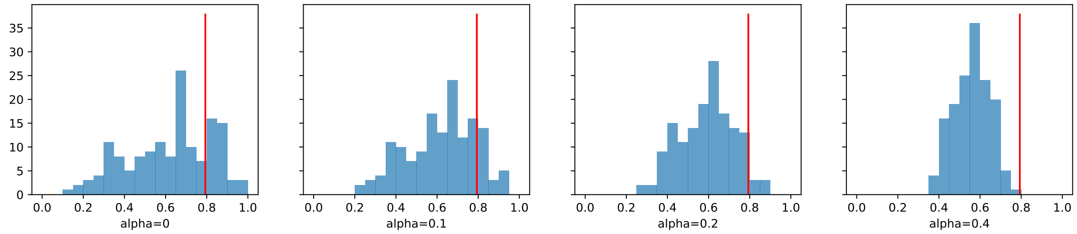
In this section, we investigate the reasons behind the drastic difference between the effects of label smoothing on the calibration of BERT and DAN models.

We start by analyzing the differences from a particular test input x. We take an input sentence from SST-2 test set; query its top-150 nearest neighbors in the pre-trained BERT embedding space; approximate the ground-truth conditional distribution $\pi(y|x)$ by aggregating the labels of its neighbors. We then compare the distributions of prediction confidences from models trained with different levels of label smoothing factors. The distributions of model confidences for the positive class with four different levels of label smoothing α values are plotted in Figure 4.1.

We can see that despite a similar average confidence, BERT and DAN models produce wildly different distributions of confidences given identical inputs. The BERT model tends to have extremely low-entropy predictions while the confidences of the DAN model



(a) prediction confidences of a BERT model



(b) prediction confidences of a DAN model

Figure 4.1: Histograms of prediction confidences for a certain input neighborhood on SST-2 under different levels of label smoothing for (a) BERT and (b) DAN models. Red vertical lines represent the approximated ground-truth conditional probability. The distribution of confidences from the BERT model is bi-modal, while the DAN model predictions are uni-modal. Label smoothing leads to lower-variance higher-entropy predictions.

spread close to a normal distribution.

Due to the softening nature of label smoothing, it artificially suppresses model uncertainty by pushing all predictions to higher-entropy ones. This can be qualitatively observed from Figure 4.1 and empirically confirmed by measuring and comparing the epistemic and aleatoric uncertainties during predictions. As a result, model calibration is almost guaranteed to improve when the ground-truth probability falls between the two posterior modes, as shown in Figure 4.1(a). However, when the posterior distribution is uni-modal and already has a mean close to the ground-truth, label smoothing can hurt model calibration by enlarging the prediction bias as shown in Figure 4.1(b).

Model	**Dataset**	**Train NLL**	**Valid NLL**
BERT	AG's News	0.18	0.26
	CoLA	0.21	1.10
	SST-2	0.07	0.32
	SST-5	0.40	2.71
DAN	AG's News	0.48	0.38
	20 Newsgroups	0.33	0.25
	SST-2	0.46	0.40
	SST-5	1.34	1.35

Table 4.2: Training and validation loss when the validation F1 score peaks. Results are averaged over 5 random runs. The gap between validation and training losses is a clear indicator whether the model can benefit from label smoothing.

4.5 When Does Label Smoothing Help?

We have described the differences between the posterior distributions of BERT and DAN models in the previous section. These differences lead to different interactions between label smoothing and model calibration. In this section, we further investigate the causes of such differences and how we can identify a model that can potentially benefit from label smoothing for model calibration.

If a model is well-trained at some specific area in the input space, the cross-entropy loss will encourage a uni-modal prediction with small variance around the ground-truth value. Therefore, the bi-modal distribution illustrated in Figure 4.1(a) indicates that the model might have a relatively large NLL on the test set. To confirm this conjecture, we record the training/validation loss of models trained without label smoothing on all data sets. The results are listed in Table 4.2. We can see that when the validation loss is comparable to the training loss, label smoothing hurts model calibration; when the validation loss is significantly larger than the training loss, label smoothing leads to better-calibrated models.

Table 4.2 suggests that for a complex model like BERT, it is often the case that

Dataset	**ECE** ($\alpha = 0$)	**ECE** / α
AG's News	4.59%	2.74% / -0.04
20 Newsgroups	2.47%	1.84% / -0.2
SST-2	8.63%	5.72% / -0.1
SST-5	3.18%	1.54% / -0.2

Table 4.3: ECE scores of DANs trained without and with label sharpening. On all four data sets, label sharpening with an appropriate negative α value leads to better calibrated models.

the trained model is overfitted, in the calibration sense, on the training data if the model selection is solely based on validation accuracy or F1 score. In these cases, label smoothing improves model calibration by pushing the posterior modes in the low-entropy areas of the probability simplex to be closer to the ground-truth value.

4.6 Label Sharpening

In this section, we investigate the question: in the cases where label smoothing hurts model calibration, is it possible to reverse its effects by instead sharpening the labels?

To reverse the effect of label smoothing, we use the same cross-entropy loss in Equation 4.3 except that the soft target y_k^{LS} is obtained using negative α values. This is equivalent to a regular cross-entropy loss with hard labels plus an additional term encouraging the divergence between the prediction and the uniform distribution. We call label smoothing with negative α values label sharpening.

We experiment with DAN models trained on AG's news, 20 Newsgroups, SST-2, and SST-5. Models are trained following the same training procedure described in Section 4.3. The label sharpening factor α is selected from [-0.04, -0.1, -0.2] to minimize the validation ECE score. We report the test set ECE scores in Table 4.3.

We can see from Table 4.3 that in all four cases where label smoothing leads to worse calibration, we can effectively reverse the effects to improve model calibration with label sharpening. However, it is worth noting that excessive label sharpening has an adverse impact on both calibration and classification performance.

4.7 Discussion and Conclusions

In this chapter, we investigate and answer three key questions regarding label smoothing and model calibration for text classification. We find that label smoothing does not always lead to better-calibrated models; label smoothing helps only when there is a significant gap between training and validation loss which happens more often when the model is complex; when label smoothing hurts calibration, it is possible to reverse the effects with label sharpening.

We focus on the training-time calibration effects of label smoothing in this chapter. It is also possible to apply label smoothing during re-calibration to potentially better re-calibrate a trained model. Many related questions can be investigated further. For example, how does label smoothing interact with model calibration methods such as temperature scaling and model ensembles? Are other soft-label methods such as Mixup [78] exhibit similar behavior like label smoothing? Can we devise an adaptive approach that automatically selects appropriate label smoothing factor which yields well-calibrated models? These are interesting future directions towards better understandings of label smoothing and model calibration.

As of now, we discussed the uncertainty and calibration of predictions from deep neural models in NLP tasks. In the next chapter, we will discuss out-of-distribution robustness. In particular, we will introduce a benchmark for label distribution estimation/quantification learning.

Chapter 5

Label Distribution Estimation Under Real-World Temporal Shift

5.1 Introduction

Quantification is a supervised learning task that estimates the aggregated label distribution of a test population given labeled training examples. Different from classification tasks, quantification learning focuses on accuracy on the aggregate level. For example, hate speech [79, 80, 81] is a major challenge for many online social platforms. Classifiers can be adopted to identify and demote individual contents categorized as hate speech on such platforms. On the other hand, quantifiers are more useful in estimating the prevalence of hate speech for a collection of contents. In other words, classifiers are mainly used in enforcement and quantifiers are suitable for measurement as illustrated in Figure 5.1. With the help of a robust quantifier, such a platform could evaluate the effectiveness of a newly developed anti-hate speech feature using A/B testing and closely monitor future hate speech prevalence on the platform. Another example of quantification learning in epidemiology is to track the prevalence of clinical reports where a specific pathology is

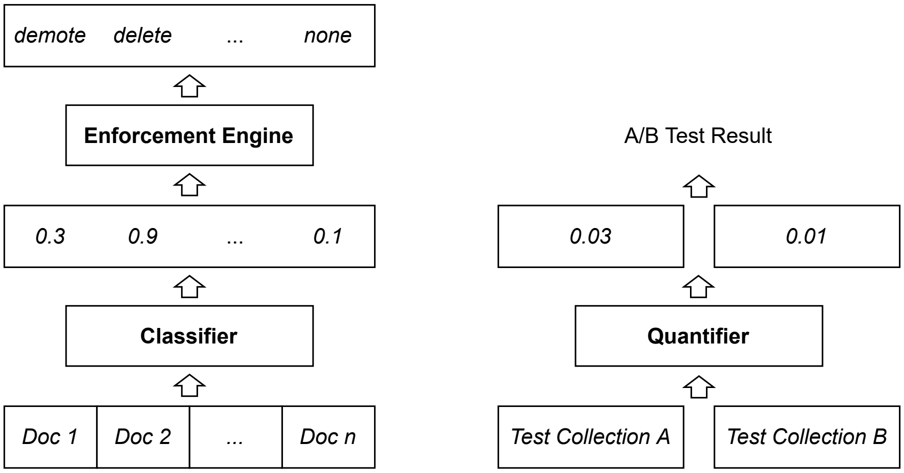
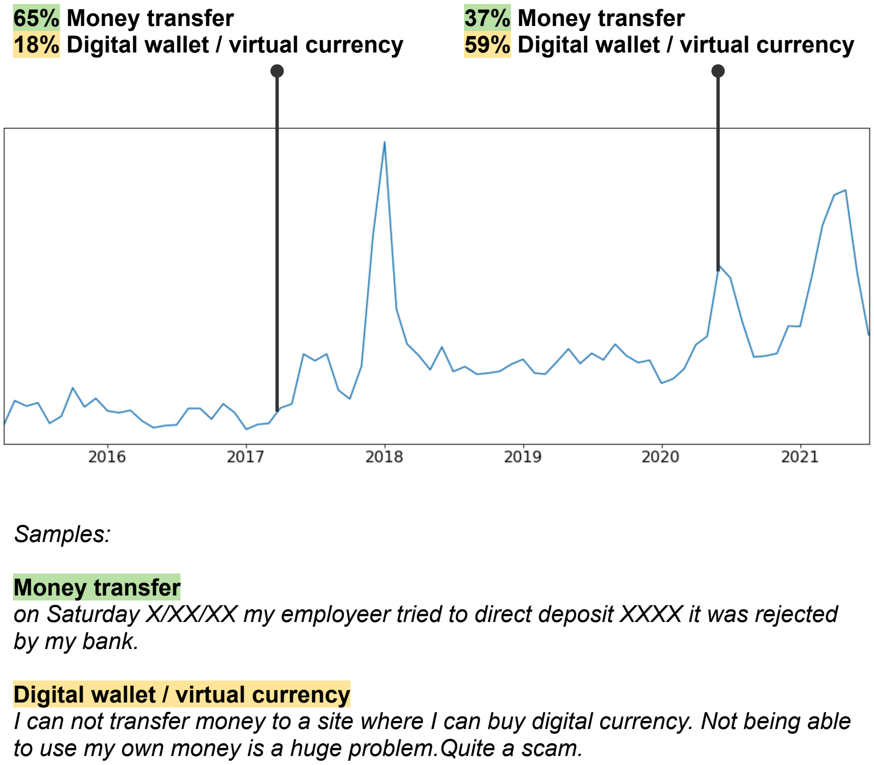


Figure 5.1: Simplified illustration of possible roles of classification versus quantification on a social platform. Classification focuses on individual classification correctness and is often used in enforcement, while quantification focuses on aggregate level estimation and is suitable for measurement.

diagnosed [82]. In both cases, an accurate estimation of the label distribution provides actionable insights.

In addition to these direct use cases, quantification learning is studied in label shift settings [24, 83] to optimize classification performance under distribution shift. In the extreme case where label prevalence is the only difference between the source and target domain, simply re-weighting the examples according to label prevalence ratios in both domains is sufficient to adjust for the domain change. [25] further shows that an accurate quantifier helps existing domain adaptation methods to adjust for label prevalence changes in the target domain through importance weighting.

Despite the many applications, quantification is relatively understudied in the NLP community. One common misunderstanding is that these problems can be solved trivially using a straightforward Classify & Count (CC) approach [84] based on an off-the-shelf classifier. However, classifiers are often trained with the assumption that the training and test examples are drawn i.i.d. from a common data distribution. Under distribution shift, simple aggregation of classification results yields sub-optimal prevalence estimate.



Samples:

Money transfer
on Saturday X/XX/XX my employeer tried to direct deposit XXXX it was rejected by my bank.

Digital wallet / virtual currency
I can not transfer money to a site where I can buy digital currency. Not being able to use my own money is a huge problem.Quite a scam.

Figure 5.2: Prevalence change of the "Money transfer, virtual currency" category across time in the Consumer Complaint Database. Within the category, the composition of complaints also changes, creating further challenges to quantifiers.

One main problem with recent studies on quantification is the dataset, especially the testing sets. As pointed out in [85], quantification methods need to be evaluated on a set of testing splits with enough variations on the label distribution. Most of the above-mentioned studies achieve this by artificially changing the test label distribution through biased sampling. For example, [84] uses a set of pre-specified positive prevalence values and constructs the test sets accordingly; [24] simulates label shift by drawing the test set label distributions from a Dirichlet distribution. However, these stratified sampling strategies change the underlying data distribution and are problematic in assessing the actual performances of quantification models in practice.

In this paper, we introduce the first text quantification benchmark with naturally occurred temporal distribution shift. We focus on temporal distribution shift to tackle a main use case for text quantifiers: monitoring future label prevalence changes given historic labeled data. Each dataset is split into subsets containing samples from the same month or year. The subsets are then grouped into training and testing according to a specified point in time. Due to the long time span, the input distribution for each class might change. For example, Figure 5.2 shows that the "Money transfer, virtual currency" category in one of the datasets has drastically different input composition in early-2017 and mid-2020, which presents significant challenges to a candidate quantifier.

A total of eight datasets are included in the benchmark spanning sentiment analysis, document categorization, and toxicity classification. We evaluate different quantification algorithms on the benchmark and find that no algorithm consistently outperforms others. The main contributions of this work are threefold:

- We create the first benchmark for text quantification learning with temporal distribution shift consisting of diverse tasks and domains to evaluate model performances in a realistic setting.
- We propose a new metric, Class-Averaged Rank Correlation (CARC), for quantification learning that measures models' ability to produce prevalence estimates that are consistent with ground-truth values in terms of ranking order.
- We evaluate various baseline algorithms on the benchmark and find that no algorithm consistently outperforms others, strongly motivating future research in this area.

5.2 Related Work

Quantification Learning. Many of the experiments reported in quantification learning literature employ datasets taken from other classification problems. For example,

[86] uses 11 sentiment classification datasets and average the performances of studied methods across all 11 datasets. The problem is that only one test set is available for each dataset. Four text classification datasets used in [87] all have a balanced training set, and artificially created test splits similar to [84]. The image datasets created in [88] are more adequate with 21 and 15 test splits under various distribution shift. [89] employs RCV1-v2, a multi-label text classification benchmark with 52 weeks of data for testing. These datasets lack either the number of test splits or the diversity of the task domains. In contrast, our benchmark comprises more diverse tasks and domains; it involves temporal distribution shift across a long period of time; it provides monthly/yearly splits that allow more fine-grained analysis.

Learning under Distribution Shift. There has been an increasing interest in studying the challenges arising from data distribution changes in the machine learning community [20, 21, 22, 23]. The focus of these studies is mainly on adapting to covariate shift and improving the performance of the underlying learner in a shifted domain. [24] makes use of a quantifier to estimate and adjust for the label prevalence changes in the target domain, assuming the feature distribution of each class stays the same. [25] shows that quantification helps in adversarial domain adaption by providing an estimation of label prevalence ratios as importance weights.

Direct utilization of domain adaptation datasets for text quantification is undesirable. Firstly, the very few datasets that involve temporal distribution shift are mostly vision datasets [90, 91]. Secondly, quantification learning requires test splits to reflect the actual ground truth prevalence for each class which is not a necessity for domain adaptation datasets.

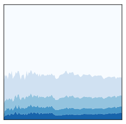
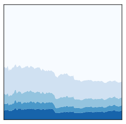
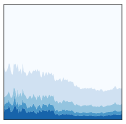
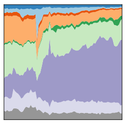
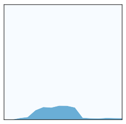
Dataset	Amazon Reviews			CCD	Wikipedia Toxicity
	Clothing	Electronics	Office		
training splits	Aug 2008 - Jul 2015			Apr 2015 - Jul 2019	2001 - 2010
test splits	Aug 2015 - Jul 2018			Aug 2019 - Jul 2021	2011 - 2015
# classes	5 or 2			9	2
# training	3,749,569	3,283,304	316,302	434,482	109,277
# test	7,453,848	3,285,326	476,785	345,914	25,809
label distribution					

Table 5.1: The text quantification benchmark contains 8 datasets (including three binary versions of the Amazon Reviews datasets) across sentiment analysis, document categorization, and toxicity classification tasks. Each dataset comprises data from a long period of time, and the benchmark is set up to evaluate models' ability to accurately estimate future test split label distributions under naturally occurred distribution shift.

5.3 The Text Quantification Benchmark

Problem Formulation

Given a labeled set of examples $D_s = \{(x_1, y_1), \ldots, (x_n, y_n)\}$ where $x_i \in \mathcal{X}$ and $y_i \in \mathcal{Y} = \{l_1, \ldots, l_k\}$, denote $\mathcal{P}(\mathcal{X})$ as the powerset of $\mathcal{X}$, and Δ^d as the standard d-simplex, the task is to induce a quantifier $h : \mathcal{P}(\mathcal{X}) \backslash \emptyset \to \Delta^{k-1}$ from the training data. For a test set $X_t = \{x'_1, \ldots, x'_m\}$, $h(X_t)$ produces a categorical distribution $\hat{\boldsymbol{p}}$ where each element in the predicted vector $\hat{p}_j$ represents the proportion of label l_j in the set of input examples. The goal is to predict $\hat{\boldsymbol{p}}$ that is as close as possible to the ground truth label distribution $\boldsymbol{p}$.

Dataset Identification and Preparation

There are several considerations when we identify potential dataset candidates for the benchmark:

- The dataset must have instance-level time information to construct test splits based on the time each example was produced.

- There should be no label-based biased sampling in any test split so that the actual underlying label distributions are available to be compared.
- The dataset ideally should span a long period of time so that there is enough label distribution variation in the test splits.
- The benchmark should cover both multi-class and binary classification problems in multiple text domains with various training data sizes.

There are in total eight datasets being included in the benchmark, an overview of the dataset statistics is shown in Table 5.1. Note that there are more test examples than training examples in Amazon Reviews datasets due to the fact that more recent splits contain more examples than older ones.

Amazon Review Data [92] collected product reviews from Amazon in the range of May 1996 to October 2018. We use the "5-core" subsets of three categories: Clothing Shoes and Jewelry, Electronics, and Office Products to account for different domains and data sizes. There is a steady trend towards a higher percentage of higher review ratings over time, making these datasets suitable for quantification.

For all three categories, reviews made between 2008-08 and 2015-07 are used for training, and reviews from 2015-08 to 2018-07 are split by month and used for testing. The original review ratings are on a scale of five stars. We create binary versions of each category by changing the task to predict the percentage of negative reviews, i.e., reviews with 1 star and 2 stars.

Consumer Complaint Database (CCD) is a collection of complaints about consumer financial products and services that the Consumer Financial Protection Bureau sent to companies for response[1]. All complaints can be classified into nine categories based on product types. Due to the evolution of the financial market, the complaint category distribution changes over time. For example, the percentage of credit reporting-related complaints increased from around 16% in 2017-01 to 57% in 2021-07.

We take all the records between 2015-04 and 2021-07 and filter out those without text content. Complaints filed before 2019-07 are used for training, and the remaining data are grouped by month as test splits. There are 434,482 training examples and 345,914 test examples. Test split size ranges from 5,127 to 18,495.

Wikipedia Talk: Toxicity [93] extracted discussion comments from English Wikipedia. Multiple annotators labeled each comment via Crowdflower on whether it is a toxic or healthy contribution. The original data was collected using two sampling types: *random* and *blocked*. The *random* dataset contains randomly selected comments; therefore, it can be used to evaluate the actual toxicity prevalences over time. The *blocked* dataset is used to ensure a sufficient number of toxic comments for training purposes. We use *blocked* and *random* examples from 2001 to 2010 as the training set, *random* examples from 2011 to 2015 as the test splits. By construction, the underlying distribution changes from training to test significantly. As the up-sampling strategy for imbalanced classification is ubiquitous in practice, this dataset is perfect for evaluating quantification methods with classifiers trained on re-sampled data.

Distribution Shift Analysis

We measure the distribution shift of each test split compared with the corresponding training set in terms of both covariate shift and label shift.

---

[1] `https://www.consumerfinance.gov/data-research/consumer-complaints/`

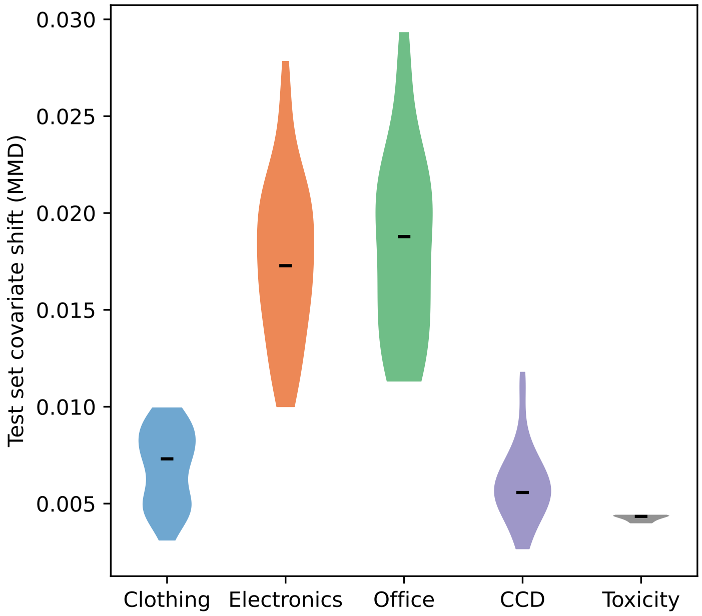


Figure 5.3: Distributions of the test split covariate shift measured with MMD in the BERT embedding space. Wikipedia Toxicity has milder covariate shift compared to others.

Covariate Shift. We capture the covariate shift of the input distribution $p(x)$ by evaluating the Maximum Mean Discrepancy (MMD) [94] on the BERT [70] encodings of the input examples. MMD allows us to compare two probability distributions in a reproducing kernel Hilbert space based on their samples. We use MMD with a radial basis function (RBF) kernel and set σ to be the median distance between points in the training set [94].

$$k(\boldsymbol{x}, \boldsymbol{x}') = e^{-\frac{\|\boldsymbol{x}-\boldsymbol{x}'\|^2}{2\sigma^2}} \tag{5.1}$$

A larger value measured by MMD indicates a larger discrepancy between the training and testing input embeddings.

We sample 10,000 examples from each test split and measure the MMD for all test

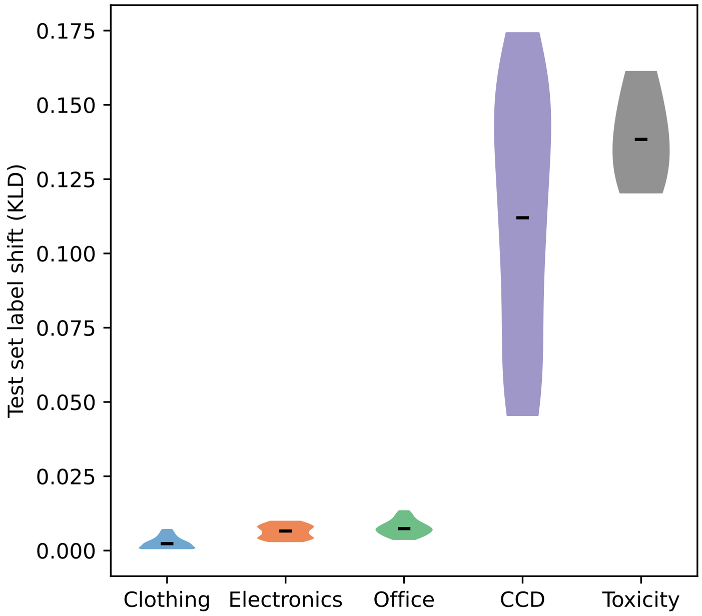


Figure 5.4: Distributions of the test split label shift measured with KLD. CCD and Wikipedia Toxicity have a higher average label shift as well as larger variations among the test splits.

splits. The distribution of the test split covariate shifts for each dataset is shown in Figure 5.3. All datasets have varying levels of covariate shift in their test splits except for Wikipedia Toxicity. It is still interesting to see how well models trained with upsampling estimate the true unbalanced label distribution.

Label Shift. We use Kullback–Leibler divergence (KLD) to measure the difference between the training and the test split label distributions. Ideally, the label shift of the test splits should cover a range of values to better evaluate candidate methods under various scenarios. The distributions of label shift values in the test splits are plotted in Figure 5.4. CCD and Wikipedia Toxicity have higher variations in the degrees of label shift from the training set than Amazon Review datasets. There are still reasonable label shift variations in the Amazon Review data as shown in Table 5.1.

Evaluation

We use two commonly reported quantification metrics for performance evaluation: Relative Absolute Error (RAE) and Knullback-Leibler Divergence (KLD). We propose a new metric, namely Class-Average Rank Correlation (CARC), to measure the ability to rank test splits by class label prevalences correctly.

Relative Absolute Error. RAE measures the relation between the absolute error and the ground truth label distribution. Adopting the same notations in Section 5.3, RAE is defined as follows:

$$\mathrm{RAE}(\boldsymbol{p}, \hat{\boldsymbol{p}}) = \frac{1}{k} \sum_{i=1}^{k} \frac{|\hat{\boldsymbol{p}}_i - \boldsymbol{p}_i|}{\boldsymbol{p}_i} \tag{5.2}$$

Intuitively, RAE measures the average percentage difference from an estimated class prevalence to the ground truth. The lower the better.

Knullback-Leibler Divergence. KLD is a popular metric for measuring the difference between two distributions.

$$\mathrm{KLD}(\boldsymbol{p}, \hat{\boldsymbol{p}}) = \sum_{i=1}^{k} \boldsymbol{p}_i \log \frac{\boldsymbol{p}_i}{\hat{\boldsymbol{p}}_i} \tag{5.3}$$

A benefit of using KLD is that it is widely adopted in the machine learning community and quantification literature. However, it is less interpretable than RAE and can be undefined when $\hat{\boldsymbol{p}}_i = 0$. As RAE and KLD values are closely correlated, reporting both values is redundant in most cases. Therefore, we only report RAE in our experiments.

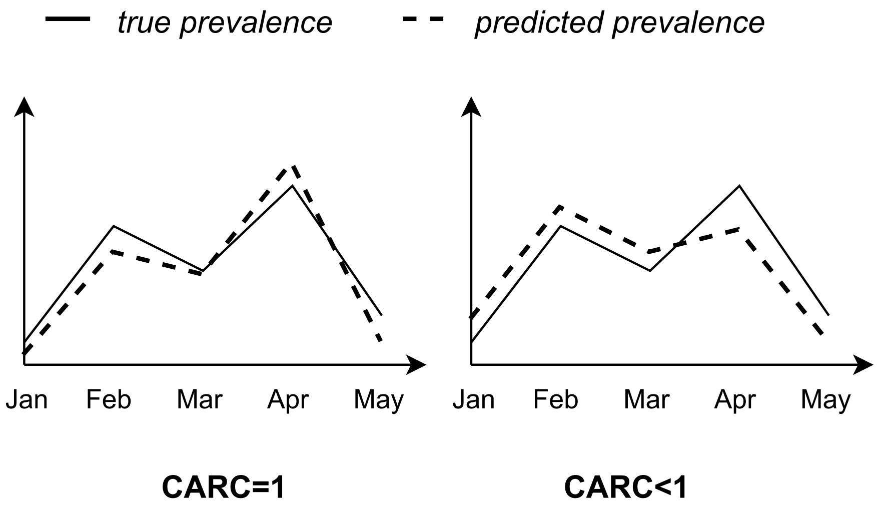


Figure 5.5: An illustration of the CARC metric. A perfect CARC score of 1 indicates that the model successfully predicts which split has a higher prevalence for any pair of test splits.

Class-Averaged Rank Correlation. While RAE and KLD capture the difference between the predicted label distribution and the ground truth for each test split, they do not account for the ranking information of the predicted label prevalence. In practice, correct ranking of the predicted prevalence is sometimes more important than capturing the exact label prevalence. Therefore, we propose a new metric for quantification named Class-Averaged Rank Correlation (CARC).

We measure the rank correlation between the predicted prevalence and the ground truth prevalence for each class label across all test splits. CARC is then defined as the average rank correlation value across all classes. Formally, let $\boldsymbol{P} = [\boldsymbol{p}^{(1)}, \dots, \boldsymbol{p}^{(t)}]$ denote the list of ground truth label distributions for the t test splits. $\hat{\boldsymbol{P}}$ represents the corresponding list of predictions $[\hat{\boldsymbol{p}}^{(1)}, \dots, \hat{\boldsymbol{p}}^{(t)}]$. Denote $\boldsymbol{P}_i$ as $[\boldsymbol{p}_i^{(1)}, \dots, \boldsymbol{p}_i^{(t)}]$, i.e., the list of true prevalences for class i in all test splits. CARC is defined as:

$$\mathrm{CARC}(\boldsymbol{P}, \hat{\boldsymbol{P}}) = \frac{1}{k} \sum_{i=1}^{k} \rho_{R(\boldsymbol{P}_i), R(\hat{\boldsymbol{P}}_i)} \tag{5.4}$$

where ρ is the correlation coefficient applied to the rank variables $R(\boldsymbol{P}_i)$ and $R(\hat{\boldsymbol{P}}_i)$.

With a higher value of CARC, if test split A has a higher predicted prevalence than test split B, the actual prevalence in A is more likely to be higher than that in B. CARC is a critical metric because if a quantifier indicates a prevalence increase, the ground truth prevalence should ideally indeed be higher. Otherwise, quantification results cannot be relied on for relevant decision making.

5.4 Baseline Algorithms

In addition to the straightforward Classify & Count (CC) algorithm, we include several methods from prior work on label shift estimation where predictions from a black box classifier can be used as inputs.

Classify & Count (CC) [84]. Given classification results from any trained classifier, CC uses the aggregated distribution to predict the test set label distribution. Probabilistic Classify & Count (PCC) is a variant that aggregates the predicted probabilities instead of class assignments.

Black Box Shift Estimation (BBSE) [24]. By making a label shift assumption that the conditional distribution of $p(x|y)$ remains the same across training and testing, BBSE uses the confusion matrix to adjust the predicted label distribution from CC. BBSE is proven to be consistent and error bounded even with biased black box predictors as long as the confusion matrix is invertible, and the label shift assumption holds. BBSE, when used for quantification, is equivalent to the Adjusted Count method [84, 95] in multi-class settings.

Regularized Learning under Label Shift (RLLS) [96]. To avoid arbitrarily bad estimation of the confusion matrix due to limited data size, RLLS makes the final distribution prediction less sensitive to the estimation performance of the confusion matrix by regularizing the ratio of test and training label distributions. RLLS is primarily designed to improve classification performance under label shift. The label distribution estimate is often a compromise between the BBSE result and the training distribution.

Maximum Likelihood Label Shift (MLLS) [83]. Like BBSE, MLLS also takes a distribution matching approach to estimate the test set label distribution. The original algorithm uses an EM-based strategy [97] to perform distribution matching in the input space of the test set. [83] show that in combination with a particular post-hoc calibration method, MLLS outperforms BBSE.

5.5 Experiments and Results

Experimental Setup

We use the huggingface [98] implementation of the BERT [70] classifier fine-tuned on the corresponding dataset as the base predictor for all algorithms. The predictor is further calibrated using bias-corrected temperature scaling (BCTS) [83] for the MLLS method. All models are trained with AdamW optimizer [77] with a learning rate of 2e-5. All models are trained on a single Titan RTX GPU with a batch size of 32. Input sequence length is capped at 256.

Method	**Binary**				**Multi-Class**				**Average Rank**
	Clothing	Electronics	Office	Toxicity	Clothing	Electronics	Office	CCD	
(RAE)(%)↓									
CC	3.05	2.07	3.93	**2.43**	9.87	12.61	11.02	9.20	3.63
PCC	2.63	2.44	**3.43**	32.79	6.74	7.14	**8.65**	12.41	3.13
BBSE	2.19	**1.90**	4.61	50.68	**5.60**	8.89	19.00	8.71	3.50
RLLS	2.51	3.22	7.37	49.47	14.10	29.92	28.23	8.56	5.25
MLLS	1.73	2.06	4.15	29.46	7.02	8.95	11.96	**6.13**	**2.75**
MLLS-BCTS	**1.26**	2.94	4.07	30.57	7.63	**7.13**	11.77	7.40	**2.75**
(CARC)↑									
CC	0.983	**0.994**	0.958	**1.000**	0.678	**0.784**	0.706	0.895	3.50
PCC	0.985	0.993	0.966	0.829	0.665	0.710	**0.746**	0.898	2.63
BBSE	0.985	0.993	0.965	nan*	**0.699**	0.695	0.616	0.892	4.13
RLLS	0.985	0.993	0.966	0.257	0.697	0.683	0.631	0.897	3.63
MLLS	0.985	0.993	**0.967**	0.314	0.685	0.721	0.733	**0.900**	**2.13**
MLLS-BCTS	**0.986**	0.993	**0.967**	0.314	0.663	0.698	0.733	0.896	3.00

Table 5.2: Quantification model performances in terms of average RAE (lower is better) and CARC (higher is better). MLLS-BCTS denotes MLLS with BCTS calibrated base predictor. Overall, MLLS performs the best, but not consistently outperforming others. **BBSE fails to produce non-zero prevalence estimates on all test sets, leading to an undefined CARC score.*

Main Results

We measure the RAE and CARC scores for all methods on the benchmark. RAE scores are averaged over test splits for each dataset. We rank the performances with respect to RAE and CARC on each dataset and report the average ranking for each algorithm. The results are summarized in Table 5.2. Some main observations from the table are:

- No algorithm outperforms others on all datasets.
- CC and PCC are still strong baselines. PCC performs better in most cases, but CC achieves significantly better results on Wikipedia Toxicity, where the positive class is rare, and the label shift is severe.
- The performance of the MLLS algorithm, with or without BCTS calibration, is more consistent across all datasets than other algorithms.

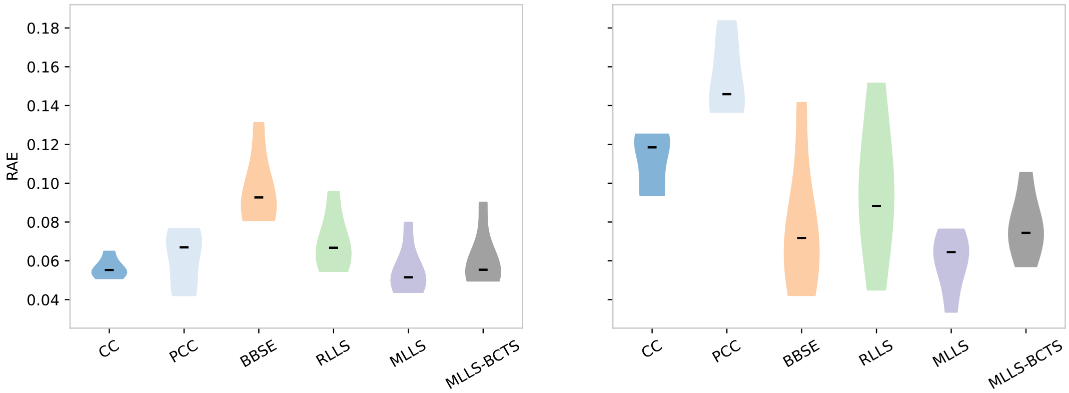


Figure 5.6: Distribution of RAE scores for test splits with the lowest 20% label shift (left) and with the highest 20% label shift (right) on CCD. CC and PCC's performances degrade significantly on test sets with a higher level of label shift.

- MLLS with BCTS calibration does not always have superior performance than the base version MLLS, contrary to what has been observed in previous studies [83].
- A better RAE score does not always indicate a better CARC score. For example, CC achieves a significantly better CARC score than other methods with a second-worst RAE score of 12.61% on Amazon Reviews (Electronics).
- BBSE fails to produce a non-zero prevalence estimate on all Wikipedia Toxicity test sets. This failure hints that BBSE might be unstable when predicting the prevalence of rare binary events.

Effect of Distribution Shift

In Section 5.3, we analyze the distribution shift estimates for each dataset across all the test splits. A natural question to ask is: how model performances change when the level of distribution shift increases? We sort the CCD test splits by label shift levels measured in KLD. We then take the bottom 20% and top 20% and visualize the RAE score distributions for all baseline algorithms in Figure 5.6.

Method	**Standard**	**Balanced**	**% Change**
(CCD)			
CC	9.20	19.44	+111.3%
PCC	12.41	19.77	+59.3%
BBSE	8.71	9.63	+10.6%
RLLS	8.56	13.84	+61.7%
MLLS	6.13	16.80	+174.1%
MLLS-BCTS	7.40	15.94	+115.4%
(Office)			
CC	11.02	14.09	+27.9%
PCC	8.65	19.94	+130.5%
BBSE	19.00	19.97	+5.1%
RLLS	28.23	27.34	-3.2%
MLLS	11.96	18.28	+52.8%
MLLS-BCTS	11.77	14.71	+25.0%

Table 5.3: Comparison of quantification performances in RAE (lower is better) using base classifiers trained with standard and balanced training set. Using a balanced training strategy almost always hurts quantification performance on CCD and Amazon Reviews (Office). BBSE is more robust to label distribution changes from stratified sampling during training.

We observe a significant performance degradation of CC and PCC methods on test splits with higher levels of label shift. MLLS and MLLS-BCTS are less affected by the label shift. The difference is expected because the underlying base predictor is likely to overestimate or underestimate the label probabilities when the test split has a significantly different label distribution.

Effect of Balanced Training

In practice, when the training data is highly skewed in terms of label distribution, we often manually up-sample the rare class examples or assign more weights to them to facilitate training. This procedure changes the underlying data distribution and could significantly impact the quantification results if we use the classifier as our base predictor.

To analyze the effect of a balanced training procedure on the quantification performance, we fine-tune the same BERT classifier on both CCD and Amazon Reviews (Office) with a weighted random sampler so that all class examples are balanced. We then use this classifier as the base predictor for all baseline algorithms and compare the performances to the main results in Table 5.3.

When using a base predictor trained with a manually balanced dataset, the quantification performance almost always degrades. However, we can see from the percentage changes that BBSE is more robust to such performance degradation than other methods. For example, on CCD, BBSE is outperformed by MLLS when using a base classifier trained on the original training set. When switching to a balanced training setup, BBSE maintains a similar level of performance and betters MLLS. This property makes BBSE more preferable when label balancing is present during training.

Effect of Invariant Representation Learning

BBSE, RLLS, and MLLS all make a label shift assumption where the conditional distribution of $p(x|y)$ remains the same across training and test. However, this assumption does not always hold in practice. The content of a 1-star review on a product posted five years ago could be significantly different from a 1-star review posted today due to many factors, such as a change of consumer expectations in similar products.

To relax the label shift assumption, [25] propose to learn a domain-invariant representation and use a similar approach to BBSE to estimate the test set label distribution by performing distribution matching in the invariant latent space. Supposedly, such methods should perform better on test splits where the conditional distribution of the input features for each class drifts heavily from the training set. A significant drawback of the method is that the underlying model needs to be retrained for each test split.

We experiment with IWDAN model [25] on both CCD and Wikipedia Toxicity datasets.

On CCD, IWDAN shows a much worse RAE score of 49.18%. On Wikipedia Toxicity, however, IWDAN achieves an RAE score of 19.40%, the second-best result after CC. As the training and testing splits of Wikipedia Toxicity come from different sampling strategies, and considering IWDAN is devised mainly for domain adaption, the performance discrepancy might be due to a more significant domain change in Wikipedia Toxicity compared to CCD.

5.6 Conclusions

Quantification learning has an increasing number of applications yet is still less studied in the NLP community. In this chapter, we propose the first text quantification benchmark with temporal distribution shift. Our experiments show that there is no baseline algorithm consistently outperforming others. We believe the proposed benchmark should enable new research into devising methods that can adapt to temporal changes and be reliably applied in practice.

Chapter 6

Conclusion

In this dissertation, we discussed and analyzed various topics around uncertainty and robustness in deep learning for NLP. We demonstrated the benefits of incorporating uncertainty estimation in the modeling process for sentiment analysis, named entity recognition, and language modeling; we investigated the relationship between predictive uncertainty and hallucination in image captioning and data-to-text generation; we discussed the role label smoothing plays with respect to model calibration in text classification; and we introduced a text quantification benchmark to evaluate methods on their abilities to estimate label distribution under temporal distribution shift. These analyses help to better understand uncertainty aware learning and robustness in NLP.

The goal of research in uncertainty and robustness is to demonstrate the benefits and promote the importance of model transparency, interpretability, and accountability. Several of future research directions in this line are:

- Development of standard evaluation methods for uncertainty estimation. Current evaluation of uncertainty estimation methods depend on external tasks such as model calibration and out-of-distribution detection. Sample level evaluations are unavailable, meaning it is hard to measure whether the uncertainty estimates on some specific inputs are accurate or not. A direct and sample-level evaluation protocol would greatly benefit the study of uncertainty estimation.

- Uncertainty estimation for pre-trained models. Many of the uncertainty estimation methods require retraining or significant modification to the base model. With the popularization of large pre-trained models, how to effectively and reliably estimate uncertainty without access to the training data or modification to the pre-trained model remains an important open problem.
- Explainable uncertainty estimation. In addition to a number describing the level of uncertainty of a model prediction, it is equally important to understand why a model is confident or uncertain when doing such predictions. This capability is especially essential in understanding how uncertainty estimation methods operate to identify possible paths for refinement and detect failures.

Bibliography

[1] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, *Attention is all you need*, in *Advances in neural information processing systems*, pp. 5998–6008, 2017.

[2] G. Ras, M. van Gerven, and P. Haselager, *Explanation methods in deep learning: Users, values, concerns and challenges*, in *Explainable and interpretable models in computer vision and machine learning*, pp. 19–36. Springer, 2018.

[3] C. Guo, G. Pleiss, Y. Sun, and K. Q. Weinberger, *On calibration of modern neural networks*, in *International Conference on Machine Learning*, pp. 1321–1330, PMLR, 2017.

[4] Y. Ovadia, E. Fertig, J. Ren, Z. Nado, D. Sculley, S. Nowozin, J. Dillon, B. Lakshminarayanan, and J. Snoek, *Can you trust your model's uncertainty? evaluating predictive uncertainty under dataset shift*, *Advances in neural information processing systems* **32** (2019).

[5] N. Houlsby, F. Huszár, Z. Ghahramani, and M. Lengyel, *Bayesian active learning for classification and preference learning*, *arXiv preprint arXiv:1112.5745* (2011).

[6] T. M. Moerland, J. Broekens, and C. M. Jonker, *Efficient exploration with double uncertain value networks*, *arXiv preprint arXiv:1711.10789* (2017).

[7] W. R. Clements, B. Van Delft, B.-M. Robaglia, R. B. Slaoui, and S. Toth, *Estimating risk and uncertainty in deep reinforcement learning*, *arXiv preprint arXiv:1905.09638* (2019).

[8] A. Der Kiureghian and O. Ditlevsen, *Aleatory or epistemic? does it matter?*, *Structural Safety* **31** (2009), no. 2 105–112.

[9] A. Kendall and Y. Gal, *What uncertainties do we need in bayesian deep learning for computer vision?*, in *Advances in neural information processing systems*, pp. 5574–5584, 2017.

[10] C. Blundell, J. Cornebise, K. Kavukcuoglu, and D. Wierstra, *Weight uncertainty in neural networks*, *arXiv preprint arXiv:1505.05424* (2015).

[11] J. M. Hernández-Lobato and R. Adams, *Probabilistic backpropagation for scalable learning of bayesian neural networks*, in *International Conference on Machine Learning*, pp. 1861–1869, 2015.

[12] Y. Gal and Z. Ghahramani, *Dropout as a bayesian approximation: Representing model uncertainty in deep learning*, in *international conference on machine learning*, pp. 1050–1059, 2016.

[13] B. Lakshminarayanan, A. Pritzel, and C. Blundell, *Simple and scalable predictive uncertainty estimation using deep ensembles*, in *Advances in neural information processing systems*, pp. 6402–6413, 2017.

[14] B. Charpentier, D. Zügner, and S. Günnemann, *Posterior network: Uncertainty estimation without ood samples via density-based pseudo-counts*, *Advances in Neural Information Processing Systems* **33** (2020) 1356–1367.

[15] A. Kendall, V. Badrinarayanan, and R. Cipolla, *Bayesian segnet: Model uncertainty in deep convolutional encoder-decoder architectures for scene understanding*, *arXiv preprint arXiv:1511.02680* (2015).

[16] Y. Gal and Z. Ghahramani, *A theoretically grounded application of dropout in recurrent neural networks*, in *Advances in neural information processing systems*, pp. 1019–1027, 2016.

[17] L. Zhu and N. Laptev, *Deep and confident prediction for time series at uber*, in *Data Mining Workshops (ICDMW), 2017 IEEE International Conference on*, pp. 103–110, IEEE, 2017.

[18] J. Platt *et. al.*, *Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods*, *Advances in large margin classifiers* **10** (1999), no. 3 61–74.

[19] R. Müller, S. Kornblith, and G. Hinton, *When does label smoothing help?*, in *NeurIPS*, 2019.

[20] H. Daumé III, *Frustratingly easy domain adaptation*, in *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, pp. 256–263, 2007.

[21] J. Blitzer, M. Dredze, and F. Pereira, *Biographies, bollywood, boom-boxes and blenders: Domain adaptation for sentiment classification*, in *Proceedings of the 45th annual meeting of the association of computational linguistics*, pp. 440–447, 2007.

[22] X. Glorot, A. Bordes, and Y. Bengio, *Domain adaptation for large-scale sentiment classification: A deep learning approach*, in *ICML*, 2011.

[23] Y. Ganin, E. Ustinova, H. Ajakan, P. Germain, H. Larochelle, F. Laviolette, M. Marchand, and V. Lempitsky, *Domain-adversarial training of neural networks*, *The journal of machine learning research* **17** (2016), no. 1 2096–2030.

[24] Z. Lipton, Y.-X. Wang, and A. Smola, *Detecting and correcting for label shift with black box predictors*, in *International conference on machine learning*, pp. 3122–3130, PMLR, 2018.

[25] R. Tachet des Combes, H. Zhao, Y.-X. Wang, and G. J. Gordon, *Domain adaptation with conditional distribution matching and generalized label shift*, *Advances in Neural Information Processing Systems* **33** (2020).

[26] W. L. Buntine and A. S. Weigend, *Bayesian back-propagation*, *Complex systems* **5** (1991), no. 6 603–643.

[27] J. S. Denker and Y. Lecun, *Transforming neural-net output levels to probability distributions*, in *Advances in neural information processing systems*, pp. 853–859, 1991.

[28] D. J. MacKay, *A practical bayesian framework for backpropagation networks*, *Neural computation* **4** (1992), no. 3 448–472.

[29] D. J. MacKay, *Probable networks and plausible predictions—a review of practical bayesian methods for supervised neural networks*, *Network: Computation in Neural Systems* **6** (1995), no. 3 469–505.

[30] R. M. Neal, *Bayesian learning for neural networks*, vol. 118. Springer Science & Business Media, 2012.

[31] A. Graves, *Practical variational inference for neural networks*, in *Advances in neural information processing systems*, pp. 2348–2356, 2011.

[32] D. A. Nix and A. S. Weigend, *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*, vol. 1, pp. 55–60, IEEE, 1994.

[33] Q. V. Le, A. J. Smola, and S. Canu, *Heteroscedastic gaussian process regression*, in *Proceedings of the 22nd international conference on Machine learning*, pp. 489–496, ACM, 2005.

[34] D. Tang, B. Qin, and T. Liu, *Document modeling with gated recurrent neural network for sentiment classification*, in *Proceedings of the 2015 conference on empirical methods in natural language processing*, pp. 1422–1432, 2015.

[35] Q. Diao, M. Qiu, C.-Y. Wu, A. J. Smola, J. Jiang, and C. Wang, *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*, pp. 193–202, ACM, 2014.

[36] Y. Kim, *Convolutional neural networks for sentence classification*, *arXiv preprint arXiv:1408.5882* (2014).

[37] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, *Dropout: a simple way to prevent neural networks from overfitting*, *The Journal of Machine Learning Research* **15** (2014), no. 1 1929–1958.

[38] D. P. Kingma and J. Ba, *Adam: A method for stochastic optimization*, *arXiv preprint arXiv:1412.6980* (2014).

[39] S. Hochreiter and J. Schmidhuber, *Long short-term memory*, *Neural computation* **9** (1997), no. 8 1735–1780.

[40] E. F. Tjong Kim Sang and F. De Meulder, *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*, pp. 142–147, Association for Computational Linguistics, 2003.

[41] W. Zaremba, I. Sutskever, and O. Vinyals, *Recurrent neural network regularization*, *arXiv preprint arXiv:1409.2329* (2014).

[42] A. Rohrbach, L. A. Hendricks, K. Burns, T. Darrell, and K. Saenko, *Object hallucination in image captioning*, in *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pp. 4035–4045, 2018.

[43] S. Wiseman, S. M. Shieber, and A. M. Rush, *Challenges in data-to-document generation*, in *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pp. 2253–2263, 2017.

[44] F. Nie, J.-G. Yao, J. Wang, R. Pan, and C.-Y. Lin, *A simple recipe towards reducing hallucination in neural surface realisation*, in *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pp. 2673–2679, 2019.

[45] A. P. Parikh, X. Wang, S. Gehrmann, M. Faruqui, B. Dhingra, D. Yang, and D. Das, *Totto: A controlled table-to-text generation dataset*, *arXiv preprint arXiv:2004.14373* (2020).

[46] Z. Cao, F. Wei, W. Li, and S. Li, *Faithful to the original: Fact aware neural abstractive summarization*, in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 32, 2018.

[47] E. Durmus, H. He, and M. Diab, *FEQA: A question answering evaluation framework for faithfulness assessment in abstractive summarization*, in *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, (Online), pp. 5055–5070, Association for Computational Linguistics, July, 2020.

[48] M. Müller, A. Rios, and R. Sennrich, *Domain robustness in neural machine translation*, *arXiv preprint arXiv:1911.03109* (2019).

[49] G. E. Hinton and D. Van Camp, *Keeping the neural networks simple by minimizing the description length of the weights*, in *Proceedings of the sixth annual conference on Computational learning theory*, pp. 5–13, 1993.

[50] R. M. Neal, *BAYESIAN LEARNING FOR NEURAL NETWORKS.* PhD thesis, University of Toronto, 1995.

[51] S. Depeweg, J.-M. Hernandez-Lobato, F. Doshi-Velez, and S. Udluft, *Decomposition of uncertainty in bayesian deep learning for efficient and risk-sensitive learning*, in *International Conference on Machine Learning*, pp. 1184–1193, PMLR, 2018.

[52] X. Chen, H. Fang, T.-Y. Lin, R. Vedantam, S. Gupta, P. Dollár, and C. L. Zitnick, *Microsoft coco captions: Data collection and evaluation server*, *arXiv preprint arXiv:1504.00325* (2015).

[53] S. J. Rennie, E. Marcheret, Y. Mroueh, J. Ross, and V. Goel, *Self-critical sequence training for image captioning*, in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 7008–7024, 2017.

[54] P. Anderson, X. He, C. Buehler, D. Teney, M. Johnson, S. Gould, and L. Zhang, *Bottom-up and top-down attention for image captioning and visual question answering*, in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 6077–6086, 2018.

[55] R. Luo, B. Price, S. Cohen, and G. Shakhnarovich, *Discriminability objective for training descriptive captions*, in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 6964–6974, 2018.

[56] A. Karpathy and L. Fei-Fei, *Deep visual-semantic alignments for generating image descriptions*, in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 3128–3137, 2015.

[57] K. Kukich, *Design of a knowledge-based report generator*, in *Proceedings of the 21st annual meeting on Association for Computational Linguistics*, pp. 145–150, Association for Computational Linguistics, 1983.

[58] K. McKeown, *Text generation.* Cambridge University Press, 1992.

[59] R. Tian, S. Narayan, T. Sellam, and A. P. Parikh, *Sticking to the facts: Confident decoding for faithful data-to-text generation*, *arXiv preprint arXiv:1910.08684* (2019).

[60] D. Bahdanau, K. Cho, and Y. Bengio, *Neural machine translation by jointly learning to align and translate*, in *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015.

[61] M. Ott, S. Edunov, A. Baevski, A. Fan, S. Gross, N. Ng, D. Grangier, and M. Auli, *fairseq: A fast, extensible toolkit for sequence modeling*, in *Proceedings of NAACL-HLT 2019: Demonstrations*, 2019.

[62] K. Papineni, S. Roukos, T. Ward, and W.-J. Zhu, *Bleu: a method for automatic evaluation of machine translation*, in *Proceedings of the 40th annual meeting on association for computational linguistics*, pp. 311–318, Association for Computational Linguistics, 2002.

[63] R. Vedantam, C. Lawrence Zitnick, and D. Parikh, *Cider: Consensus-based image description evaluation*, in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 4566–4575, 2015.

[64] Y. Xiao and W. Y. Wang, *Quantifying uncertainties in natural language processing tasks*, in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 33, pp. 7322–7329, 2019.

[65] M. Ott, M. Auli, D. Grangier, and M. Ranzato, *Analyzing uncertainty in neural machine translation*, in *International Conference on Machine Learning*, pp. 3956–3965, 2018.

[66] J. Xu, S. Desai, and G. Durrett, *Understanding neural abstractive summarization models via uncertainty*, in *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 6275–6281, 2020.

[67] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, *Rethinking the inception architecture for computer vision*, in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 2818–2826, 2016.

[68] Y. Gao, W. Wang, C. Herold, Z. Yang, and H. Ney, *Towards a better understanding of label smoothing in neural machine translation*, in *Proceedings of the 1st Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics and the 10th International Joint Conference on Natural Language Processing*, pp. 212–223, 2020.

[69] M. Lukasik, S. Bhojanapalli, A. Menon, and S. Kumar, *Does label smoothing mitigate label noise?*, in *International Conference on Machine Learning*, pp. 6448–6458, PMLR, 2020.

[70] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, *Bert: Pre-training of deep bidirectional transformers for language understanding*, in *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pp. 4171–4186, 2019.

[71] M. Iyyer, V. Manjunatha, J. Boyd-Graber, and H. Daumé III, *Deep unordered composition rivals syntactic methods for text classification*, in *Proceedings of the 53rd annual meeting of the association for computational linguistics and the 7th international joint conference on natural language processing (volume 1: Long papers)*, pp. 1681–1691, 2015.

[72] A. Niculescu-Mizil and R. Caruana, *Predicting good probabilities with supervised learning*, in *Proceedings of the 22nd international conference on Machine learning*, pp. 625–632, 2005.

[73] X. Zhang, J. Zhao, and Y. LeCun, *Character-level convolutional networks for text classification*, in *Proceedings of the 28th International Conference on Neural Information Processing Systems-Volume 1*, pp. 649–657, 2015.

[74] A. Warstadt, A. Singh, and S. R. Bowman, *Neural network acceptability judgments*, *arXiv preprint arXiv:1805.12471* (2018).

[75] R. Socher, A. Perelygin, J. Wu, J. Chuang, C. D. Manning, A. Y. Ng, and C. Potts, *Recursive deep models for semantic compositionality over a sentiment treebank*, in *Proceedings of the 2013 conference on empirical methods in natural language processing*, pp. 1631–1642, 2013.

[76] D. P. Kingma and J. Ba, *Adam: A method for stochastic optimization*, in *International Conference on Learning Representations*, 2015.

[77] I. Loshchilov and F. Hutter, *Decoupled weight decay regularization*, in *International Conference on Learning Representations*, 2018.

[78] H. Zhang, M. Cisse, Y. N. Dauphin, and D. Lopez-Paz, *mixup: Beyond empirical risk minimization*, in *International Conference on Learning Representations*, 2018.

[79] W. Warner and J. Hirschberg, *Detecting hate speech on the world wide web*, in *Proceedings of the second workshop on language in social media*, pp. 19–26, 2012.

[80] S. Malmasi and M. Zampieri, *Detecting hate speech in social media*, in *Proceedings of the International Conference Recent Advances in Natural Language Processing, RANLP 2017*, pp. 467–472, 2017.

[81] J. Qian, M. ElSherief, E. Belding, and W. Y. Wang, *Leveraging intra-user and inter-user representation learning for automated hate speech detection*, in *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, pp. 118–123, 2018.

[82] M. H. Stanfill, M. Williams, S. H. Fenton, R. A. Jenders, and W. R. Hersh, *A systematic literature review of automated clinical coding and classification systems*, *Journal of the American Medical Informatics Association* **17** (2010), no. 6 646–651.

[83] A. Alexandari, A. Kundaje, and A. Shrikumar, *Adapting to label shift with bias-corrected calibration*, *arXiv preprint arXiv:1901.06852* (2019).

[84] G. Forman, *Quantifying counts and costs via classification*, *Data Mining and Knowledge Discovery* **17** (2008), no. 2 164–206.

[85] P. González, A. Castaño, N. V. Chawla, and J. J. D. Coz, *A review on quantification learning*, *ACM Computing Surveys (CSUR)* **50** (2017), no. 5 1–40.

[86] W. Gao and F. Sebastiani, *From classification to quantification in tweet sentiment analysis*, *Social Network Analysis and Mining* **6** (2016), no. 1 19.

[87] L. Qi, M. Khaleel, W. Tavanapong, A. Sukul, and D. Peterson, *A framework for deep quantification learning*, in *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pp. 232–248, Springer, 2020.

[88] O. Beijbom, J. Hoffman, E. Yao, T. Darrell, A. Rodriguez-Ramirez, M. Gonzalez-Rivero, and O. H. Guldberg, *Quantification in-the-wild: data-sets and baselines*, *arXiv preprint arXiv:1510.04811* (2015).

[89] A. Esuli and F. Sebastiani, *Optimizing text quantifiers for multivariate loss functions*, *ACM Transactions on Knowledge Discovery from Data (TKDD)* **9** (2015), no. 4 1–27.

[90] G. Christie, N. Fendley, J. Wilson, and R. Mukherjee, *Functional map of the world*, in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 6172–6180, 2018.

[91] E. David, S. Madec, P. Sadeghi-Tehran, H. Aasen, B. Zheng, S. Liu, N. Kirchgessner, G. Ishikawa, K. Nagasawa, M. A. Badhon, *et. al.*, *Global wheat head detection (gwhd) dataset: a large and diverse dataset of high-resolution rgb-labelled images to develop and benchmark wheat head detection methods*, *Plant Phenomics* **2020** (2020).

[92] J. Ni, J. Li, and J. McAuley, *Justifying recommendations using distantly-labeled reviews and fine-grained aspects*, in *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pp. 188–197, 2019.

[93] E. Wulczyn, N. Thain, and L. Dixon, *Ex machina: Personal attacks seen at scale*, in *Proceedings of the 26th international conference on world wide web*, pp. 1391–1399, 2017.

[94] A. Gretton, K. M. Borgwardt, M. J. Rasch, B. Schölkopf, and A. Smola, *A kernel two-sample test*, *The Journal of Machine Learning Research* **13** (2012), no. 1 723–773.

[95] D. J. Hopkins and G. King, *A method of automated nonparametric content analysis for social science*, *American Journal of Political Science* **54** (2010), no. 1 229–247.

[96] K. Azizzadenesheli, A. Liu, F. Yang, and A. Anandkumar, *Regularized learning for domain adaptation under label shifts*, *arXiv preprint arXiv:1903.09734* (2019).

[97] M. Saerens, P. Latinne, and C. Decaestecker, *Adjusting the outputs of a classifier to new a priori probabilities: a simple procedure*, *Neural computation* **14** (2002), no. 1 21–41.

[98] T. Wolf, L. Debut, V. Sanh, J. Chaumond, C. Delangue, A. Moi, P. Cistac, T. Rault, R. Louf, M. Funtowicz, *et. al.*, *Transformers: State-of-the-art natural language processing*, in *Proceedings of the 2020 conference on empirical methods in natural language processing: system demonstrations*, pp. 38–45, 2020.